DETECTING DENIAL OF SERVICE ATTACKS IN NETWORK TRAFFIC WITH MAXIMUM ENTROPY AND HYPOTHESIS TESTING TECHNIQUES

by

Didem Çakırtaş

B.S., Electronics and Communication Engineering, Istanbul Technical University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2008

ACKNOWLEDGEMENTS

Firstly, I would like to thank to my thesis supervisor Prof. Dr. Emin Anarım for his kind interest, his guidance, his technical advices, encouragement, and patience.

Additionally, I would like to thank to Kıvanç Mıhçak for providing me many valuable and critical insights and ideas.

Moreover, I would like to thank to Kerem Harmancı for his kind interest to my questions and for discussions and also, many thanks to committee for their kindness of participation to my thesis.

This work is supported by the State Planning Organization of Turkey under the Next Generation Satellite Networks Project, DPT 03K 120250.

ABSTRACT

DETECTING DENIAL OF SERVICE ATTACKS IN NETWORK TRAFFIC BY USING MAXIMUM ENTROPY AND HYPOTHESIS TESTING TECHNIQUES

With the growth of computer networking and increased dependency of our every day life on the computer based systems, assuring reliable operation of computer systems has become very important. In order to render computer networks more secure, intrusion detection systems aim to recognise attacks. The objective of this work is to improve maximum entropy based intrusion detection methods and bring a formularization to ad hoc rules by using information theory and statistical signal processing.

In this work, it is intended to identify denial-of-service attacks by using maximum entropy and hypothesis testing methods. Proposed method consists of two phases: training and detection. In the training part, models are estimated for various attack types and no attack case based on the maximum entropy principle. In the detection part, hypothesis testing technique is employed to decide which of these models most probably satisfies the characteristics of the current network traffic. The method proposed in this thesis can be considered as a hybrid form of anomaly detection and misuse detection methods, since it focuses on not only the characteristics of normal network activity but also the characteristics of the known attacks. According to the experimental results, proposed method is very succesfull in identifying the denial-of-service attacks which have invariable characteristics and cause a dramatic change in network traffic. However, our method is inadequate for detecting denial-of-service attacks, which have variable characteristics and whose evidences are not noticeable from header information.

ÖZET

MAKSİMUM DAĞINTI VE HİPOTEZ TEST YÖNTEMİ İLE BİLGİSAYAR AĞLARINDAKİ HİZMET ENGELLEME SALDIRILARININ TESPİTİ

Bilgisayar ağlarındaki büyüme ve günlük yaşamımızın bilgisayar tabanlı sistemlere bağımlılığının artması ile birlikte, bilgisayar sistemlerinin güvenli birşekilde çalışmasını sağlamak büyük önem kazandı. Bilgisayar ağlarını daha güvenli hale getirmek için, saldırı tespit sistemleri (IDS), ağdaki saldırıları saptamayı hedeflemektedirler. Bu çalışmanın amacı maksimum dağıntı yaklaşımını temel alan saldırı tespit sistemlerinin iyileştirilmesi ve geçici kuralların enformasyon teorisini ve istatistik sinyal işleme yöntemlerini kullanarak formüleştirilmesidir.

Bu çalışmada hizmet engelleme saldırılarının (DoS) maksimum dağıntı ve hipotez test yöntemlerini kullanarak belirlenmesi amaçlanmaktadır. Önerilen yöntem eğitim ve saptama olarak iki safhadan oluşmaktadır. Eğitim kısmında, maksimum dağıntı yöntemi kullanılarak çeşitli saldırılar olması ve saldırı olmaması durumları için modeller kestirilmektedir. Saptama kısmında ise hipotez test yöntemi ile hangi modellin şuanki trafiğin özelliklerini daha büyük ihtimalle sağladığına karar verilmektedir. Önerilen yöntem hem normal ağ trafiğinin davranışına hem de bilinen saldırı tiplerinin davranışlarına odaklandığından, anomali saptama ve kötü kullanım saptama yöntemlerinin karışımı olarak düşünülebilir. Deneysel sonuçlar, önerilen yöntemin değişmeyen özelliklere sahip ve ağ trafiğinin çarpıcı bir şekilde değişimine sebep olan hizmet engelleme saldırılarının belirlenmesinde çok başarılı olduğunu göstermektedir. Fakat bu yöntem değişken özelliklere sahip ve kanıtları başlık bilgisinden farkedilebilir olmayan hizmet engelleme saldırılarının saptanmasında yetersiz kalmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

ÖZET v

LIST OF FIGURES x

LIST OF TABLES xiv

LIST OF SYMBOLS / ABBREVIATIONS xv

1. INTRODUCTION 1

1.1. Intrusion Detection Concept 1

1.2. Intrusion Detection Systems 2

1.3. Denial of Service Attacks 3

1.3.1. Apache2 Attack 4

1.3.2. Back Attack 5

1.3.3. Dosnuke Attack 5

1.3.4. Land Attack 5

1.3.5. Mailbomb Attack 6

1.3.6. Neptune (SYN-Flood) Attack 6

1.3.7. Ping of Dead Attack 7

1.3.8. Process Table Attack 7

1.3.9. Smurf Attack 8

1.3.10. Syslogd Attack 8

1.3.11. Tcpreset Attack 8

1.3.12. Teardrop Attack 9

1.3.13. Udpstorm Attack 9

1.4. Motivation 9
1.5. Thesis Outline 11
2. BACKGROUND 12
2.1. TCP and UDP Protocol Overview 12
2.2. Maximum Entropy Approach 14
2.3. Hypothesis Testing 15
3. METHODOLOGY 17
3.1. Introduction 17
3.2. Packet Classification 17
3.3. Maximum Entropy Estimation of the Packet Class Distribution 19
3.3.1. Feature Selection 21
3.3.2. Parameter Estimation 24
3.4. Model Induction 27
3.5. Detection of DoS Attacks with M-Hypothesis Testing 30
4. SIMULATION ENVIRONMENT 33
4.1. Introduction 33
4.2. Simulation Environment of DARPA 1999 Corpora 33
4.3. Simulation Data 35
4.3.1. Simulation Data Statistics 36
4.3.2. Target Attacks in Simulation Data 38
5. APPLICATIONS 42
5.1. Introduction 42
5.2. Maximum Entropy Based Modeling 43
5.2.1. Modeling Benign Traffic 44
5.2.2. Modeling Apache2 Attack 49
5.2.3. Modeling Back Attack 51

5.2.4. Modeling Dosnuke Attack 54
5.2.5. Modeling Neptune Attack 57
5.2.6. Modeling Udpstorm Attack 60
5.2.7. Modeling Tcpreset Attack 62
5.2.8. Modeling Mailbomb Attack 64
5.2.9. Modeling Syslogd Attack 67
5.3 Detection Part 70
5.3.1. Detection of Apache2 Attack 72
5.3.2. Detection of Back Attack 74
5.3.3. Detection of Dosnuke Attack 78
5.3.4. Detection of Neptune Attack 80
5.3.5. Detection of Udpstorm Attack 83
5.3.6. Detection of Tcpreset Attack 84
5.3.7. Detection of Mailbomb Attack 85
5.3.8. Detection of Syslogd Attack 87
5.4 Results 89
5.5. Performance Analysis 91
6. CONCLUSION AND FUTURE WORK 94
APPENDIX A: SOME DERIVATIONS 95
APPENDIX A.1. Derivation of Maximum Entropy Estimate 95
APPENDIX A.2. Derivation of the Gradient of the Log-Likelihood Function 96
APPENDIX B: FORMAT OF CD CONTAINING SIMULATION CODES 99
APPENDIX B.1. Simulation Data 99
APPENDIX B.2. Hardware and Software Requirements 99
APPENDIX B.3. MATLAB Codes 99
REFERENCES 101

REFERENCES NOT CITED 105

LIST OF FIGURES

Figure 2.1. TCP header [19] 13

Figure 2.2. UDP header [19] 13

Figure 2.3. Components of a decision problem [23] 16

Figure 3.1. The set of packet classes 19

Figure 3.2. Structure of a feature function 29

Figure 4.1. Network testbed used in 1999 [33] 34

Figure 4.2. Network topology used in 1999 [32] 35

Figure 4.3. TCP, UDP and ICMP packet percentage at week one on Monday between 08.00 am-10.00 am 37

Figure 4.4. Packet percentages of some TCP or UDP services at week one on Monday between 08.00 am-10.00 am 38

Figure 5.1. Block diagram of the proposed method 43

Figure 5.2. Empirical distribution of the packet classes calculated by using first 10000 packets in the training data 44

Figure 5.3. Empirical distribution of the packet classes calculated by using fifth set of 10000 packets in training data 45

Figure 5.4. Average empirical distribution 46

Figure 5.5. Structure of the top five feature functions in the feature matrix 47

Figure 5.6. Constructed model based on maximum entropy principle for the benign traffic 48

Figure 5.7. Estimated probability distribution for some of the unobserved packet classes 48

Figure 5.8. Average empirical distribution of packet classes for the training data of apache2 attack 50

Figure 5.9. Constructed model based on maximum entropy principle for apache2 attack 51

Figure 5.10. Average empirical distribution of packet classes for the training data of back attack 52

Figure 5.11. Constructed model based on maximum entropy principle for back attack 54

Figure 5.12. Empirical distribution of packet classes for the training data of dosnuke attack 55

Figure 5.13. PMF value of class 15 during dosnuke attack 55

Figure 5.14. Constructed model zoomed in the vicinity of class 15 57

Figure 5.15. Average empirical distribution of packet classes for the training data of neptune attack 58

Figure 5.16. Constructed model based on maximum entropy principle for neptune attack 59

Figure 5.17. Average empirical distribution of packet classes for the training data of udpstorm attack 60

Figure 5.18. Constructed model based on maximum entropy principle for udpstorm attack 61

Figure 5.19. Empirical distribution of packet classes for the training data of tcpreset attack 62

Figure 5.20. Empirical distribution of TCP RST packet classes during tcpreset attack 63

Figure 5.21. Constructed model for tcpreset attack zooming in the TCP RST packet classes 64

Figure 5.22. Average empirical distribution of packet classes for the training data of mailbomb attack 65

Figure 5.23. Constructed model based on maximum entropy principle for mailbomb attack 66

Figure 5.24. Average empirical distribution of packet classes for the training data of syslogd attack 67

Figure 5.25. Average empirical distribution for syslogd attack in the vicinity of class 1814 68

Figure 5.26. Constructed model based on maximum entropy principle for syslogd attack 69

Figure 5.27. Given that H_0, H_1, H_3, H_4, H_5 or H_6 are chosen, joint probability of each packet set between 8.00 am and 00.10 pm 71

Figure 5.28. Hypothesis testing results for apache2 instances existing on Monday at week five 73

Figure 5.29. Hypothesis testing result for apache2 instance existing on Wednesday at week five 74

Figure 5.30. Empirical distribution of packet classes during the back attack instance taking place in the data captured on Monday at week two 75

Figure 5.31. Probability of the class nine in each packet windows captured between 09.22 am and 10.22 am on Monday at week two 76

Figure 5.32. Hypothesis testing result for back attack instance taking place on Friday at week five 77

Figure 5.33. Hypothesis testing result for dosnuke attack instance taking place on Thursday at week four 79

Figure 5.34. Probability of the class 15 in each packet windows captured between 07.30 pm and 08.30 pm on Monday at week five 79

Figure 5.35. Hypothesis testing result between H_0 and H_4 for neptune attack instance taking place on Thursday at week two 81

Figure 5.36. Hypothesis testing result between all hypotheses for the data captured on Thursday at week two 82

Figure 5.37. Hypothesis testing result between H_0 and H_5 for udpstorm attack instance taking place on Tuesday at week five 84

Figure 5.38. Hypothesis testing result between H_0 and H_6 for tcpreset attack instance taking place on Wednesday at week five 85

Figure 5.39. Probability of the class three in each packet windows captured between 02.20 pm and 03.20 pm on Tuesday in the week two 87

Figure 5.40. Hypothesis testing result between H_0 and H_7 for mailbomb attack instance taking place on Tuesday at week two 88

Figure 5.41. Hypothesis testing result between H_0 and H_8 for syslogd attack instance taking place on Friday at week five 89

LIST OF TABLES

Table 4.1. Date and time of simulation data [32] 36

Table 4.2. Denial of service attacks in week two, week four and week five 39

Table 4.3. Training data used for each attack 40

Table 5.1. Top five feature functions selected for no attack 47

Table 5.2. Top five feature functions selected for apache2 attack 50

Table 5.3. Top five feature functions selected for back attack 53

Table 5.4. Top five feature functions selected for dosnuke attack 56

Table 5.5. Top five feature functions selected for neptune attack 59

Table 5.6. Top five feature functions selected for udpstorm attack 61

Table 5.7. Top five feature functions selected for tcpreset attack 63

Table 5.8. Top five feature functions selected for mailbomb attack 66

Table 5.9. Top five feature functions selected for syslogdattack 69

Table 5.10. Detection scores of target attacks 90

Table 5.11. Performance analysis table-1 92

Table 5.12. Performance analysis table-2 93

LIST OF SYMBOLS / ABBREVIATIONS

$P(w)$	Maximum Entropy (Baseline) Distribution
$\widetilde{P}(w)$	Empirical Distribution
$D(\widetilde{P} // P)$	Kullback-Leibler Distance between Empirical and Baseline Distribution
$E_P(f_i)$	Expected Value of Baseline Distribution
$E_{\widetilde{P}}(f_i)$	Expected Value of Empirical Distribution
$f_i(w)$	Feature Function
GN	Gain
M	Number of Hypothesis
Z	Normalization Constant
λ_i	Parameter
CPU	Central Processing Unit
DARPA	Defense Advanced Research Project Agency
DoS	Denial of Service
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
HIDS	Host Based Intrusion Detection System
IDS	Intrusion Detection System
IP	Internet Protocol
LBFGS	Limited Memory Variable Metric Method
NIDS	Network Based Intrusion Detection System
NPV	Negative Predictive Value
PMF	Probability Mass Function
PPV	Positive Predictive Value
RST	Reset Packet
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol

SYN	Synchronize Packet
TELNET	Teletype Network
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol

1. INTRODUCTION

With the expansion and an increase in computerization, a rise in computer misuse and attacks on networks has been encountered. The scale of this problem based on the unavoidable defects in the design of any computer or network. No software will ever be faultless and even minor bugs can leave systems vulnerable to costly malicious damage. Prevention of such crime is not possible. Therefore, monitoring and detecting are referred as the best alternative line of defense. This process is called Intrusion Detection and it is the process of detecting computer misuse and any attempt to break into networks [1].

In this thesis, a new method is proposed for intrusion detection and this method is applied to denial-of-service (DoS) attacks. Thus, the aim of this section is providing brief explanation of intrusion detection concept and denial-of-service attacks. Firstly, intrusion detection systems will be introduced and then the denial-of-service attacks will be examined in details. Finally, this section will be concluded with the motivation and the outline of this thesis.

1.1. Intrusion Detection Concept

Intrusions are defined as attempts to venture the confidentiality, integrity or availability of a computer system or to overcome its security mechanisms. They can be performed by attackers accessing a system from outside the system and by authorized users of the systems who abuse or misuse the privileges given to them [2].

The first step in securing a networked system is to detect the intrusion. Even if the system cannot prevent the intrusion, realizing it will provide valuable information to the security system. In this manner, the Intrusion Detection (ID) can be considered to be the first line of defense for any security system [3]. In other words, intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions. An Intrusion Detection System (IDS) is the software and hardware implementation that automates this monitoring and analysis process [2].

1.2. Intrusion Detection Systems

Intrusion detection systems employ techniques for modeling and recognizing intrusive behaviors in a system, which come in different variants. Generally, intrusion detection systems do not take preventive actions when an attack is detected. Only it notifies a human analyst of a possible intrusion [4]. In this manner, our proposed ID method raises alarm and does not take preventive actions. However, there are some systems as well, which take active steps to stop an intruder at the time of detection [4].

Intrusion detection systems assume that they can detect an intruder by processing the information gathered from logs, audit data or behavior observations [1]. According the source of the information, an IDS can be classified as host based IDS (HIDS) or network based IDS (NIDS). A host based IDS monitor the network traffic of a particular host and some system events on the host itself. It may be installed on each host or simply on some chosen critical ones within a network. A network based IDS monitors network traffic by capturing and analysing network packets [1], [5], [6]. In the host based IDS, system will only protect its own local machine. On the other hand, in the network based IDS, a distributed system will protect the network as a whole [3]. Proposed ID method in this work can be classified as network based, as it processes network packets captured from the network traffic and saved through the tcpdump program.

At the heart of intrusion detection lays the ability to distinguish acceptable normal system behavior from the abnormal or harmful one. In order to make this distinction, two different approaches can be employed: Misuse Detection and Anomaly Detection [7]. IDSs based on the misuse detection approach try to match computer activity to stored signatures of known exploits or attacks. Thus, misuse detection systems use a priori knowledge on attacks to look for attack traces. On the other hand, IDSs exploiting anomaly detection method first focuses on learning the characteristics of normal activity and then detecting anything that deviates from normal network activity [5], [6].

Misuse approach can detect known attacks accurately, but is ineffective against previously unseen attacks, as no signatures are available for such attacks. Anomaly detection overcomes the limitation of misuse detection by focusing on normal system

behaviors, rather than attack behaviors. This approach is characterized by two phases: in the training phase, the behavior of the system is observed in the absence of attacks, and machine learning techniques used to create a profile of such normal behavior. In the detection phase, this profile is compared against the current behavior of the system, and any deviations are flagged as potential attacks. Unfortunately, systems often exhibit legitimate but previously unseen behavior, which leads anomaly detection techniques to produce a high degree of false alarms. Moreover, the effectiveness of anomaly detection is affected greatly by what aspects (also called "features") of the system behavior are learnt. The problem of selecting an appropriate set of features has proved to be a hard problem [8].

Our proposed method can be considered as a combination of anomaly and misuse detection methods, since it learns not only the characteristics of known attacks but also the characteristics of the normal network activity and decides which one of these learned models most probably satisfies the characteristics of the real-time network traffic. However, it is closer to the misuse detection method, as it is based on prior knowledge and can not detect unseen attacks.

1.3. Denial of Service Attacks

A denial-of-service (DoS) attack can be defined as an explicit attempt to prevent access to a service. DoS attacks disable a network service by flooding connections, crashing servers or programs running on the servers and exhausting server resources. Obviously, these attacks prevent legitimate clients from accessing the network service [9].

DoS attacks range from single packet attacks that crash servers to coordinated packet floods from multiple hosts. In single packet attacks, a carefully crafted packet that exploits a known operating system or application vulnerability is sent through the network to disable a server or any associated services it performs [4]. An example of this type DoS attack is syslogd attack. Most frequently encountered DoS attacks are performed against network connectivity. The intruder aims to prevent hosts or networks from communicating on the network. A common example of this type of attack is the "SYN flood" attack. [9].

An intruder can also use your own resources against you in unexpected ways. One example is UDP port denial of service attack. In this attack the intruder uses forged UDP packets to connect the echo service on one machine to the chargen service on another machine. As a result, two services consume all available network bandwidth between them. In this manner, the network connectivity for all machines on the same networks as either of the targeted machines may be affected [9].

Furthermore, all available bandwidth on a network can be consumed by generating a large number of packets directed to that network. Usually these packets are ICMP ECHO packets, but these packets may be anything else in principle [9].

In addition to network bandwidth, intruder may be able to consume other resources that your systems need in order to operate. These may include data structures, CPU or disk space. For example, an intruder may attempt to consume disk space by generating excessive numbers of mail messages, namely mailbomb attack [9].

In the following subsections, some major types of DOS attacks will be examined in details.

1.3.1. Apache2 Attack

The apache2 attack is a denial of service attack against an apache web server. In this attack, lots of http request with http headers are sent to the web server. Since the server receives many of these http requests, it slows down, and in the long run it may crash [4].

A typical http request contains twenty or fewer headers. However, every http request submitted as part of this exploit contains many http headers. Although the exact number and value of these headers could be varied between attack instances, generally the number of headers is 500 times greater than the number of headers in a normal request. Moreover, the actual content of the header is not important for the exploit, the exploit is only dependent on the fact that http request contain many headers [4].

1.3.2. Back Attack

Back attack is a denial of service attack against the apache web server. This attack performs malformed web requests to port 80 with the payload, “\Get //// ...” followed by many slashes [10]. Since the server tries to process these requests, it will slow down and becomes unable to process other requests

An intrusion detection system can detect back attack by checking the payloads of network packets. So, the requests carrying more than some number of frontslashes in payload should be considered an attack [4].

1.3.3. Dosnuke Attack

Dosnuke is a denial service attack against NetBIOS. In this attack, “out-of-band” data (MSG_OOB) data that is delivered with higher priority than ordinary data are sent to NetBIOS port of the victim machine. These packets being sent by the attacking machines are flagged "urg". When a Windows system receives a packet with the "urg" flag set, it expects data will follow that flag. The exploit consists of setting the flag "urg", but not following it with data [11].

It can be detected by searching the sniffed data for a NetBIOS handshake followed by NetBIOS packets with the "urg" flag set.

1.3.4. Land Attack

The Land attack is a denial of service attack that is effective against some older TCP/IP implementations. The Land attack occurs when an attacker sends a spoofed TCP SYN packet that contains same IP address as the source and the destination addresses [4]. Such a packet completely locks the victim system. However, it is quite noticeable, because IP packets with identical source and destination addresses should never exist on a properly working network.

1.3.5. Mailbomb Attack

Mailbomb is an attack in which the attacker sends many messages to a server, overflowing that server's mail queue and possible causing system failure [4].

An intrusion detection system that is looking for a mailbomb attack can look for thousands of mail messages coming from or sent to a particular user within a short period of time. However, the aim of this type of attack is not to produce a high traffic volume, but to create a large size of messages in the server's queue. Therefore, by looking only at the IP header, a normal host might exhibit behavior similar to that of an attacker since it could send mail with large attachments to the SMTP server [4].

1.3.6. Neptune (SYN-Flood) Attack

In order to establish a TCP connection between a client and server systems, a set sequence of messages is required to be exchanged. Firstly, the client system sends a SYN message to the server to request a TCP connection. The server then acknowledges the SYN message by sending SYN-ACK message to the client. The client then finishes establishing the connection by responding with an ACK message. The connection between the client and the server is then open, and the service-specific data can be exchanged between the client and the server [12].

The potential for abuse arises at the point where the server system has sent an acknowledgment (SYN-ACK) back to client but has not yet received the ACK message. This is what we mean by half-open connection [12].

A SYN Flood is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). Each half-open TCP connection made to a machine causes the "tcpd" server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections. The half-open connections data structure on the victim server system will eventually fill and the system will be unable to accept any new incoming connections until the table is emptied out. Normally there is a

timeout associated with a pending connection, so the half-open connections will eventually expire and the victim server system will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim system can expire the pending connections. In some cases, the system may exhaust memory, crash, or be rendered otherwise inoperative [4].

A Neptune attack can be distinguished from normal network traffic by looking for high number of simultaneous SYN packets destined for a particular machine that are coming from an unreachable host [4].

1.3.7. Ping of Dead Attack

A ping of death (abbreviated "POD") is a denial service attack, in which attacker sends a malformed or otherwise malicious ping to a victim system. A ping is normally 64 bytes in size and carried by the ICMP protocol. Many computer systems cannot handle a ping larger than the maximum IP packet size, namely 65,535 bytes. Sending a 65,536 byte ping packet is illegal according to networking protocol, but a packet of such a size can be sent if it is fragmented; when the target computer reassembles the packet, a buffer overflow can occur. Possible reactions include crashing, freezing, and rebooting [13].

An attempted ping of death can be identified by checking the size of all ICMP packets and flagging those that are longer than 64000 bytes [4].

1.3.8. Process Table Attack

Process table attack is a type of DoS attack, which exploits the feature of some network services to generate a new process each time a new TCP/IP connection is set up. The attacker tries to make as many uncompleted connections to the victim as possible in order to force the victim’s system to generate an abundance of processes. Hence, because the number of processes that are running on the system cannot be boundlessly large, the attack renders the victim unable to serve any other request [11].

An intrusion detection system that is trying to detect a process table attack will need to use somewhat subjective criteria for identifying the attack, since this attack consists of abuse of a perfectly legal action [4].

1.3.9. Smurf Attack

Smurf attack is a denial-of-service attack, in which ICMP echo request packets are exploited. The attacker sends ICMP echo request packets to the broadcast address of many subnets with the source address spoofed to be that of the intended victim. Any machines that are listening on these subnets will respond by sending ICMP "echo reply" packets to the victim. So, this attack results in a large continuous stream of "ECHO" replies that flood the victim [4].

The smurf attack can be identified by checking if there are a large number of "echo replies" being sent to a particular victim machine from many different places, but no "echo requests" originating from the victim machine [4].

1.3.10. Syslogd Attack

The Syslogd exploit is a denial-of-service attack that allows an attacker to remotely kill the syslogd service on a Solaris server. Under Solaris, if syslogd receives a packet containing an irresolvable IP address, it will crash with a segmentation fault [4].

This attack can be identified with a network-monitoring intrusion detection system, which checks packets destined for the syslog port whether or not the source address is irresolvable. Actually, it may not be realistic for an intrusion detection system to check every packet destined for the syslog [4].

1.3.11. Tcpreset Attack

Tcpreset attack is a denial-of-service attack, in which the attacker intends to kill all TCP connections to a host. The network is monitored for TCP connection requests to the

victim. As soon as such a request is found, the malevolent attacker sends a spoofed TCP RESET packet to the victim and obliges it to terminate the TCP connection [11].

In order to detect this attack, TCP session/takedown rate can be checked and then especially focused on cases in which RESET packets appear to come from the machine that had initially attempt to begin the connection [11].

1.3.12. Teardrop Attack

In this attack, IP fragments with overlapping and oversized payloads are sent to the target machine. A bug in the TCP/IP fragmentation re-assembly code of various operating systems caused the fragments to be improperly handled, crashing them as a result of this [14].

1.3.13. Udpstorm Attack

An Udpstorm attack is a denial of service attack exploits two victim machines that unconsciously attack each other. Firstly, the attacker forges a single packet that has been spoofed to look like it is coming from the echo port on the first victim machine and sends it to the second victim. The echo service blindly responds to any request it receives by simply echoing the data of the request back to the machine and port that sent the echo request, so when the victim receives this spoofed packet it sends a response to the echo port of the second victim. This second victim responds similarly, and the loop of traffic continues. As a result, network congestion and slowdown occur [4].

Once the loop of network traffic has been initiated, an intrusion detection system that can notice network traffic on the inside of the network can note that traffic is being sent from the chargen or echo port of one machine to the chargen or echo port of another [4].

1.4. Motivation

In this thesis, statistical characterization of network events is intended, in order to detect denial-of-service attacks. Most of the statistical modeling techniques implemented

based on anomaly detection employs parametric signal processing techniques. The number of the intrusion detection approaches exploiting the non-parametric methods is not much. The leading works combining non-parametric signal processing methods and information theory approaches for anomaly detection are proposed in [15], [16] and [17].

In [16], the problem of dynamic system monitoring is examined with partial active measurements by applying the maximum entropy principle at each time point. A network anomaly detection technique based on maximum entropy and relative entropy techniques is proposed in [15]. According to their survey, the maximum entropy technique provides a flexible and fast approach to estimate the baseline distribution, which also gives the network administrator a multi-dimensional view of the network traffic. By computing a measure related to the relative entropy of the network traffic under observation with respect to the baseline distribution, it is possible to distinguish anomalies that change the traffic either abruptly or slowly [15]. In this way, some improvements have been performed on these anomaly detection methods examined in [16] and [15]. For example, in [15], detection of an anomaly is achieved by using a threshold for the difference between baseline distribution and computed current distribution. This detection algorithm has been enhanced by employing hypothesis testing technique. Furthermore; in [15], the baseline distribution is estimated by using a training set under no attack condition. Therefore; this model can only detect whether an anomaly occurred or not and it can not be used to identify the attack. However; as an improvement, more then one baseline distributions have been estimated by using training set under various attacks and no attack conditions. In this case, identifying the attack has been possible. Moreover, our proposed method can be considered as a combination of anomaly and misuse detection methods, since it learns not only the characteristics of the normal network activity but also the characteristics of known attacks characteristics of known attacks and decides which one of these learned models most probably satisfy the characteristics of the real-time network traffic. In this manner, it overcomes high degree of false alarms encountered anomaly detection methods.

As a result, the purpose of this study is to improve entropy based intrusion detection methods and bring a formularization to ad hoc rules by using information theory and statistical signal processing.

1.5. Thesis Outline

The rest of the thesis is organized in the following fashion. Section two presents background information about the transport protocols (TCP and UDP), maximum entropy framework and hypothesis testing techniques. This provides perspective for later discussion on the proposed method.

Section three describes the method proposed for the detection of denial-of-service attacks. Firstly, model estimation process based on the maximum entropy principle is examined in details. Then the detection process employing hypothesis testing technique is explained.

In the section four introduces DARPA 1999 simulation network and data set exploited throughout this work.

In section five, the implementation of the proposed method is presented and the results are discussed in details. Eventually, the thesis is concluded in the section six.

2. BACKGROUND

2.1. TCP and UDP Protocol Overview

Network packets form the input of our method. According a packet classification algorithm, packets are divided into groups. This classification is performed based on the transport protocol related information gathered from the header of the packets. This section aims to give background information about the major transport protocols in computer networks.

The major transport protocols in computer networks are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP is a widely used transport protocol and it provides a reliable transport of data. Many well-known application protocols, e.g., HTTP and FTP, are based on TCP. In contrast to TCP, UDP cannot guarantee a reliable transport of data. It is therefore mainly used for the transport of data whose performance is generally more affected by delay than by losses, e.g., real-time audio or video traffic [18].

A TCP segment is the packet of information that TCP uses to exchange data with its peers. TCP segment consists of a header and a payload. Data portion of a TCP packet is the payload and it may be carrying any application layer protocols such as HTTP, Telnet, SSH, or FTP. TCP header consist of eleven fields, however ten of them are required. The figure 2.1 illustrates the header of a TCP segment. The fields in the TCP header, which carry valuable information for our classification algorithm, are destination port and flag fields. The source and destination port of the TCP connection are carried in the first two fields in the TCP header. Together with the source and destination address of the end systems stored in the IP header these two values are used to clearly identify the two processes in the end system which are linked with the TCP connection. If the urgent bit (URG) is set, the urgent pointer is valid. Then the payload (or parts of the payload) is marked as urgent data. These data should be immediately transferred to the receiver, even if this requires the sending of short TCP segments much smaller than the allowed maximum segment size (MSS). The acknowledgment bit (ACK) is set in a TCP segment to indicate that the AckNr

field is valid. Then this segment acknowledges data or confirms the connection setup or connection-teardown request of a remote TCP instance. The push bit (PSH) is used to tell the receiver of the TCP segment to immediately acknowledge this segment after reception. With the reset bit (RSH) the whole TCP connection can be re-initialized by resetting the TCP instances. The SYN or FIN bit are set in the first or last TCP segment of a TCP stream of one direction to setup or teardown the TCP connection in this direction [19].

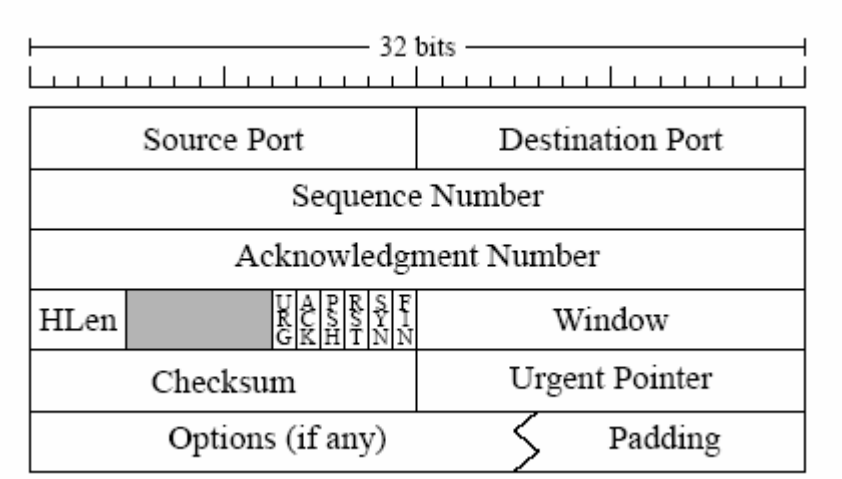

Figure 2.1. TCP header [19]

UDP datagram consists of a header and a payload carrying, for example, real-time audio data. Figure 2.2 presents the header structure of an UDP packet. The first two fields in the UDP header contain the source and destination port of the UDP flow. Together with the source and destination address of the end systems stored in the IP header these two values are used to clearly identify the two processes in the end systems which exchange data using the UDP flow. The next field stores the overall length of the UDP message, i.e., the sum of the header and the payload length of the UDP datagram. And the last field of the UDP header contains the checksum of the UDP datagram [19].

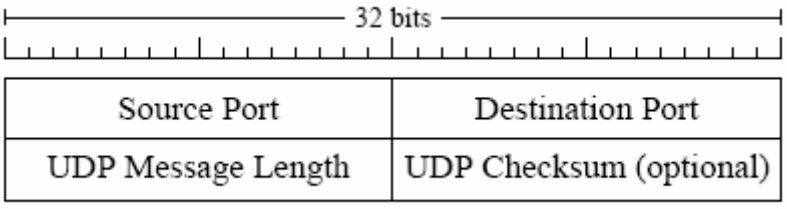

Figure 2.2. UDP header [19]

2.2. Maximum Entropy Approach

The ancient problem was how to find a probability distribution for a quantity when only some experimental measurements are available and still some information are missing due to the lack of observed data. This problem yields the born of the maximum entropy concept.

Laplace proposed in his publication “Principle of Insufficient Reason”: when one has no information to distinguish between the probabilities of two events, the best strategy is to consider them equally likely. A related example of such thinking is the Occam’s razor or least complex hypothesis selection. However, Laplace can be considered as a father of Maximum Entropy, since his “Principle of Insufficient Reason” was the first attempt to apply an unbiased method into field of probability theory.

The practical formalism and the answer to the fundamental problem was first introduced by the E.T. Jaynes [20] and then updated by I.J.Good [21]. Jaynes suggested that the most unbiased selection of the probability distribution can be made by maximizing Shannon’s entropy:

$$H(x) = -\sum p(x) \log p(x) \tag{2.1}$$

This leads to the distribution that has maximum entropy, maximum uncertainty, which is the most unbiased and most uniform. At the same time the maximum entropy principle keeps the probability distribution consistent with the observed constraints.It yields a probability distribution which is “most likely” to have caused the observed data. This does not infer that the probability distribution will necessarily be correct, but in order to obtain a better distribution, more information must be provided.

In this work, maximum entropy principle is employed while modeling the benign and malignant network traffic. Captured packets from network in a time interval form the observed constraints. Maximizing entropy, while satisfying the experimental constraints, yields a probability mass function in the form of Gibbs equations. The details of our

method can be found in section three. The inferences and calculations of our maximum entropy based modeling process are presented in Appendix A.1.

2.3. Hypothesis Testing

One may be faced with the problem of making a definite decision with respect to an uncertain hypothesis which is known only through its observable consequences. A statistical hypothesis test, or more briefly, hypothesis test, is an algorithm to state the alternative (for or against the hypothesis) which minimizes certain risks [22].

The basic components of a simple decision problem are illustrated in Figure 2.3. The first component is the source that outputs a choice. In the simplest case, one of two choices is generated as an output. These two choices are referred as hypotheses and labeled as H_0 and H_1 in the two-choice case. More generally, the output might be one of M hypotheses, which are labeled as H_0, H_1,....., H_{M-1} [23]. Some typical source mechanisms are the following [23]:

- A digital communication system transmits information by sending ones and zeros. When "one" is sent, it is called as H_1, and when "zero" is sent, it called as H_1.
- In a radar system, by looking at a particular range and azimuth and it is tried to decide whether a target is present; H_1 corresponds to the presence of a target and H_0 corresponds to no target.
- In a speaker classification problem, it is known that the speaker is German, British, or American and either male or female. There are six possible hypotheses.

In our work, it is known that one of eight attack types or no attack takes place at a certain time. This yields nine hypotheses to decide. However, it is not known which hypothesis is true at a certain time.

The second component of the problem is a probabilistic transition mechanism; the third is an observation space. The observation space consists of finite number observations used to make a decision. The transition mechanism calculates joint probabilities of

observations in the observation space by assuming respectively each hypothesis is true [23].

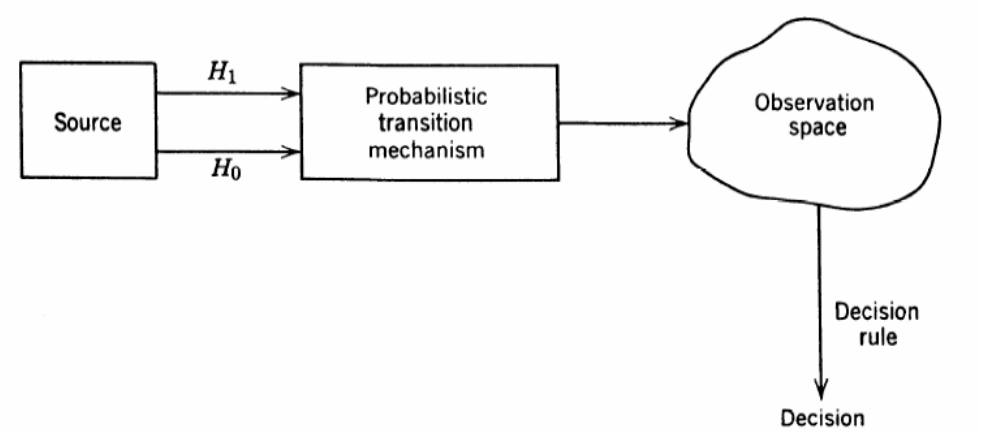

Figure 2.3. Components of a decision problem [23]

In this study, network packets captured in a window forms our observation space. Given that one of nine hypotheses is true, transition mechanism calculates the joint probability for observation space and it repeats this for each hypothesis.

The fourth component of the detection problem is a decision rule. After observing the outcome in the observation space, it should be guessed which hypothesis was true, and so there is need to develop a decision rule [23].

In this thesis, the hypothesis, of which conditional probability is maximum at an observation space, is guessed to be true.

3. METHODOLOGY

3.1. Introduction

In this section, method that we proposed for the detection of DoS attacks will be examined. Our approach exploits the concept of behavior-based anomaly detection and is based on maximum entropy and M-hypothesis testing techniques. In this approach, DoS attacks can be detected by comparing the current network traffic against the models for benign and malignant traffic.

This approach uses packets as input data and focuses on the information in the packet header. Packets are classified according to protocol, destination port and flag information. These packet classes form the domain of the probability space.

At the training part, baseline distributions of these packet classes are determined under benign and malignant traffic. Based on the maximum entropy technique, models are formed for various attack types and attack-free case. At the detection part, m-hypothesis testing is employed to decide which model is most probably representing the current network traffic.

3.2. Packet Classification

Packet classification method proposed in [15] is exploited to divide packets in the network traffic into a set of packet classes.

This packet classification method focuses on the information in the packet header. The protocol information and the destination port numbers are considered as two of the most important packet header information in the network traffic. Since TCP and UDP carry majority of today's network traffic, this method deals with anomalies concerning TCP and UDP packets, only. According to the protocol information and the destination port number in the packet header, network data is formed into a set of two-dimensional packet classes, so that this set of packet classes serve as the domain of the probability space.

In the first dimension, packets are classified according to the transportation protocol and the flag information. First, by using transportation protocol information, packets are grouped as TCP packet class and UDP packet class. Since SYN and RST packets provides additional information about the network traffic, TCP packet class is separated into two other classes according to whether or not the packets are SYN or RST packets. A SYN packet is sent to initiate a TCP connection and it causes the receiving host to assign resources for the coming TCP connection. Therefore, SYN packets are often used to perform SYN flooding attacks. RST packets are generated to reset or reject a TCP connection. For instance, if an attempt to connect a non-existing port occurs, a TCP RST packet is generated by the target host. Therefore, occurrences of TCP RST packets indicate TCP connection failures and an increase in TCP connection failures may relate to network intrusions. In this manner, first dimension includes totally four packet classes; such that TCP, UDP, TCP SYN and TCP RST.

In the second dimension, packets are classified according to their destination port numbers, since port numbers generally give information about the services related to the packet exchange. Internet Assigned Numbers Authority [24] is responsible for assigning ports to specific uses. For instance, port number 80 is assigned to web services; port numbers 20 and 21 is assigned to FTP services; port number 22 is assigned SSH Remote login protocol; port number 23 is assigned to TELNET network protocol used in Internet or local area networks (LAN) connections and port number 25 is assigned to mail services (SMTP). According to the Internet Assigned Numbers Authority [24], port numbers are classified into three categories: Well Known Ports, Registered Ports, and Dynamic and/or Private Ports. Well Known Ports are those from 0 through 1023, Registered Ports are those from 1024 through 49151, and Dynamic and/or Private Ports are those from 49152 through 65535. In [15], the classes for well known ports, registered ports, and dynamic ports are divided into sub-classes to make classification according to destination port. Packets with destination port numbers between 0 and 1023 are separated into classes of 10 port numbers each. Since packets with destination port number 80 cover the majority of the network traffic, this port is assigned to a single class. Thus, packets with destination port numbers in the range (81, 89) constitute another separate class. Additionally, packets with destination port numbers in the range (1020, 1023) constitute another class. In this manner, 104 packet classes are formed from the category of well known ports. Packets with

destination port numbers between 1024 and 49151 are divided into 482 further classes with each class containing 100 port numbers with the exception of the class that comprise port numbers from 49124 to 40151, which contains 28 port numbers. Packets with destination port numbers larger than 49151 are assigned to a single class. Hence, in this dimension, packets are grouped into a total of 104 + 482 + 1 = 587 classes.

Overall, the set of the two-dimensional classes consists of 4 * 587 = 2348 packet classes. The figure 3.1 illustrates the set of the packet classes. Each protocol class includes 587 port classes. For example, a TCP packet with the port number 25 is assigned to the class two. Similarly, a TCP-RST packet with the port number 80 is assigned to the class 1183 and a UDP packet with the port number 50000 assigned to the class 2348.

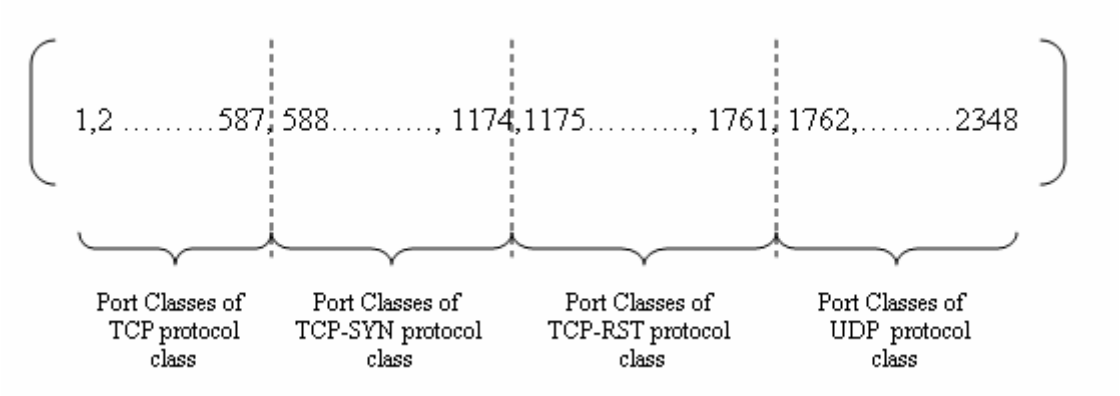

Figure 3.1. The set of packet classes

These packet classes serve as the domain of the probability space through this work. Therefore, it is important to understand the structure shown in Figure 3.1. According to this packet classification, different baseline distributions will be estimated for the traffic data without attack and with various types of attacks. Baseline distributions, which are generated with the maximum entropy estimation process, will be used to detect attacks.

3.3. Maximum Entropy Estimation of the Packet Class Distribution

The task is to learn statistical models for the attack free case and for different attack types from the training data. Calculating the empirical distribution of training data gives us an idea about the probability distribution model; however empirical distribution is a poor

estimate of the distribution model, since the training set is finite. Therefore, there is a need to handle missing parts in training data in order to estimate a probability density model generated from the empirical data in a reasonable way. Due to the lack of the observed information, the most uncertain, unbiased way to assign these probabilities is to choose uniform distribution as the target distribution. However, in the real network traffic, probability distribution model is not likely to be uniform. Therefore; based on the observation it is possible to add some constraints [25].This leads to the maximum entropy concept. According to the maximum entropy principle, the distribution model is required to satisfy a set of constraints reflecting properties of the training data and to make no other assumptions. Thus, maximum entropy estimation produces a model with the most uniform distribution among all the distributions satisfying the given constraints [20].

Let Ω be the set of packet classes defined in the previous section. Given a sequence of packets $S = \{x_1,...,x_n\}$ as the training data, the empirical distribution $\widetilde{P}$ over Ω in this training data can be calculated as follows [15]:

$$\widetilde{P}(w) = \frac{\sum 1(x_i \in w)}{n} \tag{3.1}$$

where $1(X)$ is an indicator function that takes value 1 only if X is true and 0 otherwise.

As mentioned before, maximum entropy approach keeps the probability distribution of the model consistent with the observed constraints. These constraints are added to the model by feature functions. Suppose there is a set of feature functions $F = \{f_i\}$, f_i is a feature function, which is an indicator function $f_i : \Omega \to \{0,1\}$. By using the Maximum Entropy estimation, aim is to find a density model P; such that the constraints $E_P(f_i) = E_{\widetilde{P}}(f_i)$ for all $f_i \in F$ are satisfied and entropy is maximized. In [26], it has been proved that under the constraints $E_P(f_i) = E_{\widetilde{P}}(f_i)$ for all $f_i \in F$, the solution of this problem turns out to be a distribution in the form of generalized Gibbs distribution. The equation 3.2 presents the form of this distribution.

$$P(w) = \frac{1}{Z}\exp(\sum_i \lambda_i f_i(w)) \tag{3.2}$$

For each feature f_i, a parameter $\lambda_i \in \Lambda$ determines its weight in the model. Λ is the set of parameters for the feature functions in the model. Z is a normalization constant that guarantees that the sum of the probabilities over Ω is 1. Maximizing the likelihood of the distribution in the form of (above equation) with respect to $\widetilde{P}$ is equivalent to minimizing the Kullback-Leibler divergence of $\widetilde{P}$ with respect to P

$$P = \arg\min_P D(\widetilde{P} // P) \tag{3.3}$$

The maximum entropy estimation method has two parts: feature selection and parameter estimation [26]. In the feature selection part, the most important features of the log-linear model are specified, and in the parameter estimation part a proper weight to each of the feature functions is assigned. In an iterative manner, these two parts are repeated such that a new feature function is determined at each iteration and the weights for the feature functions in the log-linear model are adjusted as each new feature function is added. As new feature functions are added into the model, the model approaches to the empirical distribution; and the parameter estimation guarantees that the maximum entropy principle is satisfied by the model [15]. In the following sub-sections, feature selection and parameter estimation will be examined in detail.

3.3.1. Feature Selection

Feature functions are employed to specify constraints on the model. The superiority of the resulting models is dependent on the ability of these feature functions to capture the relevant information. Therefore, the process of defining feature functions is the core of the modeling. This process called as feature selection chooses the most important feature function from a set of candidate feature functions that minimizes the difference between the model distribution and the empirical distribution. The steps of this process is given below, more details can be found in [15] and [26].

Let Ω be the set of all packet classes, $\widetilde{P}$ the empirical distribution of Ω in the training data, and F a set of candidate feature functions. At the initial state, no information is known about the training data. Hence the initial model distribution over Ω is uniform and expressed as follows:

$$P_0(w) = \frac{1}{Z}, \qquad Z = |\Omega|, \tag{3.4}$$

where the subscript 0 indicates that there is no feature in the model.

Suppose g is a candidate feature function $g \in F$. For each candidate feature function g, a parameter family of distributions $\{P_{\lambda_g g} : \lambda_g \in R\}$ can be defined as follows:

$$P_{0,\lambda_g,g}(w) = \frac{1}{Z'} \exp(\lambda_g g(w)) \tag{3.5}$$

The family of distribution is referred to as the indication of P_0 by g. The difference of the two Kullback-Leibler divergences related to P_0 and $P_{0,\lambda_g,g}$ can be defined as follows:

$$GN_{P_0}(\lambda_g, g) = D(\widetilde{P} // P_0) - D(\widetilde{P} // P_{0,\lambda_g,g}) \tag{3.6}$$

$GN_{P_0}(\lambda_g, g)$ is the improvement that future g brings to the model with its weight λ_g. $GN_{P_0}(g)$ to be the greatest improvement by adding g to P_0 can be presented as follows:

$$GN_{P_0}(g) = \sup_{\lambda_g} GN_{P_o}(\lambda_g, g) \tag{3.7}$$

and $GN_{P_0}(g)$ acts as the gain of the candidate feature g. The feature function in F with the maximum gain is chosen to minimize the difference between the model distribution and the empirical distribution. The selected feature function f_1 is as follows:

$$f_1 = \arg\max_g GN_{P_0}(g) \tag{3.8}$$

The new model with one feature function selected can be given as follows:

$$P_1(w) = \frac{1}{Z}\exp(\lambda_1, f_1(w)) \tag{3.9}$$

In order to generalize this process, let P_i be a model with i feature functions selected:

$$P_i(w) = \frac{1}{Z}\exp(\sum_{j=1}^{i} \lambda_j f_j(w)) \tag{3.10}$$

To select the $i+1^{st}$ feature, let g a function in $F \setminus \{f_1,...f_i\}$ that has not yet been selected. So, we obtain

$$P_{i,\lambda_g,g}(w) = \frac{1}{Z'}\exp(\sum_i \lambda_i f_i(w))\exp(\lambda_g g) \tag{3.11}$$

All parameters would require to be adjusted after the addition of a new feature function; however, it is computationally costly to calculate the exact form of the model with g added for all g in the candidate feature function set. Instead of changing all parameters, the improvement of adding a new feature g can be approximated by adjusting only the weight of the parameter of g. This approximation makes feature selection convenient for large number of candidate features [15]. Considering this approximation,

$$GN_{P_i}(\lambda_g, g) = D(\widetilde{P} // P_i) - D(\widetilde{P} // P_{i,\lambda_g,g}) = \sum_w \widetilde{P}(w)\log\frac{P_{i,\lambda_g,g}}{P_i(w)} \tag{3.11}$$

$$GN_{P_i}(\lambda_g, g) = \lambda_g E_{\widetilde{P}}(g) - \log E_{P_i}(\exp(\lambda_g g)) \tag{3.12}$$

which is a concave function with respect to λ_g, and

$$GN_{P_i}(g) = \sup_{\lambda_g} GN_{P_i}(\lambda_g, g) \tag{3.13}$$

The feature function g with the largest gain $GN_{P_i}(\lambda_g, g)$ is selected as the $i+1^{st}$ feature function to the model.

In [27], it is also proved that for a candidate feature function g in the form of an indicator function, $GN_P(\lambda_g, g)$ is maximized by

$$\hat{\lambda}_g = \arg\max_{\lambda_g} GN_P(\lambda_g, g) = \log\left\{\frac{E_{\widetilde{P}}(g)(1-E_P(g))}{E_P(g)(1-E_{\widetilde{P}}(g))}\right\} \tag{3.14}$$

Here, $E_{\widetilde{P}}(g)$ is the expected value of g with respect to the distribution of $\widetilde{P}$ and $E_P(g)$ is the expected value of g with respect to the distribution of P.

After the addition of a new feature function to the model, the weight of all feature functions in the model will be updated with the parameter estimation procedure. This procedure aims to minimize the Kullback-Leibler divergence between the model distribution and the empirical distribution.

3.3.2. Parameter Estimation

After feature selection, next step is parameter estimation process. Suppose up to now i features are selected and as a result a maximum entropy model like in equation 3.10 is obtained. The equation (3.10) is in a parametric form. Fitting such a maximum entropy model to a collection of training data entails finding values for the parameter vector $\Lambda = \{\lambda_i\}$ which minimize Kullback-Leibler divergence between the model P_i and the empirical distribution $\widetilde{P}$:

$$\Lambda = \arg\min_{\Lambda} \sum_{w\in\Omega} \widetilde{P}(w) \log \frac{\widetilde{P}(w)}{P(w)} \tag{3.15}$$

or, correspondingly, which maximize the log likelihood:

$$\Lambda = \arg\max_{\Lambda} \sum_{w \in \Omega} \widetilde{P}(w) \log P(w) \tag{3.16}$$

The gradient of the log likelihood function, or the vector of its first derivatives with respect to the parameter $\Lambda = \{\lambda_i\}$

$$G(\Lambda) = E_{\widetilde{P}}[f] - E_P[f] \tag{3.17}$$

The derivation of the gradient of log likelihood function can be found in Appendix A.2. Since the likelihood function in (3.16) is concave over the parameter space, it has a global maximum where the gradient is zero. Unfortunately, simply setting $G(\Lambda) = 0$ and solving for $\Lambda = \{\lambda_i\}$ does not yield a closed form solution; therefore it requires to be solved iteratively [28]. In order to solve this problem iteratively, there are various numerical methods that can be employed , including iterative scaling methods like generalized iterative scaling and improved iterative scaling, first order methods, and second order methods like quasi-Newton or limited memory variable metric method (LBFGS). The details and performance analysis of these parameter estimation algorithms can be found in [28]. According to the comparison between these algorithms given in [28], LBFGS algorithm performs substantially better than any of the competing methods for maximum entropy parameter estimation problem. Therefore, in this thesis LBFGS algorithm proposed in [29] and [30] is used as parameter estimation methhod.

LBFGS (stands for Limited memory Broyden-Fletcher-Goldfarb-Shanno) is a second order, limited memory, quasi newton optimization method. At each step of the parameter estimation algorithm, an estimate of the parameters $\Lambda(k)$ is adjusted to a new estimate $\Lambda(k+1)$ based on the divergence between the estimated probability distribution and the empirical distribution. This is repeated until successive improvements fail to yield a sufficiently large decrease in the divergence. Actually; all parameter estimation methods have the same general form above, however the way of computing the updates $\delta(k)$ at search step differs substantially between the parameter estimation methods.

LBFGS takes into account the gradient of the log-likelihood function and its curvature to find updates δ. The usefulness of the curvature is made clear if we consider a second-order Taylor series approximation of $L(\lambda+\delta)$:

$$L(\lambda+\delta) \approx L(\lambda) + \delta^T G(\lambda) + \frac{1}{2}\delta^T H(\lambda)\delta \tag{3.18}$$

where δ is update and H is Hessian matrix of the log-likelihood function, the d x d matrix of its second partial derivatives with respect to λ. By setting the derivative of above equation to zero and δ can be found as follows,

$$\delta^{(k)} = H^{-1}(\lambda^{(k)})G(\lambda^{(k)}) \tag{3.19}$$

However; for large scale problems the evaluation of the Hessian matrix is computationally impractical. Therefore; instead of computing the inverse of Hessian matrix LBFGS algorithm builds up an approximation of it using successive evaluations of the gradient. In this manner, $H^{-1}(\lambda^{(k)})$ is replaced with a local approximation of the inverse Hessian $B^{(k)}$:

$$\delta^{(k)} = B^{(k)}G(\lambda^{(k)}) \tag{3.20}$$

with $B^{(k)}$ a symmetric, positive definite matrix which satisfies the equation:

$$B^{(k)}y^{(k)} = \delta^{(k-1)}, \text{ where } y^{(k)} = G(\lambda^{(k)}) - G(\lambda^{(k-1)}) \tag{3.21}$$

Since storing the approximate Hessian is expensive for large scale problems, LBFGS method implicitly approximate the Hessian matrix in the vicinity of the current estimate of $\lambda^{(k)}$ using the previous m values of $y^{(k)}$ and $\lambda^{(k)}$ instead of using all previous values of $y^{(k)}$ and $\lambda^{(k)}$.

LBFGS algorithm is a very complex numerical method. In our work, it is regarded as a black box in the maximum entropy modeling process. The LBFGS codes that can be

found in [31] are modified, so that they can interact properly with the codes written for this work.

3.4. Model Induction

The theoretical explanation of our model based on maximum entropy principle is given previous sections. In this section, the implementation of this model will be described.

As explained before, the aim is to form log-linear density models which are reflecting the behavior of the network traffic without attack and with attack. So, at the end of this step, multiple models are obtained. One of these models represents the behavior of attack-free traffic and it is generated by an empirical distribution under normal network traffic. Each of the other models expresses the behavior of the network traffic, while a distinct type of DoS attacks exists. This means that repeating the model construction algorithm based on maximum entropy approach for various attack types and no attack case.

Model construction process includes iteration of two steps, those; feature selection step and parameter estimation step . The algorithm is initialized with a uniform distribution over all packet classes. In the feature selection steps, new features join to the model, and in the parameter estimation steps, the weights for the feature functions are modified. This procedure is iterated until some stopping criterion is met [15]. In this work; this stopping criteria is that the Kullback-Leibler divergence of P with respect to $\widetilde{P}$ is less than some threshold value. After a model is constructed, same procedure is repeated for other models.

- External iterated steps

  (0) Repeat it for other models

  A set of training data with empirical distribution $\widetilde{P}$; which is reflecting one of the attack types or no attack case
- Initial data:

  A set of candidate feature functions F

  An initial density model P_0, $P_0(w) = \frac{1}{Z}$, $Z = |\Omega|$

- Internal iterated steps

  (1) Set $n = 0$

  (2) Feature Selection

  For each feature function $g \in F$, $g \notin \{f_i\}$, compute the gain $G_{P_n}(g)$

  Let f_{n+1} be the feature function with the largest gain

  (3) Parameter Estimation

  Update all the parameters corresponding to the selected features and set P_{n+1} to be the updated model

  (4) Check the iteration stopping criterion

  If the iteration stopping criterion is not met, set $n = n + 1$, go to (2).

  Otherwise, return the learned model P_{n+1} for current empirical distribution and go to (0) to construct another model.

As mentioned before; feature functions are at the core of the modeling process and the quality of the constructed model is dependent on the ability of feature functions to capture relevant information. In this thesis; feature functions are binary-valued and this means that a feature function either fulfills its definition or not. Since our probability space Ω is 2348 and there are 2348 different type classes, the size of a feature function is 2348 as well. Figure 3.2 illustrates the structure of a feature function. Each digit represents port class information and each block of 587 digits represents protocol class information. For example; 1st digit stands for the TCP protocol class and first port class, which covers the port numbers zero and nine. Similarly; 587th digit stands for the TCP-SYN protocol class and first port class, which covers the port numbers zero and nine.

The feature functions are selected from a set of candidate feature functions. The set of the candidate feature functions requires to be constructed depending on a physical meaning, so that model distribution rapidly converges to the empirical distribution in feature selection step. In this work, candidate feature functions set is built by three set of indicator functions. The first set of indicator functions checks the packet's protocol information, the second set of indicator functions checks the packet's destination port number, and the third set checks both the packet's protocol information and the destination port number.

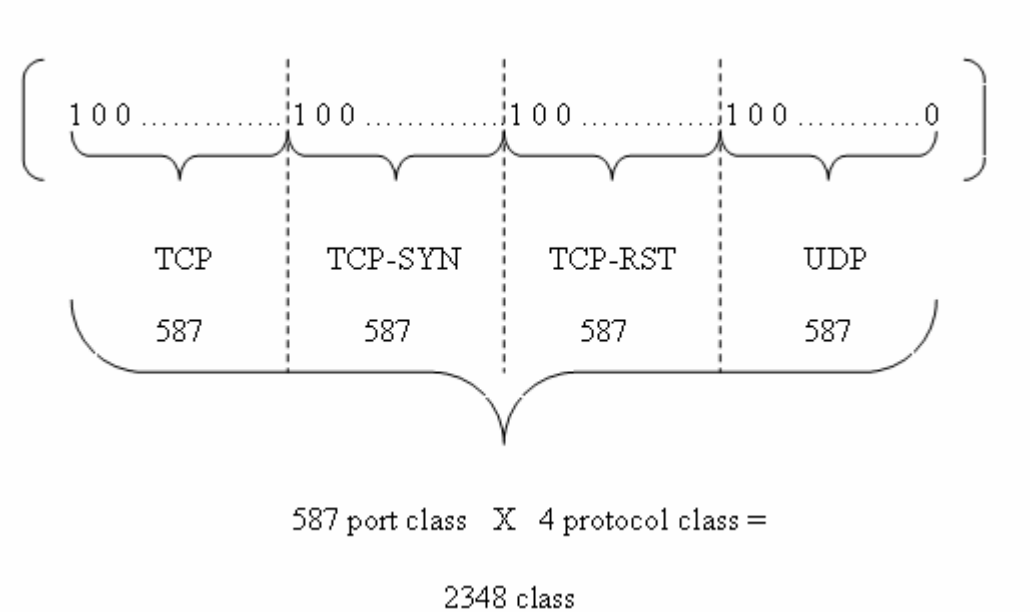

Figure 3.2. Structure of a feature function

For $w \in \Omega$, the first set includes four feature functions exploring the protocol information of the packets in w. If the packets in w satisfy the corresponding protocol information, $f(w)$ is equal to one and zero otherwise. The feature function that checks whether the packets in w are TCP packets can be expressed as follows:

$$f_{TCP}(w) = \begin{cases} 1, & \textit{packets in } w \textit{ are TCP packets}; \\ 0, & \textit{otherwise}. \end{cases} \tag{3.22}$$

The second set consists of 587 feature functions that check the destination port number of the packets in w. For instance, the feature function that checks whether the packets in w are targeted at ports between 20 and 29:

$$f_{[20,29]}(w) = \begin{cases} 1, & \textit{packets in } w \textit{ is ports between } 20-29 \\ 0, & \textit{otherwise}. \end{cases} \tag{3.23}$$

The third set, involving 4*587=2348 feature functions, explore both the protocol information and the destination port number of the packets in w. For instance, the feature

function that checks whether the packets in w are UDP packets and the destination port number of the packets fall between 20 and 29 is as follows:

$$f_{UDP\ X\ [20,29]}(w) = \begin{cases} 1, & packets\ in\ w\ are\ UDP\ packets \\ & and\ is\ ports\ between\ 20-29 \\ 0, & otherwise\ . \end{cases} \tag{3.24}$$

Totally; there are 4 + 587 + 2348 = 2939 candidate feature functions. Each candidate feature function can be considered as a row of the candidate features matrix function matrix with the dimension [2939 X 2348].

3.5. Detection of DoS Attacks with M-Hypothesis Testing

Given that M-1 models reflecting natures of M-1 distinct type attacks and one model representing the nature of attack-free network traffic, aim is to decide which of these M models most probably satisfies the characteristics of the real-time network traffic.

Suppose that K observations are performed, meaning that K packets are captured to make a decision. Let W be the set of K packets captured at real time

$$W = \{w_1, w_2, ...w_K\} \tag{3.25}$$

In this case, W defines our observation space and K is the size of the observation space.

M different models signify M different hypothesis. Each hypothesis is characterized by selected feature functions, parameters and constant Z of each model. In this manner, M hypothesis can be defined as follows:

$$\begin{aligned} &H_o: \quad \forall i, w_i \sim (f_j^0, \lambda_j^0, Z^0);\ i = 1,...K;\ j = 1,...N^0 \\ &H_1: \quad \forall i, w_i \sim (f_j^1, \lambda_j^1, Z^1);\ i = 1,...K;\ j = 1,...N^1 \\ &\text{M} \\ &H_{M-1}: \quad \forall i, w_i \sim (f_j^{M-1}, \lambda_j^{M-1}, Z^{M-1});\ i = 1,...K;\ j = 1,...N^{M-1} \end{aligned} \tag{3.26}$$

Here $N = \{N^0, N^1, ...N^{M-1}\}$ gives the number of selected feature functions for each model and observations w_i are assumed to be independent. Given that one of the M-1 hypotheses is chosen, joint probability of the observation space $W = \{w_1, w_2, ...w_K\}$ is computed and this is repeated for each hypothesis. For example, given that H_0 is chosen, joint probability of $W = \{w_1, w_2, ...w_K\}$ can be calculated as follows:

$$\begin{aligned} P(w_1, w_2, ...w_K / H_0) &= \prod_{i=1}^{K} P(w_i / H_0) \\ &= \prod_{i=1}^{K} \frac{1}{Z^0} \exp(\sum_{j=1}^{N^0} \lambda_j^0 f_j^0(w_i)) \\ &= (Z^0)^{-K} \exp(\sum_{i=1}^{K} \sum_{j=1}^{N^0} \lambda_j^0 f_j^0(w_i)) \end{aligned} \tag{3.27}$$

This calculation is performed for all hypotheses. For simplicity, let express all these joint probabilities as $P(W_1^K / Hq) \stackrel{\Delta}{=} P_q(W_1^K)$, where $q = 1, 2, ..., M-1$. Obviously, joint probability $P_q(W_1^K)$ will be in the similar form in equation 3.27 for all values of q. At the decision step, hypothesis q with highest joint probability $P_q(W_1^K)$ is determined over the current observation space. However; using logarithm of $P_q(W_1^K)$ is more practical during the comparison.

$$\log(P_q(W_1^K)) = -K \log Z^q + \sum_{i=1}^{K} \sum_{j=1}^{N^q} \lambda_j^q f_j^q(w_i) \tag{3.28}$$

Therefore; given an observation space $W = \{w_1, w_2, ...w_k\}$, log $(P_q(W_1^K))$ is calculated for each hypothesis q according to the equation (3.28) and then hypothesis with the highest value of log $(P_q(W_1^K))$ is chosen as reflecting the current traffic.

The performance of this algorithm is dependent on M and K values. M gives the number of models to define network traffic with and without attack. Therefore; M regards to be the database of the attacks that can be detected. The more models are generated for

various attacks in the training part; the more successful is the detection part. Clearly, attacks, for which no model is generated at the training part, can not be identified. However; most of the cases, one of the attack hypotheses is chosen for real time data with an untrained attack type instead of choosing no attack hypothesis. The other important value is K giving the size of the observation space. Generally, this value should be equal to the window size used to generate maximum entropy models. However; some cases it should be different than window size. The detailed analysis can be found in the applications part.

4. SIMULATION ENVIRONMENT

4.1. Introduction

In order to test and evaluate an intrusion detection system, the best environment is an operational network. However, operational networks often are not allowed to be used for research purposes. Hence, intrusion detection systems are generally tested in a simulated environment. The ability to perform accurate evaluation in a simulated environment requires high-quality data that is similar to the traffic on operational networks [4]. For evaluation network intrusion detection systems, the first standard corpora has been collected and distributed by The Information Systems Technology Group (IST) of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS) [32].

In this thesis, DARPA 1999 corpora are exploited to test and evaluate our proposed intrusion detection algorithm. The goal of this section is to provide details about the simulation environment and data of 1999 corpora.

4.2. Simulation Environment of DARPA 1999 Corpora

The main components of DARPA 1999 simulation environment are the network testbed, the background (or normal) traffic, and the attacks. The background traffic is created synthetically according to the traffic model, which is constructed by collecting traffic statistics at Airforce Base and at many other bases. In order to perform attacks in test environments, scripts are collected from the Internet or developed independently and run on the testbed network in conjunction with background traffic [33].

The conceptual view of the evaluation testbed is illustrated in the Figure 4.1. The testbed emulates live network and host traffic on small Air Force base and between that base and the Internet.

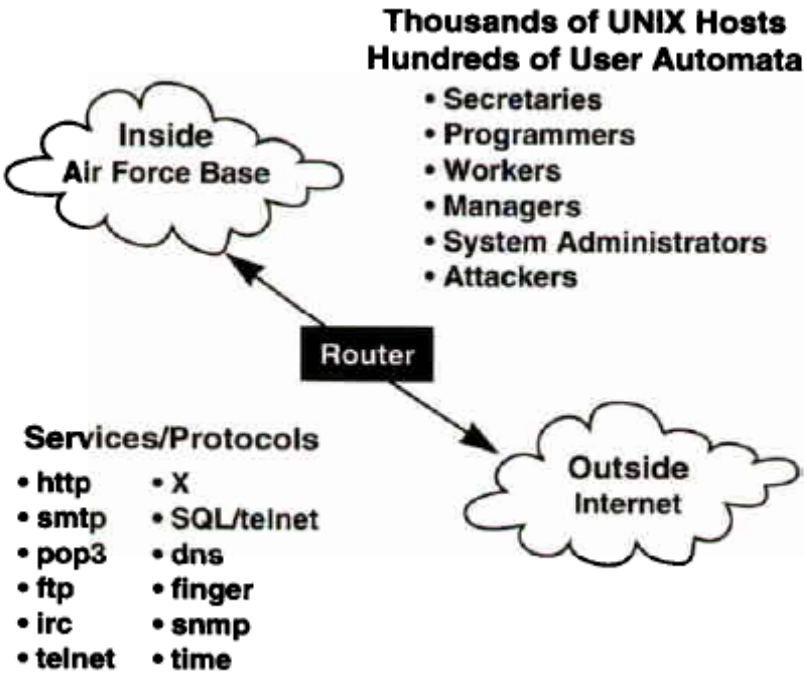

Figure 4.1. Network testbed used in 1999 [33]

Hundreds of programmers, secretaries, managers, and other types of users are simulated by software automata. An expect-based scripting environment is employed to automate background traffic sessions and many of the attacks run on the testbed [33].

Figure 4.2 shows detailed network topology of the simulation environment. The simulation network is composed of two ethernet network segments connected to each other through a router. The router is located at the top center of this figure. Everything to the left of the router in the Figure 4.2 is regarded to be the “outside” of the network and everything to the right of the router is regarded to be the “inside” of the simulation network. The inside network is connected to one interface of the router and consists of all the computers that are part of the internal domain. The computers that imitate the rest of the Internet are connected to the external interface of the router.

There are sixteen computers in the simulation network. The outside of the network contains a traffic generator, a web server, a sniffer/recorder, SNMP monitor and there machines used for attack generation. The inside of the network consists of a background traffic generator, a sniffer/recorder, five victim machines and two machines used for attack

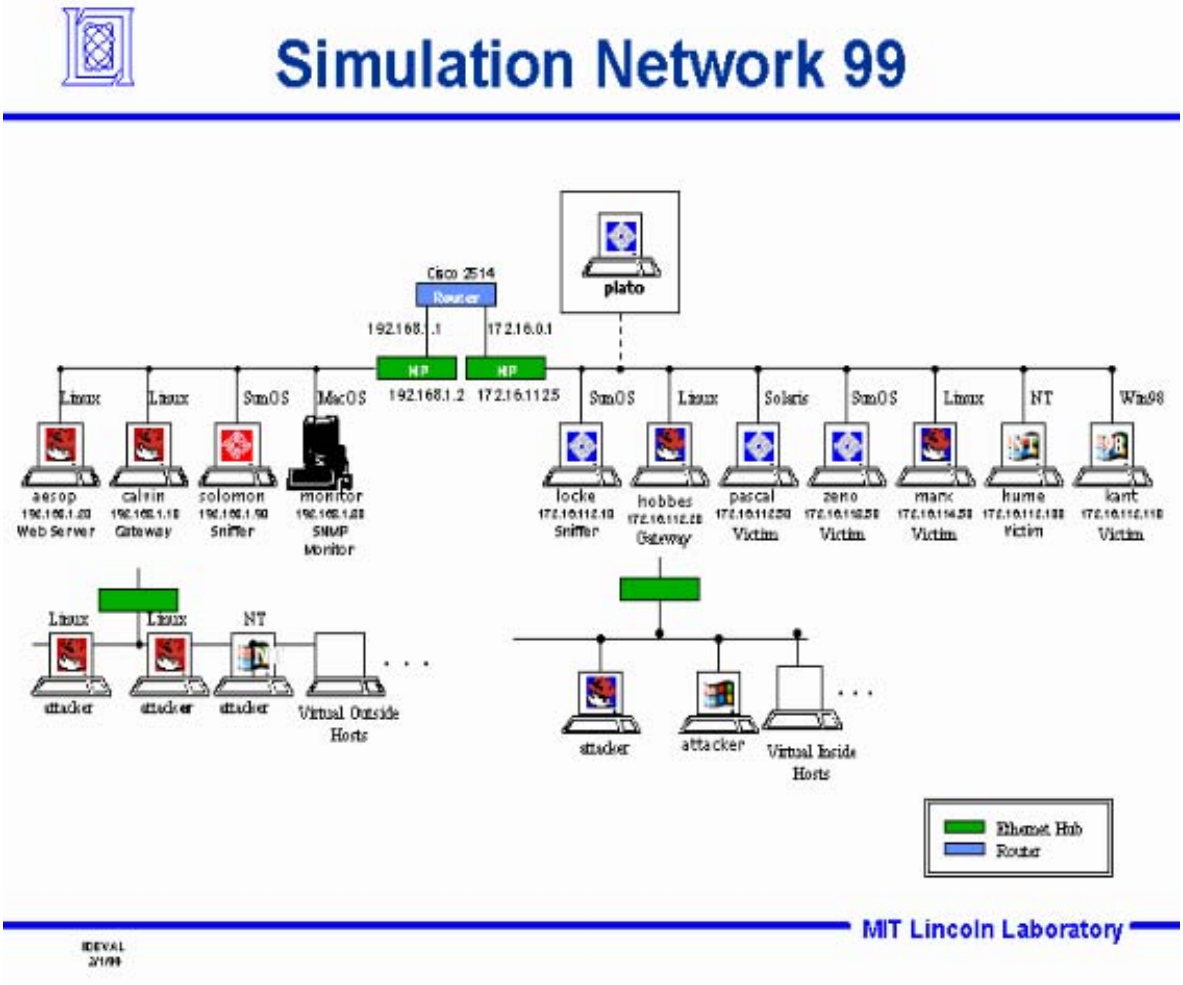

Figure 4.2. Network topology used in 1999 [32]

generation [33]. Although two computers (one inside, one outside) generated all of the background traffic in the simulation, custom software allows this simple network to model the interaction of thousands of clients and servers.

4.3. Simulation Data

For the 1999 evaluation, five weeks' worth of data was generated by using the network testbed. These weeks were five-day weeks and each 22-hour day was simulated in real time [33]. Data of the first and third weeks do not include any attack. However, the data of week 2, week 4 and week 5 contain various attack instances. Table 4.1 summarizes the times, start and end dates for each week of data provided.

Table 4.1. Date and time of simulation data [33]

Week	Description	Start	End
Week 1	Training data, without attacks	3/1/99, 8:00 a.m.	3/6/99, 6:00 a.m.
Week 2	Training data, with attacks	3/8/99, 8:00 a.m.	3/13/99, 6:00 a.m.
Week 3	Training data, without attacks	3/15/99, 8:00 a.m.	3/20/99, 6:00 a.m.
Week 4	Testing data	3/29/99, 8:00 a.m.	4/3/99, 6:00 a.m.
Week 5	Testing data	4/5/99, 8:00 a.m.	4/10/99, 6:00 a.m.

As a result of the simulation of each day, following files were provided [32]:

- Outside sniffing data (Tcpdump format)
- Inside sniffing data (Tcpdump format)
- BSM audit data (From pascal)
- NT audit data (From hume)
- Long listings of directory trees (From pascal, marx, zeno, and hume)
- Dumps of selected directories (From pascal, marx, zeno, and hume)
- A Report of file system inode information (From pascal)

Tcpdump files of inside sniffing and outside sniffing data are used as an input to train model and to test our intrusion detection. Actually, our algorithm requires only timing, protocol, port and flag information, so it captures only required information from the tcpdump file and process it.

4.3.1. Simulation Data Statistics

As explained before; our intrusion detection method requires experimental data to construct models for benign and malignant network traffic. Especially; generated model for the attack-free traffic serves as a guide to detect attacks. Therefore; the experimental data used for generating benign traffic model should reflect the behavior of the normal network traffic. Since simulation data is employed as an input to our method; it is really important

to ensure that the statistics of generated data coincide with the statistics of an operational network. In this manner, this section presents some charts and graphs to illustrate some of the features of synthetic background traffic.

In order to construct model for benign traffic, two hours data on Monday at week one is exploited. Figure 4.3 shows TCP, UDP and ICMP packet percentage in traffic data, which is used as training data for no attack case. TCP packets dominate the traffic with 70 percent of the packets. Almost 30 percent of packets are UDP packets. ICMP traffic forms a very small part of the network traffic. These packet percentages seem to be reasonable when compared the real packet percentages of an operational network given in [34].

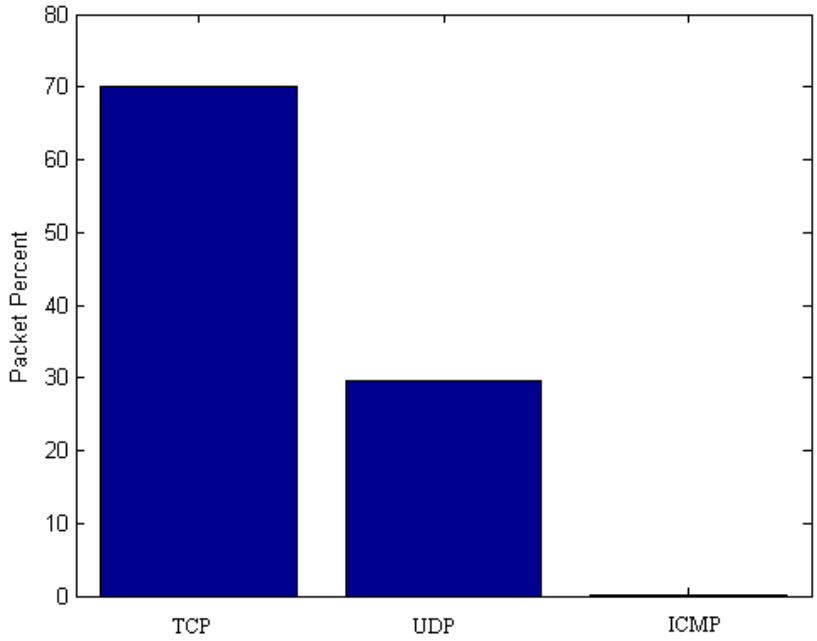

Figure 4.3. TCP, UDP and ICMP packet percentage at week one on Monday between 08.00 am-10.00 am

Figure 4.4 illustrates the packet percentages of some TCP or UDP services on Monday between 08.00 am and 10.00 am. HTTP (Hypertext Transfer Protocol) is the basis for the World Wide Web (WWW) and it is clear that web services dominate the traffic. The second most common application is DNS (Domain Name System), the on-line distributed database system used to map human-readable machine names into IP addresses. Percentages of these services are consistent with values of an operational network given in [34].

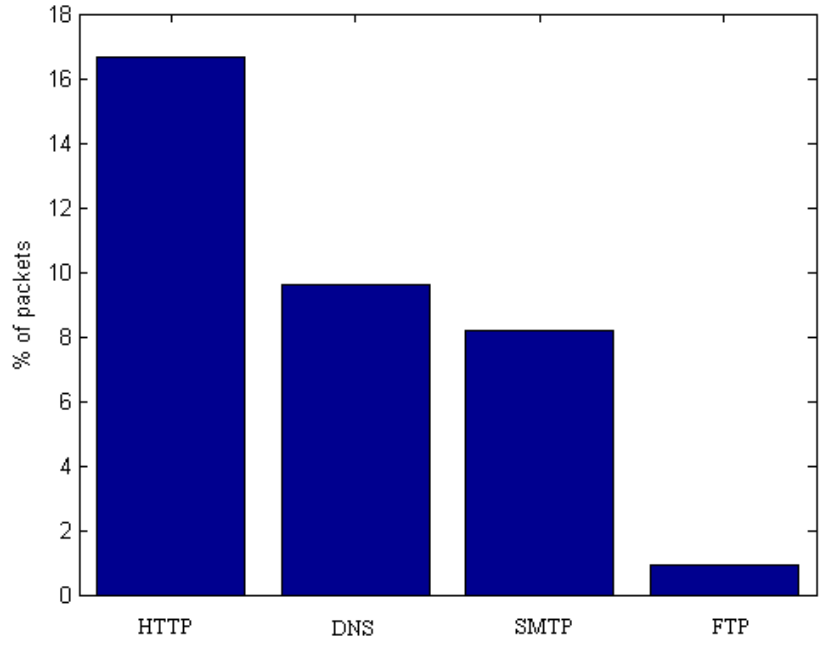

Figure 4.4. Packet percentages of some TCP or UDP services at week one on Monday between 08.00 am-10.00 am

Consequently, these statistics of the two hours data captured on Monday (at week one) overlap with the statistics of a real network. So, using these data to model the benign traffic can be considered to be realistic.

4.3.2. Target Attacks in Simulation Data

Our intrusion detection method performs the packet classification according to the protocol information and destination port information in packet header. Only TCP, TCP SYN, TCP RST and UDP packets are considered, while dividing packets into classes according their protocol information. Some other IP protocols such as ICMP are ignored by our classification process. Therefore; attacks imposing on the services of protocols like ICMP can not be detected. Similarly, attacks that carried in packet payload can not be identified, since our method only deals with packet header. Additionally, design intent of our method is based on sensing changes from normal traffic distributions. So, the attacks that do not affect the normal traffic behavior can not be detected, as well. Generally; DoS attacks make traffic distribution differ from the normal distribution. Hence, our target group is DoS attacks. However, there are some DoS attacks that can not be detected, as

well. For example, land is a denial of service performed by sending a UDP packet with the same source and destination port. Unfortunately, land can not be identified, since source port information is not considered at classification step. The list of the DoS attacks in 1999 DARPA evaluation is given in the Table 4.2.

Table 4.2. Denial of service attacks in week two, week four and week five

Denial of Service Attacks	Service	Mechanism	Effect	Number of Occurrence
Ping of Death	ICMP	Bug	Crash/Reboot	6
Smurf	ICMP	Abuse	Network Slowdown	5
Apache2	HTTP	Abuse	Crash http	3
Back	HTTP	Abuse/Bug	Slow server response	5
Neptune	Any TCP	Abuse	Deny service on one or more ports for minutes	6
Land	N/A	Bug	Freeze machine	4
Mailbomb	SMTP	Abuse	Annoyance	6
Syslogd	SYSLOG	Bug	Kill Syslogd	4
Udpstorm	Echo/Chargen	Abuse	Network Slowdown	2
Dosnuke	NetBIOS	Abuse	Crash machine	4
Sshprocesstable	SSH	Abuse	Deny new processes	1
Tcpreset	Any TCP	Bug	Reset TCP connections	3

As mentioned before, training data with each target attack is required to model the behavior of each attack. Although data collected throughout week 2 are issued as training

data by DARPA evaluation [32], it is nearly impossible to use these data to train our method, since the duration of attack instances are not remarked for these data. Our method takes packets in the vicinity of each target attack as an input. If the duration of attacks is uncertain, proper inputs can not be captured. As a result of this, the model that would represent real behavior of the attack can not be constructed. Therefore; data collected throughout week two are employed as testing data.

Data collected throughout week four and week five are used not only as testing data but also as training data. All attack instances do not take place on a specific day of a week, whereas attack instances are dispersed among all data of week four and week five.

Table 4.3. Training data used for each attack

Attack Name	Training data	Number of instances used for testing
Apache2	Week 5 Monday 10.29.22 am – 10.35.22 am	2
Back	Week 5 Tuesday 11.31.21 am – 11.37.21 am	4
Dosnuke	Week 5 Monday 11.45.27 am – 11.51.27 am	3
Neptune	Week 5 Monday 06.04.04 pm – 06.10.04 pm	5
Mailbomb	Week 4 Friday 00.32.17 pm – 00.38.17 pm	3
Syslogd	Week 5 Friday 02.56.30 pm – 03.02.30 pm	3
Tcpreset	Week 5 Tuesday 08.11.27 am – 08.20.27 am	2
Udpstrom	Week 5 Monday 08.00.27 pm – 08.06.27 pm	1

Moreover, number of occurrence of each attack is limited. Therefore, data portions in the vicinity of each attack are exploited as training data and the remaining part as testing data. Table 4.3 summarizes the data portions that used as training data for each attack. While assigning these data portions as training data, it is important to ensure that there is

no other attack in the vicinity except the relevant attack. In this manner, our training set is determined, so that each training data includes only relevant attack.

Note that some attacks such as ping of dead, smurf, land and sshprocesstable in Table 4.2 are ignored and for them no training data are assigned. Ping of dead and smurf attacks affect ICMP services and therefore they are not in our target group. As mentioned before, attacks such as land can not be detected by our method, since they do not cause big changes in normal traffic distribution. Sshprocesstable attack is disregarded, as there is only one instance for this attack at whole 1999 evaluation data. So, there is not enough data for training and testing for this attack.

In conclusion, DARPA 1999 evaluation data is suitable to constitute benign traffic model. However, it is disqualified for modeling traffic with various attacks, as it is required to separate partially testing and training data from data bulk.

5. APPLICATIONS

5.1. Introduction

In order to detect DoS attacks, our method proposes modeling both benign and malignant traffic behavior based on the maximum entropy framework. In the methodology part, the theoretical basis of our approach has been explained. This part examines how to realize our proposed approach.

At the preprocessing part, packets in the training data are classified into 2348 packet classes, which form the domain of the probability space. Then the empirical distribution of these packet classes is calculated. Given M-1 different type attack, M-1 different empirical distributions for each attack type and one empirical distribution for no attack case are calculated by using training data. The inputs of maximum entropy methods are empirical distributions, candidate feature function set and initial parameters. According to these inputs, maximum entropy method selects feature functions, estimates parameters and then constructs a model for each empirical distribution. In this manner, maximum entropy method produces M models as outputs. M models and the empirical distribution of the current packet block are the inputs of the detection part. Let K be the size of the observation space in test data. For each packet block in size K, hypothesis testing is performed over M models. As a result one of the hypotheses is chosen. Figure 5.1 summarizes the block diagram of our proposed method.

Note that same candidate feature set and same initial parameters are used while building the models. Only the empirical distributions are different. So, at the end of the training part, different feature functions are selected and different parameters are estimated based on the empirical distributions. It is important to underline that each model differs from other models depending on its selected feature functions, estimated parameters and constant Z. In fact, each hypothesis is specified by the selected features, estimated parameters and constant Z, as well.

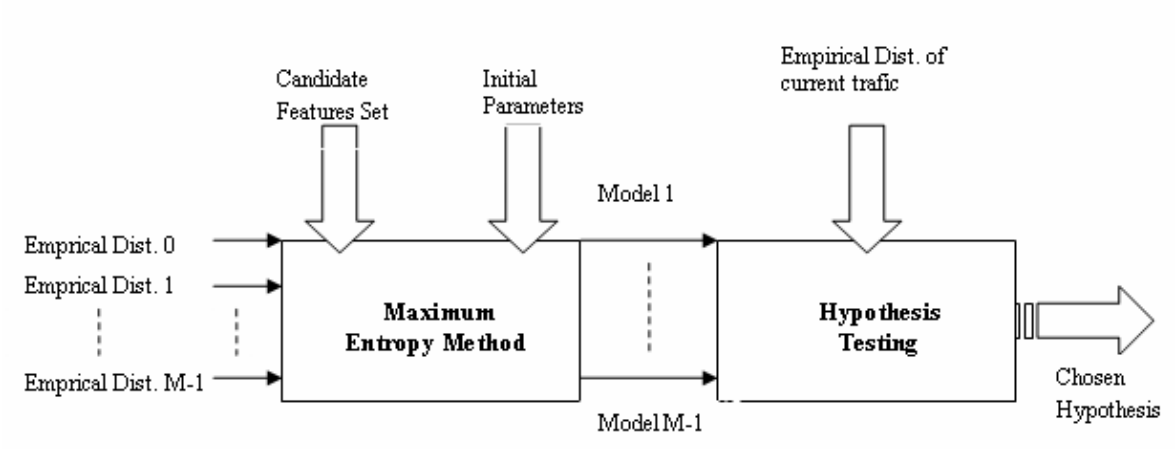

Figure 5.1. Block diagram of the proposed method

5.2. Maximum Entropy Based Modeling

Constructing the empirical distribution is the foundation of the modeling process. The main concern is how many packets should be used to determine empirical distribution of packet classes. In other words, what is the optimal packet window size for the calculation of empirical distribution? Actually, the size of packet window stands for the size of the observation space, while calculating the empirical distribution. Therefore, it seems like that the model would represent the traffic much more realistically for large numbers of packet window. However, use of a packet window, that includes many packets captured consecutively, entails considering the time dimension, as well. For instance, the use of 200000 consecutive packets for the calculation of empirical distribution causes estimation of a model representing approximately two hours traffic. The estimated model is used as a guide at the detection part by comparing the current network traffic against the model. Since model is constructed with 200000 consecutive packets, then the observation space size K at the detection part should be 200000 as well to make this comparison in a reasonable way. However, this stands for capturing 200000 consecutive packets in real time and waiting approximately for two hours to decide. Instead of such a large packet window, a small one covering 10000 packets can be exploited. 10000 packets can be collected in approximately two or three minutes. However, especially for benign traffic, it is too hard to capture all features of traffic with 10000 packets. In order to eliminate this problem and train our method with more packets, an average empirical distribution can be used as an input to our method by taking the average of empirical distributions, each of

which generated with 10000 packets. In the following subsections, calculation of empirical distribution and estimation of model is given in details for both benign and malignant traffic.

5.2.1. Modeling Benign Traffic

First step of the modeling process is the calculation of the empirical distribution. In the simulation network part, it is proved that the statistics of the two hours data collected on Monday in week one is convenient to represent properties of the benign traffic. Therefore, these data are used as training data and it consist of approximately 200000 packets. Since the goal is not modeling two hours traffic, these data are employed fragmentally while calculating the empirical distribution. Figure 5.2 illustrates the empirical distribution of the packet classes calculated by using first 10000 packets in the training data. Here x axis shows the packet classes, to which each of the 10000 packets is assigned at the preprocessing part. Similarly, Figure 5.3 demonstrates the empirical distribution of the packet classes calculated by using fifth set of 10000 packets in the training data

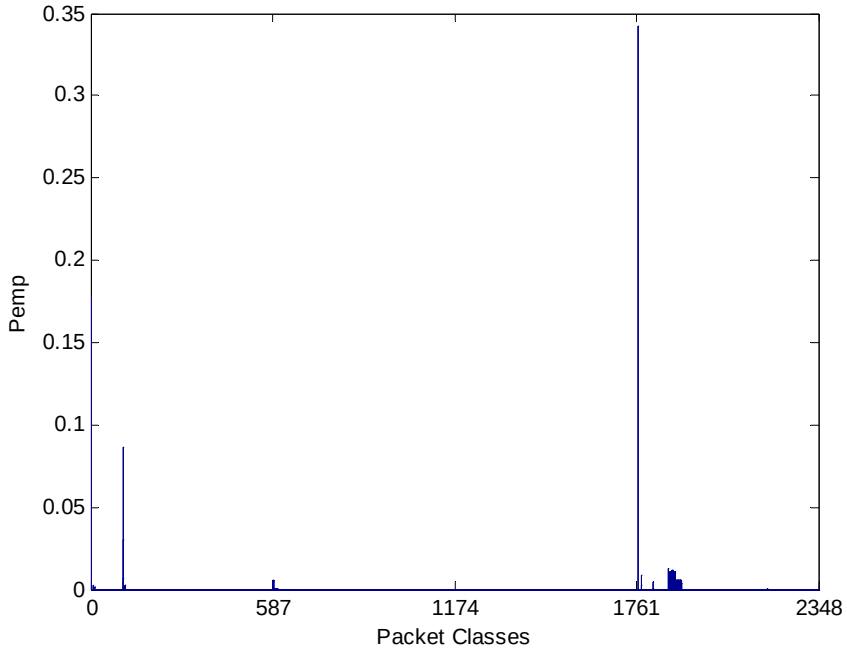

Figure 5.2. Empirical distribution of the packet classes calculated by using first 10000 packets in the training data

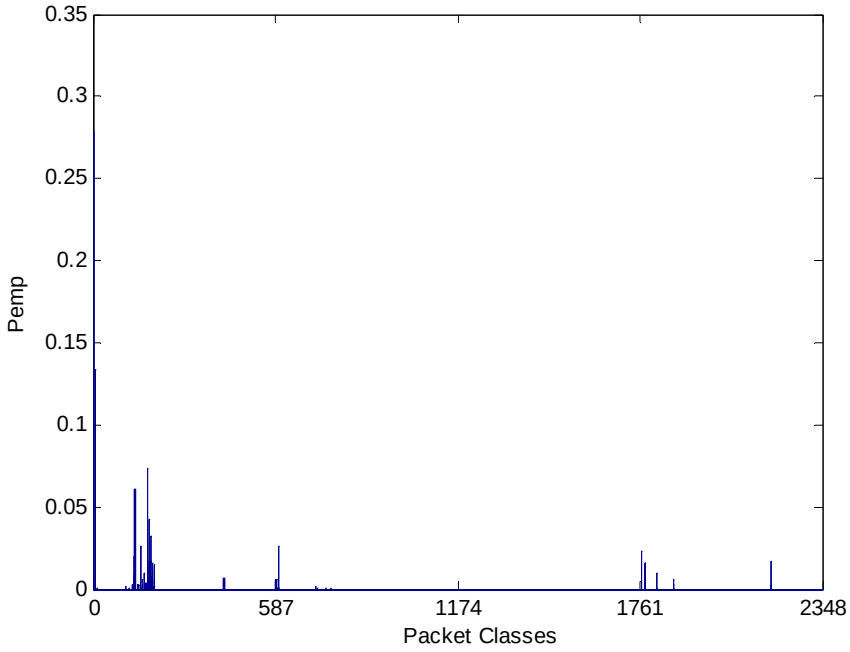

Figure 5.3. Empirical distribution of the packet classes calculated by using fifth set of 10000 packets in training data

It is clear that these empirical distributions are very different, but both are a part of the normal traffic and both provide information about the behavior of attack free traffic. In order to represent all properties of short term traffic, there are two solutions. More than one model for attack free traffic can be constructed from each of the empirical distributions calculated by using packets captured in a short period of time or these empirical distributions can be gathered in a reasonable way. In this work, we have preferred gathering the empirical distributions calculated by using packets in a short period of time and generating an ultimate empirical distribution from them.

In this manner, two hours training data are fragmented into packet sets including 10000 packets. Nineteen packet blocks are obtained and by using these 19 packet blocks, 19 empirical distributions are calculated. Then these empirical distributions are gathered by taking the average of them. So, an ultimate average empirical distribution is procured from them. Figure 5.4 introduces the average empirical distribution obtained from two hours data collected on Monday at the first week. Clearly, TCP traffic constitutes the majority of the normal traffic. The dominant classes are three, nine, 596 and 1767. Class three stands for the TCP class covering the port numbers between 20 and 29. Ports 21 and 22 are

assigned to FTP service and port 23 is assigned to TELNET service. Class nine denotes the TCP class having the port number 80 and use of port 80 implies the HTTP traffic. Class 596 expresses the TCP-SYN class with the port number 80 and it represents HTTP traffic. Class 1767 stands for the UDP class covering the port numbers between 50 and 59. Port 53 is assigned to the DNS service. Thus, these results are compatible with the properties of the attack free network traffic.

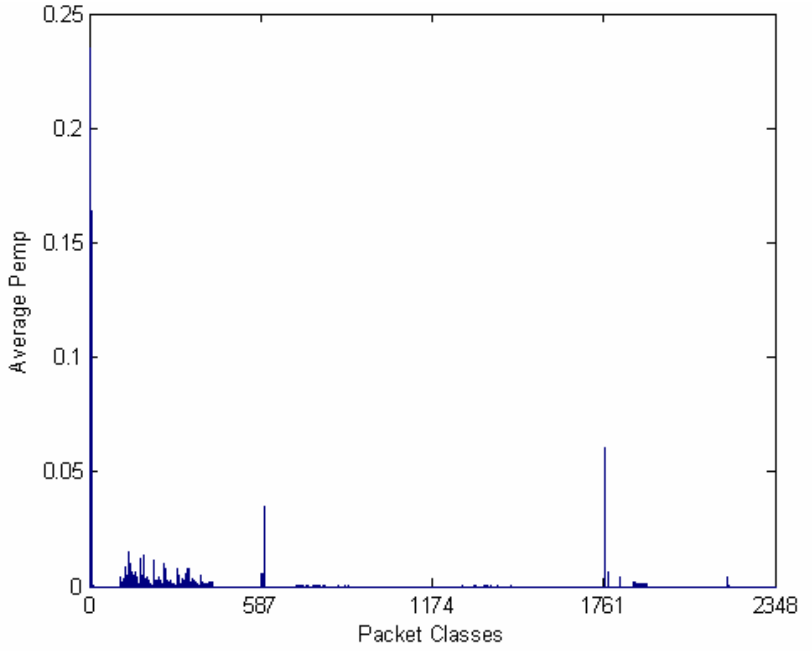

Figure 5.4. Average empirical distribution

By using this average empirical distribution, a model is built based on the maximum entropy method. Maximum entropy method selects the most important feature function from the set of candidate feature functions and then estimates proper parameters for the selected features at each iteration step. This process is repeated until the stopping criterion is met. The stopping criterion is that the KL divergence of the baseline distribution with respect to the empirical distribution is less than 0,05. At the end of this process, 248 feature functions have been selected from the set of the candidate feature functions. Table 5.1 shows the first 5 feature functions selected by maximum entropy method and Figure 5.5 shows the structure of the Top fivefeature functions in the max of the feature functions. All of the remaining feature functions have the similar form and they are stored in the same matrix, as well.

Table 5.1. Top fivefeature functions selected for no attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 20-29
2	Feature Function representing TCP packets with the port number 80
3	Feature Function representing UDP packets with the port number 50-59
4	Feature Function representing TCP packets
5	Feature Function representing TCP-SYN packets with the port number 80

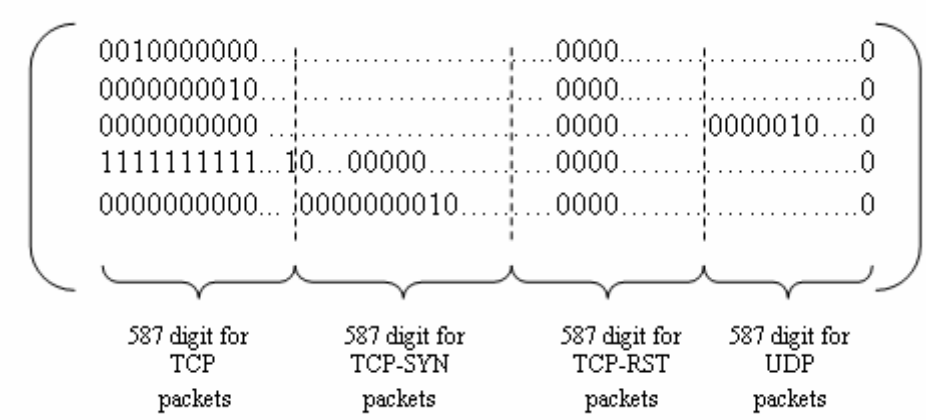

Figure 5.5. Structure of the top five feature functions in the feature matrix

In this manner, the model for the benign traffic is identified with 248 selected feature functions, parameters of the feature functions and the constant Z. This model will be in the form given in the equation (5.1) and Figure 5.6 shows this model.

$$P(w) = \frac{1}{Z^0}\exp(\sum_{j=1}^{248}\lambda_j^0 f_j^0(w)), \text{ where } Z^0 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{248}\lambda_j^0 f_j^0(w_i)) \qquad (5.1)$$

As mentioned before, this model is built based on the average empirical distribution given in the Figure 5.4. By comparing Figure 5.6 with Figure 5.4, it is obvious that they are very similar. So, our model satisfies the constraints provided by empirical distribution. By

looking the figure of the average empirical distribution, the probabilities of the packet classes, for which no packets are observed in the training data, are zero. At the constructed model, those probabilities of unobserved packet classes are not assigned to zero, whereas a very small value is assigned to all of them. Since same value assigned to them, estimated model for these unobserved classes is uniform. By zooming in the Figure 5.6, Figure 5.7 is obtained. This figure shows the probability distribution for some of the unobserved classes in the training data.

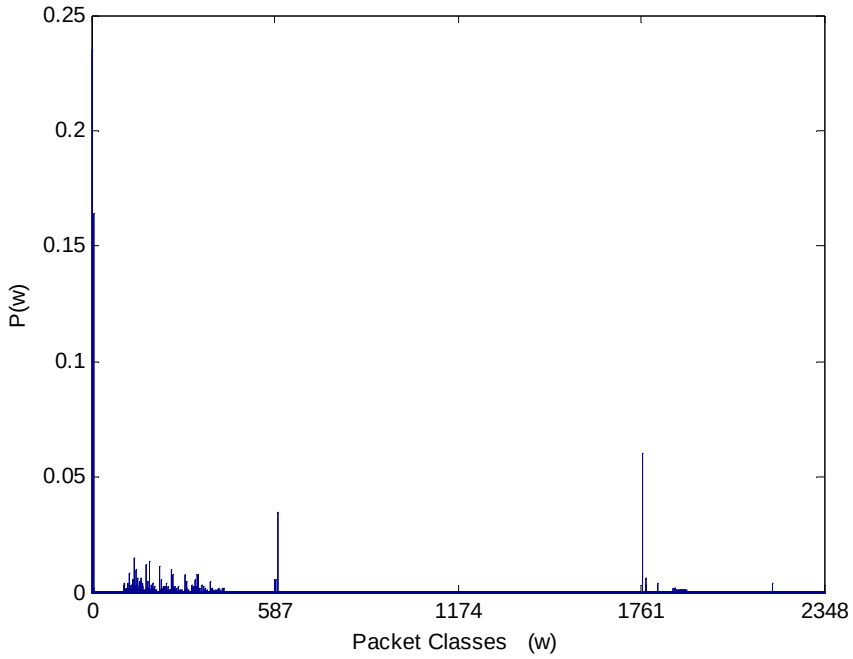

Figure 5.6. Constructed model based on maximum entropy principle for the benign traffic

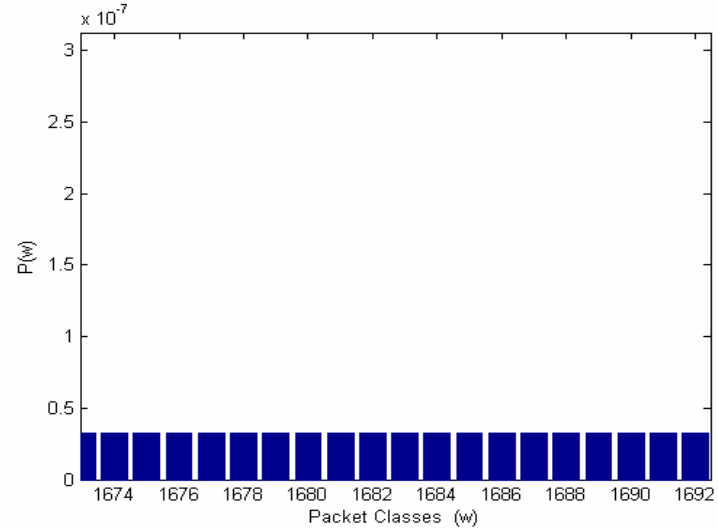

Figure 5.7. Estimated probability distribution for some of the unobserved packet classes

Consequently, constructed model covers the constraints provided by empirical distribution, but it does not make any assumptions. Thus, our model is the most uniform distribution among all the distributions satisfying the given constraints.

5.2.2. Modeling Apache2 Attack

Similar procedure that is performed for benign traffic will be carried out for each attack types, as well. First step is specifying the training data for each attack type and then the empirical distribution will be calculated by using this training data. While choosing the training data, it should be noticed that there is no other attack instance in the vicinity except the target attack.

For apache2 attack, data that is captured between 10.29.22 am and 10.35.22 am on Monday in week five is employed as training data. The real duration of the attack is approximately 17 minutes. In order to ensure that there is no other attack in the vicinity and to ensure that each attack type is trained with data collected at the same period of time, six minutes of attack duration is used. For each data block of 10000 packets in this time interval, packet classification is performed and then empirical distribution of packet classes is calculated. Nine blocks of 10000 packets is identified in the training data of apache2 attack. Thus, nine empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. Figure 5.8 demonstrates the average empirical distribution for apache2 attack. In this figure x axis shows the packet classes as usual, but it is marked based on the logarithmic scale in order to highlight class nine with the biggest pmf value. Class nine stands for the TCP packets with the port number 80 and since apache2 attack abuse http service (through port 80); a big pmf value for the class nine is expected during the apache2 attack.

Maximum entropy method is applied to this empirical distribution in order to estimate a baseline distribution for apache2 attack. The stopping criterion of the maximum entropy algorithm is that the KL divergence of the baseline distribution with respect to the empirical distribution is less than 0,05. As a result, 97 feature functions have been selected from the set of the candidate feature functions. Clearly, less feature functions than no attack case have been selected for modeling the behavior of apache2 attack, because there

were fewer constraints as can be seen in the Figure 5.8. Table 5.2 shows the first five feature functions selected by maximum entropy method and first selected one is the feature function representing TCP packets with the port number 80 as expected.

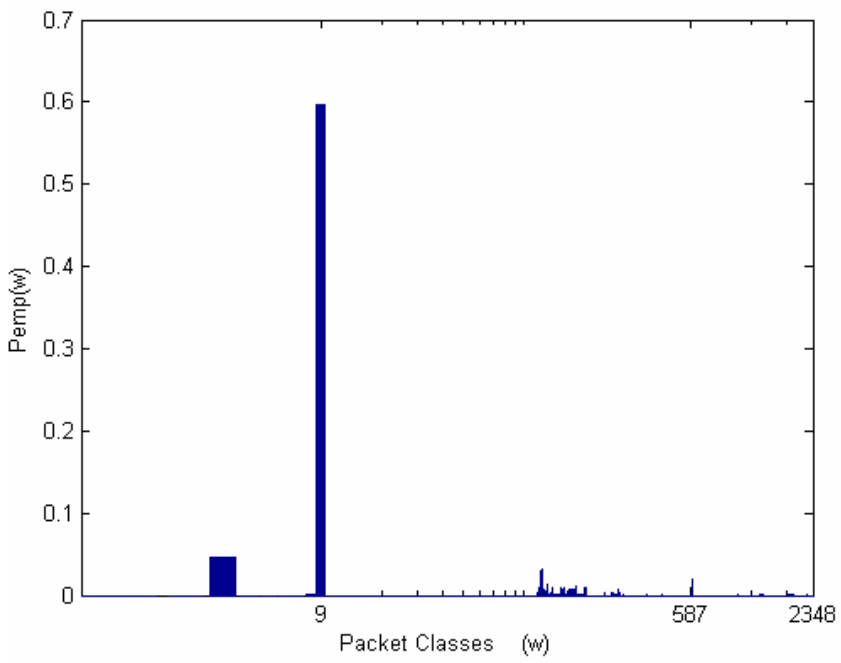

Figure 5.8. Average empirical distribution of packet classes for the training data of apache2 attack

Table 5.2. Top fivefeature functions selected for apache2 attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 80
2	Feature Function representing TCP packets
3	Feature Function representing TCP packets with the port number 20-29
4	Feature Function representing TCP-SYN packets with the port number 80
5	Feature Function representing TCP packets with the port 1524-1624

The model for the apache2 attack is specified with 97 selected feature functions, parameters of the feature functions and the constant Z. This model will be in the form given in the equation (5.2).

$$P(w) = \frac{1}{Z^1}\exp(\sum_{j=1}^{97}\lambda_j^1 f_j^1(w)), \text{ where } Z^1 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{97}\lambda_j^1 f_j^1(w_i)) \quad (5.2)$$

Figure 5.9 illustrates this model and x axis is marked based on logarithmic scale in this Figure, as well and it should be noted that probabilities of unobserved packet classes are not assigned to zero.

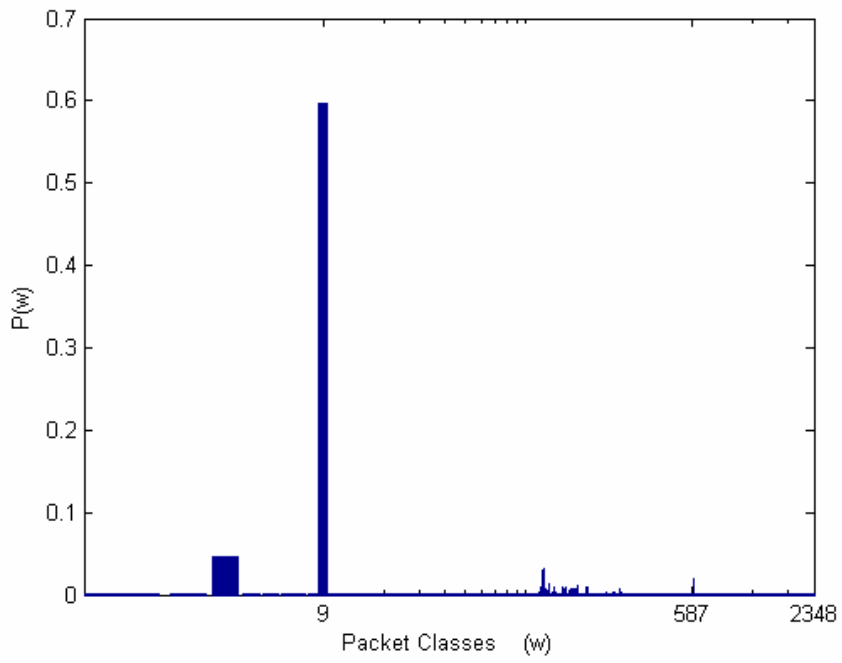

Figure 5.9. Constructed model based on maximum entropy principle for apache2 attack

5.2.3. Modeling Back Attack

For back attack, data that is collected between 11.31.21 am and 11.37.21 am on Tuesday in week five is used as training data. The real duration of the attack is approximately 20 minutes. In order to ensure that there is no other attack in the vicinity and to ensure that each attack type is trained with data collected at the same period of time, six minutes of attack duration is used. For each data block of 10000 packets in this time interval, packet classification is performed and then empirical distribution of packet classes

is calculated. Thirty-three blocks of 10000 packets are determined in the training data of back attack. Thus, thirty-three empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. Figure 5.10 shows the average empirical distribution for back attack. In this figure, x-axis shows packet classes in logarithmic scale and the classes with high probabilities are class nine and class 116. The increase in the traffic of the packet class 116 is indirectly related with the back attack. However, the increase in the traffic of the packet class nine is directly due to the back attack.

Actually, back attack is a payload attack. This attack performs a malformed web request to HTTP port with the payload, "\Get //// ..." followed by 6000-7000 slashes [10]. If the number of such requests is not numerous during the back attack, the distribution of the traffic will not represent this anomaly, since it is carried on the payload and does not have any effect on the header. If the number of such malformed requests is high, then an increase in the pmf value of class nine is expected. There are few back attack instances in the DARPA 1999 evaluation. Only one attack instance affects the traffic distribution and this is the one that is used to train our model.

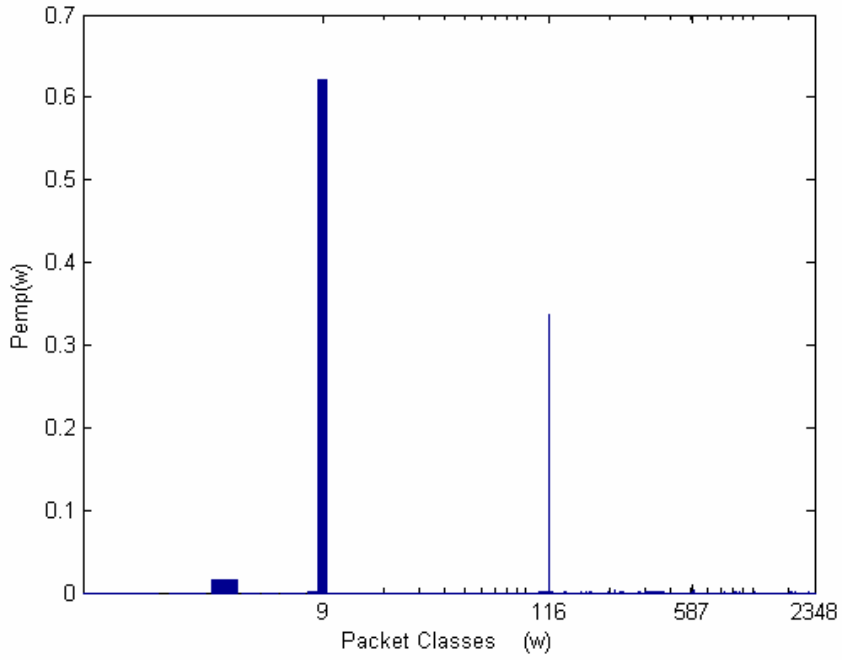

Figure 5.10. Average empirical distribution of packet classes for the training data of back attack

By using this empirical distribution, a model given in figure 5.11 is constructed based on the maximum entropy method. The stopping criterion of the maximum entropy algorithm is that the KL divergence of the baseline distribution with respect to the empirical distribution is less than 0,05. As a result, eight feature functions have been selected to estimate a model for back attack. The number of the feature functions is small, since there are a few properties to model in the empirical distribution. Table 5.3 presents the first 5 feature functions selected by maximum entropy method. The constructed model for back attack can be expressed as follows:

$$P(w) = \frac{1}{Z^2}\exp(\sum_{j=1}^{8}\lambda_j^2 f_j^2(w)) \text{ , where } Z^2 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{8}\lambda_j^2 f_j^2(w_i)) \quad (5.3)$$

The figure 5.11 illustrates this constructed model for back attack. Here, x-axis shows packet classes in logarithmic scale in order to emphasize big pmf value of class nine. Clearly, constructed model satisfy the observed constraints. Moreover, it is noticeable in the Figure 5.11 that a non-zero value is assigned to the probabilities of unobserved classes, such as class one, two and etc.

Table 5.3. Top five feature functions selected for back attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 80
2	Feature Function representing TCP packets with the port 2124-2223
3	Feature Function representing TCP packets
4	Feature Function representing TCP packets with the port number 20-29
5	Feature Function representing TCP-SYN packets with the port number 80

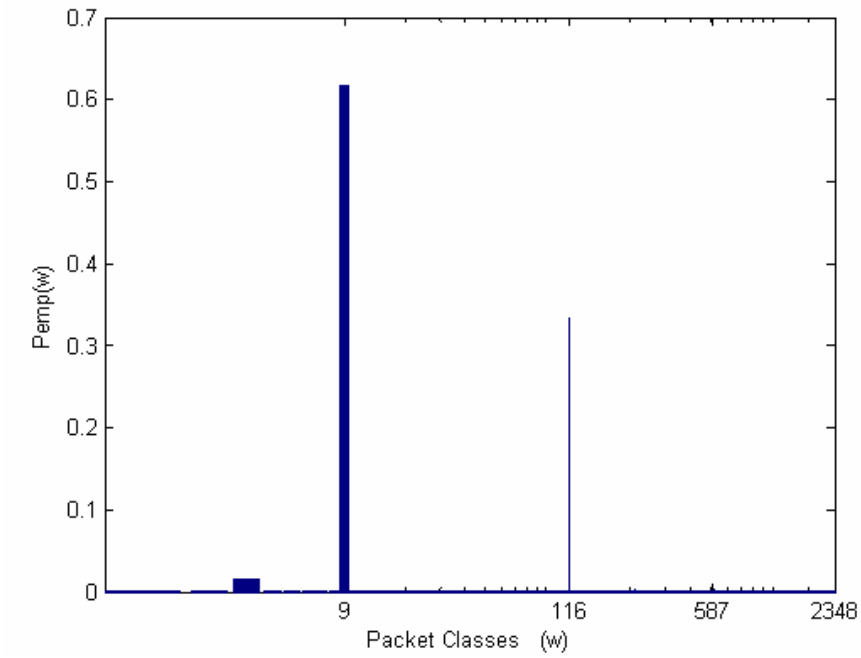

Figure 5.11. Constructed model based on maximum entropy principle for back attack

5.2.4. Modeling Dosnuke Attack

The data collected between 11.45.27 am and 11.51.27 am on Monday at week five is used as training data of dosnuke attack. The real duration of the attack was approximately 16 minutes. Only one block of 10000 packets is identified in the training data of dosnuke attack. Thus, there is no need to calculate the average empirical distribution and the empirical distribution of 10000 packets is assigned as the ultimate empirical distribution. Figure 5.12 shows the empirical distribution calculated from the training data of dosnuke attack.

Dosnuke sends Out-of-Band data (MSG_OOB) to port 139 (NetBIOS) of victim machine. During the attack instance, a sudden dramatic increase in the number of NetBIOS packets is observed. Port 139 stands for the class 15 according to our classification algorithm, so only class 15 represents dosnuke attack, pmf values of other classes do not give any information about the dosnuke attack. Thus, it is expected that the pmf value of class 15 increases during the attack. However, looking to the Figure 5.12, it is not much in evidence. By zooming in the figure above, Figure 5.13 is obtained. Figure 5.13 presents especially the pmf value of class 15. The pmf value of class 15 is equal to $8x10^{-4}$ during

the dosnuke attack and it is equal to 9,4737x10^{-5} during the normal network traffic. In fact, there is not much increase in the pmf value of the class 15 during the attack.

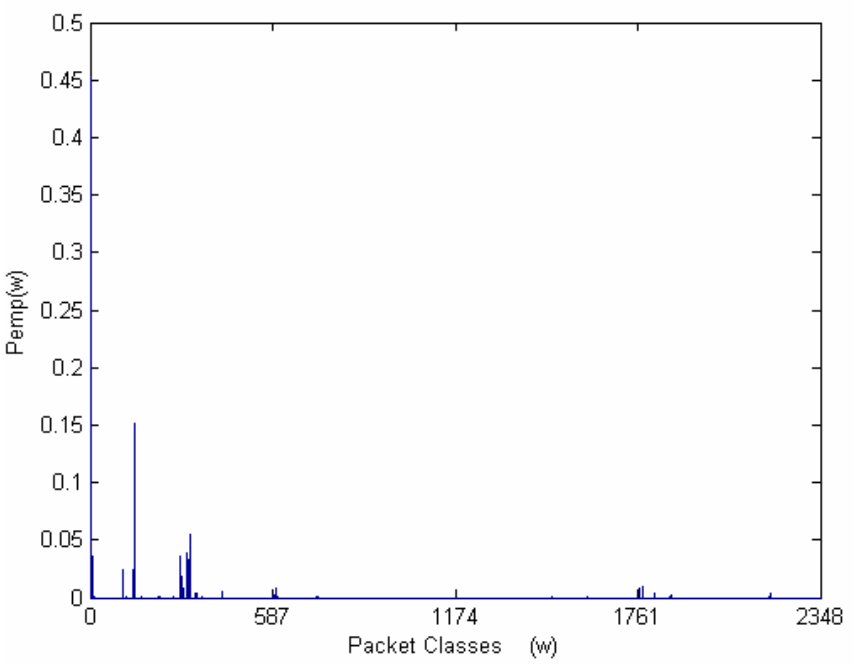

Figure 5.12. Empirical distribution of packet classes for the training data of dosnuke attack

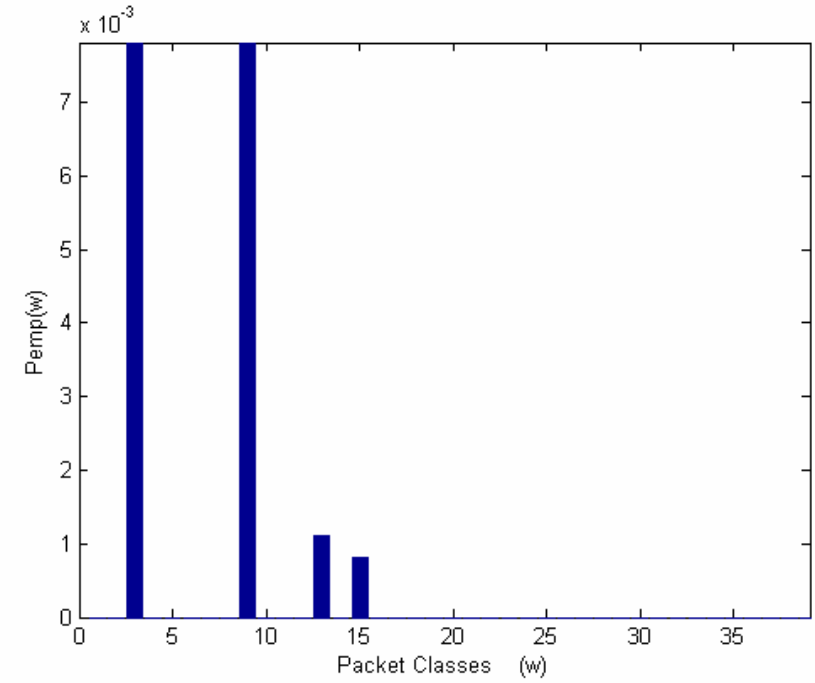

Figure 5.13. PMF value of class 15 during dosnuke attack

Using this empirical distribution, a model is estimated for the dosnuke attack based on the maximum entropy algorithm. The stopping criterion of the maximum entropy algorithm is that the KL divergence between the model and the empirical distribution is less than 0,01. This threshold is set to a smaller value than the threshold of other models, since the component that is representing dosnuke has a small value and it should be ensured that the model for dosnuke attack includes this component, as well.

Maximum entropy process comes out with a model specified by 43 feature functions and its parameters. The Table 5.4 demonstrates the first 5 feature functions selected by maximum entropy method. Unfortunately, the feature function that stands for class 15 is not in the list of first 5 feature functions. Actually, this is expected, since the pmf value of class 15 is very small. The model for dosnuke attack is expressed in equation (5.4) and it is illustrated in Figure 5.14.

$$P(w) = \frac{1}{Z^3}\exp(\sum_{j=1}^{43} \lambda_j^3 f_j^3(w)), \text{ where } Z^3 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{43} \lambda_j^3 f_j^3(w_i)) \qquad (5.4)$$

Table 5.4. Top five feature functions selected for dosnuke attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 20-29
2	Feature Function representing TCP packets with the port 5024-5123
3	Feature Function representing TCP packets
4	Feature Function representing TCP packets with the port 22524-22623
5	Feature Function representing TCP, SYN,RST and UDP packets with the port number 80

Since the dosnuke attack is identified by the activity of port 139 (stands for class 15), Figure 5.14 shows only pmf values of classes in the vicinity of class 15. Furthermore, it is noticeable from this figure that a very small value assigned to the unobserved classes in the

training data, instead of assigning to them zero. Thus, constraints of the training data is satisfied and except these constraints our model for dosnuke attack is mostly uniform.

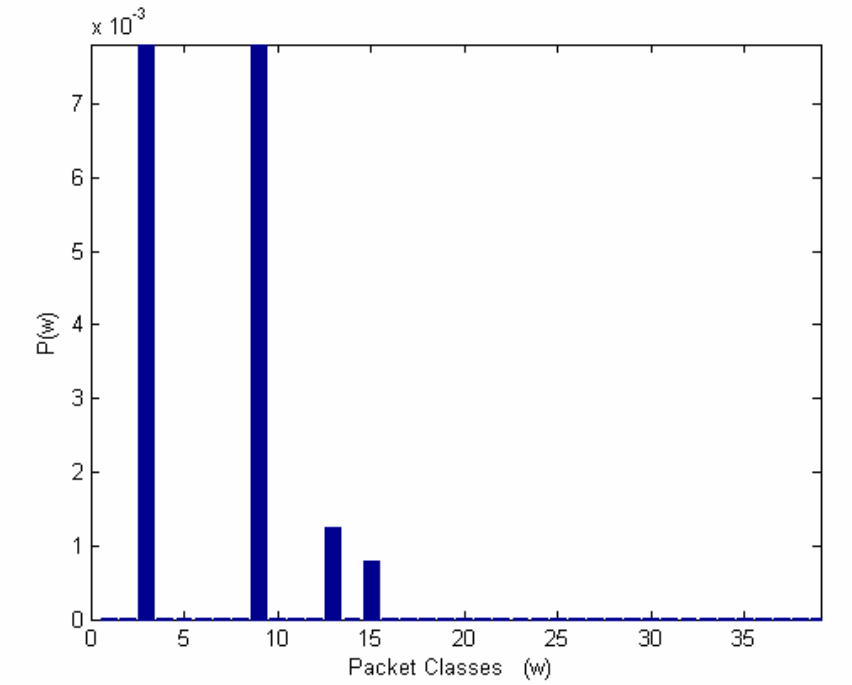

Figure 5.14. Constructed model zoomed in the vicinity of class 15

5.2.5. Modeling Neptune Attack

For neptune attack, data that is collected between 06.04.04 pm and 06.10.04 pm on Monday at week five is used as training data. The real duration of the attack is approximately six minutes, as well. For each data block of 10000 packets in six minutes, packet classification is performed and then empirical distribution of packet classes is calculated. Four blocks of 10000 packets are determined in the training data of neptune attack. Hence, four empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. Figure 5.15 presents the average empirical distribution for neptune attack.

During the port scan variant of the neptune attack, SYN packets are sent to all ports of victim machine. As can be seen from the Figure 5.15, there is an increase in pmf values of SYN packet classes. Actually, traffic distribution changes dramatically during the neptune (SYN-Flooding) attack, so that neptune attack can be noticed easily if it is in

evidence at the current traffic. Attacker sends SYN packets from private ports (49152-65535) to all ports respectively. So, for the unopened ports RST is generated as a reply of SYN packet. Therefore, class 1761 covering RST packets with the port numbers 49152-65535 has a high pmf value. RST-ACK packets are sent for half open connections, if ACK packets are not sent after waiting for a while. Thus, class 587 including RST-ACK packets with the port numbers 49152-65535 has a high pmf value, as well. Generally, all TCP ports have some activity, since malignant SYN packets are sent destined to them.

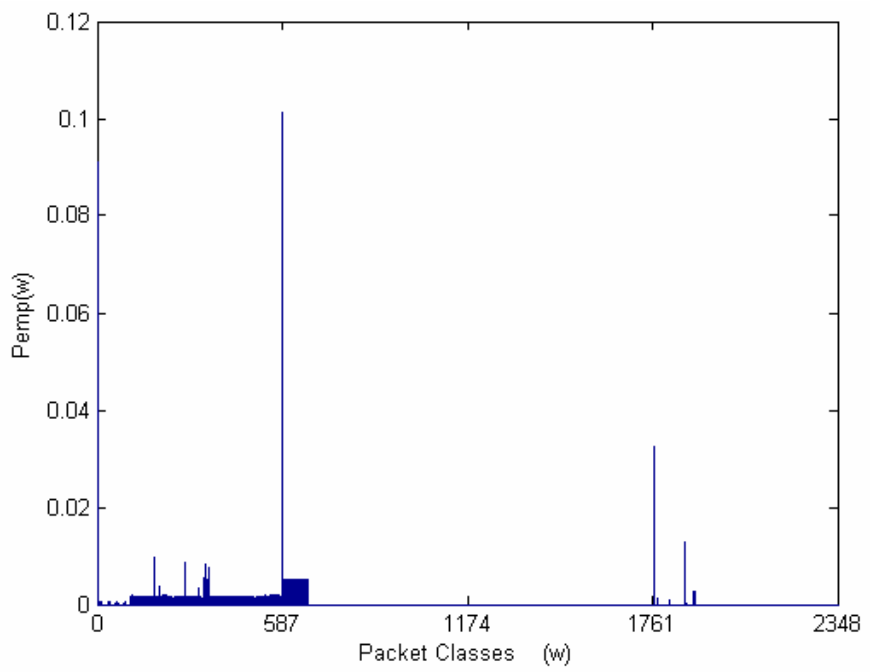

Figure 5.15. Average empirical distribution of packet classes for the training data of neptune attack

A model is estimated for the neptune attack based on the maximum entropy algorithm and the stopping criterion of this algorithm is that KL divergence between the model and the empirical distribution is less than 0,05. Consequently, 313 feature functions have been selected. Since the distribution in the Figure 5.15 has many components, many feature functions have been selected in order to characterize it completely. The first five of selected feature functions can be found in the Table 5.5

The model for neptune attack is expressed in equation (5.5) and it is illustrated in Figure 5.16.

$$P(w) = \frac{1}{Z^4} \exp(\sum_{j=1}^{313} \lambda_j^4 f_j^4(w)) \text{, where } Z^4 = \sum_{i=1}^{2348} \exp(\sum_{j=1}^{313} \lambda_j^4 f_j^4(w_i)) \quad (5.5)$$

Table 5.5. Top five feature functions selected for neptune attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port 49152-65535
2	Feature Function representing TCP packets with the port 20-29
3	Feature Function representing RST packets with the port 49152-65535
4	Feature Function representing SYN packets
5	Feature Function representing UDP packets

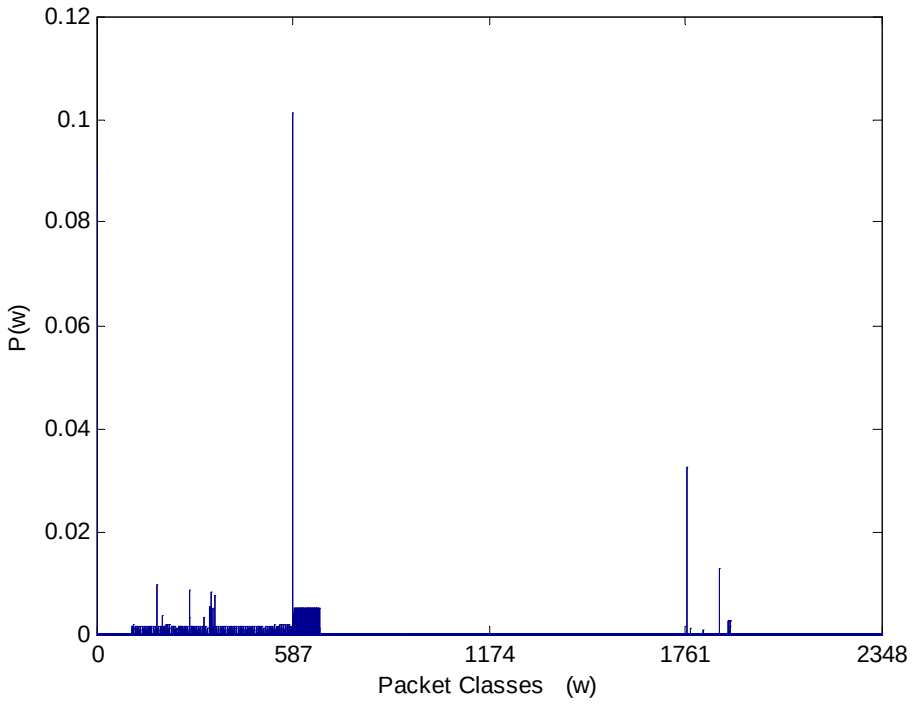

Figure 5.16. Constructed model based on maximum entropy principle for neptune attack

5.2.6. Modeling Udpstorm Attack

For udpstorm attack, data that is collected between 08.00.27 pm and 08.06.27 pm on Monday at week five is used as training data. The real duration of the attack is approximately 15 minutes. For each data block of 10000 packets in six minutes, packet classification is performed and then empirical distribution of packet classes is calculated. Seven blocks of 10000 packets are determined in the training data of udpstorm attack. Hence, seven empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. Figure 5.17 shows the average empirical distribution for udpstorm attack. Udpstorm attack uses the port 8 and port 8 stands for the packet class 1762. As can be seen in the Figure 5.17, during the udpstorm attack there is an increase in the activity of the port 8, which is assigned to class 1762.

By using this average empirical distribution, a model is built for udpstorm attack. The stopping criterion for maximum entropy algorithm is that KL divergence is 0,05. At the end of the maximum entropy process, seven feature functions have been selected. The first 5 of selected feature functions can be found in the Table 5.6.

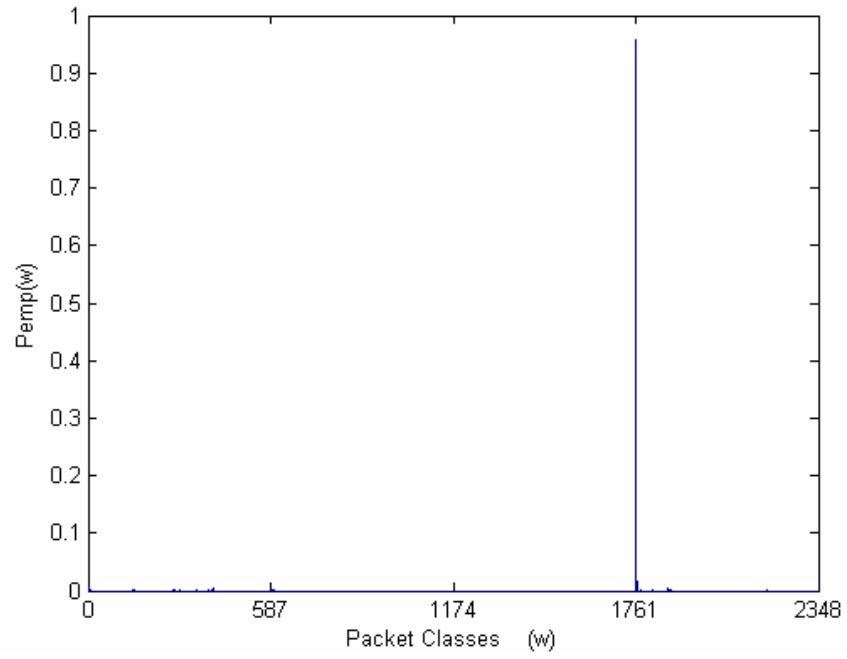

Figure 5.17. Average empirical distribution of packet classes for the training data of udpstorm attack

Table 5.6. Top five feature functions selected for udpstorm attack

Selection Order	Selected Feature Functions
1	Feature Function representing UDP packets with the port number 0-9
2	Feature Function representing UDP packets with the port number 80
3	Feature Function representing TCP packets with the port 20-29
4	Feature Function representing UDP packets with the port 1024-1123
5	Feature Function representing UDP packets

The model for udpstorm attack is given in equation (5.6) and it is illustrated in Figure 5.18.

$$P(w) = \frac{1}{Z^5}\exp(\sum_{j=1}^{7}\lambda_j^5 f_j^5(w)) \text{ , where } Z^5 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{7}\lambda_j^5 f_j^5(w_i)) \quad (5.6)$$

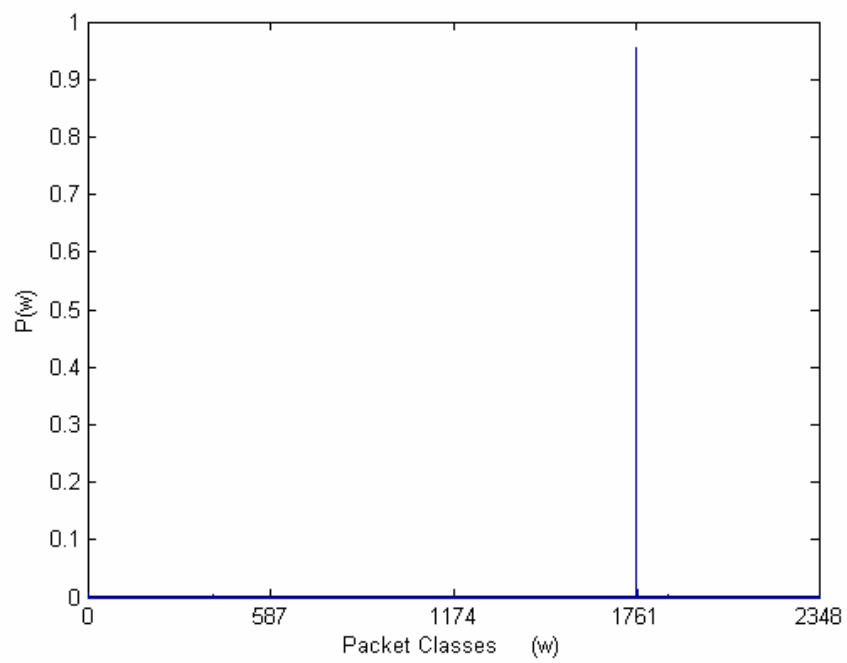

Figure 5.18. Constructed model based on maximum entropy principle for udpstorm attack

5.2.7. Modeling Tcpreset Attack

For tcpreset attack, data that is collected between 08.11.27 am and 08.20.27 am on Tuesday at week five is used as training data. The real duration of the attack is approximately 10 minutes. For each data block of 10000 packets in nine minutes, packet classification is performed and then empirical distribution of packet classes is calculated. Only one block of 10000 packets is determined in the training data of tcpreset attack. Hence, there is no need to calculate the average empirical distribution and the empirical distribution of 10000 packets is employed as the ultimate empirical distribution. Figure 5.19 presents the empirical distribution used for modeling tcpreset attack. During the tcpreset attack, an increase in the number of RST packets is expected. However it is not noticeable in the Figure 5.19. The Figure 5.20 especially focuses on the TCP RST classes of the empirical distribution. Comparing with behavior of the normal traffic, there is slightly an increase in the number of RST packets during the tcpreset attack.

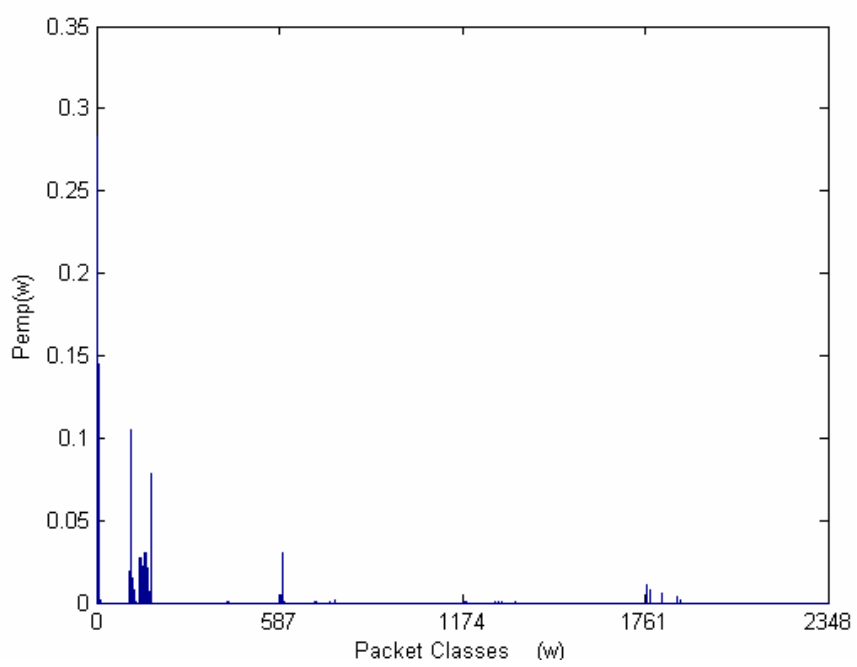

Figure 5.19. Empirical distribution of packet classes for the training data of tcpreset attack

Using this empirical distribution, a baseline distribution is estimated for tcpreset attack. The KL divergence for stopping is chosen as 0,01. This threshold is set to a smaller value than the threshold of other models, since the components that are representing

tcpreset attack have small values and it should be ensured that the model for tcpreset attack includes these components. Consequently, 73 feature functions have been selected. The first 5 of selected feature functions can be found in the Table 5.7. Unfortunately; feature functions representing the RST classes are not in the list of first five feature functions. However, it is an expected result, since components of the empirical distribution, which represents RST classes, do not have high values.

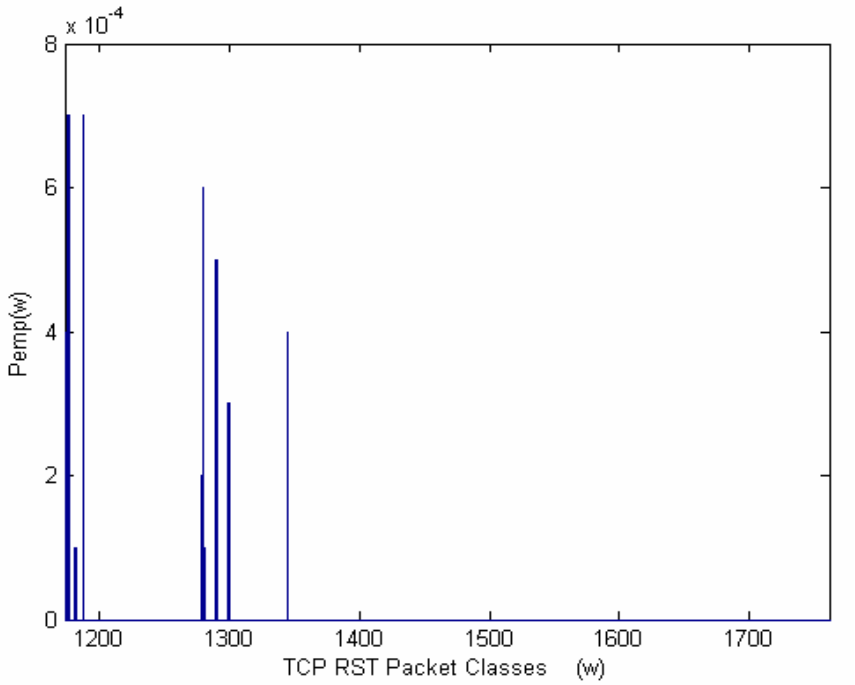

Figure 5.20. Empirical distribution of TCP RST packet classes during tcpreset attack

Table 5.7. Top five feature functions selected for tcpreset attack

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 20-29
2	Feature Function representing TCP packets with the port number 80
3	Feature Function representing TCP packets with the port 1424-1523
4	Feature Function representing TCP packets with the port 8224-8324
5	Feature Function representing TCP packets

The model for tcpreset attack is given in equation (5.7).

$$P(w) = \frac{1}{Z^6}\exp(\sum_{j=1}^{73}\lambda_j^6 f_j^6(w)), \quad \text{where} \quad Z^6 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{73}\lambda_j^6 f_j^6(w_i)) \tag{5.7}$$

This model is illustrated in Figure 5.21. However, the Figure 5.21 focuses only on RST classes, since RST classes reflect the behavior of the tcpreset attack. Basically, constructed model fulfill the constraints provided by empirical distribution. However, there are some differences. In order to estimate a model satisfying the constraints completely, a stopping criterion, which is much smaller than 0.01, should be chosen.

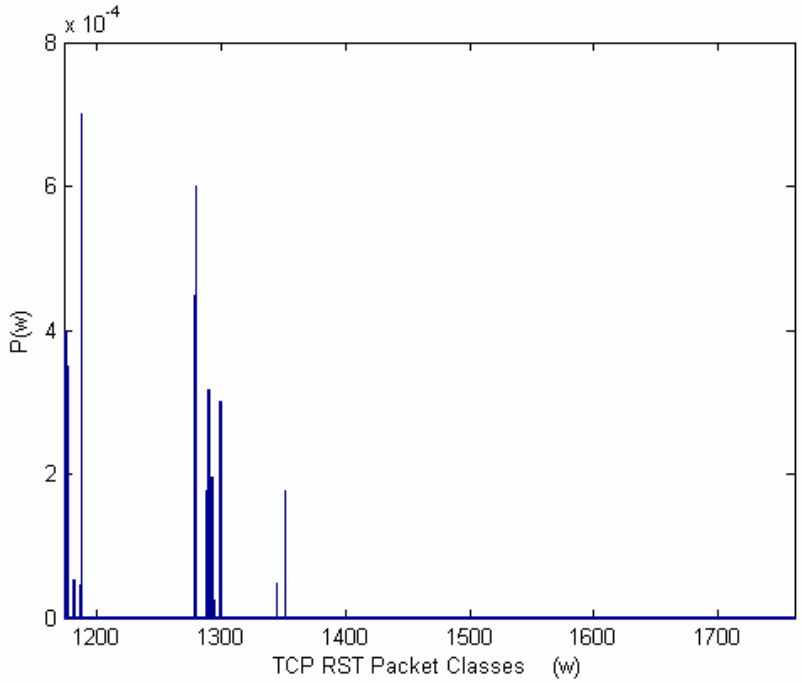

Figure 5.21. Constructed model for tcpreset attack zooming in the TCP RST packet classes

5.2.8. Modeling Mailbomb Attack

For mailbomb attack, data that is collected between 12.32.17 am and 12.38.17 am on Friday at week four is used as training data. The real duration of the attack is approximately 17 minutes. For each data block of 10000 packets in six minutes, packet classification is performed and then empirical distribution of packet classes is calculated. Two blocks of 10000 packets are determined in the training data of mailbomb attack.

Hence, two empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. The Figure 5.22 illustrates the average empirical distribution used for modeling mailbomb attack. In this figure, x axis is marked based on the logarithmic scale in order to highlight class three.

The activity of SMTP service can be observed to detect mailbomb attack. SMTP service is assigned to port 25 and the packet class for port 25 is class three, which denotes the TCP packets classes covering the port numbers 20-29. Therefore, a big value for class three in the Figure 5.22 is expected during the mailbomb attack.

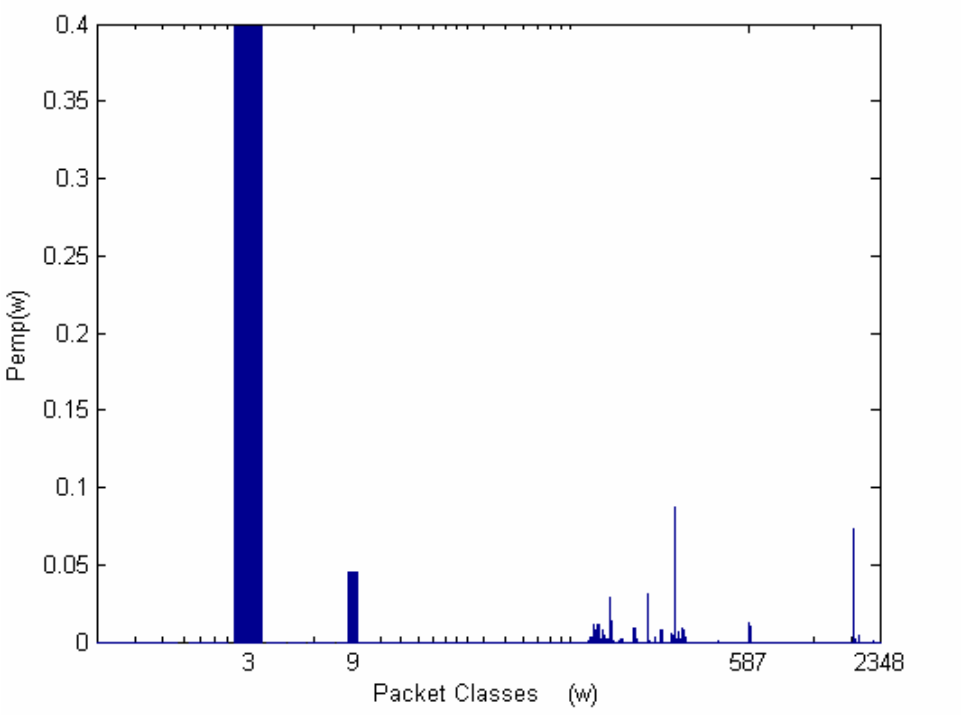

Figure 5.22. Average empirical distribution of packet classes for the training data of mailbomb attack

Using this empirical distribution, a model is built for mailbomb attack. The KL divergence for stopping is set to 0,05, as usual. Consequently, 82 feature functions have been selected. The first 5 of selected feature functions can be found in the Table 5.8. The first selected feature function is one representing the TCP packets with the port number 20-29 and it signifies directly the nature of the mailbomb attack.

The model built for mailbomb attack can be found in equation (5.8) and the Figure 5.23 shows the constructed model.

$$P(w) = \frac{1}{Z^7}\exp(\sum_{j=1}^{82}\lambda_j^7 f_j^7(w)), \text{ where } Z^7 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{82}\lambda_j^7 f_j^7(w_i)) \quad (5.8)$$

Table 5.8. Top five feature functions selected for mailbomb

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 20-29
2	Feature Function representing TCP packets with the port 17524-17623
3	Feature Function representing UDP packets with the port number 50-59
4	Feature Function representing TCP packets with the port number 80
5	Feature Function representing TCP packets

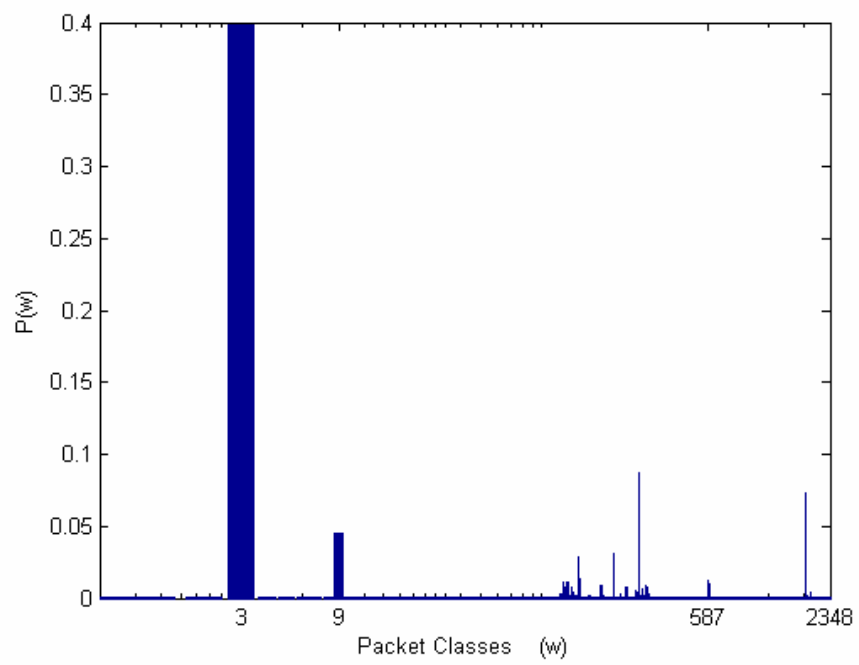

Figure 5.23. Constructed model based on maximum entropy principle for mailbomb attack

5.2.9. Modeling Syslogd Attack

For syslogd attack, data that is collected between 02.56.30 pm and 03.02.30 pm on Friday at week five is employed as training data. The real duration of the attack is approximately 15 minutes. For each data block of 10000 packets in six minutes, packet classification is performed and then empirical distribution of packet classes is calculated. Two blocks of 10000 packets are specified in the training data of syslogd attack. Thus, two empirical distributions are calculated and then the average empirical distribution is obtained from these empirical distributions. Figure 5.24 illustrates the empirical distribution used for modeling syslogd attack.

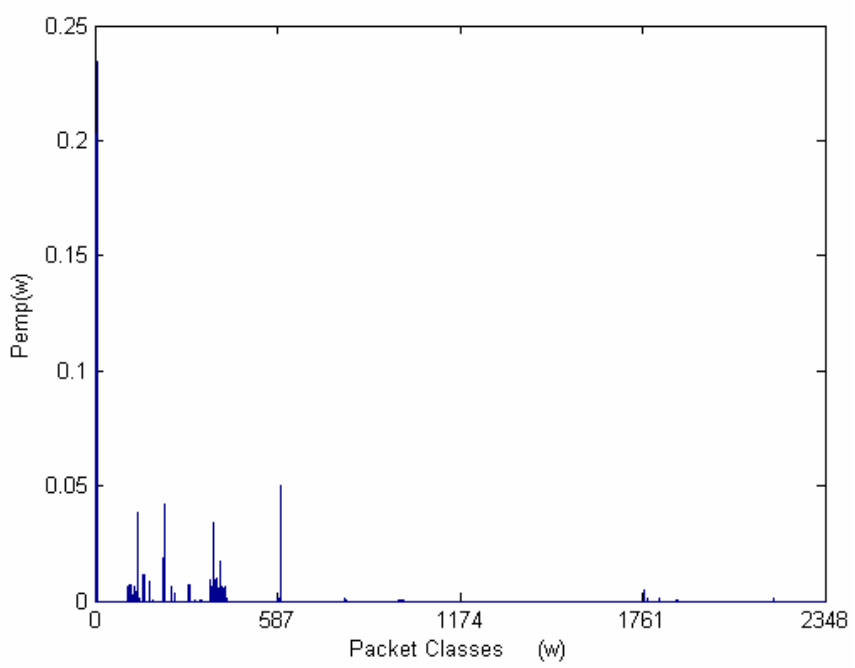

Figure 5.24. Average empirical distribution of packet classes for the training data of syslogd attack

Syslogd connects to port 514 with irresolvable source IP by using UDP protocol. Our method does not consider source address. However, this attack uses a fixed port and if the number of such malformed requests is high, our method can detect it. An UDP packet with the port number 514 stands for class 1814 according to our classification method. The Figure 5.25 focuses on the vicinity of the class 1814, since it is not noticeable in the Figure

5.24. As can be seen from the Figure 5.25, class 1814, which represents the behavior of the attack, does not have a big pmf value during the syslogd attack.

By using this empirical distribution, a model is estimated for syslogd attack based on the maximum entropy technique. The KL divergence for stopping is chosen as 0,001. This threshold is set to a much more smaller value than 0,05, since the component that is representing syslogd attack has a small value and it should be ensured that the model for syslogd attack includes this component. At the end of the training algorithm, 137 feature functions have been selected. The first five of selected feature functions can be found in the Table 5.9. Unfortunately; feature function representing class 1814 is not in the list of first 5 feature functions. However, it is an expected result, because the component related with the class 1814 does not have a high value.

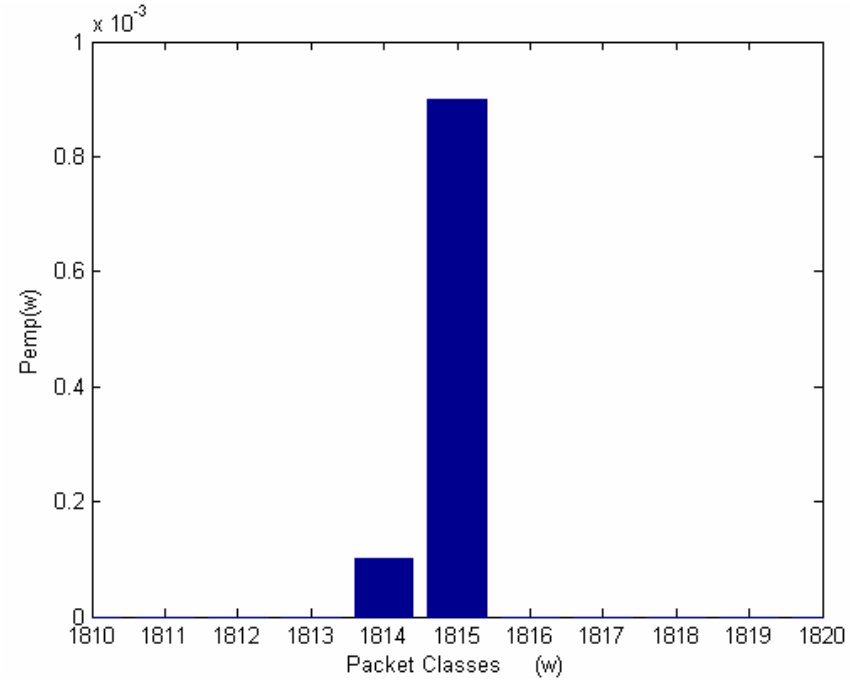

Figure 5.25. Average empirical distribution for syslogd attack in the vicinity of class 1814

The model built for mailbomb attack can be found in equation (5.9)

$$P(w) = \frac{1}{Z^8}\exp(\sum_{j=1}^{137}\lambda_j^8 f_j^8(w)) \text{, where } Z^8 = \sum_{i=1}^{2348}\exp(\sum_{j=1}^{137}\lambda_j^8 f_j^8(w_i)) \qquad (5.9)$$

This model is illustrated in Figure 5.26. However, the Figure 5.26 focuses only on class 1814, since this class reflects the nature of the syslogd attack.

Table 5.9. Top fivefeature functions selected syslogd

Selection Order	Selected Feature Functions
1	Feature Function representing TCP packets with the port number 80
2	Feature Function representing TCP packets with the port number 20-29
3	Feature Function representing TCP packets
4	Feature Function representing TCP SYN packets with the port 80
5	Feature Function representing TCP packets with the port 12824-12923

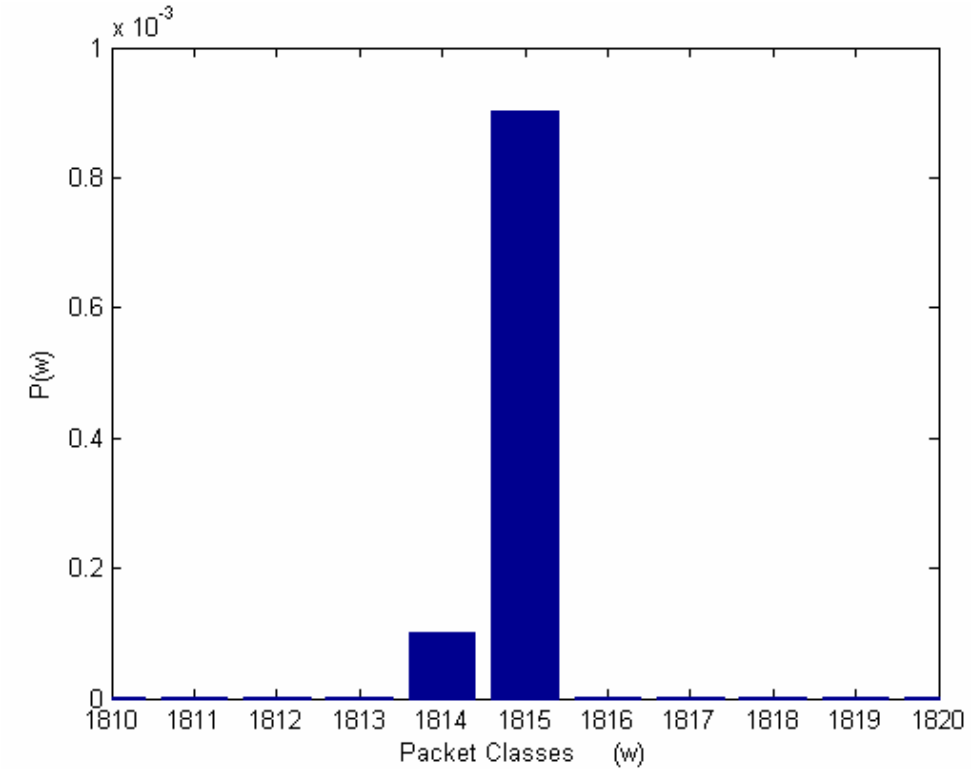

Figure 5.26. Constructed model based on maximum entropy principle for syslogd attack

5.3 Detection Part

At the training part, eight models reflecting natures of eight distinct type attacks and one model representing the nature of the attack-free network traffic are built. The goal of the detection part is to decide which of these nine models most likely satisfies the characteristics of the real-time network traffic.

As mentioned before, M-hypothesis testing technique is exploited at the detection part. Here M denotes the number of the hypothesis and in this work each model serves as a hypothesis. Thus, M is equal to nine.

Suppose that K observations are performed, meaning that K packets are captured from the test data to make a decision. Then K implies the size of the observation space at the detection part. Since each model is constructed by using average empirical distributions calculated with the blocks of 10000 packets, the size of the observation space is 10000 at the training part. In order to compare constructed models with the current traffic in a reasonable way, K should be set as 10000, as well. Actually, the value of the K affects directly the performance of our method and therefore, detection process is repeated for various values of it. However, a fixed value of K is employed throughout each run. So, fixed size packet windows are employed and the size of the window is K packets.

However, sets of K packets from the test data are not processed sequentially. In stead of consecutive windowing, sliding windowing is performed. Thus, a number of packets overlap between the current window and the prior window. Overlapping packet number is chosen as one fourth of the window size, namely 2500 packets for a window including 10000 packets.

In this manner, given a packet set $W = \{w_1, w_2, ...w_K\}$, where K is equal to 10000 and M is equal to nine, our hypotheses can be defined as in equation 5.10.

Given that one of the nine hypotheses is chosen, joint probability of the packet set $W = \{w_1, w_2, ...w_K\}$ is computed and this is repeated for each hypothesis. Then the hypothesis, for which the logarithm of the joint probability is maximized, is chosen as the

hypothesis representing the characteristics of the current packet set W at most. This procedure is performed for each overlapping packet set $W = \{w_1, w_2, ...w_K\}$ in the test data.

$$
\begin{aligned}
H_o: &\quad \forall i, w_i \sim (f_j^0, \lambda_j^0, Z^0);\ i = 1,...10000;\ j = 1,...248 \Rightarrow no\ attack \\
H_1: &\quad \forall i, w_i \sim (f_j^1, \lambda_j^1, Z^1);\ i = 1,...10000;\ j = 1,...97 \Rightarrow apache2 \\
H_2: &\quad \forall i, w_i \sim (f_j^2, \lambda_j^2, Z^2);\ i = 1,...10000;\ j = 1,...8 \Rightarrow back \\
H_3: &\quad \forall i, w_i \sim (f_j^3, \lambda_j^3, Z^3);\ i = 1,...10000;\ j = 1,...43 \Rightarrow dosnuke \\
H_4: &\quad \forall i, w_i \sim (f_j^4, \lambda_j^4, Z^4);\ i = 1,...10000;\ j = 1,...313 \Rightarrow neptune \\
H_5: &\quad \forall i, w_i \sim (f_j^5, \lambda_j^5, Z^5);\ i = 1,...10000;\ j = 1,...7 \Rightarrow udpstorm \\
H_6: &\quad \forall i, w_i \sim (f_j^6, \lambda_j^6, Z^6);\ i = 1,...10000;\ j = 1,...73 \Rightarrow tcpreset \\
H_7: &\quad \forall i, w_i \sim (f_j^7, \lambda_j^7, Z^7);\ i = 1,...10000;\ j = 1,...61 \Rightarrow mailbomb \\
H_8: &\quad \forall i, w_i \sim (f_j^8, \lambda_j^8, Z^8);\ i = 1,...10000;\ j = 1,...137 \Rightarrow sys\log d
\end{aligned}
\tag{5.10}
$$

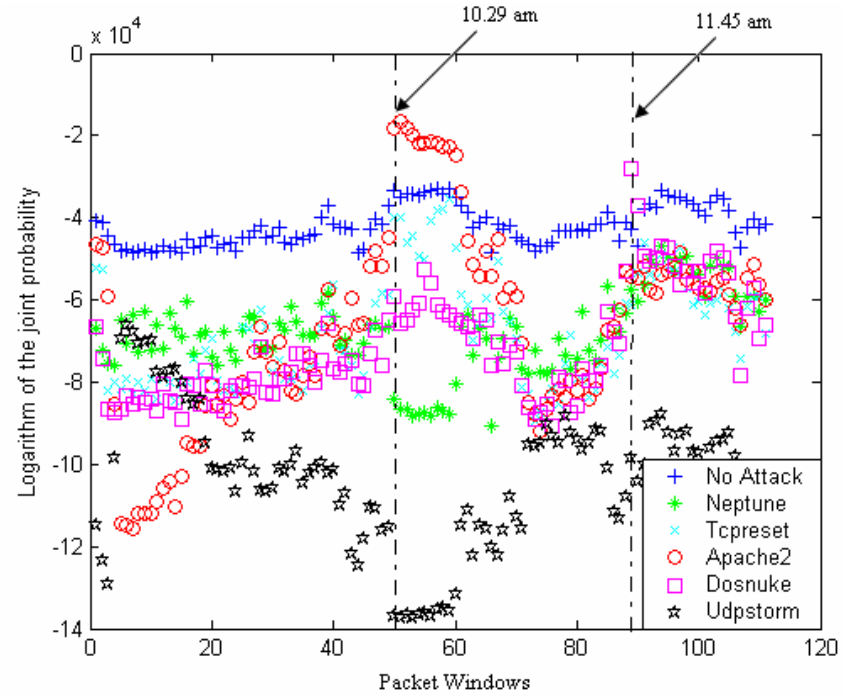

Figure 5.27. Given that H_0, H_1, H_3, H_4, H_5 or H_6 are chosen, joint probability of each packet set between 8.00 am and 00.10 pm.

For example, Figure 5.27 illustrates hypothesis testing process between H_0, H_1, H_3, H_4, H_5 and H_6 hypotheses. This figure is obtained by using the test data collected from

08.00 am to 00.10 pm on Monday at week five. In this time interval, two attack instances are labeled. One is the apache2 attack occurring at 10.29 am and the other one is the dosnuke attack taking place at 11.45 am. X axis shows the overlapping packet windows, each of which includes 10000 packets. Given that H_0, H_1, H_3, H_4, H_5 or H_6 are chosen, y axis shows the logarithm of the joint probability of each packet set severally for each hypothesis. In this figure, plus marker is used for no attack hypothesis, circle marker is used for apache2 hypothesis and square marker is used for dosnuke attack. Generally, logarithm of the joint probability is maximized for no attack hypothesis and it is an expected result. During the apache2 attack, as can be seen from the Figure 5.27, H_1 hypothesis (hypothesis for apache2 attack) maximizes the joint probability of the packet set at that moment. Similarly, H_3 maximizes the joint probability during the dosnuke attack. In this manner, hypothesis maximizing the joint probability of each packet set is determined and model satisfying the characteristics of each packet set is identified.

5.3.1. Detection of Apache2 Attack

During the apache2 attack lots of http request with http headers are sent to the web server. Since the server receives many of these http requests, it slows down, and in the long run it may crash. Due to the increase in http requests, an increase in the activity of port 80 is expected. Especially, class nine covering the TCP packets with the port number 80 represents this behavior at our work. As can be seen from Figure 5.9, the pmf value of class nine is approximately 0,6 for the model of apache2 attack; however, this value is determined as 0,165 for the model of normal traffic. Thus, the model of attack free case and the model of apache2 differ dramatically in terms of the activity of the class nine. Obviously, the model of the apache2 is distinctive and it can be detected easily.

In the DARPA 1999 dataset, three instances of the apache2 attack take place. One of them used to train the model and other two instances are employed for test purposes. These two instances are detected successfully with our proposed method.

The Figure 5.28 presents the results of the hypothesis testing between H_0 (No attack hypothesis) and H_1 (Apache2 attack hypothesis) on Monday at week five. Similarly, the Figure 5.29 illustrates the results of the hypothesis testing between H_0 and H_1 on

Wednesday at week five. Only the values of H_0 and H_1 are plotted on these figures in order to avoid the mess of the many markers; however, all hypotheses are tested surely, while deciding. All other hypotheses have a smaller joint probability value than H_0 and H_1 during the apache2 attack.

In the figure 5.28, x axis shows all packet windows (301 overlapping packet windows, each of which includes 10000 packets) captured on Monday. Two apache2 attack instances take place on Monday. The attack instance occurring at 10.29 am is used to train the method and the other instance occurring 02.05 pm is detected successfully.

X axis of the Figure 5.29 shows all packet windows (276 overlapping packet windows, each of which includes 10000 packets) captured on Wednesday. Another apache2 attack instance is labeled at 05.13 pm on DARPA 1999 data and as can be seen from the figure, it is detected successfully by our method.

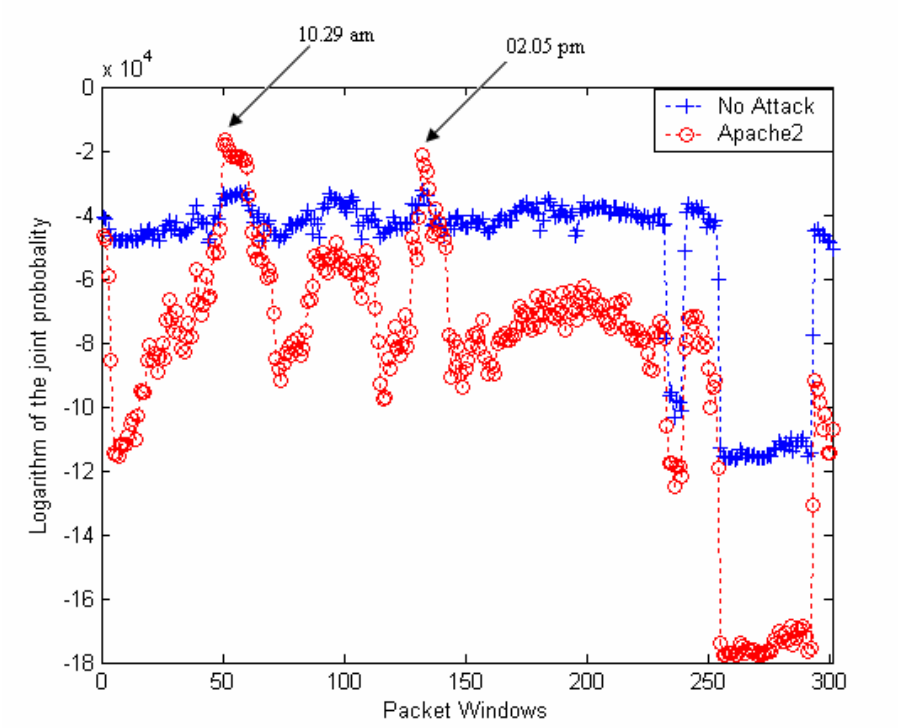

Figure 5.28. Hypothesis testing results for apache2 instances existing on Monday at week five

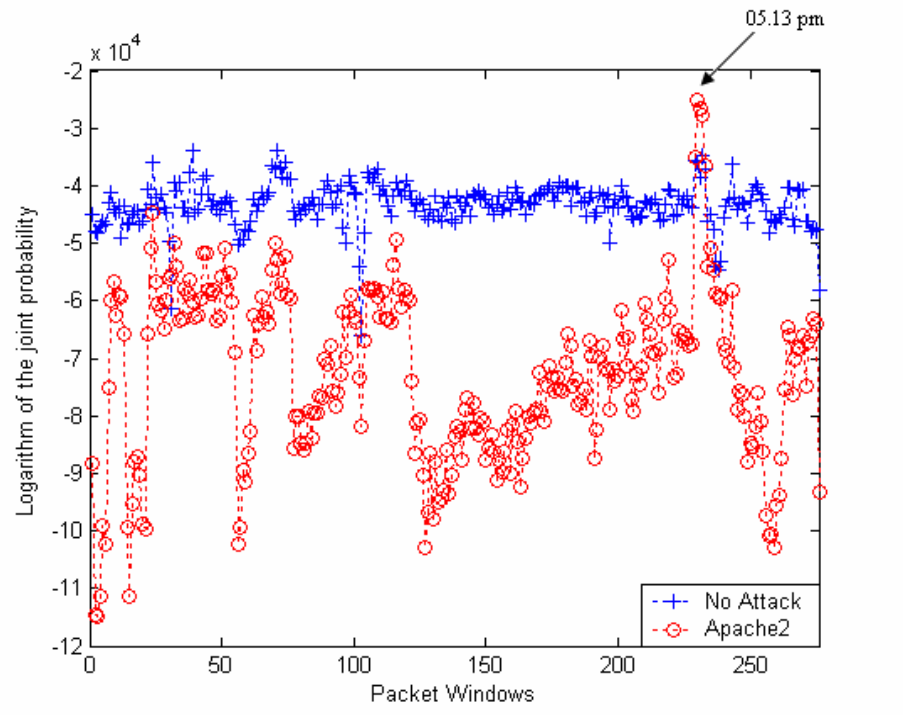

Figure 5.29. Hypothesis testing result for apache2 instance existing on Wednesday at week five

Consequently, our method is very successful detecting apache2 attacks and by looking the figures above, no false alarm is raised for apache2 attack throughout days Monday and Wednesday.

5.3.2. Detection of Back Attack

Back attack is a denial of service attack against the apache web server. This attack performs a malformed web request to port 80 with the payload, "\Get //// ..." followed by 6000-7000 slashes [10]. As a response, duplicate ACK packets are generated by the victim since an out-of-order segment is received [35]. Back attack affects the payload; however our method considers only the header not the payload. If the number of such requests is not numerous during the back attack, the distribution of the traffic will not represent this anomaly. If the number of such malformed requests is high, then an increase in the pmf value of class nine is expected.

Several back attack instances take place in the DARPA 1999 dataset; however only one attack instance affects the traffic distribution and this is the one that is used to train our model. The Figure 5.11 shows the constructed model for back attack and in this figure, there is an increase in the pmf values of class nine and class 116. The source port of the attacker is 2164 for this back attack instance and it denotes the class 116. The increase in the pmf value of class 116 is based on the duplicate ACK packets sent from the web server to the attacker. So, this is related with the back attack; however it is not a common behavior for all back attack instances, since the attack can be performed from another port and the activity of another port might increase due to the duplicate ACK packets destined to the port of attacker. Therefore, it is hard to model back attack.

Furthermore, the model of the back attack could not be distinguished from the apache2 attack, if no duplicate ACK packets are sent to attacker port during the back attack, since both attacks cause an increase in the activity of class nine.

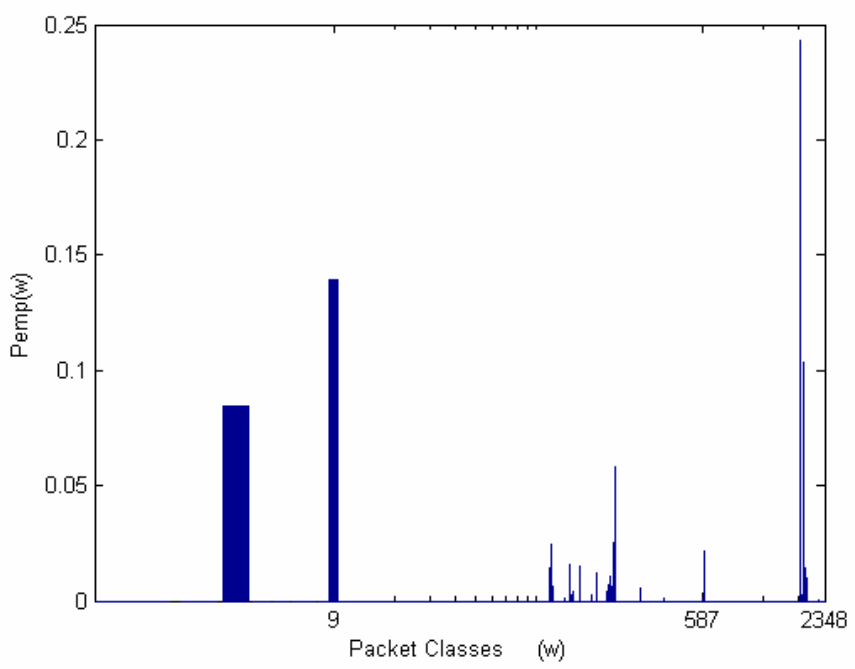

Figure 5.30. Empirical distribution of packet classes during the back attack instance taking place in the data captured on Monday at week two

Moreover, if the number of the malformed requests is not high, there will not be an increase in the activity of class nine and, this makes it impossible to detect back attack by

considering only header information. For example, the Figure 5.30 shows the empirical distribution during a back attack instance that takes place in the test data captured on Monday at week two. X axis of this figure is marked based on the logarithmic scale in order to highlight the behavior of the class nine. As can be seen from this figure, there is not an increase in the pmf value of the class nine during this back attack instance. This means that the number of the malformed requests is not high. Actually, this empirical distribution seems to represent more likely normal traffic behavior, since class nine has a pmf value 0.15, which is close to the value of no attack case. So, it is impossible to detect this attack with our method.

The Figure 5.31 demonstrates the activity change of the class nine among the packet windows. X axis shows the packet windows in the vicinity of the back instance, which takes place in the data captured on Monday at week two. Y axis shows the probability of the class nine in each packet window. Here each packet windows includes 10000 packets and these windows are not overlapping. The window seven consists of the data captured between 09.39.13 am and 09.42.20 am. Since the time of the back attack instance is labeled as 09.39.16 am, the data related with the attack are in this window. By looking the windows in the vicinity of this window, there is not an abnormal change in the pmf value of the class nine during the attack. However, an increase in the pmf value of the class nine is required during the back attack instance to be able to detect it by using our method.

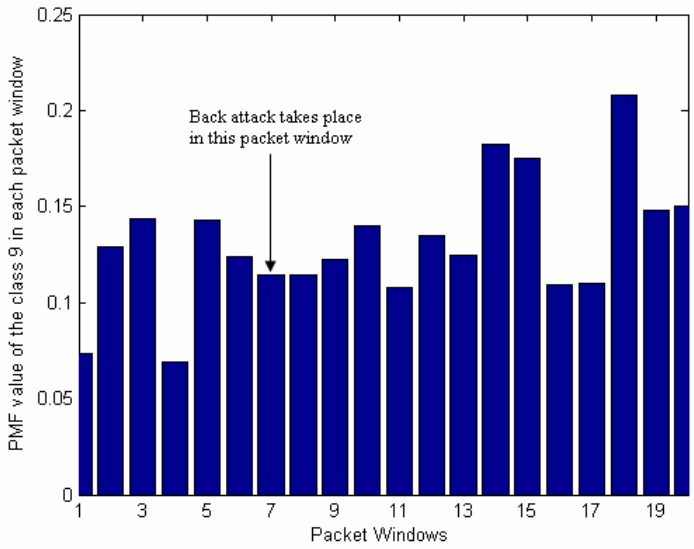

Figure 5.31. Probability of the class nine in each packet windows captured between 09.22 am and 10.22 am on Monday at week two

Other back attack instances have the similar behavior and they do not change current empirical distribution evidently, except one used to train our method. Therefore, none of them can be detected by our method. For example, Figure 5.32 shows the result of the hypothesis testing between H_0 and H_2 during another back attack instance occurring on Friday at week five. In fact, in the 16th packet window back attack occurs; however, it can not be detected, since this instance does not affect the traffic distribution. As can be seen from the figure, packet windows in the vicinity of 16th packet window have all most the same joint probability. This means that these sequential packet windows have similar packet distribution and there is no change in the packet distribution during this back attack instance.

Furthermore, it should be highlighted that the duration of all back attack instances, except the one used for training, is too short. The one with longest duration lasts approximately one and half minutes.

Eventually, our method could not detect the back attack, since this attack affects the payload. Our method can detect it, only if there the duration or the number of such malformed requests is very high.

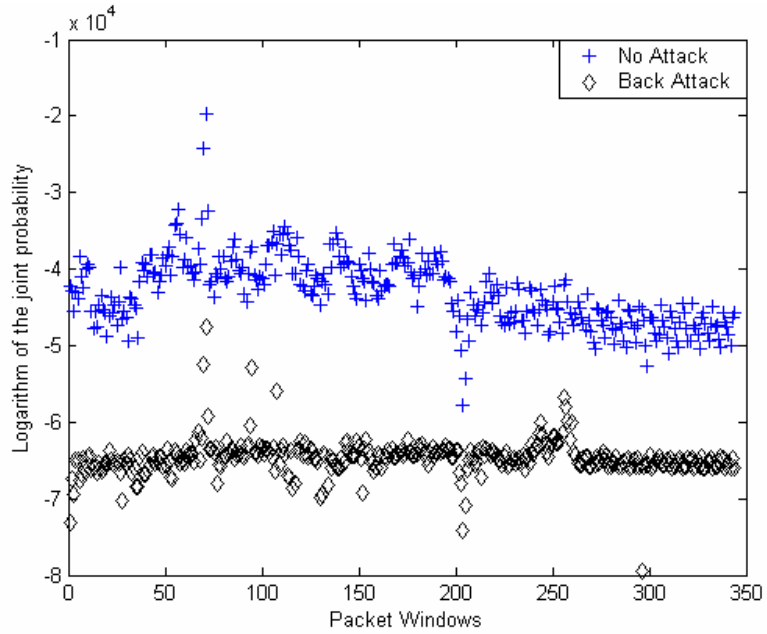

Figure 5.32. Hypothesis testing result for back attack instance taking place on Friday at week five

5.3.3. Detection of Dosnuke Attack

In this kind of attack, “out-of-band” data (MSG_OOB) data that is delivered with higher priority than ordinary data are sent to the port 139 of the victim machine. The packets being sent by the attacking machines are flagged "urg". When a Windows system receives a packet with the "urg" flag set, it expects data will follow that flag. The exploit consists of setting the flag "urg", but not following it with data [11]. Thus, it can be detected easily by searching the sniffed data for a NetBIOS handshake followed by NetBIOS packets with the "urg" flag set. However, our method does not deal with "urg" flag and therefore, our method can detect this attack, only if the empirical distribution of the packet classes differ dramatically during this attack.

This attack exploits from a fixed port. Port 139 denotes the class 15 according our classification method. However, as can be seen from the Figure 5.14 there is not a huge increase in the activity of the class 15 during the dosnuke attack. The pmf value of class 15 is equal to $8x10^{-4}$ during the dosnuke attack and it is equal to $9{,}4737x10^{-5}$ during the normal network traffic. This makes it hard to detect this attack with our method.

In the DARPA 1999 dataset, four instances of the dosnuke attack take place. One of them used to train the model and other three instances are employed for testing. Since the change in the distribution is not evident during the attack, a narrower window with 500 packets is used throughout the hypothesis testing of dosnuke attack. In this manner, only one of the dosnuke attack instances in the test data can be detected with our method. The other two instances can not be identified even with the packet window including 500 packets.

The Figure 5.33 presents the results of the hypothesis testing between H_0 and H_3 performed with the data, which is collected on Thursday at week four. The dosnuke attack takes place at 11.00 am and it lasts 16 minutes. Our method raises an alarm for this attack instance at 11.07 am. This figure focuses only on the vicinity of the attack instance; however the hypothesis testing is performed for the data of whole day. Moreover, hypothesis testing is performed by considering all hypotheses, but only the results of the H_0 and H_3 are plotted here. Since a narrower window is used throughout the detection of

dosnuke attack, false alarms are generated. For example, for the data captured on Thursday at week four, six false alarms are raised.

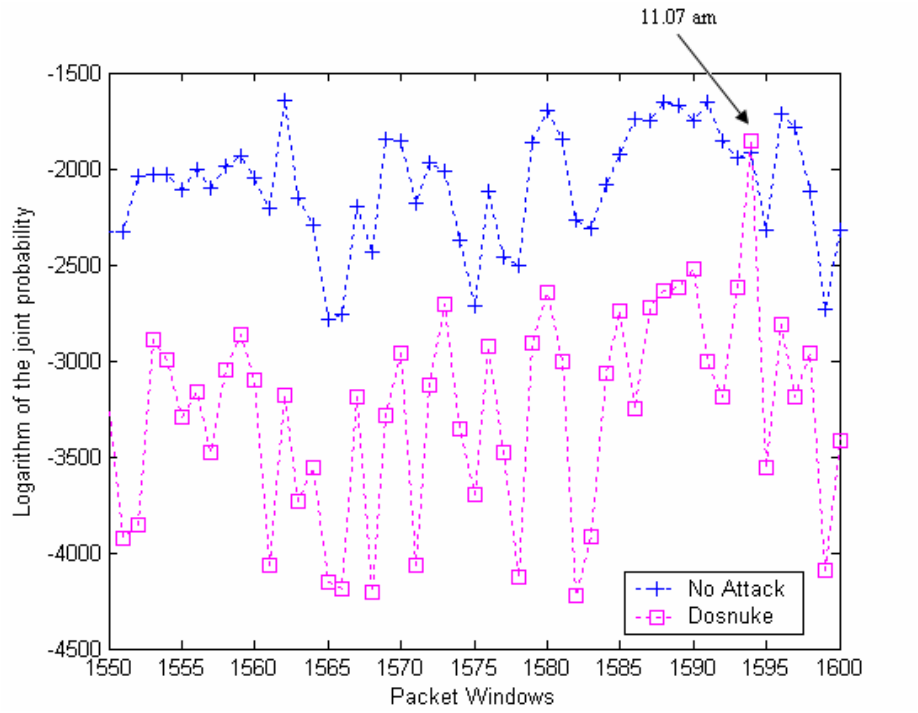

Figure 5.33. Hypothesis testing result for dosnuke attack instance taking place on Thursday at week four

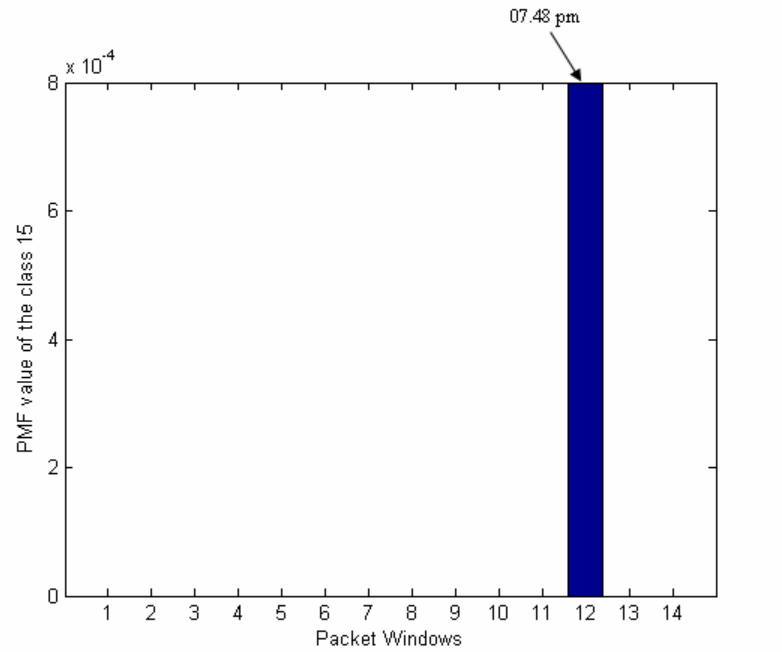

Figure 5.34. Probability of the class 15 in each packet windows captured between 07.30 pm and 08.30 pm on Monday at week five

Since this attack does not effect the packet distribution, there is no need to examine packet distribution of the undetected attack instances. Studying the activity of the class 15 in the vicinity of the undetected instances provides enough information. The figure 5.34 demonstrates the activity of the class 15 during one of the dosnuke attacks, which can not be detected. This attack instance takes place at 07.48 pm on Monday of week five. In this figure, each packet windows consist of 10000 packets, as usual. During the attack, only 8 malformed packets are sent to the victim through the port 139 denoting class 15. So, the pmf value of the class 15 is equal to $8x10^{-4}$, which is the same value in the model of the dosnuke attack. Since class 15 is not a dominant class, most likely the probability of other classes affect the decision of dosnuke attack.

As a result, our method is a poor method for detecting the dosnuke attack, since the traffic distribution does not differ from the normal traffic for the class 15.

5.3.4. Detection of Neptune Attack

During the neptune (SYN-flooding) attack, attacker sends many SYN packets to various ports of the victim machine. For the unopened ports RST is generated as a reply of SYN packet. For available ports, SYN-ACK messages are generated by the victim; however the attacker does not send the ACK messages back to the victim. This causes many half open connections [36]. RST-ACK packets are sent for half open connections, if SYN-ACK packets are not sent after waiting for a while. Accordingly, an increase in the activity of SYN classes, SYN-ACK classes, RST classes and RST-ACK classes is expected during this attack.

During the neptune instance, which is used to construct the model illustrated in the Figure 5.16, the attacker sends SYN packet from private ports to all ports respectively. Therefore, there is an increase in the activity of all TCP packet classes, namely from class 1 to class 587. Especially, class 587 has a huge value due to the RST-ACK packets destined to the private ports, so to the attacker. Additionally, class 1761 covering the RST packets with the private ports has a big probability because of the RST packets destined to the attacker. Obviously, there is a dramatic change in the packet class distribution during the neptune attack. Therefore, our method can detect the neptune attack.

In the DARPA 1999 dataset, there are six instances of the neptune attack. One of them used to train the model and other five instances are employed for test purposes. The three of them are detected easily with our proposed method. For example, a neptune attack instance takes place at 11.04 am on Thursday of week two. The Figure 5.35 illustrates the results of the hypothesis testing between H_0 and H_4 on Thursday at week two. X axis of the Figure 5.35 shows all packet windows (205 overlapping packet windows, each of which includes 10000 packets) captured on Thursday and the attack time is marked in this figure. By looking this figure, H_4 (hypothesis for neptune attack) has a greater joint probability than H_0 for the packet window including the neptune attack instance. Generally, all hypotheses contributing the testing process are not plotted in order to avoid the mess of markers. Since there is a decrease in the joint probabilities for both H_0 and H_4 during the attack, it is better to plot all hypotheses to prove that H_4 maximizes the joint probability of the packets during the attack. In this manner, the Figure 5.36 shows the results of the hypothesis testing between all hypotheses on Thursday at week two.

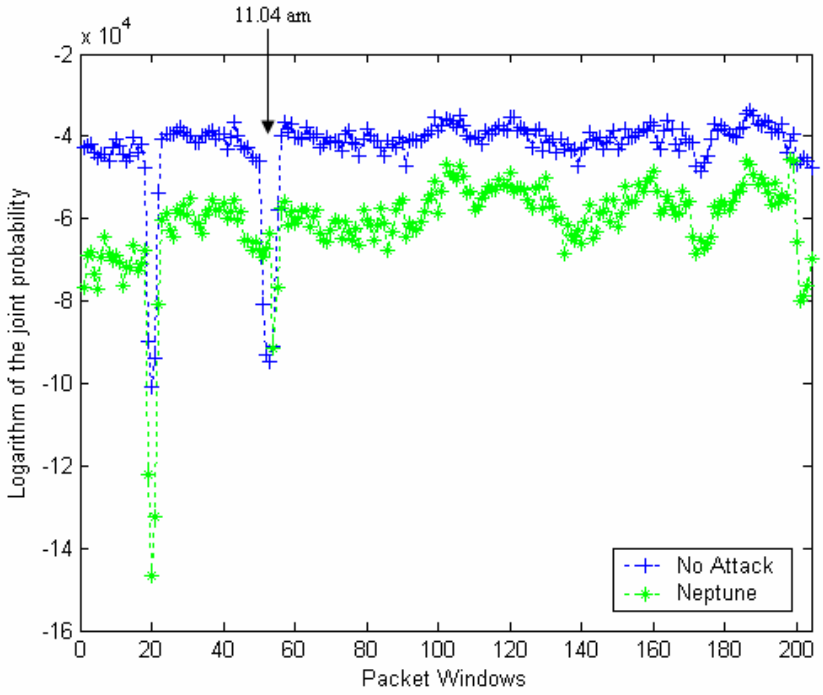

Figure 5.35. Hypothesis testing result between H_0 and H_4 for neptune attack instance taking place on Thursday at week two

In the Figure 5.36, it is evident that H_4 maximizes the joint probability of the packet window, which includes the neptune attack instance. In the data captured on Thursday,

there is only neptune attack instance from our target attack group. Therefore, note that no false alarm is raised for this data.

One of the neptune attack instances lasts only nine seconds. Therefore, it can not be detected with a packet window including 10000 packets. A window with 10000 packets denotes approximately three minutes, so this span is much longer than the duration of this attack instance. Therefore, a narrower window 1000 packets is used and in this case it can be detected successfully. However, using a narrower window causes false alarms especially for dosnuke, syslogd and tcpreset attacks, since the models of these attacks are similar to the model of normal traffic.

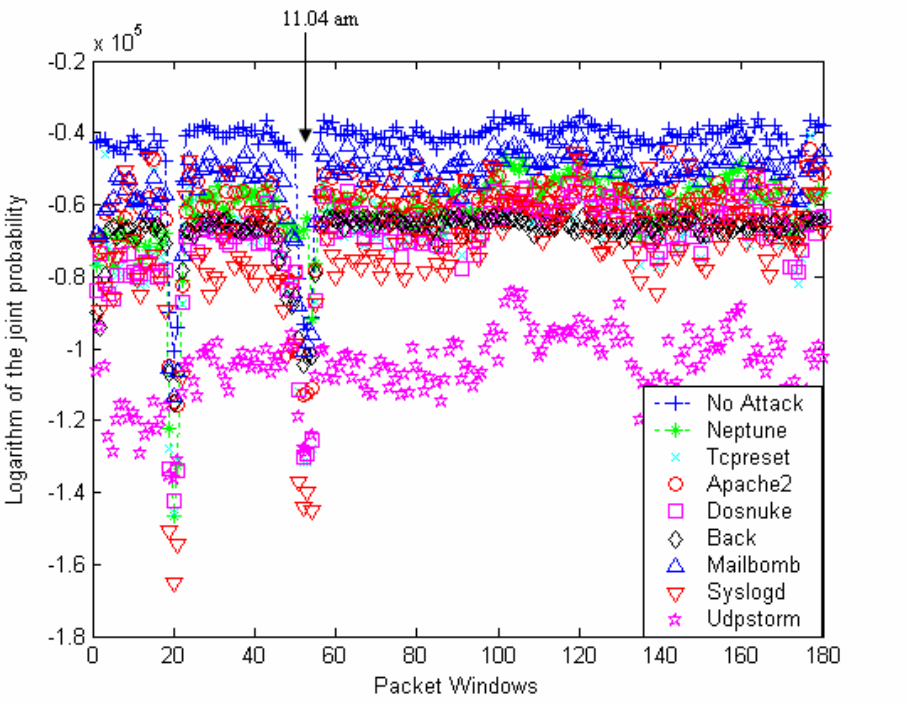

Figure 5.36. Hypothesis testing result between all hypotheses for the data captured on Thursday at week two

The final neptune instance can not be detected, since it takes place in the same time with a back attack instance. This back instance occurring at 11.31 am on Tuesday of week five lasts approximately twenty minutes and first six minutes of this instance is used to build a model for back attack. Then at 11.38 am, the neptune attack starts, but it is not evident, since the back instance is much more dominant than it. Therefore; our method chooses the hypothesis for back attack not the one for neptune attack. In order to overcome

the problem of simultaneous attack instances, joint models can be constructed at the training part additional to the models of each attack. For instance, a model for simultaneous neptune and back attacks can be constructed.

Consequently, our method is successful in detecting neptune attacks, except a few cases.

5.3.5. Detection of Udpstorm Attack

Udpstorm sets up a flood between the echo and chargen servers on two victims by sending a request to one with the spoofed source address of the other. The result is that they echo each other endlessly and waste network bandwidth [37].

Udpstorm attack uses the port 8 and port 8 stands for the packet class 1762 in our work. The huge increase in the activity of the class 1762 can be observed in the Figure 5.18. During the udpstorm, the packet class distribution attack differs evidently from the normal distribution. Therefore, this kind of attack can be easily identified by our proposed method.

Two instances of udpstorm attack exist in the DARPA 1999 data set. One of them used to train the model for udpstorm attack and the other one is exploited to test the model. This instance is detected successfully by using our method.

The udpstorm attack instance, which is used to test our method, takes place at 05.30 am on Wednesday in the week five. However, all data before 08.00 am is included by the data of the previous day. So, this attack instance occurs in the data of Tuesday. The Figure 5.37 demonstrates the results of the hypothesis testing between H_0 and H_5 hypothesis for this attack instance. As can be seen from the figure, udpstorm attack is identified easily. Accordingly, our method is eligible in order to detect udpstorm attacks, since the packet distribution differs dramatically during the udpstorm attack.

5.3.6. Detection of Tcpreset Attack

During this attack, the attacker listens for TCP connections of the victim, and sends a spoofed RST packet to the victim, causing the victim to terminate the TCP connection [4].

Comparing with behavior of the normal traffic, there is slightly an increase in the number of RST packets during the tcpreset attack (see Figure 5.21). However, this attack can be detected, since RST packets are generally generated if there is an anomaly.

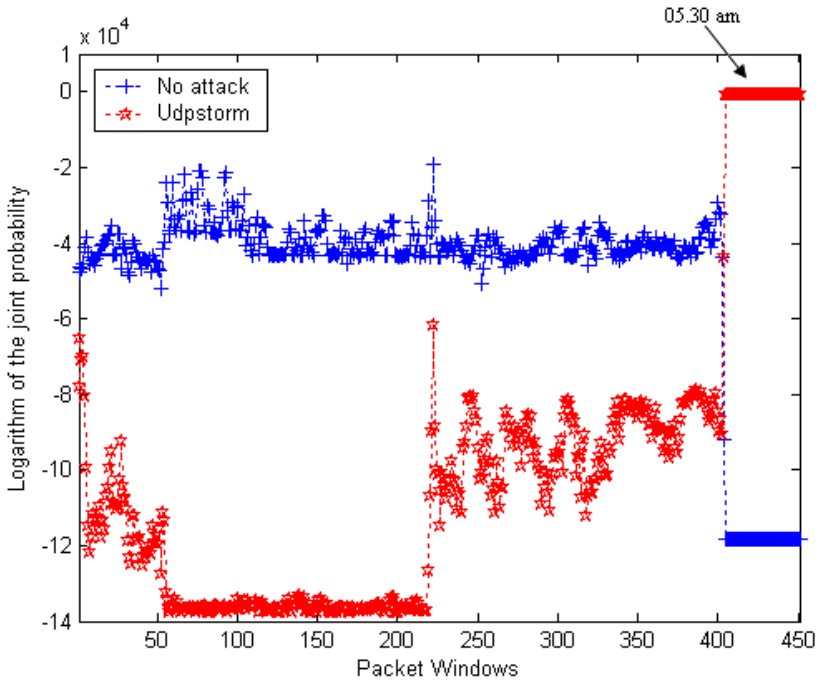

Figure 5.37. Hypothesis testing result between H_0 and H_5 for udpstorm attack instance taking place on Tuesday at week five

In the DARPA 1999 dataset, three instances of the tcpreset attack take place. One of them used to train the model and other two instances are employed for testing. The one, which starts at 09.44 am on Wednesday in the week five, identified easily with our method. This tcpreset attack instance lasts for 15 minutes and our method raises an alarm at 09.51 am for it. The Figure 5.38 illustrates the results of the hypothesis testing between H_0 and H_6 hypotheses for this attack instance. X axis shows the first fifty packet windows, each of which includes 10000 packets. Note that all hypotheses are considered for the hypothesis testing process, but all of them are not plotted here.

The other tcpreset attack instance can not be detected with a packet window including 10000 packets, since it has a shorter duration than the one used for training. Therefore, a narrower packet window, which consists of 1000 packets, is used. Eventually, it is detected with this packet window.

If the duration of the tcpreset attack is not short, then it can be detected easily with our method. Even in the case of a short duration, tcpreset attack can be detected, but a narrower packet window is required.

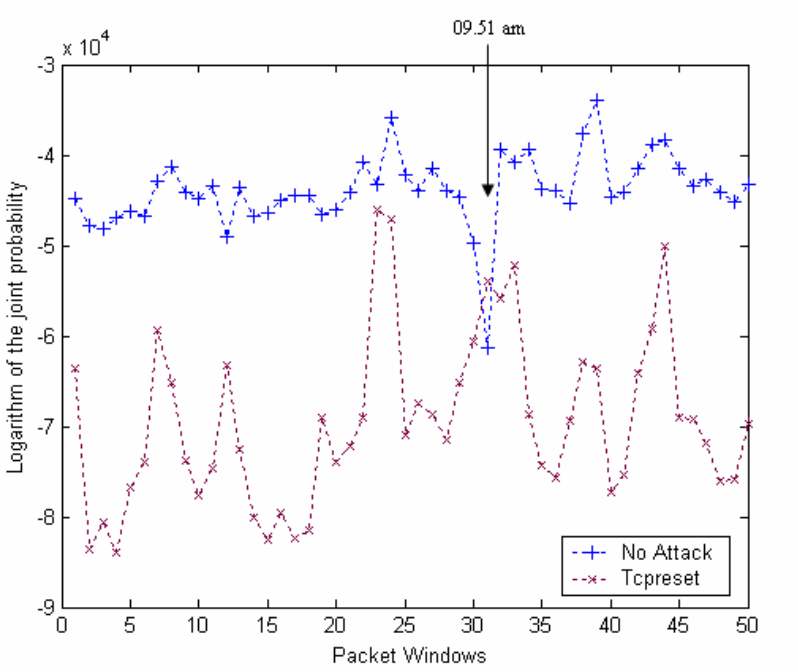

Figure 5.38. Hypothesis testing result between H_0 and H_6 for tcpreset attack instance taking place on Wednesday at week five

5.3.7. Detection of Mailbomb Attack

Mailbomb is a denial of service attack, in which the attacker sends many messages to a server. So, it causes that the mail queue of the server overflows. As a result the server is slowed down being unable to promptly respond to clients and possibly crashing [4].

An abnormal behavior of SMTP service can be encountered during the attack. SMTP service is assigned to port 25 and the packet class for port 25 is class three, which denotes

the TCP packets classes covering the port numbers 20-29. However, the aim of this type of attack is not to produce a high traffic volume, but to create a large size of messages in the server's queue. Therefore, by looking only at the IP header, a normal host might exhibit behavior similar to that of an attacker since it could send mail with large attachments to the SMTP server [37]. In that case, our method may not be able to detect this attack.

In the DARPA 1999 dataset, six instances of the mailbomb attack are labeled. One of them used to train the model. The tcpdump files, which consist of the data captured on Tuesday and Wednesday in week four, are corrupted. Two attack instances take place on that days; however these tcpdump files cannot be read till the end and data related with the attack cannot be gathered. Therefore, these two mailbomb instances are ignored and other three instances are used for testing.

None of them can be identified by using a packet window which includes 10000 packets, because these mailbomb instances do not change packet distribution dramatically. During these attack instances, mails with large attachments may be sent to the SMTP server instead of sending many short mails. In this case, distribution of class three does not change conspicuously. Furthermore, port 25 is assigned to the same class with other much used ports such as 20(FTP), 21 (FTP) and 23(TELNET) by our classification algorithm. Therefore, the activity of the class three is not dependent only SMTP service. Actually, assigning the SMTP port to a single class may result in an increase of the detection rate of the mailbomb attack. The Figure 5.39 demonstrates the activity of the class three between 02.20 pm and 03.20 pm on Tuesday in the week two. A mailbomb attack instance is labeled at 02.25 pm on that data. X axis of this figure shows the packet windows including 10000 packets and the attack instances takes place in the 11^{th}, 12th, 13th and 14^{th} packet windows. As can be seen from this figure there is not an increase in the pmf value of the class three during the attack. Although there is no attack labeled at 02.23 pm, pmf value of the class three increases in packet window seven (02.23 pm) dramatically. Thus, a false alarm is raised for mailbomb attack at that point, but the actual attack instance can not be detected with the packet window consisting of 10000 packets.

In this manner, the size of the packet window is set to 1000 packets and all mailbomb attack instances in the test data are identified with this narrow packet window.

Nevertheless, using a narrower packet window than the one including 10000 packets causes an increase in the false alarms, since all models are built by using sets of 10000 packets. The Figure 5.40 illustrates the results of the hypothesis testing between H_0 and H_7 hypotheses for the attack instance taking place at 02.25 pm on Tuesday in the week two. X axis shows the packet windows, each of which consists of 1000 packets. For same reason all hypothesis are not plotted here, but H_7 hypothesis maximizes the joint probability of the packet window including the attack instance.

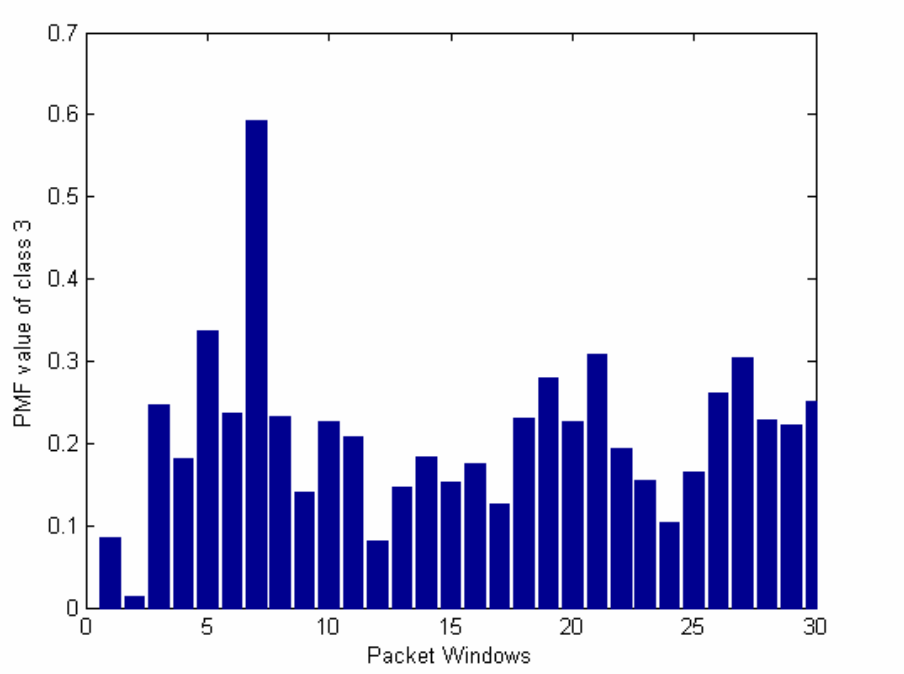

Figure 5.39. Probability of the class three in each packet windows captured between 02.20 pm and 03.20 pm on Tuesday in the week two

Consequently, a narrower packet window is employed for the detection of the mailbomb instances and so these instances are successfully detected. In order to improve the detection rate, port 25 can be assigned to a single class. In this case, attack instances can be caught with packet window in the original size, as well.

5.3.8. Detection of Syslogd Attack

Many UNIX systems provide a general logging facility, called syslogd. Under Solaris, if syslogd receives a packet containing an irresolvable IP address, it will crash with

a segmentation fault. So, syslogd attack exploits this vulnerability of Solaris and allows an attacker to remotely kill the syslogd service on a Solaris server [4].

During this attack, a few spoofed UDP packets are sent to the port 514 on the victim machine. This attack uses a fixed port; however it does not produce high volume traffic through this port. Therefore, our method may not be able to catch this attack all the time.

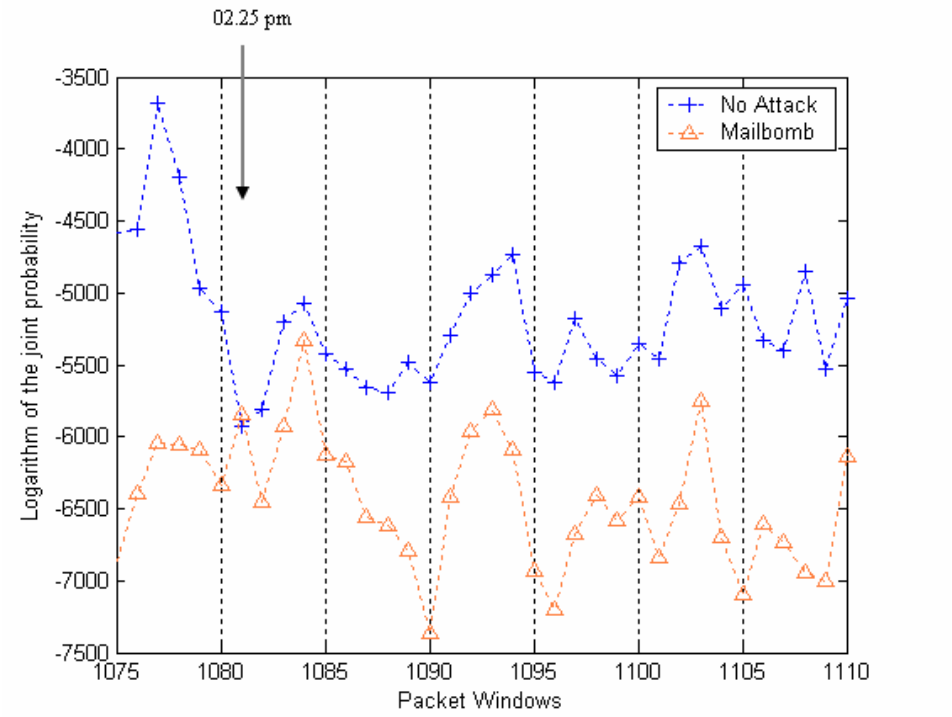

Figure 5.40. Hypothesis testing result between H_0 and H_7 for mailbomb attack instance taking place on Tuesday at week two

As mentioned before, UDP packets with the destination port number 514 stands for the class 1814 at our work and the model built for this attack can be found in the Figure 5.26. Cleary, this class has a small value during the attack instance and it is hard to identify this attack.

In the DARPA 1999 dataset, four instances of the syslogd attack take place. One of them used to train the model and other three instances are employed for testing. The attack instance, which takes place at 02.56 on Friday in the week five, is detected successfully. The Figure 5.41 illustrates the results of the hypothesis testing between H_0 and H_8 hypotheses for this attack instance.

The other instance lasts only one second, therefore it is impossible to detect this attack with our method. Although the last instance doesn't have a short duration, it can not be caught, as well.

As a result, our method is a poor method for detecting the syslogd attack, as the traffic distribution does not differ from the normal traffic for the class 1814.

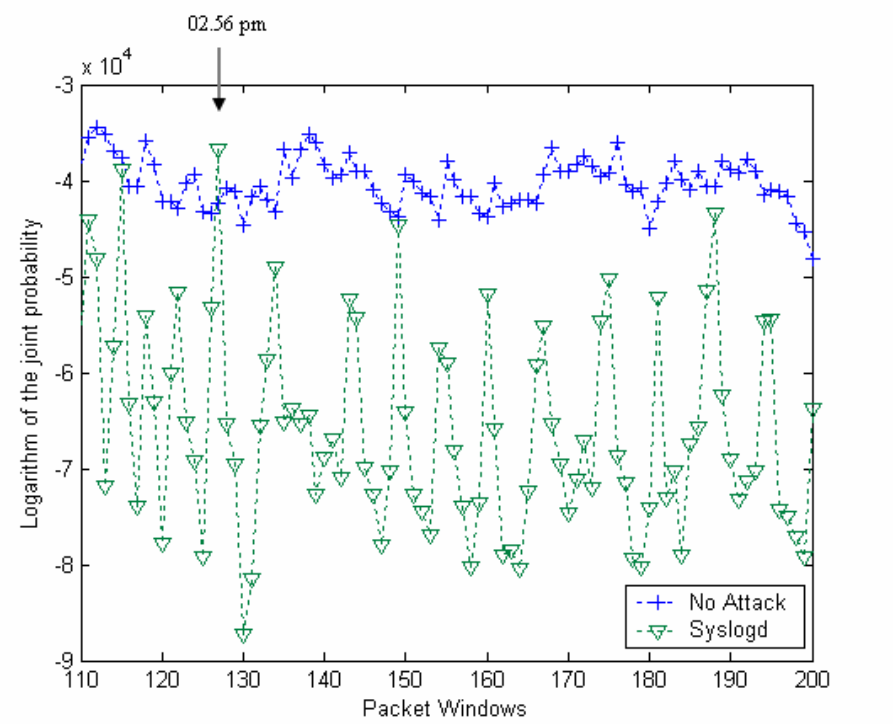

Figure 5.41. Hypothesis testing result between H_0 and H_8 for syslogd attack instance taking place on Friday at week five

5.4 Results

As mentioned before, our proposed method is behavior based and exploits from the header information of the network packets. Therefore, its target attacks require to be modeled evidently by using header information before the detection process. Furthermore, attack instances used for training should have distinctive and invariable characteristics. Moreover, these characteristics should be provided based on the packet header information. For example, an attack using a fixed port and affecting the distribution of the

related port class can be easily detected with our method. However, an attack that has different instances performed through various ports cannot be identified accurately.

It should be underlined that our method raises almost no false alarm. Obviously, the detection rate of our method is very high for the attacks, which have invariable characteristics and affect the traffic distribution of packet classes dramatically. The examples of such attacks are apache2, neptune, udpstorm, mailbomb and tcpreset. However, these attacks may not be identified as well, if their duration is too short or two different attack instances coincide in the same packet window. In defiance of increase in false alarms, such attacks with a short duration can be detected by using a narrower packet window.

Table 5.10. Detection scores of target attacks

Attack Name	Number of Instances in the Test Data	Number of Detection
Apache2	2	2
Back	4	-
Dosnuke	3	1
Neptune	5	4
Udpstorm	1	1
Tcpreset	2	2
Mailbomb	3	3
Syslogd	3	1

In test data, there are some denial of service attacks, which affect the payload carried by packets and are not noticeable only by checking the header information. Back, dosnuke and syslogd are examples of such attacks. Dosnuke and Syslogd attacks abuse fixed ports, but they do not generate high volume traffic through these ports. Since fixed ports are used

in these attacks, there is a chance to catch them. Especially, the detection rate of these attacks can be increased by using narrow packet windows. However, it is almost impossible to detect back attack with our method. During the back attack, malformed requests are destined to port 80. If the number of such requests is not high, no change in the packet distribution is encountered. Moreover, all back instances in our test data have a short duration and the longest one lasts approximately one and half minute. Furthermore, back attack instance performing many malformed requests is hard to detect as well, since the activity of the attacker's port increases due to duplicate ACK packets destined to it and port of the attacker may vary between the attack instances. Therefore, such a back instance possibly may not present same behavior with the trained one.

Table 5.10 summarizes the results of our detection process. As can be seen from this table none of the back attack instances are detected. Evidently, our method is inadequate for identifying back attack, therefore the detection results of the back attack will be ignored, while calculating the performance statistics.

5.5. Performance Analysis

For testing IDS systems, mostly used metrics are true and false positive rates. Two other metrics, namely true and false negative rates, can be easily calculated from the first two metrics.

A true positive (TP) is the probability that an IDS detects an attack when there is already an attack instance at that time. It can be calculated as follows:

$$TP = \frac{Number\ of\ \ Detected\ \ Attack\ Samples}{Number\ of\ \ Attack\ Samples} \tag{5.11}$$

False positive rate (FP), also known as false alarm, is the probability that an IDS raises an alarm when there is no attack thread and it is defined as follows:

$$FP = \frac{Number\ of\ \ False\ \ Alarm\ Samples}{Number\ of\ \ Clean\ Samples} \tag{5.12}$$

False negative rate (FN) is the probability that an IDS does not detect an attack when there is an attack instance at that time. So it can be calculated as follows:

$$FN = 1 - TP \tag{5.13}$$

True negative rate (TN) is the probability that an IDS does not detect an attack and there is no attack threat at that time.

$$TN = 1 - FP \tag{5.14}$$

As mentioned before, performance of our method is dependent on K value, in other words packet window size. Therefore, our detection process is repeated for various values of K, such as 500, 1000 and 10000. So, the performance metrics defined above should be required for these different values of K. The Table 5.11 gives performance metrics for the values of K.

Table 5.11. Performance analysis table-1

Packet Window Size (K)	True Positive Rate	False Positive Rate	False Negative Rate	True Negative Rate
10000	42,11%	4,68%	57,89%	95,32%
1000	68,42%	7,01%	31,58%	92,99%
500	73,68%	8,25%	26,32%	91,75%

Additional to these metrics above, two other useful metrics are the positive predictive value (PPV) and the negative predictive value (NPV). PPV is the probability of an intrusion when the IDS raises an alarm and NPV is the probability of no intrusion when the IDS does not raise an alarm. These metrics are very important, since the IDS alarms are functional to an intrusion response system only if the IDS has high PPV and NPV. Both metrics depend on TP and FP, and are very susceptible to the base rate (B), which is the prior probability of intrusion [38]. These metrics are formulated as follows:

$$Base\ Rate = \frac{Number\ of\ \ Attack\ Samples}{Number\ of\ \ All\ Samples} \tag{5.15}$$

$$PPV = P(I/A) = \frac{B(1-FN)}{B(1-FN)+(1-B)FP} \tag{5.16}$$

$$NPV = P(I'/A') = \frac{(1-B)(1-FP)}{(1-B)(1-FP)+B.FP} \tag{5.17}$$

Furthermore; there is another metric that combines all other metrics of the detection capability; such as, TP, FP, PPV, NPV and B. This metric is called Intrusion Detection Capability, or C_{ID} and it is an information-theoretic measure of the intrusion detection capability. It is defined as the ratio of the mutual information between IDS input and output, and the entropy of the input [38]. The formulation of this metric can be found below and the Table 5.12 summarizes the values of PPV, NPV and C_{ID} for our method.

$$C_{ID} = \frac{I(X;Y)}{H(X)} = \frac{(H(X)-H(X/Y))}{H(X)} \tag{5.18}$$

$$H(X) = -B\log B-(1-B)\log(1-B) \tag{5.19}$$

$$\begin{aligned} H(X/Y) = {} & -B(1-FN)\log\frac{B(1-FN)}{B(1-FN)+(1-B)FP} - B.FN.\log\frac{B.FN}{B.FN+(1-B)(1-FP)} \\ & -(1-B)(1-FP).\log\frac{(1-B)(1-FP)}{(1-B)(1-FP)+B.FN} - (1-B).FP.\log\frac{(1-B)FP}{(1-B).FP+B(1-FN)} \end{aligned} \tag{5.20}$$

Table 5.12. Performance analysis table-2

Packet Window Size (K)	Positive Predictive Values	Negative Predictive Values	Intrusion Detection Capability
10000	50%	99,46%	15,64%
1000	40,63%	99,48%	28,53%
500	36,84%	99,42%	30,11%

6. CONCLUSION AND FUTURE WORK

In this thesis, it is aimed to identify denial-of-service attacks by using maximum entropy and hypothesis testing methods. Packets captured from network are classified according to their protocol, flag and port information. The probability distribution of these packet classes under benign and malignant traffic are estimated based on maximum entropy approach. In this manner, the behavior of the attack free traffic and the behavior of various attack types are modeled. At the detection part, hypothesis testing method is employed for each set of packets captured from network. Consequently, it is decided which of these models most probably satisfies the characteristics of the real-time network traffic.

Our proposed method can be considered as a hybrid of anomaly and misuse detection methods, since it learns not only the characteristics of normal network activity but also the characteristics of the known attacks. Thus, our method overcomes the high degree of false alarms encountered anomaly detection methods. Experimental results show that our method successfully detects the denial-of-service attacks, which have invariable characteristics and affect the traffic distribution of packet classes dramatically. However, our method is inadequate for detecting denial-of-service attacks, which cannot be tracked by using only header information or have variable characteristics.

As a future work, several possible extensions may enhance our proposed method. Firstly, the packet classification algorithm might be changed, so that it gives more importance to the features of target attacks. For example, ignored flag information might be taken into account; also each port used by target attacks might be assigned to the separate classes. By classifying the packets according to the payload information, same method can be applied for attacks, which can be tracked only with payload information.

Another improvement might be taking into account joint probabilities of various attacks. At the training part, joint models can be built for attacks occurring at the same time. At the detection part, if two attacks take place simultaneously, then this case can be identified, as well.

APPENDIX A: SOME DERIVATIONS

APPENDIX A.1. Derivation of Maximum Entropy Estimate

Problem: Find a pmf P; such that the constraints $E_P(f_i) = E_{\tilde{P}}(f_i)$ for all $f_i \in F$ are satisfied and entropy is maximized.

Sketch: Let Ω be the set of packet classes. Given $\tilde{P} = \{\tilde{P}_k\}$, a set of feature functions $F = \{f_i\}$, where the size of $\tilde{P}$ is Ω and the size of F is N. Then, we have N feature functions denoting N constraints.

$$Costra\,\text{int}\, i = \sum_{k=1}^{\Omega} P_k f_i(w_k) = \sum_{k=1}^{\Omega} \tilde{P}_k f_i(w_k) \stackrel{\Delta}{=} C_i \qquad 1 \le i \le N \tag{A.1}$$

Additional to N constraints, we have also one constraint for $\sum_{k=1}^{\Omega} P_k = 1$.

Totally, there are N]1 constraints. We are seeking a pmf P, which satisfy these constraints and maximizes entropy $H(P) = -\sum_{w \in \Omega} P(w) \log P(w)$.

$$M(w) = -\sum_{w \in \Omega} P(w) \log P(w) \tag{A.2}$$

$$L(w) = \sum_{k=1}^{\Omega} P_k = 1 \tag{A.3}$$

$$N_i(w) = \sum_{k=1}^{\Omega} P_k f_i(w_k) = C_i \qquad 1 \le i \le N \tag{A.4}$$

To maximize (A.2) subject to (A.3) and (A.4) Lagrangian multipliers μ and λ can be introduced in the usual way.

$$\begin{aligned} L &= M(w) + \mu L(w) + \sum_{i=1}^{N} \lambda_i N_i(w) \\ &= -\sum_{k=1}^{\Omega} P_k \log P_k + \mu \sum_{k=1}^{\Omega} P_k + \sum_{i=1}^{N} \lambda_i \left(\sum_{k=1}^{\Omega} P_k f_i(w_k) \right) \end{aligned} \tag{A.5}$$

$$\frac{\partial L}{\partial P_k} = -\log P_k + 1 + \mu + \sum_{i=1}^{N} \lambda_i f_i(w_k) = 0 \tag{A.6}$$

which gives:

$$P_k = A \exp\left(\sum_{i=1}^{N} \lambda_i f_i(w_k)\right) \tag{A.7}$$

Equation (A.7) is the generalized Gibbs distribution. Here A can be expressed as follows:

$$A = \frac{1}{Z} \qquad Z = \sum_{k=1}^{\Omega} \exp\left(\sum_{i=1}^{N} \lambda_i f_i(w_k)\right) \tag{A.8}$$

APPENDIX A.2. Derivation of the Gradient of the Log-Likelihood Function

Problem: Show that the gradient of the function $\sum_{w \in \Omega} \tilde{P}(w) \log P(w)$ (log-likelihood function) is $G(\Lambda) = E_{\tilde{P}}[f] - E_P[f]$, where $P(w)$ is expressed as follows:

$$P(w) = \frac{\exp\left(\sum_{i=1}^{N} \lambda_i f_i(w)\right)}{\sum_{w}' \exp\left(\sum_{i=1}^{N} \lambda_i f_i(w)\right)} \tag{A.9}$$

Proof:

$$C = \sum_{w \in \Omega} \tilde{P}(w) \log P(w) \tag{A.10}$$

$$\frac{\partial C}{\partial \lambda_j} = \sum\nolimits_{w}^{'} \tilde{P}(w) \frac{1}{P(w)} \frac{\partial P(w)}{\partial \lambda_j} \tag{A.11}$$

$$P(w) = \frac{A(w)}{\sum\nolimits_{w'}^{'} A(w')} = \frac{A(w)}{S} \tag{A.12}$$

where $A(w) = \exp(\sum_{i=1}^{N} \lambda_i f_i(w))$.

$$\frac{\partial P(w)}{\partial \lambda_j} = \frac{\frac{\partial A(w)}{\partial \lambda_j} S - A(w) \sum\nolimits_{w'}^{'} \frac{\partial A(w')}{\partial \lambda_j}}{S^2} \tag{A.13}$$

where $\frac{\partial A(w)}{\partial \lambda_j} = A(w) f_j(w)$.

$$\begin{aligned} \frac{\partial P(w)}{\partial \lambda_j} &= \frac{\partial A(w) \big/ \partial \lambda_j}{S} - \frac{A(w)}{S^2} \sum\nolimits_{w'}^{'} A(w') f_j(w') \\ &= \frac{A(w) f_j(w)}{S} - \frac{A(w)}{S} \frac{\sum\nolimits_{w'}^{'} A(w') f_j(w')}{\sum\nolimits_{w'}^{'} A(w')} \\ &= P(w) \left[f_j(w) - \frac{\sum\nolimits_{w'}^{'} A(w') f_j(w')}{\sum\nolimits_{w'}^{'} A(w')} \right] \end{aligned} \tag{A.14}$$

By putting Equation (A.14) into the Equation (A.11), Equation (A.15) is obtained:

$$\begin{aligned}\frac{\partial C}{\partial \lambda_j} &= \sum\nolimits_{w}^{'} \tilde{P}(w) \frac{1}{P(w)} P(w) \left[f_j(w) - \frac{\sum\nolimits_{w'}^{'} A(w') f_j(w')}{\sum\nolimits_{w'}^{'} A(w')} \right] \\ &= E_{\tilde{P}}(f_j(w)) - (\sum_{w} \tilde{P(w)}) \frac{\sum\nolimits_{w'}^{'} A(w') f_j(w')}{\sum\nolimits_{w'}^{'} A(w')} \\ &= E_{\tilde{P}}(f_j(w)) - \sum_{u \in \Re} P(u) f_j(u) \\ &= E_{\tilde{P}}(f_j(w)) - E_P(f_j(w))\end{aligned} \tag{A.15}$$

APPENDIX B: FORMAT OF CD CONTAINING SIMULATION CODES

APPENDIX B.1. Simulation Data

As simulation data, DARPA 1999 data have been used. In this CD, two-days data have been added. The data captured on Monday in the week one are an example of training data and the data captured on Thursday in the week two serve as an example of the test data. In order to train all of the models and to test them, the data which are not included in this CD can be downloaded from the web pages of DARPA Intrusion Detection Evaluation [32].

APPENDIX B.2. Hardware and Software Requirements

Simulations have been run on a dual core PC with each CPU speed of 2.13 GHz and 2 GB of RAM. Operating system of this PC is Windows XP.

For writing simulation codes and running them, MATLAB program is exploited. The version of MATLAB is 6.5 and the release of it is 13.

APPENDIX B.3. MATLAB Codes

Three major code groups are in the CD. First group includes MATLAB codes used for reading tcpdump data and calculating empirical distributions. These codes can be found in the folder named Read_Data_Codes. Second group consists of modeling codes. LBFGS MATLAB codes that are written by Curt Vogel originally and modified by me are in the second group, as well. These codes are in the folder named Modelling. In the third group, there are MATLAB codes that are used for detection process. These codes can be found in the folder named Detection_and_hypothesis_testing.

It should be noticed that the modeling codes exploits the outputs of the data reading process and similarly detection codes uses the outputs of the modeling process. Therefore, work spaces of the reading and modeling simulations are saved on the hard disk and then they are loaded to the RAM during the execution of modeling and detection codes. So, this is achieved with the command of loading workspace during execution. These loading commands can be found at the beginning of each modeling and detection codes. It is important that the paths of the saved workspaces are set properly. Two examples of saved workspaces are put in the CD. One is example of a workspace obtained at the end of the process of reading tcpdump file and calculating empirical distribution for no attack case. The other workspace is the output of the maximum entropy modeling process for no attack case. Similarly, for each attack type, two workspaces are saved as output of preprocessing and modeling parts. However, all of the required workspaces for detection part are not included in the CD due to the storage limit of a regular CD. However, these workspaces can be obtained by downloading required data from web pages of DARPA, setting the paths of the required data and workspaces properly in the related MATLAB code files and then running whole simulations.

REFERENCES

1. Delamar, A., *Intrusion Detection with Data Mining*, M.S. Thesis, Donau University, 2002.

2. Bace, R., P. Mell, *NIST Special Publication on Intrusion Detection Systems*, http://www.21cfrpart11.com/files/library/reg_guid_docs/nist_intrusiondetectionsys.pdf, 2007.

3. Kabiri P., A. A. Ghorbani, "Research on Intrusion Detection and Response: A Survey", *International Journal of Network Security*, Vol. 1, No. 2, pp. 84-102, 2005.

4. Kendall, K., *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, M.S. Thesis, Massachusetts Institute of Technology, 1999.

5. Vincent, J., R. Mintram, K. Phalp, C. Anyokoha, *Artificial Techniques Intelligence for Misuse Detection in Telecommunications Environments*, http://www.sosym .co.uk /downloads/publicReports/AI_MDS_review.pdf, 2006.

6. Depren, M.Ö., M. Topallar, E. Anarim, and K. Ciliz, "An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks", *Expert Systems with Applications*, Vol. 29 Issue 4, pp. 713-722, Elsevier Ltd., 2005.

7. Verwoerd, T., R. Hunt, "Intrusion Detection Techniques and Approaches", *Computer Communications*, Vol. 25, Issue 15, pp. 1356-1365, Elsevier Ltd., 2002.

8. Sekar, R., A. Gupta, J. Frullo, T. Shanbhag, S. Zhou, A. Tiwari and H. Yang, "Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions", *ACM Computer and Communication Security Conference*, 2002.

9. Denial of service attacks, CERT. http://cert.org/tech_tips/denial_of_service.html.

10. Kelly, C., D. Spears, C. Karlsson, P. Polyakov, *An Ensemble of Anomay Classifiers for Identifying Cyber Attacks*, http://www.cs.uwyo.edu/~dspears/papers/ensemble. pdf, 2005.

11. Patrikakis C., M. Masikos, O. Zouraraki, "Distributed Denial of Service Attacks", *Internet Protocol Journal*, Vol. 7 Issue 4, pp.13-35, 2004.

12. U.S. Department of Energy, "G-48: TCP SYN flooding and IP spoofing attacks", *Computer Incident Advisory Capability (CIAC) Information Bulletin*, September 20, 1996.

13. "Ping of Death Attack", http://en.wikipedia.org/wiki/Ping_of_death.

14. "Teardrop Attack", http://en.wikipedia.org/wiki/Denial-of-service_attack# Teardrop _attack.

15. Gu, Y., A. McCallum, D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation", *Proceedings of Internet Measurement Conference IMC'05*, pp. 345-350, 2005.

16. Liang, G., B. Yu, and N. Taft, "Maximum entropy models: convergence rates and application in dynamic system monitoring", *International Symposium on Information Theory*, 2004.

17. Yoshida, K. "Entropy based Intrusion Detection", *Communications, Computers and signal Processing*, Vol. 2, pp. 840-843, IEEE, 2003.

18. "Transmission Control Protocol", http://en.wikipedia.org/wiki/Transmission _Con-trol _Protocol.

19. Savoric, M., *Improving Congestion Control in IP-based Networks by Information Sharing,* M.S. Thesis, Berlin Technical University, 2004.

20. Jaynes, E., “Information Theory and Statistical Mechanics”, *Physical Review*, Vol. 106, Issue. 4, pp. 620-630, May 1957.

21. Good, I. J., “Maximum Entropy for Hypothesis Formulation, Especially for Multidimensional Contigency Tables”, *The Annals of Mathematical Statistics*, Vol.34, pp. 911-934, 1963.

22. “Hypothesis Testing”, http://en.wikipedia.org/wiki/Hypothesis_testing.

23. Trees, H. L. V., *Detection, Estimation, and Modulation Theory: Part I,* John Wiley & Sons, New York, 2001.

24. Internet Assigned Numbers Authority, http://www.iana.org/assignments/port-numbers.

25. Argillander, J., *Maximum Entropy Modeling and Semantic Concept Detection*, M.S. Thesis, Helsinki University of Technology, 2005.

26. Pietra, S. D., V. D. Pietra,, and J. Lafferty, “Inducing features of random fields” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 380–393 1997.

27. McCallum, A. “Efficiently inducing features of conditional random fields”, *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

28. Malouf, R., “A comparison of algorithms for maximum entropy parameter estimation”, *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 49-55, 2002.

29. Liu, D. C., J. Nocedal, “On the Limited Memory BFGS Method for Large Scale Optimization”, *Mathematical Programming*, Vol. 45, pp. 503-528, 1989.

30. Nocedal J., “Updating Quasi-Newton Matrices with Limited Storage”, *Mathematics of Computation*, Vol. 35, pp.773-782, 1980.

31. LBFGS MATLAB Codes, http://www.math.montana.edu/~vogel/Courses/M591_2000/M591_codes/lbfgs.

32. DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/.

33. Haines, J. W., R. P. Lippmann, D. J. Fried, E. Tran, S. Boswell, and M. A. Zissman, "1999 DARPA Intrusion Detection System Evaluation: Design and Procedures", *Technical Report 1062*, MIT Lincoln Laboratory, 2001.

34. Abrahamsson, H., *Traffic Measurement and Analysis*, http://www.sics.se /~henrik /t9905.pdf, 1999.

35. "A TCP Tutorial", http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html.

36. SYN Flooding Attacks, CERT.http://www.cert.org/advisories/CA-1996-21.html.

37. Onut, I. V., Ghorbani A.A, "SVision: A novel visual network-anomaly identification technique" , *Computer and Security*, Vol. 26 Issue 3, pp. 201-212, Elsevier Ltd., 2007.

38. Gu G., P. Fogla, D. Dagon, and W. Lee, "An Information-Theoretic Measure of Intrusion Detection Capability", *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

REFERENCES NOT CITED

Depren, M. Ö., *Network Based Intelligent Intrusion Detection System*, M.S. Thesis., Boğaziçi University, 2003.

Jang, K. and S. J. Stolfo, "Anomalous Payload-Based Network Intrusion Detection", *in RAID Symposium*, 2004.

Ji, C., M. Thottan, "Anomaly Detection in Ip Networks", *IEEE Transactions on Signal Processing*, pp. 2191-2204, 2003.

Lee, W., D. Xiang, "Information-Theoretic Measures for Anomaly Detection", *Proceedings of the IEEE Symposium on Security and Privacy*, p.130, IEEE Computer Society, 2001.

Siris, V. A., F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks", *Computer Communications*, Vol. 29, pp. 1433-1442, Elsevier Ltd., 2006.

Wagner, A., B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks", *Proceedings of the 14th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pp. 172-177, 2005.