DETECTING INTRUSIONS IN NETWORK TRAFFIC USING MAXIMUM ENTROPY TECHNIQUE

by

Ömer Faruk Tuna

B.S., Electrical-Electronics Engineering, Boğaziçi 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
2007

DETECTING INTRUSIONS IN NETWORK TRAFFIC
USING MAXIMUM ENTROPY TECHNIQUE

APPROVED BY:

Prof. Dr. Emin Anarım ……
(Thesis Supervisor)

Assist. Prof. Dr. Kıvanç Mıhçak ..……..
(Thesis Co-supervisor)

Assist. Prof. Dr. Burak Acar ..……..

Assist. Prof. Dr. Fatih Alagoz ..……..

Assist. Prof. Dr. Kerem Harmancı ..……..

DATE OF APPROVAL: 17.9.2007

ACKNOWLEDGEMENTS

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

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

The last but not least, I would like to thank my parents. I always felt their support, courage, endless love, and belief on my success.

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

ABSTRACT

DETECTING INTRUSIONS IN NETWORK TRAFFIC USING MAXIMUM ENTROPY TECHNIQUE

As interconnections among computer systems grow rapidly, network security is becoming a major challenge. Computer networks have to be protected against denial-of-service (DoS) attacks, unauthorized disclosure of information and the modification or destruction of data. Besides, the availability, confidentiality and integrity of critical information systems need to be provided. This situation brings the need for Intrusion Detection Systems (IDS) which detects hostile activities or abuse of a network. The objective in this work is to obtain reliable IDS with low false positive rate and high true positive rate.

In this thesis, a refined IDS method is proposed. This method detects intrusions in a network by comparing the current network traffic against a baseline distribution of the benign network traffic. Baseline distribution is obtained from the empirical distribution of benign network traffic by using the Maximum Entropy technique. The distributions here are built on classes which are based on packet and destination port number information. This structure of distributions gives us a multi-dimensional view of the network traffic. Intrusions that change the network traffic slowly or abruptly can then be distinguished by computing a measure related to the relative entropy of the distribution of the network traffic under observation with respect to the baseline distribution. The innovation in this thesis is to reduce the number of classes in a distribution by clustering the classes judiciously in order to reduce the possible negative effects of slow network traffic.

ÖZET

MAKSİMUM DAĞINTI YÖNTEMİ İLE BİLGİSAYAR AĞLARINDAKİ SALDIRILARIN TESPİTİ

Bilgisayar sistemleri arasındaki iletişim ağları hızla genişledikçe, ağ güvenliği ciddi bir tehdit altına girmektedir. Bilgisayar ağları, denial-of-service (DoS) saldırıları, bilgilerin yetki dışı açığa çıkarılması ve yine bilgilerin yetki dışı değiştirilmesi ya da yok edilmesine karşı korunmalı, ayrıca bilgisayar sistemlerinin mevcudiyeti, gizliliği ve bütünlüğü sağlanmalıdır. Bu durum ise ağ içerisindeki düşmanca aktivitelerin ve ağ kaynaklarının kötüye kullanılmasının saptanmasını sağlayan Saldırı Tespit Sistemlerini (IDS) bir ihtiyaç haline getirmektedir. Bu çalışmada Saldırı Tespit Sitemlerinin güvenilirliği ile ilgili aranan temel ölçütler düşük yanlış-pozitif alarmların seyrekliği ve yüksek doğru-pozitif alarmların sıklığıdır.

Bu tezde bir Saldırı Tespit Sistemi tasarımı geliştirilmiştir. Bu yöntem, saldırıların tespitini, mevcut tehdit altındaki ağ trafiğini, atak içermeyen ağ trafiğinin baz olarak alınan dağılımıyla kıyaslayarak gerçekleştirir. Baz dağılım, Maksimum Dağıntı Yöntemi kullanılarak, atak içermeyen ağ trafiğinin deneysel dağılımından elde edilir. Çalışmamızda kullanılan dağılımlar paket ve hedef port bilgilerine dayanan sınıflardan oluşur. Dağılımların bu yapısı bize ağ trafiği hakkında çok boyutlu bir bakış açısı sağlar. Ağ trafiğini yavaşça ya da aniden değiştiren saldırılar, gözlem altındaki ağ trafiğinin baz dağılımına göre göreceli dağıntı ölçütünün hesaplanması ile tespit edilir. Bu tezdeki yenilik, dağılımlardaki sınıf sayısını kümeleyerek uygun bir şekilde azaltmak ve yavaş ağ trafiği altındaki olası kötü sonuçları azaltmaktır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

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
$L(\{\lambda_i\})$	Log-Likelihood Function
Z	Normalization Constant
λ_i	Parameter
CPU	Central Processing Unit
DARPA	Defense Advanced Research Project Agency
DDoS	Distributed Denial of Service
DoS	Denial of Service
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
IDS	Intrusion Detection System
IIS	Internet Information Server
IP	Internet Protocol
ISO	International Organisation for Standardisation
IRC	Internet Relay Chat
LAN	Local Area Network
LBFGS	Limited Memory Variable Metric Method
LLC	Logical Link Control
MAC	Media Access Control

NT	Network
OS	Operating System
OSI	Open System Architecture
PC	Personal Computer
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

Malicious abuses of Internet are commonly seen in today's network traffic. Intrusions such as worms, port scans, denial of service attacks, etc. can be found at any time in the network traffic. These intrusions waste network resources, cause performance degradation of network devices and end hosts, and lead to security issues concerning all Internet users. Thus, accurately detecting such intrusions has become an important problem for the network community to solve [1].

1.1. Network Security

Networks are complex interacting systems and are comprised of several entities such as end hosts, routers and switches. The behavior of the individual entities contributes to the ensemble behavior of the network. The evolving nature of internet protocol (IP) makes it difficult to fully understand the dynamics of the system. To obtain a basic understanding of the performance and behavior of these complex networks, vast amounts of information need to be collected and processed. In most of the cases, network performance information is not directly available, and the information obtained must be synthesized to obtain an understanding of the ensemble behavior [4].

The approaches used to address the anomaly detection problem are dependent on the nature of the data that is available for analysis. Network data can be obtained at multiple levels of granularity such as end-user-based or network based. Examples of the end-user information which is used in this thesis work are transmission control protocol (TCP) and user datagram protocol (UDP) related data that contains information that is specific to the end application. Network-based data pertains to the functioning of the network devices themselves and includes information gathered from routers. Traffic information obtained from both types of data can be used to processed for intrusion detection [4].

1.1.1. What is an intrusion detection system?

Intrusion detection is the process of monitoring networks for unauthorized access, activity or file modification. Intrusion Detection Systems can also be used to monitor network traffic, thereby detecting if system is targeted by an attack [2]. In other words, an Intrusion Detection System is a defense system, which detects hostile activities in or exploit of a network. The objective is then to detect and if possible prevent activities that may compromise system security or hacking attempt in progress. One key feature of intrusion detection systems is their ability to provide a view of unusual activity and issue alerts notifying the administrator [2].

Considering the source of data used for intrusion detection, classification of IDS can be made in terms of the type of the protected system. According to resources they monitor, there are two types of IDS: Host based IDS and (HIDS) and Network based IDS (NIDS) [3]. Host based IDS are installed locally on host machines and continuously monitor log activities in real time. Network based IDS use raw network packets as the data source and inspects packets passing through the network

According to the detection model, IDS can be divided into two parts: Misuse Detection and Anomaly Detection. Misuse detection systems try to match computer activity to stored signatures of known exploits or attacks. In other words, misuse detection systems use a priori knowledge on attacks to look for attack traces. Anomaly detection is an approach to detect intrusions by first learning the characteristics of normal activity. Then systems are designed to detect anything that deviates from normal network activity [3].

Our study makes use of network packet (TCP, UDP) headers as the data source and can be classified as Network Based IDS based on the location of the IDS system. In addition, according to detection model, it can be regarded as Anomaly Detection method.

1.2. Attack Definitions and Types

An attack or security incident is defined as a threat, intrusion, denial-of-service, or other attack on network infrastructure, computer system(s), or user account(s). There are so

many types of attacks in today's world and it is virtually impossible to explain all attack types here. Below is a list of different types of attacks and abuses that are detectable (sometimes hardly detectable) by IDS tools [5].

Those related to the unauthorized access to the resources:

- Password cracking and access violation
- Trojan horses
- Unauthorized network connection
- Taking advantage of system weakness to gain root access to resources or privileges in unix systems
- Network packet listening
- Stealing information, for example disclosure of proprietary information
- Interceptions; most frequently associated with TCP/IP stealing and interceptions that often employ additional mechanisms to compromise operation of attacked systems
- Spoofing (i.e. DNS spoofing)
- Scanning ports and services

Unauthorized alteration of resources (after gaining unauthorized access):

- Falsification of identity, for example to get system administrator rights,
- Information altering and deletion
- Unauthorized configuration changes to systems and network services (servers)
- Unauthorized transmission or creation of data

Denial of Service (DoS):

- Ping flood (Smurf) – a large number of ICMP packets sent to a broadcast address
- Send mail flood (Mailbomb) – flooding with hundreds of thousands of messages in a short period of time

- SYN flood – initiating huge amounts of TCP requests and not completing handshake mechanisms as required by the TCP protocol
- Causing network congestion and slowdown i.e. Udpstorm
- Distributed Denial of Service; coming from a multiple source
- Buffer Overflow, for example Ping of Death – sending a very large ICMP
- Remote system shutdown, reboot i.e. Teardrop

This work mainly focuses on the type of attacks that can be detected by transport control protocol (TCP) and user datagram protocol (UDP). From the above list, only the attack types that cause significant change in the TCP/IP network traffic statistics can be detected with this study. Because of this reason, denial of service (DoS) attacks will be our main focus for detection.

1.3. Denial of Service Attacks

A "denial-of-service" attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. Examples include:

- Attempts to flood a network, thereby preventing legitimate network traffic
- Attempts to disrupt connections between two machines, thereby preventing access to a service
- Attempts to prevent a particular individual from accessing a service
- Attempts to disrupt service to a specific system or person

Denial of service attacks come in variety of forms and aim at variety of services. There are three basic types of attacks: [6]

- Consumption of scarce, limited, or non-renewable resources
- Destruction or alteration of construction information
- Physical destruction or alteration of network components

1.3.1. Consumption of scarce resources

Computers and networks require certain resources to operate smoothly and continuously: network bandwidth, memory and disk space, CPU time, data structures, access to other computers and networks, and certain environmental resources such as power, cool and air.

Denial-of-service attacks are most frequently executed against network connectivity. The goal is to prevent hosts or networks from communicating on the network. A common example of this type of attack is the “SYN flood” attack. In this type of attack, the attacker begins the process of establishing a connection to the victim machine, but does it in such a way as to prevent the ultimate completion of the connection. In the mean time, the victim machine has reserved one of a limited number of data structures required to complete the impending connection. The result is that legitimate connections are denied while the victim machine is waiting to complete “half-open” connections.

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. The result is that the two services consume all available network bandwidth between them. Thus, the network connectivity for all machines on the same networks as either of the targeted machines may be affected.

An intruder may also be able to consume all the available bandwidth on your network by generating a large number of packets directed to your network. Typically these packets are ICMP ECHO packets, but in principle may be anything.

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 (Mailbomb attack).

1.3.2. Destruction or alteration of configuration information

An improperly configured computer may not perform well or may not operate at all. An intruder may be able to alter or destroy configuration information that prevents you from using your computer or network. For example, if an intruder can change the routing information in your routers, your network may be disabled.

1.3.3. Physical destruction or alteration of network components

The primary concern with this type of attack is physical security. Computers, routers, network wiring closets, network backbone segments, power and cooling stations and any other critical components of network must be guarded against unauthorized access.

1.4. TCP/IP Protocol Overview

The TCP/IP protocol architecture is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defense Advanced Research Projects Agency (DARPA), and is generally referred to as the TCP/IP protocol suite. The name TCP/IP refers to a suite of data communication protocols. The name may be misleading because TCP and IP are only two of dozens of protocols that compose the suite data communication protocols [9]. From the TCP/IP protocol suite, TCP and UDP protocols are used in this thesis.

1.4.1. TCP/IP layers

The TCP/IP model organizes the communication task into five relatively independent layers:

- Physical layer
- Network access layer
- Internet layer
- Host-to-host, or transport layer
- Application layer

The physical layer covers the physical interface between a data transmission device (e.g workstation, computer) and a transmission medium or network. This layer is concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, and related matters.

The network access layer is concerned with the exchange of data between an end system (server, workstation, etc.) and the network to which it is attached. The sending computer must provide the network with the address of the destination computer, so that the network may route the data to the appropriate destination. The sending computer may wish to invoke certain services, such as priority, that might be provided by the network. The spesific software used at this layer depends on the type of network to be used; different standards have been developed for circuit switching, packet switching (e.g. frame relay), LANs (e.g. Ethernet), and others.

The network access layer is concerned with access to and routing data across a network for two end systems attached to the same network. In those cases where two devices are attached to different networks, procedures are needed to allow data to traverse multiple interconnected networks. This is the function of the internet layer. The internet protocol (IP) is used at this layer to provide the routing function across multiple networks.

Regardless of the nature of the applications that are exchanging data, there is usually a requirement that data be exchanged reliably. That is, we would like to be assured that all of the data arrive at the destination application and that the data arrive in the same order in

which they were sent. The mechanisms for providing reliability are essentially independent of the nature of the applications. So, those mechanisms are collected in a common layer shared by all applications which is referred as the host-to-host layer, or transport layer.

The application layer contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.

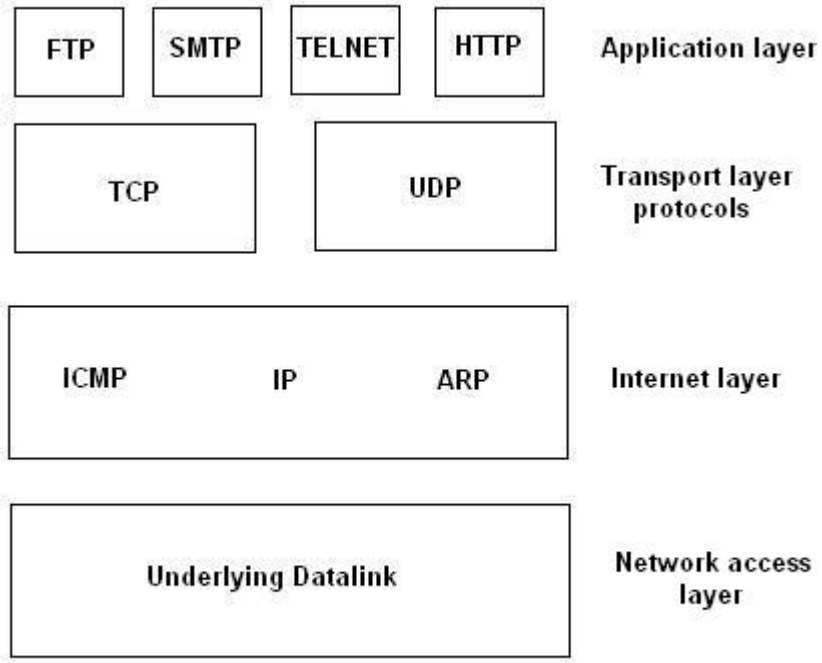

Figure 1.1. TCP/IP protocol layers

1.4.2. Transport control protocol (TCP)

TCP is one of the core protocols of the TCP/IP protocol suite. Using TCP, applications on network hosts can create connections to one another over which they can exchange streams of data [10]. This protocol provides connection-oriented, reliable, full-duplex and byte-stream delivery of data from sender to receiver. Connection oriented means that a virtual connection is established before any user data is transferred. If the

connection is not established, the user program is notified. If the connection is interrupted, again the user program is notified. Reliable means that every transmission of data is acknowledged by the receiver. If the sender does not receive acknowledgement within a specified amount of time, the sender retransmits the data. Stream means that the connection is treated as a stream of bytes. The user application does not need to package data in individual datagrams. And finally full-duplex means that TCP provides transfer in both directions.

1.4.3. User datagram protocol (UDP)

UDP is also an important part of TCP/IP protocol suite. Using UDP, programs on networked computers can send short messages known as datagrams to one another. UDP does not guarantee reliability or ordering in the way that TCP does. Datagrams may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient, at least for applications that do not need guaranteed delivery. Time sensitive applications often use UDP because dropped packets are preferable to delayed UDP packets. UDP's stateless nature is also useful for servers that answer small queries from huge number of clients. Unlike TCP, UDP supports packet broadcast (sending to all on local network) and multicasting (send to all subscribers) [11].

1.4.4. Packet header structures

Transport Control Protocol (TCP) and User Datagram Protocol (UDP) are the two protocol types which are used in this thesis. The information hidden in the headers of TCP packets (Destination Port Number and Status Flag Information) and the information hidden in the UDP packets (Destination Port Number) are important for the classification of the network traffic. Therefore, it is better to understand the header structures of TCP and UDP packets.

TCP data is encapsulated in IP datagram. Below figure shows us the format of the TCP header. Its normal size is 20 bytes unless options are present. Destination port and status flag information are hidden in the TCP header. The 6 control bits (URG, ACK, PSH,

RST, SYN, FIN) reveals the status flag information and from these bits, we are concerned with only SYN and RST. SYN means synchronize sequence numbers and RST means to reset the connection.

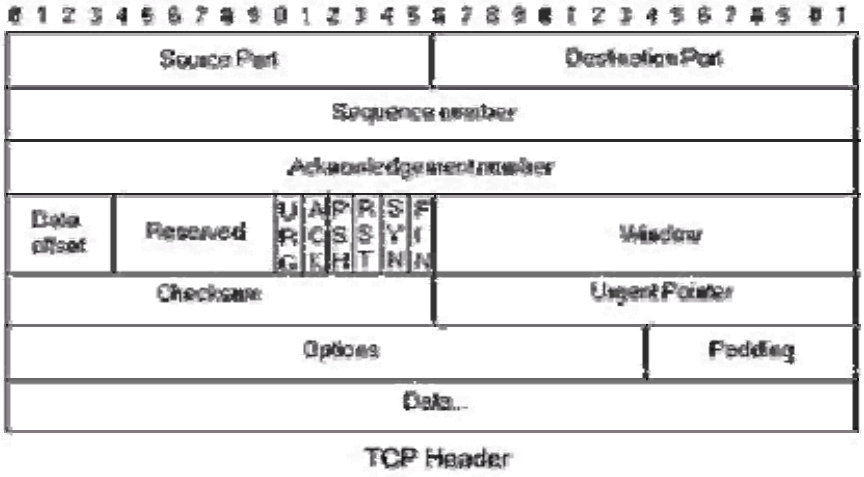

Figure 1.2. TCP header format

From Figure 1.3, it can be seen that UDP header is much smaller than the TCP header. The UDP header can be said to contain a very basic and simplified TCP header. It contains destination and source ports, header length, checksum and data fields.

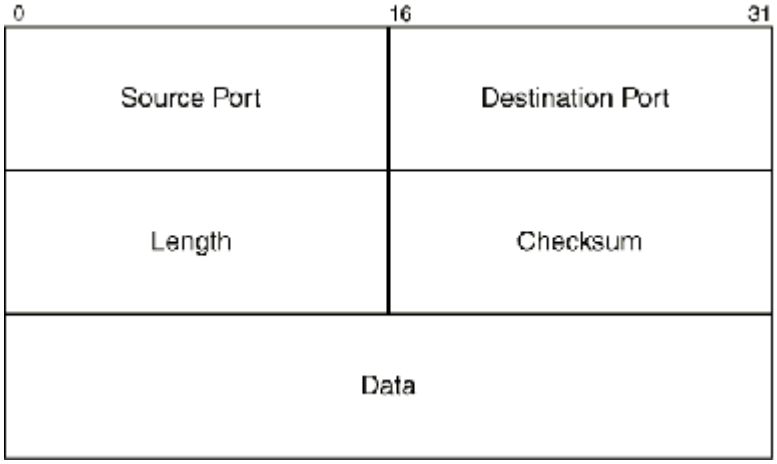

Figure 1.3. UDP header format

1.5. Motivation

The goal of this work is to show the potential to apply maximum entropy technique to the problem of intrusion detection. Application of this technique will provide better insight for improving existing detection tools. Because this technique provides many

advantages for the administrator. First, it gives the administrator a multi-dimensional view of the network traffic. Second, not only can it detect the type of intrusions that cause abrupt changes in the network traffic, it can also detect intrusions that increase the traffic slowly. Third, it provides information about the type of anomaly detected. Fourth, it only requires a constant amount of memory and consists solely of counting packets in the traffic. Finally deploying this approach is also easy due to simple computation needed to check for the anomalies.

1.6. Thesis Outline

The rest of the thesis is organized in the following fashion. Section two makes introduction to Intrusion Detection Methods. In section three, Maximum Entropy technique is explained. In section four, topology of simulation network (1999 Darpa Dataset) has been explained. In section five, results of Maximum Entropy technique has been showed. Section six concludes the thesis.

2. INTRUSION DETECTION METHODS

In this section, we review the most commonly used intrusion detection methods. The methods described are rule-based approaches, change detection methods, Wavelet Models, and statistical analysis.

2.1. Rule Based Models

Early work in the area of fault or intrusion detection was based on expert systems. In expert systems, an exhaustive database containing the rules of behavior of the faulty system is used to determine if an intrusion is occurred. Rule-based systems are too slow for real-time applications and are dependent on prior knowledge about the intrusion conditions on the network. The identification of intrusions in this approach depends on symptoms that are specific to a particular intrusion. Examples of these symptoms are excessive utilization of bandwidth, number of open TCP connections etc [4].

A variety of tools have been developed for the purpose of network anomaly detection. Some detect anomalies by matching the traffic pattern or the packets using a set of predefined rules that describe characteristics of the anomalies. Examples of this include many of the rules or policies used in snort [12]. The cost of applying these approaches is proportional to the size of the rule set as well as the complexity of the individual rules, which affects the scalability of these rules approaches. Furthermore they are not sensitive to anomalies that have not been previously defined [1].

2.2. Pattern Matching Models

SNORT is an open source Intrusion Detection System that relies heavily on the Aho-Corasick multi-pattern search engine. The performance characteristics of the Aho-Corasick algorithm implemented in Snort have a significant impact on the overall performance of Snort. Snort scans network traffic packets searching for intruders by looking for specific values in the network headers and by performing a search for known

patterns in the application layer data. Norton et al [13] presents an optimized version of Aho-Corasick algorithm. The enhanced design uses an optimized vector implementation of the Aho-Corasick state table that significantly improves performance. A memory efficient variant uses sparse matrix storage to reduce memory requirements and further improve performance on large pattern groups.

Another approach proposed and implemented by *Maxion et al* [14] describes anomalies as deviations from normal behavior. This approach attempts to deal with the variability in the network environment. In this approach, online learning is used to built using symptom-specific feature vectors such as link utilization, packet loss, and number of collisions. These profiles are then categorized by time of day, day of week, and special days, such as weekends and holidays. When newly acquired data fails to fit within some confidence interval, it is regarded as anomalous.

2.3. Change Detection Models

A number of existing approaches are variations on the change detection method. In [15], Brutlag uses the Holt Winter forecasting model to capture the history of the network traffic variations and to predict the future traffic rate in the form of a confidence band. When the variance of the network traffic continues to fall outside of the confidence band, an alarm is raised.

Wang et al present a simple mechanism, called Change-Point Monitoring (CPM), to detect denial of service (DoS) attacks. CPM compares the observed sequence with the profile that represents the user's normal behavior and detects any significant deviation from the normal behavior. The key difference of CPM from others is that CPM exploits the inherent network protocol behaviors, instead of traffic patterns, for detecting network anomalies. The objective of Change-Point Detection is to determine if the observed time series is statistically homogeneous and, if not, to find the point in time when the change happens [16].

2.4. Wavelet Models

In [17], *Barford et al* use wavelet analysis of flows exported from routers. It was found that an increase in the local variance of a time-series signal that is generated from raw traffic strongly indicated an anomaly. Wavelet analysis on a raw traffic signal allows for observation on many different levels of traffic and studying the variations in network traffic, by removing certain components of the signal at each level, generating wavelet coefficients. This feature extraction at different levels is known as Multi-Resolution Analysis (MRA), and has gained popularity as the method of analysis for non-stationary signals. Network anomalies are detected by applying a threshold to a deviation score computed from the analysis

In [18], *Huang et al* apply signal processing techniques in intrusion detection systems, and develop and implement a framework, called Waveman, for real time wavelet-based analysis of network traffic anomalies. Then, they use two metrics, namely percentage deviation and entropy, to evaluate the performance of various wavelet functions on detecting different types of anomalies like Denial of Service (DoS) attacks and port scans. Results of their evaluations show that Coiflet and Paul wavelets perform better than other wavelets in detecting most anomalies.

In [19], *Rawat et al* have proposed a method for anomaly based network intrusion detection inspired from the methods that use the self-similarity property of data network traffic as normal behavior and any deviation from it as the anomalous behavior. This method determines both the possible presence of an anomaly and its location in the data by making use of the relations present among the wavelet coefficients of a self-similar function in a different way. Empirical results are provided on KDD data

2.5. Payload Based Models

Wang et al present a payload-based anomaly detector called PAYL for intrusion detection [20]. PAYL models the application payload of network traffic in a fully automatic fashion. The detector may be deployed in high bandwidth environments either on a firewall, a network appliance, a proxy server or on the target hosts. During a training phase, they first compute a profile byte frequency distribution and their standard deviation

of the application payload flowing to a single host and port. They then use Mahalanobis distance during the detection phase to calculate the similarity of new data against the pre-computed profile. The detector compares this measure against a threshold and generates an alert when the distance of the new input exceeds this threshold.

Bolzoni et al [21] proposed an anomaly based network intrusion detection system called POSEIDON. It is payload-based, and has two-tier architecture. The first stage consists of a Self-Organizing Map (SOM) which is used exclusively to classify the payload data, while the second one is a modified PAYL system. POSEIDON is packet oriented and uses only destination address and service port numbers to build a profile for each port monitored. It does not consider other header features. The main advantage of POSEIDON is that it has intrusion prevention capability. That is, it can identify and block an attack while it is taking place.

2.6. Neural Network Based Models

A system developed by *Lichodzijewski et al* [22] applies hierarchical unsupervised neural networks (Self-Organizing Feature Maps) to the intruder detection problem. Specific emphasis is given to the representation of time and the incremental development of the hierarchy. In their study, they first try to identify the characteristics of the normal connection to the target host. This information is then used to raise a flag for any connection identified as having a different characteristic.

Depren et al [3] proposed a hybrid approach which combines misused and anomaly detection methods. This hybrid Intrusion Detection System architecture consists of an anomaly detection module, a misuse detection module and a decision support system combining the results of these two detection modules. The proposed anomaly detection module uses a Self-Organizing Map (SOM) structure to model normal behavior. Deviation from the normal behavior is classified as an attack. And the proposed misuse detection module uses J.48 decision tree algorithm to classify various types of attacks. They have used KDD Cup Data Set to verify the performance of their method.

Rhodes et al [23] uses multiple self-organizing maps for intrusion detection. They use a collection of more specialized maps to process network traffic for each layered

protocol separately. They suggest that each neural network becomes a kind of a specialist, trained to recognize the normal activity of a single protocol. After learning the characteristics of normal network traffic or user behavior, these networks can identify abnormalities without relying on expectations of what abuse will look like.

Moradi at al [24] proposes a neural network approach to intrusion detection. A Multi Layer Perceptron (MLP) is used for intrusion detection based on offline analysis approach. While most of the previous studies have focused on classification of records in one of the two general classes – normal and attack, this research aims to solve a multi class problem in which the type of attack is also detected by the neural network. And in their study, different neural network structures are analyzed to find the optimal neural network with regards to the number of hidden layers. Also an early stopping validation method is also applied in the training phase to increase the generalization capability of the neural network.

2.7. Principal Component Analysis (PCA) Models

Wang et al proposed a method for intrusion identification in computer networks based on Principal Component Analysis (PCA). Each network connection is transformed into an input data vector. PCA is employed to reduce the high dimensional data vectors and identification is handled in a low dimensional space with high efficiency and low use of system resources. The normal behavior is profiled based on normal data for anomaly detection and the behavior of each type of attack are built based on attack data for intrusion detection. The distance between a vector and its reconstruction onto those reduced subspaces representing different types of attacks and normal activities is used for identification. They have tested their study with network data from MIT Lincoln labs for the 1998 DARPA Intrusion Detection Evaluation Program [25].

Labib et al [26] have presented a multivariate statistical method called Principal Component Analysis (PCA) is used to detect Denial-of-Service attacks using the 1998 DARPA data set. Visualization of network activity and possible intrusions is achieved using Bi-plots, which are used as a graphical means for summarizing the statistics. The principal components are calculated for both attack and normal traffic, and the loading values of the various feature vector components are analyzed with respect to the principal

components. The variance and standard deviation of the principal components are calculated and analyzed. A brief introduction to Principal Component Analysis and the merits of using it for detecting the selected intrusions are discussed.

2.8. Image Based Intrusion Detection

A system (NetViewer) developed by *Kim et al*, presents a network measurement approach that can simultaneously detect, identify and visualize attacks and anomalous traffic in real-time by passively monitoring packet headers. They propose to represent samples of network packet header data as frames or images. With such formulation, a series of samples can be seen as a sequence of frames or video, revealing certain kinds of attacks to the human eye. This enables techniques from image processing and video compression to be applied to the packet header data to reveal interesting properties of traffic. In their study, they show that "scene change analysis" can reveal sudden changes in traffic behavior or anomalies. They also show that "motion prediction" techniques can be employed to understand the patterns of some of the attacks. It is stated that it may be feasible to represent multiple pieces of data as different colors of an image enabling a uniform treatment of multidimensional packet header data [27].

2.9. Statistical Analysis Models

There are no accurate statistical models for normal network operation, and this makes it difficult to characterize the statistical behavior of abnormal traffic patterns. In [4], *Thottan et al* describes in detail a statistical signal processing technique based on abrupt change detection for addressing this challenge. Furthermore, there is no single metric that captures all aspects of normal network function. This presents the problem of synthesizing information from multiple metrics, each of which has widely different statistical properties. To address this issue, they use an operator matrix to correlate information from individual metrics. 3 of the SNMP MIB variables (metrics) are used in this study. Their study shows that this signal processing technique is effective at detecting several network anomalies.

Wu et al presents a method for detecting network anomalies by analyzing abrupt change of time series data obtained from Management Information Base (MIB) variables. The method applies the Auto Regressive (AR) process to model the abrupt change of time

series data, and perform sequential hypothesis test to detect the anomalies. With time correlation and location correlation, the method determines not only the presence of anomalous activity, but also its occurring time and location [28].

In [29], Soule et al develops an approach for anomaly detection for large scale networks such as an enterprise or internet service provider. The traffic pattern they focus on for analysis are that of a network-wide view of the traffic state, called the traffic matrix. In the first step, they used a Kalman filter to filter out the normal traffic. This is done by combining their future predictions of the traffic matrix state to an inference of the actual traffic matrix that is made using more recent measurement data than those used for prediction. In the second step the residual filtered process is then examined for anomalies. They study and compare four different methods for analyzing residuals. These methods focus on different aspects of the traffic pattern change. One focuses on instantaneous behavior, another focuses on changes in the mean of the residual process, a third on changes in the variance behavior, and a fourth examines variance changes over multiple timescales.

2.10. Comments on Related Works

Signature based models and pattern matching models can not detect newly designed attacks because of lack of information of new attacks types. They need continuous updates to catch up to date attacks. Rule based systems suffer from the identification of relevant criteria for the different faults which requires a very large set of rules to be developed. In addition, using any functional approximation scheme, such as back propagation, causes an increase in computation time and complexity. The number of functions to be learned also increases with the number of faults studied.

We have performed a behavior based intrusion detection approach based on the analysis of TCP/IP network traffic. This work shows us that applying maximum entropy technique combined with a clustering mechanism on the baseline distribution gives us good results for intrusion detection. Furthermore, it provides constant memory and computational time proportional to the traffic rate.

3. SIMULATION NETWORK AND DATA

In my thesis, I have used 1999 DARPA Intrusion Detection Evaluation Corpora. 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) sponsorship has collected and distributed these standard corpora for evaluation of computer network intrusion detection systems. [30]

In any IDS performance verification, there are three major principals of evaluation. First, a stand-alone network testbed must be used to generate an evaluation corpus. Second, intrusion system performance must be measured using both attack detection rate and false alarm rate. Finally, an off-line evaluation format must be used.

3.1. Testbed Approach

Evaluation of IDS requires creation of a corpus for measuring both detection and false alarm rates. In the literature, three approaches were explored to generate a corpus that could be widely distributed and included both background traffic and attacks. The first proposal was to capture operational traffic during normal operations and with controlled live attacks against selected components of the operational network. But this approach has an important disadvantage, because it would release private information, compromise security, and possible damage systems or interrupt important network services. A second proposal was to sanitize sampled operational data and insert attacks into the sanitized data. This was not possible because of removing all security-related and private information from network traffic and the complexity of inserting attacks without introducing artifacts. Sanitization would require examining every email message, every process like file transfer, and eliminate private or confidential information. It would also require changing all IP addresses, user names and passwords. The final proposal, which was used in 1999 DARPA Intrusion Detection Evaluation Corpora, was to recreate normal and attack traffic on a private network using real hosts, live attacks, and live background traffic.

The three main components of the testbed approach are the network testbed, the background (or normal) traffic, and the attacks. An overview is given in below figure. The three major inputs in generating the 1999 corpora are synthetically modeled background traffic, attacks, and the testbed network on which the simulation is run and data collected.

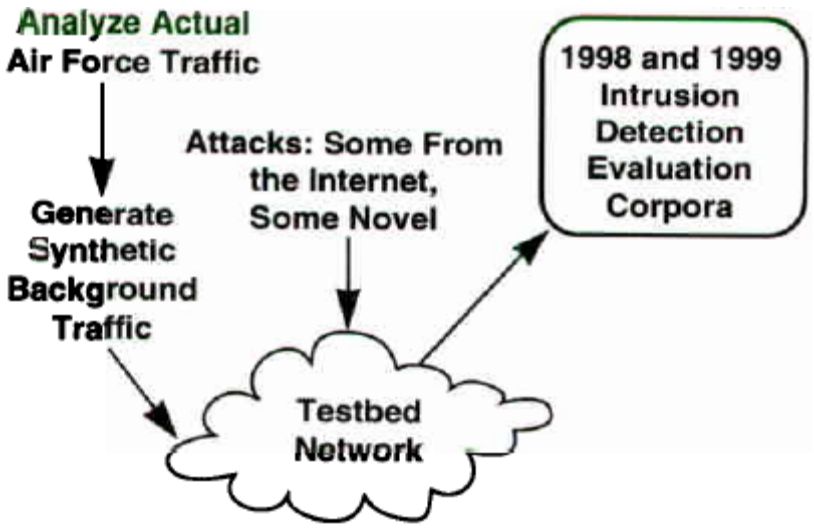

Figure 3.1. Three major components required to generate the 1999 corpora [8]

Synthetic background traffic is generated by modeling both traffic collected at Airforce Base and traffic statistics collected at many other bases. Attacks and scripts to carry out the attacks are collected from the Internet or developed independently and run on the testbed network in conjunction with background traffic [31].

3.2. Analysis Using Detection and False Alarm Rates

1998 and 1999 evaluations include realistic background traffic to measure false alarm rates. Most prior evaluations only measured detection rate with a small subset of attacks. Attack detection rate and false alarm rate, together, provide the best quantification of IDS performance. Measuring the detection rate alone only indicates the types of attacks that an IDS may detect. Such measurements do not indicate the human workload required to analyze false alarms generated by normal background traffic. False alarm rates above hundreds per day make a system almost unusable, even with high detection accuracy, because observed detections or alerts can not be believed and thus security analysts must

spend many hours each day dismissing false alarms. Low false alarm rates combined with high detection rates means that the outputs can be trusted and the human labor required to confirm detections is minimized [31].

3.3. Off-line vs. Real-time Evaluations

Two main advantages of a real-time, on-line evaluation are that intrusion detection systems are able to perform active rather than passive monitoring during the evaluation and they can take action in response to a particular attack or a possible detection. Another advantage is that a real-time evaluation can measure practical characteristics of an intrusion detction system such as CPU and memory usage, and ease of installation and configuration. The main disadvantage of the real-time evaluation is that in many cases only one intrusion detection system can be evaluated at a time and attacks and background traffic must be regenerated for each system evaluated. Therefore it may be very time-consuming process and may not be well suited to training (as opposed to testing) IDS. In contrast, corpora for the off-line evaluation are produced once and can be used by a number of systems at any time for evaluation or training [31].

In the off-line evaluation, systems are tested using network traffic collected on a testbed network and distributed to evaluation participants. Systems process this data and attempt to identify attack actions in the midst of normal activities. This off-line approach has several advantages. The evaluation allows more developers to participant unlike to the real-time evaluation. Besides, the off-line technique focuses on technology development rather than complete system development by removing the need to deliver a mature system for installation in the testbed network, making it possible for a developer to evaluate algorithms and techniques in the absence of the complete system. And that is actually what I do in my thesis work.

3.4. Summary of the 1998 DARPA Off-line Evaluation

This section summarizes DARPA off-line evaluation which took place in 1998. Figure 3.2 shows the network testbed. This testbed created live traffic similar to traffic that flows between a small Air Force base and the Internet. A rich variety of background traffic was generated that appeared to be initiated by hundreds of users on thousands of hosts. The

left side of the figure represents the “inside” network, the fictional Eyrie Air Force Base, and the right side represents the “outside” network, the Internet. The 1998 evaluation focuses on UNIX attacks launched from remote machines. Automated attacks were launched against three inside UNIX victim hosts running SUNOS, Solaris, and Linux, respectively.

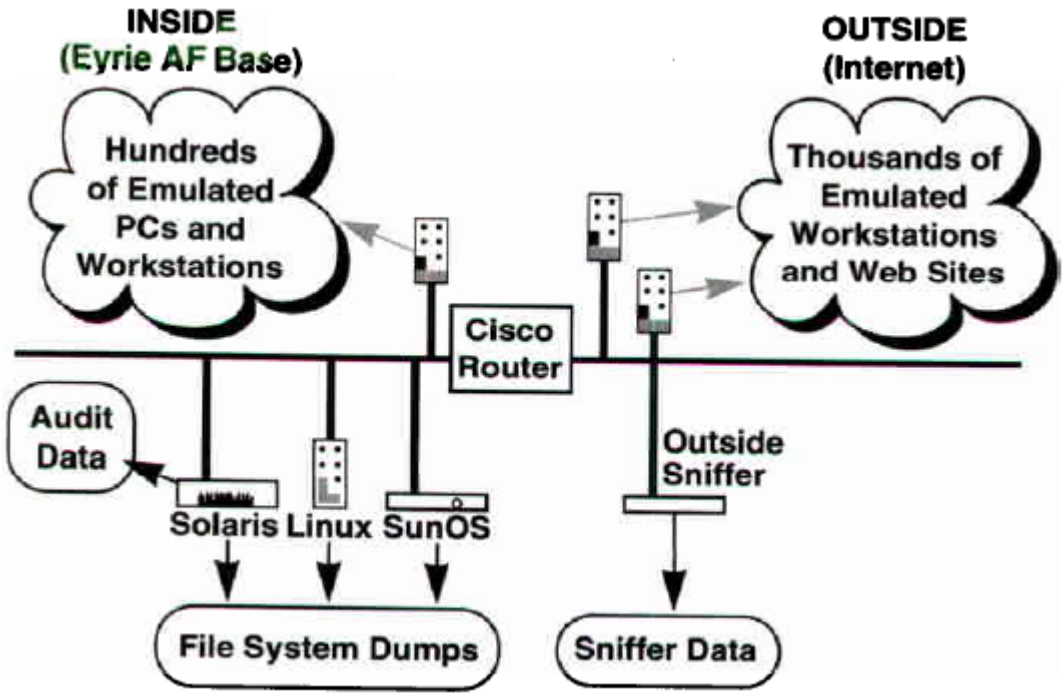

Figure 3.2. Testbed network used in the 1998 off-line evaluation [8]

3.5. Changes from 1998 to 1999

The testbed network in 1998 primarily emphasized security of UNIX hosts and UNIX-based intrusion detection systems. For 1999, the evaluation reacted to the deployment of Windows NT in many sectors of government and included attacks against Windows NT and attacks launched from NT hosts. A Windows NT Microsoft Internet Information Server (IIS) and mail server was added to the simulation network.

And also, in response to the developer requests, two weeks of the training data containing only background traffic, and, no attacks, were distributed so anomaly detection systems could be trained on large samples of normal traffic. However, no matter how diverse the simulated background traffic is, it can never match real-world traffic. Because

real-world traffic varies substantially both over time and across sites and it would be too costly to model this temporal variability in the evaluation testbed.

3.6. Network Testbed in 1999

Figure 3.3 shows the conceptual view of the evaluation testbed used in 1999. The testbed generates live network and host traffic similar to that seen on one small Air Force base and between that base and the Internet. Attacks can be automated and run on the testbed against the Air Force Base servers.

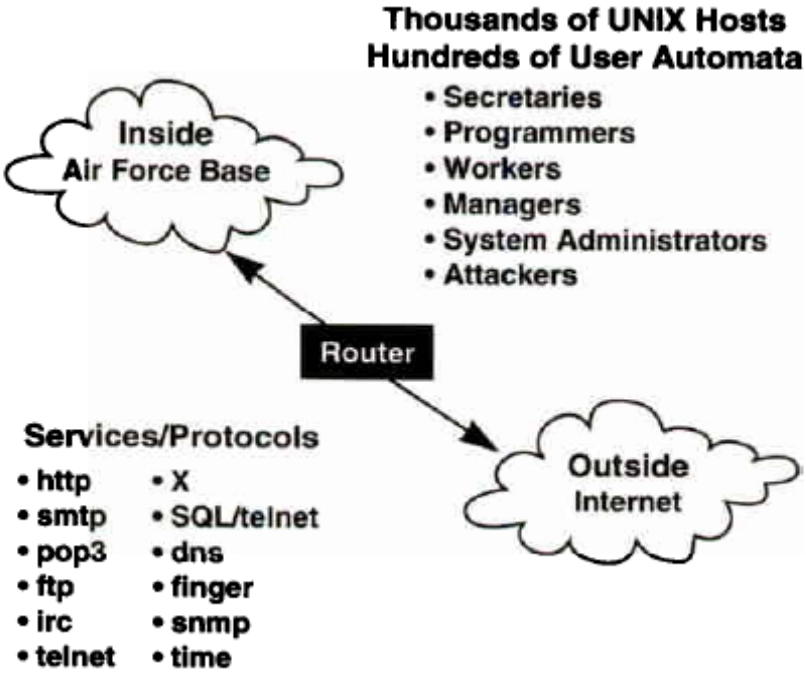

Figure 3.3. Network testbed used in 1999 [8]

Software automata simulate hundreds of programmers, secretaries, managers, and other types of users running common UNIX application programs and some Windows NT programs. A custom Linux kernel modification allows a small number of actual hosts appear as if they were thousands of hosts with different IP addresses. An Expect-based scripting environment is used to automate background traffic sessions and many of the attacks run on the testbed. Protective devices such as firewalls were omitted to focus on attack detection not prevention.

Many types of background, or normal, traffic are generated using a variety of network based application programs. User automata send and receive mail, browse web sites, send and receive files using FTP, use telnet to log into remote computers and perform work, send and receive IRC messages, and perform other tasks.

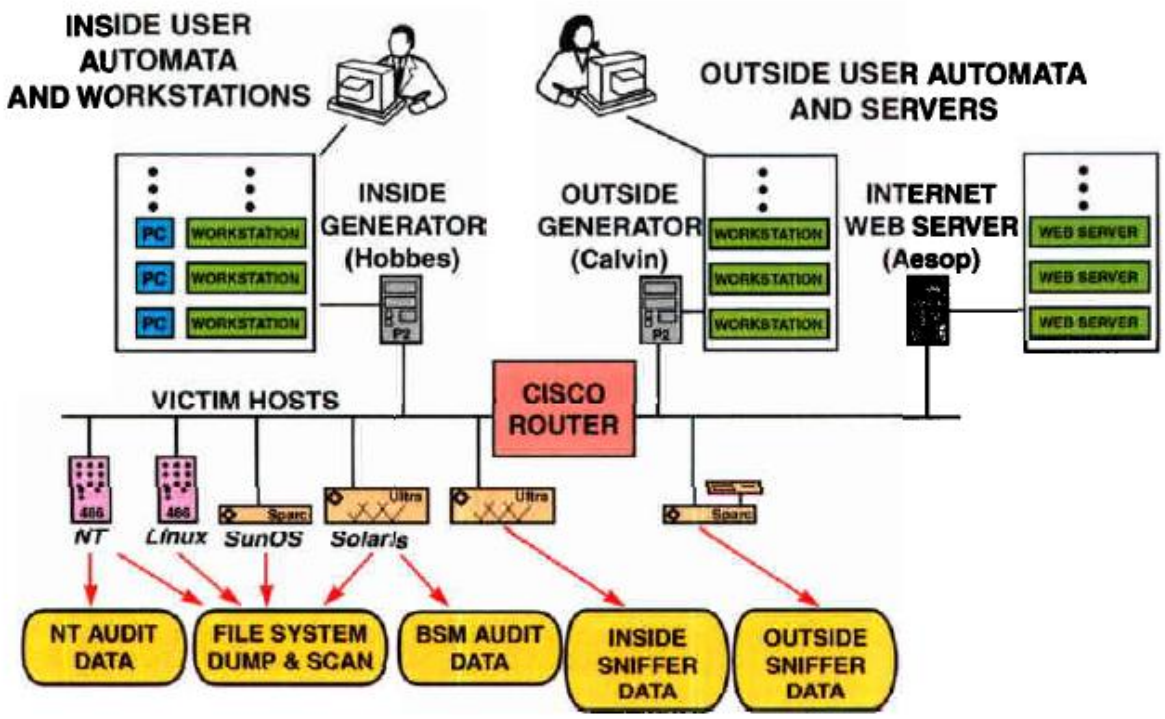

Figure 3.4. Diagram of the main operational features of the 1999 testbed network [8]

3.7. Simulation Time and Date

In 1999 evaluation, each week consisted of five days, Monday through Friday, with 22 hours each day, 8:00 a.m. to 6:00 a.m. Testbed days were simulated in real time and as though they were consecutive, starting on Monday, 1 March 1999, at 8:00 a.m. Individual days are often referred to with the notation “week-day”. Each simulation day begins by setting the date and time to the desired fictitious date and time. Table 3.1 gives the start and end dates and times for each week of the data produced.

Table 3.1. Date and time of network data

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.

In this work, I have used first week (5 days) of training data to model normal background traffic and obtain baseline distribution. And, in order to see the performance of my IDS method, I have used 4. and 5. weeks of test data. I did not use Week 2 data although it contains attacks. Because Week 2 data does not contain attack duration information. Attack duration information is very important when calculating detection rates.

3.8. Target Attack Information

The information given in Week 4 and 5 of DARPA 1999 dataset about attack instances are:

- Name: name of the attack
- Category: in which category the attack can be placed.
- Start Time: start date and time of the attack
- Duration: duration of attack in seconds, minutes and hours.
- Attacker: Attackers IP address/addresses
- Victim: Victims IP address/addresses
- Username: Username used in attack, if used.
- Ports: If attacker used well known ports in attacker/victim machine, these used ports are given.

There are different kinds of attack sets in DARPA 1999 dataset. These attacks can be categorized into four main groups: Denial of Service Attacks, User to Root Attacks,

Remote to User Attacks and Probe Attacks. In this work, we mainly focus on Denial of Service Attacks and table 3.2 gives brief information of about the attacks.

Table 3.2. Denial of service attacks

Denial of Service Attacks	Service	Mechanism	Effect	Number of Occurence
Ping of Death	Icmp	Bug	None	0
Smurf	Icmp	Abuse	Network Slowdown	1
Apache2	http	Abuse	Crash httpd	2
Back	http	Abuse/Bug	Slow server response	4
SYN Flood	Any TCP	Abuse	Deny service on one or more ports for minutes	4
Land	N/A	Bug	Freeze machine	2
Mailbomb	Smtp	Abuse	Annoyance	4
Syslogd	Syslog	Bug	Kill Syslogd	3
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

From the attack types listed in table 3.2, we do not target to detect Ping of Death and Smurf attacks. Because these attacks make use of ICMP services whereas our algorithm deals with TCP and UDP packets and services related to these protocols.

3.9. Denial of Service Type of Attacks

3.9.1. SYN flood (Neptune) attack

When a system (called the client) attempts to establish a TCP connection to a system providing a service (the server), the client and server exchange a set sequence of messages. This connection technique applies to all TCP connections--telnet, Web, email, etc.

The client system begins by sending a SYN message to the server. 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. Here is a view of this message flow:

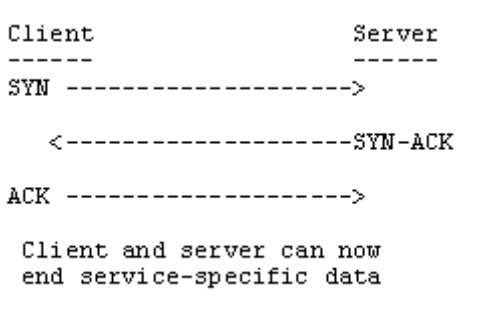

Figure 3.5. TCP connection establishment

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 [32].

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 [33].

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. These SYN packets will clearly change the distributions of the network traffic under observation and therefore this distribution will deviate from baseline distribution.

3.9.2. Udpstorm attack

Udpstorm attack is a DoS attack that causes network congestion and slowdown. When a connection is established between two UDP services, each of which produces output, these two services can produce a very high number of packets that can lead to a denial of service on the machine(s) where the services are offered. Anyone with network connectivity can launch an attack; no account access is needed. For example, by connecting a host’s chargen service to the echo service on the same or another machine, all affected machines may be effectively taken out of service because of the excessively high number of packets produced.

Echo and chargen are regarded as IP small services. Echo service uses UDP and TCP port 8 and is used as a debugging tool to send any datagrams received from a source, back to that source. Chargen service takes its name from Character Generator and this service generates random characters either in one UDP packet containing a random number of

characters, or a TCP session. The UDP chargen server looks for a UDP packet on port 19 and responds with the random character packet. With the TCP chargen, the server sends continuous stream of TCP packets once a connection is made, until the session closes. The data is thrown away. Chargen is used to find the cause for dropped packets. It uses TCP/UDP port 19.

An illustration of such an attack is presented below. Figure 3.6 demonstrates how an attacker is able to create a never-ending stream of packets between the echo ports of two victims by sending a single spoofed packet. First, 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 in like kind, and the loop of traffic continues until it is stopped by intervention from an external source [33].

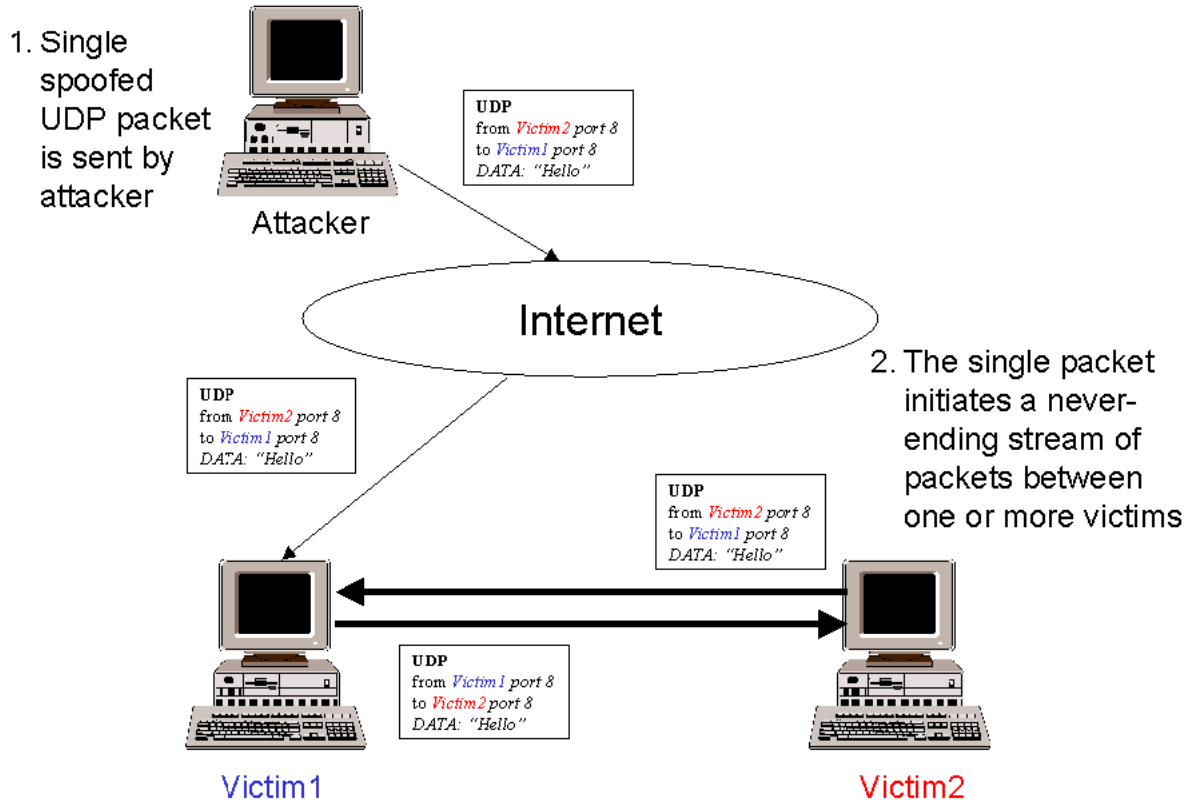

Figure 3.6. Udpstorm attack [8]

3.9.3. 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.

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. This identification is a somewhat subjective process. Each site might have a different definition of how many e-mail messages can be sent by one user or to one user before the messages are considered to be part of a mailbomb.

3.9.4. Apache2 attack

The apache2 attack is a denial of service attack against an apache web server where a client sends a request with many http headers. If the server receives many of these requests it will slow down, and may eventually crash [33].

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 by an attacker, the particular version of the exploit used in DARPA evaluation dataset sent http GET requests 10000 times in each request. 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. A typical http request contains twenty or fewer headers, so the 10000 headers used in this exploit are quite anomalous.

3.9.5. Back attack

In this denial of service attack against the Apache web server, an attacker submits requests with URL's containing many frontflashes. As the server tries to process these requests, it will slow down and becomes unable to process other requests [33]. During the

back attack, there is a considerable increase in HTTP service activity which may be tracked by observing packets destined to port 80.

3.9.6. 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. When Solaris syslogd receives an external message it attempts to do a DNS lookup on the source IP address. If this address doesn't match a valid DNS record, then syslogd will crash with a Segmentation Fault Possible ways to reliably recognize this attack with an IDS is to notice packets destined for the syslog port that contains an unreachable source address or to observe anomalous increase of network packets coming to syslog port [33].

3.9.7. 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 SYN packet in which the source address is the same as the destination address [4]. Since we do not deal with source and destination IP addresses, we do not target to detect land attack.

3.9.8. DoSnuke attack

DoSnuke is a Denial of Service attack that sends Out of Band data (MSG_OOB) to port 139 (NetBIOS), crashing the NT victim (bluescreens the machine). The attack creates a NetBIOS connection. The packets are flagged "urg" because of the MSG_OOB flag. The attack can be detected by searching the sniffed data for a NETBIOS handshake followed by NETBIOS packets with the "urg" flag.

3.9.9. Sshprocesstable attack

SSH process table is similar to the processtable attack in that the goal of the attacker is to cause sshd daemon on the victim to fork so many children that the victim can spawn no more process. This is due to a kernel limit on the number of processes that the OS will allow. This attack will be evident due to the large number of rapid ssh connections to the host, the inability of processes to spawn on the host, and the fact that request for new network logins will be denied.

3.9.10. Tcpreset attack

TCP Reset is a Denial of Service attack that disrupts TCP connections made to the victim machine. That is, the attacker listens for tcp connections to the victim, thus causing the victim to inadvertently terminate TCP connection. One way to detect the attack would be to look at the TCP session/takedown process, and note cases in which RESET packets appear to come from the machine that had initially attempt to begin the connection.

4. METHODOLOGY

4.1. Maximum Entropy Concept

Before getting into the algorithm part, it is better to understand the history and the philosophy of maximum entropy technique. The concept of maximum entropy can be traced back along multiple threads to Biblical times. Only recently, however, have computers became powerful enough to permit the widescale application of this concept to real world problems in statistical estimation.

The maximum entropy concept has a long history. Adopting the least complex hypothesis is embodied in Occam's razor ("Nunquam ponenda est pluralitas sine necesitate.") and even appears earlier, in the Bible and the writings of Herotodus. Laplace might justly be considered the father of maximum entropy, having proposed the underlying theme 200 years ago in his "Principle of Insufficient Reason" : when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely. As E. T. Jaynes, a more recent pioneer of maximum entropy, put it (Jaynes 1990): " ... the fact that certain probability distribution maximizes entropy subject to certain constraints representing our incomplete information, is the fundamental property which justifies use of that distribution for inference; it agrees with everything that is known, but carefully avoids anything that is not known. It is a transcription into mathematics of an ancient principle of wisdom ..." [34].

Intuitively, the principle of maximum entropy is simple: models all that is known and assume nothing about that which is unknown. It is actually what we do in this thesis work. We are trying to model normal network traffic behavior on which we are imposing our expertise knowledge. For example, in any network, it is expected that TCP packets is the majority of the traffic. Also, most users in the network use internet, so we expect to see high volume of network packets with destination port number 80 (HTTP service).

4.2. Packet Classification

In this section, we describe how we divide packets in the network traffic into a set of packet classes. This set of classes is the common domain of the probability densities used in our work [1].

Our work deals with TCP and UDP packets only. TCP and UDP are the two most popular protocols used in today's internet. They carry majority of today's network traffic and can be easily abused. For this reason, this work focuses on anomalies concerning TCP and UDP packets. We consider the protocol information carried by TCP and UDP packets and the destination port numbers in the packets as two of the most important packet header information in the network traffic. In order to study the distribution of these packets, we divide the packets into a set of two-dimensional classes according to the protocol information and the destination port number in the packet header.

In the first dimension, packets are divided into four classes according to the protocol related information. First, packets are divided into the classes of TCP and UDP packets. Two other classes are further split from the TCP packet class according to whether or not the packets are SYN or RST packets. A SYN packet is the first packet to establish a TCP connection. It causes the receiving host to allocate resources for the coming TCP connection. Thus, SYN packets are often used to perform SYN flooding attacks or port scans. RST packets are generated when a TCP connection has to be reset or rejected. For example, in the case of an attempt to connect a nonexisting port, a TCP RST packet is sent from the target host. So an increase in TCP RST packets reveals an increase in TCP connection failures, which may relate to network intrusions.

In the second dimension, packets are divided into 591 classes according to their destination port numbers. Port numbers often determine the services related to the packet exchange. Web services, for example, which generate the majority of today's traffic in the Internet, often have port number 80; FTP services often have port numbers 20 and 21; SSH Remote login protocol has port number 22; TELNET network protocol used in Internet or local area networks (LAN) connections has port number 23 and mail services (SMTP) often have port number 25. According to the Internet Assigned Numbers Authority [35], port numbers are divided 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 order to reduce the total number of packet classes and make classification possible, packets with destination port numbers are grouped into different classes according to whether they are ‘well known ports’, ‘registered ports’, or ‘dynamic and/or private ports’. Packets with destination port numbers between 0 and 1023 are divided into classes of 10 port numbers each. Since packets with destination port numbers 21, 22, 23, 25 and 80 comprise the majority of the network traffic, they are placed into a single class. Consequently, packets with destination port numbers in the range [81, 89] and packets with destination port numbers 20, 24, 26, 27, 28, 29 form another separate class. Furthermore, packets with destination port numbers in the range [1020, 1023] form another class. This produces 108 packet classes. Packets with destination port numbers between 1024 and 49151 are divided into 482 additional classes with each class covering 100 port numbers with the exception of the class that covers port numbers from 49124 to 40151, which contains 28 port numbers. Packets with destination port numbers larger than 49151 are grouped into a single class. Thus, in the dimension, packets are divided into a total of 108 + 482 + 1 = 591 classes.

Altogether, the set of the two-dimensional classes consists of 4 * 591 = 2364 packet classes. These packet classes form the common domain of the probability spaces in this paper. Our work is based on studying the distribution of these packets in the network traffic. We estimate the different packets in the benign traffic according to this classification, and use it as the baseline distribution to detect network traffic anomalies.

Note that this packet classification does not consider in the different distributions of packet classes at different times of the day. This multiple such distributions should be built for different times. However, time factors could be added as a dimension in the packet classes.

4.3. Maximum Entropy Estimation of the Packet Class Distribution

Maximum Entropy estimation is a framework for estimating probability distribution model from the training data. Suppose the distribution model is required to satisfy a set of constraints reflecting properties of the training data, the Maximum Entropy estimation then

follows the principle that a good model of the data should be one that satisfies these constraints only and makes no other assumptions. Here, making no other assumptions implies requiring maximum uniformity of the model. In other words, we should represent the data by the most uniform model of all the models that satisfy the constraints. A mathematical measurement of the uniformity of a distribution P is its entropy

$$-\sum_{w\in\Omega} P(w)\log P(w) \tag{4.1}$$

This entropy is bounded below by 0 corresponding to the case that there is no uncertainty, and below by the log of cardinality of Ω corresponding uniformity.

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 is

$$\widetilde{P}(w)=\frac{\sum 1(x_i\in w)}{n} \tag{4.2}$$

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

Suppose we are given 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, we are looking for a density model P that satisfies $E_P(f_i)=E_{\tilde{P}}(f_i)$ for all $f_i\in F$ and has maximum entropy. In [36], it has been proved that under the constraints $E_P(f_i)=E_{\tilde{P}}(f_i)$ for all $f_i\in F$, the Maximum Entropy estimate is guaranteed to be (a) unique, and (b) the same as the maximum likelihood estimate using the generalized Gibbs distribution, which is in a log-linear form of

$$P(w)=\frac{1}{Z}\exp(\sum_i \lambda_i f_i(w)) \tag{4.3}$$

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

ensures 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

$$P = \arg\min_{P} D(\widetilde{P} // P) \tag{4.4}$$

Feature functions in the log-linear model (above equation) represent certain properties of the training data in the learned model. For example, one of the most important feature of the current Internet traffic is that it is most commonly comprised of TCP packets. The log-linear model can reflect this observation by using a single feature function $f(w) = 1$(w consists of TCP packets), and assigning 2.246 to the parameter λ_f which determines the weight of f

$$P(w) = \frac{1}{Z}\exp(2.246*\mathbf{1}(\text{w consists of TCP packets})) \tag{4.5}$$

This means that a TCP packet is about 9.45 ~ exp(2.246) times more likely than a non-TCP packet. In general, a feature function can be any indicator function that maps Ω to $\{0,1\}$. For the sake of efficiency, feature functions are often selected to express the most important characteristics of the training data in the learned log-linear model, and in return, the log-linear model expresses the empirical distribution with the fewest feature functions and parameters.

Thus, the Maximum Entropy estimation procedure consists of two parts: feature selection and parameter estimation. The feature selection part selects the most important features of the log-linear model, and the parameter estimation part assigns a proper weight to each of the feature functions. These two parts are performed iteratively such that the set of feature functions increases one at a time and the weights for the feature functions in the log-linear model are adjusted as each new feature function is added. As the feature functions in the model are added, the model approaches to the empirical distribution; and the parameter estimation ensures that the model satisfies the Maximum Entropy principle.

4.4. Feature Selection

The feature selection step is a greedy algorithm which chooses the best feature function from a set of candidate feature functions that minimizes the difference between the model distribution and the empirical distribution.

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. So the initial model distribution over Ω is

$$P_0(w) = \frac{1}{Z}, \qquad Z = |\Omega|, \tag{4.6}$$

which is a uniform distribution over Ω. The subscript 0 indicates that there is no feature in the model.

Suppose g is a feature function $g \in F$. We define 1-parameter family of distributions $\{P_{\lambda_g g} : \lambda_g \in R\}$ as follows:

$$P_{0,\lambda_g,g}(w) = \frac{1}{Z'} \exp(\lambda_g g(w)) \tag{4.7}$$

The family of distribution is referred to as the indication of P_0 by g. We also define

$$G_{P_0}(\lambda_g, g) = D(\widetilde{P} \,//\, P_0) - D(\widetilde{P} \,//\, P_{0,\lambda_g,g}) \tag{4.8}$$

Here, $G_{P_0}(\lambda_g, g)$ is the difference of the two Kullback-Leibler divergences related to P_0 and $P_{0,\lambda_g,g}$. It is the improvement that future g brings to the model with its weight λ_g. We define $G_{P_0}(g)$ to be the greatest improvement by adding g to P_0:

$$G_{P_0}(g) = \sup_{\lambda_g} G_{P_o}(\lambda_g, g) \tag{4.9}$$

And $G_{P_0}(g)$ is referred to as the gain of the candidate feature g. In order to minimize the difference between the model distribution and the empirical distribution, the feature function in F with the maximum gain is chosen. We name the selected feature function f_1

$$f_1 = \arg\max_g G_{P_0}(g) \tag{4.10}$$

And obtain the new model with one feature function selected as

$$P_1(w) = \frac{1}{Z}\exp(\lambda_1, f_1(w)) \tag{4.11}$$

Now let P_i be a model with i feature functions selected.

$$P_i(w) = \frac{1}{Z}\exp(\sum_{j=1}^{i} \lambda_j f_j(w)) \tag{4.12}$$

And we want to select the $i+1^{st}$ feature function. Similar to the above procedure, let g a function in $F \setminus \{f_1,...f_i\}$ that has not yet been selected. We obtain

$$P_{i,\lambda_g,g}(w) = \frac{1}{Z'}\exp(\sum_i \lambda_i f_i(w))\exp(\lambda_g g) \tag{4.13}$$

Adding a new feature function would require all the parameters to be adjusted, which makes it computationally expensive to compute the exact form of the model with g added for all g in the candidate feature function set. As a compromise, the feature selection is based upon an approximation. The improvement of adding a new feature g is approximated by adjusting only the weight of the parameter of g and remaining parameters unchanged. Considering large number of candidate feature functions, this approximation makes feature selection practical. With this approximation, we have

$$G_{P_i}(\lambda_g, g)$$

$$= D(\widetilde{P} // P_i) - D(\widetilde{P} // P_{i,\lambda_g,g}) \tag{4.14}$$

$$= \sum_w \widetilde{P}(w)\log\frac{P_{i,\lambda_g,g}}{P_i(w)} \tag{4.15}$$

$$= \lambda_g E_{\widetilde{P}}(g) - \log E_{P_i}(\exp(\lambda_g g)) \tag{4.16}$$

Which is a concave function with respect to λ_g, and

$$G_{P_i}(g) = \sup_{\lambda_g} G_{P_i}(\lambda_g, g) \tag{4.17}$$

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

It is also shown in [37] that, for a candidate feature function g in the form of an indicator function, $G_P(\lambda_g, g)$ is maximized by

$$\hat{\lambda}_g = \arg\max_{\lambda_g} G_P(\lambda_g, g) = \log\left\{\frac{E_{\tilde{P}}(g)(1-E_P(g))}{E_P(g)(1-E_{\tilde{P}}(g))}\right\} \tag{4.18}$$

In which $E_{\tilde{P}}(g)$ is the expected value of g with respect to the distribution of $\tilde{P}$ and $E_P(g)$ is the expected value of g with respect to the distribution of P. At this value of λ_g,

$$G_P(g) = G_P(\hat{\lambda}_g, g) = D(B_{\tilde{P}} \,//\, B_P) \tag{4.19}$$

Where $B_{\tilde{P}}$ and B_P are random variables of Bernoulli distributions given by

$$P(B_{\tilde{P}} = 1) = E_{\tilde{P}}(g) \qquad P(B_{\tilde{P}} = 0) = 1 - E_{\tilde{P}}(g) \tag{4.20}$$

$$P(B_P = 1) = E_P(g) \qquad P(B_{\tilde{P}} = 0) = 1 - E_P(g) \tag{4.21}$$

After a new feature is added to the log-linear model, the parameter estimation procedure will update the weight of all feature functions in the model to minimize the Kullback-Leibler divergence between the model distribution and the empirical distribution.

4.5. Parameter Estimation

Given a set of training data and a set of selected feature functions $\{f_i\}$, the set of parameters is then estimated. The Maximum Entropy estimation locates a set of parameters $\Lambda = \{\lambda_i\}$ in (above equation) for $\{f_i\}$ that minimizes the Kullback-Leibler divergence of $\tilde{P}$ with respect to P:

$$\Lambda = \arg\min_{\Lambda} \sum_{w \in \Omega} \widetilde{P}(w) \log \frac{\widetilde{P}(w)}{P(w)} \tag{4.22}$$

Computing these parameters in the Maximum Entropy model directly is not easy. There are a number of numerical methods that can be exploited, including iterative scaling methods like Generalized Iterative Scaling and Improved Iterative Scaling, as well as first order methods such as steepest ascent, conjugate gradient, and second order methods like quasi-Newton or limited memory variable metric method (LBFGS). In our work, I use iterative scaling method. By using as inputs the empirical distribution, the set of features $\{f_i\}$, and the values of the features under each packet class, the Maximum Entropy estimator can compute the set of parameters $\{f_i\}$ that minimizes the Kullback-Leibler divergence of the empirical distribution and the distribution defined by the log linear model.

4.5.1. Iterative scaling methods

One popular method for iteratively refining the model parameters is Generalized Iterative Scaling (GIS), due to Darroch and Ratchliff (1972). An extension of iterative Proportional Fitting (Deming and Stephan, 1940), GIS scales the probability distribution P by a factor proportional to the ratio of $E_{\tilde{P}}[f]$ to $E_P[f]$, with the restriction that $\sum_i f_i(w) = C$ for each w in the training data (a condition that can be easily satisfied by the addition of a correction feature). We can adopt GIS to estimate the model parameters λ rather than the model probabilities P, yielding the update rule [38]:

$$\delta = \log\left(\frac{E_{\tilde{P}}[f]}{E_P[f]}\right)^{\frac{1}{C}} \tag{4.23}$$

The step size, and thus the rate of convergence, depends on the constant C: the larger the value of C, the smaller the step size. In case not all columns of feature matrix sum to a constant, the addition of a correction feature effectively slows convergence to match the most difficult case. To avoid this slowed convergence and the need for a correction feature, Della Pietra et al. (1997) propose an Improved Iterative Scaling (IIS) algorithm, whose update rule is the solution to the equation [38]:

$$\sum_w P(w) f_i(w) e^{\delta_i f_\#(w)} = \sum_w \tilde{p}(w) f_i(w) \tag{4.24}$$

The above equation is a polynomial in $\exp(\delta_i)$, and the solution can be found straight-forwardly using, for example, the Newton-Raphson method.

$$f(\delta_i) = \sum_w P(w) f_i(w) e^{\delta_i f_\#(w)} - \sum_w \tilde{p}(w) f_i(w) \tag{4.25}$$

$$f'(\delta_i) = \sum_w P(w) f_i(w) f_\#(w) e^{\delta_i f_\#(w)} \tag{4.26}$$

$$\delta_{i+1} = \delta_i - \frac{f(\delta_i)}{f'(\delta_i)} \tag{4.27}$$

Repeat the above operation until convergence.

In my study, Because of the structure of our feature matrix, $f_\#(w)$ is a constant term and is always equal to 3. So I did not need to use iterative root finding methods like Newton.

Start with $\lambda_i = 0$

Calculate:

$$\delta_i = \frac{1}{f_\#(w)} \ln\left(\frac{\sum_w \tilde{P}(w) f_i(w)}{\sum_w P(w) f_i(w)}\right) \tag{4.28}$$

$$\lambda_i^{NEW} = \lambda_i + \delta_i \tag{4.29}$$

4.5.2. First order methods

Iterative scaling algorithms have a long tradition in statistics and are widely used. Their primary strength is that, on each iteration they only require computation of the expected values $E_{P(w)}$. They do not depend on evaluation of the gradient of the log-likelihood function. In the case of ME models, the vector of expected values required by

iterative scaling essentially is the gradient G. Thus, it makes sense to consider methods which use the gradient directly [38].

The most obvious way of making explicit use of the gradient is by *Cauchy's method*, or the method of *steepest ascent*. The gradient of a function is a vector which points in the direction in which the function's value increases most rapidly. Since our goal is to maximize the log-likelihood function, a natural strategy is to shift our current estimate of the parameters in the direction of the gradient via the update rule:

$$\delta^{(k)} = \alpha^{(k)} G(\lambda^{(k)}) \tag{4.30}$$

where the step size $\alpha^{(k)}$ is chosen to maximize $L(\lambda^{(k)} + \delta^{(k)})$. Finding the optimal step size is itself an optimization problem, though only in one dimension and, in practice, only an approximate solution is required to guarantee global convergence.

Since the log-likelihood function is concave, the method of steepest ascent is guaranteed to find the global maximum. However, while the steps taken on each iteration are in a very narrow sense locally optimal, the global convergence rate of steepest ascent is very poor. Each new search direction is orthogonal to the previous direction. This leads to a characteristic "zig-zag" ascent, with convergence slowing as the maximum is approached.

4.5.3. Second order methods

Another way of looking at the problem with steepest ascent is that while it takes into account the gradient of the log-likelihood function, it fails to take into account its curvature, or the gradient of the gradient [38]. The usefulness of the curvature is made clear if we consider a second-order Taylor series approximation of $L(\lambda + \delta)$:

$$L(\lambda + \delta) \approx L(\lambda) + \delta^T G(\lambda) + \frac{1}{2}\delta^T H(\lambda)\delta \tag{4.31}$$

where H is Hessian matrix of the log-likelihood function, the d x d matrix of its second partial derivatives with respect to λ. If we set the derivative of above equation to zero and solve for δ, we get the update rule for *Newtons's method*:

$$\delta^{(k)} = H^{-1}(\lambda^{(k)})G(\lambda^{(k)}) \tag{4.32}$$

While the log-likelihood function for ME models is twice differentiable, for large scale problems the evaluation of the Hessian matrix is computationally impractical, an Newton's method is not competitive with iterative scaling or first order methods. *Variable metric or quasi-Newton methods* avoid explicit evaluation of the Hessian by building up an approximation of it using successive evaluations of the gradient. That is, we replace $H^{-1}(\lambda^{(k)})$ with a local approximation of the inverse Hessian $B^{(k)}$:

$$\delta^{(k)} = B^{(k)}G(\lambda^{(k)}) \tag{4.33}$$

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

$$B^{(k)}y^{(k)} = \delta^{(k-1)}, \text{ where } y^{(k)} = G(\lambda^{(k)}) - G(\lambda^{(k-1)}) \tag{4.34}$$

Variable metric methods also show excellent convergence properties and can be much more efficient than using true Newton updates, but for large scale problems with hundreds of thousands of parameters, even storing the approximate Hessian is expensive. For such cases, we can apply *limited memory variable metric methods*, which implicitly approximate the Hessian matrix in the vicinity of the current estimate of $\lambda^{(k)}$ using the previous m values of $y^{(k)}$ and $\lambda^{(k)}$.

4.6. Feature Functions

The feature functions are selected from a set of candidate feature functions. Since the domain Ω in our work consists of packet classes different in the protocols and the destination port numbers, our candidate feature functions set comprises of 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.

For $w \in \Omega$, the first set consists of four feature functions that check the protocol information of the packets in w. Each feature function $f(w)$ would result in 1 if the packets in w satisfy the corresponding protocol information and 0 otherwise. For example, the feature function that checks whether the packets in w are TCP packets is

$$f_{TCP}(w) = \begin{cases} 1, & packets \quad in \quad w \quad are \quad TCP \; packets \\ 0, & \end{cases} \tag{4.35}$$

The second set consists of 591 feature functions that check the destination port number of the packets in w. For example, the feature function that check whether the packets in w are targeted at ports between 30 and 39

$$f_{[30,39]}(w) = \begin{cases} 1, & packets \quad in \quad w \quad are \quad TCP \; packets \\ 0, & \end{cases} \tag{4.36}$$

Ant the third set consists of 4*591=2364 feature functions that check both the protocol information and the destination port number of the packets in w. For example, the feature function that checks whether the packets in w are TCP SYN packets and the destination port number of the packets fall between 30 and 39 is.

$$f_{SYN*[30,39]}(w) = \begin{cases} 1, & packets \quad in \quad w \quad are \quad TCP \; packets \\ 0, & \end{cases} \tag{4.37}$$

Each feature function can be regarded as a row vector and in total we have 2959 candidate feature functions. And we store these feature functions in a function matrix whose dimension is *number of features x number of class (2959 x 2364)*

4.7. Log-linear Density Model

The log-linear density model is built by iterating two steps. The algorithm begins with a uniform distribution over all packet classes. Then in the feature selection steps, new features are added to the model, and in the parameter estimation steps, the weights for the feature functions are adjusted. This procedure is iterated until some stopping criterion is met. This stopping criterion could be either that the Kullback-Leibler divergence of P with respect to $\widetilde{P}$ is less than some threshold value, or that the gain of adding a new feature function is too small to improve the model.

- Initial data:

  A set of training data with empirical distribution $\widetilde{P}$

  A set of candidate feature functions F

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

- Iterated steps

  (0) Set $n = 0$

  (1) 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

  (2) Parameter Estimation

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

  (3) Check the iteration stopping criterion

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

  Otherwise, return the learned model P_{n+1}

In the above model induction algorithm, step (1) performs the feature selection procedure, and step (2) performs the parameter estimation procedure. As more and more feature functions are added to the model, the induction algorithm is able to reduce the Kullback-Leibler divergence between the empirical distribution $\widetilde{P}$ and the model

distribution P. In my work, this feature selection and parameter estimation process is iterated until the Kullback-Leibler divergence is less than some threshold value.

The training data used are pre-labeled by humans and the packets related to the labeled anomalies are not used in computing the empirical distribution $\widetilde{P}$. In this way, we can threat the packet classes distribution defined by the log linear model from the Maximum Entropy estimation as the baseline distribution, and we are then able to apply cluster map on baseline distribution for the purpose of dimension reductioning.

4.8. Detecting Attacks Using Maximum Entropy Estimation

The relative entropy shows the difference between the distribution of the packet class in the current network traffic and the baseline distribution. If this difference is too large, it indicates that a portion of some packet classes that rarely appear in the training data increases significantly in the current network traffic, or that appear regularly decreases significantly in the current network traffic. In other words, this serves as an indication of anomalies in the network traffic. In our study, we concern only the cases when anomaly traffic increases.

In the following, I will describe a sliding window algorithm which is used by *Gu et al* [1] to detect network traffic anomalies based on this network traffic's relative entropy information.

They divide time into intervals of fixed length ∂. Suppose the traffic in an interval contains the packet sequences $\{x_1, \mathrm{K}, x_n\}$, the empirical distribution $\widetilde{P}$ of the packet classes in this interval is

$$\widetilde{P}(w) = \frac{\sum 1(x_i \in w)}{n}, \tag{4.38}$$

Then for each packet class, they define

$$D_{\tilde{P}//P}(w) = \tilde{P}(w)\log\frac{\tilde{P}(w)}{P(w)}, \tag{4.39}$$

Here, P is the baseline distribution obtained from Maximum Entropy estimation. This gives us a quantitative value that describes the distortion for each packet class w from that of the baseline distribution. If $D_{\tilde{P}//P}(w)$ is large, the portion of this class of packets deviates a lot from that in the baseline distribution, which is used as an indication of anomalies. Not that the sum of $D_{\tilde{P}//P}(w)$ over Ω is the relative entropy of $\tilde{P}$ with respect to P, also referred as the 'relative entropy of class w'.

4.9. Why to Cluster?

After applying maximum entropy method to the empirical distribution, we obtain baseline distribution. And the dimensions of both distributions are the same (n = number of classes). However, if the detection method is windowing, in cases where the network traffic is too slow, using baseline distribution which has high number of classes may lead poor results. Because, in order to obtain a reasonable and useful empirical distribution, the number of packets arrive in one second should at least be 4 or 5 times larger than the window size. But, unfortunately, the network traffic was too slow in our dataset. In order to overcome this problem, we focused on two solutions:

First one was to use fixed length of windows where each window contains enough number of packets to obtain a reasonable empirical distribution. However, this idea did not work well. Because, when considered with the duration of attacks, the time required for a fixed length of window to contain that much of packets is too high and this situation obviously reduce our detection rate.

So, we propose a clustering method and reduce the dimension of the distributions. This gives us a chance to reduce the window length. But, this time, we are able to obtain a useful empirical distribution. And also the reduced window length positively affects our detection rates.

4.10. Clustering Mechanism

Clustering is the classification of objects into different groups, or more precisely, the portioning of data set into subsets (clusters), so that the data in each subset (ideally) share some common trait – often proximity according to some defined distance measure [39]. Common distance functions used in clustering are Euclidean distance, Manhattan distance, maximum norm and Mahalanobis distance. Because of the reasons stated earlier, we perform clustering on baseline distribution and reduce number of packet classes.

Baseline distribution is an one-dimensional (n x 1) vector and applying clustering method on this vector does not give us desired result. For example, suppose PMF values of classes whose destination port values are between 10-20 (class 2, class 593, class 1184 and class 1775) are the same. And, if we use one dimensional baseline distribution vector as an input to the clustering algorithm, with a high probability, we expect to see those classes clustered into the same new class. However, protocol information of those classes are different as TCP, UDP, SYN and RST. Therefore, clustering of baseline distribution depends not only on the PMF values of the classes but also the protocol information.

In order to solve this problem, we propose a multi-dimensional baseline distribution. This multi-dimensional structure of baseline distribution enables clustering algorithm to partition classes according to protocol and destination port number information. Our one dimensional baseline distribution was a 2364 x 1 and elements of the vector were the PMF values of each class. Whereas the new multi-dimensional baseline distribution is a 2364 x 6 matrix. Number of class information does not change with the new structure, but this time, each class is represented with 6 terms, not 1. First column simply gives us the original class sequence, last column gives us PMF values, and the second, third, fourth and fifth columns show us whether it is a TCP, UDP, SYN or RST class. For example, if the second column of a class is 1, then it is a TCP class. Figure 4.1 shows us the structure of our new multi-dimensional baseline distribution.

Class No	TCP	UDP	SYN	RST	PMF Value	
1	1	0	0	0	PMF(1)	TCP Packet Class
2	1	0	0	0	PMF(2)	
3	1	0	0	0	PMF(3)	
.	.		.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
591	1	0	0	0	PMF(591)	
592	0	1	0	0	PMF(592)	UDP Packet Class
593	0	1	0	0	PMF(593)	
594	0	1	0	0	PMF(594)	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
1182	0	1	0	0	PMF(1182)	
1183	0	0	1	0	PMF(1183)	SYN Packet Class
1184	0	0	1	0	PMF(1184)	
1185	0	0	1	0	PMF(1185)	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
1773	0	0	1	0	PMF(1773)	
1774	0	0	0	1	PMF(1774)	RST Packet Class
1775	0	0	0	1	PMF(1175)	
1776	0	0	0	1	PMF(1176)	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	
2364	0	0	0	1	PMF(2364)	

Figure 4.1. Multi-dimensional baseline distribution

4.11. Overview

The first step of our intrusion detection mechanism is to classify network packets according to protocol and destination port number information. Empirical distribution is obtained from this packet classification step. Then Maximum Entropy technique is applied, so that baseline distribution is obtained. The inputs used in Maximum Entropy step are the set of future functions and parameters. Until here, dimensions of empirical and baseline distributions are the same as the number of class which is 2364. However this high number of packet classes gives us poor results at the detection step. Therefore we use a clustering

mechanism for the purpose of dimension reductioning. Figure 3.2 summarizes the methodology followed during this work.

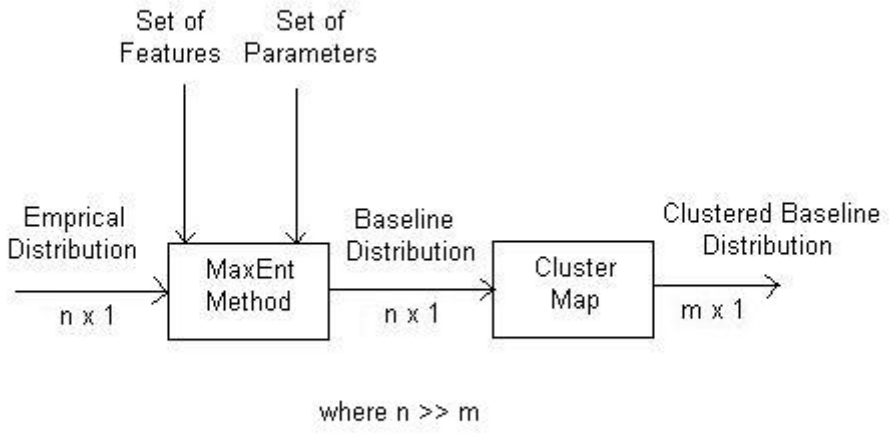

Figure 4.2. Methodology

At the final step, the same detection mechanism (sliding window algorithm) is used to detect intrusions based on network traffic's relative entropy information. But this time, we use fixed size of packet windows instead of time intervals.

Suppose the traffic in an interval contains the packet sequences $\{x_1, \mathrm{K}, x_n\}$, the empirical distribution $\widetilde{P}$ of the packet classes in this fixed packet window is

$$\widetilde{P}(w) = \frac{\sum 1(x_i \in w)}{n}, \tag{4.40}$$

And, we apply clustering method on to the empirical distribution $\widetilde{P}$. The cluster map used earlier for baseline distribution is applied and we obtain a new (clustered) empirical distribution $\widetilde{P}_{NEW}$.

Then for each new packet class, we define

$$D_{\tilde{P}_{NEW} // P_{NEW}}(w) = \tilde{P}_{NEW}(w) \log \frac{\tilde{P}_{NEW}(w)}{P_{NEW}(w)}, \tag{4.41}$$

If $D_{\tilde{P} // P}(w)$ is large, the portion of this class of packets deviates a lot from that in the baseline distribution, which is used as an indication of intrusions. We then use a sliding window detection approach. In each fixed length packet window, we record packet classes that have their divergences larger than a threshold "*d*". If for a certain packet class "w", $D_{\tilde{P}_{NEW} // P_{NEW}}(w) > d$ for more than "*h*" times in a window of "*W*" time slots, an alarm is raised together with the packet class information "*w*", which reveals the corresponding protocol and port number.

5. APPLICATIONS

5.1. Maximum Entropy Based Modeling

The first stage of this work is to model normal network traffic behavior which will give us a chance to distinguish attacks from background traffic. For modeling purpose, I have used DARPA 1999 Week 1 training data. This data do not contain any attacks. It is provided only to facilate the training of intrusion detection systems. And each day of Week 1 data can be found at DARPA Intrusion Detection Evaluation web site. I used inside sniffing data which is a binary file in tcpdump format. In order to read this captured tcpdump files, Wireshark program has been used. It is a network protocol analyzer program for Windows and Unix systems that allows examination of data from a live network, or from a capture file on disk. By using Wireshark, I have selected only protocol, time, info and destination port number fields of TCP and UDP packets. And this selected data is extracted to a text file. This extracted text file contains more information then desired and extremely big in size. In order to simplify this data, I have written a simple string parsing c program which eliminates unnecessary fields and produce a text file with only three columns of information: packet sequence number, protocol information and destination port number. After all these pre-processing operations, packet classification is done and the result is the empirical distribution of training data shown in Figure 5.1.

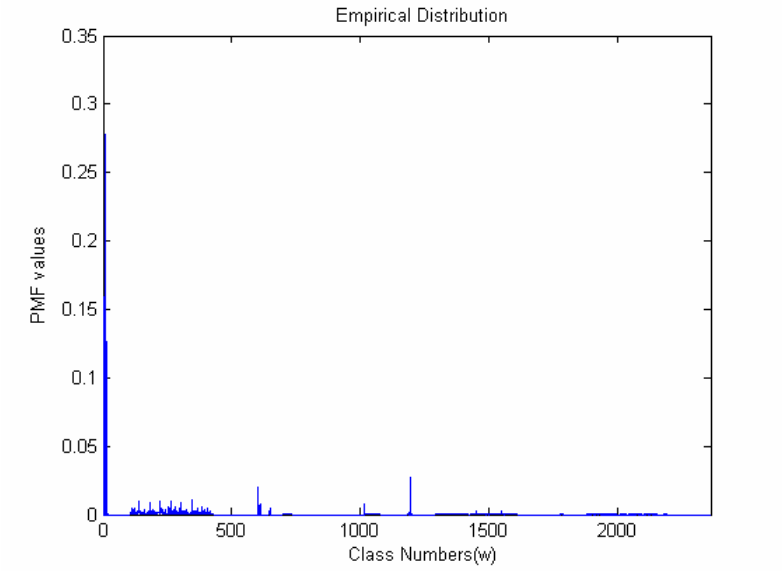

Figure 5.1. Empirical distribution of training data

In Figure 5.1, the horizontal axis shows us the packet classes and the vertical axis shows us the densities (PMF values) of each class. It is obvious from the figure that packets belonging to TCP classes constitutes majority of the training network traffic. From these TCP related classes, class numbers 4, 5, 6, 7 and 13 are the dominant ones. These classes represent FTP, SSH Remote Login, TELNET, SMTP and HTTP services respectively. These services are very popular and highly observable in any network traffic. And after obtaining the empirical distribution of training data, we are ready to obtain our baseline distribution which will then be used for detection purposes.

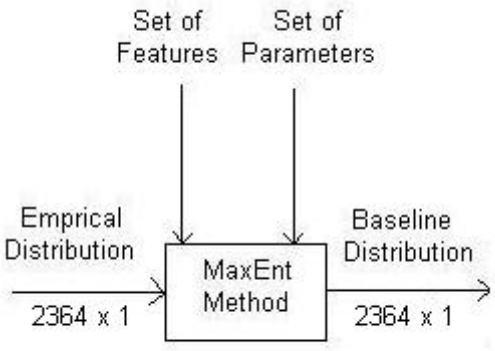

Figure 5.2. Maximum entropy block diagram

As can be seen in Figure 5.2, the size of both empirical and baseline distribution vectors are the same as 2364 which is the number of class. Set of features and parameters are also vectors with the same size and we have in total 2959 feature functions and corresponding parameter values. This feature functions reflect our knowledge about network traffic and by using a greedy algorithm, maximum entropy method selects the most important ones. Feature selection is based on calculating the gain values for each feature and choosing the one with the biggest gain. The stopping criteria for feature selection is the KL divergence between empirical and baseline distribution and we select a value for this which defines how much the empirical distribution will resemble baseline distribution. Or alternatively, you may choose how much feature may be enough to represent empirical distribution. In this work, I choose to use 200 features for this purpose. Figure 5.3 shows the baseline distribution with 200 feature functions selected.

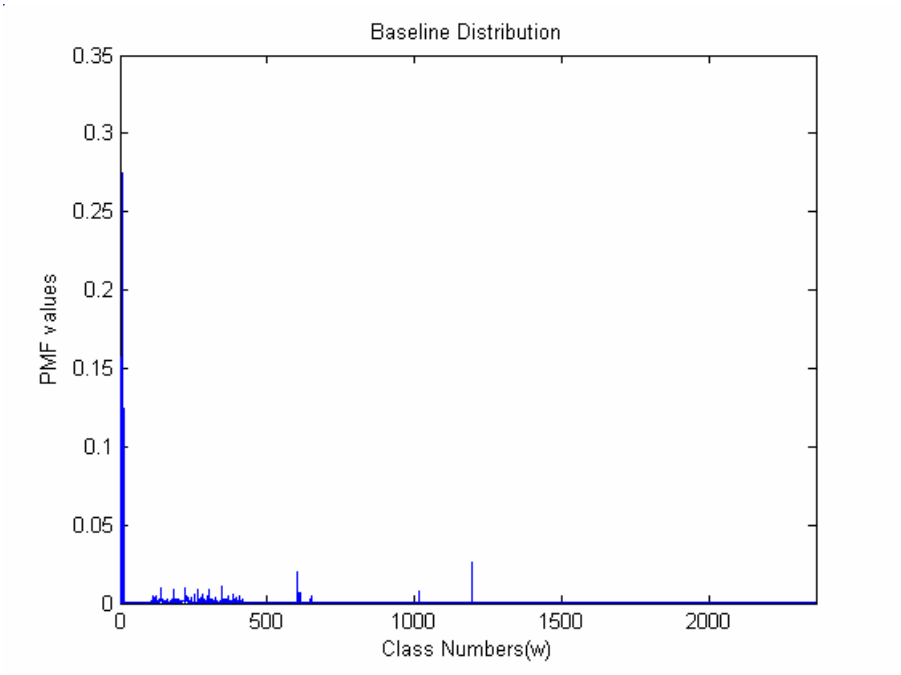

Figure 5.3. Baseline distribution

As can be seen in Figure 5.3, the peak values for several classes representing empirical distribution is preserved in baseline distribution. Apart from those classes, the distribution is uniform. Table 5.1 shows the first 10 features selected by maximum entropy method.

Table 5.1: Top 10 features selected

1	TCP packets
601	TCP packets with destination port number 23 (Telnet Service)
10	Packets with destination port number 23 (Telnet Service)
608	TCP packets with destination port number 80 (HTTP service)
600	TCP packets with destination port number 22 (SSH Remote Login Service)
1790	SYN packets with destination port number 80 (HTTP service)
1196	UDP packets with destination port number between 50-60
11	Packets with destination port number 25 (SMTP Service)
2	UDP Packets
3	SYN Packets

The result of feature selection goes hand in hand with the characteristics of network traffic. For example, the first feature selected with maximum entropy method checks if the packet is a TCP packet or not and as we know and see from the empirical distribution, majority of network traffic is TCP packets. The second and third feature functions show

that Telnet application is frequently used when connecting to a remote user. Also, the other features show the popularity of HTTP, SSH Remote Login and SMTP services.

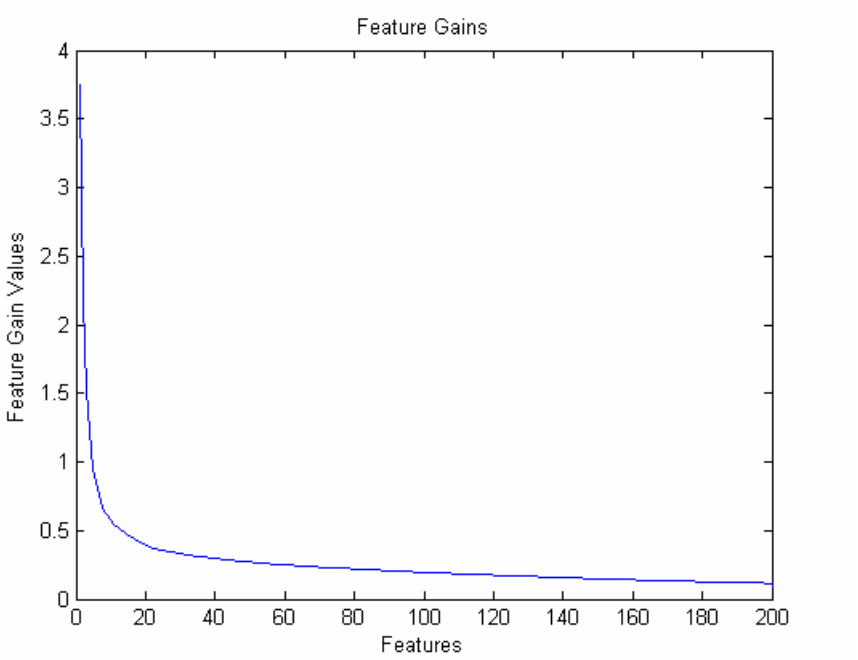

Figure 5.4. Feature gains

Figure 5.4 shows us that the gain value when a new feature is added the feature set with the maximum entropy method is decreasing continuously. This is also an expected result because the algorithm selects features according to their importance and the most important feature will result in the highest gain value.

5.2. Clustering

In detection part, we use a windowing approach. Each window contains fixed size of packets. And empirical distribution obtained from these windows is compared with the baseline distribution. However, because of the characteristics of the simulation data used in this thesis, baseline distribution does not give desired detection results. The traffic data used is slow, so packets gathered in a time window (for example in one second of time interval) do not result in a meaningful empirical distribution. Because the number of packets arrived in a time window is very small compared to number of classes.

In order to overcome this problem, we propose a clustering method. This solution reduces the dimension of our baseline distribution. However, the input to the cluster

function is not our one-dimensional baseline distribution vector. This is because our baseline distribution is composed of four different main classes as TCP, UDP, SYN and RST packets related classes and the meaning of each of them is different. Instead, we use a 2364 x 6 baseline distribution matrix. Therefore, for example, we are sure that a newly clustered class will not point to any former class related with UDP packets and SYN packets at the same time.

I have used Matlab "clusterdata" function. Clusterdata constructs clusters from m-by-n matrix data X. It takes as an argument maximum number of clusters information and by using a distance metric like 'Euclidean', '.Mahalanobis' or 'Hamming', it constructs desired number of clusters. The output C is a vector of size m containing a cluster number for each observation and can be thought as a cluster map.

Figure 5.5. Cluster map

As can be seen from Figure 5.4, I have chosen the size of clustered baseline distribution as 50 x 1. The important classes in the baseline distribution about mostly used applications like Telnet, HTTP, FTP and SMTP are clustered into single classes. Besides, some attacks target specific ports and the classes related to those ports are also clustered into a single class. For example, DoSnuke exploits the bugs in NetBIOS service and uses port 139, sshprocesstable exploits SSH service and targets port 22, Syslogd attack make use of port 514 and Udpstorm abuses echo service which uses port 8. Taking all these facts into consideration, we finally end up with the below clustered baseline distribution. In addition, Apache2 and Back attacks use HTTP service port 80 and Mailbomb attack uses SMTP service port 25. These attacks are considered in our target attack set and it is important for us that the classes related to the services they supply are clustered into a

single class. By this way, we will have a chance to observe the variations in relative entropy values of those clustered classes effectively.

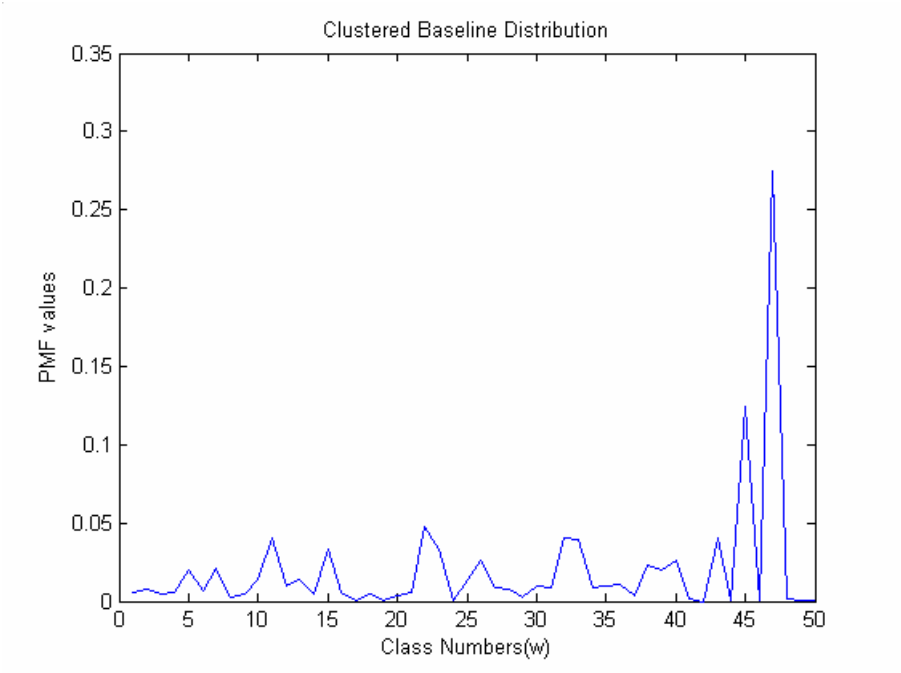

Figure 5.6. Clustered baseline distribution

In the above clustered baseline distribution figure, class number 47 which has the peak value corresponds to Telnet service port 23.Class number 45, which has the second PMF value represents HTTP service port 80. Apache2 and Back attack uses port 80 and these attack instances can be traced with the relative entropy between class 45 of clustered baseline distribution and class 45 of empirical distribution. TCP and UDP port 8 corresponds to class 50 and Udpstorm attack can be traced with the relative entropy of class 50 at detection time. In a similar fashion, mailbomb attack can be traced with class 38, DoSnuke attack can be traced with class 48 and Syslogd attack can be traced with class 49. Besides, SYN and RST packets related classes in the baseline distribution is clustered into class 10, 29, 32 and 41 in the clustered baseline distribution. So, the existence of a Neptune (SYN Flood) attack can be traced with these classes.

5.3. Detection Part

In detection part, we use relative entropy metric which shows us the difference between the distribution of the packet classes in the current network traffic and the

clustered baseline distribution. *Gu et al* divide time into intervals of fixed length. However, time windowing approach does not work in our case because of the characteristics of our network data. If we keep the time interval small, we can not obtain a meaningful distribution, and if we choose to increase the size of the time window, with a high probability we miss the attacks. Because the duration of an attack may be small in some cases. We use fixed size of packet windows instead. Suppose the traffic in a packet window contains the packet sequences $\{x_1, \mathrm{K}, x_n\}$, the empirical distribution $\widetilde{P}$ of the packet classes in this interval is

$$\widetilde{P}(w) = \frac{\sum 1(x_i \in w)}{n}, \tag{5.1}$$

Then for each packet class, they define

$$D_{\widetilde{P} // P}(w) = \widetilde{P}(w) \log \frac{\widetilde{P}(w)}{P(w)}, \tag{5.2}$$

Here, P is the clustered baseline distribution. If $D_{\widetilde{P} // P}(w)$ is large, the portion of this class of packets deviates a lot from that in the baseline distribution, which is used as an indication of intrusions. We then use a sliding window detection approach. In each fixed length packet window, we record packet classes that have their divergences larger than a threshold "*d*". If for a certain packet class "w", $D_{\widetilde{P}_{NEW} // P_{NEW}}(w) > d$ for more than *"h"* times in a window of *"W"* time slots, an alarm is raised together with the packet class information *"w"*, which reveals the corresponding protocol and port number. In this method, the threshold values can be set to different value for each attack type.

However, for some types of attacks like DoSnuke and Syslogd attack, the relative entropy of its related class makes a peak value for a short period of time (just for one packet window for example). In those cases setting only a threshold value for relative entropy is enough to check if there is an attack or not.

5.3.1. Apache2 attack

A bug in Apache web servers may allow a remote attacker to perform a Denial of Service attack. Attacker sends lots of http request with http headers. As the server receives many of these http requests, it will slow down, and eventually may crash. So, we expect to see an increase in the activity of http service (port 80) in the network traffic during Apache2 attack instances. Below figures show the relative entropy values of clustered class 45 which points to tcp packets with destination port 80.

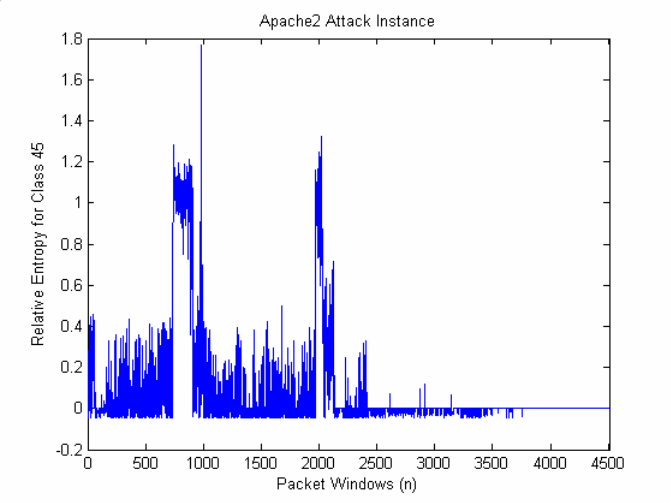

Figure 5.7. Apache2 attack - 1

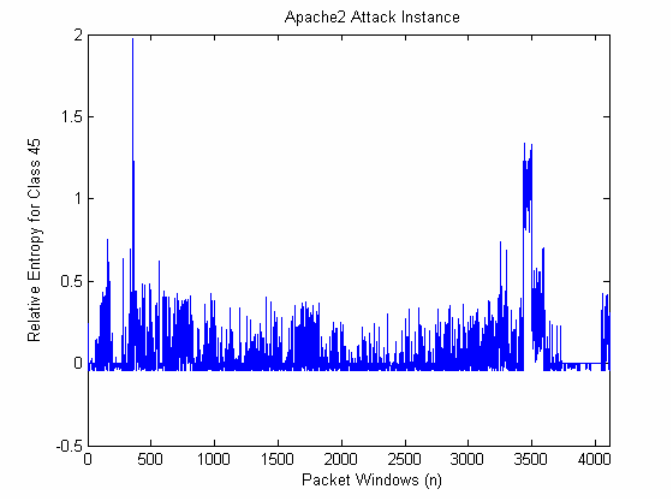

Figure 5.8. Apache2 attack – 2

5.3.2. Neptune attack

In a Neptune (SYN Flood) attack, the attacker sends streams of SYN packets to a range of ports on target machine. For a short period of time after these packets sent, other users are unable to access the services provided by these ports. And during the attack instance, an increase in the relative entropy values of classes related to SYN and RST packets is observed. Those changes are reflected in to our clustered classes 10, 29, 32 and 41.

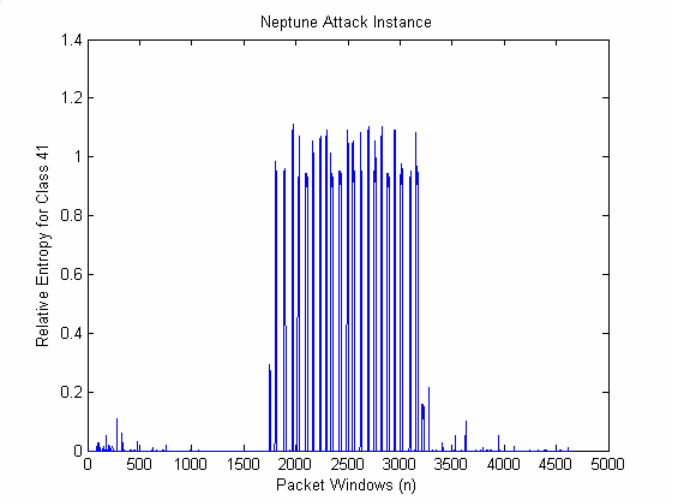

Figure 5.9. Neptune attack - 1

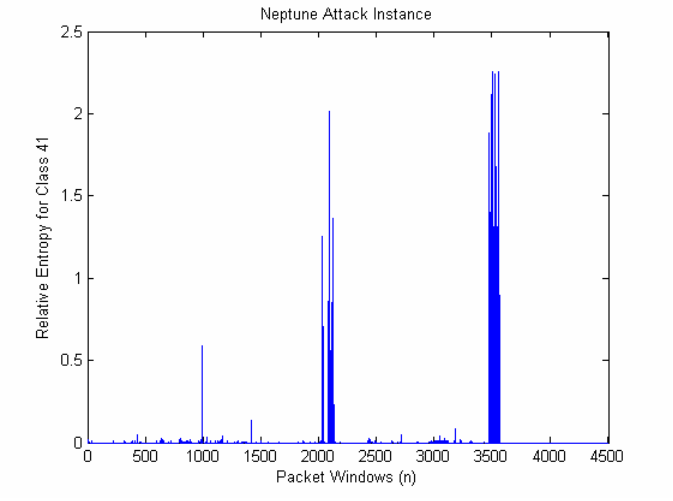

Figure 5.10. Neptune attack – 2

5.3.3. Udpstorm attack

Udpstorm attack is started by sending a UDP packet to the echo server on one victim with the spoofed source address and port of the echo or chargen server of the other victim. The result is that they echo each other endlessly and waste network bandwidth. Udpstorm attack uses UDP port 8 and can be traced in our method with class 50.

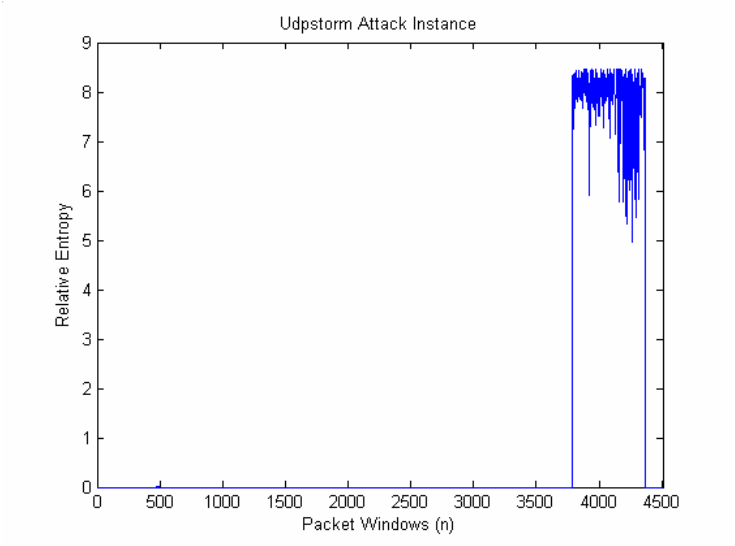

Figure 5.11. Udpstorm attack -1

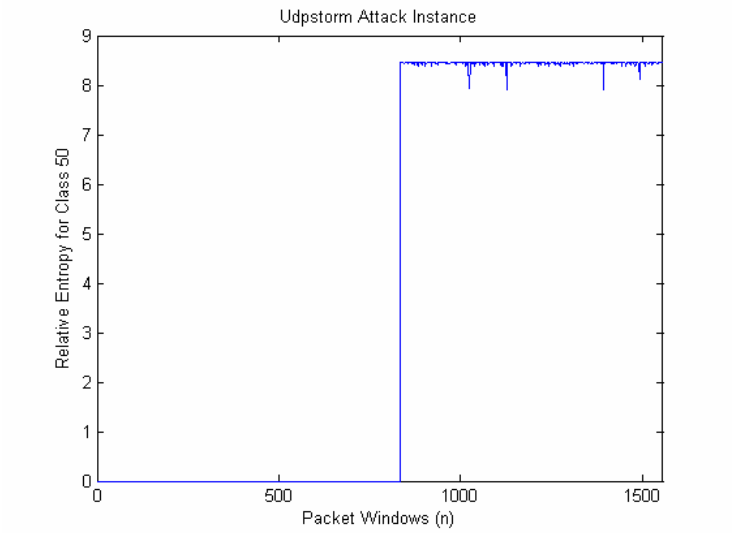

Figure 5.12. Udpstorm attack - 2

5.3.4. Syslogd attack

This is a denial-of-service for the syslog service that connects to port 514 with irresolvable source ip. During Syslogd attack instance, we observe an anomalous increase in the number of packets targeted to port 514. And this port number is clustered in to a new class as class 49 in our work for detailed inspection of syslog service. Above two graphs shows the relative entropy values of class 49 in the attack instances.

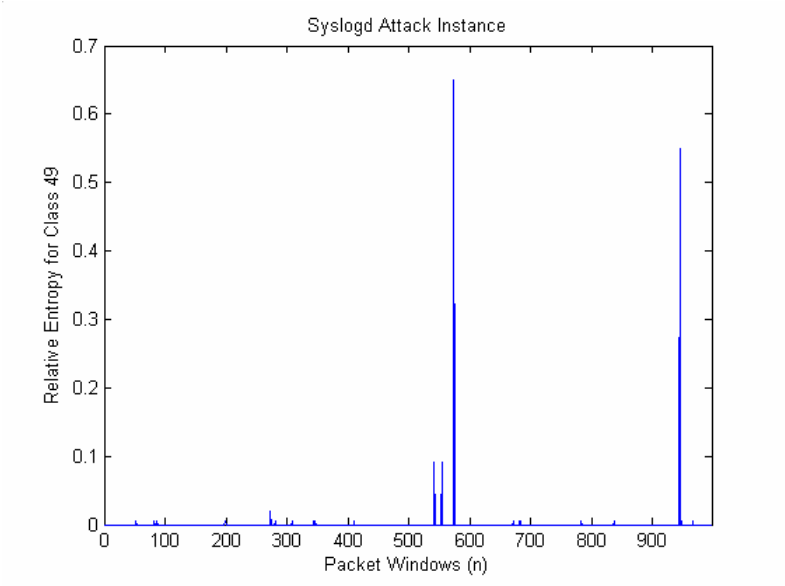

Figure 5.13. Syslogd attack - 1

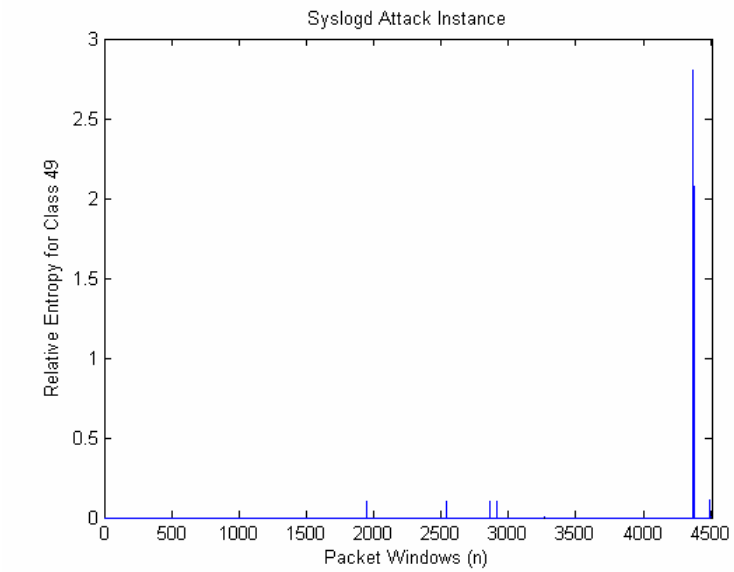

Figure 5.14. Syslogd attack -2

5.3.5. DoSnuke attack

DoSnuke is a type of the denial-of-service attacks which sends Out-of-Band data (MSG_OOB) to port 139 (NetBIOS) of victim machine, causing the victim to crash. During the attack instance, a sudden dramatic increase in the number of NetBIOS packets is observed.

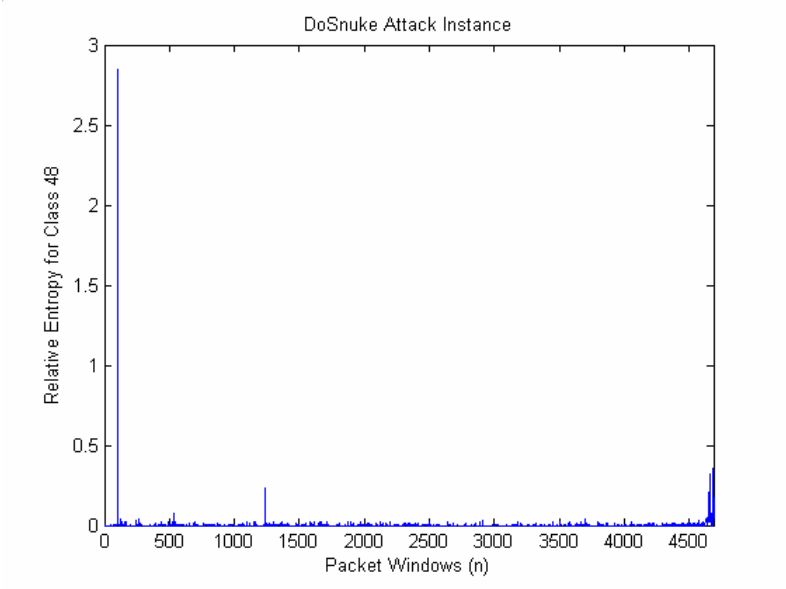

Figure 5.15. DoSnuke attack - 1

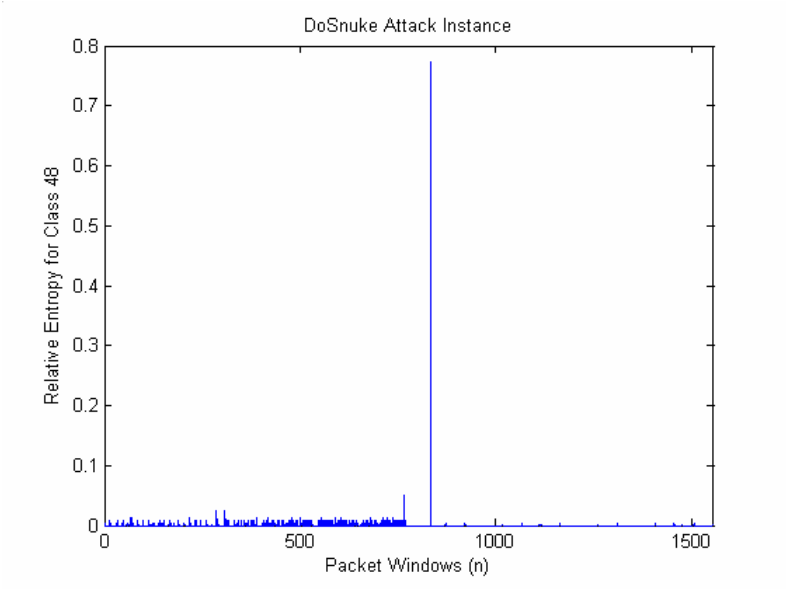

Figure 5.16. DoSnuke attack - 2

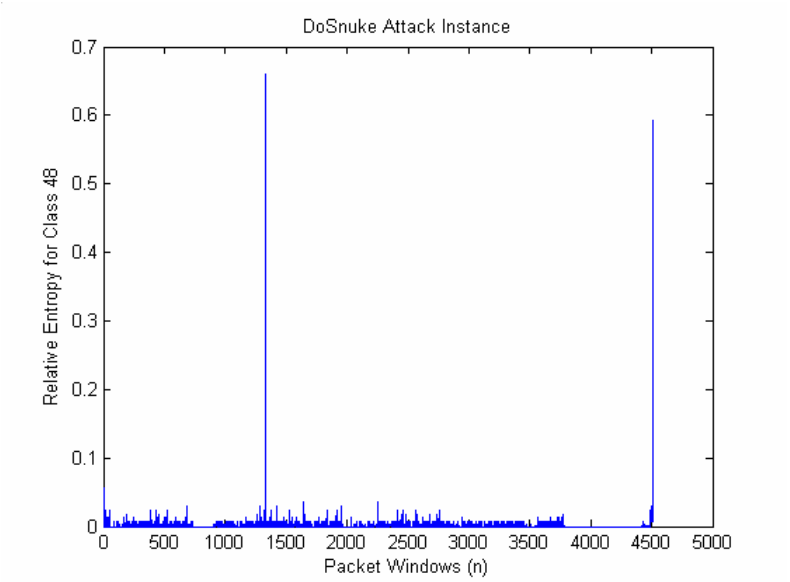

Figure 5.17. DoSnuke attack – 3

5.3.6. Back attack

In this type of denial of service attack, an attacker submits requests with URL`s containing many frontflashes. This attack also uses port TCP port 80 and during the attack instances, we observe an anomalous increase in the number of TCP packets destined to port 80.

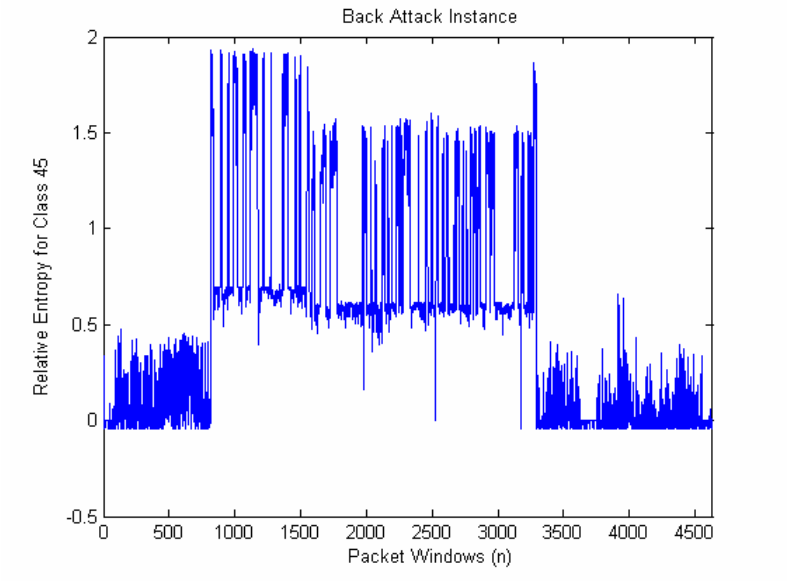

Figure 5.18. Back attack - 1

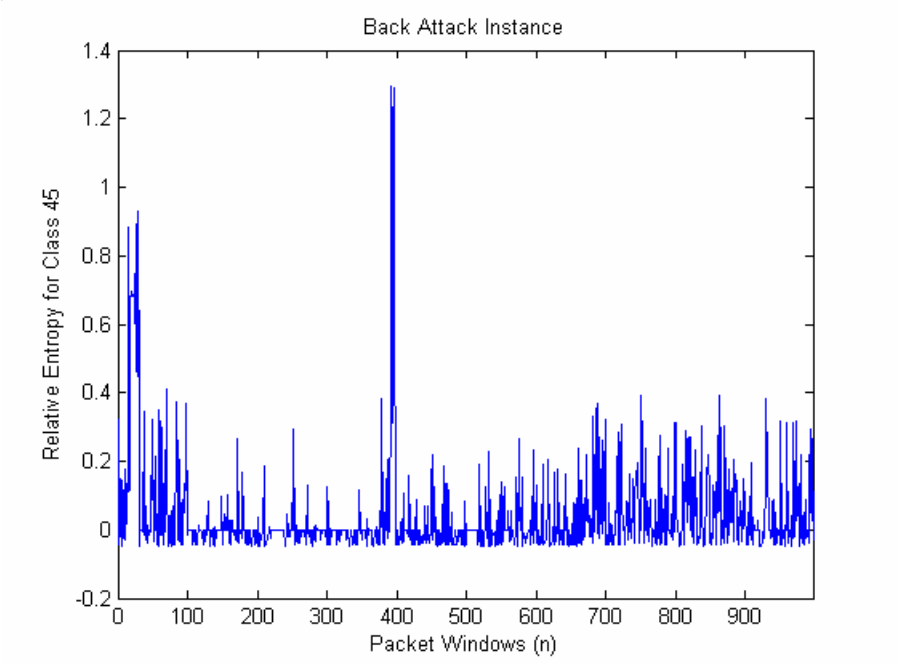

Figure 5.19. Back attack – 2

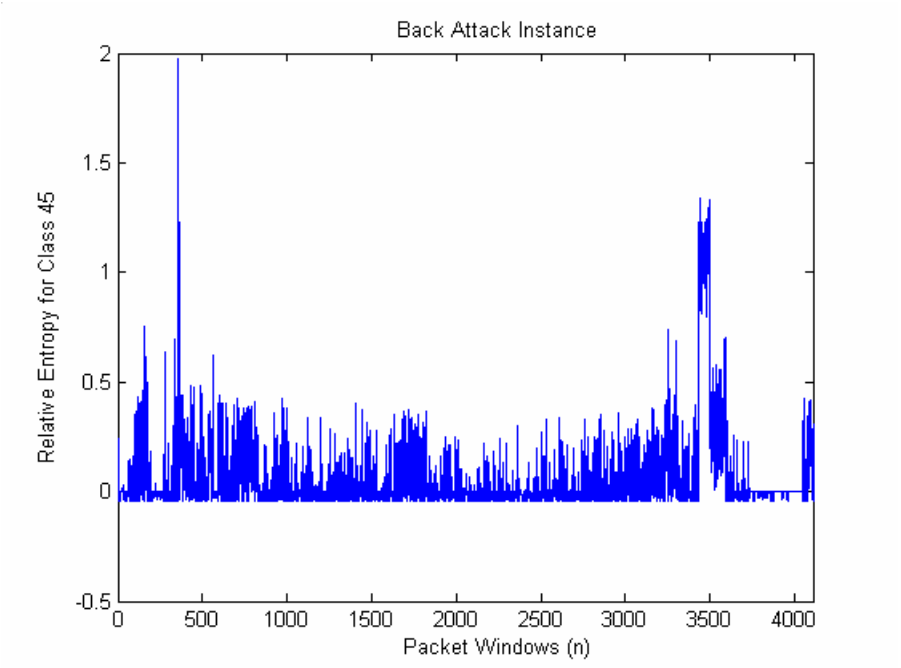

Figure 5.20. Back attack - 3

5.3.7. Tcpreset attack

It is a kind of DoS attack that disrupts TCP connections made to the victim machine. That is, the attacker listens for tcp connections to the victim, and sends a spoofed RST packet to the victim, causing the victim to terminate the TCP connection. During tcpreset attack instance we observe a dramatic increase in the number of RST packets which can be traced with clustered class 32 in this work.

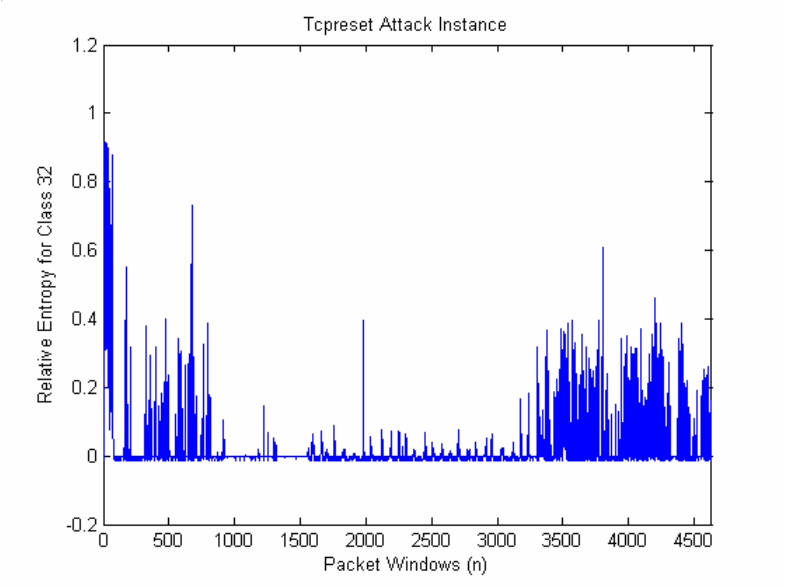

Figure 5.21. Tcpreset attack - 1

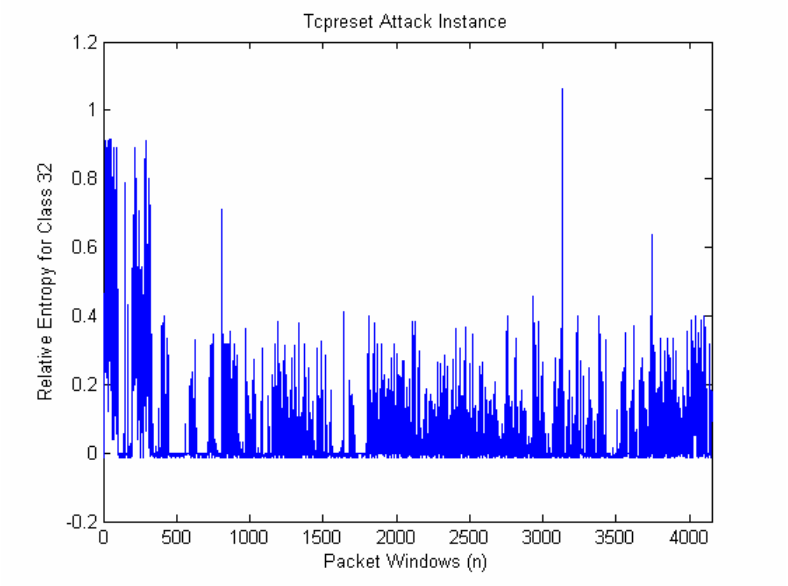

Figure 5.22. Tcpreset attack – 2

5.3.8. Mailbomb attack

In mailbomb attack, the victim's mail queue is flooded by an abundance of messages, causing system failure. The activity of SMTP service can be observed to detect this attack. SMTP service is assigned to port 25 and this port is clustered into a new single class as class 47 in our work.

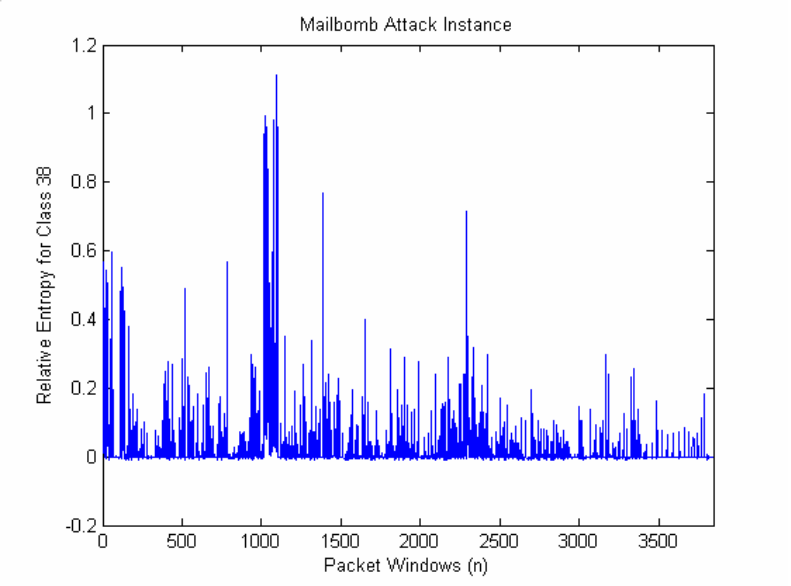

Figure 5.23. Mailbomb attack – 1

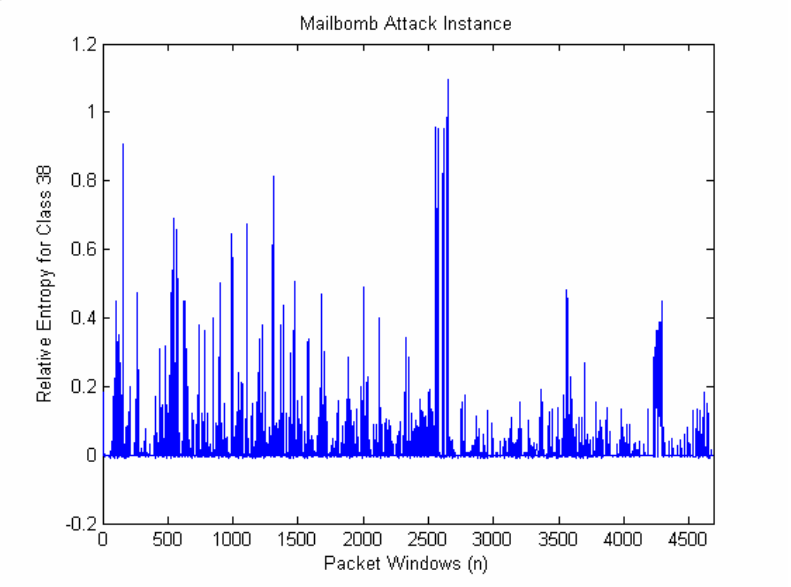

Figure 5.24. Mailbomb attack – 2

5.3.9. Sshprocesstable attack

This attack is evident due to the large number of rapid ssh connections to the host, the inability of processes to spawn on the host, and the fact that request for new network logins will be denied. During the attack instance, instance we observe a dramatic increase in the number of packets targeted to port 22 which can be traced with clustered class 43 in this work.

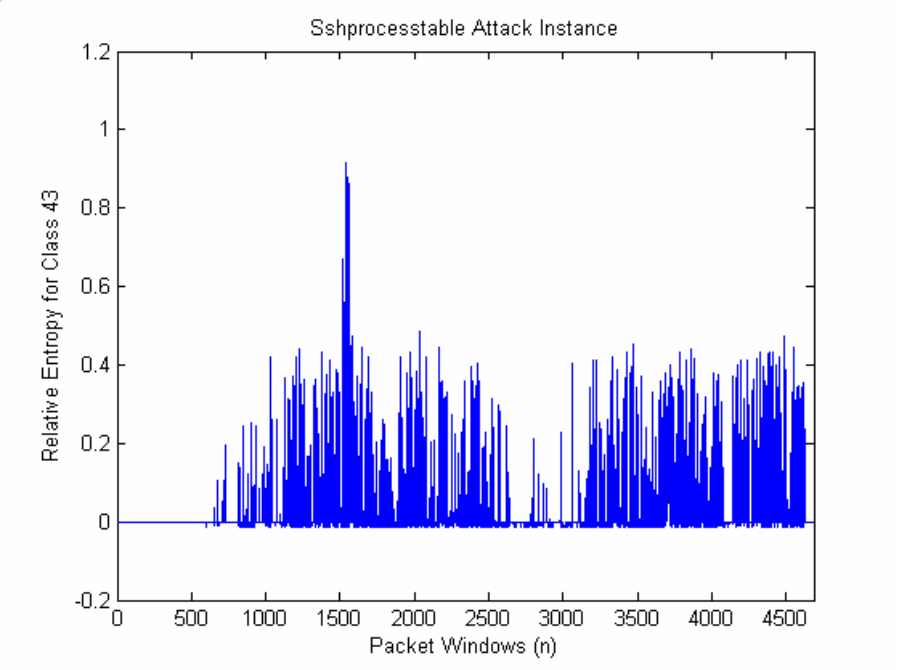

Figure 5.25. Sshprocesstable attack

5.4. Results

As can be seen from the graphs of attack instances, each type of attack has a target destination port/s. And, with the help of packet classification and clustering mechanisms, these attack footprints can be traced with their related classes.

After observing the behavior of relative entropy measure for each attack, I have chosen both detection method and threshold values accordingly. For DoSnuke and Syslogd attacks, we observe a sudden dramatic increase in the relative entropy values of their related classes. We see a peak value at the attack instance and very small values at other times. So, setting a threshold value and checking if the relative entropy value is bigger or

not is simple but very effective way of detecting these attack types. So I picked the value of 0.5 for this purpose.

For the rest of the attack types, I employed a windowing mechanism. The approach is simple. That is, in each window we record packet classes that have their divergences larger than a threshold "*d*". If for a certain packet class *"w"*, the relative entropy between clustered empirical distribution of packet window and clustered baseline distribution is bigger for more that *"h"* times in a window of *"W"* time slots, an alarm is raised together with the related class information *"w"*. In this work, I have used 15, 5 and 0.8 for the values of *"W"*, *"h"* and *"d"* respectively.

In this method, duration of attack instances play an important role in detection rates. If the attack duration is too small, it may be missed. For example, in Figure 5.20, there are two Back attack instances actually. However, the first one lasts for 4 seconds. Therefore, a peak value can be seen in the relative entropy graph no alarm is raised. And we had the same problem for one Neptune, Tcpreset and Mailbomb attack instances which results in a missed detection. Table 5.2 shows total number of attack instances for each type and the number of detection.

Table 5.2. Target attacks scoring table

Attack Type	Number of Occurrences	Number of Detection
Apache2	2	2
Back	4	2
Neptune	4	3
Mailbomb	3	2
Syslogd	3	3
Udpstorm	2	2
Dosnuke	4	4
Sshprocesstable	1	1
Tcpreset	3	2

5.5. Performance Analysis

The performance analysis of this work is done with the help of true and false positive rates. Two other metrics which are true and false negative rates can be easily calculated from the first two. Below are the brief explanations of these metrics.

False Positive Rate : Formally, it is the probability of rejecting the null hypothesis when it is true. A false positive, also known as a false detection or false alarm, occurs when an IDS detects an attack when there is no attack threat. It is calculated as:

$$False\quad Positive\quad Rate = \frac{Number\quad of\quad False\quad Alarm\quad Samples}{Number\quad of\ Clean\quad Samples} \tag{5.3}$$

False Negative Rate: It is the probability of failing to reject the null hypothesis when it is false. False negative occurs when an IDS does not detect an attack and there is already an attack instance at that time. False negative rate can be calculated as:

$$False\quad Negative\quad Rate = \frac{Number\quad of\quad Missed\ Attack\quad Samples}{Number\quad of\ Attack\quad Samples} \tag{5.4}$$

True Positive Rate: True positive rate is the complement of false negative rate. A true positive occurs when an IDS detects an attack and there is already an attack instance at that time. And true positive rate can be calculated easily as:

True Positive Rate = 1 – False Negative Rate (5.5)

True Negative Rate: True negative rate is the complement of false positive rate. A true negative occurs when an IDS does not detect an attack and there is no attack threat at that time.

True Negative Rate = 1 – False Positive Rate (5.6)

Table 5.3. Performance analysis table – 1

True Positive Rate:	94.8
False Positive Rate:	1.31
True Negative Rate:	98.69
False Negative Rate:	5.2

Apart from the above metrics, two other useful metrics are the positive predictive value (PPV), which is the probability of an intrusion when the IDS outputs an alarm, and the negative predictive value (NPV), which is the probability of no intrusion when the IDS does not output an alarm. These metrics are very important from a usability point of view because the IDS alarms are useful to an intrusion response system only if the IDS has high PPV and NPV. Both PPV and NPV depend on TP and FP, and are very sensitive to the base rate (B), which is the prior probability of intrusion. The formulas to calculate these metric are given as:

$$Base \quad Rate \quad = \frac{number \quad of \quad attack \quad samples}{number \quad of \; all \quad samples} \tag{5.7}$$

$$PPV = P(I / A) = \frac{B(1 - FN)}{B(1 - FN) + (1 - B)FP} \tag{5.8}$$

$$NPV = P(I' / A') = \frac{(1 - B)(1 - FP)}{(1 - B)(1 - FP) + B.FP} \tag{5.9}$$

There is also one other metric that takes into account all the other aspects of the detection capability like TP, FP, PPV, NPV and B. This metric is an information-theoretic measure of the intrusion detection capability and it is the ratio of the mutual information between IDS input and output, and the entropy of the input. It is called Intrusion Detection Capability, or C_{ID} [41]. Below is the calculation of this metric and the results.

$$C_{ID} = \frac{I(X;Y)}{H(X)} = \frac{(H(X) - H(X / Y))}{H(X)} \tag{5.10}$$

$$H(X) = -B\log B - (1-B)\log(1-B) \tag{5.11}$$

$$H(X/Y) = -B(1-FN)\log\frac{B(1-FN)}{B(1-FN)+(1-B)FP} - B.FN.\log\frac{B.FN}{B.FN+(1-B)(1-FP)}$$

$$-(1-B)(1-FP).\log\frac{(1-B)(1-FP)}{(1-B)(1-FP)+B.FN} - (1-B).FP.\log\frac{(1-B)FP}{(1-B).FP+B(1-FN)} \tag{5.12}$$

Table 5.4. Performance analysis table - 2

Base Rate	7.31
Positive Predictive Value	85.09
Negative Predictive Value	99.59
Intrusion Detection Capability	77.46

6. CONCLUSION AND FUTURE WORK

In my thesis, we aimed to detect denial-of-service type of attacks using Maximum Entropy method and relative entropy measure. The packet distribution of attack free network traffic is modeled using the Maximum Entropy approach and is used as a baseline to detect the attacks. In the midst of modeling and detection phase, we use a clustering approach to reduce the dimension of our baseline distribution. This gives us a chance to reduce the negative effects of slow network traffic. And in detection part, we use windowing method.

Experimental results show that, this method effectively detect different kinds if denial-of-service attacks. An important advantage of this method is that, it gives network administrator an idea of the type of attack detected. And this method requires constant memory and a computation time proportional to the traffic rate.

As a future work, several possible extensions may enhance this method. For example, instead of using only TCP and UDP packets, ICMP packets may also be used. This more general higher dimensional packet classification will provide us more general view of network traffic, and more attack types may be detected like Smurf or Ping of Death.

In addition, time information may also be added as one of the dimensions when creating the packet classes. In different times of the day, the network traffic has different packet distributions. By adding the time as one of the dimensions in the packet class, it is possible to represent the time variant packet distributions. Thus, a single model can be used as the baseline distribution at different times of the day.

APPENDIX A

Proof of Iterative Scaling Algorithm

$\widetilde{P}(w)$ = Empirical Distribution

$P(w)$ = Modeled Maximum Entropy Distribution

$$P(w,\lambda) = P(w) = \frac{e^{\sum_i \lambda_i f_i(w)}}{\sum_w e^{\sum_i \lambda_i f_i(w)}}$$

Kullback-Leibler distance between empirical and maximum entropy distribution is:

$$D(\widetilde{P} \,//\, P) = \sum_w \widetilde{P}(w) \log \frac{\widetilde{P}(w)}{P(w)}$$

$$= \sum_w \widetilde{P}(w) \log \widetilde{P}(w) - \sum_w \widetilde{P}(w) \log P(w)$$

We seek to find λ which minimizes $D(\widetilde{P} \,//\, P)$,

$$\min_{\{\lambda_i\}} D(\widetilde{P} \,//\, P) \Leftrightarrow \max_{\{\lambda_i\}} \sum_w \widetilde{P}(w) \log P(w)$$

Let $L(\{\lambda_i\}) \overset{\Delta}{=} \sum_w \widetilde{P}(w) \log P(w,\lambda)$,

$$L(\{\lambda_i + \delta_i\}) - L(\{\lambda_i\}) = \sum_w \widetilde{P}(w) \log P(w, \lambda+\delta) - \sum_w \widetilde{P}(w) \log P(w,\lambda)$$

$$\log P(w,\lambda) = \log\left[e^{\sum_i \lambda_i f_i(w)} \right] - \log\left[\sum_w e^{\sum_i \lambda_i f_i(w)} \right]$$

$$= \sum_i \lambda_i f_i(w) - \log\left[\sum_w e^{\sum_i \lambda_i f_i(w)} \right]$$

$$\log P(w,\lambda+\delta)-\log P(w,\lambda)=\sum_i(\lambda_i+\delta_i)f_i(w)-\sum_i\lambda_i f_i(w)-\log\frac{\sum_w e^{\sum_i(\lambda_i+\delta_i)f_i(w)}}{\sum_w e^{\sum_i\lambda_i f_i(w)}}$$

$$=\sum_i\delta_i(w)f_i(w)-\log\frac{\sum_w e^{\sum_i(\lambda_i+\delta_i)f_i(w)}}{\sum_w e^{\sum_i\lambda_i f_i(w)}}$$

Claim 1: $\log\alpha\le\alpha-1$

Proof of Claim 1:

Let $f(\alpha)=\alpha-1-\log(\alpha)$

We want to prove $f(\alpha)\ge 0$ for all α

$$f'(\alpha)=1-\frac{1}{\alpha}\Rightarrow\{[f'(\alpha)=0]\}\rightarrow\alpha=1$$

$$f''(\alpha)=\frac{1}{\alpha^2}\ge 0\Rightarrow \text{f is convex}$$

$$\min_\alpha f(\alpha)=f(1)=1-1-\log(1)=0$$

$$\Rightarrow f(\alpha)\ge 0$$

Results:

(i) $\log(\alpha)\le\alpha-1$

(ii) $-\log(\alpha) = \log\frac{1}{\alpha} \geq 1-\alpha$

So;

$$-\log\frac{\sum_w e^{\sum_i(\lambda_i+\delta_i)f_i(w)}}{\sum_w e^{\sum_i \lambda_i f_i(w)}} \geq 1-\log\frac{\sum_w e^{\sum_i(\lambda_i+\delta_i)f_i(w)}}{\sum_w e^{\sum_i \lambda_i f_i(w)}}$$

$$= 1-\sum_w\left[\frac{e^{\sum_i \lambda_i f_i(w)}}{\sum_w e^{\sum_i \lambda_i f_i(w)}}.e^{\sum_i \delta_i f_i(w)}\right]$$

$$= 1-\sum_w P(w)e^{\sum_i \delta_i f_i(w)}$$

$$\rightarrow \log P(w,\lambda+\delta)-\log P(w,\lambda) \geq \sum_i \delta_i f_i(w)+1-\sum_\mu P(\mu)e^{\sum_i \delta_i f_i(\mu)}$$

$$\rightarrow L(\{\lambda_i+\delta_i\})-L(\{\lambda_i\}) \geq \left[\sum_i \delta_i f_i(w)+1-\sum_\mu P(\mu)e^{\sum_i \delta_i f_i(\mu)}\right]$$

Result : (Jensen's Inequality)

If $p(\alpha)$ is a p.m.f $(\forall\alpha, p(\alpha)>0, \sum_\alpha p(\alpha)=1)$

Then, $e^{\sum_\alpha p(\alpha)q(\alpha)} \leq \sum_\alpha p(\alpha)e^{q(\alpha)}$

<u>Define</u>: $F(\mu)=\sum_i f_i(\mu)$

Then, $\left\{\frac{f_i(\mu)}{F(\mu)}\right\}$ is a valid p.m.f

$$\sum_i \frac{f_i(\mu)}{F(\mu)} = 1 \; ; \quad \forall i, \frac{f_i(\mu)}{F(\mu)}. > 0$$

Look into: $e^{\sum_i \delta_i f_i(\mu)} = e^{F(\mu)\sum_i \delta_i \frac{f_i(\mu)}{F(\mu)}}$

$$= e^{\sum_i \delta_i F(\mu)\frac{f_i(\mu)}{F(\mu)}} \leq \sum_i \frac{f_i(\mu)}{F(\mu)} e^{\delta_i F(\mu)}$$

By using Jensen's Inequality:

$$e^{\sum_i \delta_i F(\mu)\frac{f_i(\mu)}{F(\mu)}} \geq -\sum_i \frac{f_i(\mu)}{F(\mu)} e^{\delta_i F(\mu)}$$

So;

$$L(\{\lambda_i + \delta_i\}) - L(\{\lambda_i\})$$

$$\geq \sum_w \widetilde{P}(w)\left[\sum_i \delta_i f_i(w) + 1 - \sum_\mu P(\mu) e^{\sum_i \delta_i F(\mu)\frac{f_i(\mu)}{F(\mu)}}\right]$$

$$\geq \sum_w \widetilde{P}(w)\left[\sum_i \delta_i f_i(w) + 1 - \sum_\mu P(\mu)\sum_i \frac{f_i(\mu)}{F(\mu)} e^{\delta_i F(\mu)}\right]$$

Let,

$$g(\{\delta_i\}) = \sum_i \left[\sum_w \widetilde{P}(w)\delta_i f_i(w)\right] + 1 - \sum_i \left[\sum_w \sum_\mu \widetilde{P}(w)P(\mu)\frac{f_i(\mu)}{F(\mu)} e^{\delta_i F(\mu)}\right]$$

$$\frac{dg}{d\delta_i} = \sum_w \tilde{P}(w) f_i(w) - \sum_w \sum_\mu \tilde{P}(w) P(\mu) \frac{f_i(\mu)}{F(\mu)} F(\mu) e^{\delta_i F(\mu)} \quad (*)$$

$$= \sum_w \tilde{P}(w) f_i(w) - \sum_\mu P(\mu) f_i(\mu) e^{\delta_i F(\mu)}$$

$$= \sum_w \tilde{P}(w) f_i(w) - \sum_w P(w) f_i(w) e^{\delta_i F(w)}$$

$$\frac{dg}{d\delta_i} = 0 = \sum_w \tilde{P}(w) f_i(w) - \sum_w P(w) f_i(w) e^{\delta_i \sum_i f_i(w)}$$

Algorithm:

1. Start with some λ
2. Repeat until convergence

   At the k-th step:

   (a) Solve for $\{\delta_{i,k}\}$ using *

   (In (*) use $\{\delta_{i,k}\}$)

   (b) $\lambda_{i,k+1} = \lambda_{i,k} + \delta_{i,k}$

   Stop, when $|\lambda_{i,k+1} - \lambda_{i,k}| < Epsilon$

$$(*) \sum_w \tilde{p}(w) f_i(w) = \sum_w P(w) f_i(w) e^{\delta_i \sum_i f_\#(w)}$$

$$f_\#(w) = \sum_i f_i(w)$$

And finally, we end up with the below equation for updating parameter values:

$$= \sum_w P(w) f_i(w) e^{\delta_i f_\#(w)} = \sum_w \tilde{p}(w) f_i(w)$$

REFERENCES

1. 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.

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

3. 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*, Volume. 29 Issue. 4, pp. 713-722, Elsevier Ltd.,2005.

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

5. Kazienko, P., *Intrusion Detection Systems (IDS) Part 1 – network intrusions, attack symptoms; IDS tasks; and IDS architecture*, http://www.windows.security, April 2003.

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

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

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

9. Stallings, W., *Data and Computer Communications*, Prentice Hall, New York, 2007.

10. “Transport Control Protocol”, http://en.wikipedia.org/.

11. “User Datagram Protocol”, http://en.wikipedia.org/.

12. SNORT: The Open Source Network Intrusion Detection System, http://www.snort.org/.

13. Norton, M., “Optimizing Pattern Matching for Intrusion Detection” *IDS Research Papers*, 2004

14. Feather, F. and R. Maxion, “Fault detection in an ethernet network using anomaly signature matching,” *Proceedings of. ACM SIGCOMM*, vol. 23, pp. 279–288. San Francisco, CA, Sept. 1993,

15. BRUTLAG, J. D., “Aberrant behavior detection in time series for network service Monitoring”, *Proceeding of the 14th Systems Administration Conference*, pp. 139-146. New Orleans, Louisiana, 2001

16. Wang, H., D. Zhang and Kang G. S., “Change-Point Monitoring for the Detection of DoS Attacks”, *IEEE transactions on dependable and secure computing*, Vol. 1, No. 4, October-December 2004.

17. Barford, P., J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies”, *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pp. 45-54, 2002.

18. Huang, C., S. Thareja, Y. Shin, “Wavelet-based Real Time Detection of Network Traffic Anomalies”, *International Journal of Network Security*, pp. 23-35, 2006

19. Rawat, S. and C. Sastry, “Network Intrusion Detection Using Wavelet Analysis”, *Lecture Notes in Computer Science*, pp. 224-232, 2004.

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

21. Bolzoni D., S. Etalle and P. Hartel, "POSEIDON: a 2-tier Anomaly- based Network Intrusion Detection System", *Preceedings of Fourth IEEE International Workshop on Information Assurance*, pp. 10-24, 2006.

22. Lichodzijewski P., A.N. Zincir-Heywood and M.I. Heywood., "Dynamic Intrusion Detection Using Self Organizing Maps", *14th Annual Canadian Information Technology Security Symposium*, May 2002.

23. Rhodes B., J. Mahaffey, J. D. Cannady, "Multiple Self-Organizing Maps for Intrusion Detection", *Proceedings of the NISSC 2000 Conference*, Baltimore M.D. 2000.

24. Moradi M., M. Zulkernine. "A Neural Network Based System for Intrusion Detection and Classification of Attacks", *Soft Computing Journal*, vol. 1, pp. 6-18, Springer-Verlag 1997

25. Wang W. and R. Battiti, "Identifying Intrusions in Computer Networks bases on Principal Component Analysis", *Proceedings of 15th Annual Computer Security Applications Conference (ACSAC '99)*, Phoenix, AZ, pp. 371-377, 1999.

26. Labib K and V. R. Vemuri, "Detecting and Visualizing Denail-of-Service and Network Probe Attacks using Principal Component Analysis", *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, pp. 1714-1719, 2004

27. Kim S., and A. L. Narasimha, "Image-based Anomaly Detection Technique: Algorithm, Implementation and Effectiveness", *IEEE Journal on Selected Areas in Communications - High-Speed Network Security*, Volume. 24, Oct. 2006

28. Wu, Q., Z. Shao, "Network Anomaly Detection Using Time Series Analysis", *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, 2005.

29. Soule A., Salamatian K., Taft N., "Combining Filtering and Statistical Methods for Anomaly Detection", *Proceedings of the 10th USENIX Security Symposium*, pp. 9–22. 2001

30. Data sets Overview, http://www.ll.mit.edu/IST/ideval/data/data_index.html

31. 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.

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

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

34. Adam L., Stephen A., Vincent J., "A Maximum Entropy Approach to Natural Language Processing", *Lecture notes on Machine Learning*, MIT, Fall 2003.

35. Internet Assigned Numbers Authority , http://www.iana.org/assignment/port-numbers

36. 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.

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

38. 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

39. Cluster Analysis, " http://en.wikipedia.org/wiki/Cluster_analysis”

40. 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