University of Crete
Department of Computer Science

FO.R.T.H.
Institute of Computer Science

A statistical Approach for Intrusion Detection

MSc. Thesis

Ioannis Sfakianakis

Heraklion

February 2007

Department of Computer Science

University of Crete

A statistical Approach for Intrusion Detection

Submitted to the

Department of Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science

February 1, 2007

Abstract

Since the Internet's growth, network security plays a vital role in the computer industry. Attacks are becoming much more sophisticated and this fact lead the computer community to look for better and advanced anti-measures. Malicious users existed far before the Internet was created, however the Internet gave intruders a major boost towards their potential compromisations. Naturally, the Internet provides convenience and comfort to every users and "bad news" is merely an infelicity. Clearly the Internet is a step forward; it must be used for the correct reasons and towards the right cause, nevertheless.

As computer technology becomes more elaborate and complex, programme vulnerabilities are more frequent and compromisations effortless. A means of attack containment are the so called "Intrusion detection systems" (IDS).

In this thesis we built a network anomaly IDS, using statistical properties from the network's traffic. We were interested in building general purpose, adaptive and data independent system with as few parameters as possible. The types of attacks it can detect are Denial of Service attacks and probing attacks. We used three models for our experiments; Fisher's Linear Discriminant, Gaussian mixture model and Support vector machines.

In our experiments we found that the most important part of statistical intrusion detection is the feature selection. Better results can be achieved when both classes are modeled (attack and normal traffic). Best results were achieved using Fisher's Linear Discriminant method, that is 90% detection rate with 5% false alarm rate

Acknowledgements

I would like to thank my supervisor Dr. Ioannis Stylianou for his guidance and his enlightenment in several aspects of this study. I would like to thank co-supervisor Dr. Euaggelos Markatos for his support and making this work possible.

I would like to thank all my lab partners in Signal Processing Lab (Miltos Vasilakis, Elena Karamihali, Christine Lionoudaki) for coping with my eccentricity, for being there when I needed any help, for entertaining me and for their encouragement in general.

I would like to thank my family for their financial support and their belief in my abilities.

Finally, I would like to thank Miltos Vasilakis for his Roc Curve implementation, Christos Tzagarakis for providing me a GMM implementation and Ioannis Pantazis for his GMM and SVM implementation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet is a worldwide publicly available network of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol (IP) (see figure 1.1). It is a "network of networks" that consists of million of smaller domestic, academic, business and government networks, which together carry various information and services, such as electronic mail, on-line chat, file transfer, and the interlinked Web pages and other documents of the World Wide Web.

The Internet in the last decades has evolved into the greatest and most popular network in the world. With the rapid growth of the Internet, networked computers are becoming more and more vital in our society. Despite all the benefits the Internet brings, many drawbacks come with it as well. It has been reported that the number of computer attacks has been increasing exponentially in the last years. In addition, sophistication and severity of the attacks have been rising. The fact that attacking tools are widely available, helped notably the malicious users.

To protect computers from being infected, defensive mechanisms were developed (antivirus, firewall, antispywere, etc). Depending on the infection, each software performs a different action. In this thesis we are going to built a new protection mechanism based on statistical models that will hopefully prevent intruder from illegal acts. Below we describe in details some types of observed attacks.

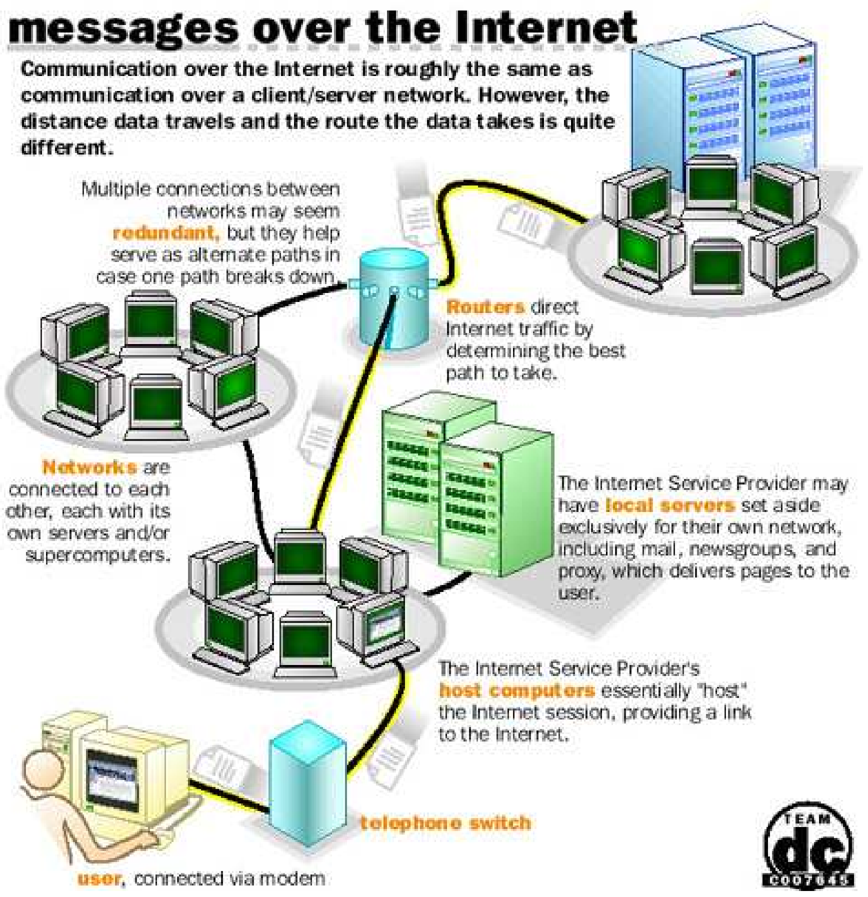

messages over the Internet
Communication over the Internet is roughly the same as communication over a client/server network. However, the distance data travels and the route the data takes is quite different.
Multiple connections between networks may seem redundant, but they help serve as alternate paths in case one path breaks down.
Routers direct Internet traffic by determining the best path to take.
Networks are connected to each other, each with its own servers and/or supercomputers.
The Internet Service Provider may have local servers set aside exclusively for their own network, including mail, newsgroups, and proxy, which delivers pages to the user.
The Internet Service Provider's host computers essentially "host" the Internet session, providing a link to the Internet.
telephone switch
user, connected via modem
TEAM dc C007645

Figure 1.1: A general view of the internet.

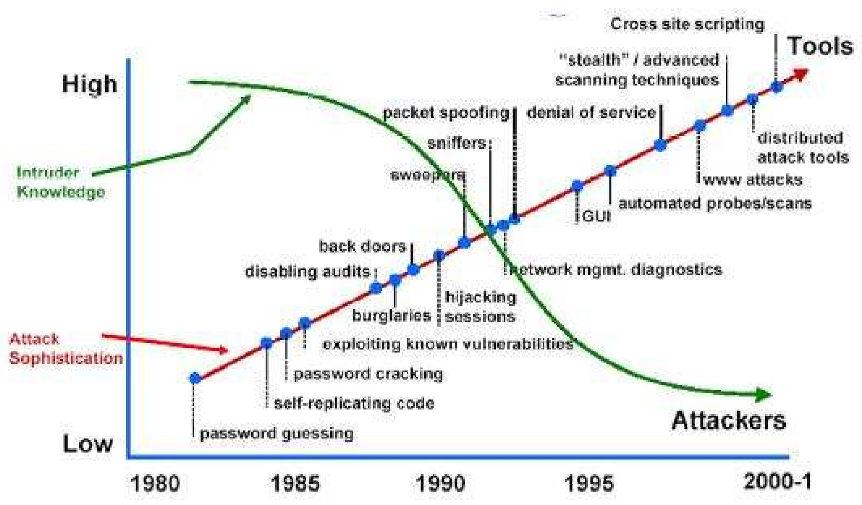


Figure 1.2: Attackers' Knowledge versus attack sophistication

1.1 Attacks

There exist a really fulfilling attack categorisation in [48]. On this basis, attacks can be categorised depending on:

- The attack type
- The number of network connections involved in the attack (NoC)
- The source of the attack
- The environment
- The automation level

1.1.1 Attack type taxonomy

Denial of Service attacks: These attacks attempt to "shut down a network, computer, or process; or otherwise deny the use of resources or services to authorized users" [61]. There are two types of DoS attacks: (i) operating system attacks, which target bugs in specific operating systems and can be fixed with patches; (ii) networking attacks, which exploit inherent limitations of networking protocols and infrastructures. Typical example of networking DoS attack is a "SYN flood" attack (synchronisation flood attacks), which takes advantage of three-way handshake for establishing a connection. In this attack, attacker establishes a large number of "half-open" connections using IP spoofing. The attacker first sends SYN packets with the spoofed (faked) IP address to the victim in order to establish a connection. The victim creates a record in a data structure and responds with SYN/ACK message to the spoofed IP address, but it never receives the final acknowledgment message ACK for establishing the connection, since the spoofed IP addresses are unreachable or unable to respond to the SYN/ACK messages. Although the record from the data structure is freed after a time out period, the attacker attempts to generate sufficiently large number of "half-open" connections to overflow the data structure that may lead to a segmentation fault or locking up the computer. DoS and DDoS attacks have posed an increasing threat to the Internet, and techniques to thwart them have become an active research area [72, 66, 89, 68, 86]. Researchers that analyze DoS attacks have focused on two main problems: (i) early detection mechanisms and identification of ongoing DoS activities [50, 33, 85, 93]; (ii) response mechanisms for alleviating the effect of DoS attacks (e.g. damage caused by the attack). Response mechanisms include identifying the origin of the attack using various traceback techniques [42, 45, 13] and slowing down the attack and reducing its intensity [59, 61, 21] by blocking attack packets. In addition to these two main approaches, some systems use measures to suppress DoS attacks.

Probing (surveillance, scanning): These attacks scan the networks to identify valid IP addresses and to collect information about them (e.g. what services they offer, operating system used). Very often, this information provides an attacker with a list of potential vulnerabilities that can later be used to perform an attack against selected machines and services. Examples of probing attacks include IPsweep (scanning the network computers for a service on a specific port of interest), portsweep (scanning through many ports to determine which services are supported on a single host), nmap (tool for network mapping), etc. The existing scan detection schemes essentially look for IP addresses that make more than N connections in T seconds. Unfortunately, tools based on these techniques are quite inefficient at detecting slow/stealthy scans or scans targeted specifically at the monitored enterprize - the type of scans that analysts would really be interested in. Due to these reasons, sophisticated adversaries typically attempt to adjust their scans by reducing the frequency of their transmissions in order to avoid detection. For detecting stealthy scans, there are a few recently proposed more sophisticated techniques based on collecting various statistics [49, 20, 74, 65, 38, 28].

Compromises: These attacks use known vulnerabilities such as buffer overflows [9] and weak security points for breaking into the system and gaining privileged access to hosts. Depending upon the source of the attack (outside attack vs. inside attack), the compromises can be further split into the following two categories:

1. R2L (Remote to Local) attacks

   In this attack the intruder gains access (either as a user or as root) to a machine over a network, without his having an account on that machine. In most R2L attacks, the attacker breaks into the computer system via the Internet. Typical examples of R2L attacks include guessing passwords (e.g. guest and dictionary attacks) and gaining access to computers by exploiting software vulnerability (e.g. phf attack, which exploits the vulnerability of the phf program that allows remote users to run arbitrary commands on the server).

2. U2R (User to Root) attacks

   An attacker who has an account on a computer system is able to misuse/elevate their privileges by exploiting a vulnerability in computer mechanisms, a bug in the operating system or in a program that is installed on the system. Unlike R2L attacks, where the hacker breaks into the system from the outside, in U2R compromise, the local user/attacker is already in the system and typically becomes a root or a user with higher privileges. The most common U2R attack is buffer overflow, in which the attacker exploits the programming error and attempts to store more data into a buffer that is located on an execution stack. Since buffers are created to contain a specific amount of data, the additional information used by the attacker can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. This data may contain codes designed to trigger specific actions, such as damaging users files or providing the user with root access. Many approaches have recently been proposed for detection and prevention of buffer overflow attacks [34, 53] due to increased interest in them. It is important to note that buffer overflow attacks can also belong to R2L attacks, where remote users attempts to compromise the integrity of target computer. For example, a vulnerability discovered in Microsoft Outlook and Outlook Express in July 2000 [6] allowed the attackers to simply send an e-mail message and to overflow the specific areas with superfluous data, which allowed them to execute whatever type of code they desired on the recipient's computers.

Viruses/Worms/Trojan horses: They are programs that replicate on host machines and propagate through a network.

1. Viruses are programs that reproduce themselves by attaching them to other programs and infecting them. They can cause considerable damage (e.g. erase files on the hard disk) or they may only do some harmless but annoying tricks (e.g. display some funny messages on the computer screen). Viruses typically need human interaction (e.g. trading files on a floppy or opening e-mail attachments) for replication and spreading to other computers. One of the most well known virus examples is Michelangelo virus that infects the hard disks master boot record and activates a destructive code on March 6, which is Michelangelo's birthday. There are various types of viruses, classifying them is not easy, as many viruses have multiple characteristics and may belong into multiple categories. Most commonly viruses are classified depending on: the environment, the operating system, different algorithms of work and destructive capabilities [64], although there are other categorizations based on what and how viruses infect [22, 35].

2. Worms are self-replicating programs that aggressively spread through a network, by taking advantage of automatic packet sending and receiving features found on many computers. Worms can be organized into several categories:
    - traditional worms (e.g. Slammer [8]) usually use direct network connections to spread through the system and do not require any user interaction.
    - e-mail (and other client application) worms, (e.g. Melissa worm [4]) infect other hosts on the network (Internet) by exploiting users e-mail capabilities or utilizing other client applications (e.g. ICQ - "I seek you").
    - windows file sharing worms (e.g. ExploreZip [?]) replicate themselves by utilizing MS Windows peer-to peer service, which is activated every time a networking device is detected in the system. This type of a worm very often occurs in combination with other attacks, such as MS-DOS and Windows viruses.
    - hybrid worms (e.g. Nimda [7]) typically exploit multiple vulnerabilities that fall into different categories specified above. For example, Nimda used many different propagation techniques to spread (e-mail, shared network drives and scanning for backdoors opened by the Code Red II and Sadmind worms). Success of Nimda demonstrated that e-mail and http traffic are effective ways to penetrate the network system, and that the file sharing is quite successful in replicating within the system [83].

    It is important to note that some of the worms that appeared recently have also been used to launch DoS attacks [57]. For example, the erkms and li0n worms were used to deploy DDoS tools via BIND vulnerabilities [57], while Code Red was used to launch TCP SYN DoS attacks [57]. However, traditional DoS attacks typically target a single organization, while worms (e.g. SoBig.F worm) typically affect a broad range of organizations. Over the last few years, many DoS attacks have gradually mutated and merged with more advanced worms and viruses (e.g. Blaster worm in August 2003). Analysts also expect that in the future DoS attacks will be more often part of worm payloads [57].

3. Trojan horses Trojan horses are defined as "malicious, security-breaking pro-

grams" that are disguised as something benign [56]. For example, the user may download a file that looks like a free game, but when the program is executed, it may erase all the files on the computer. Victims typically download Trojan horses from an archive on the Internet or receive them via peer-to-peer file exchange using IRC/instant messaging/Kazaa etc. Some actual examples include Silk Rope and Saran Wrap. Many people use terms like Trojan horse, viruses and worms interchangeably since it is not easy to make clear distinction between them. For example, "Love Bug" is at the same time a virus, worm, and Trojan horse. It is a trojan horse since it pretends to be a love letter but it is a harmful program. It is a virus because it infects all the image files on the disk, turning them into new Trojan horses. Finally, it s also a worm since it propagates itself over the Internet by hiding in trojans that it sends out using peoples email address book, IRC client, etc.

1.1.2 NoC taxonomy

Attacks that involve multiple network connections: Typical examples of such attacks are DoS, probing and worms.

Attacks that involve single or very few network connections: Typical attacks in this category usually cause compromises of the computer system.(e.g buffer overflow)

1.1.3 Source of the attack

Computer attacks may be launched from a single location (single source attacks) or from several different locations (distributed/coordinated attacks). Most of the attacks typically originate from a single location (e.g. simple scanning), but in the case of large distributed DoS attacks or other organized attacks, multiple source locations may participate in the attack. In addition, very often distributed/coordinated attacks are targeted not only to a single computer, but also to multiple destinations. Detecting such distributed attacks typically requires the analysis and correlation of network data from several sites.

1.1.4 Environment

Intrusions on the host machine: Intrusions that occur on a specific machine, which may not even be connected to the network. These attacks are usually detected by investigating the system information (e.g. system commands, system logs). The identity of the user that performs an attack in this case is typically associated with the username, and is therefore easier to discover.

Network intrusions: Intrusions that occur via computer networks usually from outside the organization. Detection of such intrusions is performed by analyzing network traffic data (e.g. network flows, tcpdump data). However, such analysis often cannot reveal the precise identity of the attackers, since there is typically no direct association between network connections and a real user.

Intrusions in a P2P environment: Intrusions that occur in a system where connected computers act as peers on the Internet. Unlike standard "client/server" network architectures, in P2P environment, the computers have equivalent capabilities and responsibilities and do not have fixed IP address. They are typically located at "the edges of the Internet" [90], and actually disconnected from the DNS systems. Although P2P file sharing applications can increase productivity of enterprize networks, they can also introduce vulnerabilities in them, since they enable users to download executable codes that can introduce rogue or untraceable "backdoor" applications on users' machines and jeopardize enterprize network security.

Intrusions in wireless networks: Intrusions that occur between computers connected through wireless network. Detection of attacks in wireless networks is based on analyzing information about the connections in wireless networks, which is typically collected at wireless access points [92].

1.1.5 Automation level

Depending on the level of the attack automation, there are several categories of attacks as follows:

Automated attacks Attacks that use automated tools that are capable of probing and scanning a large part of the Internet in a short time period. Using these easily available tools, even inexperienced attackers may create highly sophisticated attacks. Such attacks are probably the most common method of attacking the computer systems today.

Semi-automated attacks Attack that deploy automated scripts for scanning and compromise of network machines and installation of attack code, and then use the handler (master) machines to specify the attack type and victims address.

Manual attacks Attacks that involve manual scanning of machines and typically require a lot of knowledge and work. Manual attacks are not very frequent, but they are usually more dangerous and harder to detect than semi-automated or automated attacks, since they give to attackers more control over the resources. Experts or organized groups of attackers generally use these attacks for attacking systems of critical importance.

1.2 Intrusion Detection/Prevention Systems (IDS/IPS)

1.2.1 IDS Taxomonomy

There has been many attempts to categorize Intrusion Detection Systems. We will use one that has been proposed in [48].

1. ***Information source.*** This criterion discriminates between IDS based on what source the information comes from, e.g. a host machine, a network etc.

    - *Host based.* They analyse users' normal behaviour and habits on a specific host. This provides host based IDSs the advantage of having high quality data. The drawback however is that it may be create a major processing load on the machine they are running. Aside from that, audit sources (in host based intrusion analysis), can be easily modified by a successful attack. Hence, the host-based IDS must process so fast the audit trail as to be able to raise alarms before the attacker can modify the audit trail.

There are several helpful information types used in host-based IDSs, e.g. system commands [26, 52, 76, 43], system accounting [19, 30], syslog [2, 25, 37, 39] etc.

- *Network based.* They search for network attacks, that have been more popular since the growing of the Internet's esteem. Apart from that, analysing packet flow and payload several intrusions against servers can be detected [10, 1]. Network based IDSs have several benefits compared to host-based IDSs. Firstly, they do not burden the mainframes. Secondly, they are more resistant since they do not reside on the hosts that could be compromised. Thirdly, they usually do not depend on the operating systems and can extract information at network level. Last but not least, they can be placed carefully at certain routers and nodes generally where all traffic flows can be observed. A considerable drawback however is the high possibility for dropping packets and they are unable to perform detection on encrypted data.
- *Application log files.* They monitor only specific applications such as database management systems, content management systems, etc. Application based IDSs have access to information types that the former IDS do not have. For instance, analysing application log files, application based IDSs can detect many attack types, suspicious activity that may became tricky for a host or a network based IDS. In addition, they are able to identify unauthorized activities from individuals or to analyse encrypted data by employing application-based services [18].
- *Wireless networks.* They try to contain intrusions in wireless networks. The modern trend towards wireless networks has lead to many serious threats have been emerged in wireless networks, with potentially devastating results, which are due to the following reasons [12, 73, 92]:
  - Physical layer in wireless networks is essentially a broadcast medium and therefore less secure than in fixed computer networks. For example, an attacker that enters the wireless network, bypasses existing security mechanisms and can easily sniff sensitive and confidential information. In addition, the attacker also has access to all the ports that are regularly available

only to the people within the network. In wired networks, attempts to access these ports from outside world through Internet are stopped at the firewalls. Finally, the attacker can also excessively load network resources thus causing denial of service to regular users.

  – There are no specific traffic concentration points (e.g. routers) where packets can be monitored, so each mobile node needs to run an intrusion detection system.

  – Separation between normal and anomalous traffic is often not clear in wireless ad-hoc networks, since the difference between compromised or false node and the node that is temporarily out of synchronization due to volatile physical movement can be hard to observe.

2. ***Analysis Strategy.*** There are two major techniques for attack detection. The so called misuse detection is one and anomaly detection the other. Misuse detection, uses the extensive information about existing attacks (extracted by human experts). These kind of approaches can detect exceptionally accurately known vulnerabilities. However, they are unable to detect unknown attacks.

   On the other hand, anomaly detection, creates model for normal behaviour of users, hosts or network connections. Numerous methods have been applied for the characterisation of legitimate traffic. The main benefit of anomaly detection compared to misuse is that they can potentially detect unknown attacks. However, they have usually high false alarm rate.

   • *Misuse detection.* Misuse detection can be classified into the subsequent classes:

     Signature based techniques. These IDSs operate analogously to virus scanners. They search a database of signatures for a known signature for each specific In signature-based IDSs, monitored events are matched against a database of attack signatures to detect intrusions. Each signature has to be manually built for each new type of intrusion that is discovered. Typical examples of signature based IDS are, snort [11], NetRanger [41], RealSecure [3], etc.

Rule-based systems. Rule-based systems, consist of a set of if-then-else commands to detect vulnerabilities. Packet traffic is observed carefully and events regarding attacks are converted into rules. These rules are used later for traffic labeling. Some rule-based IDSs are, IDES [27, 44, 87, 58], NIDX [17], ComputerWatch [29], P-BEST [55] , AutoGuard [79, 80], Piranha [15], E2xB [16] etc.

Methods based on state-transition analysis. State-transition systems utilize a finite state machine. Depending on the event that has been monitored, the FSM transists from a state to another. Different states correspond to different states of the network protocol stacks or to the integrity and validity of current running processes or certain files. USTAT [70], NetSTAT [88] and STRIDE [69] are such IDSs.

Data mining based techniques. Data mining systems, contain a learning algorithm that is trained over certain labeled data (normal or intrusive). These systems can smoothly adjust to different input data (different network in other words) as long as, they are appropriately labeled. Unlike signature-based intrusion detection, models are created automatically, this leads to more precise and sophisticated signatures. MADAM ID [84, 54], Earlybird and "The Network that Never Forgets" [14] are systems that use data. minining techniques.

- *Anomaly detection.* The interested in anomaly detection techniques has been raised substantially, due to the increasing number of unseen computer attacks. There exist many types of anomaly detection algorithms proposed in the literature that differ according to the information used for analysis and behaviour. These techniques can be chategorised in five groups, statistical methods, rule based methods, distance based methods, profiling methods and model based approaches.

Statistical methods. Statistical methods measure specific variables over a specific time window. After the variables have been computed the system checks whether certain thresholds have been exceeded. Some advanced statistical models create long-term and short-term user activities. Such anomaly detection algorithms were used in IDES [27, 44, 87, 58], EMERALD [67], and SPADE [38]

Distance based methods. Statistical models have difficulty of detecting outliers in higher dimensional spaces, as it is extremely inaccurate to estimate multidimensional distributions. In distance based approaches we compute the distance between the points. As a result, the limitation statistical models is eliminated. MINDS [51] is an example of this method.

Rule based methods. Rule based methods characterise what consists normal behaviour by a set of rules for users, networks and/or computer systems. ComputerWatch [29] is a typical Rule based system.

Profiling methods. In profiling methods, normal behaviour for network traffic, users, programmes etc, is build individually (one profile for each entity). Profiling methods have a number of utilisations ranging from data mining techniques to heuristic-based approaches. ADAM [91], PHAD [60], ADMIT [81] are examples of these methods.

Model based methods. In the model-based approaches, anomalies are detected as deviations for the model that represents the normal behavior.

3. ***Prediction Time.*** There two main sets considering the prediction Time: real-time (on-line) IDSs and off-line IDSs. Real-time IDSs must raise an alarm as soon as an attack is detected, as a result the proper action will be taken.

   Off-line IDSs investigate stored audit data. This method of audit data analysis is common among security analysts who often examine network behavior, as well as behavior of different attackers, in an off-line (batch) mode. Many early host-based IDSs used this timing scheme, since they used operating system audit trails that were recorded as files.

   Off-line analysis is also often performed using static tools that analyze the snapshot of the environment (e.g. host vs. network environment), look for vulnerabilities and configuration errors and assess the security level of the current environment configuration.

4. ***Architecture.*** IDSs may be centralised or distributed depending on the architecture. There has been observed a recent trend towards distributed IDSs, due to the increasing coordinated and distributed attack [82]. Centralised architectures are still dominant, however. Many modern attacks are unnoticeable by analysing a singe node (which is the common practice for most IDSs). Despite several, drawbacks of distributed IDSs, many commercial vendors adapted their system to detect distributed cyber attacks.

5. ***Response.*** IDSs can respond in an possible compromisation in two ways, either passively or actively. The majority of IDSs react passively; they just sound an alarm to the responsible staff.

   On the other hand, IDSs may also take preventing measures. Specifically, an IDS could log off a user, reconfigure routers and firewalls or disconnect a port.

   One of the most harmless, but often most productive, active responses is to collect additional information about a suspected attack and to perform damage control. This might involve increasing the sensitivity level of information sources (e.g., increasing the number of events logged by an operating system audit trail, or increasing the sensitivity of a network monitor that captures all packets). This additional information is most likely to help us in resolving the detection of the attack (assisting the system in diagnosing whether an attack did or did not take place in the first place) thus allowing the IDS to gather information that can be used to support investigation of the attacker.

1.2.2 IDS Performance

Prediction Performance. Intrusion Detection systems need to classify correctly both intrusions and legitimate actions. False alarm rate and detection rate are the most typical measures for evaluating IDS performance. Detection rate is ratio of the number of successfully detected attacks and the total number of attacks, while false alarm rate is the ratio of the misclassified normal connections and the total number of

`Traffic`	**Detected attacks**	**Detected normal**
Attacks	True positives	False negatives
Normal	False positives	True negatives

Table 1.1: Intrusion attack evaluation

connections (table 1.1). A visual representation of these rate is performed using ROC (Receiver Operating Characteristics) curves. The ROC curve is a plot of detection rate versus false alarm rate [75].

Time Performance. The time performance is another important measure for an IDS. It represents the time needed for the IDS to decide whether the observed traffic is intrusive or not! If this rate is sufficiently small then real-time prediction is out of the question and any damage will be difficult to be contain quickly.

Fault tolerance. The IDS should be independent, robust and resistance to attacks. It should be able to recover quickly from successful attacks and to continue provide a secure service. This is especially true in the case of very large DoS attacks, buffer overflow attacks and various deliberate attacks that can shut down the computer system and thus IDS.

For more information see [25, 70]

In this thesis we are going to build a network based and anomaly detection IDS, that uses statistical properties for the detection of the attacks. This IDS will detect two attack types, DoS attacks and probing attacks. The statistical models tested in this study were, the Gaussian Mixture Model, the Fisher's Linear Discriminant and the Support Vector Machines. The required features for the charactersation of the normal traffic were drown from transferred packets' header through the network. The features extracted were the following: For a window of 10 seconds, the number of connections, the number of packets and bytes transferred, the maximum number of connections to a specific destination, the number of packets and bytes transferred to a specific destination, the number of distinct services and the differential of the number of connections during the time window.

The remainder of this thesis is organised as follows. Chapter 2 focuses on the previous work in the field of anomaly detection. In Chapter 3 we provide the required background of the statistical models we used. In Chapter 4 we illustrate and explain our results. Finally a summary and a possible future work are provided in the last Chapter.

Chapter 2

Previous Work

2.1 Statistical Traffic Modeling for Network Intrusion Detection

In [77] a statistical system for Network Intrusion Detection is described. This is used to detect Denial of Service attacks and scanning attacks by monitoring network traffic volume, which is modeled as a poisson process. We recall some properties of poisson processes below.

Poisson process. Suppose that X(t) defines the number of events taking place in time interval [0,t). If the number of waiting time before an event occurs follows exponential PDF (i.i.d.), with common parameter λ then X(t) is a Poisson Process.

1. $P[X(t) = k] = \frac{\lambda^k}{k!}e^{-\lambda}$
2. $X(t_2) - X(t_1), X(t_3) - X(t_2), \ldots, X(t_n) - X(t_{n-1})$ are independent for $0 \leq t_1 \leq t_2 \leq \ldots \leq t_n$
3. If $X_1(t), X_2(t), \ldots, X_n(t)$ are independent Poisson processes with parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$, then $Y(t) = \sum_{i=1}^{n} X_i(t)$ is also a Poisson process with parameter $\lambda = \sum_{i=1}^{n} \lambda_i$

In [77] DARPA's 1998 Offline Intrusion Detection Evaluation was used for the experimental work. This data set consists of 7 weeks of training data and 2 weeks of testing data. The weeks from 3 to 7 were only used for attack characterization.

The following notations were introduced in [77],

- $X_N^T(k)$: Total number of normal connections in a time interval $[kT, (k+1)T)$.
- $X_A^T(k)$: Total number of attack connections in a time interval $[kT, (k+1)T)$.
- $X_C^T(k)$: Total number of connections in a time interval $[kT, (k+1)T)$. Obviously $X_C^T(k) = X_N^T(k) + X_A^T(k)$.

and the following features were observed in the DARPA dataset,

- **Diurnal patterns for the normal connections.** There are three operating regimes for $X_N^T(k)$. *Day regime* from 8 am to 4 pm, a *evening regime* from 4 pm to 8 pm and a *night regime* from 8 pm to 6 am the next day.
- **Poisson models for the normal connections.** For the *day* and *night* regimes, $X_N^{10}(k)$ and $X_N^{100}(k)$ were used.
- **Heterogeneous behaviour for the attacks connections.** The DoS and Probe attacks appear in bursts, with widely varying amplitudes, which are hard to characterize statistically.

The attack and the normal intervals were defined as follows:

$$\begin{aligned} X_A^T(k) &> X_N^T \Rightarrow [kT, (k+1)T)\text{window, attack interval} \\ X_A^T(k) &\leq X_N^T(k) \Rightarrow [kT, (k+1)T)\text{window, normal interval} \end{aligned}$$

In [77] the discriminating power of $X_C^T(k)$ is provided and it is reproduced in table 2.1. We observe that results for the *day* regime are substantially better if a window interval 100 seconds is chosen. However, for *night* regime better results are observed if 10 seconds window is used.

Regime	Window	Detection rate	False positive rate
Day	10	62%	10%
Day	10	61%	5%
Day	10	60.5%	1%
Day	100	88%	10%
Day	100	85%	5%
Day	100	82%	1%
Day	1000	77%	10%
Day	1000	74%	5%
Day	1000	72.5%	1%
Night	10	63%	10%
Night	10	25%	1%
Night	100	18%	10%
Night	100	10.5%	1%
Night	1000	17%	10%
Night	1000	10%	1%

Table 2.1: Performance of the Statistical Traffic Modeling for Network Intrusion Detection for day and night regimes, 10, 100 and 1000 seconds length windows.

2.2 A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusion in Unlabeled Data.

Eleazar Eskin et al. [71] presented a geometric framework for unsupervised anomaly detection. Signature based-techniques provide detection for already known attacks, information extracted by human experts. Thus, much research has been done in data mining and supervised machine learning techniques. Supervised anomaly detection algorithms however, require a set of exclusively normal data for their training. If there are buried instances inside the training data the IDS will not be able to detect these instances in the future, because they will be assumed normal. Having purely normal data is not the case in practise that's why unsupervised anomaly detection algorithm are important.

The geometric framework that Eleazar Eskin et al. proposed, maps the data from $\mathcal{R}^d$ feature space to $\mathcal{R}$. Afterwards, outlier points are found and labeled. Anomalies are considered the points that are in sparse regions of the feature space. Once the mapping is performed, the same algorithms are applied to these different kinds of data.

The algorithms applied are all capable of handling high dimensional data efficiently enough. The first algorithm is a variant of cluster-based algorithm presented in [31]. The second algorithm is a k-nearest neighbour based algorithm. The third algorithm is a Support Vector Machine-based algorithm.

2.2.1 Feature Spaces for Intrusion Detection

The choice of feature space for unsupervised anomaly detection is application specific. The performance greatly depends on the ability of the feature space to capture information relevant to the application.

In these experiments, two data sets were used. The first data set is a set of network connection records. This data set contains records which contain 41 feature describing a network connection. The second data set is a set of system call traces. Each entry is a sequence of all of the system calls that a specific process makes during its execution.

Normalisation of the data is essential. Without normalisation an attribute could dominate all the others if it is too large. The attributes are normalised to the number of standard deviations away from the mean. This scales our distances based on the likelihood of the attribute values.

2.2.2 Results

Two data sets where used for the results. The first one was the KDD Cup 1999 Data [5]. The simulated attacks in the data set are organised into the following attacks:

DOS - Denial of Service.

R2L - Unauthorised access from a remote machine.

U2R - Unauthorised access to superuser or root functions.

Probing - surveillance access and other probing for vulnerabilities.

The KDD data set was obtained by simulating a large number of different types of attacks, with normal activity in the background. The goal was to produce a good training set for learning methods that use labeled attacks. AS a result, the proportion of attack instances

Programme name	Total # of Attacks	#Intrusion Traces	#Intrusion System Calls	# Normal Traces	# Normal System Calls	#Intrusion Traces
ps	3	21	996	208	35092	2.7%
eject	3	6	726	7	1278	36.3%

Table 2.2: Licoln Labs Data Summary

to normal ones in the KDD training data set is very large as compared to data that we would expect to observe.

In order to make the data realistic many of the attacks were filtered so that the resulting data would be 1% to 1.5% attack and 98.5% to 99% normal instances.

The second data set was from BSM data portion of the 1999 DARPA Intrusion Detection Evaluation data data created by MIT Lincoln Labs. The data consist of 5 week of BSM data of all processes run on a Solaris machine. Three weeks of programmes' traces were examined which were attacked during that time. The programmes examined were eject, and ps.

Table 2.2 summarises the system call trace data sets and list the number of system calls.

Each of the data set is split into two groups the training and the testing data. The parameters are set on the training set. Finally for each algorithm a ROC curve over the data set is obtained.

In the case of the system call data, each of the algorithm performed extremely satisfyingly. This means, that at a certain threshold, there was at least one process trace from each of the attacks identified as being malicious without any false positives.

As far as the network connections are concerned, data is not so regular as the system call traces. They have certain kinds of attacks that could not be detected. Table 2.3 shows the detection Rate and False positive Rate for some selected points from the ROC curves.

Algorithm	Detection rate	False positive rate
Cluster	93%	10%
Cluster	66%	2%
Cluster	47%	1%
Cluster	28%	0.5%
K-NN	91%	8%
K-NN	23%	6%
K-NN	11%	4%
K-NN	5%	2%
SVM	98%	10%
SVM	91%	6%
SVM	67%	4%
SVM	5%	3%

Table 2.3: Performance of each algorithm used in the geometric unsupervised anomaly detection approach over the KDD Cup 1999 Data.

Chapter 3

Statistical Properties

In this chapter, the background for the statistical models is provided. The performance of three models was tested; the Gaussian Mixture Model, the Fisher's Linear Discriminant and the Support Vector Machines. The GMM model and SVM are two very powerful learning tools that have been used with success in other areas (such as voice recognition). We expect to yield remarkable results in Intrusion Detection as well. The Fisher's Linear Discriminant is a very simple regression technique, that has provided noticeable results in clustering.

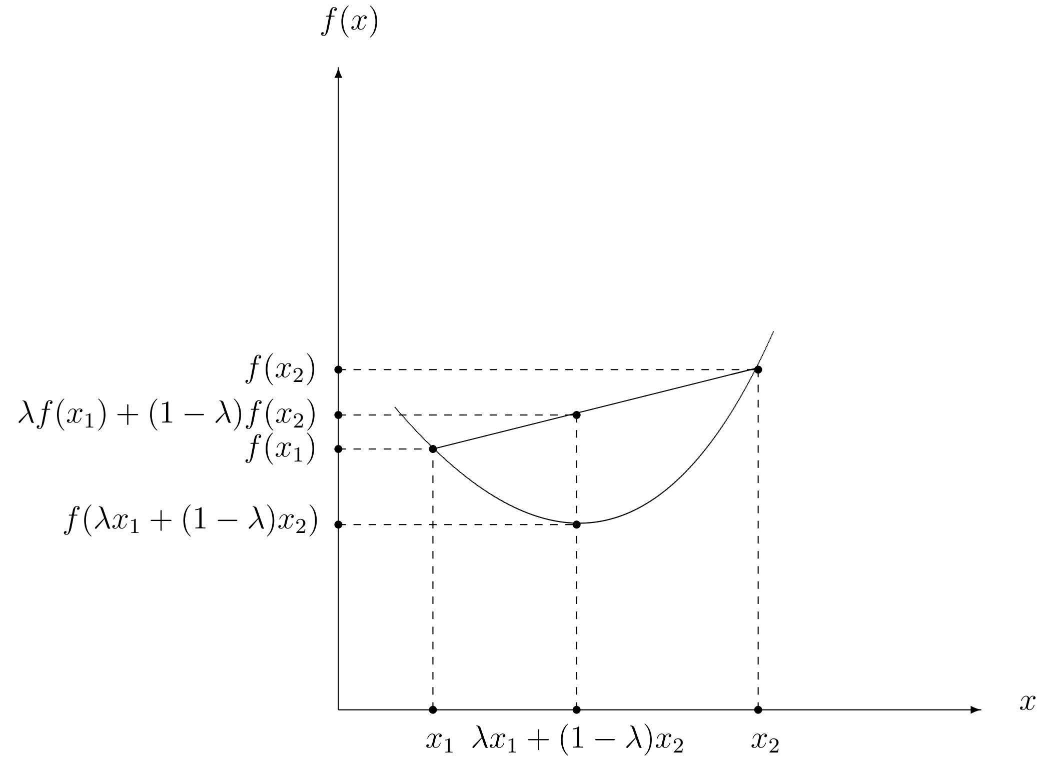
3.1 EM algorithm

The Expectation Maximization algorithm is a very popular tool in statistical processing, for estimation problems with missing (or incomplete) data, or in similar problems such as mixture estimation [78, 62].

3.2 Convex Functions

Let f denote a real valued function, $f : A \rightarrow \mathbb{R}$, where $A \subseteq \mathbb{R}$. f is said to be convex if $\forall x_1, x_2 \in A$ and $\lambda \in [0, 1]$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2). \tag{3.1}$$


$f(x)$
$f(x_2)$
$\lambda f(x_1) + (1-\lambda)f(x_2)$
$f(x_1)$
$f(\lambda x_1 + (1-\lambda)x_2)$
x
x_1
$\lambda x_1 + (1-\lambda)x_2$
x_2

Theorem 3.2.1 *Let f be a convex function on an interval A. If $x_1, x_2, \ldots, x_n \in A$ and $\lambda_1, \lambda_2, \ldots, \lambda_n \geq 0$ with $\sum_{i=1}^{n} \lambda_i = 1$,*

$$f\Big(\sum_{i=1}^{n} \lambda_i x_i\Big) \leq \Big(\sum_{i=1}^{n} \lambda_i f(x_i)\Big)$$

Proof. We will prove the theorem using the mathematical induction. For $n = 1$ $f(\lambda_1 x_1) \leq \lambda_1 f(x_1)$ is true. Suppose this is true for $n = 1, \ldots, k$. We will show that is also true for $n = k+1$.

$$\begin{aligned}
f\Big(\sum_{i=1}^{n+1}\lambda_i x_i\Big) &= f\Big(\lambda_{n+1}x_{n+1}+\sum_{i=1}^{n}\lambda_i x_i\Big)\\
&= f\Big(\lambda_{n+1}x_{n+1}+(1-\lambda_{n+1})\frac{1}{1-\lambda_{n+1}}\sum_{i=1}^{n}\lambda_i x_i\Big)\\
&\leq \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})f\Big(\frac{1}{1-\lambda_{n+1}})\sum_{i=1}^{n}\lambda_i x_i\Big)\\
&= \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})f\Big(\sum_{i=1}^{n}\frac{\lambda_i}{1-\lambda_{n+1}}x_i\Big)\\
&\leq \lambda_{n+1}f(x_{n+1})+(1-\lambda_{n+1})\Big(\sum_{i=1}^{n}\frac{\lambda_i}{1-\lambda_{n+1}}f(x_i)\Big)\\
&= \lambda_{n+1}f(x_{n+1})+\sum_{i=1}^{n}\lambda_i f(x_i)\\
&= \sum_{i=1}^{n+1}\lambda_i f(x_i)
\end{aligned}$$

Jensen's inequality will be used in the EM algorithm for the GMM estimation in the following manner:

$$ln\sum_{i=1}^{n}\lambda_i x_i \geq \sum_{i=1}^{n}\lambda_i ln(x_i).$$

3.2.1 The Expectation-Maximization Algorithm

The EM is an iterative method to estimate some unknown parameters Θ, given measurement data X. However, we are not given some missing variables Y. The observed and the missing data together form the complete data, which are essential for computing the unknown parameters Θ. Let p be the joint PDF of the complete data with parameters given by Θ: $p(X, Y|\Theta)$. In addition, note that the conditional distribution of the missing data given the observed can be expressed as:

$$p(Y|X,\Theta)=\frac{p(X,Y|\Theta)}{p(X|\Theta)}=\frac{p(X|Y,\Theta)p(X|\Theta)}{\int p(X|Y^*,\Theta)}$$

An EM algorithm improves an initial estimation of Θ (Θ_0). The estimation step takes the following form

$$\Theta_{n+1} = argmax_{\Theta} Q(\Theta)$$

where $Q(\Theta)$ is the expected log-likelihood. We do not know the complete data, so we cannot say what is the exact value of the likelihood, but given the data that we do know (the x's), we can find a posteriori estimates of the probabilities for the various values of the unknown y's. For each set of y's there is a likelihood value for Θ, and we can thus calculate an expected value of the likelihood with the given values of y's.

$$Q(\Theta) = \int_{-\inf}^{+\inf} p(y|x, \theta_n) log p(x, y|\Theta) dz$$

In other words, each iteration of the EM consists of two steps: The E-step and the M-step. In the expectation step (E-step), the missing data are estimated given the observed data and the current estimate of the model parameters. In the maximisation step (M-step), the likelihood function is maximized under the assumption that the missing data are known. The estimate from the E-step of the missing data are used in place of the actual missing data.

The EM algorithm can alternatively find the maximum a posteriori (MAP) estimates rather than the maximum likelihood, in the M-step. In this case, we have to maximise the following,

$$\Theta^* = argmax_{\Theta} log P(X, \Theta) = argmax_{\Theta} log \sum_{y \in Y} P(X, Y, \Theta)$$

The idea has been introduced previously, we start with an initial estimation of the parameters Θ^t and maximise the posterior. Instead we will use a lower bound $B(\Theta; \Theta^t)$ which is much easier than maximising $P(\Theta|X)$.

$$B(\Theta; \Theta^t) \triangleq \sum_{y \in Y} p^t(Y) log \frac{P(X, Y, \Theta)}{p^t(Y)}$$

By Jensen's inequality,

$$B \leq log \sum_{y \in Y} log \frac{P(X, Y, \Theta)}{p^t(Y)} = log P(X, \Theta)$$

3.2.2 Gaussian Mixture

For a random vector $x \in \mathbb{R}^{\mathrm{d}}$, a **density** mixture is defined as

$$p(x) = \sum_{s=1}^{k} \pi_s p(x|\theta_s) \tag{3.2}$$

where $p(x|\theta_s)$ is a distribution with parameters θ_s and k is the number of components the mixture has. Notice that, each density has a prior probability π_s, and π_s denotes a discrete prior distribution $\pi_s = p(s)$, over the components, where the random variable s takes values from 1 to k. To sum up, the mixture in general, is fully dependent of its parameters $\Theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k\}$, and these parameters define it completely.

A *Gaussian* mixture, is a mixture that uses Gaussian distributions. Thus, $\theta_s = \{m_s, C_s\}$ and $p(x|\theta_s) = (2\pi)^{-d/2}|C_s|^{-1/2}\exp[-(x-m_s)^{\top}C_s^{-1}(x-m_s)/2]$, for this case.

Imagine a set of independent, identically distributed samples $\{x_1, \ldots, x_n\}$, from a k-component density mixture $p(x)$. The learning task is to estimate the parameters Θ of the mixture. We assume the best parameters of a mixture are the ones, that maximize the log-likelihood $\mathcal{L}(\Theta) = \sum_{i=1}^{n} \log p(x_i|\Theta)$. However, it is impossible to solve $\partial\mathcal{L}(\Theta)/\partial\Theta = 0$, without knowing which observation (x_i) belong to which component. The EM algorithm provides an appropriate solution below.

3.2.3 EM algorithm for Gaussian Mixtures

Let us now apply the EM algorithm to mixture distributions. Consider a set of random vectors $X = \{x_1, \ldots, x_n\}$ distributed independent and identically from a mixture $p(x_i|\Theta) = \sum_{s=1}^{k} \pi_s p(x_i|\theta_s)$. We define a corresponding random vector $z_i \in Z$ for each sample x_i as follows: $z_i = (z_{1i}, \ldots, z_{ki})^{\top}$, where $z_{si} = 1$ if x_i belongs to the sth component. Using this notation we can proceed to apply the EM to the mixture[1]. Therefore,

$$\mathcal{L}(\Theta) = p(X|\Theta) = \prod_{i=1}^{n} \sum_{s=1}^{k} p(x_i, z_{si}|\Theta)$$

[1]Samples x_i are the incomplete data and z_i the missing one.

$$\mathcal{F}(\Theta, \Theta^{(t)}) = \sum_i \sum_s \log(p(x_i, z_{si}|\Theta))p(Z|X, \Theta^{(t)})$$

$$H(\Theta, \Theta^{(t)}) = \sum_i \sum_s \log(p(z_{si}|X, \Theta))p(z_{si}|X, \Theta^{(t)})$$

The likelihood of $y = (x, z)$ is

$$p(y|\Theta) = p(x|z, \Theta)p(z|\Theta) = \pi_s p(x|\theta_s)$$

which may be written as,

$$p(y|\Theta) = \prod_{j=1}^{k} [\pi_j p(x|\theta_j)]^{z_j}$$

Since z_j is zero except for $j = s$. For n data points we have:

$$p(y_1, \ldots, y_n|\Theta) = \prod_{i=1}^{n} \prod_{s=1}^{k} [\pi_s p(x_i|\theta_s)]^{z_{si}}$$

with

$$\log(p(y_1, \ldots, y_n|\Theta)) = \sum_{i=1}^{n} \sum_{s=1}^{k} [z_{si} \log(\pi_s) + z_{si} \log(p(x_i|\theta_s))$$

Now, we can describe the 2 steps of the EM specified for a mixture:

E-step:

$$\begin{aligned} \mathcal{F}(\Theta, \Theta^{(t)}) &= E[\log(p(y_1, \ldots, y_n|\Theta)|X, \Theta^{(t)})] \\ &= \sum_{i=1}^{n} \sum_{s=1}^{k} E[z_{si}|x_i, \Theta^{(t)}](\log(\pi_s) + \log(p(x_i|\theta_s))) \end{aligned} \tag{3.3}$$

Note that $q_i(s) \triangleq E[z_{si}|x_i, \Theta^{(t)}]$ (responsibility) is the posterior probability of the observed data. Using the bayes rule we get:

$$q_i(s) = \frac{\pi_s^{(t)} p(x_i|\theta_s^{(t)})}{\sum_j \pi_j^{(t)} p(x_i|\theta^{(t)})} \tag{3.4}$$

M-step: Maximize $\mathcal{F}$ with respect to Θ. This yields the following updates:

$$\pi_s = \frac{\sum_{i=1}^{n} q_i}{n}, \qquad m_s = \frac{\sum_{i=1}^{n} q_i x_i}{n\pi_s}, \qquad C_s = \frac{\sum_{i=1}^{n} q_i x_i x_i^\top}{n\pi_s} - m_s m_s^\top. \tag{3.5}$$

Hence, in the E-step we assign the responsibilities with the posterior probability for an observation x_i given the current parameters, while in the M-step we maximize $\mathcal{F}$ for the q_i found in the E-step.

Notice that, each of the equations in the M-step is an average. π_s is the average of q_i, m_s is the average of products $q_i(s)x_i$ and C_s is the average of matrices $q_i(s)x_i x_i^\top$.

3.3 Fisher's Linear Discriminant

There are many examples dimensionality reduction. The most popular one is "principal component analysis (PCA)". PCA searches for directions in the data set that have the largest variance and projects the data onto it. We accomplished a "better" representation that way, if we consider that the noisy directions are gone and fewer dimensions exist [32, 63].

Fisher's liner discriminant is also looking for dimensionality reduction. It attempts to express one dependent variable as a linear combination of other features/measurements. It looks for linear combinations of these features which best explain the data. The difference between the two methods is that Fisher's linear discriminant tries to model the distinctions between classes, while PCA does not take it into account.

Consequently, if Fisher's linear discriminant is to be used, our features must have labels of what class they belong to. Suppose that C classes exist and each class has μ_i mean and Σ_i covariance, $(i = 1, \ldots, C)$. Then the "between classes scatter matrix" is defined as:

$$\Sigma_b = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

where N_i is the number of vectors in class i and μ is the mean if the class means.

The "within classes scatter matrix" is defined as:

$$\Sigma_W = \sum_{i=1}^{c} \sum_{j \in ci} (x_j - \mu_i)(x_j - \mu_i)^T$$

.

The Fisher's linear discriminant maximises the following objective:

$$S(w) = \frac{w^T \Sigma_b w}{w^T \Sigma_W w} \tag{3.6}$$

An important property to notice about the objective S is that it is invariant of the vector's rescalings $w \to \alpha w$. Hence, we can always choose w such that $w^T \Sigma_W w = 1$. For this reason we can transform the problem of maximising S into the following constrained optimisation problem,

$$\begin{aligned} min_w \quad & -1/2 w^T \Sigma_b w \\ s.t. \quad & w^T \Sigma_W w = 1 \end{aligned}$$

corresponding to the lagrangian,

$$L_P = -w^T \Sigma_b w + \lambda (w^T \Sigma_W w - 1)$$

The KKT conditions tell us that the following equation needs to hold at the solution,

$$\Sigma_b w = \lambda \Sigma_W w \Rightarrow \Sigma_W^{-1} \Sigma_b w = \lambda w$$

Using the fact that Σ_b is symmetric positive definite, we can apply the following transformation: $\Sigma_b = U \Lambda U^T \to \Sigma_b^{\frac{1}{2}} = U \Lambda^{\frac{1}{2}} U^T$. Let v be $v = \Sigma_b^{\frac{1}{2}} w$ we have,

$$\Sigma_b^{\frac{1}{2}} \Sigma_W^{-1} \Sigma_b^{\frac{1}{2}} v = \lambda v$$

The problem is a regular eigenvalue problem for a symmetric, positive definite matrix $\Sigma_b^{\frac{1}{2}} \Sigma_W^{-1} \Sigma_b^{\frac{1}{2}}$ and for which the solution λ_k and v_k corresponds to solutions $w_k = \Sigma_b^{-\frac{1}{2}} v_k$

We must now choose the correct eigenvalue and eigenvector for maximising S. We find,

$$S = \frac{w^T \Sigma_b w}{w^T \Sigma_W w} = \lambda_k \frac{w^T \Sigma_W w}{w^T \Sigma_W w} = \lambda$$

which means that we have to choose the largest eigenvalue!

3.3.1 Kernel Fisher's LDA

In many cases linear discriminants are not suitable. Fisher's LDA can be extended for use in non-linear classification via the kernel trick. We use the following key assumption,

$$w = \sum_i \alpha_i \Phi(x_i)$$

The objective S now becomes,

$$S(\alpha) = \frac{\alpha^T \Sigma_b^\Phi \alpha}{\alpha^T \Sigma_W^\Phi \alpha}$$

The scatter matrices are expressed as follows (after some algebra calculations)

$$\Sigma_b^\Phi = \sum_{i=1}^{C} k_i k_i^T - k k^T \Sigma_W^\Phi = K^2 \sum_{i=1}^{C} N_i k_i^T k_i = \frac{1}{N_i} \sum_{j \in c(i)} K_{ij} k = \frac{1}{N} \sum_i K_{ij} \tag{3.7}$$

The problem is now expressed in kernel terms, the objective has the same form as before, hence the solution has the same form. The projections of the new data-points can be found by,

$$w^T \Phi(x) = \sum_i \alpha_i K(x_i, x)$$

An extremely important issue is the regularisation, otherwise kernel machine will overfit. A common regularisation is, $\Sigma_W \rightarrow \Sigma_W + \beta I$. Now the small eigenvalues are not close to zero and the computation of the inverse matrix is possible.

3.3.2 Two class case

Let x be a set of observations that belong in 2 classes y. Suppose, that the covariance matrices $(\Sigma_{y=0}, \Sigma_{y=1})$ and the mean vectors $(\mu_{y=0}, \mu_{y=1})$ of them are known. The linear combination of these attributes $w \cdot x$ have mean $w \cdot \mu_{y=i}$ and variance $w^T \Sigma_{y=i} w$ (for $i = 1, 2$). The separation between these two clusters that fisher defined is the ration of the variance between the classes to the variance within the classes,

$$S = \frac{w \cdot (\mu_{y=0} - \mu_{y=1})^2}{w^T (\Sigma_{y=0} + \Sigma_{y=1}) w}$$

The maximum separation occurs when $w = (\Sigma_{y=1} + \Sigma_{y=0})^{-1}(\mu_{y=1} - \mu_{y=0})$

3.4 SVM

Support vector machines (SVMs) are methods used for classification and regression and are considered as supervised machine learning techniques [24, 40, 46]. They belong to specific classifiers called "Generalised liner classifiers". They are also know as maximum margin classifiers, due to the fact that they simultaneously minimise the empirical classification error and maximise the geometric margin.

Support vector machines map the input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. The separating hyperplane is the hyperplane that maximises the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalisation error of the classifier will be.

There are many occasions in which we are required to classify data that belong to only two classes. Each data point is represented by a p-dimensional vector (p numbers). We would like to know weather the data are separated with a $p - 1$-dimensional hyperplane. This is a typical form of a linear classifier and there exist many of them that satisfy this property. Apart from that, we are interested in finding out if we can achieve maximum separation (margin) between the two classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. Another way of stating it is that the nearest distance between a point in one separated hyperplane and a point in the other separated hyperplane is maximized. If such a hyperplane exists, it is clearly of interest and is known as the maximum-margin hyperplane and such a linear classifier is known as a maximum margin classifier.

3.4.1 Formal definition

Suppose we have data points:

$$(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)$$

where c_i is either 1 or -1 depending on which class x_1 belongs to. Each x_1 is a p-dimensional real vector of normilised values($[0, 1] or [-1, 1]$). Normalisation is really important to guard against features with large variances that might dominate over other ones. We can view this as training data, which denotes the correct classification which we would like the SVM to eventually distinguish, by means of the dividing (or separating) hyperplane, which takes the form

$$w \cdot x - b = 0$$

The vector w points perpendicular to the separating hyperplane. Adding the offset parameter b allows us to increase the margin. In its absence, the hyperplane is forced to pass through the origin, restricting the solution.

As we are interested in the maximum margin, we are interested in the support vectors and the parallel hyperplanes (to the optimal hyperplane) closest to these support vectors in either class. It can be shown that these parallel hyperplanes can be described by equations (by scaling w and b if not)

$$\begin{aligned} w \cdot x_i - b &\geq 1 \text{ or} \\ w \cdot x_i - b &\leq 1 \end{aligned}$$

which can be rewritten as:

$$c_i(w \cdot x_i - b) \geq 1, 1 \leq i \leq n. \tag{3.8}$$

The problem that arises now is how to minimise $|w|$ subject to the constrain 3.8. This is a quadratic programming (QP) optimisation problem.

$$\text{minimise}\frac{1}{2}|w|^2, \text{subject to } \; c_i(w \cdot x_i - b) \geq 1, 1 \leq i \leq n.$$

Writing the classification rule in its unconstrained dual form reveals that classification is only a function of the support vectors, i.e., the training data that lie on the margin. The dual of the SVM can be shown to be:

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j x_i^T x_j \text{ subject to } \alpha_i \geq 0, \text{and } \sum_{i=1}^{n} \alpha_i c_i = 0$$

where the α terms constitute a dual representation for the weight vector in terms of the training set:

$$w = \sum_i \alpha_i c_i x_i$$

3.4.2 Soft margin

Corinna Cortes and Vladimir Vapnik suggested a modified maximum margin idea that allows for mislabeled examples [23]. If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. This work popularized the expression Support Vector Machine or SVM. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum x_i

$$c_i(w \cdot x_i - b) \geq 1 - \xi_i, 1 \leq i \leq n \tag{3.9}$$

The objective function is then increased by a function which penalizes non-zero ξ_i and the optimisation becomes a trade off between a large margin and a small error penalty. If the penalty function is linear, the equation 3.9 now transforms to:

$$\min |w|^2 + C \sum_i \xi_i \text{ such that } c_i(w \cdot x_i - b) \geq 1 - \xi_i, 1 \leq i \leq n \tag{3.10}$$

This constrain in 3.9 along with the objective of minimising $|w|$ can be solved using Lagrange multipliers. The key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant C appearing only as an additional constraint on the Lagrange multipliers. Non-linear penalty functions have been used, particularly to reduce the effect of outliers on the classifier, but unless care is taken, the problem becomes non-convex, and thus it is considerably more difficult to find a global solution.

3.4.3 Kernel functions

We are able to use the same trick as in fisher's linear discriminant to create non-linear classifiers as proposed by Bernhard Boser, Isabelle Guyon and Vapnic in [36]. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space it may be non-linear in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include,

- Polynomial (homogenous): $k(x, x') = (x \cdot x')^d$
- Polynomial (inhomogenous): $k(x, x') = (x \cdot x' + 1)^d$
- Radial Basis Function: $k(x, x') = exp(-\gamma|x - x'|^2)$, for $\gamma > 0$
- Gaussian Radial Basis Function: $k(x, x') = exp\Big(-\frac{|x-x'|^2}{2\sigma^2}\Big)$
- Sigmoid: $k(x, x') = \tanh(\kappa x \cdot x' + c)$, for some (not every) $\kappa > 0$ and $c > 0$

Chapter 4

Results

We are interested in identifying intrusions from network traffic using only headers of packet flows. We solved this problem with the algorithms presented in the former chapter (GMM, Fisher's Linear Discriminant, SVM). The GMM helped us to create a model for users' normal behaviour in the network, while the Fisher's linear discriminant and the SVM uses the statistics of the 2 classes to separate them.

For our experiments we used 4 personal computers with an Intel 3.0 GHz processor and 1 GB memory (Linux and Windows operating system). We tested our results in the DARPA data-set created by MIT Lincoln Labs which is described in details below.

4.1 DARPA Data-set [47]

The 1998 DARPA intrusion detection evaluation contains over 300 attacks in 9 weeks of data traces (7 weeks for training and 2 for testing). These attacks come from 32 different attack types. All data in the 1998 DARPA Intrusion Detection System evaluation was synthesised and recorded on a network which simulated an operational network connected to the Internet. The hardware for this simulated network consists of eleven machines and one router while the software allows to model interaction of thousands of clients and servers. In table 4.1 a description of all the attacks in the 1998 DARPA intrusion detection evaluation are displayed.

Name	**Attacks Performed in the Clear**			**Stealthy Attacks**			**All Attacks**
	Training	Testing	Total	Training	Testing	Total	Total
User to Root	**67**	**24**	**91**	**18**	**5**	**23**	**144**
Eject	29	7	36	8	2	10	46
Fdformat	12	7	19	4	3	7	26
Ffbconfig	11	2	13	6	0	6	19
Perl	15	1	16	0	0	0	16
Ps	0	4	4	0	0	0	4
Xterm	0	3	3	0	0	0	3
Remote to Local User	**19**	**15**	**34**	**0**	**0**	**0**	**34**
Dictionary	3	2	5	0	0	0	5
Ftp-write	2	1	3	0	0	0	3
Guest	4	0	4	0	0	0	4
Imap	4	1	5	0	0	0	5
Named	0	3	3	0	0	0	3
Phf	6	1	7	0	0	0	7
Sendmail	0	3	3	0	0	0	3
Xlock	0	2	2	0	0	0	2
Xsnoop	0	2	2	0	0	0	2
Denial Of Service	**65**	**34**	**99**	**0**	**0**	**0**	**99**
Apache2	0	3	3	0	0	0	3
Back	4	2	6	0	0	0	6
Land	7	2	9	0	0	0	9
Mailbomb	0	1	1	0	0	0	1
Neptune	13	7	20	0	0	0	20
Process Table	0	3	3	0	0	0	3
Smurf	19	8	27	0	0	0	27
Syslog	4	2	6	0	0	0	6
Teardrop	18	4	22	0	0	0	22
UDPStorm	0	2	2	0	0	0	2
Probe/Surveillance	**50**	**14**	**64**	**0**	**0**	**0**	**64**
IPSweep	22	3	25	0	0	0	25
Mscan	0	1	1	0	0	0	1
Nmap	12	6	18	0	0	0	18
Saint	0	2	2	0	0	0	2
Satan	16	2	18	0	0	0	18
Total for All Attacks	**201**	**87**	**288**	**18**	**5**	**23**	**311**

Table 4.1: All the attacks (311 in total) from the 1998 DARPA intrusion detection evaluation

4.2 Data preparation

The DARPA data-set required some processing (data is stored in Tcpdump format). So an appropriate programme that implements lib-pcap (library packet capturing) was used, the Tcpick (Transfer control protocol connections picking). Tcpick is a programme for parsing tcp-connections and providing information the status change of each connection. Some modifications due to certain bugs and additional needs for the experiments, were made in the source code of the Tcpick. Firstly the time of the capture packet was not displayed correctly. Tcpick, showed the execution time in any case (real time packet capture or off-line packet capture). To tackle this problem a few files had to be changed (conn.h, verify.c, display.c, etc). In addition we required some extra statistics about the connections that we would apply in our models. Tcpick provided a time stamp, date, destination and source ip, destination and source port. We were interested also in connection duration, packets number and bytes transferred in a connection. Apart from that, every connections that reaches the closed status is removed from the data-base. Taking into consideration that fact and that Tcpick code is not so expandable, we stored the whole output that it provided in a file (it would require rebuilding the software from scratch if chosen otherwise). To achieve that goal we changed the display.c a little and the tracker.c file of the Tcpick source code. Finally, the Tcpick programme was designed to find Tcp (Transfer control protocol) connections in a Tcpdump file or to detect Tcp connections real-time. The DARPA data set however, contained attacks not only in Tcp packets but in UDP (User Datagram Protocol) and ICMP (Internet Control Message Protocol) packets as well. To conclude with the Tcpick, the following features were extracted:

Counter.

Timestamp.

Date.

Number of packets.

Number of bytes.

Duration.

Status of the connection.

Destination IP.

Destination Port.

Source IP.

Source Port.

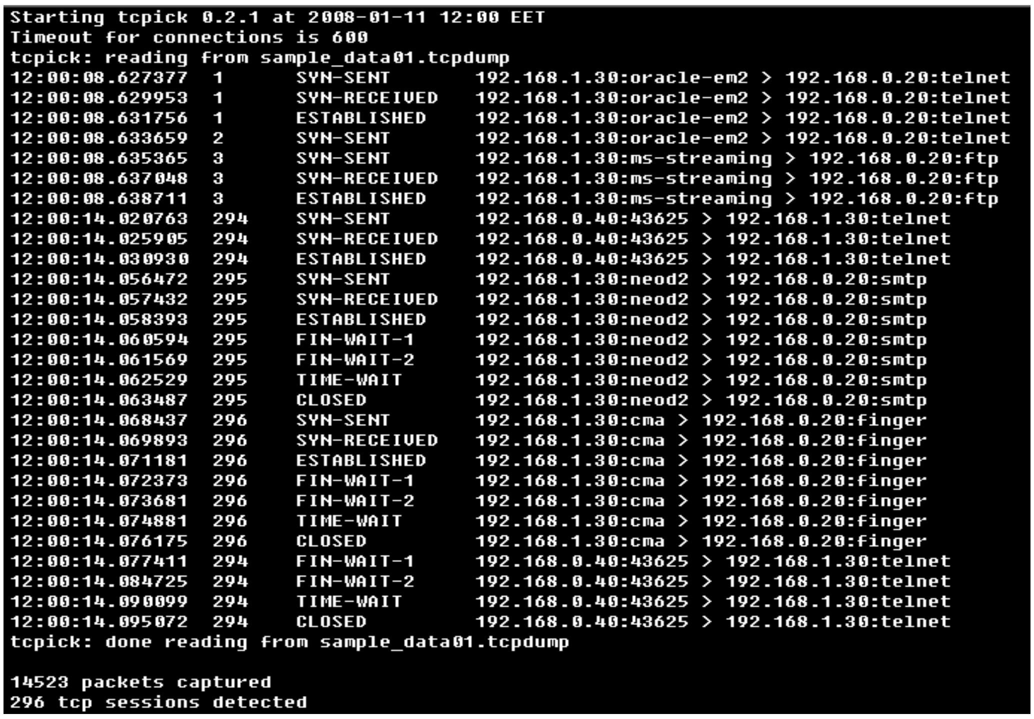
Attacks.

In figure 4.1 the output of Tcpick before the source hacking is displayed and in figure 4.2 after the hacking. The last problem we encountered from the tcpick was that it did not simulate accurately the expired connections (This problem's source is the timing again. Tcpick considers expired a connection that does not change its status for 300 seconds. These 300 seconds however are counted real time, not according to the packets timestamp. So for a connection to characterised expired, 6 minutes of processing must pass which is not the typical case). Under this constrain, we stored as output all the instances of all the connections' status change.

Clearly, we wanted to keep the last status for each connections. So the output file from the Tcpick was parsed with a perl programme that removed all instances of the same connection and kept the last one. Another perl programme was used to find which of the connections were normal and which where attack connections (The information about which connection is intrusive was provide by a file in the DARPA data-set). At this point the data is properly formed, which enables us to proceed with the feature extraction.

4.3 Experiment Preparation

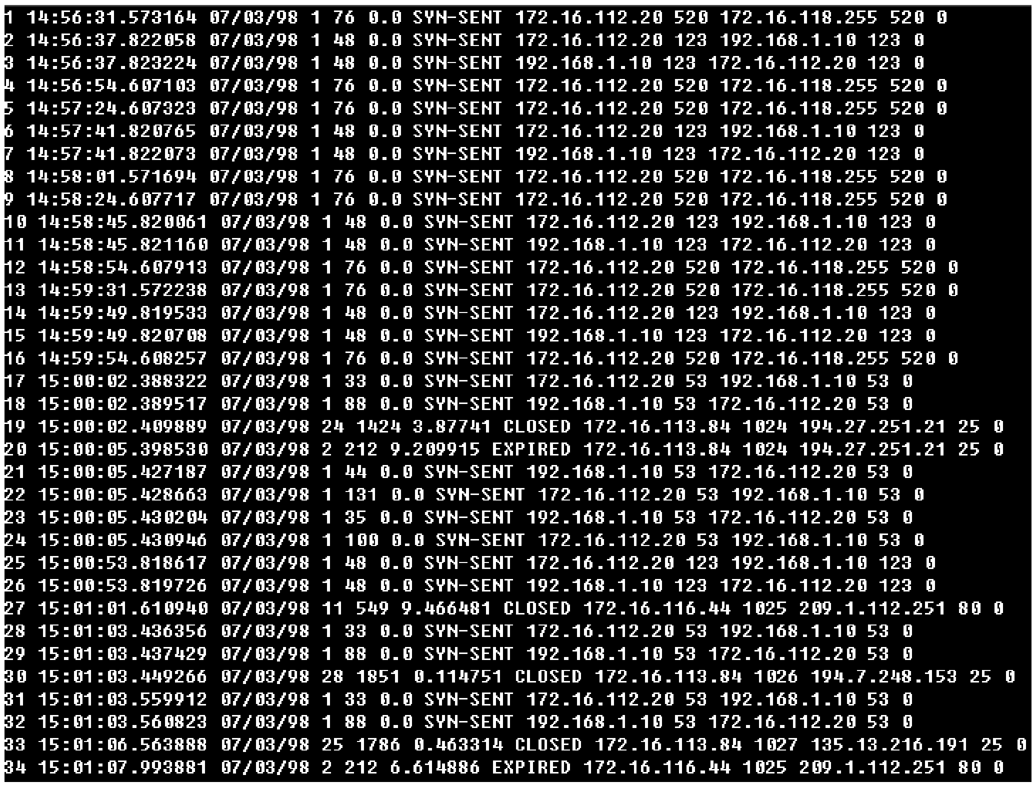
The basis of our statistical analysis is a t second non-overlapping window. The duration of the window that we used on our experiments is 10 seconds. Each timestamp of the day was mapped to seconds past from the first timestamp of the day. Afterwards, we converted the date and time into seconds and we monitored certain characteristics of the network's



```
Starting tcpick 0.2.1 at 2008-01-11 12:00 EET
Timeout for connections is 600
tcpick: reading from sample_data01.tcpdump
12:00:08.627377  1      SYN-SENT       192.168.1.30:oracle-em2 > 192.168.0.20:telnet
12:00:08.629953  1      SYN-RECEIVED   192.168.1.30:oracle-em2 > 192.168.0.20:telnet
12:00:08.631756  1      ESTABLISHED    192.168.1.30:oracle-em2 > 192.168.0.20:telnet
12:00:08.633659  2      SYN-SENT       192.168.1.30:oracle-em2 > 192.168.0.20:telnet
12:00:08.635365  3      SYN-SENT       192.168.1.30:ms-streaming > 192.168.0.20:ftp
12:00:08.637048  3      SYN-RECEIVED   192.168.1.30:ms-streaming > 192.168.0.20:ftp
12:00:08.638711  3      ESTABLISHED    192.168.1.30:ms-streaming > 192.168.0.20:ftp
12:00:14.020763  294    SYN-SENT       192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.025905  294    SYN-RECEIVED   192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.030930  294    ESTABLISHED    192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.056472  295    SYN-SENT       192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.057432  295    SYN-RECEIVED   192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.058393  295    ESTABLISHED    192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.060594  295    FIN-WAIT-1     192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.061569  295    FIN-WAIT-2     192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.062529  295    TIME-WAIT      192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.063487  295    CLOSED         192.168.1.30:neod2 > 192.168.0.20:smtp
12:00:14.068437  296    SYN-SENT       192.168.1.30:cma > 192.168.0.20:finger
12:00:14.069893  296    SYN-RECEIVED   192.168.1.30:cma > 192.168.0.20:finger
12:00:14.071181  296    ESTABLISHED    192.168.1.30:cma > 192.168.0.20:finger
12:00:14.072373  296    FIN-WAIT-1     192.168.1.30:cma > 192.168.0.20:finger
12:00:14.073681  296    FIN-WAIT-2     192.168.1.30:cma > 192.168.0.20:finger
12:00:14.074881  296    TIME-WAIT      192.168.1.30:cma > 192.168.0.20:finger
12:00:14.076175  296    CLOSED         192.168.1.30:cma > 192.168.0.20:finger
12:00:14.077411  294    FIN-WAIT-1     192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.084725  294    FIN-WAIT-2     192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.090099  294    TIME-WAIT      192.168.0.40:43625 > 192.168.1.30:telnet
12:00:14.095072  294    CLOSED         192.168.0.40:43625 > 192.168.1.30:telnet
tcpick: done reading from sample_data01.tcpdump

14523 packets captured
296 tcp sessions detected
```

Figure 4.1: Sample of tcpick parsing file *sample_data01.tcpdump* from DARPA data set.



```
1 14:56:31.573164 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
2 14:56:37.822058 07/03/98 1 48 0.0 SYN-SENT 172.16.112.20 123 192.168.1.10 123 0
3 14:56:37.823224 07/03/98 1 48 0.0 SYN-SENT 192.168.1.10 123 172.16.112.20 123 0
4 14:56:54.607103 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
5 14:57:24.607323 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
6 14:57:41.820765 07/03/98 1 48 0.0 SYN-SENT 172.16.112.20 123 192.168.1.10 123 0
7 14:57:41.822073 07/03/98 1 48 0.0 SYN-SENT 192.168.1.10 123 172.16.112.20 123 0
8 14:58:01.571694 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
9 14:58:24.607717 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
10 14:58:45.820061 07/03/98 1 48 0.0 SYN-SENT 172.16.112.20 123 192.168.1.10 123 0
11 14:58:45.821160 07/03/98 1 48 0.0 SYN-SENT 192.168.1.10 123 172.16.112.20 123 0
12 14:58:54.607913 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
13 14:59:31.572238 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
14 14:59:49.819533 07/03/98 1 48 0.0 SYN-SENT 172.16.112.20 123 192.168.1.10 123 0
15 14:59:49.820708 07/03/98 1 48 0.0 SYN-SENT 192.168.1.10 123 172.16.112.20 123 0
16 14:59:54.608257 07/03/98 1 76 0.0 SYN-SENT 172.16.112.20 520 172.16.118.255 520 0
17 15:00:02.388322 07/03/98 1 33 0.0 SYN-SENT 172.16.112.20 53 192.168.1.10 53 0
18 15:00:02.389517 07/03/98 1 88 0.0 SYN-SENT 192.168.1.10 53 172.16.112.20 53 0
19 15:00:02.409889 07/03/98 24 1424 3.87741 CLOSED 172.16.113.84 1024 194.27.251.21 25 0
20 15:00:05.398530 07/03/98 2 212 9.209915 EXPIRED 172.16.113.84 1024 194.27.251.21 25 0
21 15:00:05.427187 07/03/98 1 44 0.0 SYN-SENT 192.168.1.10 53 172.16.112.20 53 0
22 15:00:05.428663 07/03/98 1 131 0.0 SYN-SENT 172.16.112.20 53 192.168.1.10 53 0
23 15:00:05.430204 07/03/98 1 35 0.0 SYN-SENT 192.168.1.10 53 172.16.112.20 53 0
24 15:00:05.430946 07/03/98 1 100 0.0 SYN-SENT 172.16.112.20 53 192.168.1.10 53 0
25 15:00:53.818617 07/03/98 1 48 0.0 SYN-SENT 172.16.112.20 123 192.168.1.10 123 0
26 15:00:53.819726 07/03/98 1 48 0.0 SYN-SENT 192.168.1.10 123 172.16.112.20 123 0
27 15:01:01.610940 07/03/98 11 549 9.466481 CLOSED 172.16.116.44 1025 209.1.112.251 80 0
28 15:01:03.436356 07/03/98 1 33 0.0 SYN-SENT 172.16.112.20 53 192.168.1.10 53 0
29 15:01:03.437429 07/03/98 1 88 0.0 SYN-SENT 192.168.1.10 53 172.16.112.20 53 0
30 15:01:03.449266 07/03/98 28 1851 0.114751 CLOSED 172.16.113.84 1026 194.7.248.153 25 0
31 15:01:03.559912 07/03/98 1 33 0.0 SYN-SENT 172.16.112.20 53 192.168.1.10 53 0
32 15:01:03.560823 07/03/98 1 88 0.0 SYN-SENT 192.168.1.10 53 172.16.112.20 53 0
33 15:01:06.563888 07/03/98 25 1786 0.463314 CLOSED 172.16.113.84 1027 135.13.216.191 25 0
34 15:01:07.993881 07/03/98 2 212 6.614886 EXPIRED 172.16.116.44 1025 209.1.112.251 80 0
```

Figure 4.2: Sample of modified tcpick parsing a random file from the DARPA data set.

traffic. For example, how many connections exist in the specific time window. Each of the 35 days is treated the same. Therefore, we created a feature vector for each 10 seconds in 35 days. If a vector has only zero values (in other words, if the was no traffic activity in a certain 10 second instance) this vector is ignored. The final step was to train the 2 models and test them. We see below the corresponding features we used for our experiments.

Total number of connections. The total number of connections that take place in an attack and the number of connections of normal traffic.

Maximum number of a host's connections. The maximum number of connections a host creates at the 10 second window.

Total number of bytes. The total number of bytes transferred through the network during the window.

Maximum number of a host's bytes. The maximum number of bytes a host transfers in the network in the 10 second window.

Total number of packets. The total number of packets transferred through the network during the window.

Maximum number of a host's packets. The maximum number of bytes a host transfers in the network in the 10 second window.

Number of services used. The number of different services that were used for each connection in during the 10 second window.

Total number of connection differential. The differential of the total connections' number.

Maximum number of host's connection differential. The differential of a host's maximum connections.

In conclusion, we performed the tasks below:

1. We sniffed the packets from each of the tcpdump files of the data set. The packets were sniffed using a modified version of a programme named "Tcpick".

2. Next, the output file was parsed from two perl scripts so that the file would be transformed in a proper format for our experiments

3. Subsequently, we calculated through a non-overlapping time window certain properties of the network's traffic and extracted efficient separable features (as far as the 2 classes are concerned).

4. Finally, these features were used in our models for evaluation. The evaluation checked two properties. How good the models are in relation to the features and how good the features were in regard to the models.

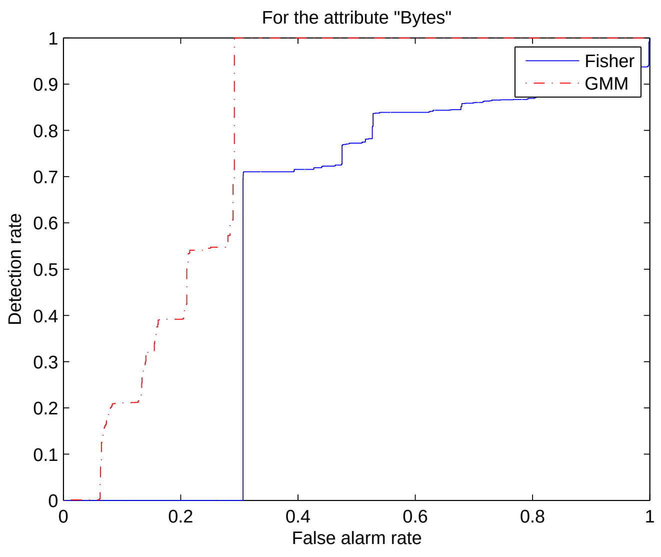


Figure 4.3: Roc curve for attribute “Number of bytes”.

4.4 Experimental work

In this section we will present our results and discuss some conclusions we made about the features. Firstly we are illustrating what detection rate we can achieve using only one attribute (see figures 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10). From the figures we can see that the differential of connections’ number does not contribute in classification by itself.

It can also be observed that the maximum packets per destination is the best feature for the discrimination of the two classes.

The fact that with only 2 features we can achieve very satisfying, was unexpected. Indeed we notice from figures 4.11 and 4.12 we achieve 80% detection rate with below 20% false alarm rate (For attributes connections, connections per ip).

For three and above characteristics the results are sufficient for detection (see figures 4.13, 4.14, 4.15, 4.16). The most significant features appear to be the Number of connections in total, the number of different services within the window, the numbers of connections, bytes and packets to a specific destination. The examination of services is performed during a probing attack, that is the reason why the service feature is important. In figure 4.7 we see the how good can one discriminate between classes using only the service attribute. The service attribute has many steps and that’s because usually many services are scanned when a probing attack takes place. In other words the models

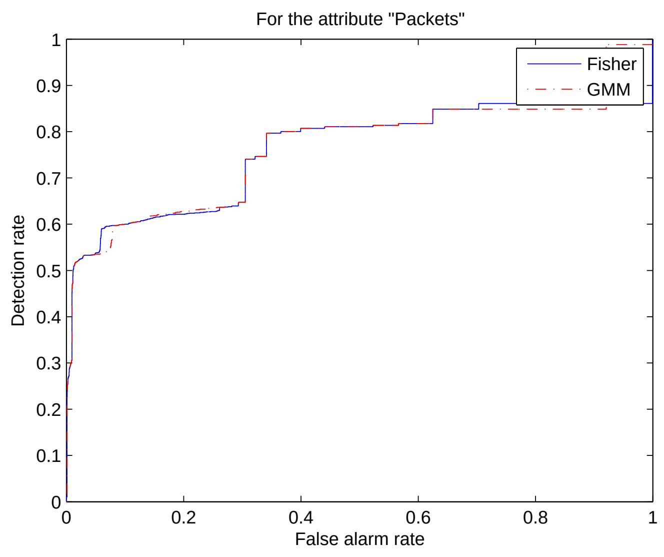


Figure 4.4: Roc curve for attribute “Number of packets”.

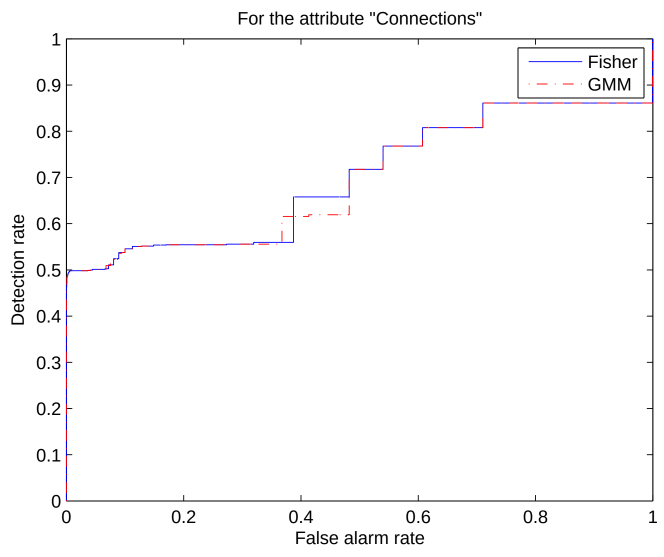


Figure 4.5: Roc curve for attribute “Number of connections”.

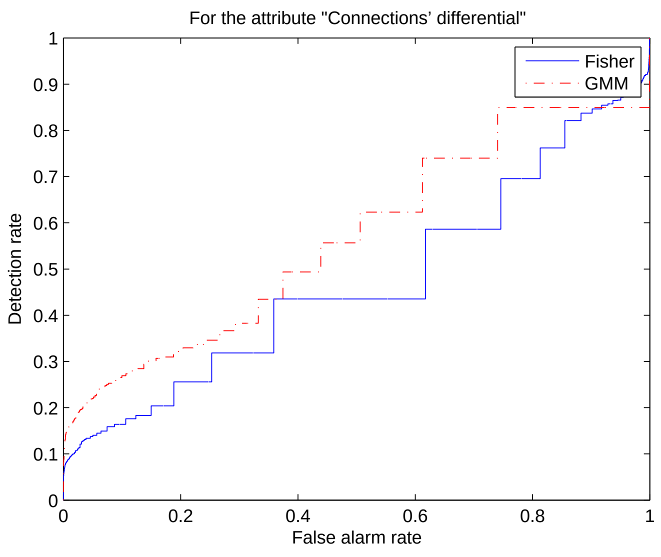


Figure 4.6: Roc curve for attribute “Difference in connections in 2 sequential time windows”.

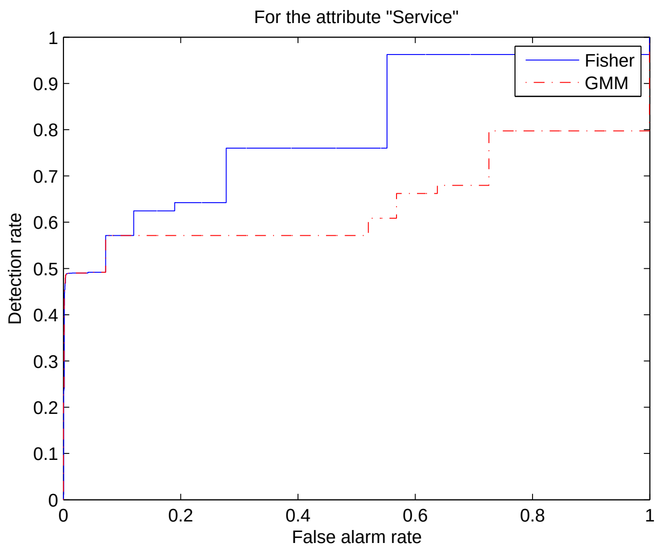


Figure 4.7: Roc curve for attribute “Number of different services used within a time window”.

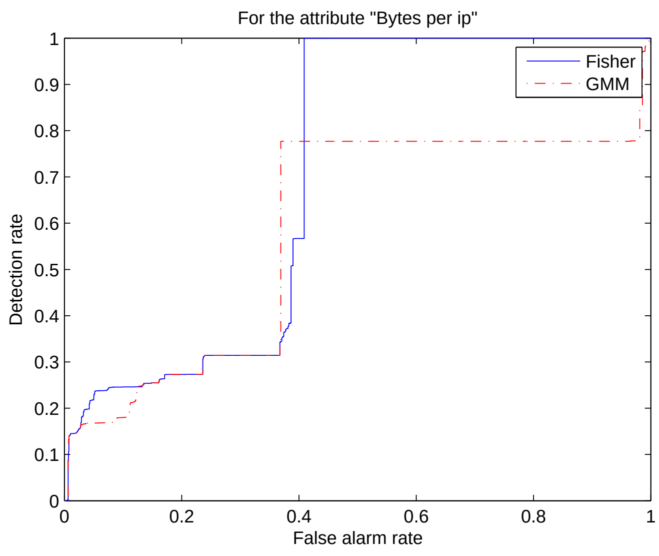


Figure 4.8: Roc curve for attribute “Maximum number of bytes to one destination”.

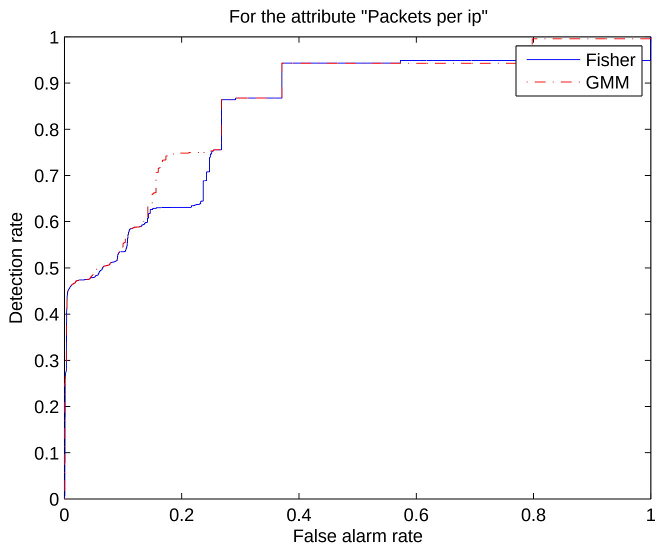


Figure 4.9: Roc curve for attribute “Maximum number of packets to one destination”.

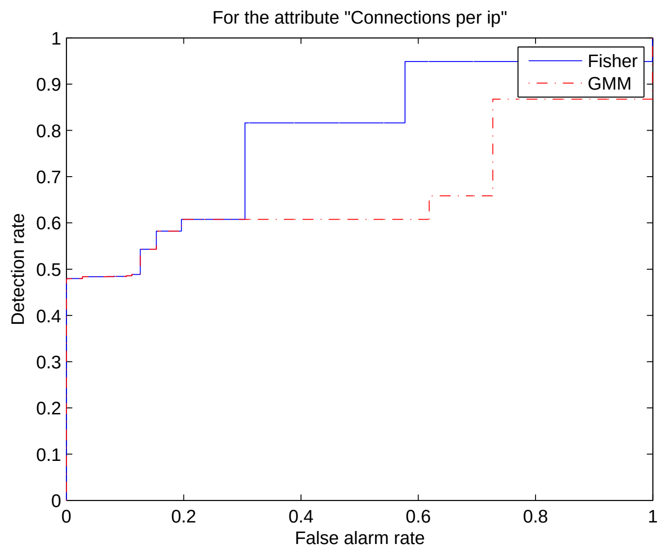


Figure 4.10: Roc curve for attribute “Maximum number of connections to one destination”.

consider normal traffic when they check a DoS attack and attack traffic when they check a probing attack. The connection feature applies well for half of the connections since in some instances (when there is not that much traffic involved they are really distinguishable from both the DoS and the Probing attacks). The last three attributes (Packets, bytes and connections per destination) are really sufficient because all the attacks are performed against one destination.

The best ROC curve is shown in figure 4.14. The ROC curve of figure 4.13 is similar to the best one. Maximum number of bytes to a destination and maximum number of packets to destination are correlated attributes that is why the results of the two figure above are so close.

To conclude with the results in table 4.2 below, the best results from the 3 models are displayed:

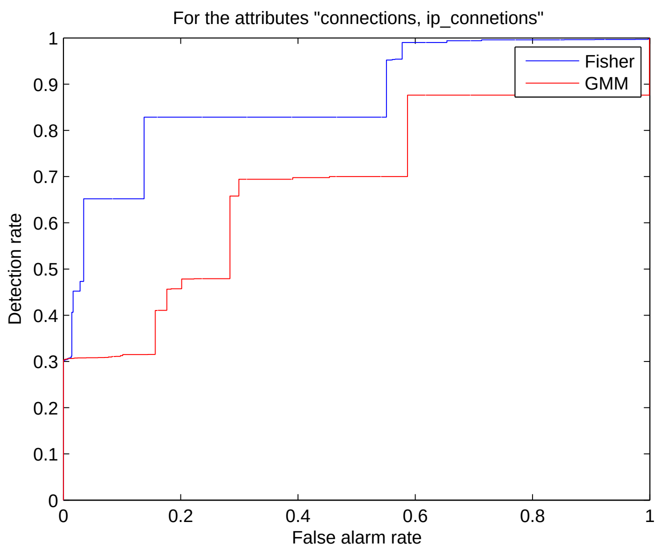


Figure 4.11: Roc curve for attributes “Number of connections, maximum number of connections to a destination”.

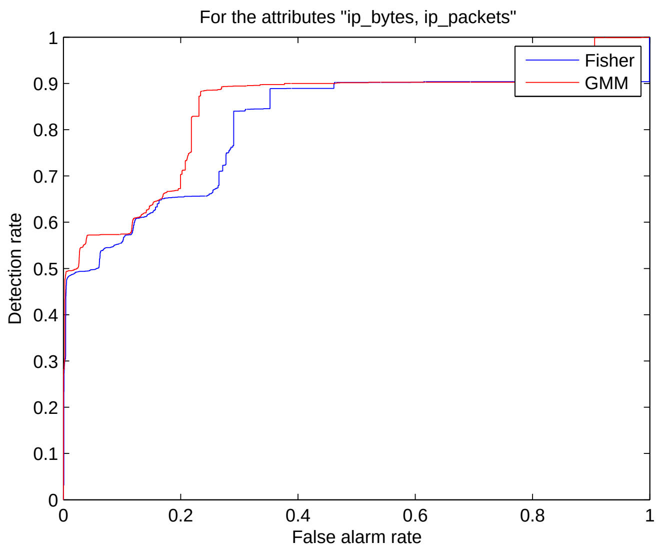


Figure 4.12: Roc curve for attributes “Maximum number of bytes, packets to a destination”.

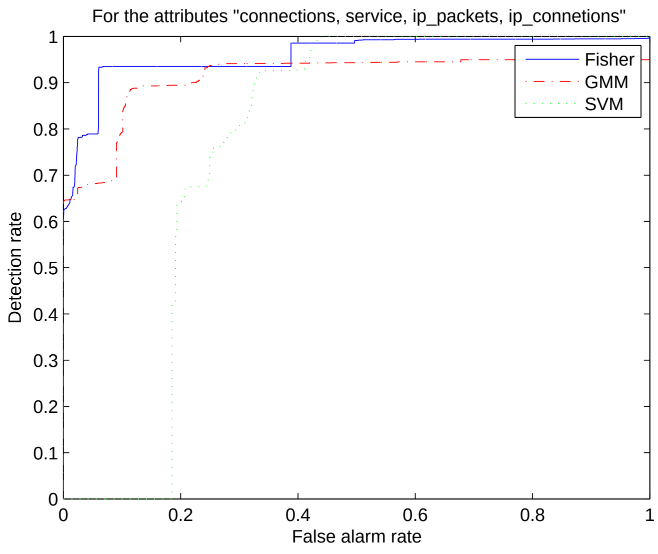


Figure 4.13: Roc curve for attributes "Connections, number of different services, maximum number of packets and connections to a destination".

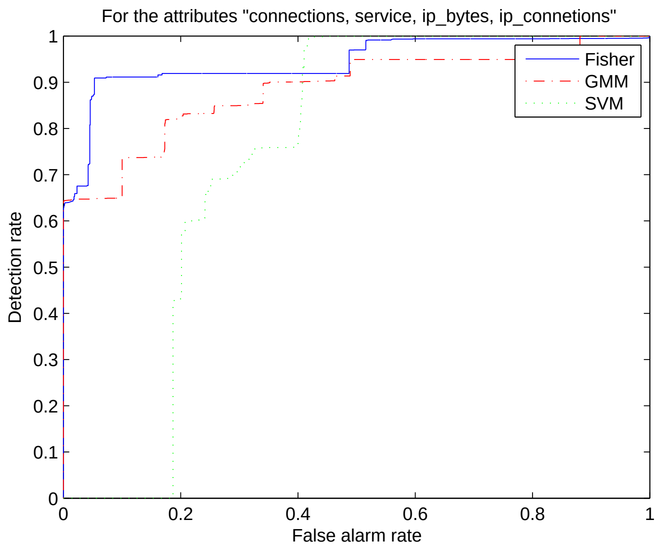


Figure 4.14: Roc curve for attributes "Connections, number of different services, maximum number of bytes and connection to a destination".

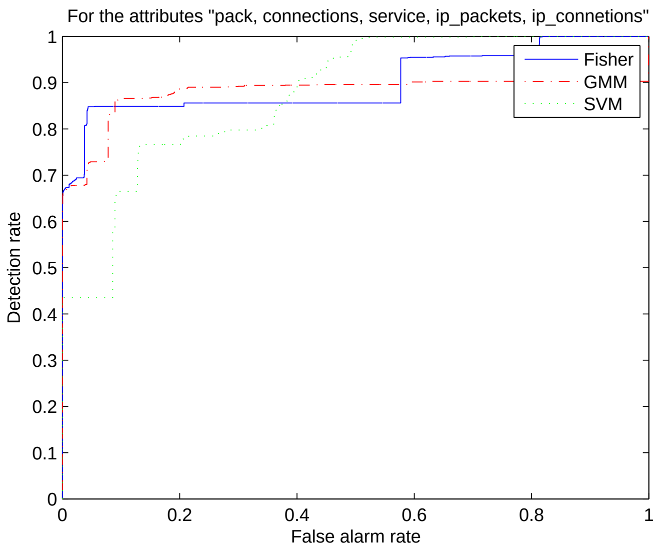


Figure 4.15: Roc curve for attributes “Packets, connections, number of different services, maximum number of packets, connection to a destination”.

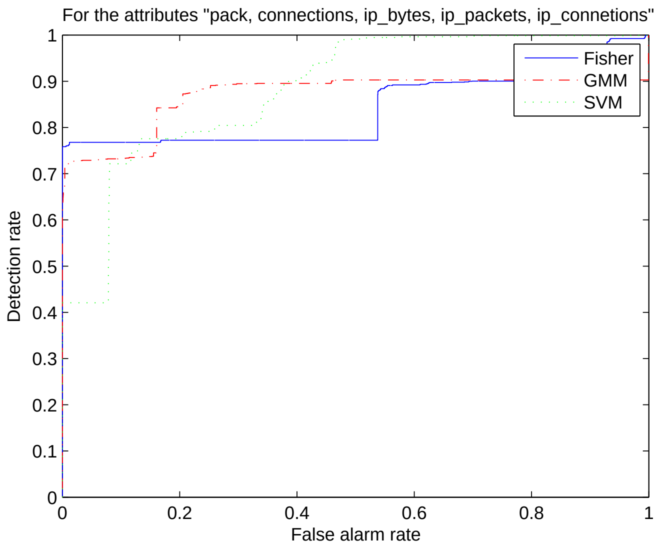


Figure 4.16: Roc curve for attributes “Packets, connections, maximum number of bytes, packets and connections to a destination”.

Model	**Detection Rate**	**False Alarm**
Fisher	70%	0%
Fisher	80%	2%
Fisher	90%	5.3%
Fisher	91%	6%
Fisher	93%	7%
Fisher	94%	9%
GMM	70%	0.4%
GMM	80%	7%
GMM	84%	8%
GMM	86%	10%
SVM	50%	2%
SVM	60%	4%
SVM	62%	6%
SVM	65%	7%
SVM	69%	8%
SVM	72%	10%

Table 4.2: The best results from each of the three statistical models (Fisher's Linear Discriminant, Gaussian Mixture Model, Support Vector Machines) are shown

Chapter 5

Conclusion/Future Work

In this thesis we studied Intrusion Detections Systems and the problem of security in a network. We proposed an Intrusion Detection system that depends on as few as possible parameters (for example window length) and examined its behaviour and the detection that it can achieve. Using the correct features of the network's traffic one can detect many types of attacks, with low false alarm rate. Even though it looks like the field is almost fully defined, new attacks arise daily, with different characteristics, which require to be alternatively encountered. Our results are very promising and they show that it is a proper system for at least additional measures against compromisations.

The most effective IDSs presently are the ones that are based on misuse (for commercial use). These IDSs however cannot prevent a new attack from compromising a host. It would be valuable if we could build a reliable anomaly detection IDS. Our IDS has the capability of becoming such an IDS.

For the future we could use new features and apply our models on other more resent test sets for the detection. The new characteristics may help us separate more accurately the two classes (normal and abnormal traffic), while the new models might provide better boundaries. Apart from DDoS (Distributed DoS) and probing attacks we would like to observe internet worms' behaviour and compromisation. Internet worms constitute a serious threat to modern network. Among other things, they can easily perform a distributed denial of Service.

Intrusion detection does not stop in wired networks on the other hand. Wireless networks are the future of network industry and a field of significant needs in research. Clearly intrusion detection in wireless networks is still in elementary level and it would be mandatory to extend it to more sophisticated detection systems.

Our study showed that the most important aspect of statistical intrusion detection is feature selection. Choosing the right two features for example we achieve using a very simple statistical tool (Fisher's Linear Discriminant) 80% detection rate with 15% false alarm rate. In addition modeling both normal traffic and intrusive traffic yields worst results (The Gaussian Mixture Model performed worse than Fisher's Linear Discriminant). The best features that arise from our experiments are the following 4: Connections, services, packets per destination and connections per destination. We found that the best model in our evaluation was the Fisher's Linear Discriminant and yielded 90% with 5% false alarm.

In conclusion, we showed that a statistical intrusion detection system may provide satisfactory results if the right feature/attributes are selected.

Bibliography

[1] TCPDUMP public repository. http://www.tcpdump.org.

[2] CERT® Advisory CA-1995-13 Syslog Vulnerability - A Workaround for Sendmail, September 1997. http://www.cert.org/advisories/CA-1995-13.html.

[3] Internet Security Systems, Inc., RealSecure, 1997. http://www.iss.net/prod/rsds.html.

[4] CERT® Advisory CA-1999-04 Melissa Worm and Macro Virus, March 1999. http://www.cert.org/advisories/CA-1999-04.html.

[5] The third international knowledge discovery and data mining tools competition dataset KDD99-CUP, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[6] CERT® Advisory CA-2000-14 Microsoft Outlook and Outlook Express Cache Bypass Vulnerability, July 2000. http://www.cert.org/advisories/CA-2000-14.html.

[7] CERT® Advisory CA-2001-26 Nimda Worm, September 2001. http://www.cert.org/advisories/CA-2001-26.html.

[8] CERT® Advisory CA-2003-04 MS-SQL Server Worm, 2003. http://www.cert.org/advisories/CA-2003-04.html.

[9] CERT® Advisory CA-2003-25 Buffer Overflow in Sendmail, September 2003. http://www.cert.org/advisories/CA-2003-25.html.

[10] Pcap, libpcap, winpcap, libdnet, and libnet Applications and Resources, 2004. http://www.stearns.org/doc/pcap-apps.html.

[11] SNORT Intrusion Detection System, 2004. www.snort.org.

[12] A. AirDefense, 2004. http://www.airdefense.net/products/index.html.

[13] S.T. Kent A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E Jones, F. Tchakountio and W.T. Strayer. Hash-Based IP Traceback. *In Proceedings of the ACM SIGCOMM Conference*, August.

[14] Elias Athanasopoulos, Kostas G. Anagnostakis and Evangelos P. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network that Never Forgets. *In Proceedings of the 4th International Conference on Applied Cryptography and Network Security (ACNS'06)*, June 2006.

[15] S. Antonatos, M. Polychronakis, P. Akritidis, Kostas D. Anagnostakis, and Evangelos P. Markatos. Piranha: Fast and Memory-efficient Pattern Matching for Intrusion Detection. *In Proceedings of the 20th IFIP International Information Security Conference (IFIP/SEC2005)*, May 2005.

[16] K. G. Anagnostakis, E. P. Markatos, S. Antonatos, and M. Polychronakis. E2xB: A domain-specific string matching algorithm for intrusion detection. *In Proceedings of the 18th IFIP International Information Security Conference (SEC2003)*, May 2003.

[17] D.S. Bauer and M.E. Koblentz. NIDX - An Expert System For Real-Time. *Computer Networking Symposium*, 1988.

[18] T. Baving. Network vs. Application-Based Intrusion Detection. *Natwork and Internet Network Security, Computer Science Honours*, 2003.

[19] H. Debar, M. Becker and D. Siboni. A Neural Network Component for an Intrusion Detection System. *In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 240–250, 1992.

[20] J. Jung, V. Paxson, A. W. Berger and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. *In Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.

[21] E. Zwicky, S. Cooper, D. Chapman and D. Ru. Building Internet Firewalls. *In Proceedings of the IEEE Infocom.*

[22] cknow.com. Virus Tutorial, 2001. http://www.cknow.com/vtutor/vtmap.htm.

[23] Corinna Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning, 20*, 1995.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* 2000.

[25] H. Debar, M. Dacier and A. Wespi. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, 31, 8:805–822, September 1999.

[26] V. Dao and R. Vemuri. Computer Network Intrusion Detection: A Comparison of Neural Networks Methods, Differential Equations and Dynamical Systems. *Special Issue on Neural Networks*, 2002.

[27] D. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13, 2:222–232, September 1987.

[28] System Detection. Anomaly Detection: The Antura Difference, 2003. http://www.sysd.com/library/anomaly.pdf, 2003.

[29] C. Dowell and P. Ramstedt. The Computerwatch Data Reduction Tool. *In Proceedings of the 13th National Computer Security Conference*, 1990.

[30] S. Eschrich. *Real-Time User Identification Employing Standard Unix Accounting.* PhD thesis, Florida State University, Fall 1995. PhD Thesis.

[31] L. Portnoy, E. Eskin, and S. J. Stolfo. Intrusion detection with unlabeled data using clustering. *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, 2001.

[32] R.A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics, 7*, pages 179–188, 1936.

[33] T.M. Gil and M. Poletto. MULTOPS: A Data-Structure for Bandwidth Attack Detection. *In Proceedings of the USENIX Security Symposium*, July.

[34] C. Cowan, C. Pu, D. Maier, H. Hinton, J. Walpole, P. Bakke, S. Beattie, A. Grier and P. Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer Overflow Attacks. *In Proceedings of the 7th USENIX Security Symposium*, pages 63–77.

[35] Internet Guide. Computer Viruses / Virus Guide, 2002. http://www.internet-guide.co.uk/viruses.html.

[36] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *ACM Press*, 7, 1:144–152, 1992.

[37] S.E. Hansen and E.T. Atkins. Automated System Monitoring and Notification With Swatch. *In Proceedings of the Seventh Systems Administration Conference (LISA'93)*, November 1993.

[38] S. Staniford, J. Hoagland and J. McAlerney. Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security*, 10, 1-2:105–136, 2002.

[39] D. Hughes. TkLogger. ftp://coast.cs.purdue.edu/pub/tools/unix/tklogger.tar.Z.

[40] Huang T.-M., Kecman V., Kopriva I. Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning. *Springer-Verlag*, pages 37–71, 2006.

[41] Cisco Systems, Inc. NetRanger-Enterprise-scale, Real-time, Network Intrusion Detection System, 1998. http://www.cisco.com/univercd/cc/td/doc/product/iaabu/netrangr/.

[42] J. Ioannidis and S. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. *In Proceedings of the Network and Distributed System Security Symposium*, February 2002.

[43] M-J. Lin J. Ryan and R. Miikkulainen. Intrusion Detection with Neural Networks. *In Proceedings of the AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 72–77, July 1997.

[44] H.S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. *In Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1991.

[45] S. Savage, D. Wetherall, A. Karlin and T. Anderson. Practical Network Support for IP Traceback. *In Proceedings of the IEEE ACM SIGCOMM Conference*, August.

[46] Vojislav Kecman. Learning and Soft Computing - Support Vector Machines, Neural Networks, Fuzzy Logic Systems. *The MIT Press, Cambridge, MA*, 2001.

[47] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, June 1999.

[48] Aleksandar Lazarevic, Vipin Kumar and Jaideep Srivastava. *Intusion Detection: A survey*, volume 5, chapter 2. Springer US, 2006.

[49] L. Ertoz, E. Eilertson, P. Dokas, V. Kumar and K. Long. Scan Detection - Revisited. *Army High Performance Computing Research Center Technical Report*, 2004.

[50] C. Cheng, H.T. Kung, and K. Tan. Use of Spectral Analysis in Defense Against DoS Attacks. *In Proceedings of the IEEE GLOBECOM*, 2002.

[51] J. Srivastava, V. Kumar L Ertoz, E. Eilertson, A. Lazarevic, P. Tan and P. Dokas. The MINDS - Minnesota Intrusion Detection System. *in Data Mining: Next Generation Challenges and Future Directions*, 2004.

[52] T. Lane and C. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. *ACM Transactions on Information and System Security*, 2,3:295–331, 1999.

[53] H. Feng, O. Kolesnikov, P. Fogla, W. Lee and W. Gong. Anomaly Detection Using Call Stack Information. *In Proceedings of the IEEE Symposium Security and Privacy*, May 2003.

[54] W. Lee and S.J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3,4:227–261, 2000.

[55] U. Lindqvist and P.A. Porras. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). *In Proceedings of the IEEE Symposium on Security and Privacy*, May 1999.

[56] J. Lo. Trojan Horse Attacks, April 2004. www.irchelp.org/irchelp/security/trojan.html.

[57] K. Houle, G. Weaver, N. Long and R. Thomas. Trends in Denial of Service Attack Technology, October 2001. CERT® Coordination Center, Pittsburgh.

[58] T. Lunt. Real-Time Intrusion Detection. *In Proceedings of the the Thirty Fourth IEEE Computer Society International Conference (COMPCON), Intellectual Leverage*, February 1989.

[59] R. Mahajan, and P. Cahn. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. *In Proceedings of the Eight ACM International Conference on Knowledge Discovery and Data Mining*, July.

[60] M. Mahoney and P. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. *In Proceedings of the Eight ACM International Conference on Knowledge Discovery and Data Mining*, pages 376–385, July 2002.

[61] D. Marchette. Computer intrusion detection and network monitoring, a statistical viewpoint. *Springer*, 2001.

[62] Robert Hogg, Joseph McKean and Allen Craig. Introduction to Mathematical Statistics. pages 359–364.

[63] G.J. McLachlan. Discriminant Analysis and Statistical Pattern Recognition. *Wiley-Interscience*, August 2004.

[64] Metropolitan. Metropolitan Network BBS, Inc., Kaspersky.ch, Computer Virus Classification, 2003. http://www.avp.ch/avpve/classes/classes.stm.

[65] S. Robertson, E. Siegel, M. Miller and S. Stolfo. Surveillance Detection in High Bandwidth Environments. *In Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX 2003)*, April 2004.

[66] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attacks and Defense Mechanisms. *ACM Computer Communication Review*, April 2004.

[67] P. Neumann and P. Porras. Experience with Emerald to Date. *In Proceedings of the First Usenix Workshop on Intrusion Detection and Network Monitoring*, 1999.

[68] K.P. Park and H. Lee. On the effectiveness of router-based Packet Filtering for Distributed Dos Attack Prevention. *In Proceedings of the ACM SIGCOMM Conference*, November 2002.

[69] P. Akritidis, Evangelos P. Markatos, M. Polychronakis, and Kostas D. Anagnostakis. STRIDE: Polymorphic Sled Detection through Instruction Sequence Analysis. *In Proceedings of the 20th IFIP International Information Security Conference (IFIP/SEC 2005)*, May 2005.

[70] P.A. Porras and A. Valdes. Live Traffic Analysis of TCP/IP Gateways. *In Proceedings of the ISOC Symposium on Network and Distributed System Security (NDSS'98)*, March 1998.

[71] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *In Data Mining for Security Applications*, 2002.

[72] J. Mirkovic, G. Prier and P. Reiher. Attacking ddos at the source. 10^{th} *IEEE International Conference on Network Protocols*, November 2002.

[73] Internet Security Systems Wireless Products. Active Wireless Protection, An XForce's white paper, September 2002. documents.iss.net/whitepapers/ActiveWirelessProtection.pdf.

[74] Mazu Profiler^{TM}, December 2002. An Overview, http://www.mazunetworks.com/solutions/white_papers/download/Mazu_Profiler.pdf.

[75] F. Provost and T. Fawcett. Robust Classification for Imprecise Environments. *Machine Learning*, 42, 3:203–231, 2001.

[76] J. Marin, D. Ragsdale and J. Surdu. A Hybrid Approach to Profile Creation and Intrusion Detection. *In Proceedings of the DARPA Information Survivability Conference and Exposition*, June 2001.

[77] J. B. D. Cabrera, B. Ravichandran and R. K. Mehra. Statistical Traffic Modeling for Network Intrusion Detection. *In Proceedings of the Eighth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 466–473, August 2000. IEEE Computer Society.

[78] Arthur Demster, Nan Laird, Donald Rubin. *Maximum likelihood from incomplete data via the EM algorithm.* 1977.

[79] M. Esmaili, B. Balachandran, R. Safavi-Naini and J. Pieprzyk. Case-Based Reasoning For Intrusion Detection. *In Proceedings of the 12th Annual Computer Security Applications Conference*, December 1996.

[80] M. Esmaili, R. Safavi-Naini and B.M. Balachandran. Autoguard: A Continuous Case-Based Intrusion Detection. *In Proceedings of the Australian Computer Science Conference, Australian Computer Science Communications*, pages 392–401.

[81] K. Sequeira and M. Zaki. ADMIT: Anomaly-base Data Mining for Intrusions. *In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.

[82] E. Spafford and D. Zamboni. Intrusion Detection Using Autonomous Agents. *Computer Networks*, 34:547–570, 2000.

[83] N. Weaver, V. Paxson, S. Staniford and R. Cunningham. A Taxonomy of Computer Worms. *In Proceedings of the The Workshop on Rapid Malcode (WORM 2003), held in conjunction with the 10th ACM Conference on Computer and Communications Security*, October 27 2003.

[84] W. Lee, S. Stolfo and K. Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14:533–567, 2001.

[85] R. Stone. Centertrack:An IP Overlay Network for Tracking DoS Floods. *In Proceedings of the USENIX Security Symposium*, July.

[86] B. Tod. Distributed Denial of Service Attacks. *OVEN Digital*, 2002. http://www.linuxsecurity.com/resource_files/intrusion_detection/ddos-faq.html.

[87] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P.G. Neumann, H.S. Javitz, A. Valdes and T.D. Garvey. A Real Time Intrusion Detection Expert System (IDES), sri technical report, 1992.

[88] G. Vigna and R.A. Kemmerer. Netstat: A Network-Based Intrusion Detection Approach. *Journal of Computer Security*, 7, 1:37–71, 1999.

[89] D. Moore G. M. Voeker and S. Savage. *Inferring Internet Denial-of-Service Activity*, pages 9–22. August 2001.

[90] D. Winer. Clay Shirky on P2P. November 2000. davenet.scripting.com/2000/11/15/clayShirkyOnP2p.

[91] D. Barbara, N. Wu and S. Jajodia. Detecting Novel Network Intrusions Using Bayes Estimators. *In Proceedings of the First SIAM Conference on Data Mining*, April 2001.

[92] J. Levine Y.X. Lim, T. Schmoyer and H.L. Owen, Wireless Intrusion Detection and Response. *In Proceedings of the IEEE Workshop on Information Assurance*, June 2003.

[93] H. Wang, D. Zhang, and K. Shin. Detecting SYN Flooding Attacks. *In Proceedings of the IEEE Infocom*, June.