İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

INTRUSION DETECTION WITH PATTERN CLASSIFICATION

M.Sc. Thesis by
Müge ÇEVİK

Department : Computer Engineering

Programme: Computer Engineering

JANUARY 2005

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

INTRUSION DETECTION WITH PATTERN CLASSIFICATION

M.Sc. Thesis by
Müge ÇEVİK
504011404

Date of submission : 27 December 2004

Date of defence examination: 31 December 2004

Supervisor (Chairman): Prof. Dr. Bülent ÖRENCİK

Members of the Examining Committee Prof.Dr. Bilge GÜNSEL

Assoc. Prof.Dr. Coşkun Sönmez

JANUARY 2005

ACKNOWLEDGEMENTS

First of all, I am deeply appreciated to Prof.Dr. Bülent ÖRENCİK for his supervising and his kind tolerance to me. His support and encouragement made me write this thesis.

I am also appreciated to Prof.Dr. Bilge GÜNSEL, for her patience and guidance to my questions about Pattern Classification.

I dedicate this thesis to my father and mother who supported me in every phase of my educational life, and all my teachers who thought me analysing, researching, and determined working. Without them everything would diffucult for me.

December 2004 Müge ÇEVİK

TABLE OF CONTENTS

ABBREVIATIONS

ACK :Acknowledgement
ACM :Association for Computer Machinery
ADAM :Audit Data Analysis and Mining
ATM :Asynchronous Transfer Mode
CLIDS :Cluster Based Intrusion Detection System
DARPA :The Defense Advanced Research Projects Agency
DOS :Denial of Service
DNA :Deoxyribase Nucleic Acid
DNS :Domain Name Service
EQ :Equation
EX :Example
FCBF :Fast Correlation Based Filter
FDDI :Fiber Distributed Data Interface
FIN :Finish
FTP :File Transfer Protocol
HTTP :Hypertext Transfer Protocol
IBL :Instance Based Learning
ICMP :Internet Control Message Protocol
ID :Intrusion Detection
IDDM :Intrusion Detection using Data Mining
IDES :Intrusion Detection Expert System
IDIOT :Intrusion Detection in Our Time
IDS :Intrusion Detection System
IP :Internet Protocol
IPSEC :Internet Protocol Security
ISDN :Integrated Services Digital Network
JAM :Java Agents for Metalearning
KDD :Knowledge Discovery and Data Mining
KNN :K-Nearest Neighbour
LAN :Local Area Network
MADAM ID :Mining Audit Data for Automated Models for Intrusion Detection
MAX :Maximum
MIN :Minimum
MINDS :Minnesota Intrusion Detection System
MIT :Massachusetts Institute of Technology
NETSTAT :Network-based State Transition Analysis Tool
NIDES :Next Generation Intrusion Detection Expert System
NIDX :Network Intrusion Detection Expert System
NNID :Neural Network Intrusion Detector
NP :Nondeterministic Polynamial
OSPF :Open Shortest Path First
POP :Post Office Protocol

PSH	:Push
R2L	:Unauthorized Access From a Remote Machine
RFC	:Request for Comment
RST	:Reset
SLIP	:Serial Line Internet Protocol
SMTP	:Simple Mail Transport Protocol
SNMP	:Simple Network Management Protocol
SVM	:Support Vector Machine
SYN	:Synchronize
TCP	:Transmission Control Protocol
U2R	:Unauthorized Access to Local Superuser
UDP	:User Datagram Protocol
URG	:Urgent
U.S	:United States
USTAT	:Unix State Transition Analysis Tool

LIST OF TABLES

Page Number

LIST OF FIGURES

Page Number

ÖRÜNTÜ SINIFLANDIRMASI İLE SALDIRI TESPİTİ

ÖZET

Bilgisayarların ve bilgisayar ağlarının hızlanması, bilgisayar kullananların ve internete ulaşabilenlerin sayısı artması teknolojik gelişmenin göstergeleridir. Ne yazık ki herkes teknolojiyi iyi amaçlar doğrultusunda kullanmamaktadır, bazı kişiler kendilerinin ya da başkalarının çıkarlarına hizmet etmek için teknolojinin açıklarını bulmaya çalışmaktadırlar.

Bilgisayar saldırıları günümüzde çok popüler bir araştırma konusudur ve olmaya da devam edecektir. Çünkü her yeni saldırıya karşı bir önlem bulundukça, saldırganlar da yeni saldırılar yaratmaktadırlar. Bugün bir çok büyük ya da küçük şirket, kamu kuruluşu ya da organizasyon saldırıya maruz kalmaktadır ve bu organizasyonlar prestijlerinin kaybetmemek için bu saldırıların çok azını kamuya açıklamaktadırlar.

Saldırı tespit sistemleri, 1980'lerden beri geliştirilmektedirler. Temel olarak iki tip saldırı tespit sistemi vardır: Davranış bazlı ve bilgi bazlı. Bilgi bazlı sistemler sadece bildikleri saldırıları yakalayabilirler. Yeni saldırılara karşı dayanaksızdırlar. Davranış bazlı saldırı tespit sistemleri ise normal davranışları öğrenirler ve bu davranıştan farklı olan davranışları anormal olarak tanımlarlar. Her iki tip yakalama yönteminde uzman sistemler, veri madenciliği gibi belli algoritmalar kullanılmış ve bir çok birbirine alternatif saldırı tespit sistemi geliştirilmiştir.

Örüntü sınıflandırması son yıllarda saldırı tespitinde kullanılmaya başlamıştır. Örüntü sınıflandırması çok uzun yıllardan beri biyoloji, görüntü tanıma gibi bir çok alan kullanılmış ve bu konuda bir çok algoritma geliştirilmiştir. Örüntü sınıflandırması hem bilgi bazlı hem davranış bazlı saldırı tespitini biraraya getirerek optimum sonuca ulaşmada yol gösterici olmaktadır.

Örüntü sınıflandırmada iki tür yöntem vardır: Öğretimli sınıflandırma, öğretimsiz sınıflandırma. Öğretimli sınıflandırmada belli bir örüntü kümesiyle algoritma çalıştırılır ve algoritma bu kümede önceden belirlenmiş sınıfları ve sınıfların özelliklerini öğrenir. Test örüntüsü algoritmaya verildiğinde bu test verisinin hangi sınıftan olduğu belirlenir. Öğretimsiz sınıflandırmada ise örüntülerin hangi sınıfta oldukları önceden bilinmez. Örüntülerden birbirine yakın özellikte olanlar aynı sınıfa toplanır. Daha sonra bunlar etiketlenir. Tüm algoritmalarda özelliklerin neler olduğu, hangi özelliklerin seçileceği, hangi özellliğe ne ağırlık atanacağı gibi bilgiler sonuca doğrudan etki eder.

ACM Special Interest Group on Knowledge Discovery and Data Mining tarafından her yıl yapılan veri madenciliği yarışmasında 1999 yılında saldırı tespit verileri kullanılmış ve bu veriler bir çok saldırı tespit sisteminin gelişmesinde rol oynamıştır.

Bu tezde de bu veriler kullanılarak bir örüntü sınıflandırması ile saldırı tespit sistemi gerçeklenmeye çalışılmıştır. Bu saldırı tespit sistemi CLIDS (Cluster based Intrusion Detection System) olarak adlandırılmıştır. Bu sistemde öncelikle bilinen ataklarla eğitim verileri içinde sınıf karakterisitikleri çıkarılmaktadır. Bunu yaparken de bilinen sınıflandırma algoritmaları doğrudan kullanılmamıştır. Eğitim verileri, Lei Yu ve Huan Liu tarafından geliştirilmiş ve sonuçları kanıtlanmış FCBF algoritmasıyla ayırt

edici özellikleri bulunduktan sonra sınıflandırılmıştır. Bu sınıflandırmada saldırı tespitinde çok önemli rol oynayan sembolik veriler (protokol tipi, hizmet tipi, bayrak tipi vb) öne çıkarılarak, FCBF tarafından seçilmiş sembolik verilerle etiketlenen sınıflar oluşturulmuştur. Bundan sonra CLIDS 'in içindeki algoritma test örüntülerini, "normal – atak " verilerinin oluşturduğu sınıflara göre yaptığı karşılaştırmalarla hangi sınıflara yakın olduğunu, eğer birden fazla sınıfa yakın bulduysa bunlardan hangisinin seçilmesi gerektiğini bulur ya da hiç bir sınıfa önceden belirlenmiş bir eşikten daha yakın değise "anormal" olarak etiketler.

CLIDS gerçek zamanlı çalışmamakla birlikte öğretimli örüntü sınıflandırmasının ve özellik seçiminin saldırı tespitinde kullanılabileceğini kanıtlayan, anormal durumları bulmada yeni bir bakış açısı getiren ve ilerde geliştirilmeye çok açık bir çalışma niteliğindedir.

INTRUSION DETECTION WITH PATTERN CLASSIFICATION

SUMMARY

The computers become more and more faster, and number of computer users and internet users increase day by day, which are indicators of technology improvements. Unfortunately, not all of these people use technology in the good way, some of them use it for his/her or others benefit in bad way, to find vulnerable sides of it.

Computer hacking is in our day a very popular research topic, and it is going to be also. Then, as more and more preventions of computer attacks are developed, attackers create new, unseen attacking methods. Today, many big or little firms are exposed of computer hijacking and in order not to lose their prestige, they explain only a few of these happenings.

Intrusion detection systems are developed since 1980's. Basically, there are two types of intrusion detection systems: Behaviour based and knowledge-based. Knowledge-based systems can only detect the intrusions which are defined in their knowledge database. They are incapable of detecting new and unseen intrusions. Behaviour based intrusion detection systems learn first normal behaviour and then they define deviations from these behaviour as anomaly. In both types of intrusion detecting, algorithms like expert systems and data mining are used and many intrusion detection systems are developed alternative to each other.

Pattern classification is used in last years in the field of intrusion detection, it is used for many years by many fields as biology, image recognition, and there are many algorithms by this subject. Pattern classification can combine knowledge-based and behaviour based intrusion detection and guide to find the optimum solution.

In Pattern Classification there are two methods: Supervised clustering and unsupervised clustering. By supervised clustering the algorithm runs first with training data, so the algorithm learns the clusters and their characteristic. If the algorithm runs then with test data, it determines that which cluster this test data belongs to. By unsupervised clustering the clusters of the training data is not known. The patterns with similar features are grouped into same cluster, then these clusters are labeled. By all of these algortihms, the features of the patterns, the selected features and the weights of the features influence the result directly.

By KDD cup, organized every year by ACM Special Interest Group on Knowledge Discovery and Data Mining , is in 1999 intrusion detection data used, and these data has been a guide to development of many intrusion detection systems.

In this thesis, these data has been used to develop an intrusion detection system with pattern classification. This system is named by CLIDS (Cluster based Intrusion Detection System). The system is trained first with known attacks and the cluster characteristics are determined in the training data set. So, by doing this, the known clustering algorithms are not used directly. The distinctive features of the training data are selected by the FCBF algorithm developed by Lei Yu and Huan Liu, which is proven by its results, and these features are used to make clusters. By this clustering, the symbolic values (protocol type, service type, flag type etc.) which have a big role by intrusion detection are brought forward and the symbolic values

which are selected by FCBF, are given as labels to the clusters. Then, the algorithm in CLIDS compares the test patterns with the clusters of the “normal – attack” data and finds the nearest clusters, if it finds more than one cluster, than it finds which cluster should be selected, if the test pattern is not near enough than the limit defined previously, it labels it as “anomaly”.

However CLIDS is not working real time, it is a work which proves that supervised pattern classification and the feature selection can be used by intrusion detection, it brings a new look for finding “anomalies” and it is very open to be developed more.

1. INTRODUCTION

1.1. Aim of This Thesis

Intrusion detection is a part of computer security. Other parts may be firewalls, electronic signatures, encrypting, IPSEC protocol, antivirus programs etc. However common features of all these security items are the same:

- Authentication
- Authorization
- Non-Repudiation
- Confidentiality
- Integrity
- Availability

Every security system perform some or all features above.

In this thesis, Chapter 1 gives a first look to intrusion detection. Intrusion detection is introduced briefly, in order to give an idea everybody, who are not familiar with the term computer security.

Chapter 2 gives some basic knowledge to understand the network protocols and the attacks which use the vulnerabilites of these protocols. In Chapter 3 the classification of ID systems is explained and some example ID systems are introduced. In Chapter 4, some basic knowledge of pattern classification is studied. Chapter 5 introduces research in intrusion detection with pattern classification. And at the end Chapter 6 studies CLIDS which is implemented and presented in this thesis in detail.

1.2. Definition of Intrusion Detection

Intrusion detection is the process of monitoring the events occuring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or bypass the security mechanisms of a computer or network. [1]

1.3. Intrusions and Intruders in History

Internet was born in 1990's , so network intrusions have a history of about 15, but host based intrusions are more old. In the Figure 1.1 it is shown that the attack sophistication becomes more and more complicated, although intruder knowledge becomes low, because of attack tools, which can be found on internet widely. [3]

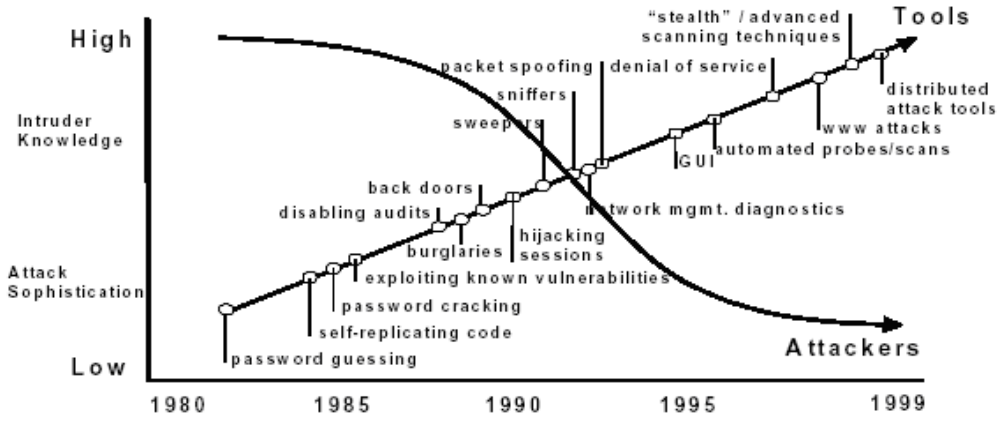

Figure 1.1 Intruder Knowledge vs. Attack Sophitication [3]

1.4. Terminology

Intrusion detection is a young field, and many terms are not used consistently. Here are some ID concepts explained:

Attack: An action conducted by one adversary, the intruder, against another adversary, the victim.The intruder carries out an attack with a specific objective in mind. From the perspective of an administrator responsible for maintaining a system, an attack is a set of one or more events that may have one or more security consequences. From the perspective of an intruder, an attack is a mechanism to fulfill an objective.

Exploit: The process of using a vulnerability to violate a security policy. A tool or defined method that could be used to violate a security policy is often referred to as an exploit script.

False negative: An event that the IDS fails to identify as an intrusion when one has in fact occurred

False positive: An event, incorrectly identified by the IDS as being an intrusion when none has occurred

Incident: A collection of data representing one or more related attacks. Attacks may be related by attacker, type of attack, objectives, sites, or timing.

Intruder: The person who carries out an attack. Attacker is a common synonym for intruder. The words attacker and intruder apply only after an attack has occurred. A potential intruder may be referred to as an adversary. Since the label of intruder is assigned by the victim of the intrusion and is therefore contingent on the victim's definition of encroachment, there can be no ubiquitous categorization of actions as being intrusive or not.

Intrusion: A common synonym for the word “attack”; more precisely, a successful attack.

Vulnerability: A feature or a combination of features of a system that allows an adversary to place the system in a state that is contrary to the desires of the people responsible for the system and increases the probability or magnitude of undesirable behaviour in or of the system. [2]

2. NETWORK PROTOCOLS AND NETWORK INTRUSIONS

2.1. Network Protocols

The TCP-IP protocol is the protocol that the computers use to communicate each other. This protocol is used in local area networks as well as in wide area netwoks, such as Internet.

2.2. Structure of the Protocol Stack

The TCP-IP stack contains four protocol layers, as shown in Figure 2.2. The four layers are stacked so that each one uses the services of the layer below it. [21]

- Applications: Such as mail, login, file transfer, http...
- Transport: The TCP protocol, supports the applications by providing a reliable “virtual circuit”. The UDP protocol do not provide a reliable “virtual circuit”.
- Internet: The IP potocol serves as a packet multiplexer.
- Network interface: The bottom layer consists of device drivers that manage the physical communications medium, such as ethernet. [21]

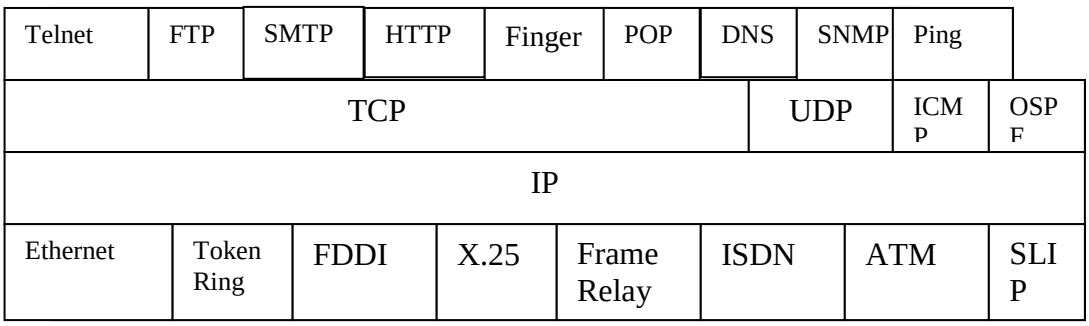

Figure 2.2 Simplified TCP-IP Protocol Stack [21]

2.2.1. Encapsulation and the Packet Headers

A packet which is used in TCP-IP protocol is formed of data and headers of protocols , which are encapsulated, as shown in Figure 2.3

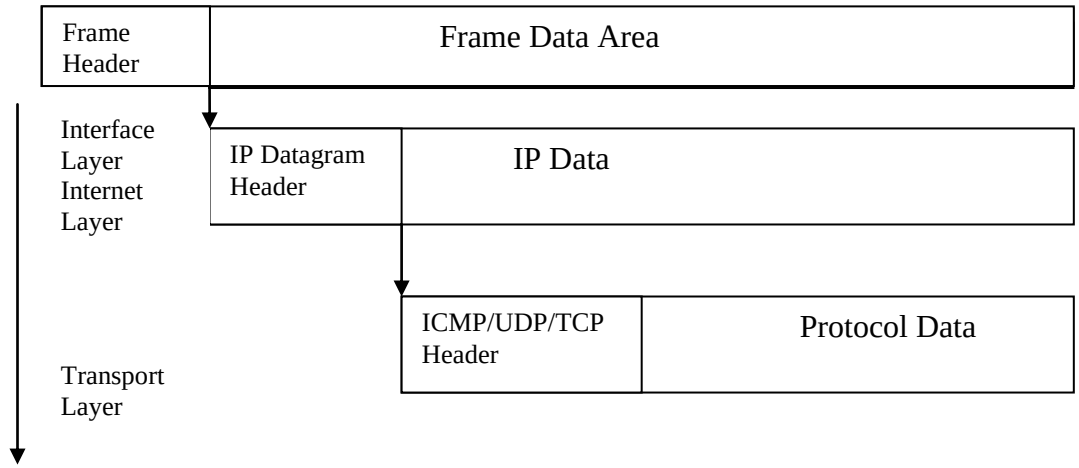

Figure 2.3 Encapsulation of headers [22]

2.2.1.1. TCP Header

The TCP header, which is shown in Figure 2.4, is the inner header of packet. The data area contains the application data.

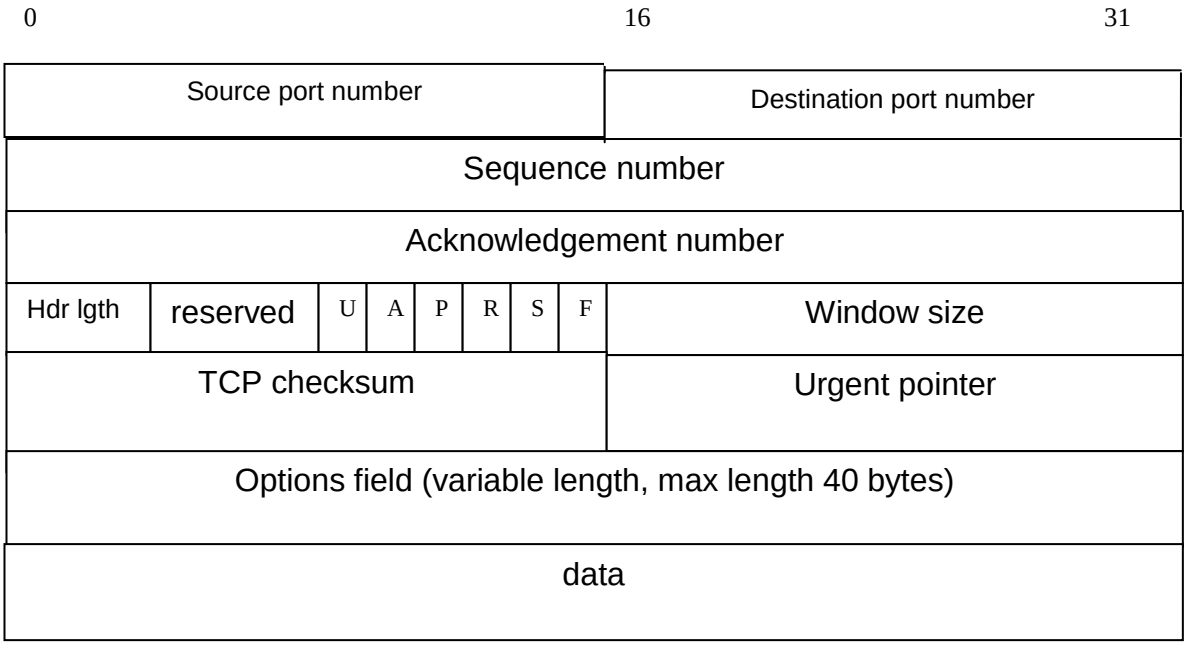

Figure 2.4 The TCP Header [22]

The header segments have the following meanings:

Source port number (16 bits): The port number of the source system

Destination port number(16 bits): The port number of the destination system

Sequence number (32 bits): The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgement number: If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Header Length(Hdr lgth): The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.

Reserved (6 bits): Reserved for future use. Must be zero.

The segment flags:

SYN (S) : synchronize the sequence numbers to establish a connection

ACK (A): acknowledgement number is valid

RST (R): reset (abort) the connection

FIN (F): sender is finished sending data –initiate a half close

PSH (P): tells receiver not to buffer the data before passing it to the application (interactive applications use this)

URG (U): urgent pointer is valid (often results from an interrupt)

Window (16 bits): The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

Checksum(16 bits) : The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text.

Urgent Pointer(16 bits): This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment.

Options(variable): Options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum. [35]

2.2.1.2. UDP Header

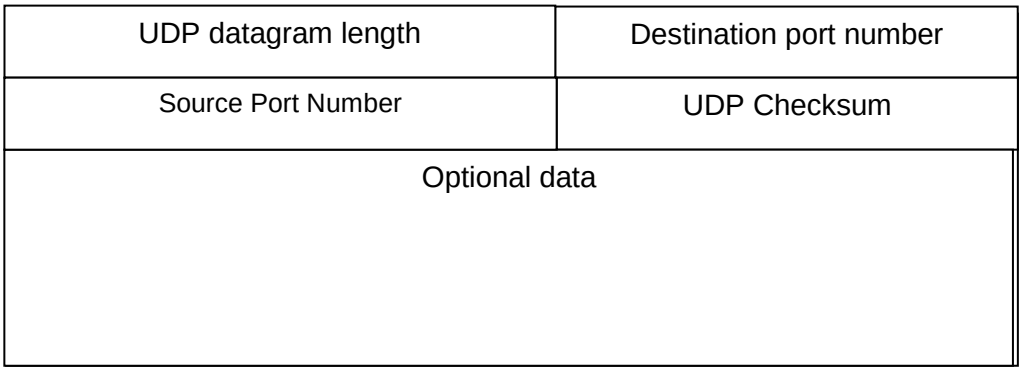

Figure 2.5 The UDP Header [22]

If the UDP protocol is used as the transport protocol, the UDP Header, which is shown Figure 2.5 is the inner header.

Source Port(16 bits):The port number of the sender. Cleared to zero if not used.

Destination Port(16 bits)The port this packet is addressed to.

Length(16 bits):The length in bytes of the UDP header and the encapsulated data. The minimum value for this field is 8.

Checksum(16 bits) Computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is cleared to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF. [35]

2.2.1.3. ICMP Header

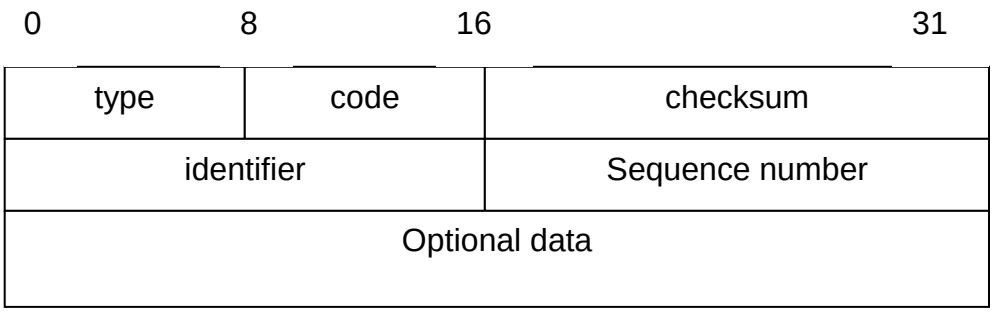

Figure 2.6 The ICMP Header [22]

The ICMP Header format depends on type and code. In Figure 2.6 is example specific format , echo request and reply is illustrated.

The Internet Control Message Protocol (ICMP) is used for error reporting and debugging of the IP Protocol. Some of ICMP's functions are to:

Announce network errors: Such as a host or entire portion of the network being unreachable, due to some type of failure.

Announce network congestion: When a router begins buffering too many packets, due to an inability to transmit them as fast as they are being received, it will generate ICMP Source Quench messages. Directed at the sender, these messages should cause the rate of packet transmission to be slowed.

Assist Troubleshooting: ICMP supports an Echo function, which just sends a packet on a round--trip between two hosts.

Announce Timeouts: If an IP packet's TTL field drops to zero, the router discarding the packet will often generate an ICMP packet announcing this fact. [35]

2.2.2. TCP Session Establishment and Closing

The setup phase of a TCP connection is a three way handshake. The client machine sends a TCP packet to the server with an initial TCP sequence number and the SYN flag set. The server sends back a packet with both SYN and ACK bit is set, an initial sequence number , as well as an acknowledgement for the client’s initial sequence number. Finally the client sends back a packet acknowledging the server’s initial sequence number. For remainder of the session the ACK bit is set.

At the end of TCP session one party initiates the closing sequence by sending a FIN packet (the ACK packet is still valid to keep the packets sequnced in correct order). The FIN is ACKed by the other end of the connection and a “half close” is taken place, which means that no more data will be flowing in that direction. Since TCP connection is full-duplex (data can be flowing in each direction independently), each directional channel mest be shut down independently. Figure 2.7 shows a TCP session establishment and closing. [22]

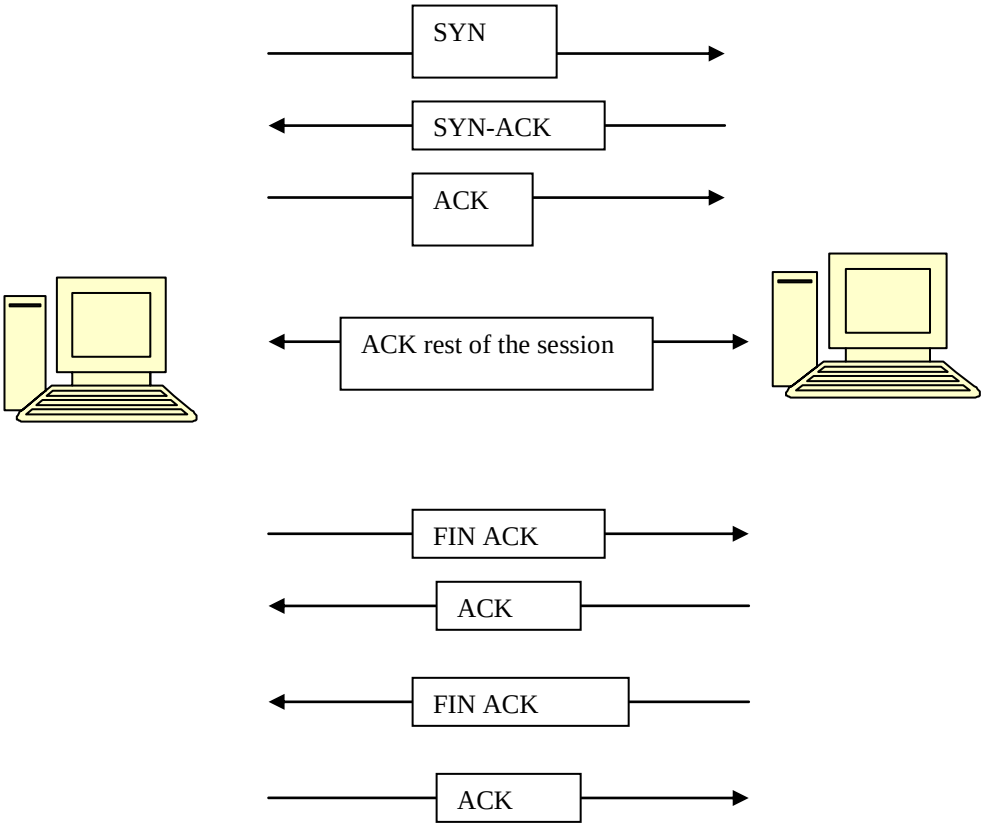

Figure 2.7 TCP Session Establishment and Closing [22]

2.3. Types of Network Intrusions

There are many intrusions types, some of them are the most seen types. The network intrusions can be grouped in two groups: Denial of Service Attack, Probe Attack

2.3.1. Denial of Service Attacks

The common feature of this type of intrusions is to bring the operating system of victim machine in a blocked and unstable state.

2.3.1.1. Smurf Attack

By Smurf attack, the victim machine gets too many packets, so that its operating system is blocked.

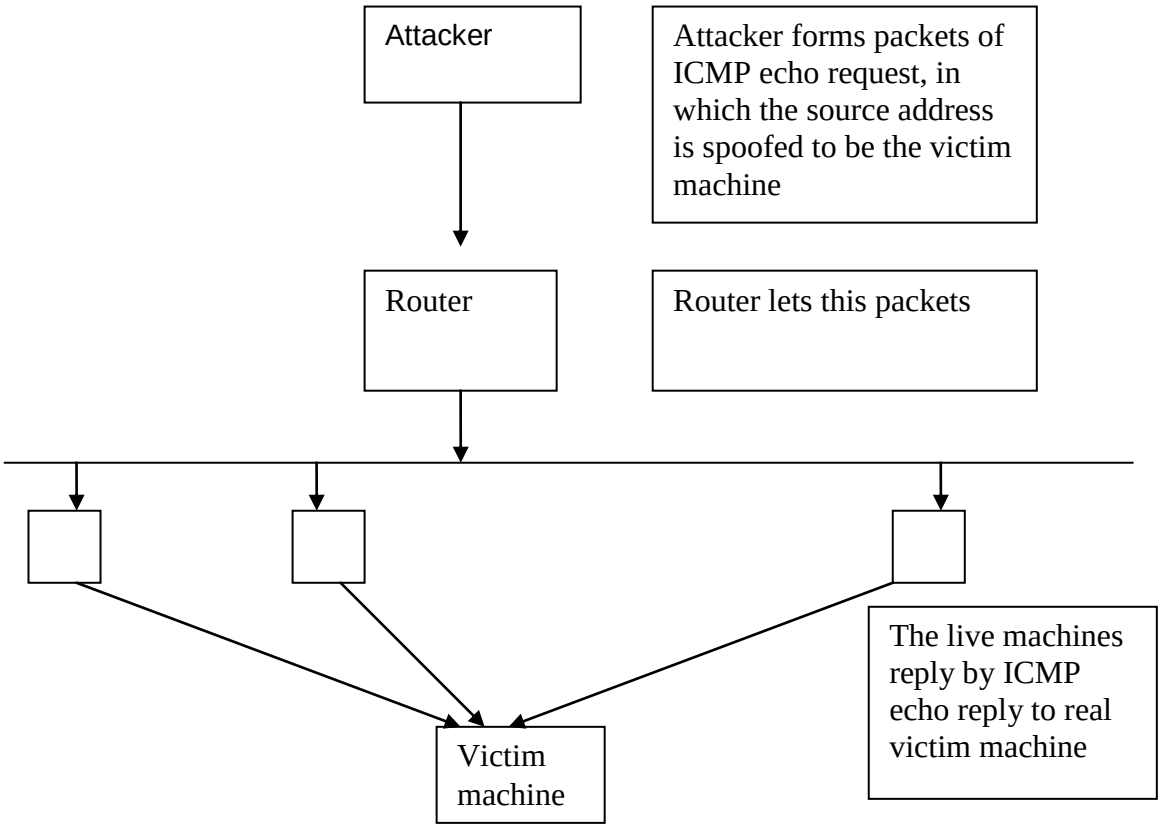

Figure 2.8 The Smurf Attack [23]

If a Smurf attack, which is illustrated in Figure 2.8 network traffic is sniffed by Tcpdump program, these data may be seen. In Figure 2.9 , the logs shows a Smurf attack. The timestamps are close together and ICMP echo request is broadcasted. [22]

```
00:00:05.327 spoofed.target.com > 192.168.15.255: icmp echo request
00:00:05.342 spoofed.target.com > 192.168.1.255: icmp echo request
00:00:14.154 spoofed.target.com > 192.168.15.255: icmp echo request
00:00:14.171 spoofed.target.com > 192.168.1.255: icmp echo request

05:20:48.261 spoofed.target.com > 192.168.0.0: icmp echo request
05:20:48.263 spoofed.target.com > 255.255.255.255: icmp echo request
05:21:35.792 spoofed.target.com > 192.168.0.0: icmp echo request
05:21:35.819 spoofed.target.com > 255.255.255.255: icmp echo request
```

Figure 2.9 Smurf attack logs [22]

2.3.1.2. Ping of Death Attack

The Ping of Death attack causes a buffer to overflow on the target host by sending an echo request packet that is larger than the maximum IP packet size of 65535 bytes. Theoretically, any IP packet that is larger than the maximum packet is could be used, but the attack has been popularized in the form of an ICMP echo request.

In order to generate such as an “impossible packet”, the attacker uses special tools to craft fragments and send them to the target. Because no intermediary network devices will attempt to reassemble the fragments, the packets are simply forwarded until they reach the specified destination address. When the target host receives these fragments and tries to reassemble them or process the reassembled datagram its operating system may crash or hang.

If a Ping of Death attack network traffic is sniffed by Tcpdump program, these data may be seen. In Figure 2.10 , the logs show a Ping of Death attack. In the last line it is seen that the attacker sends a ping packet that is larger than the maximum IP packet size of 65535 bytes (380+65360=65740) [22]

```
12:43:58.431 big.pinger.org > www.mynetwork .net : icmp echo request
(frag 4321: 380@0+)
12:43:58.431 big.pinger.org > www.mynetwork .net : icmp echo request
(frag 4321: 380@2656+)
12:43:58.431 big.pinger.org > www.mynetwork .net : icmp echo request
(frag 4321: 380@3040+)

...
12:43:58.431 big.pinger.org > www.mynetwork .net : icmp echo request
(frag 4321: 380@649476+)
12:43:58.431 big.pinger.org > www.mynetwork .net : icmp echo request
(frag 4321: 380@65360+)
```

Figure 2.10 Ping of Death attack logs [22]

2.3.1.3. TearDrop Attack

The Teardrop attack depends on the fact that the network protocols are not good at math. They are especially bad at negative numbers.

In Figure 2.11, the logs show a Teardrop attack. The top line shows a fragment named 242 with 36 octets of data of offset 0. The second line shows 4 more octets of data for offset 24. Therefore to service this packet the operating system would have to rewind from 36 to 24. Negative numbers can translate to very large positive numbers, and so the operating system, and so the operating system is likely to

scribble all over some other program's section of memory. [23] If this many times happens, the system may be blocked.

```
while-e-coyote.45599  > target.net.3964 :udp 28 (frag 242:36@0+)
while-e-coyote  > target.net.3964 :udp 28 (frag 242:4@24)
```

Figure 2.11 TearDrop Attack [23]

2.3.2. Probe Attacks

The common feature of these intrusions is to find live hosts or ports.

2.3.2.1. PortSweep Attack

The Portsweep attack tries to find live ports on a host, because an open port indicates that a service is offered and if an attacker knows what services are offered, he/she may be able to guess what security vulnerabilities are available to exploit. For a Linux operating system it is determined which flag will be set by which flag of source packet. On Table 2.1 the flags on response packets are showed.

Table 2.1 TCP flags on response packets with TCP flags [22]

Flags	Live Port	Dead Port
None	0	RA
F	0	RA
S	SA	RA
SF	SFA	RA
R	0	0
RF	0	0
SR	0	0
SRF	0	0
A	R	R
FA	R	R
SA	R	R
SFA	R	R
RA	0	0
RFA	0	0
SRA	0	0
SFRA	0	0

If the attacker do not want to use packets having the SYN flag set, he/she can use packets with no flag is set. Because based on RFC specifications

- a closed port should respond with RESET
- an open port should simply discard the probe packet and not respond at all [22]

In Figure 2.12, the logs show Portsweep attack.

```
11:33:36.225 scanner.org.63816 > target.com.821: .
11:33:36.225 scanner.org.63816 > target.com.405: .
11:33:36.225 scanner.org.63816 > target.com.391: .
11:33:36.225 scanner.org.63816 > target.com.59: .
11:33:36.225 scanner.org.63816 > target.com.91: .
```

Figure 2.12 Scanning with Null packets (no flags) [22]

2.3.2.2. Ipsweep Attack

The Ipsweep attack is to find live hosts. If the attacker finds live hosts, he/she can begin other types of attacks. These attack is also done in the same way of portsweep attack, but by this attack hosts more than one are scanned.

2.4. The KDD Cup 99 Data

The KDD Cup is the annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining, the leading professional organization of data miners.

The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest used a version of this dataset, which is also used by this thesis.

Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. In Chapter 3.3.1.5 is this detailed explained.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

The datasets contain a total of 22 training attack types, with an additional 18 types in the test data only. This data has the features as defined in Table 2.2. [27]

Table 2.2 Features used by KDD Cup data to identify packets and connections [27]

Feature Name	Description	Type
duration	length (number of seconds) of the	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http,	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	Continuous
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised''	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete

su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	Discrete
count	number of connections to the same host as the current connection in the	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

The attack types trained in KDD Cup data are as in Table 2.3.

Table 2.3 KDD Cup 99 attack types [27]

Dos	Probe	R2L	U2R
smurf	portsweep	ftp_write	buffer_overflow
teardrop	ipsweep	guess_passwd	perl
neptune	satan	imap	loadmodule
back	nmap	multihop	rootkit
pod		phf	
land		spy	
		warezclient	
		warezmaster	

The data in these attack categories are not in the same number as in Figure 2.13.

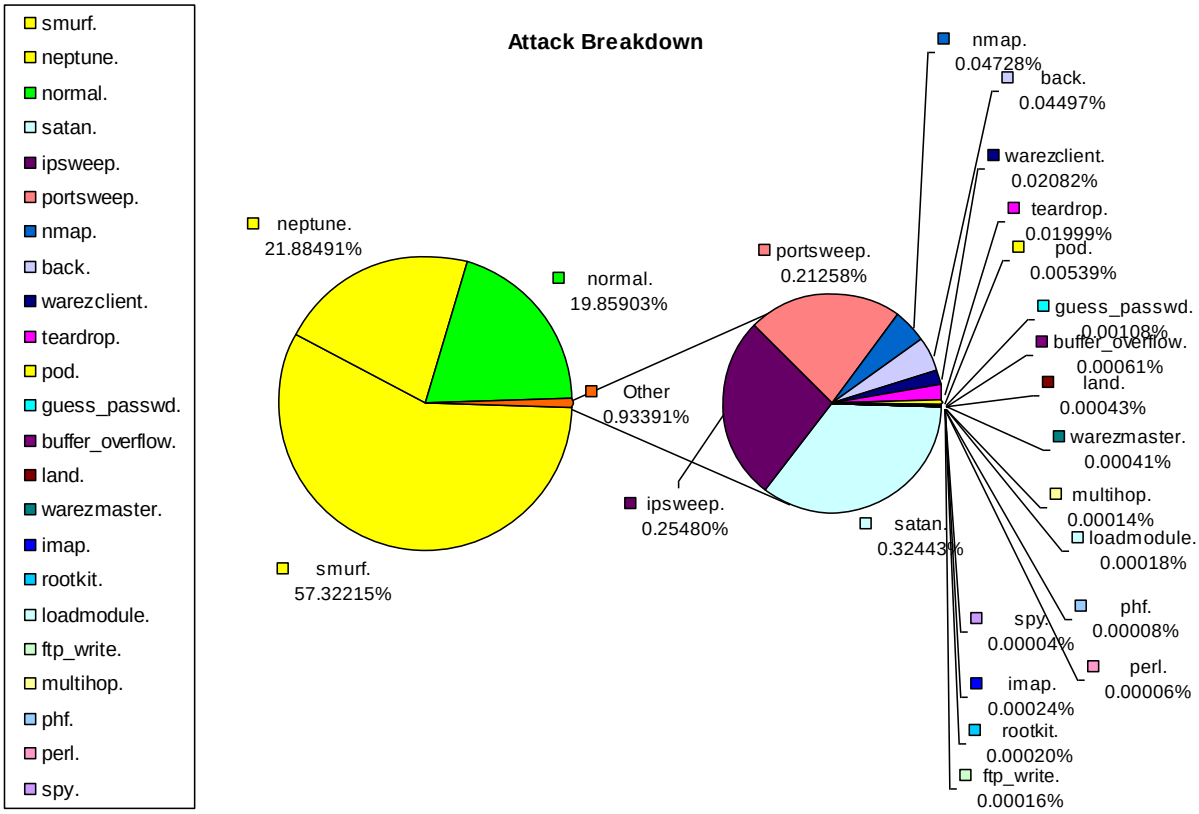

Figure 2.13 KDD Cup 99 Attack Categorization [29]

In KDD Cup 99 Data, the TCP flags has some meanings which are defined as in Table 2.4.

Table 2.4 Flags by KDD Cup 99 Data [24]

Flag	Meaning
S0	Connection attempt seen, no reply.
S1	Connection established, not terminated.
SF	Normal establishment and termination. Note that this is the same symbol as for state S1. You can tell the two apart because for S1 there will not be any byte counts in the summary, while for SF there will be.
REJ	Connection attempt rejected.
S2	Connection established and close attempt by originator seen (but no reply from responder).
S3	Connection established and close attempt by responder seen (but no reply from originator).
RSTO	Connection established, originator aborted (sent a RST).
RSTR	Established, responder aborted.
RSTOS0	Originator sent a SYN followed by a RST, we never saw a SYN ACK from the responder.
RSTRH	Responder sent a SYN ACK followed by a RST, we never saw a SYN from the (purported) originator.
SH	Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was ``half'' open).
SHR	Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.
OTH	No SYN seen, just midstream traffic (a ``partial connection'' that was not later closed).

3. INTRUSION DETECTION SYSTEMS

Intrusion detection system history begins by 1980, when James P. Anderson wrote a report published in planning study for the U.S Air Force. In this report, he proposed changes to computer audit mechanisms to provide information for use by computer security personnel when tracking problems. He proposed a taxonomy fo classifiying risks and threats to computer systems. If the not authorized user of computer uses data or program, it is “External Penetration”, if the authorized user of computer uses not authorized data or program, it is “Internal Penetration”. He devotes to the problem associated with masquearades, those adversaries who acess systems using purloined user ids and passwords. He suggests that some sort of statistical analysis of user behaviour, capable of determining unusual patterns of system use, might represent a way of detecting masquerades. This suggestion was tested in the next milestone of intrusion detection the IDES project. [21]

First intrusion detection systems were host-based, because the internet is born and grows up in early 1990’s. Then the network-based ID systems were developed.

3.1. Classification of Intrusion Detection Systems

The intrusion detection systems are classified as in Figure 3.14. [2]

The detection method describes the characteristics of the analyzer. When the intrusion detection system uses information about the normal behaviour of the system it monitors, it is behaviour based. This means , if IDS finds deviation from normal behaviour, then this type of detection is described as anomaly detection. When the intrusion detection system uses information about the attacks, it is knowledge-based. This means IDS have a information(a database) about the known intrusions, so it matches the behaviours with that information. This type of detection is described as misuse detection. [2]

The behaviour on detection describes the response of the intrusion detection system to attacks. When it actively reacts to the attack by taking either corrective (closing holes) or pro-active (logging out possible attackers, closing down services) actions,

then the intrusion detection system is said to be active. If the intrusion detection system merely generates alarms (including paging, etc), it is said to be passive. [2]

The audit source location discriminates intrusion detection systems based on the kind of input information they analyze. This input information can be audit trails (for example system logs) on a host, network packets, application logs or intrusion detection alerts generated by other intrusion detection systems. [2]

The detection paradigm describes the detection mechanism used by the intrusion detection system. Intrusion detection systems can evaluate states (secure or insecure) or transitions (from secure to insecure). In addition, this evaluation can be performed in a non-obtrusive way or by actively stimulating the system to obtain a response. [2]

The usage-frequency is an orthogonal concept. Certain intrusion detection systems have real-time continuous monitoring capabilities, whereas others have to be run periodically. [2]

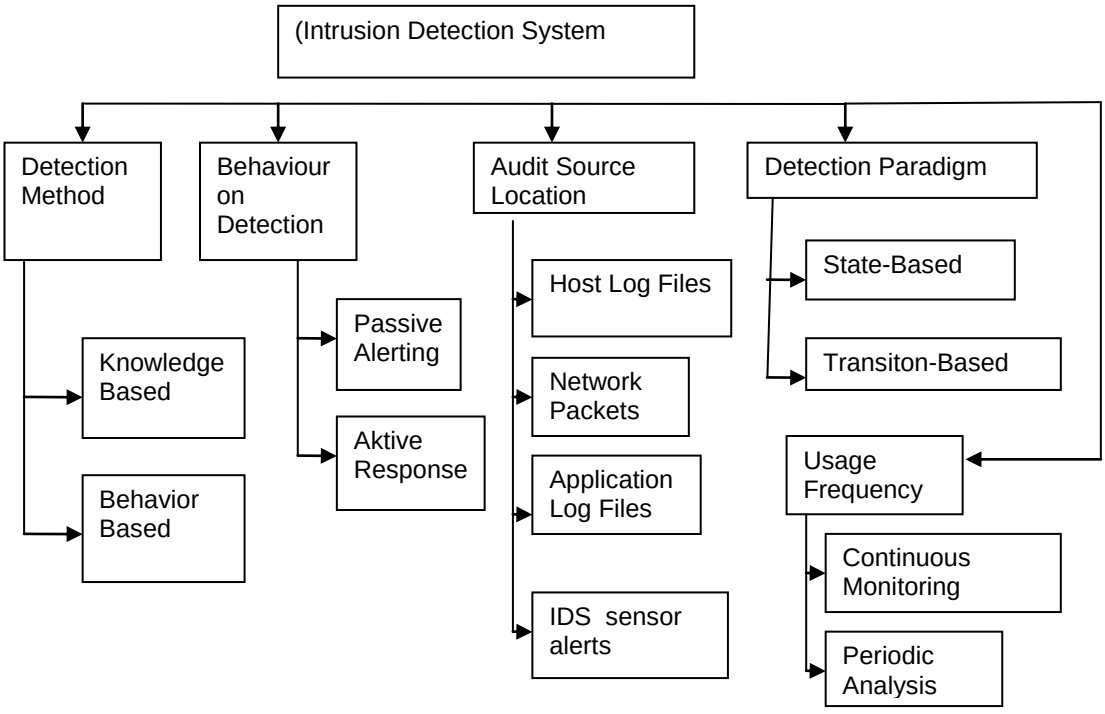

Figure 3.14 Intrusion Detection Taxonomy [2]

3.2. Intrusion Detection System Components

Most intrusion detection systems have common features. The functionality of an generic IDS can be logically distributed into three components: sensors, analyzers, and a user interface. [3]

Sensors : Sensors are responsible for collecting data. The input for a sensor may be any part of a system that could contain evidence of an intrusion. Example types of input to a sensor are network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer. [3]

Analyzers: Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred. The output of this component is an indication that an intrusion has occurred. The output may include evidence supporting the conclusion that an intrusion occurred. The analyzer may provide guidance about what actions to take as a result of the intrusion. [3]

User interface: The user interface to an IDS enables a user to view output from the system or control the behaviour of the system. In some systems, the user interface may equate to a “manager,” “director,” or “console” component. [3]

In addition to these three essential components, an IDS may be supported by a “honeypot,” i.e., a system designed and configured to be visible to an intruder and to appear to have known vulnerabilities. A honeypot provides an environment and additional information that can be used to support intrusion analysis. The honeypot serves as a sensor for an IDS by waiting for intruders to attack the apparently vulnerable system. Having a honeypot serve as a sensor provides indications and warnings of an attack. Honeypots have the ability to detect intrusions in a controlled environment and preserve a known state. [3]

3.3. Intrusion Detection Systems by Detection Method

3.3.1. Knowledge Based Intrusion Detection Systems

An ID System that uses misuse detection, have information about specific attacks and system vulnerabilities. So, it compares the logs with that information, and when it finds a match, it raises alarm.

Advantages of the knowledge-based approches are that they have low false positive alarm rate. It is more easy to understand and to update.

Disadvantages are the difficulty gathering of required information on the known attacks and keeping it up to date with new vulnebarilities and environments. When it is not enough often updated, the false negative alarm rate can be very high, which means intrusion patterns are treated as normal.

Misuse-based systems were some of the earliest systems proposed and having reduced false positive rate they are most common form of IDS used in production today, for example SNORT.

3.3.1.1. Expert Systems

Expert Systems are used primarily by knowledge based intrusion detection. The expert system contains a set of rules that describe attacks. Audit events are translated into facts carrying their semantics in the expert system, and the inference engine draws conclusions using these rules and facts. [2]

Examples of misuse detection systems using expert systems are IDES (Intrusion Detection Expert System)(1987), ComputerWatch (1990), NIDX (Network Intrusion Detection Expert System)(1988) [6]

3.3.1.2. Signature Analysis

The semantic description of the attacks is transformed into information that can be found in the audit trail in a straightforward way. For example, attack scenarios might be translated into the sequences of audit events they genarate or into patterns of data that can be sought in the audit trail generated by the system. [6]

This technique allows a very efficient implementation and is therefore applied in commercial intrusion detection products. [2]

Systems that use signature analysis include Haystack(1988), NetRanger(1990), RealSecure(1990) and MuSig(1998).[6]

3.3.1.3. Petri Nets

To represent signatures of intrusions, IDIOT, a knowledge-based intrusion detection system developed by Purdue University uses Colored Petri Nets. Figure 3.15 shows a simple example of a Colored Petri Net that issues an alarm if the number of unsuccessful login attempts exceeds four within one minute. The transition represented by a vertical bar, from state S1 to S2 can occur if there is a token in state S1 and an unsuccesful login attempt. The time of the first unsuccessful login attempt is stored in the token variable T1. The transition from state S4 to state S5

can happen if there is a token in S4, an unsuccesful login attempt, and the time difference between this and the first unsuccesful login attempt is less then 60 seconds. Reaching final state S5 corresponds to a matched signature and may therefore result in an alarm being issued. [2]

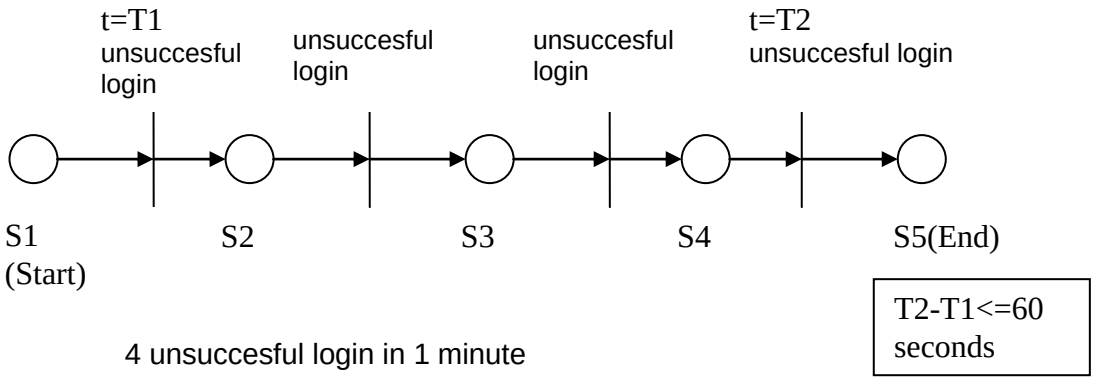

Figure 3.15 PetriNet State Diagram used by IDIOT [2]

Advantages of colored Petri nets include their generality, their conceptual simplicity,and their ability to be represented as graphs. However, matching a complex signature against the audit trail can become computationally expensive. [2]

3.3.1.4. State Transition Analysis

State transition analysis describes attacks with a set of goals and transitions based on state transition diagrams. Any event that triggers an attack state will be considered an intrusion. Examples of systems applying state transition analysis are USTAT (Unix State Transition Analysis Tool) (1992) and NetSTAT (Network-based State Transition Analysis Tool) (1998).[6]

3.3.1.5. Data Mining

Data mining approach can be used in misuse detection as well as in anomaly detection.

Data mining refers to a process of nontrivial extraction of implicit, previously unknown, and potentially useful information from databases. Example misuse detection systems that use data mining include JAM (Java Agents for Metalearning) (1998), MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) (2000), and Automated Discovery of Concise Predictive Rules for Intrusion Detection (1999). [6]

JAM (developed at Columbia University) uses data mining techniques to discover patterns of intrusions. It then applies a meta-learning classifier to learn the signature of attacks. The association rules algorithm determines relationships between fields in the audit trail records, and the frequent episodes algorithm models sequential patterns of audit events. Features are then extracted from both algorithms and used to compute models of intrusion behaviour. The classifiers build the signature of attacks. So essentially, data mining in JAM builds a misuse detection model. [6]

MADAM ID uses data mining to develop rules for misuse detection. The motivation is that current systems require extensive manual effort to develop rules for misuse detection. While MADAM ID performed well in the 1998 DARPA evaluation of intrusion detection systems, it is ineffective in detecting attacks that have not already been specified. [6]

In the paper “A Data Mining Framework for Building Intrusion Detection Models” [5] , Wenke Lee, Salvatore J. Stolfo and Kui W. Mok (Columbia University) explain how they mine intrusion data. They participated in the DARPA Intrusion Detection Evaluation Program, prepared and managed by MIT Lincoln Labs. They were provided with about 4 gigabytes of compressed tcpdump data of 7 weeks of network traffic. This data can be processed into about 5 million of connection records of about 100 bytes each. The data contains content (i.e., the data portion) of every packet transmitted between hosts inside and outside a simulated military base.

Four main categories of attacks were simulated, they are:

- DOS, denial-of-service, for example, ping-of-death,teardrop, smurf, syn flood, etc.,
- R2L, unauthorized access from a remote machine, for example, guessing password,
- U2R, unauthorized access to local superuser privileges by a local unprivileged user, for example, various of buffer overflow attacks,
- PROBING, surveillance and probing, for example, port-scan, ping-sweep, etc.

They used Bro [24] tool, which perform IP packet filtering and reassambling,and allow event handlers to output summarized connection records.

Example network connection records are in Table 3.5.

Table 3.5 Network connection records by BRO [5]

timestamp	duration	service	src_host	dst_host	src_bytes	dst_bytes	flag	...
1.1	0	http	spoofed_1	victim	0	0	S0	...
1.1	0	http	spoofed_2	victim	0	0	S0	...
1.1	0	http	spoofed_3	victim	0	0	S0	...
1.1	0	http	spoofed_4	victim	0	0	S0	...
1.1	0	http	spoofed_5	victim	0	0	S0	...
1.1	0	http	spoofed_6	victim	0	0	S0	...
1.1	0	http	spoofed_7	victim	0	0	S0	...
...	...	...	...	...	...	...	...	...
10.1	2	ftp	A	B	200	300	SF	...
12.3	1	smtp	B	D	250	300	SF	...
13.4	60	telnet	A	D	200	12100	SF	...
13.7	1	smtp	B	C	200	300	SF	...
15.2	1	http	D	A	200	0	REJ	...
...	...	...	...	...	...	...	...	...

The approach taken by MADAM ID differs from the others covered in that instead of looking at individual packets it focuses on connection sessions. The approach is unique in that it is data-led rather than model-led. Data Mining tools and methods are used to distinguish anomalous sessions from normal sessions in an iterative manner using the training data as reference [4]

In their approach, the learned rules replace the manually encoded intrusion patterns and profiles, and system features and measures are selected by considering the statistical patterns computed from the audit data. Meta-learning is used to learn the correlation of intrusion evidence from multiple detection models, and produce a combined detection models. [5]

Their experiment results on intrusion data are shown in Table 3.6.

Table 3.6 Example "traffic" connection records [5]

label	service	flag	host_count	srv_count	host_REJ_%	host_diff_srv_%	duration	...
normal	ecr_i	SF	1	1	0	1	0	...
smurf	ecr_i	SF	350	350	0	0	0	...
satan	user-level	REJ	231	1	85%	89%	0	...
normal	http	SF	1	0	0	1	3	...
...	...	...	...	...	...	...	...	...

Their RIPPER [25] algorithm used in MADAM ID gives rules as in Table 3.7.

Table 3.7 Example RIPPER Rules for DOS and PROBING attacks [5]

RIPPER Rule	Meaning
Smurf:- service= ecr_i, host-count >= 5, Host_srv_count >= 5	If the service is icmp echo request and for the past 2 seconds, the number of connections that have the same destination host as the current one is at least 5, and the number of connections that have the same service as the current one is at least 5, then this is a smurf attack (a DOS attack).
Satan: host_REJ_% >= %83, host_diff_srv_% >= %87	If for the connections in the past 2 seconds that have same the destination host as the current connection, the percentage of rejected connections are at least %87, then this is a satan attack (a PROBING attack).

Although their models were intended for misuse detection, they experiment the features for new intrusion data. The results are as in Table 3.8.

Table 3.8 Comparing Detection Rates (in %) on Old and New Attacks by MADAM ID [5]

Category	Old	New
DOS	79.9	24.3
PROBING	97.0	96.7
U2R	75	81.8
R2L	60.0	5.9
Overall	80.2	37.7

3.3.2. Behaviour Based Intrusion Detection Systems

Behaviour based intrusion detection techniques assume that an intrusion can be detected by observing a deviation from the normal or expected behaviour of the system or the users. The model of normal or valid behaviour is extracted from reference information collected by various means. The intrusion detection system later compares this model with the current activity. When a deviation is observed, an alarm is raised. [2]

Advantages of behaviour based approaches are that they can detect attempts to exploit new and unforeseen vulnerabilities, so they can even discover new attacks. [2]

The high false positive alarm rate is the main drawback of behaviour based techniques because the entire scope of the behaviour of an information system may not be covered during the learning phase. Also behaviour can change over time, so the behaviour profile should be periodically updated. Here should be considered that the behaviour profile do not include intrusive behaviour. [2]

3.3.2.1. Statistics

The most widely used to build behaviour based intrusion detection systems is statistics. The user or system behaviour is measured by a number of variables over time. Examples of these variables are the login and logout time of each session, the resource duration, and the amount of processor-memory-disk-resources consumed during the session. The original model keeps averages of all these variables and detects whether thresholds are exceeded based on the standard deviation of the variable. [2]

Example systems employing statistical methods for anomaly detection are IDES (Intrusion Detection Expert System), NIDES (Next- Generation Intrusion Detection Expert System), and Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD). [2]

3.3.2.2. Expert systems

Expert system used in behaviour based intrusion detection depends also on statistical anomaly detection. Two examples are Wisdom&Sense and AT&T's Computer Watch. The tool of Wisdom&Sense first builds a set of rules that statistically describe the behaviour of the users based on recordings of their activities over a given period of time. Current activity is then matched against these rules to detect inconsistent behaviour. [2]

The tool of AT&T checks the actions of users according to a set of rules that describe proper usage policy. [2]

3.3.2.3. Neural Networks

Neural networks are algorithmic techniques used to first learn the relationship between two sets of information, and then "generalize" to obtain new input-output

pairs in a reasonable way. In the intrusion detection field, neural networks have been mainly used to learn the behaviour of actors in the system (e.g users, daemons). [2]

An example is NNID (Neural Network Intrusion Detector). A host-based, backpropagation neural network intrusion detection system was tested experimentally on a system of 10 users. The system was 96% accurate in detecting unusual activity, with 7% false alarm rate.[13]

3.3.2.4. Computer Immunology

The idea of using immunological principles in computer security has been described by Stephanie Forrest in 1994. [6]

This technique attempts to build a model of normal behaviour of the UNIX network services, rather than of the behaviour of users. This model consists of short sequences of system calls made by processes. The tool first collects a set of reference audits, which represent the appropriate behaviour of the service, and extracts a reference table containing all the known “good” sequences of system calls. These patterns are then used for live monitoring to check whether the sequences generated are listed in the table; if not the intrusion detection system generates an alarm. This technique has a very low false alarm rate if the reference table is sufficiently exhaustive. [2]

3.3.2.5. Data Mining

Intrusion detection attempts to identify existing attack patterns and recognise new intrusion methods, employing methods from sciences such as mathematics, statistics and machine learning. Data mining, generally perceived to be a tool to discover unknown regularities in data, also lends itself to this task. In particular, it promises to help in the detection of previously unseen attacks by establishing sets of commonly observed regularities in network data. These sets can be compared to current traffic for deviation analysis. Data mining techniques, however, are traditionally employed on large amounts of off-line data. It therefore remains to be seen how well they are able to support ID systems commonly required to operate in real time. [33]

Applications of data mining to anomaly detection include ADAM (Audit Data Analysis and Mining) (2001), IDDM (Intrusion Detection using Data Mining) (2001), and eBayes (2000). [6]

ADAM (developed at George Mason University Center for Secure Information Systems) uses a combination of association rules mining and classification to discover attacks in TCP dump data. The ADAM system is able to detect network intrusions in real time with a very low false alarm rate. [6]

IDDM (Intrusion Detection using Data Mining) project focuses on the use of data mining in the latter context, by producing descriptions of network data and using this information for deviation analysis. It aims to explore data mining as a supporting paradigm in extending intrusion detection capabilities. The system characterizes change between network data descriptions at different times, and produces alarms when detecting large deviations between descriptions. However, IDDM has problems achieving real-time operation. [6]

3.3.2.6. Pattern Classification

All intrusion detection systems that use pattern classification algorithms are behaviour based. Some examples of them are given in Chapter 5, after studying pattern classification algorithms in Chapter 4.

4. PATTERN CLASSIFICATION

Pattern classification is a type of machine learning, that is to build machines that can recognize patterns [16]. The areas where pattern classification used are:

- Speech recognition
- DNA sequence identification
- Fingerprint identification
- Optical character recognition
- Archeology
- Geology etc. [8]

4.1. Definitions and Notation

Pattern (feature vector, observation or datum) : x is a single data item used by the clustering algorithm. It typically consists of a vector of *d* measurements : $x = (x_1, x_2, ..., x_d)$

Feature : The individual scalar components x_i of a pattern x are called features (or attributes).

Dimensionality : *d* is the dimensionality of the pattern or of the pattern space.

Pattern set : A pattern set is denoted $H = \{x_1, x_2, .., x_n\}$. The *i*th pattern in *H* is denoted $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})$. In many cases a pattern set to be clustered is viewed as *an n x d* pattern matrix.

Class : A class, in the abstract, refers to a state of nature that governs the pattern generation process in some cases. More concretely, a class can be viewed as a source of patterns whose distribution in feature space is governed by a probability density specific to the class. Clustering techniques attempt to group patterns so that the classes thereby obtained reflect the different pattern generation processes represented in the pattern set.

Hard clustering : Hard clustering techniques assign a class label l_i to each patterns x_i identifying its class. The set of all labels for a pattern set *H* is $L = \{l_1, l_2, .., l_n\}$ with $l_i \in \{1, .., k\}$, where k is the number of clusters.

Fuzzy clustering : Fuzzy clustering procedures assign to each input pattern x_i a fractional degree of membership f_{ij} in each output cluster *j.*

Distance measure : Distance measure is a metric on the feature space used to quantify the similarity of patterns. [19]

4.2. Typical Components of Clustering

Although there are many clustering algorithms, they have all common tasks:

a. Pattern representation: The features of patterns are determined. This features are optionally selected and/or extracted and normalized. (Chapter 4.2.2)

b. Definition of pattern similitary measure: The measure is selected appropriate to the data domain and clustering algorithm, such as Euclidean distance, Manhattan distance or Mahalonobis distance. (Chapter 4.2.1)

c. Clustering or grouping: The algorithm makes clusters of the data

d. Data abstraction: The clusters are simple labeled, usually in terms of cluster prototypes or representative patterns such as centroid.

e. Assesment of output: The output is validated if needed. [19]

The common stages in clustering is shown in Figure 4.16.

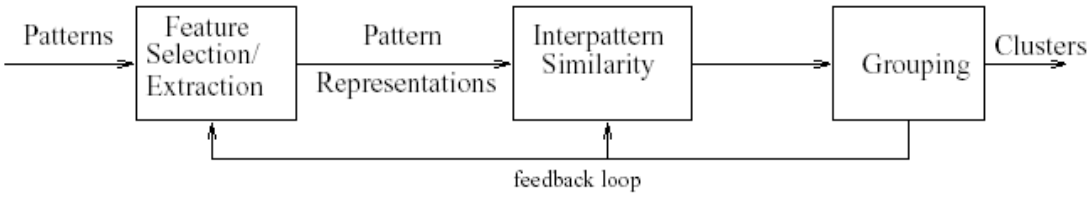

Figure 4.16 Stages in clustering [19]

4.2.1. Distance Measures

The distance functions or metrics must have four properties. D is distance and for all vectors a,b, and c, these properties are as follows: [16]

Nonnegativitiy: $D(a,b) >= 0$

Reflexitivity: $D(a,b) = 0$ if and only if $a = b$

Symmetry : $D(a,b) = D(b,a)$

Triangle inequality: $D(a,b) + D(b,c) >= D(a,c)$

The Minkowski Metric :

$$L_k(a,b) = \left(\sum_{i=1}^{d} | a_i - b_i |^k \right)^{1/k} \quad (4.1)$$

The Euclidean Metric :

The Euclidean metric is the L_2 form of Minkowski metric:

$$D(a,b) = \left(\sum_{k=1}^{d} (a_k - b_k)^2 \right)^{1/2} \quad (4.2)$$

The Manhattan or City Block Metric :

The Manhattan metric is the L_1 form of Minkowski metric

$$D(a,b) = \sum_{k=1}^{d} | a_k - b_k | \quad (4.3)$$

4.2.2. The Normalization of Features

The drawback to direct use of the Minkowski metrics is the tendency of the largest scaled feature to dominate the others. Solutions to this problem include normalization of the continuous features (to a common range or variance) or other weighting schemes. [19]

Some normalization approaches are: [34]

- Min-max normalization:

  v is a variable, v' is the normalized variable, $\max A$ is the maximum value which can v take, $\min A$ is the minimum value which v can take, $new\_\max A$ is the maximum value which v' can take, $new\_\min A$ is the minimum value which v' can take.

$$v' = \left[\frac{v - \min A}{\max A - \min A} * (new\_\max A - new\_\min A) \right] + new\_\min A \quad (4.4)$$

- Zero-mean normalization:

In pattern set *H* with N patterns , v_j is *j*th element of a pattern, v_j' is the normalized *j*th element of the pattern.

$$v_j' = \frac{v_j - mean_j}{std\_dev_j} \quad (4.5)$$

The mean vector of *j*th feature in *H* $mean_j$ is calculated as follows: [12]

$$mean_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \quad (4.6)$$

The standard deviation is calculated as follows:

$$std\_dev_j = \sqrt{\left(\frac{1}{n-1}\right)\sum_{i=1}^{n}(x_{i,j} - mean_j)^2} \quad (4.7)$$

4.3. Pattern Classification Algorithms

Pattern classification algorithms can be grouped as in Figure 4.17:

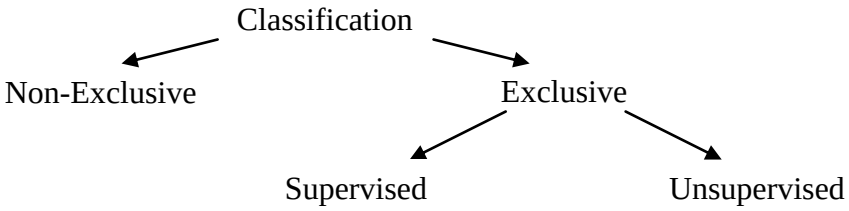

Figure 4.17 Classification of Pattern Classification algorithms [8]

- Exclusive vs. Nonexclusive: By exclusive classification each object belongs to exactly one subset, or cluster. Nonexclusive classification can assign an object to several classes.
- Unsupervised vs. Supervised: An unsupervised classification uses only the proximity matrix to perform the classification. Supervised classification uses category labels on the subjects as well as the proximity matrix. Unsupervised classification is named also clustering. [8]

Bayesian decision theory is a fundemantal statistical approach to the problem of pattern classification [16]. So, before studying some pattern classification algorithms, we will first study, bayesian decision rule.

Bayesian decision rule:

Suppose we have two categories w_1, and w_2. The prior probabilities are P(w_i). And we know the value of the feature of the categories which is defined by x. The conditional probability is *p*(x|w_i). The posterior probabilty is

$$P(w_i \mid x) = \frac{p(x \mid w_i)P(w_i)}{p(x)} \tag{4.8}$$

where p(x) is the probability density.

Bayesian decison rule in two categories is as follows:
Decide w_1 if $P(w_1) > P(w_2)$; otherwise decide w_2 . [16]

4.3.1. Supervised Classification

In supervised learning, a teacher provides a category label or cost for each pattern in a training set, and seeks to reduce the sum of the costs for these patterns [16]. So , the patterns in test set are labeled by labels in training set in order to minimize the error. Some supervised classification algorithms are k-NN nearest neighbour algoritm and support vector machines.

4.3.1.1. K-Nearest Neighbour Rule

K-Nearest Neighbour rule is based on bayesian classificaition. Let V_n be the volume of n dimensional Euclidean space R_n, k_n be the samples falling in R_n, and $p_n(x)$ be the *n*th estimate for $p(x)$:

$$p_n(x) = \frac{k_n / n}{V_n} \tag{4.9}$$

Here the volume V_n is grown until it encloses k_n neighbours of x, and k_n is a function of *n* such as $k_n = \sqrt{n}$. This is the k_n-nearest neighbour estimation method. [16]

The k-Nearest Neighbour algorithm is the most basic of all Instance-Based Learning (IBL) methods. Instance-Based Learning (IBL) algorithms consist of simply storing the presented training examples (data). When a new instance is encountered, a set of similar, related instances is retrieved from memory and used to classify the query instance (target function). [18]

Other most common IBL methods are:

- Locally Weighted Regression
- Radial Basis Function [18]

The k-Nearest Neighbour algorithm assumes all instances correspond to points in the *n*-dimensional space R_n. The nearest neighbours of an instance are defined in terms of standard Euclidean geometry (distances between points in n-dimensional space). More precisely, let an arbitrary instance *x* be described by the feature attribute lists: < a1(x), a2(x), a3(x), ..., an(x)>, where ar(x) denotes the value of the r^{th} attribute of instance *x*. The distance between the two instances xi and xj is given by Equation 4.8. This is the general form for calculating distance in *n*-dimensional space. [18]

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{r=n} \left[a_r(x_i) - a_r(x_j)\right]^2} \qquad (4.10)$$

The k – nearest-neighbor rule is to classify x by assigning it the label most frequently represented among the k nearest samples and use a voting scheme.[16]

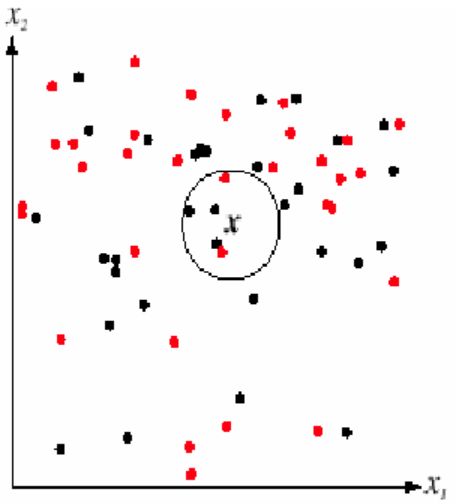

Figure 4.18 An example for the k-Nearest Neigbour rule [16]

In Figure 4.18 is an example for the k-Nearest Neighbour shown. The k-Nearest Neighbour query starts at the best point *x* and grows a spherical region until it

encloses k training samples and it labels the test point by a majority vote of these samples. The computational complexity of k–Nearest Neighbor rule is $O(n^2)$. So it has to be optimized for use in intrusion detection.[16]

4.3.1.2. Support Vector Machines

Support vector machines rely on preproccesing the data to represent patterns in a high dimension - typically much higher than the original feature space. With an appropriate nonlinear mapping $\rho(.)$ to a sufficiently high dimension, data from two categories can always be sperated by a hyperplane. Here it is assumed that x_k has been transformed to $y_k = \rho(x_k)$. For each of the *n* patterns, *k* = 1,2,...,*n* , $z_k = \pm 1$, according to whether pattern *k* is in w_1 or w_2. [16]

$$g(y) = a^t y \tag{4.11}$$

$$z_k g(y_k) \geq 1 \text{ , } k= 1,...,n \tag{4.12}$$

$$\frac{z_k g(y_k)}{\| a \|} \geq b, \; k=1,...,n \tag{4.13}$$

The goal is to find weight vector *a* that maximizes *b*. The solution vector can be scaled arbitrarily and still preserve the hyperplane, so to ensure the uniquness it should be

$$b \| a \| = 1 \tag{4.14}$$

The support vectors are the (transformed) training patterns for which Eq. (4.12) represents an equality, that is, the support vectors are (equally) close to the hyperplane(Figure 4.19). The support vectors are the training samples that define the optimal separating hyperplane and are the most difficult patterns to classify. Informally defined, they are the patterns most informative for the classification task. [16]

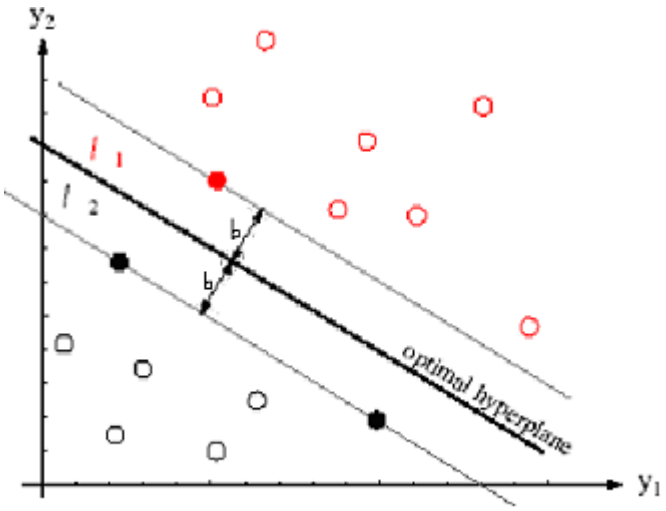

Figure 4.19 Support vectors and the hyperplane [16]

The hyperplane divides the R_n into two regions. To use the maximal margin classifier, it is determined on which side the test pattern lies and assign the corresponding class label. [26]

4.3.2. Unsupervised Learning and Clustering

In unsupervised learning or clustering there is no explicit teacher, and the system forms clusters or “natural groups” of the input patterns. “Natural” is always defined explicitly or implicitly in the clustering system itself, and given a particular set of patterns or cost function , different clustering algorithms lead to different clusters. [16]

The clustering algorithm taxonomy is illustrated in Figure 4.20. The best algorithm depends on data, which is to be clustered.

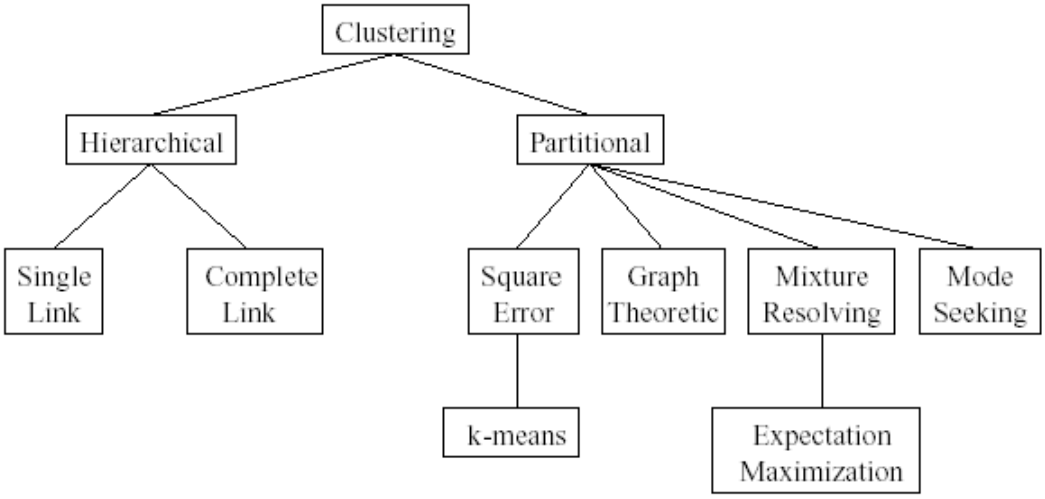

Figure 4.20 A taxonomy of clustering approaches [19]

A hierarchical classification is a nested sequence of partitions, whereas a partitional classification is a single partition. [8]

Here are k-means clustering as a type of partional clustering and hierarchical clustering studied.

4.3.2.1. K-Means Clustering

The most intuitive and frequently used criterion function in partitional clustering techniques is the squared error criterion, which tends to work well with isolated and compact clusters. The squared error for a clustering *L* of a pattern set *H* (containing *K* clusters) is

$$e^2(H,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i^{(j)} - c_j \|^2 \qquad (4.15)$$

where $x_i^{(j)}$ is the i^{th} pattern belonging to the j^{th} cluster and c_j is the centroid of the j^{th} cluster. [19]

The k-Means Algorithm [8]

Step 1. Select an initial partition with K clusters.

Step 2. Generate a new partition by assigning each pattern to its closest cluster center.

Step 3. Compute new cluster centers as the centers of the clusters.

Step 4. Repeat step2 and 3 until an optimum value of the criterion function is found. Typical convergence criteria are: no (or minimal) reassignment of patterns to new cluster centers, or minimal decrease in squared error.

Step 5. Adjust the number of clusters by merging and splitting existing clusters or by removing small, or outlier, clusters.

The advantage of this algorithm is that it has a computational complexity of O(n), where n is the number of patterns, but the disadvantage is it is sensitive of the initial clusters. [19]

4.3.2.2. Hierarchical Clustering

Hiearchical clustering algorithms yield a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change. [19]

Hierarchical Clustering Algorithm [8]

Step 1: Assign each object to its own cluster.

Step 2: Computer the distances between all clusters.

Step 3: Merge the two clusters that are closest to each other.

Step 4: Return to step 2 until there is only one cluster left.

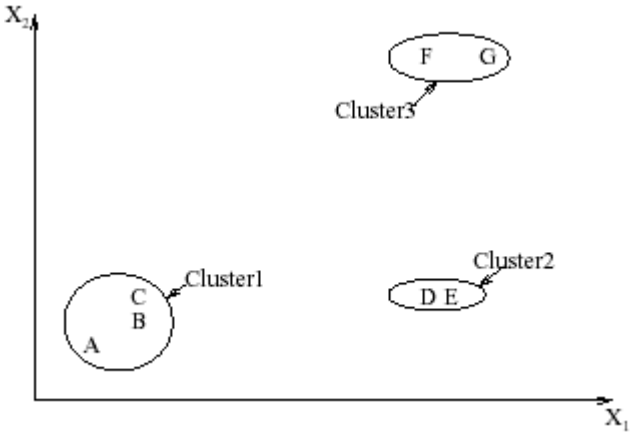

Figure 4.21 Points falling in three clusters [19]

In Figure 4.21, there are 7 points (A,B,C,D,E,F,G). If the hiearchical clustering algorithm is applied, first the points (B,C), (D,E), and (F,G) are merged in a cluster. The second loop of the algorithm merges the point A with the cluster of (B,C). The (D,E) cluster is also merged with (F,G) cluster. In the third and last loop, the (A,B,C) cluster is merged with (D,E, F,G) cluster. In Figure 4.22 the dendogram, which is the output of this algorithm is showed. [19]

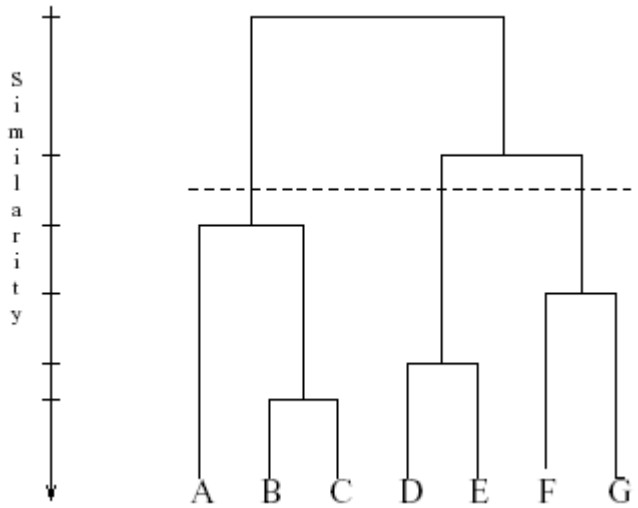

Figure 4.22 The dendogram obtained using hierarchical clustering [19]

In hierarchial clustering algorithm, cutting the dendogram, which means stopping the loop of the algorithm creates the clusters.

Most popular hierarchical clustering algorithms are single-link, complete-link and avearge link algorithms.

By the single-linkage clustering algorithm the distance between the closest nodes calculated: [8]

$$D_{SL}(C_i, C_j) = \min_{a \in c_i, b \in C_j} d(a,b) \tag{4.16}$$

By the complete-linkage clustering algorithm the distance between the farthest nodes calculated: [8]

$$D_{CL}(C_i, C_j) = \max_{a \in c_i, b \in C_j} d(a,b) \tag{4.17}$$

By the average-linkage clustering algorithm the distance between the median calculated: [8]

$$D_{AL}(C_i, C_j) = \frac{1}{N_i N_j} \sum_{a \in c_i, b \in C_j} (a,b) \tag{4.18}$$

The single-linkage algorithm allows clusters to grow long and thin. The complete-linkage algorithm produces more compact clusters. Both the single-linkage algorithm and the complete-linkage algorithm are susceptible to distortion by outliers or deviant observation. The average-linkage algorithm is an attempt to compromise between the extreme of the single-linkage algorithm and the complete-linkage algorithm.

4.3.2.3. Comparison of Hierachical vs. Partitional Algorithms

Hierarchical algorithms are more versatile than partitional algorithms. For example, the single-link clustering algorithm works well on data sets containing non-isotropic clusters including well-separated, chain-like, and concentric clusters, whereas a typical partitional algorithm such as the *k*-means algorithm works well only on data sets having isotropic clusters On the other hand, the time and space complexities of the partitional algorithms are typically lower than those of the hierarchical algorithms.

4.4. Feature Selection

Feature selection is highly important in pattern classification, especially for patterns with high dimensionality, because reducing the feature size can improve the speed of pattern classification.

In all feature selection algorithms the concept is the same: A feature is good if it s relevant to the class concept but is not redundant to any of other relavant features. That means a feature is good if it is higly correlated with the class but not highly correlated with any of the other feautures. [30]

To overcome this problem there are two approaches to measure the correlation between two random variables. One is based on classical linear correlation and the other is based on information theory. Under the first approach, the most well known measure is linear correlation coefficent. For a pair of variables (X,Y), the linear correlation coefficient r is given by the formula

$$r = \frac{\sum_i (x_i - x_i)(y_i - \bar{y}_i)}{\sqrt{\sum_i (x_i - \bar{x}_i)^2} \sqrt{\sum_i (y_i - \bar{y}_i)^2}} \tag{4.19}$$

where $\bar{x}_i$ is the mean of X and $\bar{y}_i$ is the mean of Y. The value r lies between -1 and 1, inclusive. If X and Y are completely correlated, r takes the value of 1 or -1; if X and Y are totally indpendent r is zero. It is symmetrical measure for two variables. [30]

It is known that if data is linearly seperable in original representation, it is still linearly seperable if all but one of a group of linearly dependent features are removed. However, it is not safe to always assume linear correlation between features in the real world. Linear correlation measures may not be able to capture correlations that are not linear in the nature. Another limitation is that the calculation requires all features contain numerical values. [30]

The FCBF(Fast Correlation Based Feature) solution adopts the second approach . So, it chooses a correlation measure based on information-theoretical concept of entropy, a measure of uncertainty of a random variable. The entropy of a variable X is defined as

$$H(X) = -\sum_i P(x_i) \log_2 (P(x_i)) \tag{4.20}$$

and the entropy of X after observing values of another variable Y is defined as

$$H(X \mid Y) = -\sum_i P(y_j) \sum_j P(x_i \mid y_j) \log_2 (P(x_i \mid y_j)) \quad (4.21)$$

where $P(x_i)$ is the prior probalities for all values of X and $P(x_i \mid y_i)$ is the posterior probabilities of X given the values of Y. The amount by which the entropy of X decreases reflects additional information about X provided by Y and is called information gain, given by

$$IG(X \mid Y) = H(X) - H(X \mid Y) \quad (4.22)$$

According to this measure, a feature Y is regarded more correlated to feature X than to feature Z, if $IG(X \mid Y) > IG(Z \mid Y)$. The information gain is symmetrical for two random variables X and Y. The algorithm uses symmetrical uncertaintity to ensure that the values are normalized:

$$SU(X,Y) = 2\left[\frac{IG(X \mid Y)}{H(X) + H(Y)}\right] \quad (4.23)$$

It compensates for information gain's bias toward features with more values and normalizes its values to the range [0,1] with the value 1 indicating that knowledge of the value of either one completely predicts the value of the other and the value 0 indicating that X and Y are independent. [30]

The FBCF algorithm is described in [30].

One of the other famous feature selection algorithms is Relief. Relief is a feature weighting technique. It is originally introduced by Rendell L. Kira K. and extended to Relief-f by Kononenko I. for handling noisy, incomplete and multi-class data sets. [32]

Relief is based on L_1 Metric. Similarities between feature values are defined by

$$\varphi^R(x_q^i, x_q^j) = \begin{cases} \dfrac{|\, x_q^i - x_q^j \,|}{|\, x_q^{\max} - x_q^{\min} \,|} & \text{if feature } q \text{ is continuous} \\ 1 & \text{if feature } q \text{ is discrete and } x_q^i \neq x_q^j \\ 0 & \text{if feature } q \text{ is discrete and } x_q^i = x_q^j \end{cases} \quad (4.24)$$

The similarity between instances is defined by

$$d_{ij}^{R} = \sum_{q=1}^{Q} \varphi^{R}(x_q^i, x_q^j) \tag{4.25}$$

Iniatilizing every feature weight with zero, the weight algorithm iterates m times through a procedure for weight optimization. First the procedure randomly selects an instance from the instance base, secondly determines *k* nearest neighbours of the instance in the same class (nearest hits) *k* nearest neighbours in diffrent classes (nearest misses) and thirdly updates each feature weight by

$$w_q = w_q - \frac{1}{k * m}\left(\sum_{i=1}^{k} \rho^{R}\left(x_q^j, x_{qhit}^{j_i}\right) + \sum_{i=1}^{k} \rho^{R}\left(x_q^j, x_{qmiss}^{j_i}\right)\right) \tag{4.26}$$

The Relief algorithm expects a user defined threshold value Φ. Feature *q* is selected if $w_q > \Phi$ holds, otherwise it is neglected. [32]

The performance results of FCBF algorithm is given as in Table 4.9 and Table 4.10.

Table 4.9 The running time (in ms) and the number of selected features for each feature selection algorithm [30]

Title	Running Time				# Selected Features			
	FCBF	CorrSF	ReliefF	ConsSF	FCBF	CorrSF	ReliefF	ConsSF
Lung-cancer	20	50	50	110	5	8	5	4
Promoters	20	50	100	190	4	4	4	4
Splice	200	961	2343	34920	6	6	11	10
USCensus90	541	932	7601	161121	2	1	2	13
CoIL2000	470	3756	7751	341231	3	10	12	29
Chemical	121	450	2234	14000	4	7	7	11
Musk2	971	8903	18066	175453	2	10	2	11
Arrhythmia	151	2002	2233	31235	6	25	25	24
Isolet	3174	177986	17025	203973	23	137	23	11
Multi-Features	4286	125190	21711	133932	14	87	14	7
Average	995	32028	7911	109617	7	30	11	12

Table 4.10 Accuracy of C4.5 on selected features for each feature selection algorithm [30]

Title	Full Set	FCBF	CorrSF	ReliefF	ConsSF
Lung-cancer	80.83 ±22.92	87.50 ±16.32	84.17 ±16.87	80.83 ±22.92	84.17 ±16.87
Promoters	86.91 ±6.45	87.73 ±6.55	87.73 ±6.55	89.64 ±5.47	84.00 ±6.15
Splice	94.14 ±1.57	93.48 ±2.20	93.48 ±2.20	89.25 ±1.94	93.92 ±1.53
USCensus90	98.27 ±0.19	98.08 ±0.22	97.95 ±0.15	98.08 ±0.22	98.22 ±0.30
CoIL2000	93.97 ±0.21	94.02 ±0.07	94.02 ±0.07	94.02 ±0.07	93.99 ±0.20
Chemical	94.65 ±2.03	95.51 ±2.31	96.47 ±2.15	93.48 ±1.79	95.72 ±2.09
Musk2	96.79 ±0.81	91.33 ±0.51	95.56 ±0.73	94.62 ±0.92	95.38 ±0.75
Arrhythmia	67.25 ±3.68	72.79 ±6.30	68.58 ±7.41	65.90 ±8.23	67.48 ±4.49
Isolet	79.10 ±2.79	75.77 ±4.07	80.70 ±4.94	52.44 ±3.61	69.23 ±4.53
Multi-Features	94.30 ±1.49	95.06 ±0.86	94.95 ±0.96	80.45 ±2.41	90.80 ±1.75
Average	88.62 ±9.99	89.13 ±8.52	89.36 ±9.24	83.87 ±14.56	87.29 ±11.04

5. RESEARCH IN INTRUSION DETECTION WITH PATTERN CLASSIFICATION

In literature there are many recently published papers about intrusion detection with pattern classificaiton. Some of them are studied in this thesis.

5.1. Intrusion Detection with Unsupervised Clustering

5.1.1. Intrusion Detection with Single-Linkage Clustering

By this work KDD Cup 99 data has been used [12] .The data was normalized with zero-mean normalization (Chapter 4.2.2). Then with the algorithm as shown in Figure 5.23 the data has been classified into clusters.

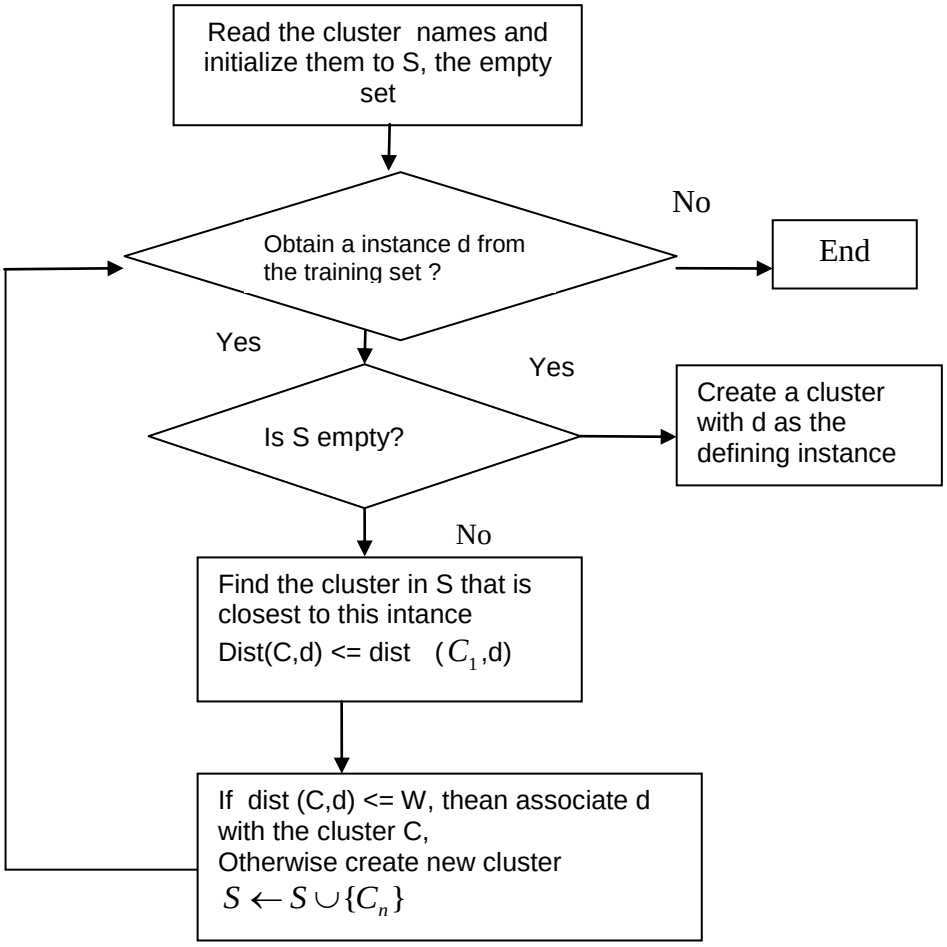

Figure 5.23 The algorithm of IDS with single linkage clustering [12]

S is the cluster set, *C* is the cluster, *W* is the constant width, *d* is the distance between vectors.

In Table 5.11 are the results.

Table 5.11 Results of Single Linkage algorithm [12]

Width	N	Detection Rate	False Positive Rate
20	%15	35.7%	1.44%
20	%7	66.2%	2.7%
20	%2	0.88%	8.14%

N is percentage of the largest cluster to label normal during detection.

5.1.2. Intrusion Detection with Optimized KNN Algorithm

Though a k-Nearest Neighbor value will give yield an excellent sense of how closely a new instance fits in with the rest of the data, it is extremely costly to calculate, with a complexity of $O(n^2)$. This problem is accentuated with the complex data used for intrusion detection, as a large amount of data-points are required for a solid cross-section of the data. Thus, it is necessary to find computational shortcuts that will allow performing anomaly detection with greater celerity. The clusters by optimized k-NN Algorithm is showed in Figure 5.24 . [30]

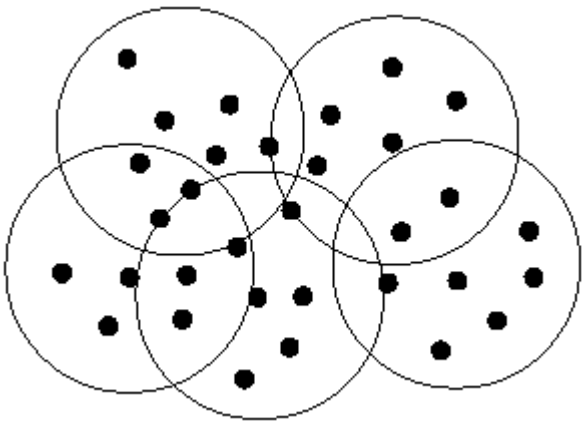

Figure 5.24 Clusters by optimized k-NN algortithm [30]

In this project, KDD Cup 99 Data are used. The instances in data classified into clusters with fixed width and most of the computation is spent checking the distance

between points in D to the cluster centers. This is significantly more efficient than computing the pair wise distances between all points.

Performance of this algorithm is as in Table 5.12.

Table 5.12 Performance of optimized k-NN-Algorithm [30]

Attack Type	Detection Rate	False Positive Rate
DOS	91%	8%
Probe	23%	6%
U2R	11%	4%
R2L	5%	2%

5.1.3. Intrusion Detection with Y-means Algorithm

The k-Means algorithm is further researched and new algorithms are devoleped: One of them is Y-means algorithm. [19]

The k-Means algorithm has two shortcomings in clustering large data sets: number of clusters dependency and degeneracy. Number of clusters dependency is that the value of *k* is very critical to the clustering result. Obtaining the optimal *k* for a given data set is an NP-hard problem. Degeneracy means that the clustering may end with some empty clusters. This is not what we expect since the classes of the empty clusters are meaningless for the classification. [19]

By Y-means algorithm number of clusters , *k*, can be a given integer between 1 and *n* exclusively, where *n* is the total number of instances. The next step is to find whether there are any empty clusters. If there are, new clusters will be created to replace these empty clusters; and then instances will be reassigned to existing centers. This iteration will continue until there is no empty cluster. Subsequently, the outliers of clusters will be removed to form new clusters, in which instances are more similar to each other; and overlapped adjacent clusters will merge into a new cluster. In this way, the value of *k* will be determined automatically by splitting or merging clusters. The last step is to label the clusters according to their populations; that is, if the population ratio of one cluster is above a given threshold, all the instances in the cluster will be classified as normal; otherwise, they are labeled intrusive. The Y-means algorithm is showed in Figure 5.25. [19]

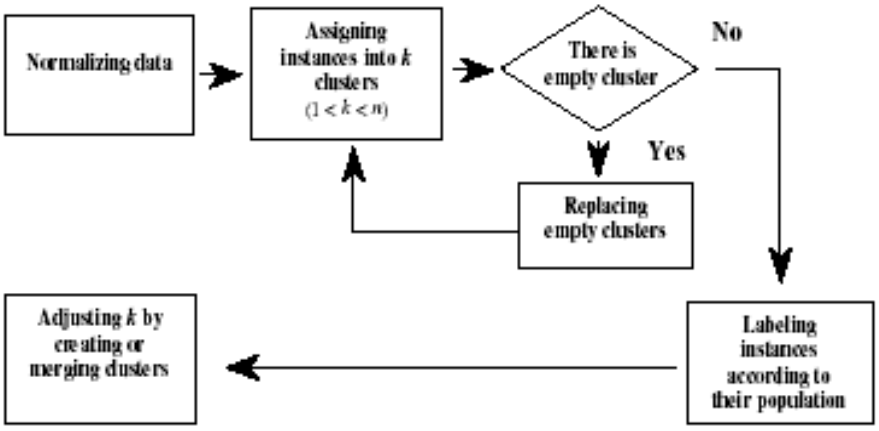

Figure 5.25 The Y-means Algorithm [19]

Y-means is tested with a subset of KDD Cup 99 Data, with 2,456 instances.

On average, it detected 86.63% of intrusions with a 1.53% of false alarm rate, as shown in Figure 5.26.

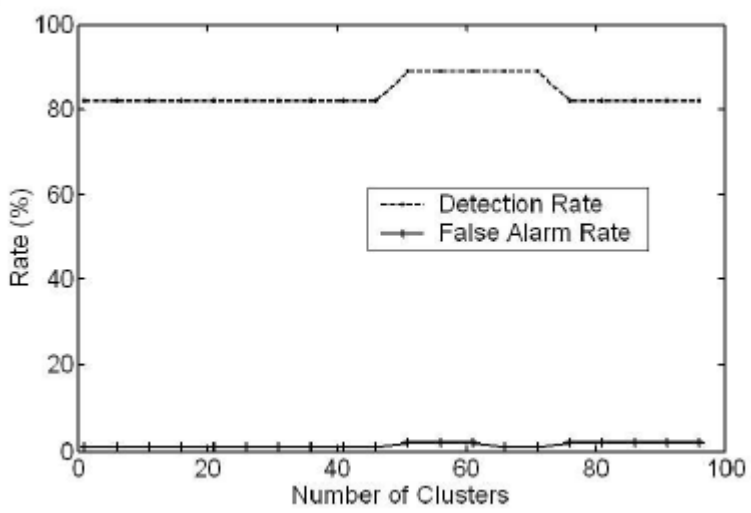

Figure 5.26 Y-means with different initial number of clusters [19]

5.2. Intrusion Detection with Supervised Clustering

5.2.1. MINDS (Minnesota Intrusion Detection System)

The Minnesota Intrusion Detection System (MINDS) uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. Input to MINDS is Netflow version 5 data collected using Netflow tools. Netflow data for each 10 minute window, which typically result in 1 to 2 million flows, is stored in

a flat file. The analyst uses MINDS to analyze these 10-minute data files in a batch mode. Before applying MINDS to these data files, a data filtering step is performed by the system administrator to remove network traffic that the analyst is not interested in analyzing. For example, the removed attack-free network data in data filtering step may include the data coming from trusted sources, non-interesting network data (e.g. portions of *http* traffic) or unusual/anomalous network behavior for which it is known that it does not correspond to intrusive behavior. [16]

In Figure 5.27 , the architecture of MINDS is shown.

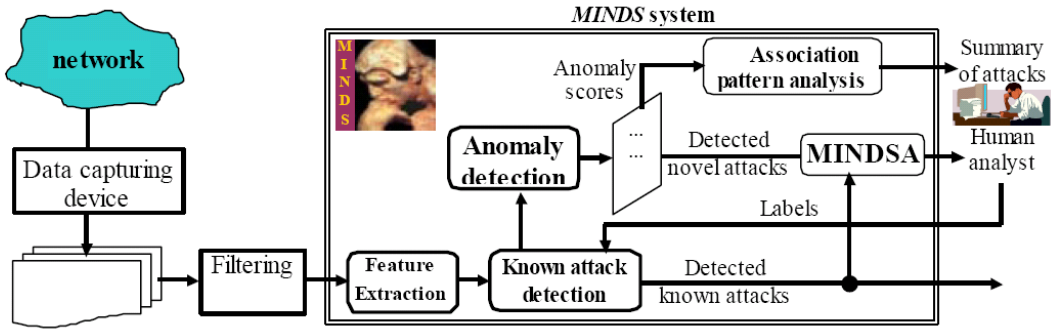

Figure 5.27 Architecture of Minnesota Intrusion Detection System [16]

The first step in MINDS includes constructing features that are used in the data mining analysis. Basic features include source IP address and port, destination IP address and port, protocol, flags, number of bytes, and number of packets. After the feature construction step, the known attack detection module is used to detect network connections that correspond to attacks for which the signatures are available, and then to remove them from further analysis. Next, the data is fed into the *MINDS* anomaly detection module that uses an outlier detection algorithm to assign an anomaly score to each network connection. A human analyst then has to look at only the most anomalous connections to determine if they are actual attacks or other interesting behavior. [16]

MINDS anomaly detection module assigns a degree of being an outlier to each data point, which is called the local outlier factor (LOF). The outlier factor of a data point is local in the sense that it measures the degree of being an outlier with respect to its neighborhood. For each data example, the density of the neighborhood is first computed. The LOF of specific data example p represents the average of the ratios of the density of the example p and the density of its nearest neighbors. [16]

In Figure 5.28 two outlier examples p1 and p2 are shown. The simple nearest neighbour approach based on computing the distances fail in these scenarios. However, the example p1 may be detected as outlier using the distances to the nearest neighbor. On the other side, LOF is able to capture both outliers (p1 and p2) due to the fact that it considers the density around the points. [16]

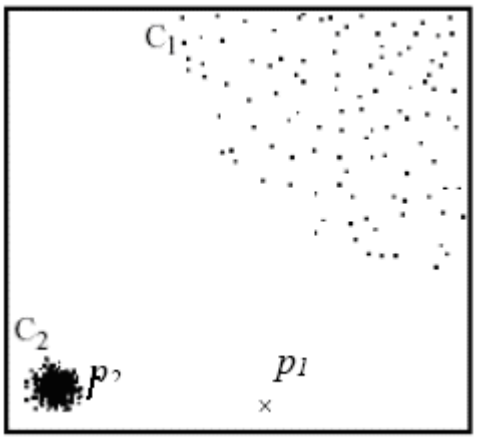

Figure 5.28 Outlier Examples [16]

As a result, MINDS and SNORT have been comparison. Here are the results:

- Content based attacks: These attacks are out of scope for the anomaly detection module in MINDS since it does not consider the content of the packets, and therefore SNORT is superior in identifying those attacks. However, SNORT is able to detect only those contentbased attacks that have known signatures/rules. Despite the fact that SNORT is more successful in detecting the content-based attacks, it is important to note that once a computer has been attacked successfully, its behavior could become anomalous and therefore detected by the anomaly detection module in MINDS.

- Scanning activities: When detecting various scanning activities SNORT and MINDS anomaly detection module have similar performance for certain types of scans, but they have very different detection capabilities for other types.

- Policy violations: MINDS anomaly detection module is much more successful than SNORT in detecting policy violations (e.g. rogue and unauthorized services), since it looks for unusual network behavior. SNORT may detect these policy violations only if it has a rule for each of these specific activities. [16]

6. APPLICATION – CLIDS (Cluster based Intrusion Detection System)

The implementation of this thesis is an intrusion detection system, which is named CLIDS (Cluster Based Intrusion Detection System) and it is based on feature selection and pattern classification. The training and the test values come from KDD Cup 99 data.

6.1. Specification of CLIDS

The feature selection algorithm is FCBF algorithm modified for using continuous values of KDD Cup 99 data. The FCBF algorithm expects that the value scala for every feature is given in a configuration file, so this is possible for symbolic values (like tcp,udp,ismp etc.) but impossible for continuous values (numbers). In order to use the continuous values in FCBF algorithm, the value scala for every feature with continuous values is split by a factor, default 10. So, in this example there will be 10 groups for every continuous feature.

The FCBF algorithm expects a configuration file, in which there are feature values in value file and the value file, in which there are the values with the cluster names. It evaluates the clusters given by these files and gives as the result the selected features, which value is bigger than predefined threshold.

So, our clusters are our attacks. CLIDS evalautes features pairwise between attacks. That means the selected features between normal-smurf, normal-teardrop, normal-nmap, etc., and smurf-teardrop, smurf-nmap etc. are evaluated. After getting all selected features and feature weights pairwise between all clusters, the “training clusters” are created. Then in the test phase, a test vector is tested if it is enough near to these training clusters by calculating the Euclidean distance. By this calculation the continuous values are normalized by min-max normalization.

6.1.1. Creating of Clusters by Training

The count of the training clusters depend on the attack clusters which they come from. After getting selected features between two attacks, the values which

correspond these features are selected. The values are also normalized by min-max normalization.

Example: For the example, let us see the “normal” and “neptune” clusters. We take 1000 “normal” and 1000 “neptune” instances. The modified FCBF algorithm gives us the result in Table 6.13.

Table 6.13 The selected features by normal-neptune

Selected Feature	Selected Feature Weight	Type of the Feature
flag	1.0	symbolic
serror_rate	0.9892109978609052	continuous
logged_in	0.8368075392414093	symbolic
service	0.6794651909145254	symbolic

The selected features with symbolic values gives the labels to the training clusters. So, in this example the training clusters with following labels are created.

Normal: http, SF,1
Smtp, SF, 1
Finger, SF, 0
Domain_u, SF, 0
Auth, SF, 1
http, SF, 0
telnet, SF, 1
ftp , SF, 1
eco_i, SF,0
ecr_i, SF, 0

Neptune: private, S0, 0
smtp, S0, 0
Nnsp, S0,0
Login, S0, 0

When a test vector is tested, if it is near these training clusters the following method is used: The values of the selected features with symbolic values of the test vector are compared with all training clusters of an attack. If none of them are equal then it will be compared with other pairwise selected training clusters. If a training cluster is found then the Euclidean distance is calculated between the test vector and the mean of the training cluster. By calculating this, the selected features with continuous value are multiplied by (selected factor)*weight, if the weight is greater

than 1/feature weight. So the weight of the selected features is also used. In Figure 6.29 the training clusters of “normal-neptune” are shown.

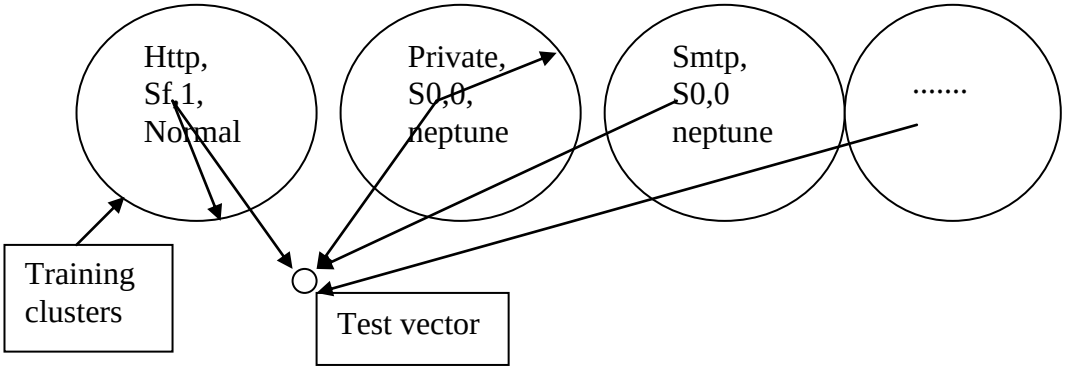

Figure 6.29 Finding the distance of a test vector to the temproary training clusters

By this calculation, the decision is that the test vector is either “normal” or “neptune”. But if the test vector is none of these actually, then the decision will be false. So we need an algorithm to be sure what type of cluster this is (here an attack or “normal”). If we can not name the test vector then we will name it “anomaly”.

6.1.2. Implementation Specification

The implementation is written in JAVA, so it is platform-independent and object-oriented.

An attack class knows the other known attack instances and the selected features, which distinguish this type of attack and the other known attacks, so it knows the training clusters between other attack instances. When a test vector should be tested, it is first tested with the class “Normal”. The class of “Normal” tests that, to which of the known attacks the test vector near is. So, in first step it can be found that the test vector is near to more than one attack.

So, we need a second step to be sure to which type of attack this is. The classes of every found attack run the same algorithm as in the first step. That means, they also test the test vector with the training clusters of the known attacks. And the found attacks by these tests are put in the “suspicious nearest neighbour attacks”. The maximum count of the attack in the set of the “suspicious nearest neighbour attacks” gives the decision what type of attack this test vector is.

In the training phase, every attack class is trained to create the training clusters and also the maximum radius of every cluster is also calculated. This ensure also that the found type of attack is in the cluster. If the found cluster is nearest to the test

attack, but the distance of the test attack to the median of the cluster is greater than [factor*(radius of the cluster)] , than this test attack is labeled as “anomaly”.

In Figure 6.30 the decision of test between classes is illustrated.

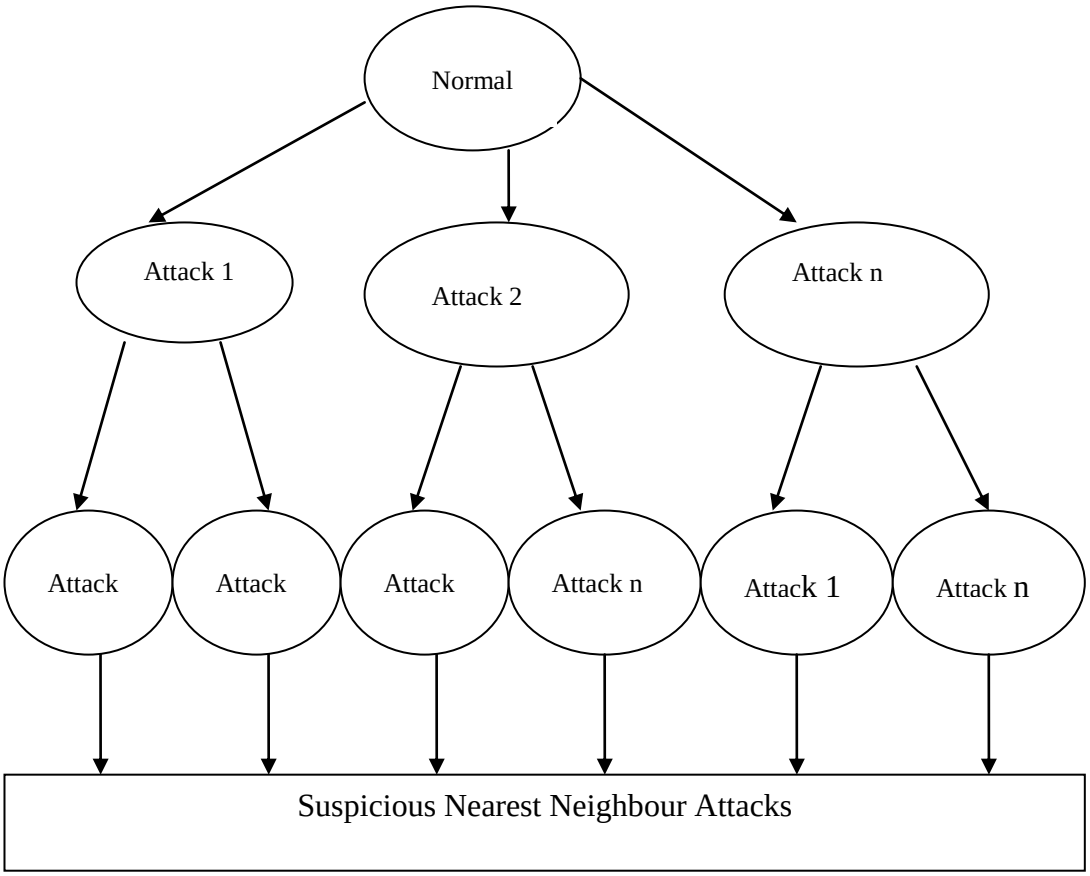

Figure 6.30 Test decision of suspicious Nearest Neighbour Attacks

6.2. The Algorithms Used in CLIDS

The implementation consists of training phase and the testing phase. The testing phase consists of two algorittms.

6.2.1. Training Phase Algorithm

The training phase algorithm which is explained before is shown in Figure 6.31.

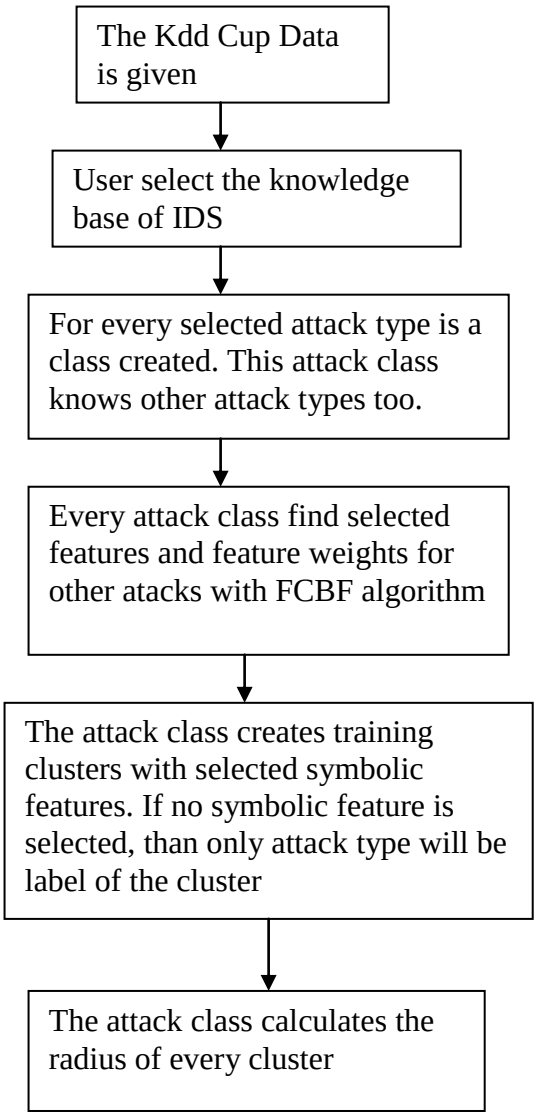

Figure 6.31 Algorithm of training phase

6.2.2. Testing Phase Algorithm

The testing phase algorithm consists of two parts. In the first part, it is found that , which of the training clusters of the known attacks of class “normal” to the test vector nearest is. In the second part, it is found that which type of attack this test vector is.

In Figure 6.32 and in Figure 6.33 the algorithm of the test part 1 is shown.

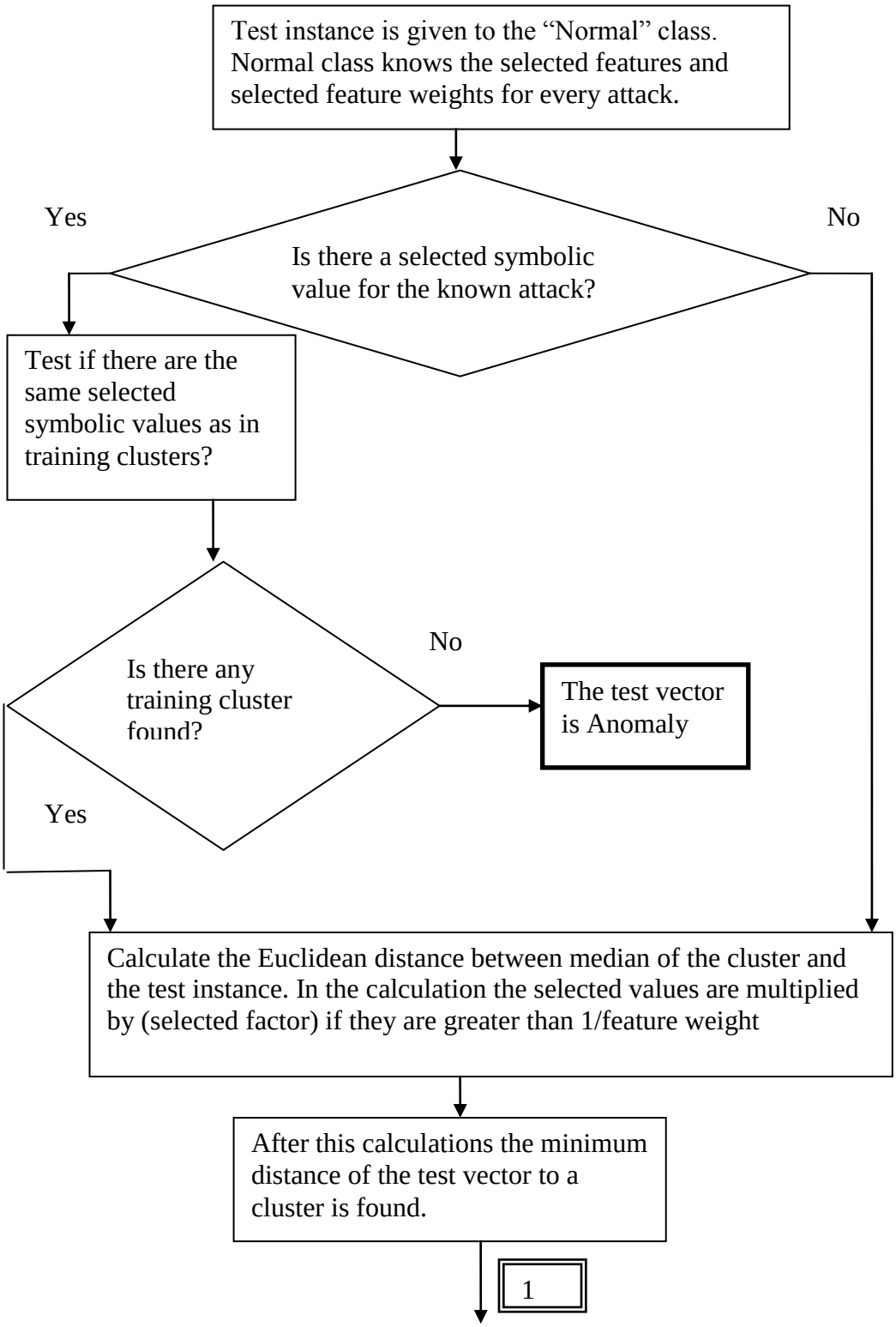

Figure 6.32 Algorithm of the part 1 of test phase

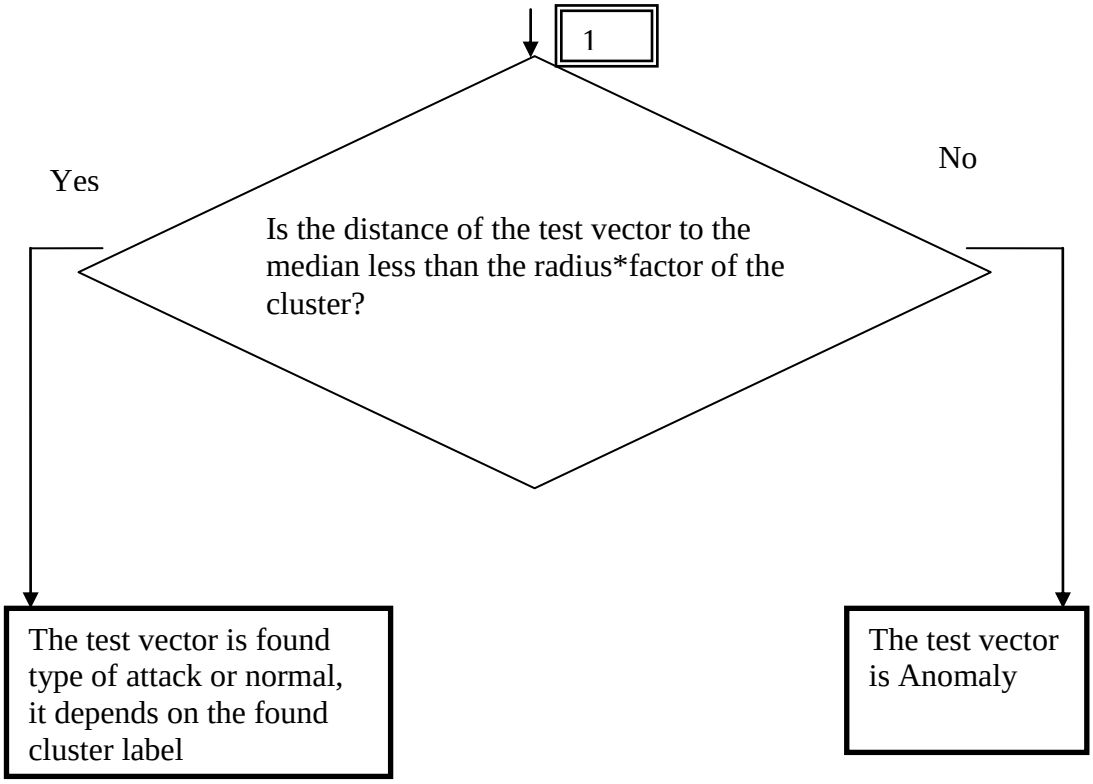

Figure 6.33 Algorithm of the part 1 of test phase (continued)

Every attack class knows every other type attacks, so if more than one known attack tells that the test attack is that type of attack, this must come to a result, which type of this test instance is. In algorithm of test phase 2 the individual test algorithm is explained as shown in Figure 6.34, in Figure 6.35 and in Figure 6.36.

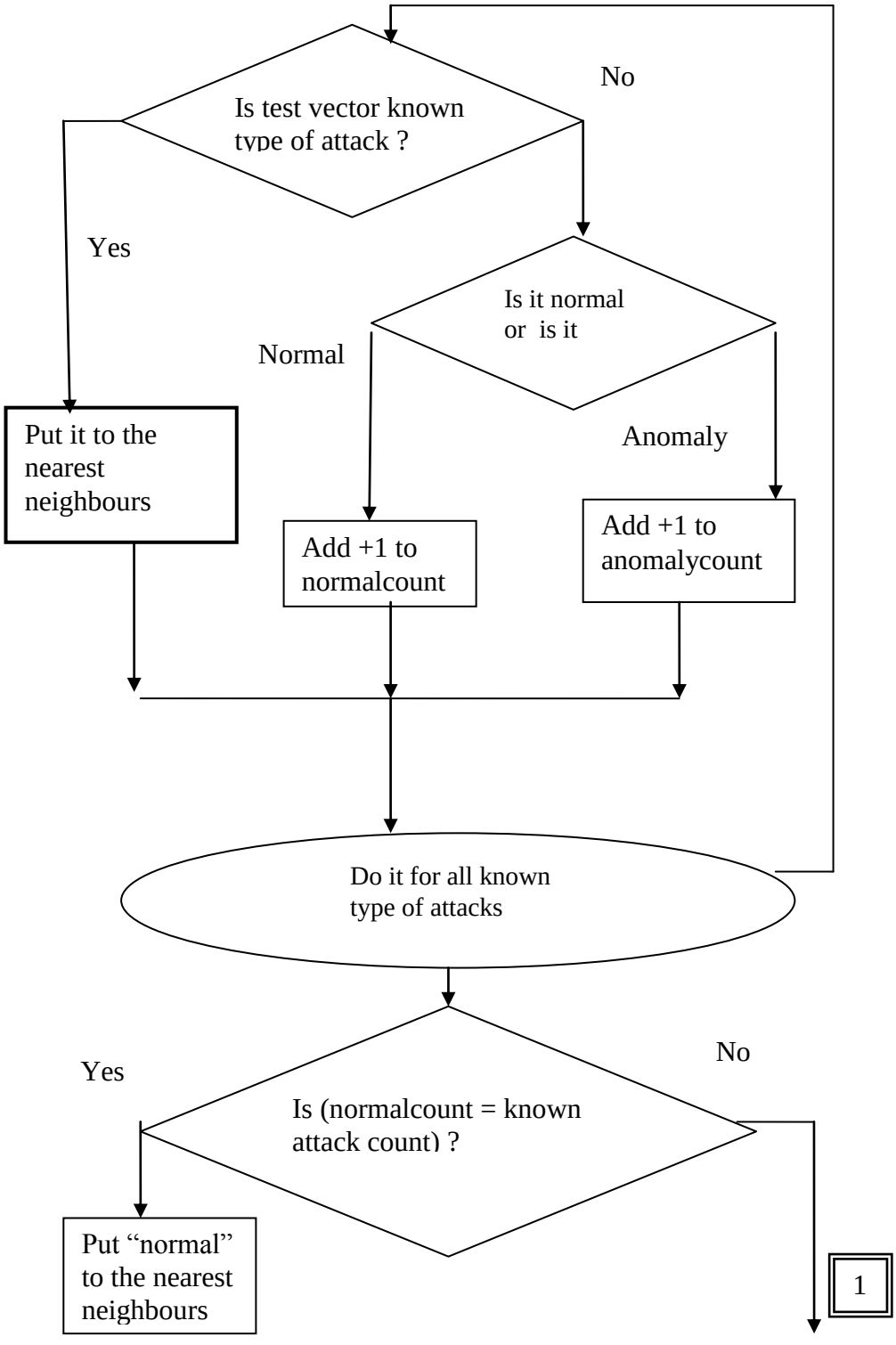

Figure 6.34 Algorithm of the part 2 of test phase

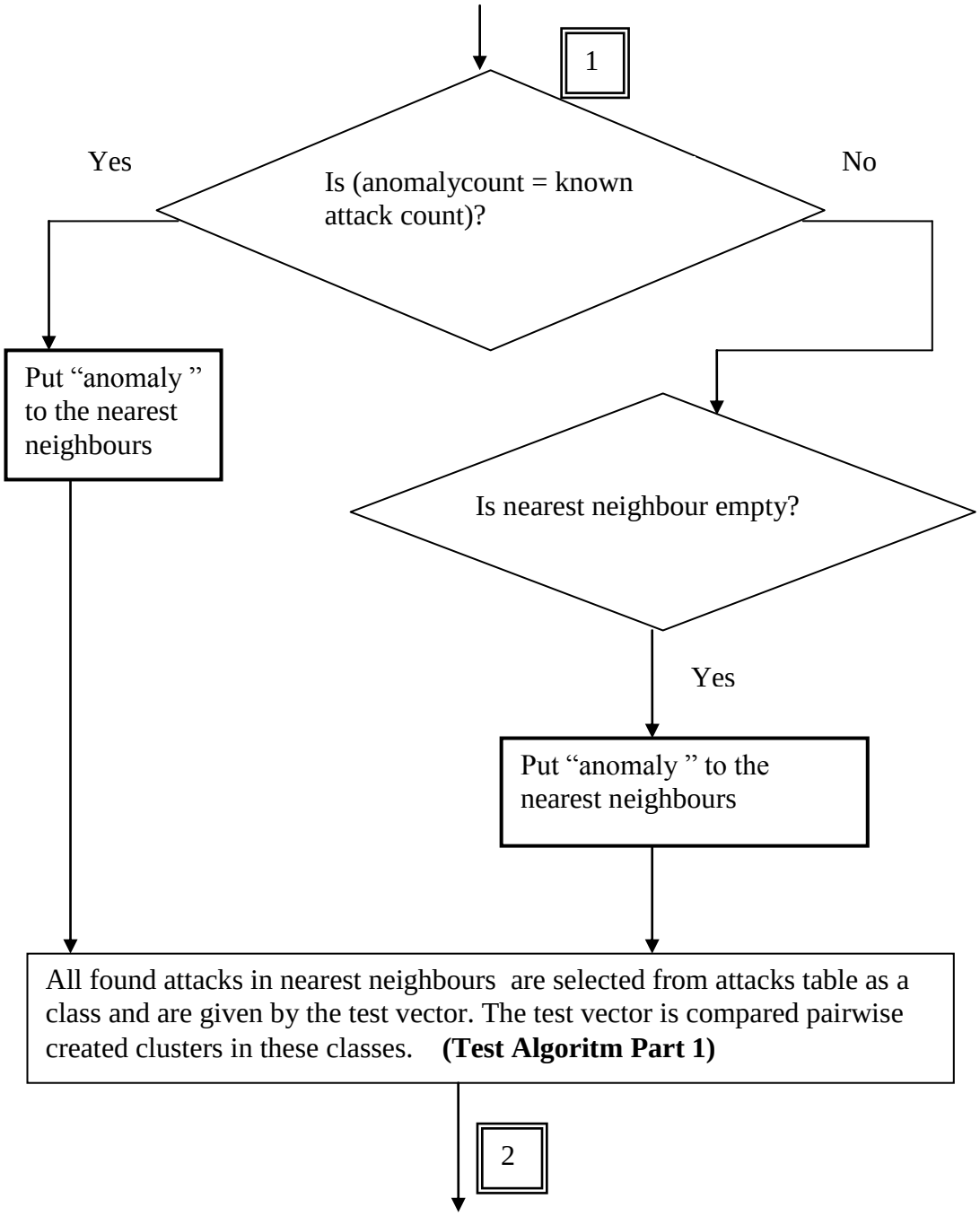

Figure 6.35 Algorithm of the part 2 of test phase (continued)

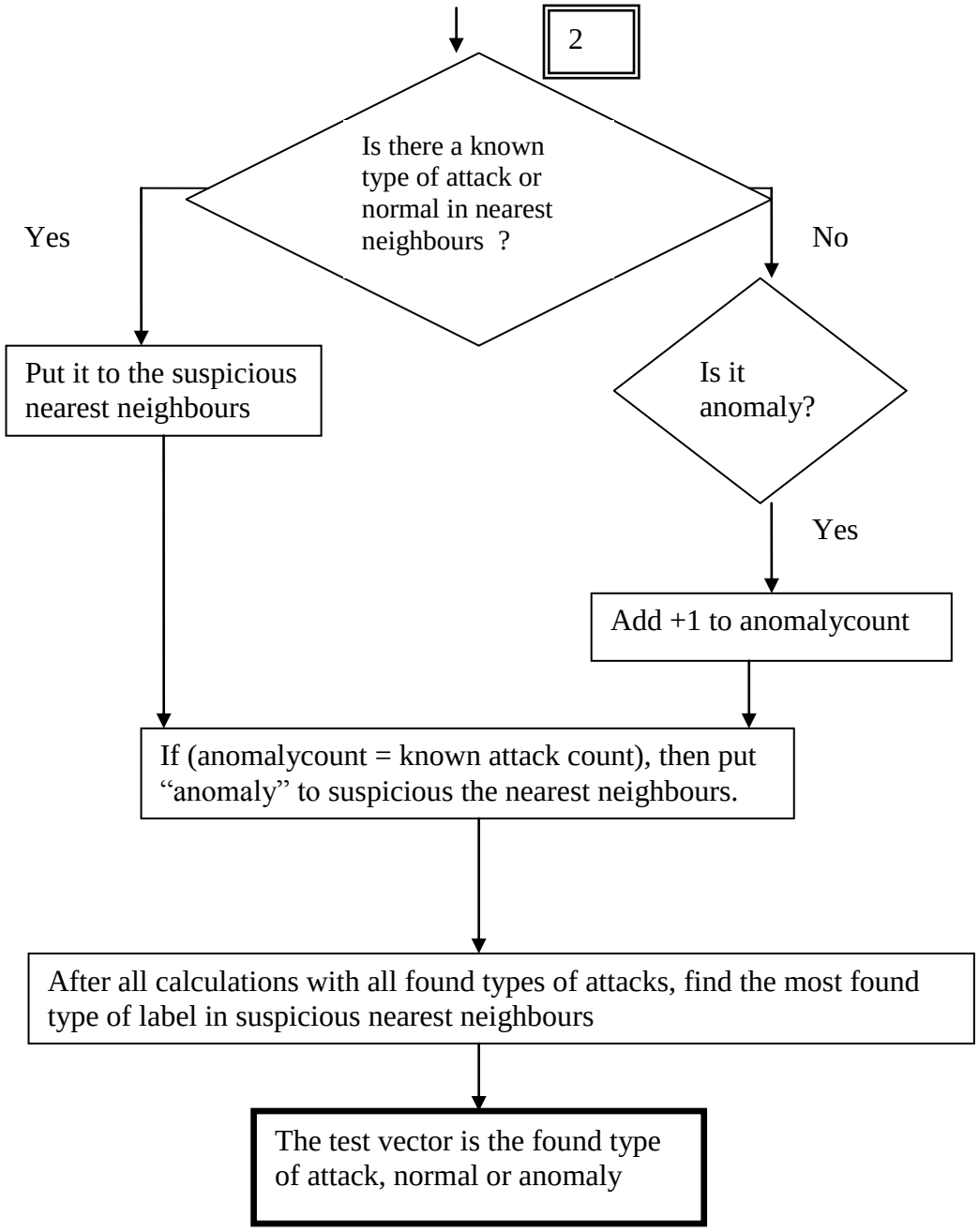

Figure 6.36 Algorithm of the part 2 of test phase (continued)

6.2.3. The implementation parameters of the program

The implementation parameters of the program are shown in Table 6.14.

Table 6.14 The implementation parameters, their description and the default values

Parameter Value	Description	Default Value
Feature Weight Calculation	The multiply factor of the selected continuous features by the Euclidean distance calculation	10
Continuous Feature Split Count	The split count of the continuous features to calculate the entropy value	10
Radius Factor	Radius = Radius * Radius Factor The difference between radius and the distance of the test vector to the median of the cluster	1.2

6.2.4. Training and Test Procedures

In all of tests, the instances in test data set are labeled individually and the results are compared with corrected data set, which are published in KDD Cup Results page.

In the training data, there are 22 types of attacks, which are:

Probe: ipsweep, nmap, portsweep, satan

DOS: back, land, neptune, pod, smurf, teardrop

U2R: buffer_overflow, loadmodule, perl, rootkit

R2L: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster

In the test data, there are 40 types of attacks, which are:

Probe: ipsweep, mscan, nmap, portsweep, saint, satan

DOS: apache2, back, land, neptune, pod, processtable, smurf, teardrop, udpstorm

U2R: buffer_overflow, httptunnel, loadmodule, mailbomb, multihop, perl, ps, rootkit, sqlattack , xterm

R2L: ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, spy, sendmail, snmpgetattack, snmpguess, worm, xlock, xsnoop, warezmaster

The test dataset contains 311,029 examples.

As seen above, there are types of attacks in test data set, which are not in training data set. These type of instances are labeled as “anomaly” in the CLIDS.

Because of the limits of memory and CPU of the computer the traning attack instances are maximum for 1000 selected.

Training attack counts (max:1000 intances):

Normal: 1000

Probe: ipsweep:1000, nmap: 231, portsweep:1000, satan:1000

DOS: back:1000, land:21, neptune:5000, pod:264, smurf:1000, teardrop:1000

U2R: buffer_overflow:30, loadmodule:9, perl:3, rootkit:10

R2L: ftp_write:8, guess_passwd:53, imap:11, multihop:7, phf:4, spy:2, warezclient:1000, warezmaster:20

As seen, the training counts are not equal because of the training file. So, the instances of back, teardrop, ipsweep, portsweep, warezclient, smurf, normal, neptune, satan are in feature selection algorithm for 1000 instances.

Then, in order to balance the counts of instances of pod, nmap, land, buffer_overflow, guess_passwd, loadmodule, perl, rootkit, ftp_write, imap, multihop, phf, spy and warezmaster attacks, these instances have been copied to reach 1000 instances.

Training Procedure:

CLIDS has 2 configuration files. In one of them, there are selected features and weights between the attacks, and in the other one there are cluster specifications. So, these files has been created once and used in every test .

In every test CLIDS is trained with DOS, Probe, U2R and R2L attacks.

Test Procedure:

There are 2 files, in one of them there are unlabeled instances, and in the other one there are labeled instances, which corresponds the other file.

These files are each 43 MB, so because of the physical limits of memory of the test computer, these files are split in files. In every test, the same 10 test files, in which 10000 test instances are, are used and the sum of counts of attacks in these test files is given in Table 6.15.

Table 6.15 Sum of counts of attack instances in test files

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 34448	smurf. 32529	portsweep. 347	buffer_overflow. 17	ftp_write. 3	mscan. 1053
	pod. 68	ipsweep. 306	loadmodule. 2	guess_passwd. 4366	saint. 0
	Teardrop. 6	satan. 1242	perl. 1	imap. 1	apache2. 596
	neptune. 10865	nmap. 44	rootkit. 13	multihop. 15	processtable. 506
	back. 99			phf. 2	udpstorm. 2
	land. 8			spy. 0	httptunnel. 143
				warezclient. 0	mailbomb. 4999
				warezmaster. 1595	ps. 16
					sqlattack. 1
					xterm. 13
					named. 10
					sendmail. 15
					snmpgetattack. 5919
					snmpguess. 0

					worm. 1
					xlock. 9
					xsnoop. 4

Some terminology used in test procedure is given below:

Anomaly for Attacks: The count of instances which have been labeled as anomaly instead of an attack name which they are really

False Positive for Attack :The count of intances which have been labeled as an attack type or anomaly instead of type “normal” which are really

False Positive for Anomaly: The count of intances which have been labeled as an anomaly instead of type “normal” which are really

Attack False: The count of instances which have been labeled as a type of an attack which are really another type of attack

False Negative for Attacks: The count of instances which have been labeled “normal” instead of an attack which are really

False Negative for Anomaly: The count of instances which have been labeled “normal” instead of an anomaly which are really.

6.3. Experimental Results with Min-Max Normalization

In these tests, the training and test instances are normalized with min-max normalization. In the trainig phase the minimum and maximum values for every feature in every attack class are found and written to the configuration file of CLIDS. Then in test phase, these values are used for normalization of test instances.

6.3.1. Test Results with Change of the Radius

6.3.1.1. Test Results with Radius Factor 1.2

In this test CLIDS is trained with DOS, Probe, U2R, R2L attacks and tested with radius factor 1.2. Feature weight is selected as 10 and continuous feature split count is selected as 10.

The sum of all true identifed test instances is given in Table 6.16.

Table 6.16 Sum of true indentified instances with Radius Factor 1.2

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 24975	smurf. 32370	ipsweep.0	buffer_overflow. 7	ftp_write. 0	mscan. 120
	neptune. 110	portsweep. 3	loadmodule. 1	guess_passwd. 2	saint. 12
	teardrop. 5	satan. 1134	perl. 0	imap. 0	apache2. 582
	pod. 58	nmap. 44	rootkit. 0	multihop. 0	processtable. 453
	back. 98			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 16
				warezclient. 0	mailbomb. 4944
				warezmaster. 498	ps. 4
					sqlattack. 0
					xterm. 3
					named. 3
					sendmail. 3
					snmpgetattack. 5911
					snmpguess. 0
					worm. 0
					xlock. 5
					xsnoop. 2

6.3.1.2. Calculated Results with Radius 1.2

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.17 with calculated results of percentages.

Table 6.17 Detection, False Negative and False Positive counts of test files with radius factor 1.2

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32649	0	29	%74.9	%0	%0.1
Probe	1939	1181	0	515	%60,9	%0	%26.5
U2R	33	8	0	40	%24,2	%0	%121,2
R2L	5982	501	0	688	%8,4	%0	%11.5
Normal	34448	24975	-	-	%72,5	-	-
Anomaly	14020	12060	4	8202	%85,9	%0	%58,5

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.18 with calculated results of percentages.

Table 6.18 Attack False and Anomaly for Attack counts with radius factor 1.2

Attack False Count	15199
Anomaly for Attack Count	3950
Attack False/ Attack Count	%29,5
Anomaly for Attack Count / Attack Count	%7,7

6.3.1.3. Test Results with Radius Factor 2.0

In this test CLIDS is trained with DOS, Probe, U2R, R2L attacks and tested with radius factor 2.0. Feature weight is selected as 10 and continuous feature split count is selected as 10.

The sum of all true identifed test instances is given in Table 6.19.

Table 6.19 Sum of true indentified instances with Radius Factor 2.0

Normal	DOS	Probe	U2R	R2L	Anomaly
normal.24975	smurf. 32400	ipsweep. 0	buffer_overflow. 7	ftp_write. 0	mscan. 120
	neptune. 110	portsweep. 3	loadmodule. 1	guess_passwd. 2	saint. 12
	teardrop. 5	satan. 1134	perl. 0	imap. 0	apache2. 582
	pod. 58	nmap. 44	rootkit. 0	multihop. 0	processtable. 453
	back. 98			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 16
				warezmaster. 498	mailbomb. 4944
				warezclient. 0	ps. 4
					sqlattack. 0
					xterm. 3
					named. 3
					sendmail. 3
					snmpgetattack. 5911
					snmpguess. 0
					worm. 0
					xlock. 5
					xsnoop. 2

6.3.1.4. Calculated Results with Radius 2.0

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.20 with calculated results of percentages.

Table 6.20 Detection, False Negative and False Positive counts of test files with radius factor 2.0

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative Rate	False Positive Rate
DOS	43575	32649	0	12	%74,9	%0	%0
Probe	1939	1181	0	567	%60.9	%0	%29,2
U2R	33	8	0	40	%24,2	%0	%121,1
R2L	5982	501	0	630	%8,4	%0	%11,1
Normal	34448	24975	-	-	%72,5	-	-
Anomaly	14020	12054	4	8202	%86,0	%0	%58,5

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.21 with calculated results of percentages.

Table 6.21 Attack False and Anomaly for Attack counts with radius factor 2.0

Attack False Count	15199
Anomaly for Attack Count	3950
Attack False/ Attack Count	%29,5
Anomaly for Attack Count / Attack Count	%7,7

6.3.1.5. Test Results with Radius Factor 1.0

In this test CLIDS is trained with DOS, Probe, U2R, R2L attacks and tested with radius factor 1.0. Feature weight is selected as 10 and continuous feature split count is selected as 10. The sum of all true identifed test instances is given in Table 6.22.

Table 6.22 Sum of true indentified instances with Radius Factor 1.0

Normal	DOS	Probe	U2R	R2L	Anomaly
normal.24572	smurf. 32370	ipsweep. 0	buffer_overflow. 4	ftp_write. 0	mscan. 184
	neptune. 60	portsweep. 11	loadmodule. 1	guess_passwd. 2	saint. 6
	teardrop. 5	satan. 1217	perl. 1	imap. 0	apache2. 582
	pod. 58	nmap. 44	rootkit. 0	multihop. 0	processtable. 453
	back. 98			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 16
				warezmaster. 588	mailbomb. 4944
				warezclient. 0	ps. 6
					sqlattack. 0
					xterm. 4
					named. 5
					sendmail. 6
					snmpgetattack. 5911
					snmpguess. 0
					worm. 0
					xlock. 7
					xsnoop. 2

6.3.1.6. Calculated Results with Radius 1.0

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.23 with calculated results of percentages.

Table 6.23 Detection, False Negative and False Positive counts of test files with radius factor 1.0

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative Rate	False Positive Rate
DOS	43575	32599	0	30	%74,8	%0	%0,1
Probe	1939	1272	0	482	%65,6	%0	%24,9
U2R	33	6	0	376	%18,2	%0	%1139,4
R2L	5982	591	0	284	%9,9	%0	%4,7
Normal	34448	24572	-	-	%71,3	-	-
Anomaly	14020	12134	2	8644	%86,5	%0	%61,5

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.24 with calculated results of percentages.

Table 6.24 Attack False and Anomaly for Attack counts with radius factor 1.0

Attack False Count	14707
Anomaly for Attack Count	4241
Attack False/ Attack Count	%28,5
Anomaly for Attack Count / Attack Count	%8,2

6.3.1.7. Test Results with Radius Factor 0.8

In this test CLIDS is trained with DOS, Probe, U2R, R2L attacks and tested with radius factor 0.8. Feature weight is selected as 10 and continuous feature split count is selected as 10. The sum of all true identifed test instances is given in Table 6.25.

Table 6.25 Sum of true indentified instances with Radius Factor 0.8

Normal	DOS	Probe	U2R	R2L	Anomaly
normal.20330	smurf. 32370	ipsweep. 0	buffer_overflow. 8	ftp_write. 0	mscan. 357
	neptune. 22	portsweep. 12	loadmodule. 0	guess_passwd. 6	saint. 13
	teardrop. 4	satan. 1235	perl. 1	imap. 0	apache2. 582
	pod. 58	nmap. 44	rootkit. 4	multihop. 0	processtable. 453
	back. 97			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 16
				warezmaster. 34	mailbomb. 4989
				warezclient. 0	ps. 10
					sqlattack. 0
					xterm. 8
					named. 8
					sendmail. 10
					snmpgetattack. 5917
					snmpguess. 0
					worm. 1
					xlock. 7
					xsnoop. 2

6.3.1.8. Calculated Results with Radius 0.8

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.26 with calculated results of percentages.

Table 6.26 Detection, False Negative and False Positive counts of test files with radius factor 0.8

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative Rate	False Positive Rate
DOS	43575	32559	0	24	%74,8	%0	%0.1
Probe	1939	1291	0	431	%66,6	%0	%22,2
U2R	33	13	0	405	%39,4	%0	%1227,3
R2L	5982	41	0	139	%0,7	%0	%2.3
Normal	34448	20330	-	-	%59,0	-	-
Anomaly	14020	12375	1	13118	%88,3	%0	%93,6

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.27 with calculated results of percentages.

Table 6.27 Attack False and Anomaly for Attack counts with radius factor 0.8

Attack False Count	14830
Anomaly for Attack Count	4442
Attack False/ Attack Count	%28,8
Anomaly for Attack Count	%8,6

6.3.1.9. Graphical Results for Rates with Change of the Radius Factor

In Figure 6.37 Rates for attacks DOS, in Figure 6.38 Rates for attacks PROBE**,** in Figure 6.39 Rates for attacks U2R, in Figure 6.40 Rates for attacks R2L, in Figure 6.41 Rates for attacks Anomaly, in Figure 6.42 Detection Rate for Normal are given.

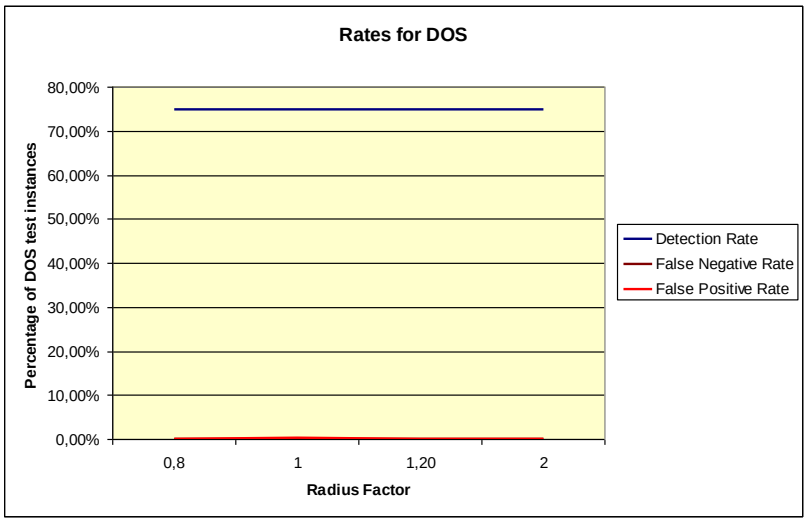

Figure 6.37 Rates for attacks DOS with change of the radius factor

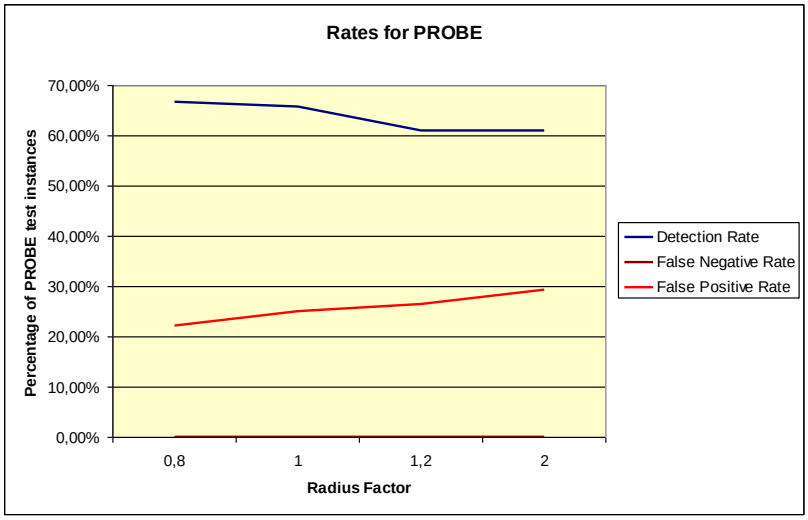

Figure 6.38 Rates for attacks PROBE with change of the radius factor

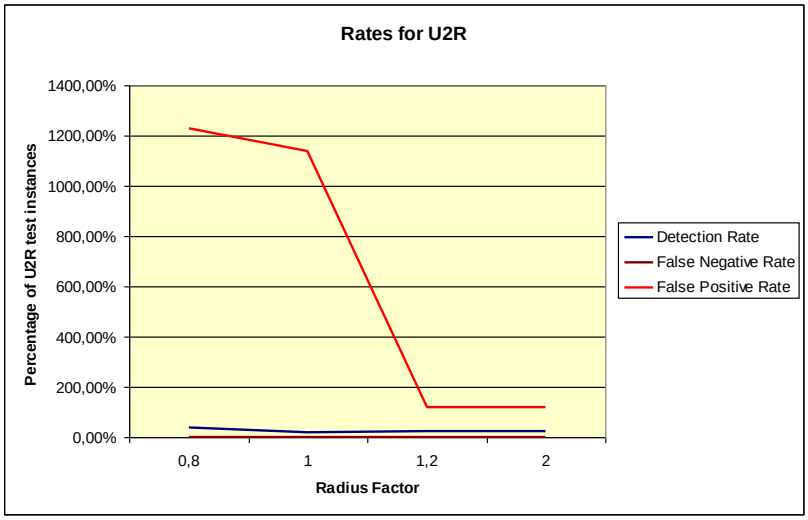

Figure 6.39 Rates for attacks U2R with change of the radius factor

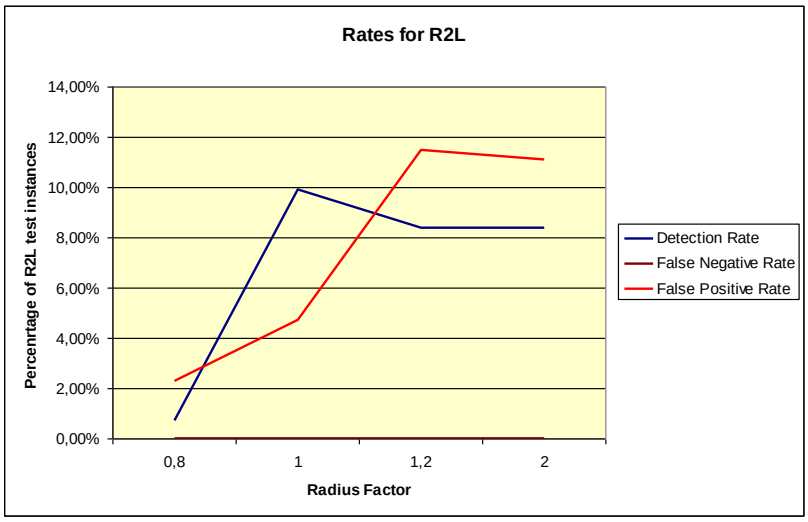

Figure 6.40 Rates for attacks R2L with change of the radius factor

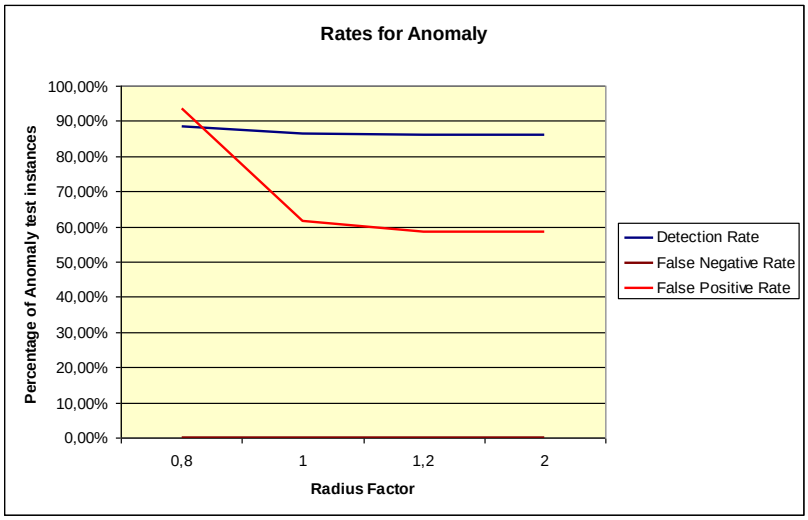

Figure 6.41 Rates for attacks Anomaly with change of the radius factor

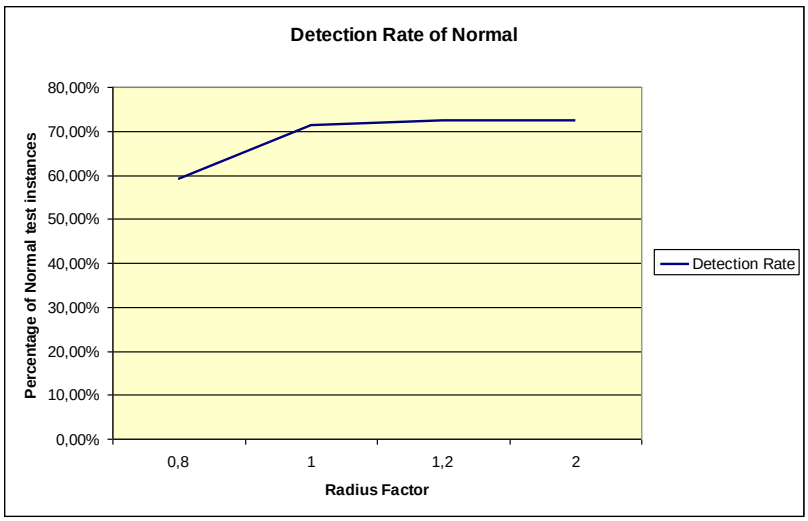

Figure 6.42 Detection Rate for Normal with change of the radius factor

6.3.2. Test Results with Change of the Continuous Feature Split Count

6.3.2.1. Test Results with Continuous Feature Split Count 100

Continuous features are split by a factor in order to get entropy for using FCBF algorithm. So default value of this factor is 10. In this test this value is selected as 100, to show the difference of rates. The radius factor is 1.2, the feature weight factor is 10. The sum of all true identifed test instances is given in Table 6.28.

Table 6.28 Sum of true indentified instances with Feature Split Count 100

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 25069	smurf. 32337	ipsweep. 0	buffer_overflow. 4	ftp_write. 0	mscan. 0
	neptune. 38	portsweep. 0	loadmodule. 0	guess_passwd. 0	saint. 13
	teardrop. 3	satan. 1231	perl. 0	imap. 0	apache2. 554
	pod. 60	nmap. 44	rootkit. 4	multihop. 0	processtable. 499
	back. 99			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 15
				warezmaster. 362	mailbomb. 4986
				warezclient. 0	ps. 4
					sqlattack. 0
					xterm. 6
					named. 4
					sendmail. 6
					snmpgetattack. 5909
					snmpguess. 0
					worm. 0
					xlock. 6
					xsnoop. 2

6.3.2.2. Calculated Results with Continous Split Factor 100

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.29 with calculated results of percentages.

Table 6.29 Detection, False Negative and False Positive counts of test files with continuous split factor 100

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32545	0	20	%74,7	%0	%0
Probe	1939	1275	0	446	%65,8	%0	%23,0
U2R	33	8	0	52	%24,2	%0	%157,6
R2L	5982	363	0	606	%6,1	%0	%10,1
Normal	34448	25069	-	-	%72,8	-	-
Anomaly	14020	12018	7	8227	%85,7	%0	%58,7

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.30 with calculated results of percentage

Table 6.30 Attack False and Anomaly for Attack counts with continuous split factor 100

Attack False Count	15297
Anomaly for Attack Count	4039
Attack False/ Attack Count	%29,7
Anomaly for Attack Count / Attack Count	%7,8

6.3.2.3. Graphical Results for Rates with Change of the Continous Split Factor

In Figure 6.43 Rates for attacks DOS, in Figure 6.44 Rates for attacks PROBE**,** in Figure 6.45 Rates for attacks U2R, in Figure 6.46 Rates for attacks R2L, in Figure 6.47 Rates for attacks Anomaly, in Figure 6.48 Detection Rate for Normal are given.

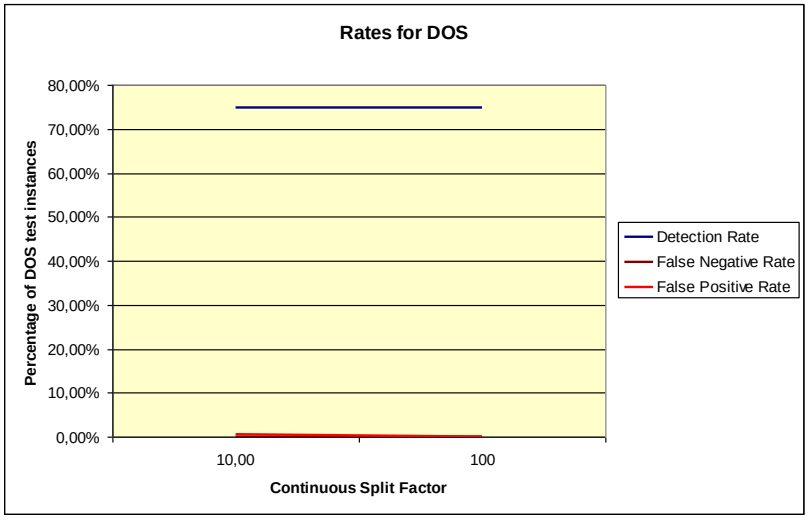

Figure 6.43 Rates for attacks DOS with change of the continous split factor

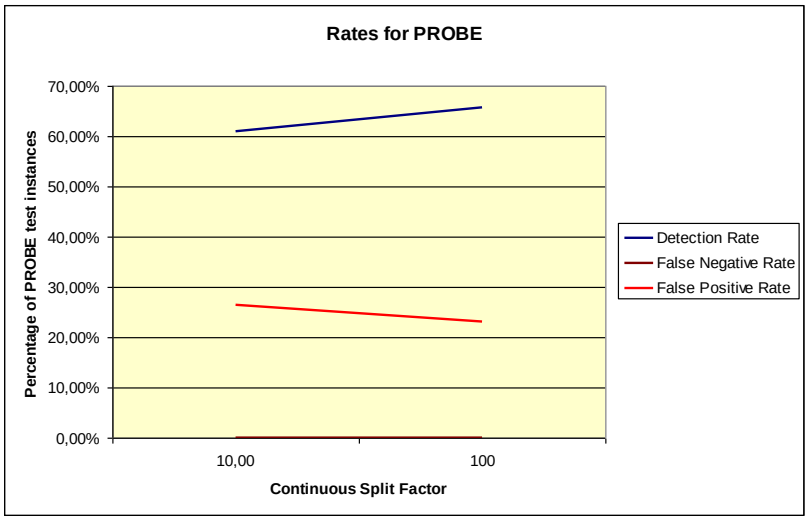

Figure 6.44 Rates for attacks PROBE with change of the continous split factor

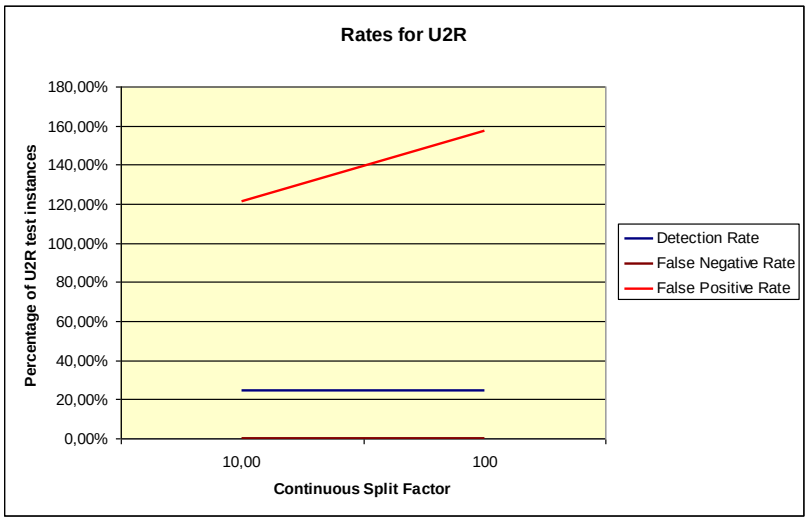

Figure 6.45 Rates for attacks U2R with change of the continous split factor

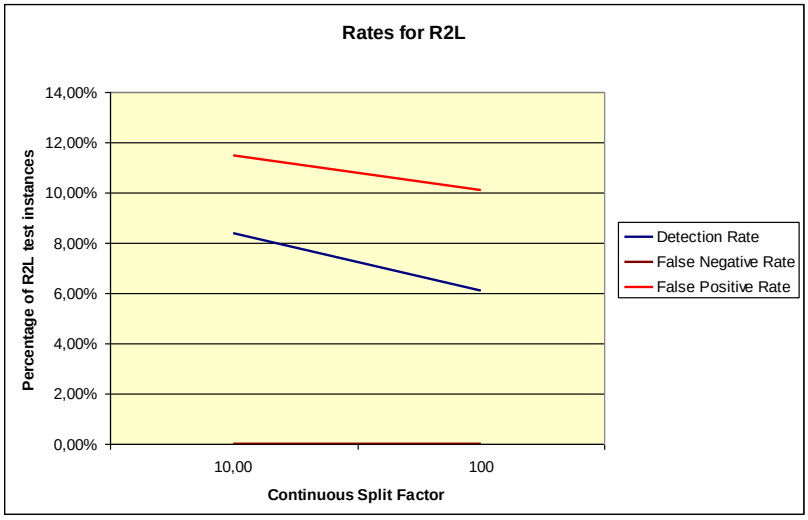

Figure 6.46 Rates for attacks with change of the continous split factor

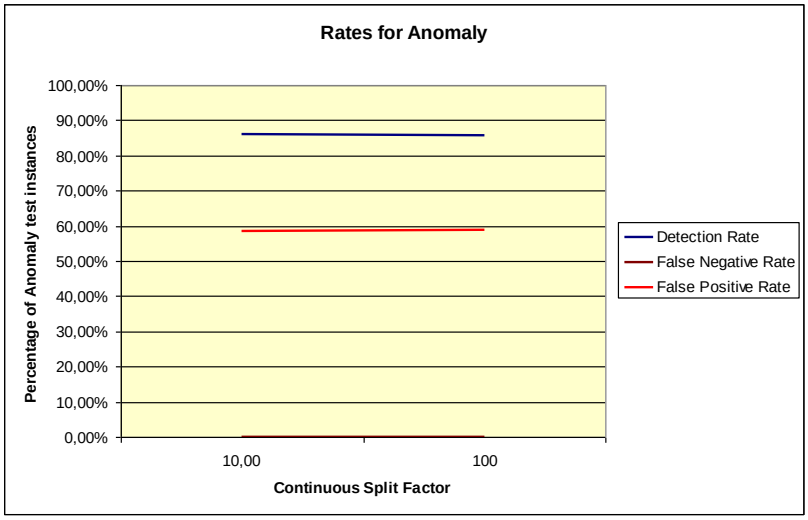

Figure 6.47 Rates for attacks Anomaly with change of the continous split factor

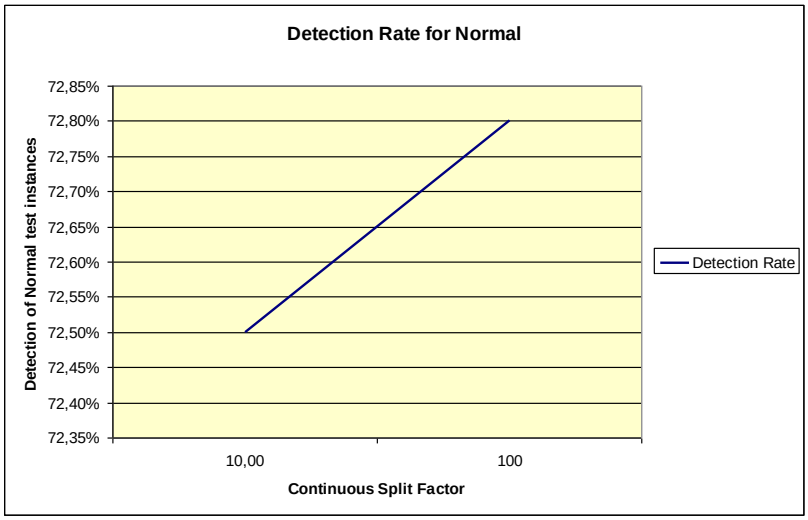

Figure 6.48 Detection Rate for Normal with change of the continous split factor

6.3.3. Test Results with Change of the Feature Weight Factor

6.3.3.1. Test Results with Feature Weight Factor 100

In the tests before, the feature weight factor used to calculate the Euclidean distance between instances was selected as default 10. So, in this test , the feature weight factor is selected as 100. The radius factor is 1.2, the continuous split factor is 10. The sum of all true identifed test instances is given in Table 6.31.

Table 6.31 Sum of true indentified instances with Feature Weight Factor 100

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 1775	smurf. 32260	ipsweep. 249	buffer_overflow. 5	ftp_write. 0	mscan. 259
	neptune. 254	portsweep. 205	loadmodule. 0	guess_passwd. 15	saint. 6
	teardrop. 2	satan. 927	perl. 0	imap. 0	apache2. 596
	pod. 59	nmap. 44	rootkit. 4	multihop. 0	processtable. 504
	back. 80			phf. 1	udpstorm. 2
	land. 7			spy. 0	httptunnel. 125
				warezmaster. 27	mailbomb. 4999
				warezclient. 0	ps. 4
					sqlattack. 0
					xterm. 5
					named. 7
					sendmail. 7
					snmpgetattack. 5911
					snmpguess. 0
					worm. 0
					xlock. 7
					xsnoop. 2

6.3.3.2. Calculated Results with Feature Weight Factor 100

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.32 with calculated results of percentages.

Table 6.32 Detection, False Negative and False Positive counts of test files with feature weight factor 100

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32662	0	9	%75,0	%0	%0
Probe	1939	1425	0	276	%73,5	%0	%14,2
U2R	33	9	0	27	%27,2	%0	%81,8
R2L	5982	43	0	582	%7,2	%0	%9,7
Normal	34448	1775	-	-	%5,2	-	-
Anomaly	14020	12495	0	31747	%89,1	%0	%226,4

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.33 with calculated results of percentages.

Table 6.33 Attack False and Anomaly for Attack counts with feature weight factor 100

Attack False Count	14362
Anomaly for Attack Count	4556
Attack False/ Attack Count	%28
Anomaly for Attack Count / Attack Count	%9

6.3.3.3. Test Results with Feature Weight Factor 1

In the tests before, the feature weight factor used to calculate the Euclidean distance between instances was selected as default 10. So, in this test , the feature

weight factor is selected as 1. The radius factor is 1.2, the continuous split factor is 10. The sum of all true identifed test instances is given in Table 6.34.

Table 6.34 Sum of true indentified instances with Feature Weight Factor 1

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 23526	smurf. 32387	ipsweep. 47	buffer_overflow. 1	ftp_write. 0	mscan. 24
	neptune. 2	portsweep. 4	loadmodule. 0	guess_passwd. 0	saint. 6
	teardrop. 2	satan. 1068	perl. 1	imap. 0	apache2. 511
	pod. 12	nmap. 44	rootkit. 0	multihop. 0	processtable. 390
	back. 98			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 14
				warezmaster. 526	mailbomb. 4999
				warezclient. 0	ps. 1
					sqlattack. 0
					xterm. 1
					named.3
					sendmail. 4
					snmpgetattack.1
					snmpguess. 0
					worm. 0
					xlock.4
					xsnoop. 2

6.3.3.4. Calculated Results with Feature Weight Factor 1

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.35 with calculated results of percentages.

Table 6.35 Detection, False Negative and False Positive counts of test files with feature weight factor 1

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32507	0	97	%74,6	%0	%0,2
Probe	1939	1163	0	9178	%60,0	%0	%473,3
U2R	33	4	0	21	%12,1	%0	%63,6
R2L	5982	527	0	574	%8,8	%0	%9,6
Normal	34448	23526	-	-	%68,3	-	-
Anomaly	14020	963	0	680	%6,9	%0	%4,9

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.36 with calculated results of percentages.

Table 6.36 Attack False and Anomaly for Attack counts with feature weight factor 1

Attack False Count	26663
Anomaly for Attack Count	3724
Attack False/ Attack Count	%51,7
Anomaly for Attack Count / Attack Count	%7,2

6.3.3.5. Graphical Results for Rates with Change of the Feature Weight Factor

In Figure 6.49 Rates for attacks DOS, in Figure 6.50 Rates for attacks PROBE**,** in Figure 6.51 Rates for attacks U2R, in Figure 6.52 Rates for attacks R2L, in Figure 6.53 Rates for attacks Anomaly, in Figure 6.54 Detection Rate for Normal are given.

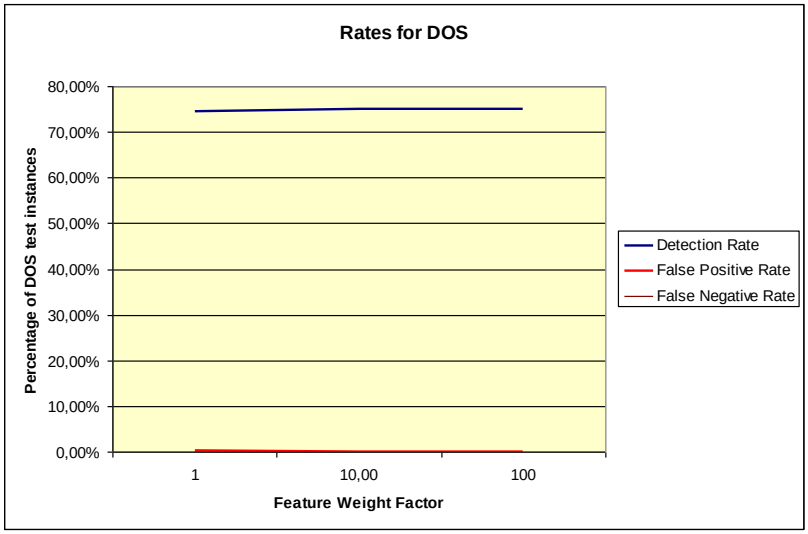

Figure 6.49 Rates for attacks DOS with change of the feature weight factor

Figure 6.50 Rates for attacks PROBE with change of the feature weight factor

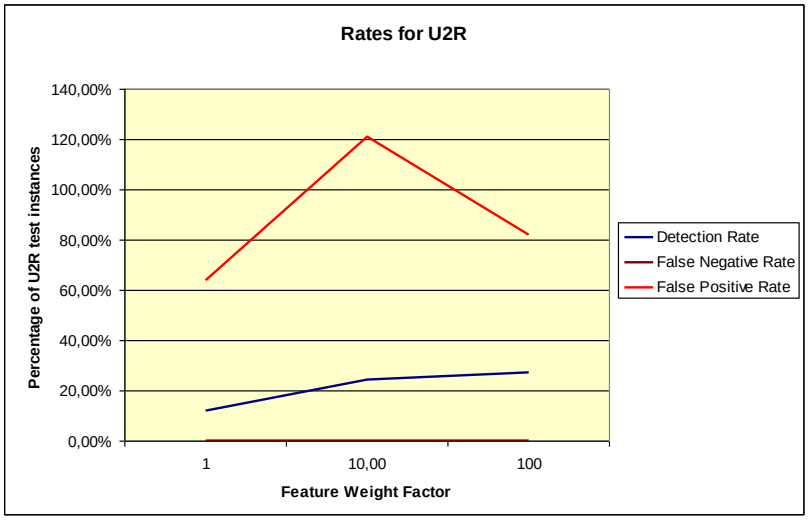

Figure 6.51 Rates for attacks U2R with change of the feature weight factor

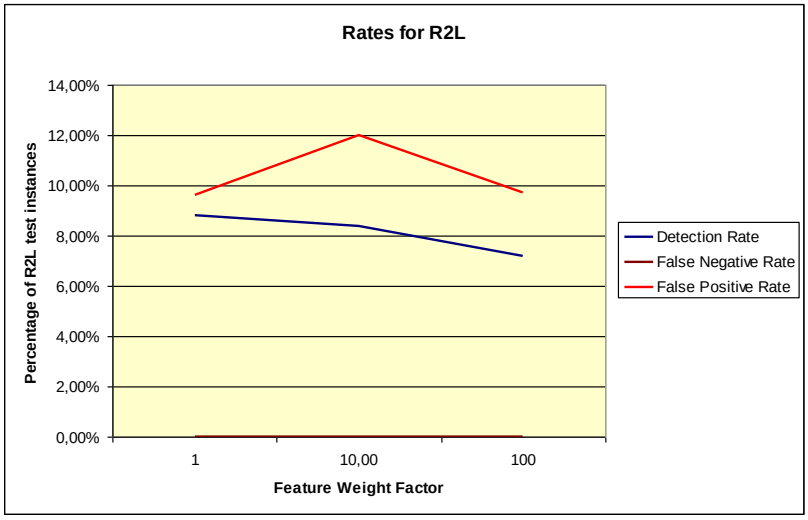

Figure 6.52 Rates for attacks R2L with change of the feature weight factor

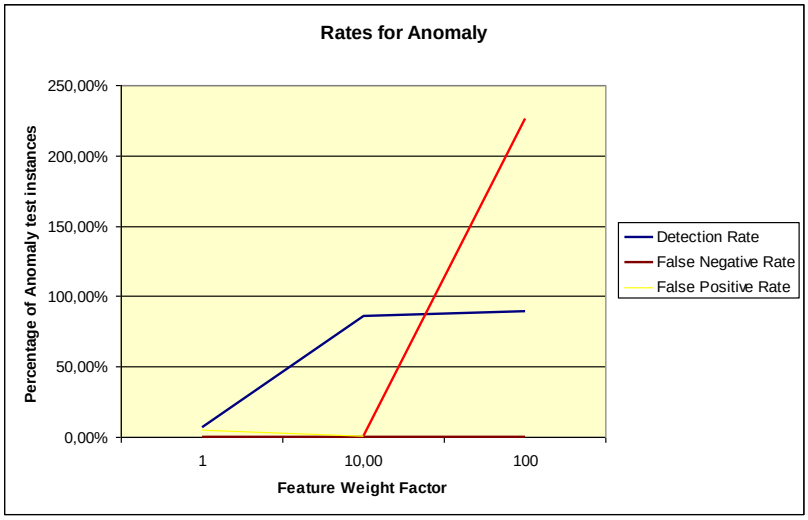

Figure 6.53 Rates for attacks Anomaly with change of the feature weight factor

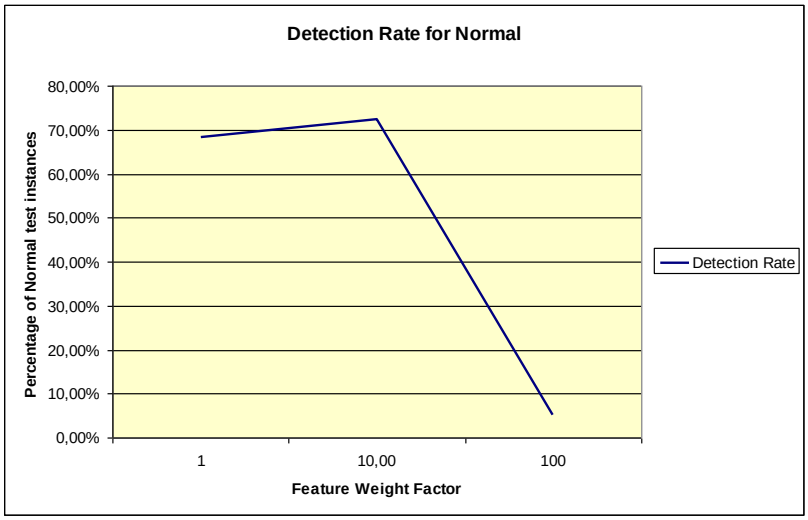

Figure 6.54 Detection Rate for Normal with change of the feature weight factor

6.4. Experimental Results with Zero-Mean Normalization

In these tests, the training phase the mean and the standard deviaton for every feature in every attack class are found and written to the configuration file of CLIDS. Then in test phase, these values are used for normalization of test instances.

6.4.1. Test Results with Change of the Radius

6.4.1.1. Test Results with Radius Factor 1.2

The radius factor is selected as 1.2, the feature weight factor is 10, the continuous split factor is 10. The sum of all true identifed test instances is given in Table 6.37.

Table 6.37 Sum of true indentified instances with Radius Factor 1.2 with Zero-Mean Normalization

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 109	smurf. 32398	ipsweep. 47	buffer_overflow. 4	ftp_write. 0	mscan. 44
	neptune. 49	portsweep. 121	loadmodule. 0	guess_passwd. 20	saint. 22
	teardrop. 6	satan. 1153	perl. 1	imap. 0	apache2. 339
	pod. 60	nmap. 44	rootkit. 0	multihop. 0	processtable. 276
	back. 97			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 13
				warezmaster. 414	mailbomb. 4897
				warezclient. 0	ps. 6
					sqlattack. 1
					xterm. 5
					named.3
					sendmail. 3
					snmpgetattack. 5911
					snmpguess. 0

					worm. 0
					xlock. 7
					xsnoop. 2

6.4.1.2. Calculated Results with Radius Factor 1.2

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.38 with calculated results of percentages.

Table 6.38 Detection, False Negative and False Positive counts of test files with Radius Factor 1.2 with Zero-Mean Normalization

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32618	0	81	%74,9	%0	%0,2
Probe	1939	1318	0	225	%68,0	%0	%11,6
U2R	33	5	0	26	%15,2	%0	%78,8
R2L	5982	435	0	469	%7,3	%0	%7,8
Normal	34448	109	-	-	%0,3	-	-
Anomaly	14020	11531	0	32774	%82,2	%0	%233,8

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.39 with calculated results of percentages.

Table 6.39 Attack False and Anomaly for Attack counts with Radius Factor 1.2 with Zero-Mean Normalization

Attack False Count	15660
Anomaly for Attack Count	3985
Attack False/ Attack Count	%30,4
Anomaly for Attack Count / Attack Count	%7,7

6.4.1.3. Test Results with Radius Factor 2

The radius factor is selected as 2, the feature weight factor is 10, the continuous split factor is 10. The sum of all true identifed test instances is given in Table 6.40.

Table 6.40 Sum of true indentified instances with Radius Factor 2 with Zero-Mean Normalization

Normal	DOS	Probe	U2R	R2L	Anomaly
normal. 119	smurf. 32306	ipsweep. 47	buffer_overflow. 1	ftp_write. 0	mscan. 8
	neptune. 49	portsweep. 115	loadmodule. 0	guess_passwd. 5	saint. 21
	teardrop. 6	satan. 1047	perl. 1	imap. 0	apache2. 331
	pod. 60	nmap. 44	rootkit. 0	multihop. 0	processtable. 276
	back. 97			phf. 1	udpstorm. 2
	land. 8			spy. 0	httptunnel. 3
				warezmaster. 607	mailbomb. 421
				warezclient. 0	ps. 6
					sqlattack. 1
					xterm. 3
					named.2
					sendmail. 2
					snmpgetattack. 58
					snmpguess. 0
					worm. 0
					xlock. 5
					xsnoop. 1

6.4.1.4. Calculated Results with Radius Factor 2

The sum of counts of test instances, grouped by attack types, if they are true identifed or if they are “false negative” found or if they are “false positive” found, is given in Table 6.41 with calculated results of percentages.

Table 6.41 Detection, False Negative and False Positive counts of test files with Radius Factor 2 with Zero-Mean Normalization

Attack Type	Counts Tested	True Count	False Negative Count	False Positive Count	Detection Rate	False Negative	False Positive Rate
DOS	43575	32477	0	150	%74,5	%0	%0,3
Probe	1939	1206	0	604	%62,2	%0	%31,2
U2R	33	1	0	131	%3,0	%0	%397,0
R2L	5982	613	0	18112	%10,3	%0	%302,8
Normal	34448	119	-	-	%0,3	-	-
Anomaly	14020	1139	0	15131	%8,1	%0	%108,0

The sum of test instances, which are found as “false attack” and which are found “anomaly for attack” is given in Table 6.42 with calculated results of percentages.

Table 6.42 Attack False and Anomaly for Attack counts with with Radius Factor 2 with Zero-Mean Normalization

Attack False Count	29563
Anomaly for Attack Count	553
Attack False/ Attack Count	%57
Anomaly for Attack Count / Attack Count	%1,1

6.4.1.5. Graphical Results for Rates with Change of the Radius Factor

In Figure 6.55 Rates for attacks DOS, in Figure 6.56 Rates for attacks PROBE**,** in Figure 6.57 Rates for attacks U2R, in Figure 6.58 Rates for attacks R2L, in Figure 6.59 Rates for attacks Anomaly, in Figure 6.60 Detection Rate for Normal are given.

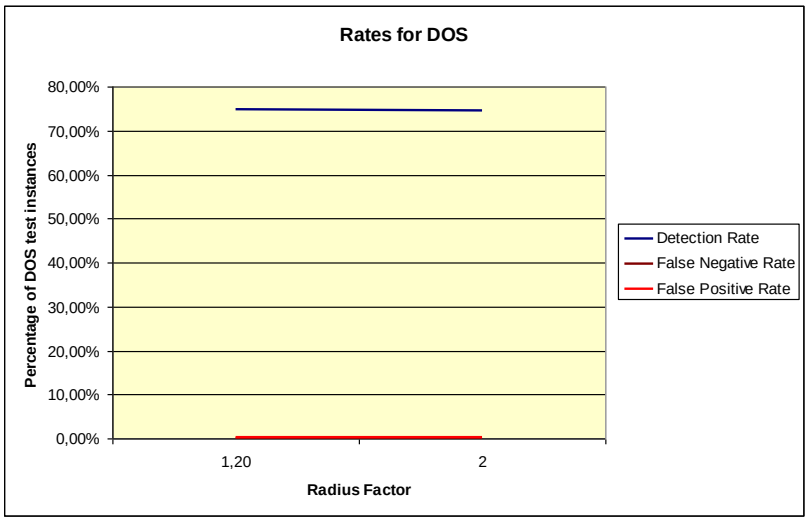

Figure 6.55 Rates for attacks DOS with change of the radius factor with Zero-Mean Norm.

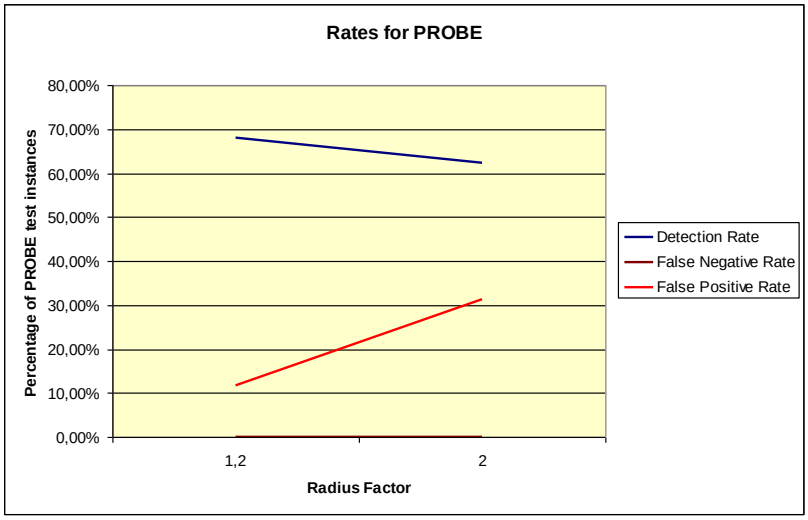

Figure 6.56 Rates for attacks PROBE with change of the radius factor with Zero-Mean Norm.

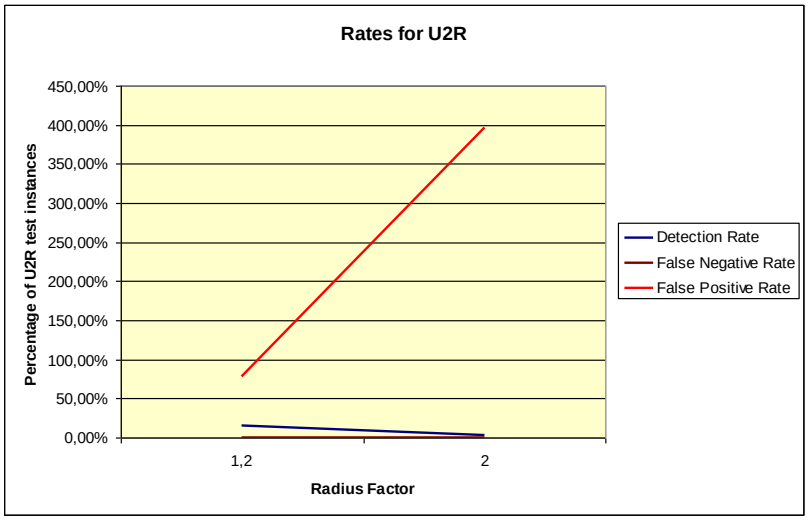

Figure 6.57 Rates for attacks U2R with change of the radius factor with Zero-Mean Norm.

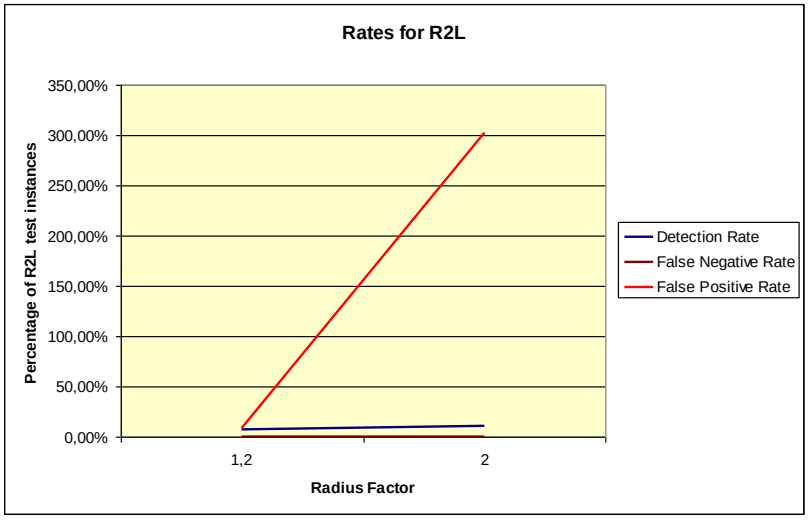

Figure 6.58 Rates for attacks R2L with change of the radius factor with Zero-Mean Norm.

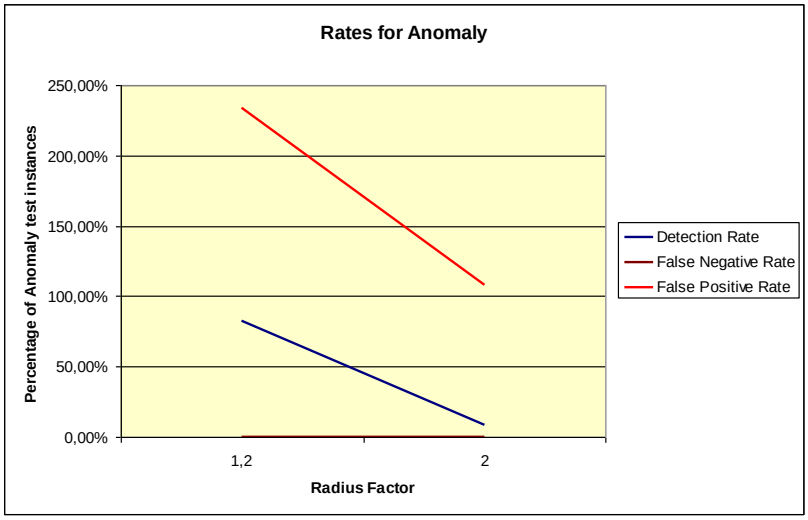

Figure 6.59 Rates for attacks Anomaly with change of the radius factor with Zero-Mean Norm.

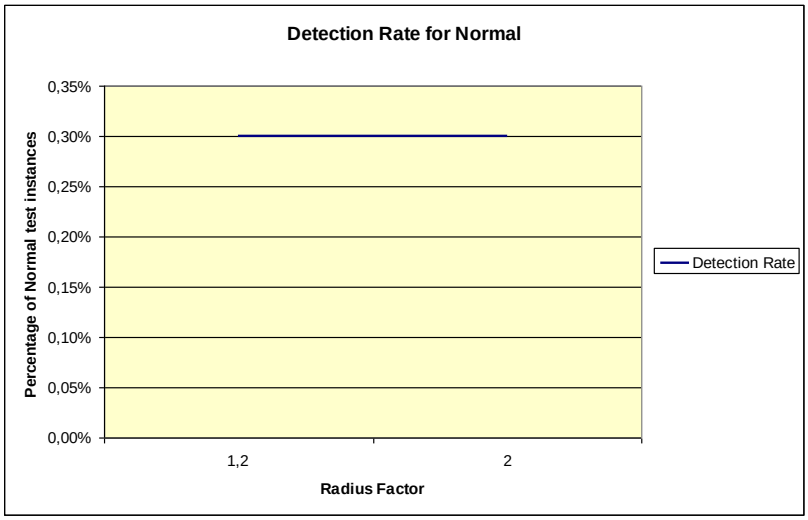

Figure 6.60 Rates for attacks Normal with change of the radius factor with Zero-Mean Norm.

7. CONCLUSION and FUTURE WORK

CLIDS (Cluster Based Intrusion Detection) gives a new methodology in intrusion detection , which is pattern classification. Until today many methodologies are used in intrusion detection, which depend mostly to match the database of intrusion detection system. If the packet or log matches the rules in database then it is defined as an intrusion. In this methodology new rules should be added manually.

Then, to overcome not to add the rules manually, the database mining methodology is used. So, an algorithm is given by the training instances of the intrusions and normal patterns, and the algorithm gives the rules automatically.

By all of these methodologies, in which there are no machine learning algorithm, it is impossible to identify new intrusions, which are not defined as an intrusion in the rules database of the intrusion detection system.

One methodology to identify the “anomalies”, the patterns which do not suit the rules of the IDS, is the pattern classification. Pattern classification algorithms are used in many areas, including genetics, speech recognition and fingerprint identification.

The pattern classification algorithms can be grouped mainly in two areas: Supervised classification, unsupervised classification. Supervised classification algorithms are trained first with training instances. So, the test instances can be labeled in this way. But the unsupervised classification algorithms are not trained. They are given by the test instances, and the algorithm makes clusters in these instances, but it can not label them, because it doesn’t know.

In this thesis, two problems are tried to solve: First: The IDS should label the test intances, whatever the instance is. Second: The IDS should find the anomalies.

To produce intrusion data is a very hard and expensive work. So, the KDD Cup Data 99, the data which are produced in 1998 by DARPA for using in Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining, which are used also in many works, are used in this thesis. There are 311,029 test instances which should be labeled.

In the KDD Cup 99, the database mining algorithms are used, the winning entry gives results of %99 for normal, % 83.3 for DOS, % 97 for Probe, % 13 for U2R and %8 for R2L. In these results, there are no information to identify the anomalies.

The patterns, in general, have features which are learned by the pattern classification algorithms. These features can be reduced by a feature selection algorithm to speed up the training and testing. In KDD Cup Data 99, every pattern has 41 features. So, in CLIDS the FCBF algorithm is used to reduce these features. These algorithm is a new and fast algorithm and it gives better results than the other known feature selection algorithms. These algorithm depends on finding the entropy of the features of the patterns.

CLIDS has 3 implementation parameters. These parameters have default values in the program, but they are changed and the tests are repeated to show the difference in the test results.

The FCBF algorithm should be given first by a value scala for every feature in a configuration file, so this is possible for symbolic values but impossible for continuous values. In order to use the continuous values in FCBF algorithm, the value scala for every feature with continuous values is split by a factor. This split factor is a parameter in CLIDS.

By calculating the Euclidean distance between instances a feature weight factor is used. This feature weight factor is also a parameter in CLIDS.

To find the anomalies, the distance between the test instance and the median of the clusters of the intrusions are tested , if it is bigger than the radius of the cluster. This radius is multiplied by a radius factor to make the cluster bigger. This radius factor is also a parameter in CLIDS.

By using the default values(Radius Factor 1.2, Continous Split Factor 10, Feature Weight Factor 10) the test results are as the following: Detection Rate DOS: %74.9, Probe % 60.9, U2R % 24.2 , R2L % 8.4, Normal, %72.5, Anomaly %85.9.

First, by making the radius factor bigger, it is seen that the detection rate of DOS doesn’t change, the detection rate of PROBE is getting less, the detection rate of U2R and R2L depend on radius. The false positive rate of DOS is getting less, the false positive rate of PROBE is getting higher, the false positive rate of U2R is very high by radius factor 0.8 and 1, the false positive rate of R2L is getting higher with making bigger the radius factor. The detection rate of normal is getting higher, but the detection rate of anomaly is getting less also with making bigger the radius

factor. This results are expected, since making the radius of all clusters bigger, makes to find the anomalies hard and more attack instances are labeled as “normal”, instead of an attack type.

Second, by training phase, the continuous split factor is selected as 100, which is 10 times bigger than the default value. The results are as the following: The detection rate of DOS has a small change of getting less, the detection rate of PROBE is getting higher, the detection rate of U2R doesn’t change, the detection rate of R2L is getting less, the detection rate of normal and anomaly have a small change. These have the following meaning: The continuous features of PROBE are and R2L attacks are not grouped as by attacks DOS and U2R. So, by training phase, the continuous split factor should be selected differerent for every type of attack, which is not supported by CLIDS yet.

Third, the feature weight factor is selected as 1 and 100. The detection rates of DOS, PROBE and U2R are getting higher , the detection rate of R2L is getting less, the detection rate of anomaly is getting higher but the detection rate of normal is getting very less, by making the feature weight factor is bigger. The false positive rate of DOS has a small change, the false positive rate of PROBE is very high by feature weight factor 1, the false positive rates of U2R and R2L are less by feature weight factor 1, the false positive rate of anomaly is getting very high by feature weight factor 100. It is also seen that the attack false/attack count ratio is getting very high, too. These results have the following meaning: Making the feature weight factor higher, makes the clusters to sit each other, so, they intersect each other, and this makes the results worse.

By all of these tests, the normalization was selected as Min-Max normalization. The fourth test was with Zero-Mean Normalization. In this test, it is seen that, the detection rates of the attacks are higher, but the detection rate of normal is very low, and the false positive rates of the attacks U2R, R2L and anomaly are very high. So, in this type of normalization, the other parameters should be changed to come in a suitable result.

In conclusion, CLIDS is an implementation of feature selection and pattern classification algorithms specified for intrusion detection. It shows, in general, better results than the other pattern classification ID systems that are mentioned in this thesis. It finds the trained attacks and the anomalies, which was the goal of this thesis.

As a future work, it should be developed more to run faster offline. Making the training database bigger will give better results, since there will be more clusters. CLIDS labels anomalies as “anomaly”, so these anomalies can be categorized as DOS, probe, U2R or R2L also. But to give a label to the found anomalies is an manual work. After labelling these instances, they can be also trained, and can be given CLIDS as training attacks.

REFERENCES

[1] **Mace R. and Mell P.,** 2000, Intrusion Detection Systems NIST Special Publication

[2] **Debar H., Dacier M., Wespi A.,** 1999, Research Report, A Revised Taxonomy for Intrusion-Detection Systems IBM Reserch Zurich Research Laboratory

[3] **Allen J., Christie A., Fithen W., McHugh J., Pickel J, Stoner E.,** 2000, State of the Practice of Intrusion Detection Technologies, Carnegie Mellon, Software Engineering Institute

[4] **Werret J.,** 2003, Review of Anomaly-Based Network Intrusion Detection, School of Computer Science & Software Engineering, University of Western Australia

[5] **Lee W., Stolfo S., Mok K. W.,** 1999, A Data Mining Framework for Building Intrusion Detection Models,Computer Science Department, Columbia University

[6] **Noel S., Wijesekera D., Youman C.,** 2001, Modern intrusion detection system , data mining, and degrees of attack guilt

[7] **Dasgupta D.,** 1999, Immunity-Based Intrusion Detection System:A General Framework, Division of Computer Science Mathematical Sciences Department, The University of Memphis

[8] **Su M.,** 2003, Cluster Analysis, Department of Computer Science and Information Engineering National Central University

[9] **Puttini, R. S., Marrakchi R., Mé L.,** 2003, A Bayesian Classification Model for Real-Time Intrusion Detection

[10] **Zanero S., Savaresi S. M.,** 2004, Unsupervised learning techniques for an intrusion detection system , *SAC'04* March 14-17 2004, Nicosia

[11] **Prerau M. J., Eskin, M.,** 2001, Unsupervised Anomaly Detection Using an Optimized K-Nearest Neighbors Algorithm

[12] **Portnoy L.,** 2000, Data Mining Lab, Columbia University, Intrusion Detection with unlabeled data using clustering

[13] **Ryan J., Lin M. J., Miikkulainen R.,** 1998, Intrusion Detection with Neural Networks, Advances in Neural Information Processing Systems 10, Cambridge, MAMIT Press

[14] **Barry P. J.** , 2002, Intrusion Detection – Evolution beyond Anomalous Behaviour and Pattern Matching, Security Essentials Version 1.4

[15] **Chittur A.,** 2001, Model Generation for an Intrusion Detection System Using Genetic Algorithms , Ossining High School Ossining, NY

[16] **Ertoz L., Eilertson E., Lazarevic A., Tan P. N., Dokas P., Kumar V., Srivastava J.,** 2004, Detection of Novel Network Attacks Using Data Mining, Computer Science Department

[17] **Duda R. O. Hart P. E., Stork D.G.,** 2001, Pattern Classification, Wileyans Sons, Second Edition

[18] http://www.developer.com/java/article.php/10922_1491641_1

[19] **Jain A.K., Murt M.N. Flynn P.J,** 1999, Data Clustering: A Review, ACM Computing Surveys, Vol. 31, No. 3

[20] **Guan Y., Ghorbani A.A, Belacel N.,** 2003, Y-MEANS: A Clustering method for intrusion detection, Montreal, May/mai 2003 IEEE
[21] **Bace R. G.,** 2000, Intrusion Detection, Macmillan Technical Publishing
[22] SANS Conference Notes, 2000
[23] **Northott S., Novak J., McLachlan D.,** 2001, Network Intrusion Detection An Analyst's Handbook, New Riders Publishing
[24] **Paxon V.,** 1998, Bro: A system for detecting network intruders in real-time. In Proceedings of the 7th USENIX SecuritySymposium, San Antonio, TX.
[25] **Cohen W. W.,** 1995, Fast effective rule induction. In MachineLearning: the 12th International Conference, Lake Taho,CA.

[26] **Bozkurt B.,** 2003, Predicton of protein subcellular localization using global protein sequence feature, M.Sc.,Thesis, The Middle East Technical University

[27] http://kdd.ics.uci.edu/databases/kddcup99/task.html

[28] http://www-cse.ucsd.edu/users/elkan/clresults.html

[29] **Mukkamala S., Sung A. H.,** 2003, Feature Ranking and Selection for Intrusion Detection using Support Vector Machines , Computer Science Department, New Mexico Tech

[30] **Eskin E., Prerau M.J**, 2001,Unsupervised Anomaly Detection Using an Optimized K-Nearest Neighbors Algorithm

[31] **Yu L., Liu H.,** 2003, Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution , Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC

[32] **Sherf M., Brauer W.,** 1997, Feature Selection by Means of Feature Weighting Approach, Technical Report, Technics University, Munich

[33] **Abraham T.,** 2001, IDDM: Intrusion Detection using Data MiningTechniques, Information Technology Division Electronics and Surveillance Research Laboratory

[34] **Guozhu Dong,** 2003, Data Preparation and Preprocessing Lecture Notes

[35] http://www.freesoft.org/CIE/index.htm

AUTOBIOGRAPHY

Müge ÇEVİK was born in 18/07/1977 in İstanbul. She graduated in 1996 from Istanbul High School and in 2000 from Istanbul Technical University Control&Computer Engineering. She worked from November 2000 to April 2004 by TÜBİTAK UEKAE (National Research Institute of Electronics&Cryptology) as Researcher. She has been working now by SIEMENS TURKEY in Programming and System Engineering as Software Engineer from May 2004.