ANN BASED NETWORK INTRUSION DETECTION SYSTEM

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

MAJOR SEBY THOMAS

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

JUNE, 2005

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in this dissertation entitled "**ANN BASED NETWORK INTRUSION DETECTION SYSTEM**" in the partial fulfillment of the requirements for the award of **Master of Technology** in the **System Engineering & Operation Research**, submitted in the **Department of Electrical Engineering** of the institute is an authentic record of my own work carried out under the guidance of Prof J.D. Sharma and Prof M. K. Vasantha, Department of Electrical Engineering, IIT Roorkee, Roorkee.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

Dated: 17 June 2005

[illegible]

Major Seby Thomas

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

[illegible]

Prof. J. D. Sharma

PSE group

M Vasantha 17/6/05

Prof. M. K. Vasantha

S.E.O.R group

ACKNOWLEDGEMENT

At the outset I express my gratitude to Corps of Signals, Indian Army for having offered me this excellent opportunity of furthering my academic ambitions. I am also grateful to Military College of Telecommunications Engineering, Mhow for suggesting me to work in this area and laying out an ambitious scope, ensuring that the subject be dealt in its depth.

I am grateful to Prof. J. D. Sharma for being the beacon that ensured that no effort was aimless. With deep gratitude, I recall all the efforts that he took to achieve the project goals. I recollect, that in the very first visit to his office, he had handed over four papers on the subject using ANN tools and not to leave things open, asked me to discuss them a day after. The continuous monitoring, his vision in setting the goals and deadlines ensured that the work steered through smoothly. I am greatly influenced by his personal quality in deriving more out of every possible moment that life offers.

As I pen this acknowledgement, I recollect with nostalgia the day a year back, while in Prof. M. K. Vasantha office, I was expressing my fears over attempting this work. As I left his office then, I was ballooning with self-pride, immense confidence and charged with energy that made my fears look dwarf. Interactions and those short inspirational talks over the year have recharged my energy levels at the hours of my need. With him I have learned that pure motivation, optimism and personal warmth can drive a person to stretch his limits. Working under the guidance of them has been a great learning for now and life.

I also acknowledge Prof. Hari Om Gupta (H.O.D Electrical Department) efforts in helping me visualize the optimization techniques during his course study on Operational Research. The insight that Dr. N. P. Padhy provided in ANN techniques have made it possible for me to try beyond. I also acknowledge Prof. Vinod Kumar efforts in taking time to explain concepts beyond the dimensions of course study and Dr. Indra Gupta for her course on Data Structures that made me opt for C language to implement the work. I would cherish the memories of interactions with Dr. Rajendra Prasad and Dr. E.

[illegible] support extended by Prof A. K. Pant (Group Leader S.E.O.R) and Dr. Surendra Kumar (O. C M Tech) in the course of this work. The facilities offered by the Department and its lab Technicians (Mr. Kalyan Singh and C. M. [illegible]) offered flexibilities that deserve praises.

These last two years, I have lived a new life poles apart from the military way; my new found friends in Research scholars Mr. Vishal Kumar, Mr. Rahul Dubey, Mr Bhander Singh and Ms Nidhi Singh have made those moments sweet and nostalgic. The lessons derived from these thought provoking discussions with Mr. Vishal Kumar are cherished within. I am grateful to all my colleagues of S.E.O.R group (Mr. Naveen [illegible], Srikanth Reddy, Praveen Kundu and H S Rathod) for being excellent peers and providing a congenial environment for work.

The pleasure of nearing completion for the course requirements is immense, but its accompanied the pain of leaving behind these wonderful two years of life in the sprawling green campus of this great historically reputed Institute. The pride of being a part of this great Alma mater will always be evident in me.

I thank my wife Mrs. [illegible] Thomas and my daughters, Simran and Nikita for [illegible] ordeal with optimism and smile. I am grateful to our [illegible] for their constant support and well wishes.

[illegible] Vishal Kumar, without whose intervention this knowledgment would have just been the black ink on a white paper. I am grateful that I ok his advice to spare few moments for all those who mattered.

(Seby Thomas)
Major
M. Tech. (S.E.O.R)

ABSTRACT

The rapid proliferations of Internet and our dependence on networks in all domains of life have made us more vulnerable to breaches of internet/network security. It is difficult to prevent such attacks by security policies; firewalls or other mechanisms alone as operating system and application software are known to contain weaknesses or bugs. The attackers continually exploit these loopholes in network protocols and software component. Intrusion detection systems are designed to detect such attacks that enviably occur despite security precautions.

An attack on a network is considered an abnormal activity. It is this underlying assumption that is critical in detecting an attack in an anomaly based detection technique,. where as misuse detection identifies a pending attack based on its prior knowledge of attack signatures. In this dissertation work, an amalgamation of both misuse and anomaly based detection technique, employing their individual strength in detecting attacks, is proposed using Hybrid Neural Network-in which the output of Kohonen's Self Organized Map provides input to feed forward neural network. The data from MIT Lincoln Laboratory created DARPA 1999 Intrusion Detection Evaluation data set (approximately of size 10 GB) was applied for training and testing of prototype.

The system prototype designed, is a network based intrusion detection system that scrutinizes tcpdump data on a source-by-source basis in a time window to develop windowed traffic behavioral trends. It is assumed that the evidences of an attack lie within the packets and can be identified either by individual analysis of packet in some cases or by ascertaining the attackers intention by analyzing sequel of packets in a time window frame. The core detection engine of our system is based on anomaly based detection technique, which detects attack by sensing deviations from its learned normal trait. The abnormality is self-learned by the system by way of Kohonen based Self-organizing mapping techniques. The clustering mechanism maps the windowed traffic trend of individual machines to clusters indicative of behavior pattern based on features extracted from network activity. The features extracted are decisive in forming abnormal clusters in its outliers. Data mining skills are applied to compute statistical trend and features that are flagged by the presence of attack signatures. The features presented to the clustering mechanism, reflect the behavioral trend of source machine in communication with victim in terms of both statistical features as well as flags indicative of attack signatures. The clusters so formed during training are learned as normal or abnormal by the neural network. The supervised training of the neural network is carried out by means of the labeled tcpdump data using Levenberg-Marquardt algorithm for back propagation.

The work involves design and development of a network based intrusion detection system. The program is wholly written in GNU C and based on Linux platform. The prototype developed can be executed both using graphic user interface and console terminal using command line arguments. The graphic interface for the project has been built using libglade.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

Addr	Address
ALAD	Application Layer Analysis Detection
BSM	Basic Security Module
BIOS	Basic Input Output System
CGI	Common Gateway Interface
DARPA	Defence Advance Research Project Agency
DoS	Denial of Service
Dst	Destination
Dup	Duplicate
FP	False Positive
GIMP	GNU Image Manipulation Program.
GNU	Gnu Not Unix
GNOME	GNU Network Object Modeling Environment
GLADE	Glade is a free user interface builder for GTK+ and GNOME, released under the GNU GPL License.
GrIDS	Graphical based Intrusion Detection System
GTK+	GIMP Tool Kit
HTTP	Hypertext Transfer Protocol (world wide web protocol)
HBIDS	Host based Intrusion Detection System
ICMP	Internet Control Message Protocol
ID	Intrusion Detection
IDA	Intrusion Detection Agents
IDS	Intrusion Detection System
Invalid_port	Feature used for clustering that keeps a count of number of invalid ports accessed by the source machine on a victim machine
IP	Internet Protocol
IT	Information Technology

Key	Feature used for clustering that takes a numeric value based on the attack signature if present the activity of the source machine.
LAN	Local Area Network
MLFFBPNN	Multi Layer Feed Forward Back Propagation Neural Network
M.S.E	Mean Square Error
NBIDS	Network based Intrusion Detection System
NUM_FEATURES	The number of features being extracted from the network traffic by the data mining module.
PHAD	Packet Header Analysis Detection
R2L	Remote to Local
Same_addr	Feature used in clustering mechanism which takes a value 1 if both src and dst addr are same
SBIDS	Signature based Intrusion Detection System
SOM	Self Organized Mapping
Src	Source
TCP	Transport Control Protocol
TP	True Positive
TTL	Time to Live
U2R	User to Root
UDP	User Datagram Protocol
URG	Urgent

CONTENTS

Chapter1

INTRODUCTION TO INTRUSION DETECTION SYSTEM

1.1 DEFINATION

An Intrusion Detection System (IDS) can be defined as a hardware/software system that monitors events in computer/network to identify unlawful attempts to penetrate it. An intrusion is an unauthorized access or usage of the resources of a computer system [1]. IDS are the software with the functions of detecting, identifying and responding to the unauthorized or abnormal activities on a target system [4]. The goal of the IDS is to provide a mechanism for the detection of security violations either in real-time or batch-mode [2, 3]. Violations are initiated either by outsiders attempting to break into a system, or by insiders attempting to misuse their privileges [6]. IDS collect information from a variety of systems and network sources, and then analyze the information for signs of intrusion and misuse [5].

1.2 CLASSIFICATION OF IDS BASED ON SOURCE OF AUDIT DATA

An attack can be detected by monitoring the behavior of the system either at the user level or at network level. The behavior of a user can be ascertained by monitoring the system logs that are generated by the various processes run by the user, while the nature of traffic in the network governs the behavior of the network. The data from the system logs or the network traffic forms the audit data. Based on the sources of audit data, an intrusion detection system can be classified as HBIDS and NBIDS.

Host based intrusion detection system (HBIDS) normally use system call data from an audit process that tracks all system calls made on behalf of each user on a particular machine.

Network based intrusion detection system (**NBIDS**) typically use network traffic data from a network packet sniffer (eg tcpdmp). Many computer networks including the widely accepted Ethernet (IEEE 802.3) networks use a shared medium for communication. Therefore, the packet sniffer only needs to be on the same shared subnet as the monitored network.

1.3 IDS: ARCHITECTURE

There are three architecture of intrusion detection system i.e., **Distributed, Central** and **Hierarchical.**[11] The hierarchical architecture consists of several tiers with each tier containing several intrusion detection agents (IDAs). IDAs are IDS components that monitor the activities of a host or a network. Different tiers correspond to different network scopes that are protected by agents affiliated to them. While in a centralized architecture a central console controls the other IDSs/IDAs. In a distributed architecture the intrusion detection system collects audit data from several sensors. To understand it better let us consider a sample network as shown in figure 1.1 [12].

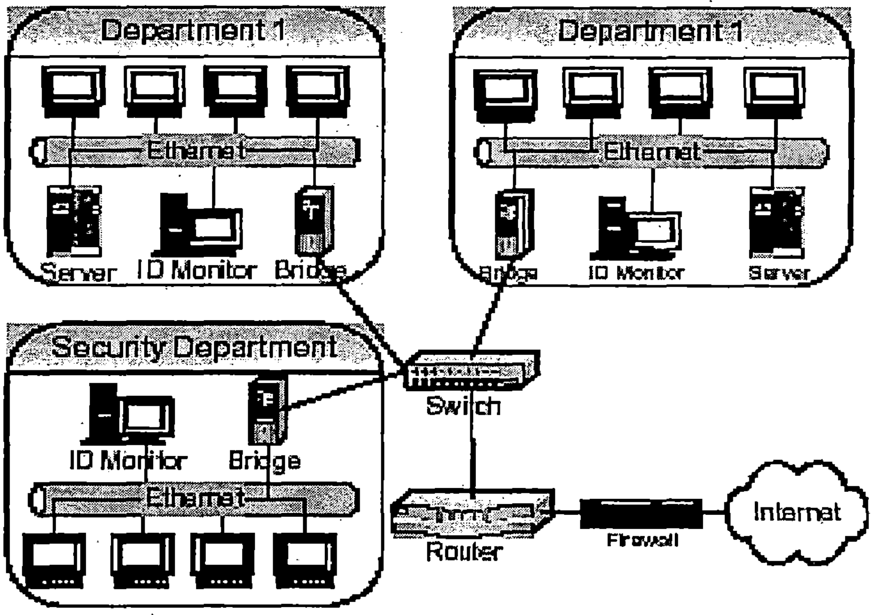


Figure 1.1: A Sample Network

For the sample network shown in figure 1.1, the hierarchical architecture IDS can be divided in three tiers as shown in figure 1.2. Tier 1 agents monitor system activities of the servers and bridges within a department and periodically generate reports for tier 2 agents. Tier 2 agents detect the network status of the departmental LAN based on network traffic that they observe as well as reports from Tier 1 agents from the LAN. The tier 3 agents at the security department collect data from the tier 2 and tier1 agents. In hierarchical based IDS the system hierarchy is followed.

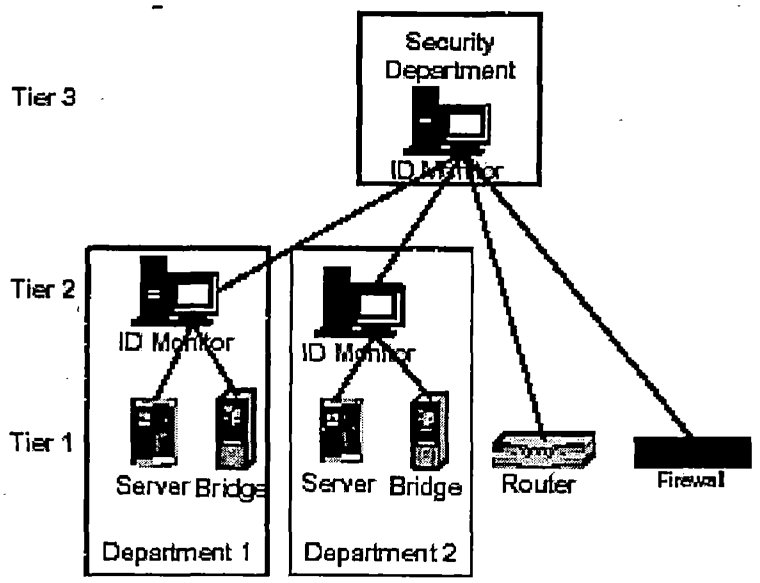


Figure 1.2: System Hierarchy of Hierarchical architecture IDS

In Central architectural IDS, the ID monitors as shown in the sample network in figure 1.1, act as intrusion detection agents for the respective departments, while any alerts or alarms are centrally logged. The start/stop functionality of the departmental IDS can be centrally controlled; also the departmental IDA's can be forewarned of new attacks (updating of signatures), if known.

In the case of Distributed architectural IDS the ID monitors act as network sniffers i.e., collect network data of respective departmental LAN, which forms the audit data for the intrusion detection system at the security department in the above example.

1.4 CLASSIFICATION BASED ON DETECTION TECHNIQUE

Based on the detection technique employed, intrusion detection system falls into one of the two categories i.e. Anomaly detection and misuse detection[10].

Anomaly detection is a statistical approach that gathers a variety of parameters concerning network usage and weighs this against incoming network activity. If the statistical deviation is significant then the IDS notifies an attack.

Misuse detection considers a pattern of attacks and then compares network activities against these patterns. If a given activity resembles a known pattern of attack, the IDS notify that an attack may be imminent.

Both these approaches suffer from the problems that the sheer volume of network traffic often renders it infeasible to conduct the necessary analysis. As well, it is often difficult to set benchmarks as to what constitutes a significant statistical deviation (in the case of anomaly detection) or a strong resemblance between patterns (in the case of misuse detection). This can result in incorrectly labeling valid network activity as an attack, or in failing to detect one. Both of these seriously undermine the usefulness of the IDS. Therefore different methods of detection are needed to address the inadequacies of these approaches. They are discussed in later section of this report. But before we discuss the methods of detection it is important to understand the attack techniques and the basic components of an intrusion detection system. The attacks will be covered in adequate detail in chapter 3.

1.5 COMPONENTS OF INTRUSION DETECTION SYSTEM

The core module of an intrusion detection system is the detection engine which identifies normal and intrusive activities based on knowledge facilitated by detection model; for instance a signature based detection model will contains patterns of attacks as rules which are then matched by the detection engine with the current network activity to detect malicious activity. The network traffic is monitored by a sniffer module which forms the audit data for a network based intrusion detection system. The audit data is then fed to a preprocessor module, which computes the network activity. In a audit data preprocessor a given packet is broken down into number of fields such as protocol, source IP, destination IP, ports used and flag settings (in the case of TCP or UDP) or message type (in the case of ICMP) and length. This network activity is one of the input to the detection engine the other being attack patterns fed by the detection model in case of signature based detection system, where as in case of statistical based detection engine the network activity is further assigned a anomaly score which is then compared to the probabilistic scores of normal network activity provided by the statistical based detection model. The detection engine raises an alarm if the pattern is matched (signature based detection model) or if the anomaly score of a packet is above the threshold (statistical based detection model).

The alarm generated by the detection engine is fed to the decision engine, which based on its decision table (more applicable incase of statistical model since scores are probabilistic in nature) it takes action (like reconfiguring firewall) and reports the action to system administrator as illustrated in Figure 1.3

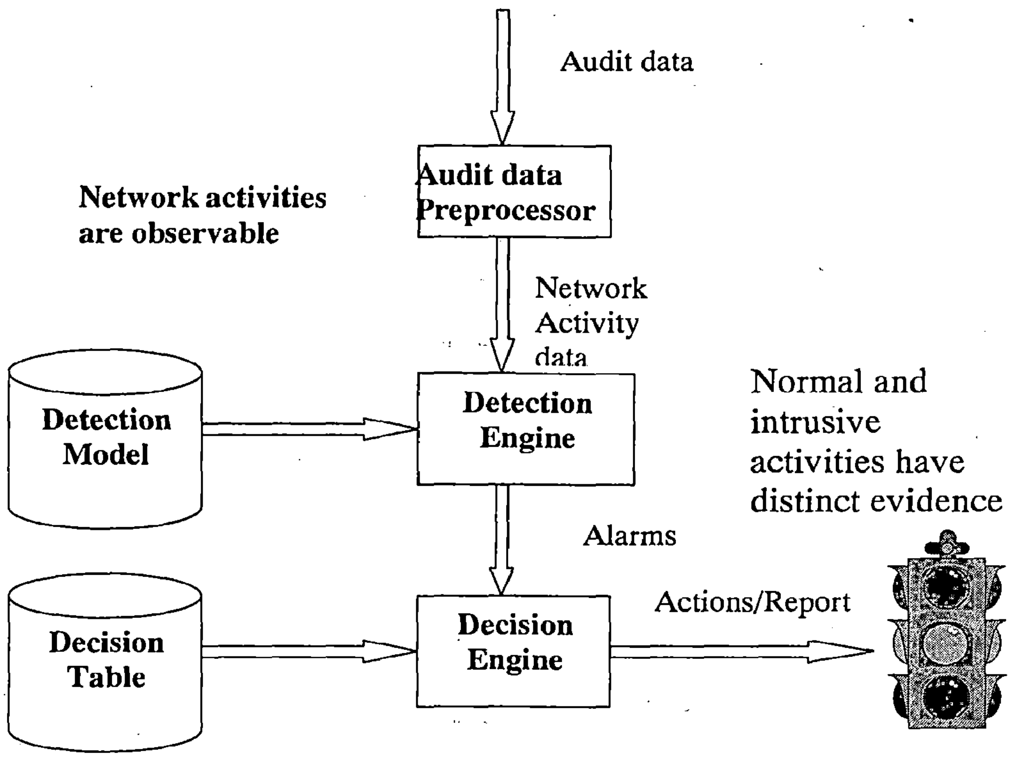


Figure 1.3 Components of Intrusion detection system

1.6 METHODS OF INTRUSION DETECTION

Intrusion detection system monitors computer network traffic and attempts to identify, alert and present all anomalous activities to the user/system administrator. The key to an intrusion detection system is to maximize accurate alerts (true positive) while at the same time minimizing the occurrence of non-justified alerts (false positive). There are various methods of intrusion detection; the most popular of them are enumerated. All these methods fall under one of the two categories mentioned in earlier section,

(a) Statistical based detection..

(b) Signature based (rule-based) detection

(c) Neural network based detection.

(d) Graphical based detection.

(e) Artificial immune system based detection.

1.6.1 Statistical based detection technique

In any network an intrusion is considered not a normal activity and it is this very fact that is used to detect intrusion, by detecting any deviation from the normal network activity. There are various methodologies employed to identify the deviation. A simple Statistical based intrusion detection systems (SBIDS) relies on statistical models, to identify anomalous packets on the network. To identify an anomaly, the system uses data compiled from previous network behavior. Since warnings are based on actual usage patterns, statistical systems can adapt to behaviors and therefore create their own rule usage-patterns. Anomalous activity is measured by a number of variables sampled over time and stored in a profile. The reporting process will alert the user if the packet's anomaly score is greater than or equal to the threshold level set by the user. The SBIDS identifies and tracks patterns and usage of the network data and then assigns an anomaly score to each packet. Based on the data used to compile previous network activity a SBIDS can be host based or network based IDS. For example a host based IDS will compile data from users behavioral pattern or servers service pattern using the system log of the host machine, while a network based IDS compiles data from previous network behavior.

Each packet coming into the anomaly detector is assigned a anomaly score A(x). This score is calculated from the negative log of the probability of the event, P(x). i.e. A(x) = -log(P(x)). The calculation of P(x) is based on observed network traffic, there are four methods of calculating P(x): P(destination IP, destination port), P(source IP, destination IP, destination port), P(source IP, source port, destination IP, destination port) and a Bayes network approximation of P(source IP, source port, destination IP, destination port). From observations, a packet to port 80 on a web server will be more probable than say, port 37337 to the same server. The higher P(x) is, the lower A(x) will be. If A(x) exceeds the provided threshold, detection engine will generate an alert. Optimally, a report will be generated on all significant anomalous activity. What constitutes "significant" can vary from user to user. Therefore, it is ultimately up to the user to decide how many alerts are generated for a specific environment.

Unlike a signature based system, which has the benefit of being implemented and immediately utilized, the statistical based system must initially adapt to the network at hand to learn what is defined as 'normal' traffic. The longer a SBIDS is placed on a specific network, the more accurate the results will be. If the normal network traffic is malicious, the SBIDS will be rendered useless. Also the alerts generated will be relatively difficult to assess compared to a signature-based system. The alert will be packet information, which will require the expertise of trained security professional to decipher the reason for alert.

1.6.2 Signature based intrusion detection technique

In the manner an antivirus program scans through the files looking for malicious virus code (referred to as signature of virus being detected), a signature based intrusion detection system scans through the data contents of packets flying in the network for malicious code/script known to form part of a computer attack All attacks will form a pattern, in terms of sequence of codes/scripts. Misuse detection is the ability to identify intrusion based on known patterns for the malicious activity. These known patterns are referred to as signatures. These signatures are transformed as rule base. Rules are developed as new vulnerabilities and scanning techniques are identified. A signature based IDS (SIDS) is as strong as its rule set and if the attack is new, there will simply not be any signatures developed to identify the probe. An SIDS is programmed to look into the header of each incoming packet, whatever be the underlying protocol used on the network. Each packet through the network is scanned by the IDS core 'engine' against hundreds or thousands of signatures stored within it for presence of any malicious activity. Most of the popular SIDS are based on Expert System, where the pattern of attack is transformed into a rule base, obviously only those attacks can be detected whose patterns are known

1.6.3 Neural network based detection system

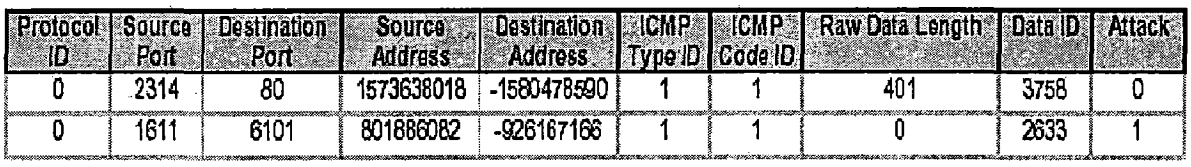
The properties of neural network are put to advantage for exploring new methods for detecting intrusions based on statistical deviation. The neural network is trained on the normal traffic for duration of n days and the statistical parameters computed stored in tables, after the completion of training the network is tested and any anomaly from the normal statistical parameters obtained from the tables mentioned above cause's alarms to be raised. There are various approaches to using neural network for intrusion detection; the most popular of those are explained in this section.

1.6.3.1 Expert System Based Misuse Detection

There are two general implementation of neural network in misuse detection[7]. The first involves incorporating them in existing expert system based intrusion detection system. This implementation is more preferred if an organization already has an expert system based IDS in place. The expert system based intrusion detection is the most common rule based approaches that encodes the knowledge of 'human expert' on the security related data. Expert system is a computer application, which then utilizes that knowledge to identify activities that match the defined characteristics of attack or misuse. The neural network here is first trained by thousands of individual sequences of attacks. Once trained it filters the incoming packets for suspicious events, which may be indicative of misuse or attack and forward these to the expert system. This implementation improves the false alarm rate of the expert system. The disadvantage of this approach is as the neural network improves its ability in identifying new attacks, the expert system also needs to be updated to recognize the attack lest it would ignore the anomaly detected by the neural network.

1.6.3.2 Stand Alone Misuse Detection

The second approach [8] involves neural network as stand alone misuse detection system. In this configuration the neural network would receive data from the network stream and analyze the information for instances of misuse. There are a number of architectures that can be used, however the multilayer feed forward network was found to be more suitable due its flexibility and applicability in variety of problems. The MLP utilized for in the model explained here consisted of four fully connected layers with nine input nodes and two output nodes. Each of the hidden and output nodes applied a sigmoidal transfer function to the various connection weights. The data from the network stream used for training and testing is grabbed first by using packet sniffer software like Tcpdump, Realsecure network monitor and then organized in the formatsuitable for the input pattern of the neural network. A misuse detection technique heavily relies on the data content of the packet since any attack can be detected, if its signature is known. Therefore for accurate detection by the neural network based IDS the raw data contents forms one of the key element besides the source, destination IP and port address, data length, protocol ID. However since the input pattern to the neural network is required to be numerical in nature. The data portion of the packet is converted to a numerical number denoted as Data ID from a look up table which assigns specific numbers to identify attack scripts. This lookup table forms the key factor for the accuracy of misuse detection and the most difficult to devise too. A sample training set consisting of the input and output pattern identifying normal and attack traffic by 0 and 1 respectively is shown in figure 1.4. The raw data length of the packet can identify a buffer overflow attack, while an unreasonable port activity can be a cause for raising an alarm.



Protocol ID	Source Port	Destination Port	Source Address	Destination Address	ICMP Type ID	ICMP Code ID	Raw Data Length	Data ID	Attack
0	2314	80	1573638018	-1580478590	1	1	401	3758	0
0	1611	6101	801886082	-926167166	1	1	0	2633	1

Figure 1.4: A Sample Training set for neural network

1.6.4 Graphical technique in intrusion detection

Both Statistical and signature based Intrusion detection systems suffer from an inability to detect an attack that is built from a sequence of valid network activity. Hence the needs to develop a methodology that can detect a malicious action that may consist of valid network activity. Graph based intrusion detection system (GrIDS) is one such method developed by the University of California. The idea behind GrIDS is to detect large scale automated attacks on networked systems. The approach in GrIDS is to build activity graphs reflecting activity in a network, and then analyze these graphs to assess whether an attack is occurring. The activity graphs are generated by graph engines, which take basic activity reports and convert them into graphs. The nodes or vertices of the activity graph represent hosts in a system, while edges of an activity graph represent network activity between the hosts. Both the edges and vertices have the property that they may have attributes associated with them, which provide additional information regarding the nature of connection or host. If the traffic between two hosts contains the transfer of password files then the attributes of the edges will reflect that. GrIDS fall in the category of misuse detection technique.

1.6.5 Artificial immune system based detection technique

The characteristics of the natural immune system are the inspiration for developing an intrusion detection system based on immunity model. The universe of all patterns of network traffic P is partitioned into two disjoint sets (P_s and P_n). The patterns (P_s) that frequently occur in network traffic in last n (user defined) days are classified as self and the ambiguous patterns (P_n) as nonself. It is assumed that the traffic of n days that formed the self-pattern did not contain any attack or intrusion attempt i.e., Ps. P_n= . .

The intrusion detection system based on natural immune system focuses on following aspects.

(a) The method of using data mining technique to explore the adaptable set of self (i.e. normal patterns of client and server activity) and sequential patterns of TCP services.

(b) The method of generating valid detectors based on genetic algorithm;

(c) The mechanism of memorizing previously seen intrusion patterns and of learning sequential patterns of intrusions through vaccination.

(d) The mechanism of dynamic detectors with finite lifetime

1.7 SHORTFALLS IN CURRENT MISUSE IDS

While the ability to develop and use signatures to detect attacks is a useful and viable approach there are shortfalls to only using this approach, which should be addressed.

- *Variants.* As stated previously signatures are developed in response to new vulnerabilities or exploits, which have been posted or released. Integral to the success of a signature, it must be unique enough to only alert on malicious traffic and rarely on valid network traffic. The difficulty here is that exploit code can often be easily changed. It is not uncommon for an exploit tool to be released and then have its defaults changed shortly thereafter by the hacker community.
- *False positives.* A common complaint is the amount of false positives an IDS will generate. Developing unique signatures is a difficult task and often times the vendors will err on the side of alerting too often rather than not enough. This is analogous to the story of the boy who cried wolf. It is much more difficult to pick out a valid intrusion attempt if a signature also alerts regularly on valid network activity. A difficult problem that arises from this is how much can be filtered out without potentially missing an attack.
- *False negatives* detecting attacks for which there are no known signatures. This leads to the other concept of false negatives where an IDS does not generate an alert when an intrusion is actually taking place. Simply put if a signature has not

been written for a particular exploit there is an extremely good chance that the IDS will not detect it.

- *Data overload.* Another aspect which does not relate directly to misuse detection but is extremely important is how much data can an analyst effectively an efficiently analyze. That being said the amount of data he/she needs to look at seems to be growing rapidly. Depending on the intrusion detection tools employed by a company and its size there is the possibility for logs to reach millions of records per day.

1.8 Hacking

1.8.1 General:

Hacking is an act by an intruder to unlawfully access a computer/system. For detecting such an activity knowledge of hacking techniques is a prerequisite. In this section, the topic is introduced and references for further reading are listed[34-44].

Hackers have more tools and information at their disposal and enterprises have to react very quickly to vulnerabilities to minimize security incidents. The nature of attacks against enterprises began changing dramatically as the Internet started to have an impact on IT architecture (figure 1.5)

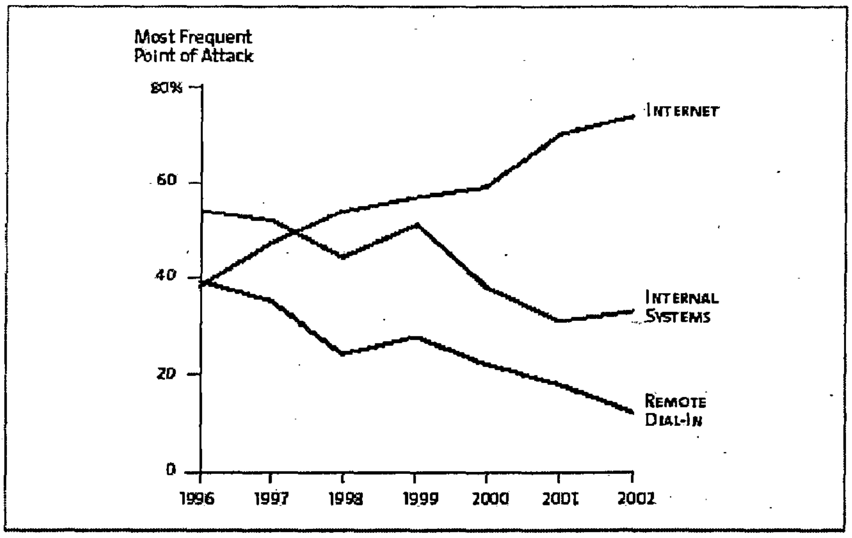


Figure 1.5: Change in Nature of Attacks
(FBI/CSI Computer crime and Security Survey 2001)

With the advent of easy availability of hacker tools, ever more sophisticated attacks are being launched, by ever less sophisticated attackers (figure1.6).

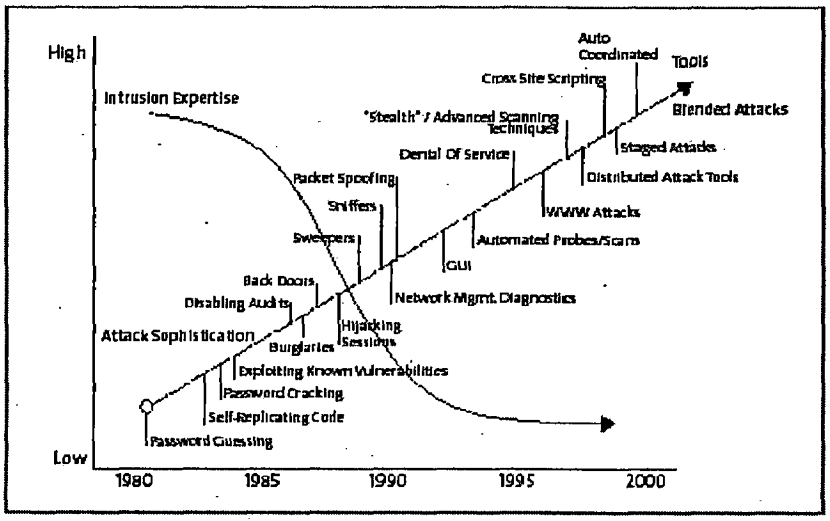


Figure 1.6: The Growth of Attack Sophistication over Time

(CERT/Carnegie Mellon University 2001)

1.8.2 Types of Intruders

An intrusion is somebody (Hacker or Cracker) attempting to break into or misuse your system. Intruders can be classified into two categories.

Outsiders - Intruders from outside the network, who may come from the Internet, dial up lines, physical break-ins or from neighborhood networks.

Insiders – Intruders that legitimately use your internal network, these include users who misuse privileges. A frequently quoted statistics is that insiders commit 80% of security breaches.

The primary ways an intruder can get into a system are:

Physical intrusion If an intruder has physical access to a machine i.e., they can use the keyboard or physically take apart the system and remove the disk drive. Even BIOS protection is easy to bypass, virtually all BIOSes have backdoor passwords.

System intrusion This type of hacking assumes the intruder already has a low privilege user account on the system and he is able to use a known exploit in order to gain additional administrative privileges.

Remote intrusion This type of hacking involves a intruder who attempts to penetrate a system remotely across the network.

1.8.3 Typical Intrusion Scenario

A typical intrusion scenario might be

Step 1: outside reconnaissance The intruder will find out as much as possible without actually giving themselves away. The intruder will do a 'whois' lookup to find as much information as possible about your network as registered along with your domain name. The intruder will walk through your DNS tables (using 'nslookup', 'dig' or other utilities) to find names of your machines.

Step 2: inside reconnaissance The intruder uses invasive technique to scan for information, but still doesn't do anything harmful. They might walk through all your web pages and look for CGI scripts. They might do a ping sweep in order to see which machines are alive or a TCP/UDP scan/probe to see which services are available.

Step 3: exploit The intruder crosses the line and starts exploiting possible holes in target machines. The intruder may attempt to compromise a CGI script by sending shell commands in input fields. The intruder might attempt to exploit well known buffer overrun holes by sending large amount of data or break passwords of user accounts by brute force.

Step 4: foothold At this stage the hacker has gained access into the network and his main goal is to hide evidence of the attacks (by doctoring audit trails and log files). They may install 'toolkits' that give them access, replace existing services with their own Trojan horses that have backdoor passwords or create their own user accounts. So even if the machine may not in itself have any thing of interest to an intruder but he uses it as a stepping-stone to attack other systems also in turn hiding his identity,

Step 5: capitalize The intruder takes advantage of their status to steal confidential data, misuse system resources.

1.9 RELATED WORK

Intrusion detection has traditionally focused on one of two approaches. Misuse detection compares a user's activities with the known behaviors of attackers attempting to penetrate a system. The second approach, anomaly detection seeks to identify activities that vary from established patterns for users, or network. Anomaly detection is a widely used method in the field of computer security, and there are approaches that utilize it for detecting intrusions [3].

Various techniques for modeling anomalous and normal data have been developed for intrusion detection. A survey of these techniques is given in [7]. A method that is closely related to the work in this dissertation employs clustering mechanism to classify abnormal activities and detects attack based on the assumptions that the unlabeled dataset contains large amount of normal activity and relatively few anomalies [8] and [9]. The first use of Kohonen self-organizing map in misuse detection is described in [15]. There a hybrid neural network – in which the output of a Kohonen map provided input to a conventional feed forward neural network, was prototyped to address temporally dispersed, and possibly collaborative, attacks in a simulated data stream. Temporally dispersed attacks are those conducted by a single attacker over an extended period of time, while multiple attackers working in concert to achieve a single intrusion conduct collaborative attacks.

The other more related technique involves use of neural networks for detecting intrusions. An approach, which detects network based attacks as anomalies using statistical preprocessing and neural network classification is discussed in [12]. The paper tested five different types of neural network classifiers: perceptron, Backpropogation (BP), Perceptron-backpropogation-hybrid (PBH), Fuzzy ARTMAP and Radial based function.

A technique given in [13] discusses detecting intrusions using neural networks by training on packet header fields extracted in the preprocessing stage for a customized network. A comparative performance study of various type of NN perceptron, BP and PBH is also included. In [14] it proposes a learning algorithm that constructs models of normal behavior from attack-free network traffic and the behavior that deviates from the learned normal model signals a novel attacks. The model for normal behavior is prepared based on the packet header (PHAD) and application layer (ALAD) analysis. The first component PHAD monitors 33 fields from the Ethernet, IP and transport layer (TCP, UDP, or ICMP) packet headers, however, 15 fields were found to contribute towards detection. The anomaly scores of each packet is calculated based on probability study such that any new instance of field under consideration would raise the anomaly score of the packet. The total anomaly score of individual packets are thus computed and if above the set threshold it signals alarm for detecting attacks. The second component of the model ALAD, instead of assigning anomaly scores to each packet, assigns a score to an incoming server TCP connection. TCP connections are reassembled from packets. The performance of the intrusion detection is enhanced by using keyword selection in neural networks as discussed in [5]. Many an attacks like DOS and its variants comprise of valid field values at packet level, however the flooding of these packets on a victim machine leads to the attack. Hence techniques attempting to detect attacks based on individual packet headers will fail to detect such anomalous activities comprised of valid commands/requests.

1.10 AUTHOR'S CONTRIBUTION

The author's contribution is towards designing a system that add focus to anomaly based IDS by enhancing its potency against known attacks by including attack signatures as additional statistical feature. The advantage of high rate of detecting known attacks by misuse-based detection is complemented to this system whose core detection engine is primarily based on anomaly based detection technique.

The system detects attacks based on source machine's activity over a dynamic time window, which overcomes the handicap of other systems assessing on individual packet analysis. Data mining skills are applied in computing statistical trends as well as flagging features based on likelihood of attack signature in the source activity. The features thus presented to the clustering mechanism reflect the behavioral trend of a source machine in communication with the victim machine in terms of statistical features and flags indicative of attack signatures. The evidence of attacks is self-learned by the Kohonen based SOM technique and these clusters are classified by MLFFNN using Levenberg Marquardt algorithm for backpropagation. Labeled tcpdump data are used for supervised training of neural network. The complexity of training data and its size (approximately 4.4 GB) demanded a faster convergence algorithm like Levenberg Marquardt for backpropagation of neural network weights.

One of the main assumptions made was that data instances presented by the data-mining module having similar characteristic would be close together under some metric in the clustering mechanism. Therefore finding or constructing an appropriate metric is essential for clustering. In detecting network intrusions, it is imperative that some features of the data instances would be more important (have greater weight) than others viz. flags indicative of attack signatures, and thus differences in the values of those features should have a greater contribution to the overall distance. Therefore, several-weighted metrics were tried, with higher weights assigned to different subsets of features in this work and it was decided to use a standard Euclidean metric, with weighted features so that all data instances pertaining to an attack fallinto the same cluster.

The system prototype was tested on tcpdump data of the test week (4-5th week); however the system is capable of adapting to real time detection of network traffic if the statistical parameters of the tcpdump data used for trainingmatch that of the live network traffic. This is feasible as the detection algorithm is largely independent of machine specific details viz. source/destination IP address etc except for the normal traits of the victim machine. The use of neural network based design reduces the computational and memory needs of system during real time detection stage, the bulk of the systems overheads in terms of processing and memory needs are limited to the training phase.

The use of pcap libraries give an inherent advantage to the system of adapting to real time detection of network traffic since the traffic are handled as network packets rather than as text files. The prototype developed is made user friendly by means of GUI built using libglade.

Chapter 2

DATASET DESCRIPTION

2.1 GENERAL

The development and design of a neural network based detection system will necessitate a vast amount of labeled network data comprising of both with and without attacks, for training of neural network. The problems envisaged in building such an enormous data is limited to the researchers resources of originating variants of attacks and capabilities of simulating a large network. To promote more researchers to work in the field of IDS and the need to compare different IDS, Massachusetts Institute of Technology (MIT) Lincoln Lab created DARPA 1999 Intrusion Detection Evaluation data set. This data set is publicly available [16] and is approximately of 10 GB size.

DARPA (Defense Advanced Research Projects Agency) is the independent research branch of the U.S. Department of Defense (DoD) that funded a project that in time was to lead to the creation of the Internet. Originally called ARPA (the "D" was added to its name later), DARPA came into being in 1958 as a reaction to the success of Sputnik, Russia's first manned satellite. In the late 1960s, ARPA provided funds and oversight for a project aimed at interconnecting computers at four university research sites. By 1972, this initial network, now called the ARPANET, had grown to 37 computers. Because ARPA's name was changed to Defense Advanced Research Projects Agency (DARPA) in 1971, some people refer to ARPANET as DARPANET. (DARPA was changed back to ARPA in 1993 and back to DARPA again in 1996). It manages and directs selected basic and applied research and development projects for DoD, and pursues research and technology that may provide dramatic advances for traditional military roles and missions; funding research activities for Intrusion Detection System is one of it's key thrust areas.

2.2 1999 DARPA INTRUSION DETECTION EVALUATION PLAN

2.2.1 Introduction

The 1999 intrusion detection off-line evaluation data set is the second of an ongoing series of yearly evaluations conducted by MIT Lincoln Laboratory ("Lincoln") under DARPA ITO and Air Force Research Laboratory sponsorship. These evaluations are contributing significantly to the intrusion detection research field by providing direction for research efforts and calibration of current technical capabilities.They are of interest to all researchers working on the general problem of workstation, or host-based, and network intrusion detection. The evaluation is designed to be simple, to focus on core technology issues, and to encourage the widest possible participation by eliminating security and privacy concerns and by providing data types that are used by the majority of intrusion detection systems. DARPA 99 data set contains 5 weeks of network traffic data .Each week contains 5 days of network data collected at the packet level .Of these 3 weeks of training data (weeks 1-3, 1st and 3rd weeks do not contain intrusions) and 2 weeks of testing data (weeks 4 and 5).

2.2.2 Technical Objective

The 1999 DARPA evaluation was designed to find the strength and weaknesses of existing approaches and lead to large performance improvements and valid assessments of intrusion detection systems. The concept was to generate a set of realistic attacks, embed them in normal data, evaluate the false alarm and detection rates of systems with these data, and then improve systems to correct the weaknesses found. The following attack events were inserted during the simulation run:

1. Denial of Service (DoS) - Unauthorized attempt to disrupt the normal functioning of a victim host or network.
2. Remote to Local (R2L) - Unauthorized obtaining of user privileges on a local host by a remote user without such privileges.
3. User to Root (U2R) - Unauthorized access to local superuser or administrator privileges by a local unprivileged user.

4. Surveillance or Probe (probe) - Unauthorized probing of a machine or network to look for vulnerabilities, explore configurations, or map the network's topology.
5. Data Compromise (data) - Unauthorized access or modification of data on local host or remote host.

These attacks occur in the context of normal usage of computers and networks as one might observe on a military base. The evaluation is designed to foster research progress, with the following four goals:

1. Explore promising new ideas in intrusion detection.
2. Develop advanced technology incorporating these ideas.
3. Measure the performance of this technology.
4. Compare the performance of various newly developed and existing systems in a systematic, careful way.

2.2.3 Physical Network

The simulation network is divided into two segments representing the networks **inside** an Air Force base and the Internet **outside** the Air Force base as shown in figure 2.1. The outside includes two workstations, which simulate gateways to a virtual outside internet. One workstation simulates many workstations using custom software modifications of the Linux kernel provided by the Air Force group. One gateway leads to roughly 100 workstations and the other leads to 1000's of web sites with actual content that is updated daily. The inside includes victim machines of many types (e.g. Linux, Solaris, and Sun OS) and a gateway to many others inside workstations. Data is collected from the inside victim running Solaris and from an outside sniffer. The list of hosts of simulated network 1999 is attached as Appendix 'B'.

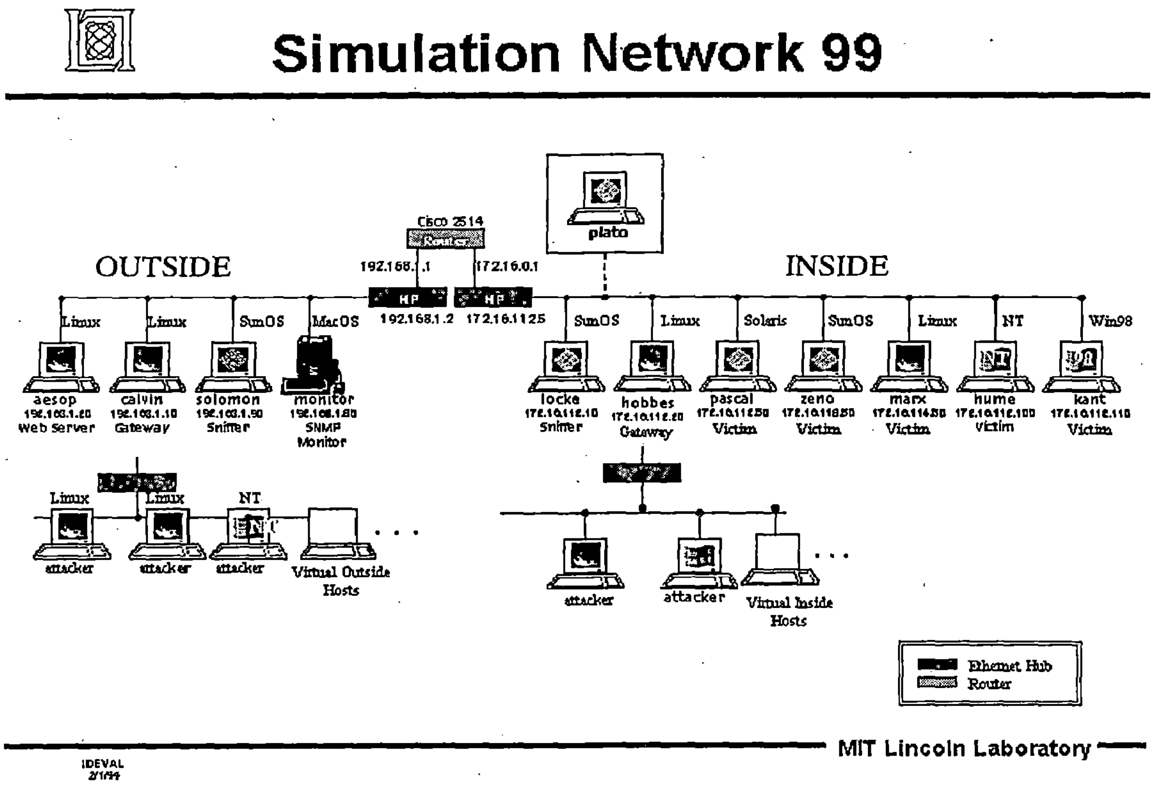


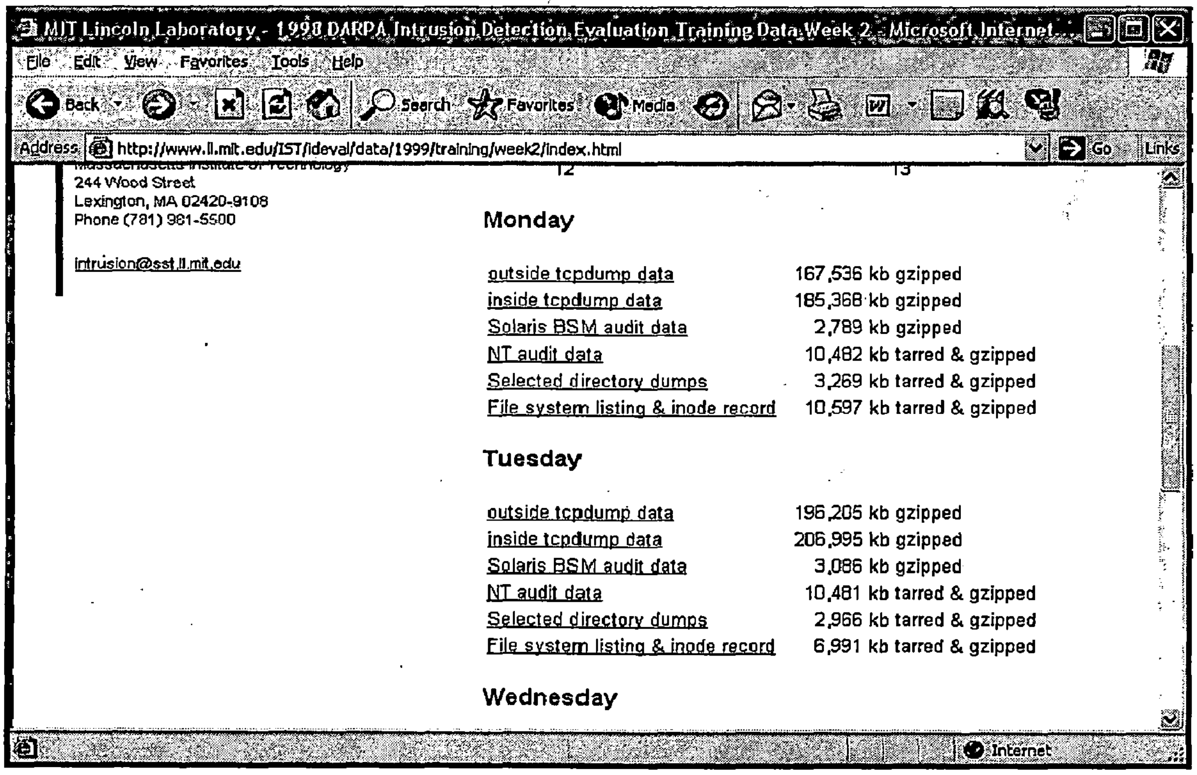
Figure 2.1 Simulation Networks 1999

Many software tools were required to make this approach work. These include many types of traffic generators, tools to schedule and create traffic in real time, and tools to analyze the sniffing and audit data to verify that the system ran correctly and label each attack.

2.2.4 Data layout: day wise

Training data will consist of the following elements however this work will detect attacks based on inside tcpdump data and the second week (attack data) is used for the purpose of training. Figure 2.2 depicts the day wise data for the second week from the dataset. The data available for each day include:

1. Outside tcpdump data for roughly one month of network traffic as collected by a tcpdump packet sniffer. This data contains the contents of every packet transmitted between computers inside and outside a simulated military base.



File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go
Links
Internet

MIT Lincoln Laboratory - 1998 DARPA Intrusion Detection Evaluation Training Data: Week 2 - Microsoft Internet ...

http://www.ll.mit.edu/IST/ideval/data/1999/training/week2/index.html

244 Wood Street
Lexington, MA 02420-9108
Phone (781) 981-5500

intrusion@sst.ll.mit.edu

Monday

outside tcpdump data	167,536 kb gzipped
inside tcpdump data	185,368 kb gzipped
Solaris BSM audit data	2,789 kb gzipped
NT audit data	10,482 kb tarred & gzipped
Selected directory dumps	3,269 kb tarred & gzipped
File system listing & inode record	10,597 kb tarred & gzipped

Tuesday

outside tcpdump data	196,205 kb gzipped
inside tcpdump data	206,995 kb gzipped
Solaris BSM audit data	3,086 kb gzipped
NT audit data	10,481 kb tarred & gzipped
Selected directory dumps	2,966 kb tarred & gzipped
File system listing & inode record	6,991 kb tarred & gzipped

Wednesday

Figure 2.2: Day wise data layout for second week

2. Inside tcpdump data collected by a sniffer located inside the simulated military base.
3. Sun Basic Security Module (BSM) audit data from one UNIX Solaris host. This data contains audit information describing system calls made to the Solaris kernel. Raw BSM binary output files are provided along with BSM configuration files and shell scripts used to initialize BSM auditing to record events from processes that implement important TCP/IP services.
4. Windows NT audit event logs as contained in the three files NTAuditdata, Selected directory dumps, File system listings.

2.2.5 Labeled data

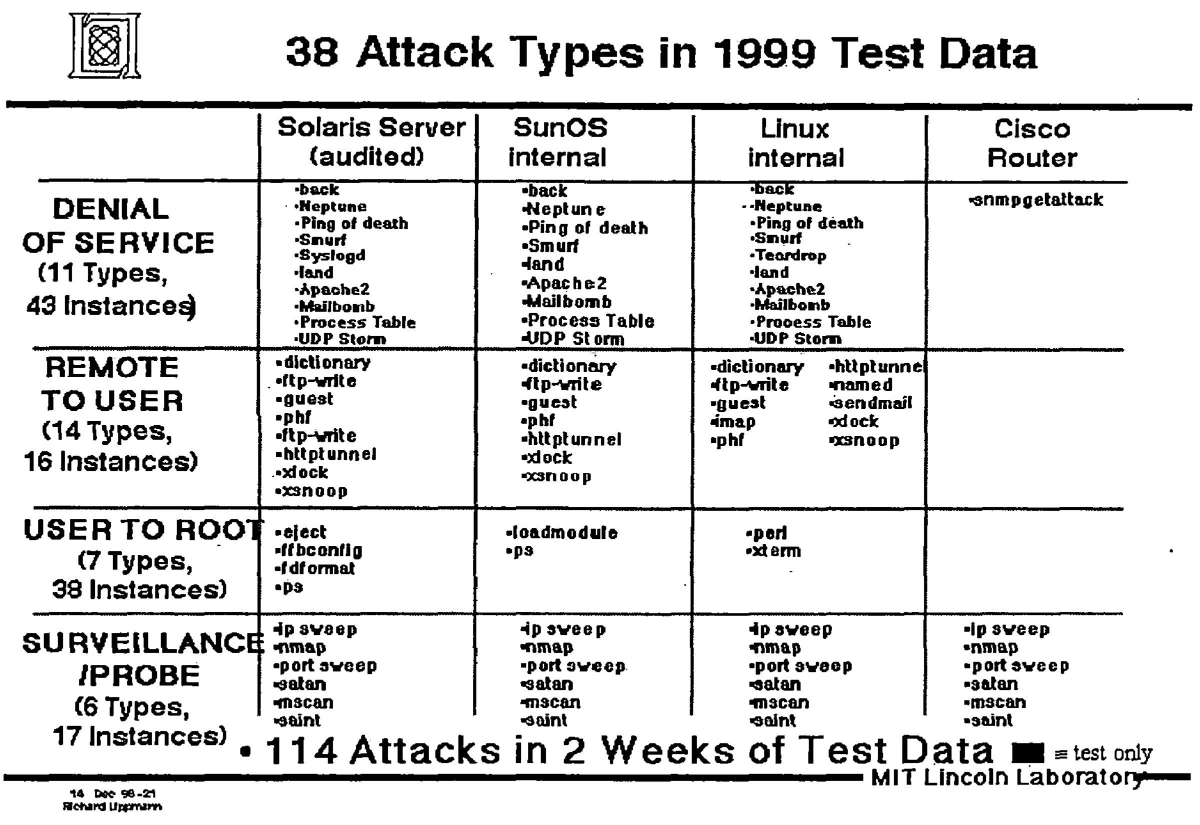
The first three weeks of data are labeled and listed in [19]. The date, starting time, and destination(s) of each attack are provided. In addition, the name of the attack is provided as a source of identification. However the identification of the attacker is not provided, as a result, the simulated network since being a mammoth one has many a source machines accessing the server machine at the timestamp listed in the label as start time of attack. This discrepancy leads to normal activity also being labeled as attack and thereby leading to false negatives. To add to this dilemma it was found that the timestamp provided were inaccurate and a leeway of 60 secs was required, this furthered the tally of false negatives. To overcome this shortfall in our the data is not trained purely on the labeled timestamp but the probability of the source cluster being an attacker is calculated based on number of times it has been labeled as an attack against total number of times the cluster center is selected over the complete training period. This aspect will be better grasped in the later chapters. The fourth and fifth week of data form the test data and the intrusions detected by the system discussed in this thesis can be verified from [20], which catalogs the detect list for test data in the following format:

```
ID: 41.084031
Date: 03/29/1999
Name: ps
Category: u2r
Start_Time: 08:18:35
Duration: 00:46:05
Attacker: 209.154.098.104
Victim: 172.016.112.050
Username: haraldl
Ports:
At_Attacker: 80{1}, 6000{2}
At_Victim: 23{3}
```

2.3 PITFALLS IN DARPA DATASET

On analysis of the DARPA evaluation data set [16] it was found that the 12 million packets in the DARPA training set contain only 8 distinct TTL (Time To Live) values (2, 32, 60, 62-64, 127-128, 254-255). TTL is an 8-bit counter (0-255) that is decremented with each router hop until it reaches zero, in order to prevent infinite routing loops. Most of the detections and all of the false alarms due to TTL result from the anomalous values 126 or 253, which are absent in the training data. This is not realistic, as in real life large variations in TTL values are observed. It is possible that an attacker might manipulate the TTL field to thwart an IDS using methods described by [22], but these techniques involve using small values in order to expire packets between the target and the IDS. A more likely explanation is that the attacks were launched from a real machine that was 2 hops away from the sniffer in the simulation, but all of the other machines were at most one hop away. It is extremely difficult to simulate Internet traffic correctly [21], so such artifacts are to be expected.

Another major drawback realized in the DARPA dataset that affects the application of neural network techniques for detection are that very few instances of an attack is available and more so not all attack types are launched against a single machine as a result a system that learns attacks against individual machines limits the generalization of the neural network learning. Figure 2.3 illustrates the number of attack instances and types. For example there are 43 instances of 11 types of DOS attack therefore at an average only 4 instances of individual attack in the complete dataset which cannot be considered adequate as even missing a single attack will result in 25% failure in result. Maintaining high detection rate, demands near perfect algorithm in detecting attacks.



38 Attack Types in 1999 Test Data

	Solaris Server (audited)	SunOS internal	Linux internal	Cisco Router
DENIAL OF SERVICE (11 Types, 43 Instances)	•back •Neptune •Ping of death •Smurf •Syslogd •land •Apache2 •Mailbomb •Process Table •UDP Storm	•back •Neptune •Ping of death •Smurf •land •Apache2 •Mailbomb •Process Table •UDP Storm	•back •Neptune •Ping of death •Smurf •Teardrop •land •Apache2 •Mailbomb •Process Table •UDP Storm	•snmpgetattack
REMOTE TO USER (14 Types, 16 Instances)	•dictionary •ftp-write •guest •phf •ftp-write •httptunnel •xlock •xsnoop	•dictionary •ftp-write •guest •phf •httptunnel •xlock •xsnoop	•dictionary •ftp-write •guest •imap •phf •httptunnel •named •sendmail •xlock •xsnoop	
USER TO ROOT (7 Types, 38 Instances)	•eject •ffbconfig •fdformat •ps	•loadmodule •ps	•perl •xterm	
SURVEILLANCE /PROBE (6 Types, 17 Instances)	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint	•ip sweep •nmap •port sweep •satan •mscan •saint

• 114 Attacks in 2 Weeks of Test Data ■ ≡ test only

MIT Lincoln Laboratory

14 Dec 98-21
Richard Lippmann

Figure 2.3 Attack types in 1999 DARPA test data

Chapter 3

ATTACKS: HOW TO DETECT

3.1 INTRODUCTION

In this chapter we broadly classify and identify with the variants of attacks with an eye to their attack signatures. It will help reader understand how attacks can be detected by observing the traffic; knowledge of it is must for devising algorithms for datamining module.

There are 4 types/categories of attacks in the DARPA 99 data set:

- Denial of Service (DOS) - An attack that can deny use of a resource or service
- Probe - When network services are used to collect information about host
- User-to-Root (U2R) - a user attacks a computer from inside network
- Remote-to-Local (R2L) - a user attacks a computer from outside network
- Data - someone (user or administrator) performing some action that they may be not allowed as per security policy.

Each of these will be discussed in detail in the subsequent sections.

3.2 Denial of Service Attacks

A denial of service attack is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. There are many varieties of denial of service (or DoS) attacks. Some DoS attacks (like a mailbomb, neptune, or smurf attack) abuse a perfectly legitimate feature. Others (teardrop, Ping of Death) create malformed packets that confuse the TCP/IP stack of the machine that is trying to reconstruct the packet. Still others (apache2, back, syslogd) take advantage of bugs in a particular network daemon.. The following sections describe in detail each of the Denial of Service attacks that were included in the 1999 DARPA intrusion detection evaluation.

3.2.1 Apache2 attack

3.2.1.1 Description: The Apache2 attack is a denial of service attack against an apache web server where a client sends a request with many http headers. If the server receives many of these requests it will slow down, and may eventually crash [23].

3.2.1.2 Attack Signature: Every http request submitted as part of this exploit contains many http headers. Although the exact number and value of these headers could be varied by an attacker, the particular version of the exploit which was used in the 1999 DARPA evaluation sent http GET requests with the header 'User-Agent: sioux\r\n" repeated 10000 times in each request. The actual content of the header is not important for the exploit, the exploit is only dependent on the fact that http request contains many headers. A typical http request contains twenty or fewer headers, so the 10000 headers used by this exploit are quite anomalous.

3.2.2 Back

3.2.2.1 Description: In this denial of service attack against the Apache web server, an attacker submits requests with URL's containing many front slashes. As the server tries to process these requests it will slow down and becomes unable to process other requests [24].

3.2.2.2 Attack Signature: An intrusion detection system looking for the Back attack needs to know that requests for documents with more than some number of front slashes in the URL should be considered an attack. Certainly, a request with 100 front slashes in the URL would be highly irregular on most systems. This threshold could be varied to find the desired balance between detection rate and false alarm rate.

3.2.3 Crashiis

3.2.3.1 Description: CrashIIS is a Denial of Service attack against the NT IIS webserver. The attacker sends a malformed GET request via telnet to port 80 on the NT victim. The command "GET ../.." crashes the web server and sometimes crashes the ftp and gopher daemons as well, because they are part of IIS.

3.2.3.1 Attack signature: Sniffing the network traffic will reveal the malformed GET command. The victim's security audit log will show that Dr. Watson ran when the service(s) crashed. However, Dr. Watson will also run for other reasons. Therefore, using this audit signature for detection will most likely result in false alarms.

3.2.4 dosnuke

3.2.4.1 Description: DoSNuke is a Denial of Service attack that sends Out Of Band data (MSG_OOB) to port 139 (NetBIOS), crashing the NT victim (bluescreens the machine).

3.2.4.2 Attack signature: The attack creates a NetBIOS connection. The packets are flagged "urg" because of the MSG_OOB flag. The attack can be detected by searching the sniffed data for a NetBIOS handshake followed by NetBIOS packets with the "urg" flag.

3.2.5 Land

3.2.5.1 Description: The Land attack is a denial of service attack that is effective against some older TCP/IP implementations. The only vulnerable platform used in the 1999 DARPA evaluation was SunOS 4.1. The Land attack occurs when an attacker sends a spoofed SYN packet in which the source address is the same as the destination address.

3.2.5.2 Attack Signature: The Land attack is recognizable because IP packets with identical source and destination addresses should never exist on a properly working network.

3.2.6 Mailbomb

3.2.6.1 Description: A Mailbomb is an attack in which the attacker sends many messages to a server, overflowing that server's mail queue and possible causing system failure.

3.2.6.2 Attack Signature: An intrusion detection system that is looking for a mailbomb attack can look for thousands of mail messages coming from or sent to a particular user within a short period of time. This identification is a somewhat subjective process. Each site might have a different definition of how many e-mail messages can be sent by one user or to one user before the messages are considered to be part of a mailbomb.

3.2.7 SYN Flood (Neptune)

3.2.7.1 Description: A SYN Flood is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). Each half-open TCP connection made to a machine causes the 'tcpd' server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially open connections. The half-open connections data structure on the victim server system will eventually fill and the system will be unable to accept any new incoming connections until the table is emptied out. Normally there is a timeout associated with a pending connection, so the half-open connections will eventually expire and the victim server system will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim system can terminate the pending connections. In some cases, the system may exhaust memory, crash, or be rendered otherwise inoperative [25].

3.2.7.2 Attack Signature: A Neptune attack can be distinguished from normal network traffic by looking for a number of simultaneous SYN packets destined for a particular machine that are coming from an unreachable host

3.2.8 Ping Of Death

3.2.8.1 Description: The Ping of Death is a denial of service attack that affects many older operating systems. Although the adverse effects of a Ping of Death could not be duplicated on any victim systems used in the 1999 DARPA evaluation, it has been widely reported that some systems will react in an unpredictable fashion when receiving oversized IP packets. Possible reactions include crashing, freezing, and rebooting.

3.2.8.2 Attack Signature: noting the size of all ICMP packets and flagging those that are longer than 64000 bytes can identify an attempted Ping of Death.

3.2.9 Process Table

3.2.9.1 Description: The Process Table attack is a novel denial-of-service attack that was specifically created for this evaluation. The Process Table attack can be waged against numerous network services on a variety of different UNIX systems. The attack is launched against network services that fork () or otherwise allocate a new process for each incoming TCP/IP connection. Although the standard UNIX operating system places limits on the number of processes that any one user may launch, there are no limits on the number of processes that the superuser can create, other than the hard limits imposed by the operating system. Since servers that run as root usually handle incoming TCP/IP connections, it is possible to completely fill a target machine's process table with multiple instantiations of network servers. Properly executed, this attack prevents any other command from being executed on the target machine. An example of a service that is vulnerable to this attack is the finger service. On most computers, finger is launched by inetd. The authors of inetd placed several checks into the program's source code that must be bypassed in order to initiate a successful process attack. In a typical implementation (specifics will vary depending on the actual UNIX version used), if inetd receives more than 40 connections to a particular service within 1 minute, that service is disabled for 10 minutes. The purpose of these checks was not to protect the server against a process table attack, but to protect the server against buggy code that might create many connections in rapid-fire sequence.

3.2.9.2 Attack Signature: Because this attack consists of abuse of a perfectly legal action, an intrusion detection system that is trying to detect a process table attack will need to use somewhat subjective criteria for identifying the attack. The only clue that such an attack is occurring is an unusually large number of connections active on a particular port. Unfortunately 'unusual' is different for every host, but for most machines, hundreds of connections to the finger port would certainly constitute unusual behavior.

3.2.10 Smurf

3.2.10.1 Description: In the "smurf" attack, attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to create a denial-of-service attack. There are three parties in these attacks: the attacker, the intermediary, and the victim (note that the intermediary can also be a victim). The attacker sends ICMP 'echo request' packets to the broadcast address (xxx.xxx.xxx.255) of many subnets with the source address spoofed to be that of the intended victim. Any machines that are listening on these subnets will respond by sending ICMP 'echo reply' packets to the victim. The smurf attack is effective because the attacker is able to use broadcast addresses to amplify what would otherwise be a rather innocuous ping flood. In the best case (from an attacker's point of view), the attacker can flood a victim with a volume of packets 255 times as great in magnitude as the attacker would be able to achieve without such amplification.

3.2.10.2 Attack Signature: The Smurf attack can be identified by an intrusion detection system that notices that there is a large number of 'echo replies' being sent to a particular victim machine from many different places, but no 'echo requests' originating from the victim machine.

3.2.11 sshprocesstable

3.2.11.1 Description: SSH Processtable is similar to the processtable attack in that the goal of the attacker is to cause sshd daemon on the victim to fork so many children that the victim can spawn no more processes. This is due to a kernel limit on the number of processes that the OS will allow.

3.2.11.2 Attack Signature: This attack will be evident due to the large number of rapid ssh connections to the host, the inability of processes to spawn on the host, and the fact that request for new network logins (requiring child processes) will be denied, for the duration of the attack. There may be other obvious signs as well.

3.3 User to Root Attacks

User to Root exploits are a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system. There are several different types of User to Root attacks. The most common is the buffer overflow attack. Buffer overflows occur when a program copies too much data into a static buffer without checking to make sure that the data will fit. For example, if a program expects the user to input the user's first name, the programmer must decide how many characters that first name buffer will require. Assume the program allocates 20 characters for the first name buffer. Now, suppose the user's first name has 35 characters. The last 15 characters will overflow the name buffer. When this overflow occurs, the last 15 characters are placed on the stack, overwriting the next set of instructions that was to be executed. By carefully manipulating the data that overflows onto the stack, an attacker can cause arbitrary commands to be executed by the operating system. The following sections describe each of the User to Root attacks that was used in the 1999 DARPA intrusion detection evaluation in greater detail.

3.3.1 anypw

3.3.1.1 Description: NukePW is a Console User to Root attack that allows the attacker to logon to the system without a password. A boot disk is used to modify the NT authentication package so that a valid username can login with any password string. Logins via telnet also work with any password.

3.3.1.2 Attack signature: The sniffed data will reveal remote logons with incorrect password strings.

3.4 Remote to User Attacks

A Remote to User attack occurs when an attacker who has the ability to send packets to a machine over a network, but who does not have an account on that machine, exploits some vulnerability to gain local access as a user of that machine. There are many possible ways an attacker can gain unauthorized access to a local account on a machine. Some of the attacks discussed within this section exploit buffer overflows in network server software. The following sections provide details of each of these attacks.

3.4.1 HttpTunnel

3.4.1.1 Description: In an Http Tunnel attack, the attacker gains local access to the machine to be attacked and then sets up and configures an http client to periodically query a web server that the attacker has setup at some remote host. When the client connects, the server is able to send cookies that could request information be sent by the client, such as the password file on the victim machine. In effect, the attacker is able to "tunnel" requests for information through the http protocol.

3.4.1.2 Attack Signature: The Http Tunnel attack can be recognized by watching for the setup login session, transfer of the client to the victim, and perhaps setting up a job to be run periodically, or starting a background process to run the client. The using of the tunnel could be noticed by periodic connections to from the victim to the attacker on non-well-known ports, or ports greater than 1024. (However- port 80 could be used as well).

3.4.2 Phf

3.4.2.1 Description: The Phf attack abuses a badly written CGI script to execute commands with the privilege level of the http server. Any CGI program which relies on the CGI function escape_shell_cmd() to prevent exploitation of shell-based library calls may be vulnerable to attack. In particular, the "phf" program that is distributed with the example code for the Apache web server manifests this vulnerability.

3.4.2.2 Attack Signature: To find the Phf attack, an intrusion detection system can monitor http requests watching for invocations of the phf command with arguments that specify commands to be run. Examples of commands that an attacker might attempt to execute by exploiting the phf exploit are: cat /etc/passwd, id, whoami, or xterm.

3.4.3 Sendmail

3.4.3.1 Description: The Sendmail attack exploits a buffer overflow in version 8.8.3 of sendmail and allows a remote attacker to execute commands with superuser privileges. By sending a carefully crafted email message to a system running a vulnerable version of sendmail, intruders can force sendmail to execute arbitrary commands with root privilege.

3.4.3.2 Attack Signature: The Sendmail attack overflows a buffer in the MIME decoding routine of the sendmail program. In order for an intrusion detection system to identify a Sendmail attack it must monitor all incoming mail traffic and check for messages that contain a MIME header line that is inappropriately large.

3.5 Probes

In recent years, a growing number of programs have been distributed that can automatically scan a network of computers to gather information or find known vulnerabilities. These network probes are quite useful to an attacker who is staging a future attack. An attacker with a map of which machines and services are available on a network can use this information to look for weak points. Some of these scanning tools (Satan, saint, mscan) enable even a very unskilled attacker to very quickly check hundreds or thousands of machines on a network for known vulnerabilities. The following sections describe in detail each of the probes that was used in the 1999 DARPA intrusion detection evaluation

3.5.1 Ipsweep

3.5.1.1 Description: An Ipsweep attack is a surveillance sweep to determine which hosts are listening on a network. This information is useful to an attacker in staging attacks and searching for vulnerable machines.

3.5.1.2 Attack Signature: An intrusion detection system looking for the simple Ipsweep used in the simulation can look for many Ping packets, destined for every possible machine on a network, all coming from the same source.

3.6 Data

Data Attacks involve someone (user or administrator) performing some action that they may be able to do on a given computer system, but that they are not allowed to do according to site policy. Often, these attacks will involve transferring "secret" data files to or from sources where they don't belong

3.6.1 Secret

3.6.1.1 Description: A "secret" attack is an attack where the attacker maliciously or mistakenly transfers data which they have access to a place where it doesn't belong. For example, transferring data from a classified computer/network to a non-classified computer/network would constitute a "secret" attack.

3.6.1.2 Attack Signature: To recognize these attacks, the detection system must know which files are considered "secret", what the policies are regarding use of these files, and then simply look for actions carried out involving them. Naturally, attacks such as these can be hard to detect - a legitimate user could "cut-and-paste" information from one desktop window to another.

Chapter 4

METHODOLOGY

4.1 GENERAL DESCRIPTION

The system designed is a network based intrusion detection system that scrutinizes tcpdump data in a time window to develop traffic trends and characteristic evidences of an attack. It is assumed that the evidences of an attack lie within the packets and can be identified either by individual analysis of packet in some cases or by ascertaining the attackers intention by analyzing sequel of packets in a time window frame for others. Since the system is primarily an anomaly based detection technique which detects attack by detecting deviations from its learned normal trait, hence the efficacy of the system to detect abnormality will depend on datamining skills in extracting features from packet/packets such that they form a distinctive trait for each attack. Akin to misuse detection technique the data-mining module also extracts attack signatures of known attacks. The presence of attack signatures in the payload of the packet is either flagged or mapped to a specific numeric value, which acts as an attack identifier. The abnormality is self-learned by the system by way of Kohonen based Self organizing mapping techniques. The clustering mechanism maps the activity of individual machines within the window to clusters indicative of behavior pattern. Prior to the system being used some of the components need time to be trained on the traffic traits. The system is decidedly modular in nature; the system block diagram is illustrated in figure 4.1 and detail system flowchart is depicted in Figure 4.2. There are three distinctive stages in the system namely the **architectural learning stage, System learning stage** and the **detection stage.**

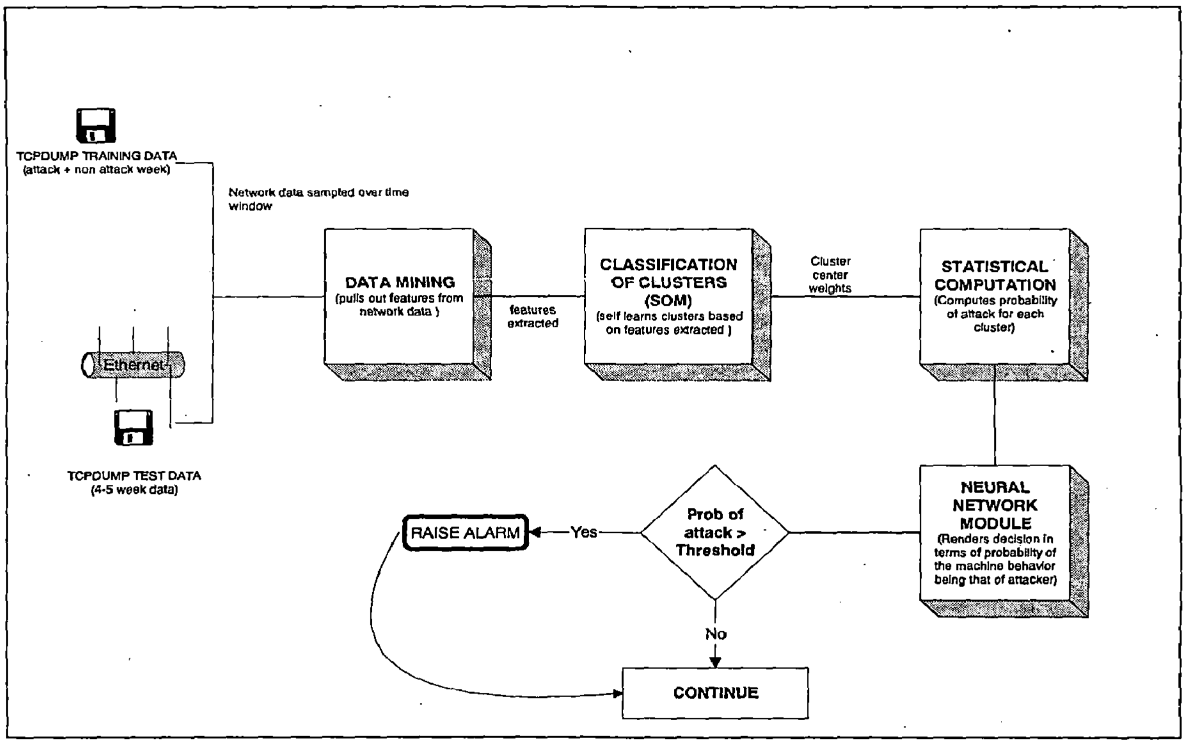

TCPDUMP TRAINING DATA
(attack + non attack week)
Network data sampled over time window
Ethernet
TCPDUMP TEST DATA
(4-5 week data)
DATA MINING
(pulls out features from network data)
features extracted
CLASSIFICATION OF CLUSTERS (SOM)
(self learns clusters based on features extracted)
Cluster center weights
STATISTICAL COMPUTATION
(Computes probability of attack for each cluster)
NEURAL NETWORK MODULE
(Renders decision in terms of probability of the machine behavior being that of attacker)
Prob of attack > Threshold
Yes
RAISE ALARM
No
CONTINUE

Figure 4.1 Block diagram of system

In the architectural learning period the architecture of the SOM and MLFFBP neural network are decided, as well as the number of ports to be monitored and the services offered by the server or the list of valid ports are also learned. Based on the structure decided in this phase the features from the tcpdump data are extracted and the data preprocessed via datamining skills over a time window to reveal the behavior of the machine, which then is fed to the system learning stage where both the number of clusters are ascertained and based on the labeled tcpdump data [27] the normal/abnormal behavior of the cluster learned by the MLFFBP neural network. Once these courses of action have been taken the system evolves into detection mode. In the subsequent sections we will dwell on these stages in detail.

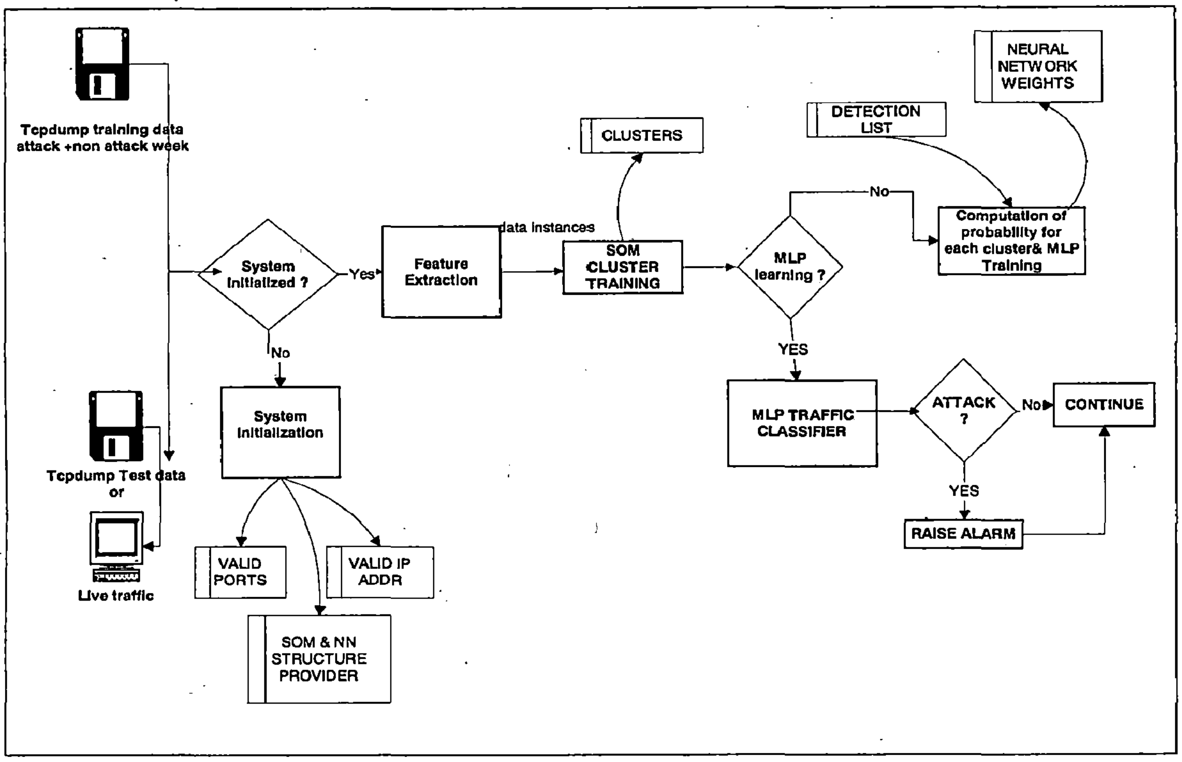


Figure 4.2 System Flowcharts

The system is designed to detect intrusions against a few servers within the monitored network at any instance all the same since the clusters learned and the neural network trained are on behavior traits of source machine and are independent of machine specific features viz. source IP, destination IP, the system can be easily carried forward to other networks by adopting to specific machine attributes like valid list of ports in use.

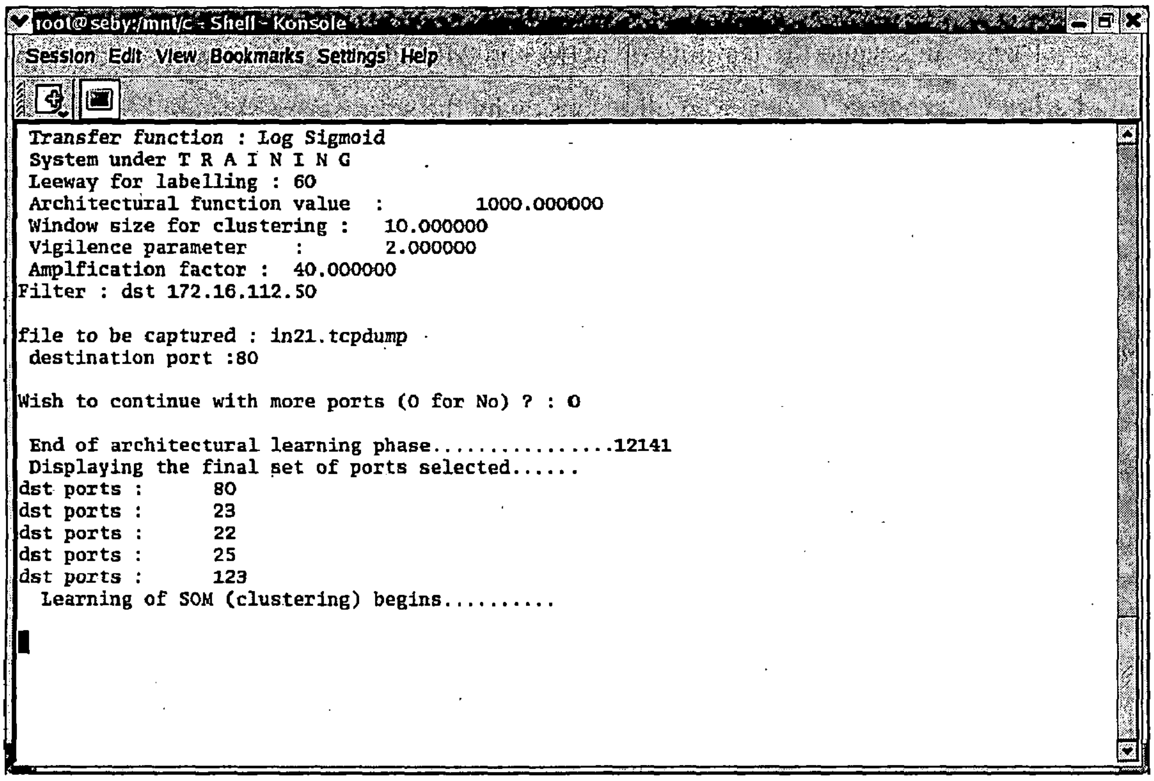
4.2 ARCHITECTURAL LEARNING STAGE

This stage lays the outline for the core work to follow in successive stages, in this stage the structure of the SOM and MLFFBP neural network is ascertained. The structure of both the neural networks is direct fallout of the number of ports to be monitored and the number of features being extracted in a time window. To monitor all the 65,536 ports [28] of the victim machine (server) is unnecessary as only few ports would be active depending on the services being offered by the server; also the administrator would prefer to monitor some vulnerable ports. The number of ports to be monitored is arrived at as the sum of list of known ports (say *KP*) defined by the system administrator to watch and the number of extra ports (*EP*) the algorithm chooses to add to the list. The algorithm employed is based on selecting the highest EP number of ports that are accessed the most in the observed network data. Since the system is primarily designed for detecting attacks against server victim machines hence a criteria of port number being less than 1024 is added. The final set of ports used is *FINALSET = KP + EP.*

In this stage the list of valid ports are computed as all the ports accessed in a non-attack tcpdump data viz. 1st, 3rd week of the victim machine. This is required for detecting variants of probe attacks in which a server attempts to access inactive or invalid ports on a server machine. In the preprocessing module the port activity over the time window is computed based on the *FINALSET* number of ports on a source by source basis, this form as one of the features for the clustering mechanism with the other features extracted from the network data using data mining skills discussed in chapter 5. Each or the combination of these other features extracted contribute towards detecting attacks. Viz. number of invalid/inactive ports accessed within a time window will be instrumental in detecting all probe attacks. As an attacker trying to hit upon the services offered by the victim machine would tend to scan all its ports thereby escalating the score of invalid ports accessed by this attacker machine on the victim machine. If *NUM_FEATURES* is the total number of features used for clustering then it is defined as

$$NUM\_FEATURES = FINALSET + OTHER\_F$$

Where *OTHER_F* are the number of features extracted besides port activity, this is based on the number of attacks that the system is designed to detect, for experimental purpose and to limit the scope of discussion we would be considering six such features, all the same the strength of the system in detecting other attacks by increasing these features will be considered as future scope. Figure 4.3 depicts the screen shot at the end of architectural learning stage in the prototype system. Port No. 80 was specified by the system administrator and 4 Extra ports were decided by the algorithm.


root@seby:/mnt/c - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
 Transfer function : Log Sigmoid
 System under T R A I N I N G
 Leeway for labelling : 60
 Architectural function value  :        1000.000000
 Window size for clustering :   10.000000
 Vigilence parameter    :       2.000000
 Amplfication factor :  40.000000
Filter : dst 172.16.112.50

file to be captured : in21.tcpdump
 destination port :80

Wish to continue with more ports (0 for No) ? : 0

 End of architectural learning phase................12141
 Displaying the final set of ports selected......
dst ports :      80
dst ports :      23
dst ports :      22
dst ports :      25
dst ports :      123
  Learning of SOM (clustering) begins..........
```

Figure 4.3. Screen shot depicting end of Architectural learning stage

Once we have determined the *NUM_FEATURES* the SOM structure is decided as having NUM_FEATURES number of input neurons and the architecture of MLFFBP neural network is as shown in figure 4.4. The discussion on number of hidden layers and its nodes is carried forward for chapter 7.

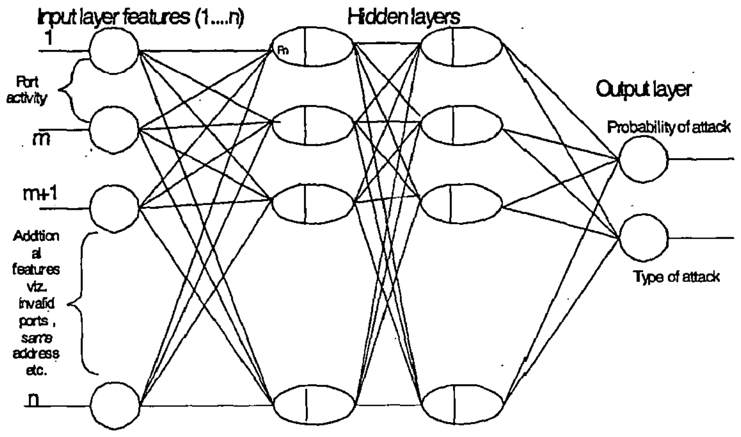

Input layer features (1....n)
Hidden layers
1
Port activity
m
m+1
Additional features viz. invalid ports, same address etc.
n
Pn
Output layer
Probability of attack
Type of attack

Figure 4.4 MLFFBP neural network structure

4.3 SYSTEM LEARNING STAGE

During this stage the learning of SOM and MLFFBP neural network take shape. But before we progress further we will discuss why the sources are clustered for input to the neural network. The NUM_FEATURES computed on source-by-source basis over the time window reflects the behavioral pattern of that source machine and could have been directly fed to the neural network eluding the use of clustering technique. But at any time there are a large number of sources in communication with the victim machine. Therefore simply grouping the information by sources will firstly not create a uniform representation of data for the neural network and more significantly since the labels in tcpdump data are based on timestamps and as numerous sources are concurrently in communication during that instant so all the sources or rather their behavior patterns will be fallaciously classified as attacks thereby making the training of neural network problematic i.e. since the identical behavior pattern would be trained as an attack in one time window and as a non attack in another thereby leading to a convergence problem. To address this issue we send the preprocessed source information to a clustering module which groups sources with similar trends together during the system learning phase. The numbers of thus created clusters are constant and fixed in this phase. When a source is assigned to a particular cluster, the clusters center is updated using this sources features; more on the SOM technique is discussed in chapter 6.

The second part of the system learning stage involves supervised training of the neural network to render a decision as to the likelihood of a pending attack. In this stage the tcpdump data are first preprocessed in a source by source basis within a time window and the features extracted fed to the clustering algorithm which selects the cluster closest to the source machine behavior and then this cluster depending on the tcpdump data label is classified as an attack/non attack for the supervised learning of the MLFFNN. The timestamps give out the machine under attack and the time at which attack was effective and as discussed earlier since numerous sources are concurrently in communication with the victim machine and therefore all these would be erroneously classified as attacks if labeled solely on the basis of timestamps. Besides the clustering algorithm or more specifically the datamining technique may not always succeed in allotting a separate cluster for each anomalous behavior, as a consequence normal activity may also fall in the same cluster thus leading to higher false alarms.

This impasse is resolved by conducting the supervised training in two stages. In the first stage the probability of each cluster being an attack is calculated by computing the number of times a cluster was labeled as an attack as against the number of times the cluster center was selected. The neural network for supervised training then employs this individual probability of each cluster; the subject is dealt in more detail in chapter 7.

4.4 DETECTION STAGE

In this stage the system can be put in detection mode for testing either on tcpdump data file (4-5 week) or live network data for detection. In case of latter, the valid list of ports of the machines to be monitored will require to be maintained in valid_port.txt (Appendix 'J'), while for the former the system generates the same during training itself. In this work we would limit our scope the former.

The tcpdump data of the test week is preprocessed within the time window and on the basis of its proximity to the clusters learned by the system, the cluster center is selected which is then fed to the neural network for decision in terms of the probability of the activity of that machine in that span corresponding to an attack and classifying the type of attack Based on the state of alertness viz. cautious, alert, paranoid the alarms are raised and logged by the system.

Chapter 5

DATAMINING MODULE

5.1 GENERAL

The datamining module preprocesses the network data both during training and detection stage and presents those features to the system that empower it to detect intrusions. It is also referred to as the preprocessing module in this document. The datamining skills employed during this stage define the efficacy of the system. An effort to identify how the data needs to be looked at in order to provide us with a better picture is surely vital in providing accurate and effective results. This is done in four sub-phases;

(a) System Initialization

(b) Packet Capture : Sniffing program

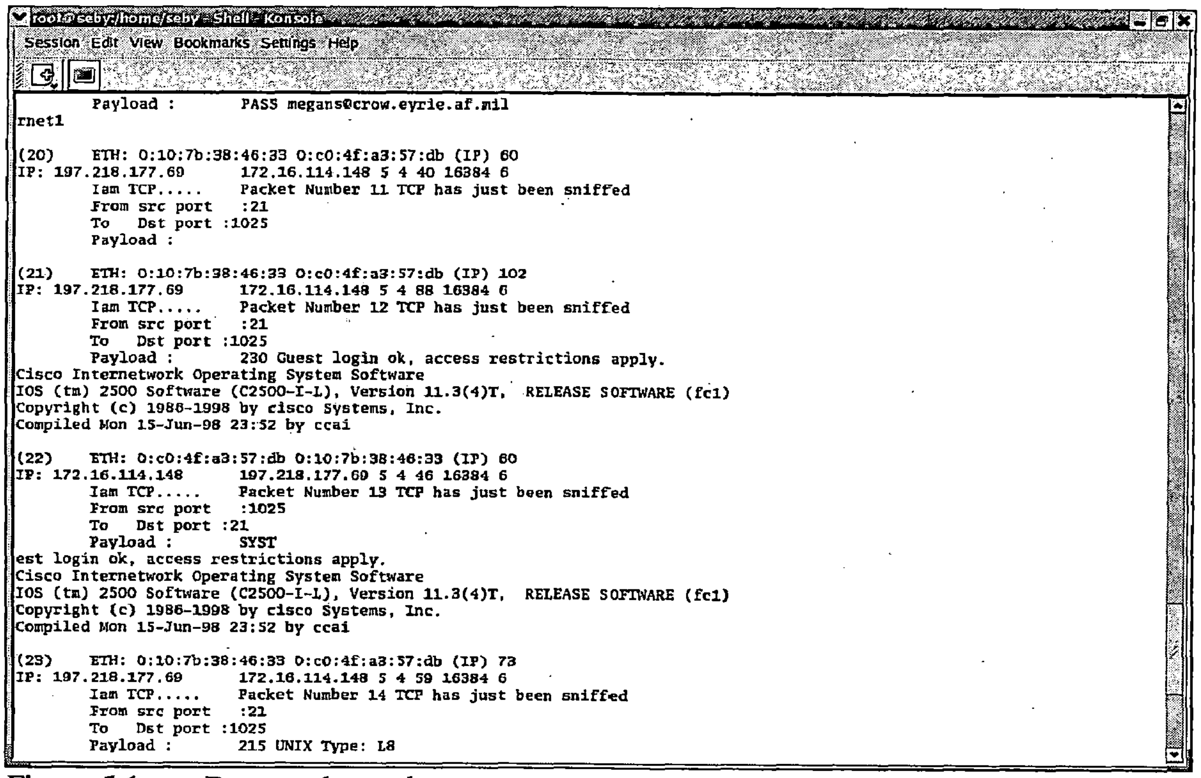
(c) Feature Extraction : Data mining Algorithm

(d) Normalization

In the System Initialization phase the parameters of the system with respect to the victim or Server machine being monitored is initialized as explained in Architectural learning stage, i.e. the number of ports to be monitored, list of valid ports on the victim machine ascertained by preparing the list of ports active on the server. These parameters are ascertained by analyzing the packets from the non-attack week training data (week 3) for a period defined by the architectural factor (*archi_factor*). The packets captured in the first 10,000 seconds were considered during architectural learning stage i.e. *archi_factor=10000.* The parameters initialized in this stage will aid the datamining algorithm in pulling out anomalies in machine activity.

The sniffing program captures the packets from the training or the test tcpdump files and only IP packets are considered to limit the scope of this work. Based on the protocols the packet header and application layer fields are extracted and this raw field values processed by the datamining algorithm to extract features for clustering. Figure 5.1 illustrates the screen shot of raw tcpdump data captured by the sniffing program. In [14]

33 fields were analyzed for detecting attacks and only 15 fields were found tc contribute towards detection. Table 5.1 lists the contribution of these fields.


root@seby:/home/seby - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
        Payload :        PASS megans@crow.eyrie.af.mil
rnet1

(20)    ETH: 0:10:7b:38:46:33 0:c0:4f:a3:57:db (IP) 60
IP: 197.218.177.69       172.16.114.148 5 4 40 16384 6
        Iam TCP.....     Packet Number 11 TCP has just been sniffed
        From src port    :21
        To   Dst port :1025
        Payload :

(21)    ETH: 0:10:7b:38:46:33 0:c0:4f:a3:57:db (IP) 102
IP: 197.218.177.69       172.16.114.148 5 4 88 16384 6
        Iam TCP.....     Packet Number 12 TCP has just been sniffed
        From src port    :21
        To   Dst port :1025
        Payload :        230 Guest login ok, access restrictions apply.
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-I-L), Version 11.3(4)T,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1998 by cisco Systems, Inc.
Compiled Mon 15-Jun-98 23:52 by ccai

(22)    ETH: 0:c0:4f:a3:57:db 0:10:7b:38:46:33 (IP) 60
IP: 172.16.114.148       197.218.177.69 5 4 46 16384 6
        Iam TCP.....     Packet Number 13 TCP has just been sniffed
        From src port    :1025
        To   Dst port :21
        Payload :        SYST
est login ok, access restrictions apply.
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-I-L), Version 11.3(4)T,  RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1998 by cisco Systems, Inc.
Compiled Mon 15-Jun-98 23:52 by ccai

(23)    ETH: 0:10:7b:38:46:33 0:c0:4f:a3:57:db (IP) 73
IP: 197.218.177.69       172.16.114.148 5 4 59 16384 6
        Iam TCP.....     Packet Number 14 TCP has just been sniffed
        From src port    :21
        To   Dst port :1025
        Payload :        215 UNIX Type: L8
```

Figure 5.1 Raw tcpdump data

The sniffing program displays only the Ethernet source and destination address, Ethernet Header size, IP src and dst, packet size, protocol field value, src and dst port numbers and the data payload.

Field	TP	Dup	F P	Detected attacks
Ethernet Size	1	2	1	Ipsweep
Ethernet Dst Hi	1	0	6	Mscan
IP TTL	33	8	20	Netcat_breakin,netbus,ntinfoscan,dosnuke,queso, casesen,satan,apache2,mscan,mailbomb,ipsweep, ppmacro,Neptune,sechole,crashiis,named,smurf, portsweep,guesstelnet.
IP Packet length	2	2	1	Teardrop,portsweep,satan
IP dst address	2	1	7	Portsweep,sendmail
TCP Flags UAPRSF	7	3	2	Queso,portsweep,dosnuke
TCP window size	0	1	2	Apache2

TCP checksum	1	0	29	Insidesniffer
TCP URG Ptr	3	0	5	Dosnuke
TCP options	2	0	4	Apache2
UDP checksum	2	0	0	Udpstorm
ICMP checksum	2	0	0	Smurf

Table 5.1 Contribution of fields to detection in [14].

It is evident from the table that the source or destination port did not contribute towards detection of attacks. But by applying data mining skills and computing statistical parameters using the ports, the system proposed in this work is able to detect variants of probe attacks and flooding attacks. As will be seen later in this report it also detects new attacks, which were not available in the training tcpdump data.

In the Feature extraction stage, the algorithm for heightening the anomalous nature in network traffic is worked out, and then the numbers of features predefined by the algorithm are extracted over a dynamic time window, whose width depending upon the nature of network activity is decided by the system administrator such that a high activity would lead to a shorter width. The network data features are then normalized to portray a correct representation of network behavior. This then forms as input to the clustering mechanism.

5.2 DATAMINING ALGORITHM

5.2.1 Introduction

According to R.L. Grossman in [28], he defines data mining as being "concerned with uncovering patterns, associations, changes, anomalies, and statistically significant structures and events in data." In simple terms it is the ability to take data and pull from it patterns or deviations which may not be seen easily to the naked eye. Another term sometimes used is knowledge discovery. While they will not be discussed in detail in this report, there exist many different types of data mining algorithms to include link analysis, clustering, association, rule abduction, deviation analysis, and sequence analysis.

Data mining can help improve intrusion detection by adding a level of focus to anomaly detection. By identifying bounds for valid network activity, data mining will aid the system to distinguish attack activity from common everyday traffic on the network. A major obstacle was to tailor data mining algorithms and processes to fit intrusion detection. The ensuing section will discuss these aspects with certain attacks in mind.

5.2.2 Feature extraction

To limit the scope of discussion in this section we will discuss feature extractions leading to detection of at least one or more attacks with high detection rate and near zilch false positive rate. The attacks chosen are those that were not or poorly detected by [17]. The network has large number of machines concurrently in communication with the victim machine and the statistics is computed on a source by source basis in each time interval by this module. The statistics computed are the features extracted from the group of packets originating from a source machine to the victim machine in a time interval. The subject is discussed in this section from the perspective of attack detection.

5.2.2.1 Port activity

This foremost feature is computed over the time window by calculating the traffic intensity to all the ports being monitored. The selection of ports is dealt in detail in section 4.2. The port activity reflects the behavior pattern of the source machine against the victim machine in a particular time window. To corroborate the point lets consider detection of mailbomb attack on a mail server. An enormous amount of mails being sent to the victim machine (mail server) by an attacker thereby overflowing the servers mail queue and possibly lead to system failure would get highlighted as an anomalous activity by computing the port activity in this case a high traffic intensity at port 25. This feature can detect variants of flooding attacks provided that the destination port on which the attack is launched by the attacker is under surveillance by the system.

5.2.2.2 Probe attack

The basic intention of an attacker launching a probe attack is to hit upon the active services running on the victim machine so that depending on his expertise in exploiting the bugs in the services identified, an attack can be launched. The detection of this attack lies in this activity itself; since the attacker is unaware of the services offered by the victim machine and hence would probe all the ports in the band of his interest. This attack is detected by our system by identifying if any source machine attempts to access an inactive port on the server machine in a suspicious manner.

During the architectural stage of the system, it learns from the non-attack tcpdump file the active ports of servers and this list of valid ports is stored in a file (valid_port.txt). The feature extracted to detect the probe attack viz. portsweep is computed by calculating the number of invalid ports accessed by each source machine during a time window. A large number of invalid ports accessed by a source machine during a time frame would be indicative of a probe attack.

5.2.2.3 Land attack

Since no machine would need to use the network resources to access itself therefore a packet on the network with the same source and destination address is anomalous in nature. This attack is fairly simple to detect in our system unlike in [17] since the activity of all the source machines against a specific destination machine is under purview in a time interval. This attack is detected by flagging the feature 'same_address ` as true when the destination and source addresses were similar.

5.2.2.4 Back and Crashiis attack

For detail description of attack refer section 3.2.2 and 3.2.3 respectively. These attacks can be detected by scanning through the HTTP (port 80) packets for data contents having more than 100 front slashes or malformed GET requests. In such case the feature 'key' is loaded a predefined value (0.9 for back and 0.5 for crashiis) in the above case and otherwise a default value (0).

5.2.3 Conclusion

Once the features are extracted based on the attacks being detected by the system it is required to be normalized. Since the feature port activity and number of invalid ports accessed cannot be purely seen from a time window aspect as the traffic intensity varies from day to day and more so on hourly basis. Highly anomalous port traffic intensity at 0600hrs may not be considered so at 1100 hrs as network activity is time dependent. Normalizing these features will present a correct picture so that the system is able to detect the deviations from the normal.

5.3 NORMALIZATION

Since the system was designed to be general, it must be able to create clusters given a dataset from an arbitrary distribution. The data instances are assigned to clusters if they are closer than a constant distance; which defines the neighborhood boundary. If this neighborhood boundary or vigilance parameter is hard coded into the algorithm, then it will be used for data instances from other distributions as well. To highlight the discrepancy caused thereby lets consider two sets of two 3-feature vectors, each set coming from different distribution:

1. {(1, 3, 2), (1, 4, 3)}
2. {(900, 1000, 700), (1000, 1100, 800)}

Under a Euclidean metric, the squared distance between feature vectors in the first set will be $(1\text{-}1)^2 + (3\text{-}4)^2 + (2\text{-}3)^2 = 2$, while it will be 30,000 for the second set. If in our clustering algorithm the cluster width is hard coded (e.g. say 1.5) then the two patterns in the first set will fall in the same cluster while the patterns in the second set will be put in different clusters. Since we cluster the input data instances in an adaptable time window, the distribution will differ over the time windows itself, as the traffic pattern during early hours will differ from peak hours.

One possible solution to this is to determine the width dynamically based on the dataset (based on early /peak hours the distribution varies) and the width of the time window. However, it would result in complexities in training of SOM and MLP neural network. In this work we take a different approach [8] and make the viglance parameter/cluster width hard coded constant (user defined) but convert the data instances to a standard form based on training dataset's distribution and time window width. Instead of presenting the traffic patterns within a time window directly we map it to a standard space by computing their mean values and for every feature value we calculate how many standard deviations it is away from the average and this result becomes the new value for their feature.

Given a training dataset, the average and standard deviation feature vectors are calculated:

$$Avg\_vector[j] = 1/N \sum_{i=1}^{N} instance_i[j]$$

$$Std\_vector[j] = ((1/(N-1) \sum_{i=1}^{N} (instance_i[j] - avg\_vector[j])^2)^{1/2}$$

Where *avg_vector[j]* is the average of the j^{th} feature of the vector and *instance*$_i$*[j]* is the value of j^{th} feature of i^{th} pattern in the dataset.

Then each feature vector in the dataset is converted as follows:

$New\_instance[j] = (instance_i[j] - avg\_vector[j])/std\_vector[j]$

In effect this is a transformation of an instance from its own space to a standardized space, based on statistical information retrieved from the dataset over the time window. These normalized features pulled out from network traffic in a time window are then presented to the clustering module.

CLUSTERING MODULE BASED ON SOM

6.1 GENERAL

The unsupervised learning using self-organizing mapping (SOM) techniques, clusters the input patterns according to similarities discovered among the input features [33]. The clustering process is governed by a threshold called vigilance parameter (cluster width) and metric function. To create clusters from the input features, we use a simple variant of single-linkage clustering – Kohonen self-organizing neural network. It is an unsupervised neural network which maps multi- dimensional inputs to a two dimensional outputs [29]. Kohonen proposed new neural network architecture based on the idea, that brain uses spatial mapping to model complex data structure internally. This architecture is popularly known as Kohonen Self-Organizing Map.

The input features presented by the data-mining module of the system prototype, for clustering is depicted in Table 6.1. * Features are those that are flagged or mapped into numeric value indicating presence of attack signatures or anomalous activity. Total of five ports were monitored which included four Extra ports ascertained by the algorithm as discussed in the data mining module (Chapter 5).

Features presented for clustering				
Port Activity (1....m)	Same address *	TTL	Invalid port*	Payload*

Table 6.1 Input Features presented by data mining module for clustering

6.2 METRIC

One of the main assumptions made was that data instances presented by the datamining module having similar characteristic would be close together under some metric. Therefore finding or constructing an appropriate metric is essential for clustering. In detecting network intrusions, it is imperative that some features of the data instances would be more important (have greater weights) than others viz. flagging of existence of attack signatures, and thus differences in the values of those features should have a greater contribution to the overall distance. Therefore, several-weighted metrics were tried, with higher weights assigned to different subsets of features. Figure 6.1 depicts a comparative study, note combination 1 represents equal weighted features while rest are weighted features as listed in Appendix 'G'. Combination 2 and 3 accord higher weight to features that are flagged i.e. invalid port (indicates inactive port on server machine being accessed), same address (indicates anomalous IP src and dst address) and key (indicates presence of attack signature in data payload), resulting in better detection rate than the rest of the weight combinations.

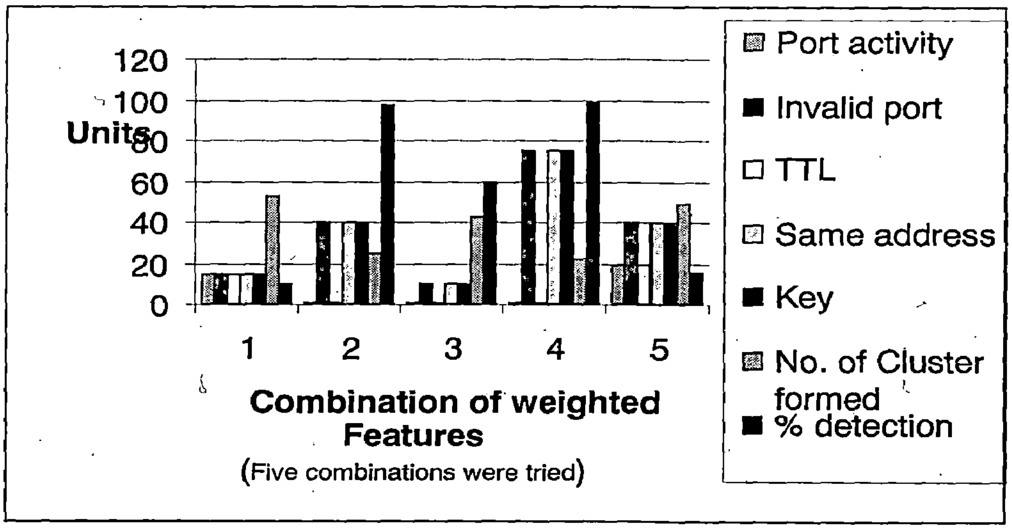


Figure 6.1 Comparative study of effect of weighted features on efficacy of clusters

As a consequence it was decided to use a standard Euclidean metric, with weighted features so that all data instances pertaining to an attack fall into the same cluster i.e. not many clusters representing the same attack are formed. Further tuning the metric will show some increase in performance, however tuning the metric parameters to achieve maximum performance for a particular data distribution and feature set would undermine the systems generality and contribute to over fitting; hence only critical features (those flagging presence of attack signatures) were weighted by a factor referred to as amplification factor in this report.

6.3 CLUSTERING ALGORITHM

Kohonen network consists of an input layer and a two dimensional Kohonen layer, which maps a distribution of *n*-dimensional *np* inputs onto *mp* output nodes in a non-linear way. The algorithm employs competitive learning rule, first pattern is selected as the centre of the first cluster. Then, the next pattern is compared to the first cluster centre. If the distance is less than vigilance parameter, it is clustered with the first. Otherwise, it is a centre of a new cluster. This process is repeated for all the patterns. The algorithm for the clustering mechanism is summarized below.

Let N_1 is the number of input features, P is the number of patterns in the training set and N_2 is the number of clusters. The upper bound of N_2 is P. If m ports are being monitored then $(N_1 - m)$ input features are besides the port activity.
Assume $X_i^{(p)}$ $(i= 1,2\ldots\ldots N_1)$ to be the i^{th} feature of the input pattern $(p=1,2,\ldots\ldots P)$ and b_k to be the centre of the cluster k then

$$b_k(n_k) = [b_{k1}, b_{k2,\ldots\ldots\ldots}\ b_{k\ n1}]^T$$

where n_k indicates the number of patterns that belong to the cluster k and $k = 1,2,\ldots\ldots,N_2$.

Step 1: Set p=1, $N_2=0$, $n_{N2}=0$

Step 2: Form the new cluster and set

$N_2 = N_2 + 1$, $n_{N2}=1$

The cluster centre co-ordinates are calculated with the help of following equation

$b_{N2}i(n_{N2}) = X_i^{(p)}$ *for* $i= 1,\ldots., N_1$

Step 3: Increment p by 1 and if p . P, determine the Euclidean distance ED_k between the pattern p and centre of the cluster k, b_k *for* $k=1, 2\ldots\ldots, N_2$

$$ED_k = amp\_factor * \ sqrt. \sum_{i=1}^{N_1} (b_{kl} - X_i^{(p)} \}^2 \ldots$$

for i=1 to (N_1-m) the amp_factor =1 i.e. equal weighted metric for port activity

Step 4: Find k such that

$ED_k = min \{ ED_k \}$

Step 5: If ED_k . ., go to Step 6 else go to step 2.

Step 6: Pattern p belongs to the cluster K, new co-ordinates of cluster centre are calculated as

$b_{kl}(n_{k+1}) = (n_k/(n_{k+1}))\, b_{kl}(n_k) + (1/(n_{k+1}))\, X_i^{(p)}$

and go to step 3.

6.4 DETECTION OF NEW ATTACKS

The clustering algorithm will form clusters based on the inherent relation between the data instances (behavior of source machines over the time window). Based on the training data and the efficacy of the datamining technique to pull out features indicative of an attack the clustering algorithm will form clusters pinpointing to normal and abnormal behavior of machine activity. If the number of clusters formed is fixed during SOM training then subsequently a data instance in test data or live traffic corresponding to a new attack will be clubbed to the closest learned cluster and thereby go undetected or be falsely detected.

It is one of our main assumptions that the training data set used for learning of SOM and MLP neural network is exhaustive in terms of normal behavior of machines and therefore any data instance during detection stage not within the cluster width of all learned clusters is in itself indicative of an attack and training of neural network follows this thinking which empowers our system to detect new attacks. This is implemented by forming a dummy cluster at the end of cluster learning which adaptively learns its cluster centre from that data instance which does not fall into the neighborhood boundary of all learned clusters.

6.5 FUNCTIONALITY

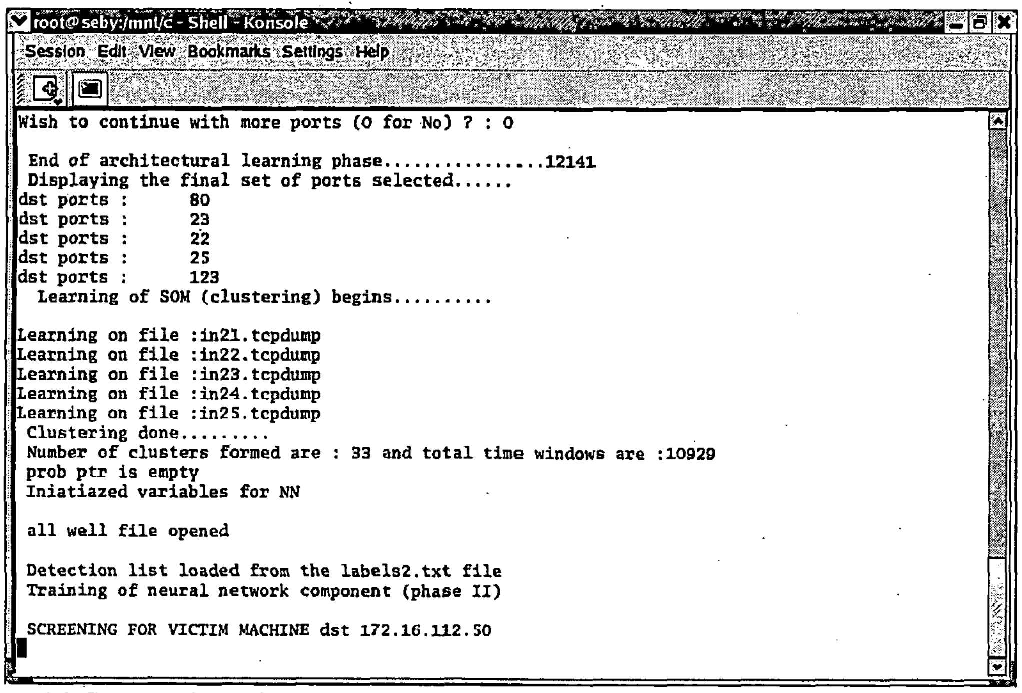
During training stage (refer system flowchart for period) the data instances are mapped to various clusters using clustering algorithm mentioned in section 6.3 and the total number of clusters formed include the dummy cluster created to accommodate for new attacks. These clusters learned i.e. cluster centre weights are averaged weightof the group of data instances it represents. These cluster centers are written to a file cluster.txt (refer Appendix 'K'). Samples of clusters formed are listed in table 6.2. The system was trained on second week of tcpdump training data and a total of 33 clusters were formed including dummy cluster as depicted from the screen shot in Figure 6.2. The tuning parameters applied for clustering are listed below:

Vigilence parameter = 2

Amplification factor = 40

Cluster No.	Port Activity					Payload	TTL	Same address	Invalid port
	Port 80	Port 23	Port 22	Port 25	Port 123				
1	0	-0.13012	-0.17197	-0.00735	1.355499	0	0	0	0
2	-0.00049	-0.29451	-0.18617	-0.06904	-0.4267	0	0	0	0
3	0	-0.16786	1.038775	-0.01494	-0.61176	0	0	0	0
4	0	-0.31647	-0.20605	1.820978	-0.41398	0	0	0	0
5	0	1.202854	-0.17113	-0.03989	-0.50339	0	0	0	0
6	0	-0.35717	-0.24381	-0.11111	-0.46136	0	0	0	1.5
7	0	-0.49228	-0.25609	-0.17432	-0.36959	0	0	0	2.474874
8	0	1.54969	-0.33087	-0.03981	-0.445	0	0	0	1.788854
9	0	-0.55884	2.277098	0	-0.38293	0	0	0	2.03266
10	-0.00039	-0.395	-0.23195	-0.12742	1.82802	0	0	0	-0.79611

Table 6.2. Samples of clusters formed


root@seby:/mnt/c - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
Wish to continue with more ports (0 for No) ? : 0

 End of architectural learning phase................12141
 Displaying the final set of ports selected......
dst ports :      80
dst ports :      23
dst ports :      22
dst ports :      25
dst ports :      123
  Learning of SOM (clustering) begins..........

Learning on file :in21.tcpdump
Learning on file :in22.tcpdump
Learning on file :in23.tcpdump
Learning on file :in24.tcpdump
Learning on file :in25.tcpdump
 Clustering done.........
 Number of clusters formed are : 33 and total time windows are :10929
 prob ptr is empty
 Iniatiazed variables for NN

 all well file opened

 Detection list loaded from the labels2.txt file
 Training of neural network component (phase II)

 SCREENING FOR VICTIM MACHINE dst 172.16.112.50
```

Figure 6.2 Screen shot of program depicting the clusters formed

During neural network training and detection stage the clusters learned ae read from this file and based on the proximity to the cluster; the cluster centre selected. The weights of the cluster centre selected are then fed to the neural network module.

Chapter 7

MLFF NEURAL NETWORK MODULE USING LEVENBERG MARQUARDT ALGORITHM FOR BACK PROPOGATION

7.1 GENERAL

The center weights of clusters selected are the input to the neural network under both training and detection stage. The neural network learns both normal cluster centers as well as those that point towards a pending attack in two phases. It firsts computes the probability of a cluster center being characteristic of an attacker using the labeled data. Once that being done it is then used to train the neural network to learn the attribute of each clusters selected. The attributes of clusters learned by the neural network include the probability of a cluster being indicative of an attack and its classification.

7.2 NEURAL NETWORK ARCHITECTURE

The system employs multi layer feed forward back propagation (MLFFBP) neural network. It contains an input layer, one or more hidden layers and an output layer. An MLFFBP Neural network has strong generalization capabilities and has been successfully applied to solve difficult and diverse problems. The neural networks are widely considered as an efficient approach to adaptively classify patterns, but the high computation intensity and the long training cycles in our case greatly hinder their application. Hence a faster convergence algorithm viz. Levenberg Marquart algorithm was used for backpropogation of weights. The algorithm is discussed in the section 7.3.

The neural network linearises the nonlinear relationship between the input and output vectors. More complex the relationship (complexity increases with number of non-linear relations) more number of layers are required to map the relation, also the number of hidden nodes vary depending upon the number of output and input vectors. The ideal number of hidden nodes that were arrived for our system was based on the following relation. .

No. of hidden nodes = ½(input vectors + output vectors)

The point is brought out well in figure 7.1. During this chapter all such comparative study is conducted on a sample case i.e. training neural network on second week of DARPA training data (approximately 2.4 GB of data) for server pascal.eyrie.af.mil (172.16.112.50) using log sigmoid transfer function. Both output and input data are normalized within a range of 0-1 and the activation function was found to give better results with log sigmoid transfer function. The figures are based on tables attached as appendices in this report.

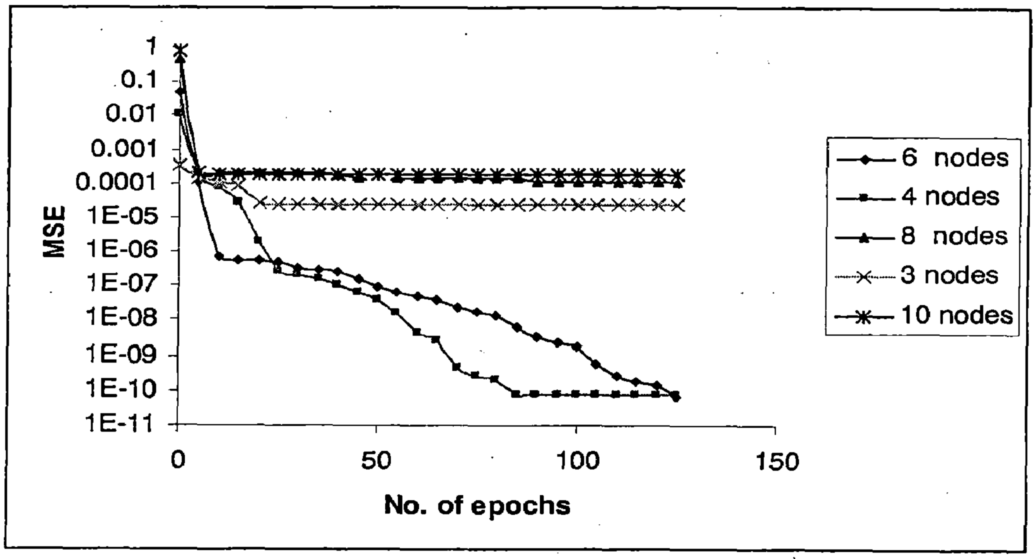


Figure 7.1 Effect of number of nodes in hidden layer for sample case of two hidden layers

The numbers of layers in the hidden layers were decided based on the characteristic of the training data. A comparative study on number of layers in hidden layer performed for the aforesaid example case is given in figure 7.2.

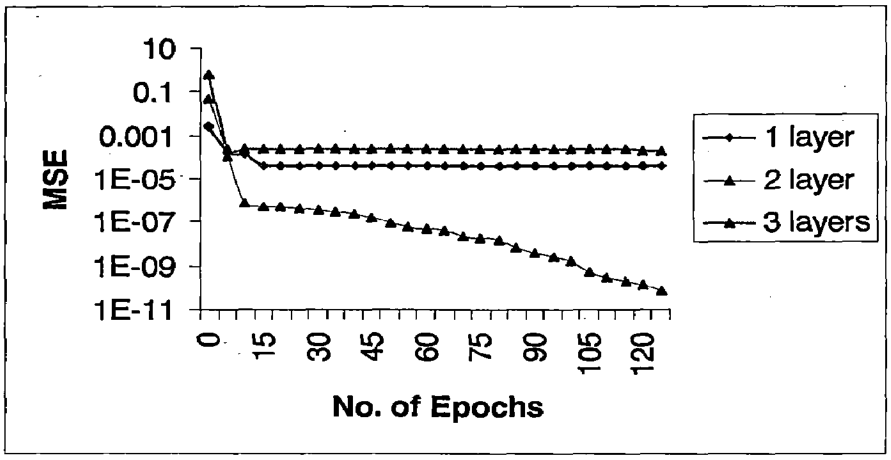


Figure 7.2 Comparative study of effect of number of hidden layers on error reduction

The system prototype employed two hidden layers each consisting of 6 nodes, the number of nodes in the input layers consisted of 9 features and the number of nodes in the output layer consisted of two nodes giving the probability of attack and the identifying the attack type as shown in figure 7.3. The training data and the target patterns for the neural network were stored in trgset.txt and target.txt. Samples of input and output patterns are listed in table 7.1.

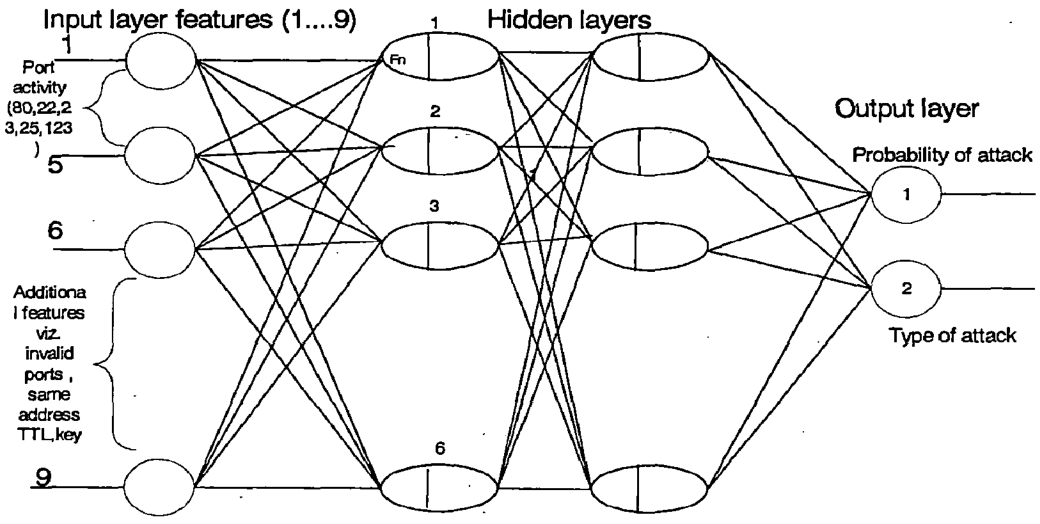

Input layer features (1....9)
Hidden layers
Output layer
Probability of attack
Type of attack
Port activity (80,22,23,25,123)
Additional features viz. invalid ports, same address TTL,key
1
5
6
9
Fn
1
2
3
6
1
2

Figure 7.3. Architecture of MLFFBP Neural Network

Input Patterns								
Port Activity					Other Features			
0.377401	0.308289	0.78512	0.370059	0.324814	0.377401	0.377401	0.377401	0.44089
0.377401	0.564314	0.350808	0.371202	0.299178	0.377401	0.377401	0.377401	0.377401
0.377324	0.331637	0.348471	0.366672	0.311095	0.377401	0.377401	0.377401	0.377401
0.377401	0.564314	0.350808	0.371202	0.299178	0.377401	0.377401	0.377401	0.377401
0.377324	0.331637	0.348471	0.366672	0.311095	0.377401	0.377401	0.377401	0.377401
0.377324	0.331637	0.348471	0.366672	0.311095	0.377401	0.377401	0.377401	0.377401
0.377401	0.357181	0.350677	0.376259	0.588034	0.377401	0.377401	0.377401	0.377401
0.377401	0.357181	0.350677	0.376259	0.588034	0.377401	0.377401	0.377401	0.377401
0.377401	0.564314	0.350808	0.371202	0.299178	0.377401	0.377401	0.377401	0.377401
0.377324	0.331637	0.348471	0.366672	0.311095	0.377401	0.377401	0.377401	0.377401

Table 7.1. Samples of training and target patterns used byNeural Network

Target Patterns	
Probability of attack	Attack identifier
0.003607	0
0.003607	0
0.006981	0
0.003049	0
1	1
0.003374	0

7.3 MARQUARDT ALGORITHM

Since the backpropogation learning algorithm [30] was first popularized, there has been considerable research on methods to accelerate the convergence of the algorithm. In multilayer perceptron networks, the most often encountered among these methods are modifications of backpropogation as error backpropogation with adaptive learning rate and momentum; conjugate gradient, quickprop, etc. All these algorithms can be considered as variations of the steepest descent method, because they only use information of the objective function and its gradient. The most popular approaches from the second category have used conjugate gradient methods. Another area of numerical optimization that has been applied to neural networks is nonlinear least squares. These are considered to be more efficient but their storage and computational requirements go up as the square of the size of the network. However, for networks with few hundred weights that is the case with our system the algorithm is very efficient when compared with conjugate gradient techniques. Levenberg Marquardt algorithm is an approximation to Newton's method. Based on the value of a parameter ., in each epoch the algorithm shifts i.e. when . is large the algorithm becomes steepest descent while when . is small it becomes Gauss-Newton method. The Marquart algorithm is discussed in detail in [31]. A comparison with other methods is drawn in figure 7.4 which highlights the suitability of Levenberg Marquart backpropogation algorithm in the neural network module of our system.

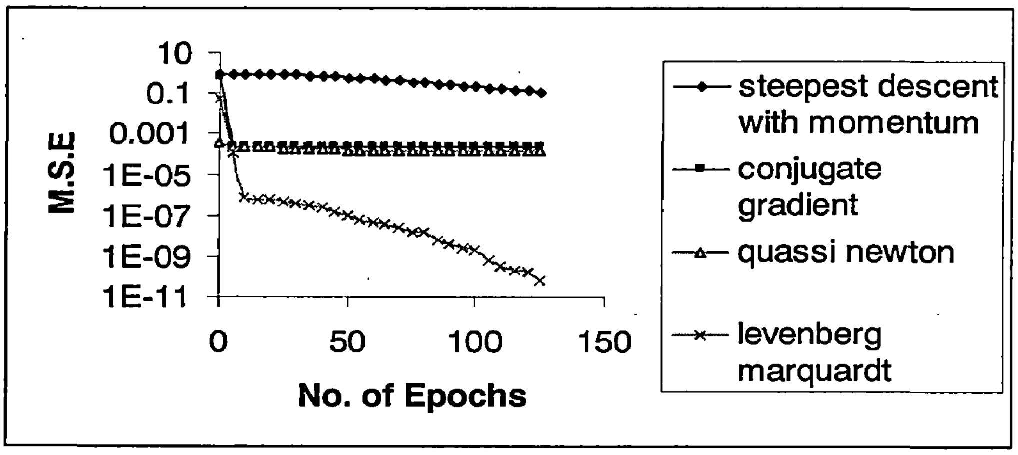


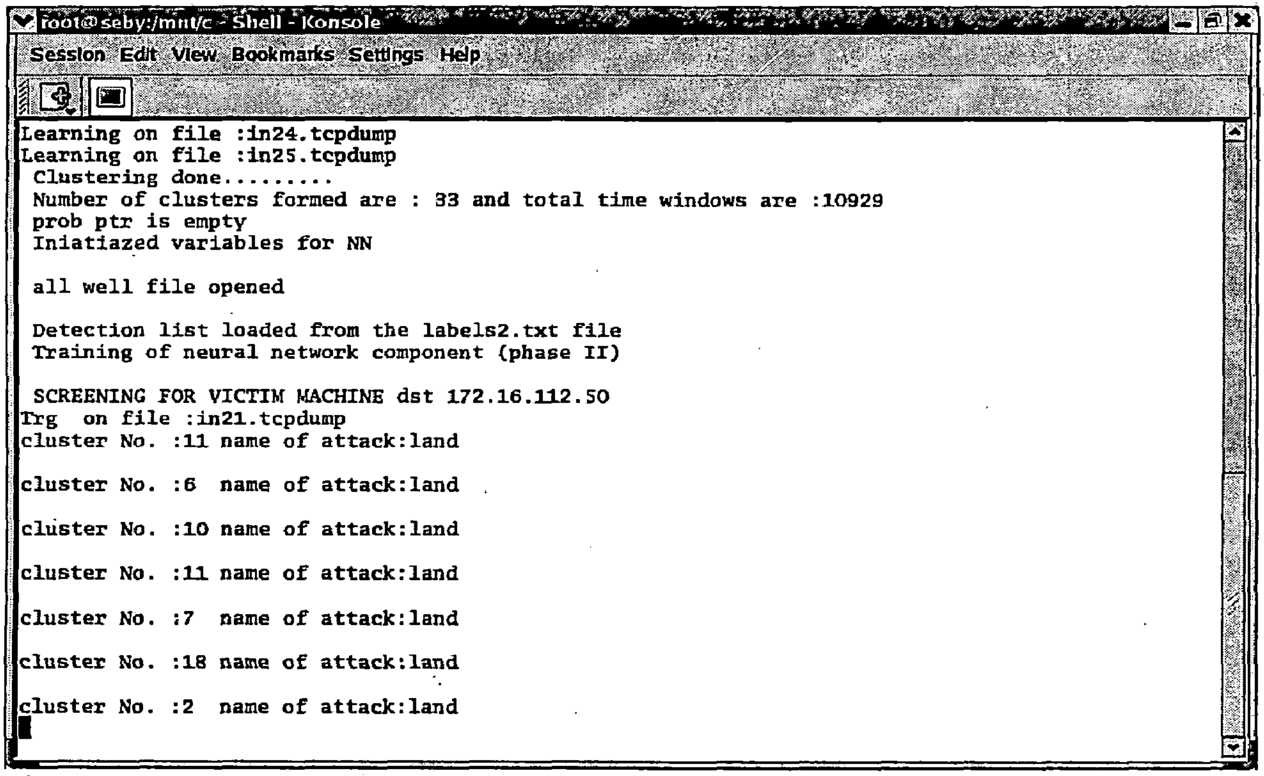
Figure 7.4 Comparison of various backpropogation methods used in neural network training

7.4 TRAINING NEURAL NETWORK

The neural network trains on the data instances that are the centers of the cluster selected by the clustering module. The training is supervised using the labeled tcpdump training data; the labeled data gives out the timestamp of the attack and the IP addr of the victim machine. A sample case is enumerated below in which the cluster 18 is correctly labeled as land attack, but since the other packets are also within the leeway of 60 seconds they too are erroneously labeled as attack. Figure 7.4 shows the other clusters (except cluster No. 18) being erroneously labeled as attack based purely on timestamp provided by labeled data.

```
** timestamp cluster 03/08/1999 15:57:07   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.112.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :18   name of attack:land
        20.000000
 (80) : 0.000000
 (23) : -0.466072
 (22) : -0.408248
 (25) : 0.000000
 (123) : -0.408248
 key : 0.000000
 TTL : 0.000000
 same addr : 1.000000
 invalid port : -0.461069
** timestamp cluster 03/08/1999 15:58:12   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.113.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:land
       20.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
learning  on file :in22.tcpdump
```

The pitfalls in the labeled data and their fallout on our system are discussed in section 2.2.5 and 4.3 respectively. To outweigh these shortcomings supervised learning is conducted in two phases. In the first phase the training is conducted in batch mode and based on the labeled data the number of times a cluster has been labeled as an attack and the type of attack is maintained. At the end of the first phase the probability of a cluster being an attack is computed based on the number of times it was labeled as attack against

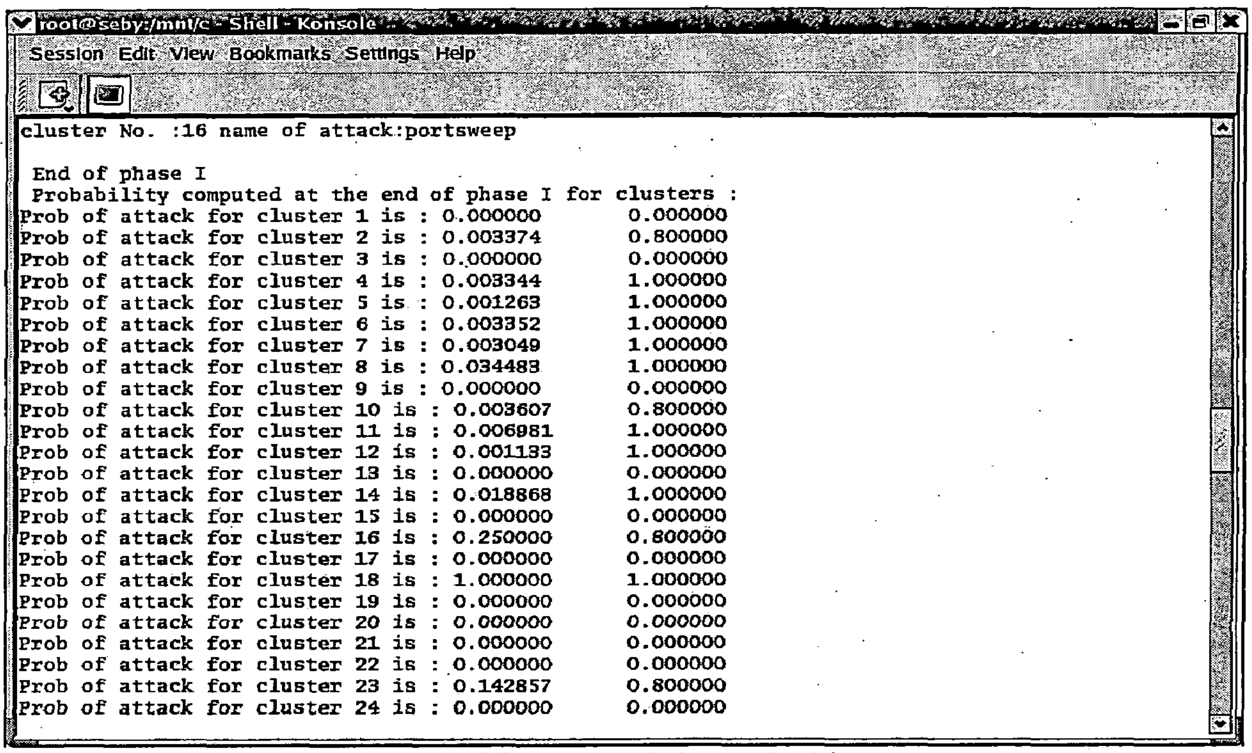

root@seby:/mnt/c - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
Learning on file :in24.tcpdump
Learning on file :in25.tcpdump
 Clustering done.........
 Number of clusters formed are : 33 and total time windows are :10929
 prob ptr is empty
 Iniatiazed variables for NN

 all well file opened

 Detection list loaded from the labels2.txt file
 Training of neural network component (phase II)

 SCREENING FOR VICTIM MACHINE dst 172.16.112.50
Trg  on file :in21.tcpdump
cluster No. :11 name of attack:land

cluster No. :6  name of attack:land

cluster No. :10 name of attack:land

cluster No. :11 name of attack:land

cluster No. :7  name of attack:land

cluster No. :18 name of attack:land

cluster No. :2  name of attack:land
```

Figure 7.4. Labeling of clusters as attack based on labeled tcpdump detection list

the number of times it was selected by the clustering module. By doing so firstly a indiscriminate opinion of a cluster being indicative of an attack behavior is ascertained i.e. since at any time several machines are concurrently in communication with the victim machine and since the labeled data does not specify the attacker machine so our system will label all these machines rather cluster centers having the same timestamp as attack, the probability computation will rule out those clusters wrongly labeled by assigning very low probability of being an attacker. Secondly the false positives due to leeway in timestamp will also be nullified by this method.

The neural network was trained to learn the behavior of a source machine (emulated by cluster centers) as indicative of an attack and to classify the type of attack. The actual supervised training of neural network is done in the second phase in which using the probability computed of each cluster, the first output attribute of the neural network is trained as shown in figure 7.5. Note that all the clusters which were maliciously labeled, their probabilities computed overcome the shortcoming as discussed. Also the probability of cluster 18 being an attack is 1. The name of attack launched on the victim


root@seby:/mnt/c - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
cluster No. :16 name of attack:portsweep

 End of phase I
 Probability computed at the end of phase I for clusters :
Prob of attack for cluster 1 is : 0.000000       0.000000
Prob of attack for cluster 2 is : 0.003374       0.800000
Prob of attack for cluster 3 is : 0.000000       0.000000
Prob of attack for cluster 4 is : 0.003344       1.000000
Prob of attack for cluster 5 is : 0.001263       1.000000
Prob of attack for cluster 6 is : 0.003352       1.000000
Prob of attack for cluster 7 is : 0.003049       1.000000
Prob of attack for cluster 8 is : 0.034483       1.000000
Prob of attack for cluster 9 is : 0.000000       0.000000
Prob of attack for cluster 10 is : 0.003607      0.800000
Prob of attack for cluster 11 is : 0.006981      1.000000
Prob of attack for cluster 12 is : 0.001133      1.000000
Prob of attack for cluster 13 is : 0.000000      0.000000
Prob of attack for cluster 14 is : 0.018868      1.000000
Prob of attack for cluster 15 is : 0.000000      0.000000
Prob of attack for cluster 16 is : 0.250000      0.800000
Prob of attack for cluster 17 is : 0.000000      0.000000
Prob of attack for cluster 18 is : 1.000000      1.000000
Prob of attack for cluster 19 is : 0.000000      0.000000
Prob of attack for cluster 20 is : 0.000000      0.000000
Prob of attack for cluster 21 is : 0.000000      0.000000
Prob of attack for cluster 22 is : 0.000000      0.000000
Prob of attack for cluster 23 is : 0.142857      0.800000
Prob of attack for cluster 24 is : 0.000000      0.000000
```

Figure 7.5 Computation of Probability of attack for each cluster

machine is available in the labeled data which is mapped into a numeric value between 0.4 -1.0 to classify the type of attack (refer column 2 of fig 7.5). The choice of the above range is taken since for a cluster center of non-attack kind the second output attribute (i.e. type of attack) would be zero. Since in the first phase each time a cluster is labeled as an attack it's type is also learned but for those clusters which were wrongly labeled for reasons discussed above though the probability computed rules out the cluster being an attack but the type of attack (numeric value in range 0.4-1.0) learned leads to convergence problem for the network learning. Since for a feed forward network having two outputs as in our case and specially incase of data instance pertaining to falsely labeled cluster i.e. having a very low probability of attack example 0.00004 and trained as a back attack (numeric value mapped is say 0.5) will lead to convergence problem since the weights till the last hidden layer is mostly common for both the output layer attributes and it's the weights between the last hidden layer and output layers that has to adapt weights to learn this diversified range of output attributes (0.00004 and 0.5).

Once again data mining skills came to our rescue, by applying a rule based condition at the end of first phase those clusters having a very low probability of being an attack, its type of attack is classified as non-attack (numeric value mapped is zero). By doing so the convergence problem earlier noticed was by far resolved. Figure 7.6 illustrates the effect of applying datamining skills on our example case for better appreciation.

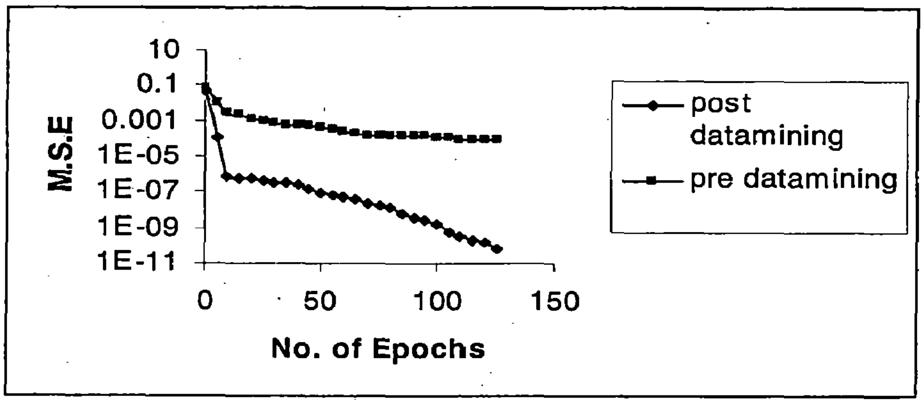
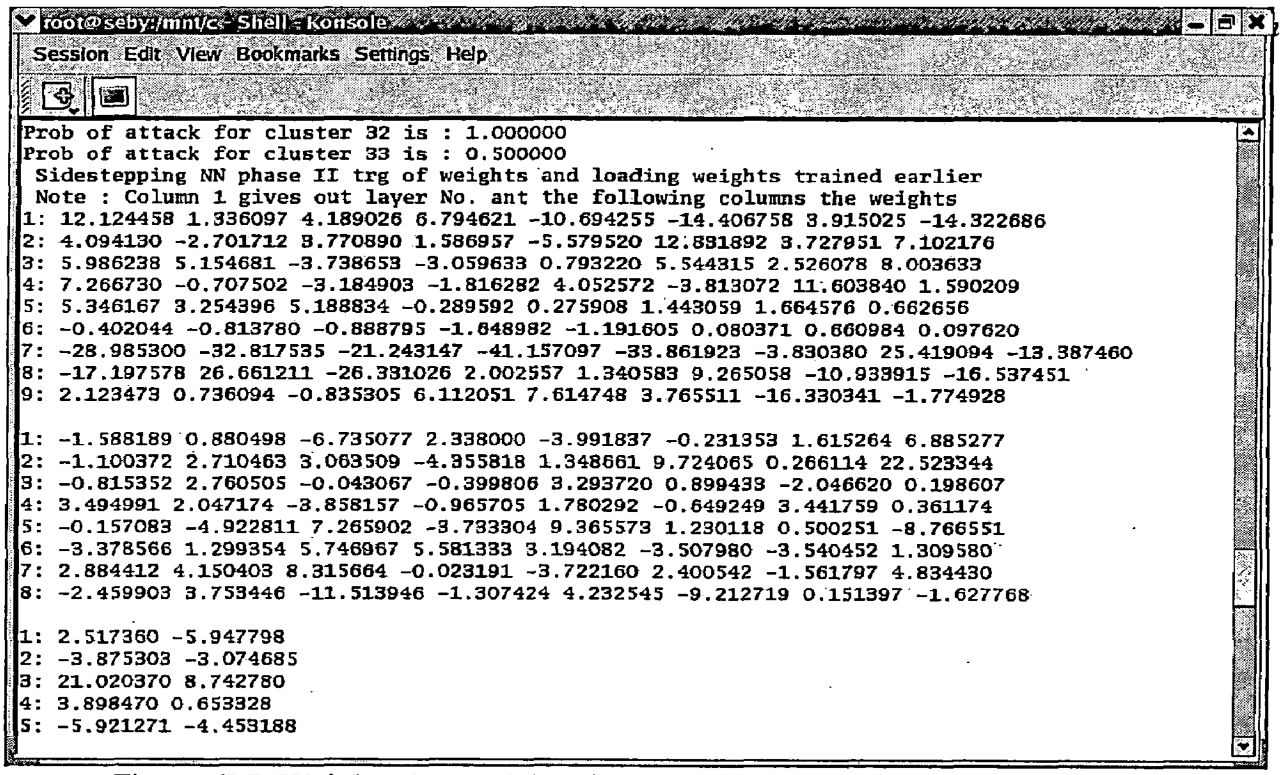


Figure 7.6 Comparative study: Effects of datamining techniques on neural network convergence

At the close of training phase the weights learned of the neural network are stored in file weights.txt (refer Appendix 'M') as shown in figure 7.7 and the system is geared up to fall in detection mode.

7.5 DETECTION MODE: NEURAL NETWORK

In this stage the system prepares itself for detection by loading the clusters centers and MLFFBP weights learned during training phase from cluster.txt and weights.txt files. The data mining module extracts the features from the test tcpdump data or live traffic over the time window and presents it to the clustering module, which in turn based on the proximity to cluster centers learned selects the cluster center weight to this data instance and forwards it to the neural network module for rendering decision on pending attack. The neural network module equipped with the weights learned during training phase


root@seby:/mnt/c - Shell - Konsole
Session Edit View Bookmarks Settings Help

```
Prob of attack for cluster 32 is : 1.000000
Prob of attack for cluster 33 is : 0.500000
 Sidestepping NN phase II trg of weights and loading weights trained earlier
 Note : Column 1 gives out layer No. ant the following columns the weights
1: 12.124458 1.336097 4.189026 6.794621 -10.694255 -14.406758 3.915025 -14.322686
2: 4.094130 -2.701712 3.770890 1.586957 -5.579520 12.831892 3.727951 7.102176
3: 5.986238 5.154681 -3.738653 -3.059633 0.793220 5.544315 2.526078 8.003633
4: 7.266730 -0.707502 -3.184903 -1.816282 4.052572 -3.813072 11.603840 1.590209
5: 5.346167 3.254396 5.188834 -0.289592 0.275908 1.443059 1.664576 0.662656
6: -0.402044 -0.813780 -0.888795 -1.648982 -1.191605 0.080371 0.660984 0.097620
7: -28.985300 -32.817535 -21.243147 -41.157097 -33.861923 -3.830380 25.419094 -13.387460
8: -17.197578 26.661211 -26.331026 2.002557 1.340583 9.265058 -10.933915 -16.537451
9: 2.123473 0.736094 -0.835305 6.112051 7.614748 3.765511 -16.330341 -1.774928

1: -1.588189 0.880498 -6.735077 2.338000 -3.991837 -0.231353 1.615264 6.885277
2: -1.100372 2.710463 3.063509 -4.355818 1.348661 9.724065 0.266114 22.523344
3: -0.815352 2.760505 -0.043067 -0.399806 3.293720 0.899433 -2.046620 0.198607
4: 3.494991 2.047174 -3.858157 -0.965705 1.780292 -0.649249 3.441759 0.361174
5: -0.157083 -4.922811 7.265902 -3.733304 9.365573 1.230118 0.500251 -8.766551
6: -3.378566 1.299354 5.746967 5.581333 3.194082 -3.507980 -3.540452 1.309580
7: 2.884412 4.150403 8.315664 -0.023191 -3.722160 2.400542 -1.561797 4.834430
8: -2.459903 3.753446 -11.513946 -1.307424 4.232545 -9.212719 0.151397 -1.627768

1: 2.517360 -5.947798
2: -3.875303 -3.074685
3: 21.020370 8.742780
4: 3.898470 0.653328
5: -5.921271 -4.453188
```

Figure 7.7 Weights learned by the neural network using Levenberg-Marquardt algorithm

indicates the probability of the cluster center (behavior of the source machine in the time window) being that of an attacker and if so classifies the type of attack.

For exercising control over the number of false positives being generated by the system the system can be set to one of the three states i.e. Normal, alert or Paranoid. These states in turn set the threshold for the attack probability, above that the system generates an alarm. These thresholds are user defined and 0.75, 0.5 and 0.3 have been used in this work. The probability of being an attack computed for each cluster center is explained in the earlier section, this is done as in the training tcpdump data sufficient instances of those attacks were available. However, in case of new attacks where the clustering module will select the dummy cluster and load the features presented by the data-mining module as it's cluster center, this then is fed to the neural network module but the outputs in such a case cannot be predicted and will depend on the weights learned. The performance of the system can be furthered by including new attack data instances during training of neural network and training it with the probability of an attack equal to the paranoid threshold while the classification of an attack can be trained as 'new attack' (mapped to numeric value 0.2).

RESULTS AND CONCLUSION

8.1 RESULTS

The report discusses our system design in detail and to substantiate the claims the prototype is developed to prove the efficacy of the design. The system prototype developed as part of this work focuses on detection of few attacks not or poorly detected by [17]. The adaptation required to bring other attacks under the system is merely in the datamining module where depending on the attack signature; algorithms to pull out corresponding features are worked out.

The system was trained on second week of training data (attack week), where as the architectural learning stage was carried out on the first day of third week tcpdump data (non-attack). The vigilance parameter set for clustering was 2 where as the amplification factor = 40 were set for according more importance to features viz. payload, same_address, the difference of whose distances mattered in the Euclidean metric. The number of clusters fixed during training was 33, of which four clusters (cluster No. 18, 30, 32 and 33) were declared abnormal i.e. indicative of attacks as listed in table 8.1, where as the rest of the clusters were considered normal activity. As seen cluster No. 18 and 30 reflected attack attributable to Land attack (attack identifier = 1), cluster No. 32 was computed having high probability of being a portsweep attack (attack identifier = 0.8).

Cluster No.	Probability of attack	Attack identifier	Remarks
18	1.0	1.0	Land attack
30	1.0	1.0	Land attack
32	1.0	0.8	Portsweep attack
33	0.5	0.2	New attack

Table 8.1. List of anomalous clusters learned by system

Cluster No. 33 is the dummy cluster which holds the weights of any machine activity that does not confirm to the normal behavior as learned during training. The cluster aids in detecting '*new attacks*' , these are those attacks whose patterns were not available in the training data, Table 8.2 lists few of these attacks.

Known attacks	New attacks
ntinfoscan	dosnuke
Crashiis	Sshprocesstable
Httptunnel	Smurf
Ps	Guessftp
Eject	Xsnoop
Secret	Sqlattack
Mailbomb	Guest
Ftpwrite	Arppoison
Portsweep	Sendmail
Perl	
Neptune	
Phf	
Satan	
Land	

Table 8.2. List of Known and new attacks

Type of attack	Correct prediction of normal trend	False Negatives	Correct prediction of attack trend	False Positives
Portsweep	100	0	100	0
Crashiis	100	0	100	0
Back	100	0	100	0
Land	100	0	100	0
Mailbomb	92	8	88	12
New attacks				
Smurf	100	0	100	0
Union of all attacks	76	24	66	44

Table 8.3 Results obtained for detection of attacks in test data

The results obtained for the attacks as shown in table 8.3 are against three server machines Pascal.eyrie.af.mil, Hume.eyrie.af.mil and Marx.eyrie.af.mil (refer Appendix 'B') monitored by the system for traces of attacks and these are near perfect but so may not be the case with all other attacks. The results are direct consequence of the data mining skills employed at data mining module. As brought out in earlier a large amount of attacks can be detected by fairly good results however, with each feature appended the neural network training will get more demanding.

8.2 CONCLUSION

With the advent of technology in all spheres, information and its dissipation have become a core activity. The networks are augmenting the purpose of optimizing the work environment by information and resource sharing. As its progress gallops towards higher bounds the dark horses (network miscreants) are not far behind in finding novel ways in threatening its very freedom and spirit. The network security administrator tries to prevent hackers from misusing the network resources by using security tools like firewalls and other mechanisms. These tools can be used to prevent attacks if the system administrator knows the attacker identity; an IDS aids a system administrator in identifying both known and unknown attacks. Though several methods for detection exists as discussed in chapter 1 of this report however, none in itself can be used as a silver bullet. Instead an ideal security tool for a system should consists of two- three tiers of defense with the firewalls forming the first layer and an IDS the second and third which employs the strength of both misuse and anomaly based detection system.

The system designed as part of dissertation work is based on this very principle but attempts to blend the potency of both the techniques in one. It detects attacks based on deviation from the normal while the clustering mechanism is used for learning the normal/abnormal behavior. The strength of the system lies in its ability of also using signatures of attacks for clustering the machine behavior besides the statistical feature computation. The signatures of attacks (discussed in chapter 2) are extracted from the network data over a time window by employing datamining skills. The prototype developed uses keyword selection to enhance the detection capability with encouraging results. Back and crashiis attack are detected based on misuse detection technique. The computation of the source machines behavior over a time window gives an added advantage of analyzing anomalous activity based on machine behavior during the window unlike packet level analysis in traditional anomaly based detection system.

The use of neural network based design reduces the computational and memory needs of system during real time detection stage, the bulk of the systems overheads in terms of processing and memory needs are limited to the training phase where for faster convergence of neural network Levenberg Marquart algorithm was used.

The use of pcap libraries give an inherent advantage to the system of adapting to real time detection of network traffic since the traffic are handled as network packets rather than as text files. The graphic interface developed for the prototype project is done using libglade and the core programming is done on GNU C on Linux platform [A1].

8.3 FUTURE RESEARCH

We have seen in this work how amalgamation of misuse and anomaly based detection system can be done exploiting the individual strength of each of these techniques. Use of neural network based system gave an added advantage of self learning the anomalous activity while reducing the computational and memory overheads during detection stage. The clustering mechanism learned the anomalous behavior while the datamining module unraveled the attack signatures within the packets over a time window. We remark that the improvements we suggest are meant to result in small improvements in performance but that we have not yet conducted implementation testing to verify this. Some areas for further research suggested here are:

1. By devising better mathematical algorithm in the datamining module several known attacks can be detected by this system. To detect newer attacks the system (neural network module) can be trained on the segment of test data consisting of new attacks with a forced probability of attack equal to the threshold set for paranoid status (0.3 in this work), so that any activity not in line with normal activity and neither an attack activity can then be clubbed by the clustering module as a new cluster (referred to as dummy cluster in this document) and the system administrator alerted. The training of the prototype developed was limited to the training data (DARPA 1999 dataset) which did not contain any instance of new attacks however the system design has accommodated for such a contingency.

2. The work in its present state is unable to detect stealthy attacks which take place over a larger period of time than the time window selected by the system since an attacker will mask his activity as a normal one in a fixed time frame. It is suggested that to detect such attacks the foot trace of a source machine in terms of

the clusters it belongs to over successive time windows can also be learned by the neural network to detect stealthy attacks. It is understood that an attacker launching a stealthy attack will follow a typical trait.

3. The usage of the system for real time detection of network traffic is overshadowed by the requirement of clustering mechanism training and neural network. This shortfall can be overcome by employing datamining skills to extract those portions of DARPA dataset that represent the network behavior similar to the network where it is to be used. The system has limited dependence of machine specific features like IP address etc. and detects attacks based on machine behavior; therefore those portions of dataset which possess similar statistical features as the live network can be used for training these components of our system.

References:

1. Esmaili, M., Safavi-Naini, R., Balachadran, B., & Pieprzyk, J. (1996). Case-based reasoning for intrusion detection. *12th Annual Computer Security Application Conference*, 214–223.

2. Debar, H., Dacier, M., & Wespi, A. (1989). Towards a taxonomy of intrusion-detection systems. Computer Networks, *Proceedings of the IEEE, International Joint Conference*, 205–210. 31, 805–822.

3. Debar, H., Becker, M., & Siboni, D. (1992). A neural network component for an intrusion detection system. *IEEE Computer Society Symposium Research in Security and Privacy*, 240–250.

4. Richards, K. (1999). Network based intrusion detection: a review of technologies. *Computer and Security*, 18, 671–682.

5. Lippmann, R. P., & Cunningham, R. K. (2000). Improving intrusion detection performance using keyword selection and neural networks. *Computer Networks*, 34, 597–603.

6. Weber, R. (1999). Information Systems Control and Audit. Upper Saddle River, NJ: Prentice Hall

7 C. Warrender, S Forest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models, 1999.

8. Leonid Portnoy. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACMCSS (DMSA-2001)*, Philadelphia, PA Nov 5-8.

9. E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *In proceedings of the International Conferences on Machine Learning*, 2000.

10. Denning, D. E. (1987). An intrusion detection model. *IEEE Trans. S.E.*, SE-13(2), 222–232.

11. Zhao Junzhong, Huang Houkuan(2002). An evolving intrusion detection system based on natural immune system. In*Proceedings of IEEE TENCON'02.*

12. Zheng Zhang, Jun Li et al (2001). HIDE: a Hierarchical network intrusion detection system using statistical preprocessing and neural network classification In *Proceedings of the 2001 IEEE Workshop on Informational Assurance and Security*

13. Walid A. Salmeh.(2004) Detection of intrusion using neural networks: A customized study. *Studies in Informatics and Control,* Vol 13, No. 2, June 2004.

14. Mathew V. Mahoney and Philip K. Chan. Learning Nonstationary models of normal network traffic for detecting novel attacks.*SIGKDD '02.* July 23-26 2002 ACM

15. James Canady and Jim Mahaffey. The application of artificial intelligence to misuse detection . In *Proceedings of First Recent Advances in Intrusion Detection (RAID) Conference,* 1998.

16. An Analysis of the 1999 DARPA/Lincoln Laboratories Evaluation Data for Network Anomaly Detection by Matthew V. Mahoney and Philip K. Chan, TR-CS-2003-02. (See also Proc. RAID, 2003, pp. 220-237).

17. Learning Rules for Anomaly Detection of Hostile Network Traffic, Proc. ICDM 2003 (© 2003, IEEE) and the longer Technical Report TR-CS-2003-18.

18. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks by Matthew V. Mahoney and Philip K. Chan, Proc. Eighth Intl. Conf. Knowledge Discovery and Data Mining, p376-385, 2002. (C) 2002, ACM (PDF, 10 pages)

20. http://www.ll.mit.edu/IST/ideval/docs/1999/detection_1999.html

21. Floyd, S. and V. Paxson, "Difficulties in Simulating the Internet." IEEE/ACM Transactions on Networking, 2001. http://www.aciri.org/vern/papers.html

22. Horizon, "Defeating Sniffers and Intrusion Detection Systems", Phrack 54(10), 1998,http://www.phrack.org

23. Bugtraq Archives (e-mail regarding Apache vulnerability). http://www.geek-girl.com/bugtraq/1998_3/0442.html. August 7, 1998.

24. Rootshell Website.http://www.rootshell.com/archive- 457nxiqi3gq59dv/199801/ beck.tar.gz.html. Jan 1, 1998.

25. CERT Advisory CA-96.21. http://www.cert.org/ftp/cert_advisories/CA-96.21.tcp_syn_flooding. September 19, 1996.

26. Simson Garfinkel and Gene Spafford. Practical Unix & Internet Security. O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol CA, 95472, 2nd edition, April 1996.
http://www.ll.mit.edu/IST/ideval/docs/1999/master-listfile-condensed.txt
http://www.ll.mit.edu/IST/ideval/docs/1999/master_identifications.list

27. http://www.ll.mit.edu/IST/ideval/docs/1999/detections_1999.html

28. Stevens. R. 1994, "TCP/IP Illustrated Volume I: The Protocols", Addison-Wesley Publishing Company, Reading. Massachusetts, vol. I pp 12.

28. Grossman, R.L. "Data Mining: Challenges and Opportunities for Data Mining During the Next Decade." May 1997. URL: http://www.lac.uic.edu/grossman-v3.htm (10 Oct 00).

29. Teuvo Kohonen, "The Self-Organizing Map",Proceedings of the IEEE,Vol 78,No 9,September 1990.

30. D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Representations by backpropogating errors", Nature, Vol 323 pp 533-536, 1986

31. Martin T. Hagan, Mohammad B. Menhaj, "Training feed forward networks with the marquardt algorithm", IEEE transaction on neural networks, vol 5 No. 6 November 1994.

32. http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html

33. Niggermann. O., Stein. B. and Tolle, J., 2001, "Visualization of traffic structures", IEEE International Conference on Communications, ICC 2001, vol. 5, pp 1516-1521.

34. Heatley, S. K., & Otto, J. R. (1998). Data mining computer audit logs to detect computer misuse. International Journal of Intelligent Systems in Accounting, Finance, and Management, 7, 125–134.

35. Hecht-Nielsen, R. (1988). Applications of counterpropagation networks. Neural Networks, 1, 131–139.

36. Helman, P., & Liepins, G. (1993). Statistical foundations of audit trail analysis for the detection of computer misuse. IEEE Transactions on Software Engineering, 19, 866–901.

37. Hill, T.,&Remus, W. (1994). Neural network models for intelligent support of managerial decision making. Decision Support Systems, 11, 449–459.

38. Hong, T., & Han, I. (2002). Knowledge-based data mining of news information on the Internet using cognitive maps and neural networks. Expert Systems with Applications, 23, 1–8.

39. Hruschka, H., & Natter, M. (1999). Comparing performance of feedforward neural nets and K-means for cluster-based market segmentation. European Journal of Operational Research, 114, 346–353. building intrusion detection models. IEEE Symposium on Security and Privacy.

40. Lunt, T. F. (1988). Automated audit trail analysis and intrusion detection: a survey. Proceedings of the 11th National Computer Security Conference, 65–73. Baltimore.

41. Maxion, R. A., & Townsend, T. N. (2002). Masquerade detection truncated command lines. International Conference on Dependable Systems and Networks, 219–228. Washington, DC.

42. Sasisekharan, R., & Shortland, R. J. (1996). Data mining and forecasting in large-scale telecommunication networks. IEEE Expert, 37–43.

43. Vaccaro, H. S., & Liepins, G. E. (1989). Detection of anomalous computer session activity. Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy, 280–289.

44. Verwoerd, T., & Hunt, R. (2002). Intrusion detection techniques and approaches. Computer Communication, 25, 1356–1365.

APPENDICES

Appendix 'A'

IMPLEMENTATION

A1. PROJECT DESCRIPTION

The prototype of the system discussed in this report is written in GNU C on Linux platform. The tools used for development are discussed in succeeding section. Some of the components of the project require to be trained before the system can be used. The incubation period of the project for training lasts for about 3 hours. The system was trained on complete second week (attack + non attack data) and first day of third week (non attack data) of inside.tcpdump files of DARPA 1999 dataset. These files were renamed as in[w] [d].tcpdump, where w stands for week and d stands for the day of the week i.e. inside.tcpdump file for first day of second week is renamed as in21.tcpdump. The project takes all the parameters as command line arguments and can be executed through console terminal using following command:

```
./nnconsole1 –c -1 –e 4 –r 1.5 –a 1000 –w 10 –j 2 –m 40 –f "dst 172.16.112.50" –F
in31.* in2*
```

Where options signify

-c [number of packets to be captured live or file specified in –F (-1 indicates infinite/EOF)]

-r [vigilance parameter for clustering mechanism]

-e [No. of extra ports to be monitored by the system besides those specified by system administrator]

-a [Architectural time in seconds over which the architectural structure is to decided]

-w [Width of time window in seconds]

-j [No. of hidden layers in the MLP NN]

-m [Amplification factor by which some of the features are weighted in clustering mechanism]

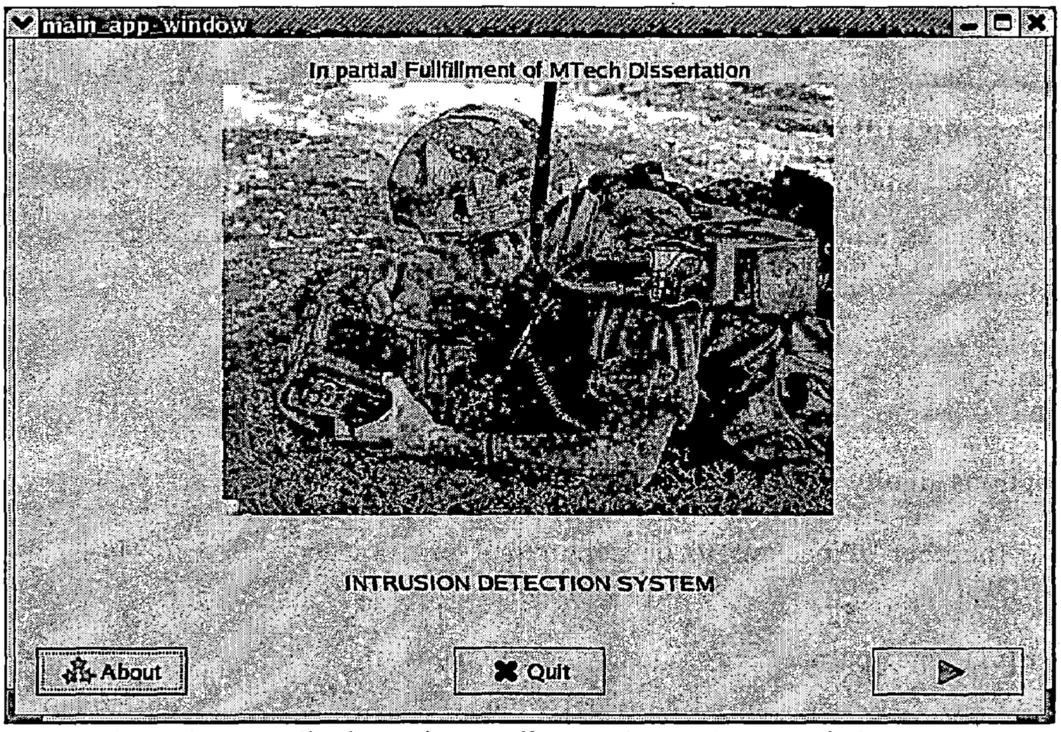
-f [Victim server machine]

-F [Name of files to be used for training components of system]

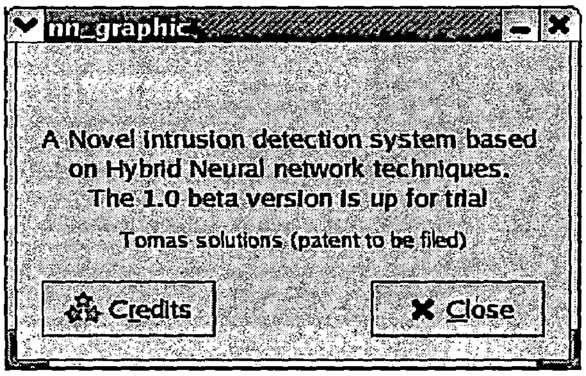
The graphic interface for the project is developed using libglade, which is inherently available with Linux operating system when loaded along with developmental tools. The front end graphic interface is activated by the following command:

```
./nngraphic  –F in31.* in2*
```

The main application window is as shown below:

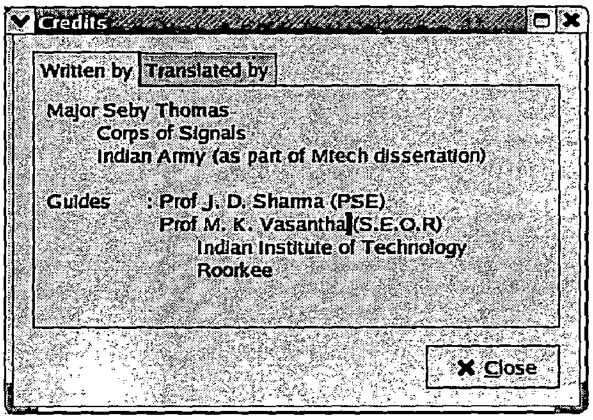

main_app_window
In partial Fullfillment of MTech Dissertation
INTRUSION DETECTION SYSTEM
About
Quit

The about button displays the credits to the authors, quit button returns control back to console terminal, where as the next button passes the control to the menu window displayed in the next page. The menu window offers to change the default values, specify the input parameters and execute the program.


nn_graphic

A Novel Intrusion detection system based on Hybrid Neural network techniques.
The 1.0 beta version is up for trial

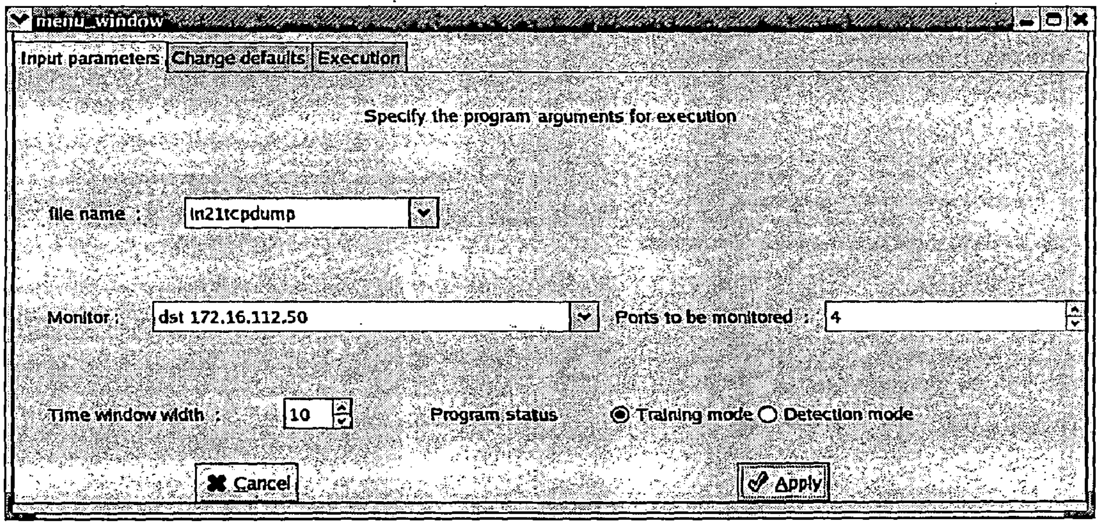
Tomas solutions (patent to be filed)


Credits

Written by | Translated by

Major Seby Thomas
Corps of Signals
Indian Army (as part of Mtech dissertation)

Guides : Prof J. D. Sharma (PSE)
Prof M. K. Vasantha (S.E.O.R)
Indian Institute of Technology
Roorkee


menu_window

Input parameters | Change defaults | Execution

Specify the program arguments for execution

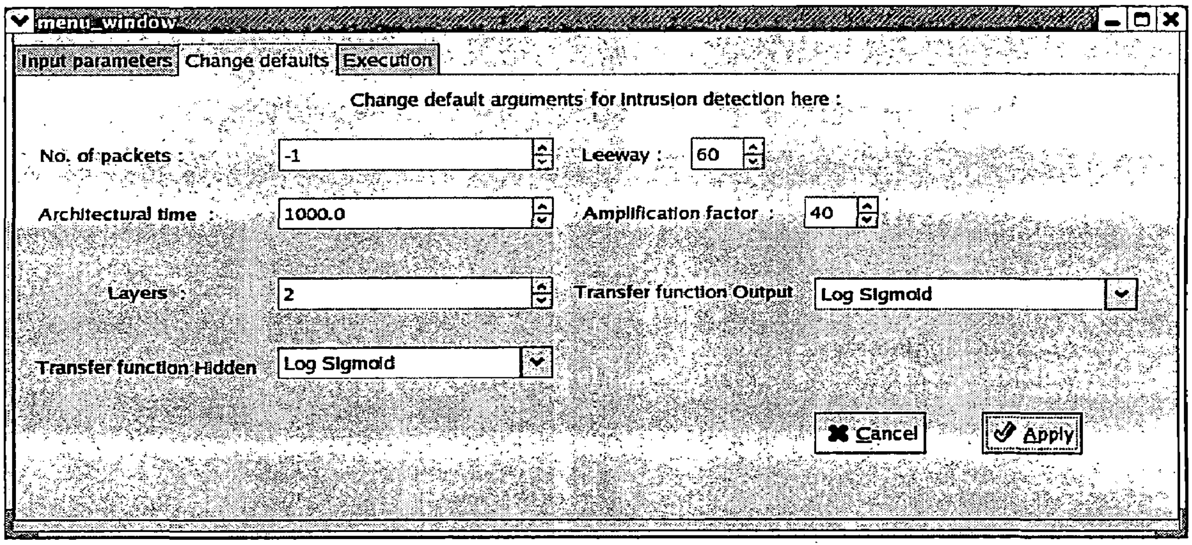
file name : ln21tcpdump

Monitor : dst 172.16.112.50

Ports to be monitored : 4

Time window width : 10

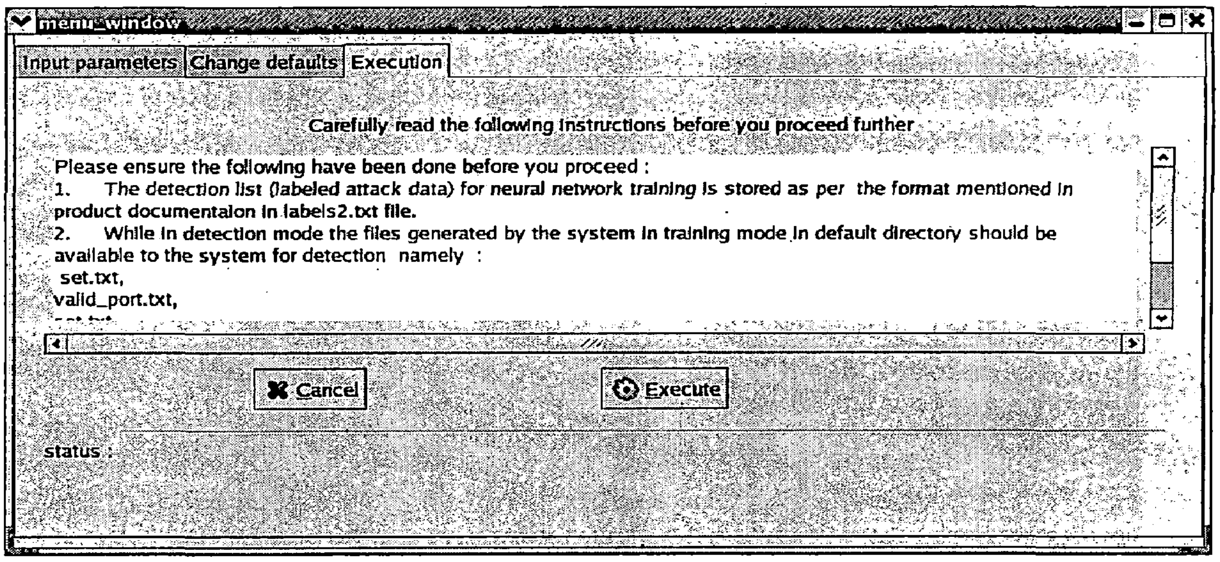
Program status [✓ Training mode] Detection mode



Change default arguments for intrusion detection here :

No. of packets : -1
Leeway : 60
Architectural time : 1000.0
Amplification factor : 40
Layers : 2
Transfer function Output : Log Sigmoid
Transfer function Hidden : Log Sigmoid

Once these parameters are entered using the graphical interface program the control is passed back to the core program which is console driven.



Carefully read the following instructions before you proceed further

Please ensure the following have been done before you proceed :
1. The detection list (labeled attack data) for neural network training is stored as per the format mentioned in product documentaion in labels2.txt file.
2. While in detection mode the files generated by the system in training mode in default directory should be available to the system for detection namely :
set.txt,
valid_port.txt,

A2. PROJECT DEVELOPMENT: TOOLS

The project was developed using system and programming tools inherent with Linux Fedora core version 10.0, Red Hat Distribution. The choice of the operating system is justified in this section.

A2.1 PCAP libraries

The Packet Capture library provides a high level interface to packet capture systems. All packets on the network, even those destined for other hosts, are accessible through this mechanism. The functions provided with the library help in basic operations like opening tcpdump file or applying filters and to capture packet from live traffic. The pcap.h file needs to be included in the source code and linked while compiling. The man pages on pcap provide elaborate details on usage of these functions.

A2.2 Libglade libraries

Glade is a user interface builder for GTK+ and GNOME, released under the GNU GPL License. The user interfaces designed in Glade are saved as XML, and by using the libglade library these can be loaded by applications dynamically as needed. By using libglade, Glade XML files can be used in numerous programming languages like C, C++ etc. GNOME is an acronym for GNU's Network Object Model Environment.GNOME's main objective is to provide a user friendly suite of applications and easy-to-use desktop. GTK+ stands for Gimp toolkit. It is a library for creating graphical user interfaces. GTK+ provides some unique features over standard widget libraries.

A3. ORGANISATION OF SOFTWARE

The logical flow of the software is in synch with the system flowchart (figure 4.2) and can be better assimilated in its light. The software consists of two main programs, the first handles the graphical interface while the second is a console driven intrusion detection system. The second program is called from the first program using execve function (execve function is defined in the unistd libraries and is used for executing programs; Linux Programmers Manual provides details on its usage). The second main program mainly consists of four main sub programs and several sub subprograms defined as functions within the source code; the key functions only are illustrated in figure A3.1 for paucity of space.

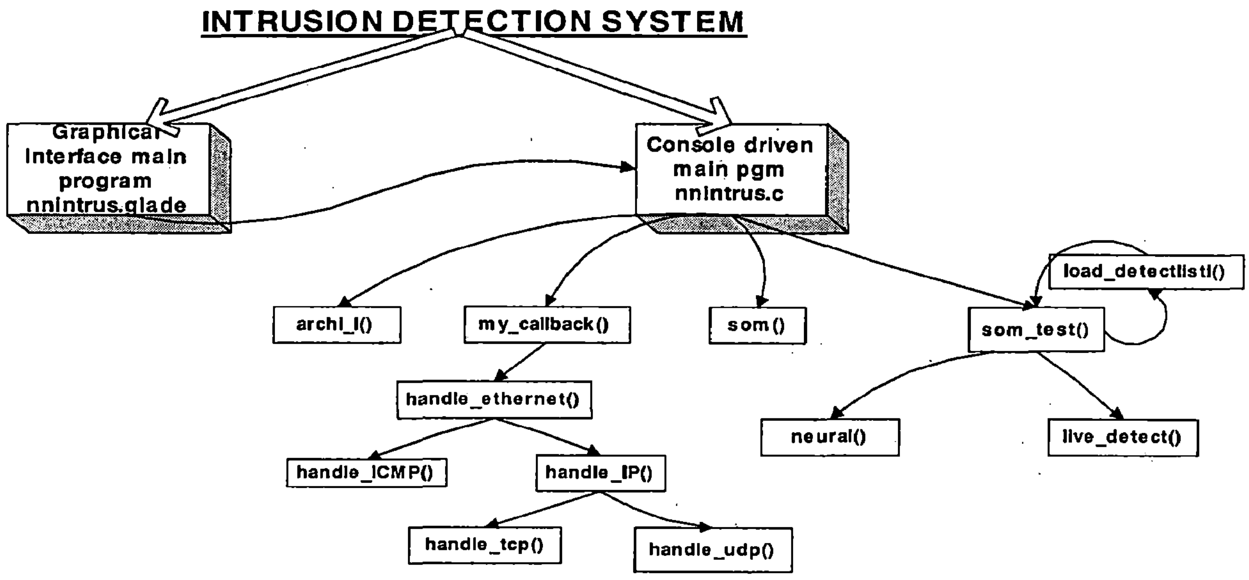


Figure A3.1 Organization of Software

The purpose of these functions or subprograms can be directly linked to the system flowchart. Their roles are described below:

(a)	my_callback_arch:	Architectural learning module
(b)	my_callback:	Datamining module
(c)	som	Clustering module under training
(d)	som_test	Clustering module post training stage
(e)	Neural	Neural network module under training using Levenberg – Marquardt algorithm.

(f) live_detect Neural network module under detection stage.

(g) handle_ethernet(),handle_IP,handle_TCP,handle_ICMP(),handle_UDP() : Handle_ethernet() handles the Ethernet frame and based on the protocol and application header respective sub subprograms handle the data.

(h) load_detectlist(): Subprogram loads the detection list from labeled DARPA dataset into the data structures for use by the neural network for supervised training.

The organization of the software is highly modular in nature and facilitates test points at each stage for debugging. The system consists of following programs written in GNU C (approximately 3500 lines of code):

(a)	nnconsole.c |	790 lines of code
(b)	nngraphic.c |	
(c)	initialize.c	447 lines of code
(d)	functions.c	980 lines of code
(e)	neural.c	504 lines of code
(f)	NN_levmar.c	611 lines of code
(g)	gui.c	126 lines of code
(h)	nnintrusion.glade	used libglade tools

Appendix 'B'

LIST OF OUTSIDE, INSIDE HOSTS ,ROUTERS AND HUBS IN SIMULATED 1999 DARPA DATASET

Outside Hosts

IP Address	Hostname	Operating System	Notes
135.13.216.191	alpha.apple.edu	Redhat 5.0	kernel 2.0.32
135.8.60.182	beta.banana.edu	Solaris 2.5.1	
194.27.251.21	gamma.grape.mil	SunOS 4.1.4	
194.7.248.153	delta.peach.mil	Redhat 5.0	kernel 2.0.32
195.115.218.108	epsilon.pear.com	Solaris 2.5.1	
195.73.151.50	lambda.orange.com	SunOS 4.1.4	
196.37.75.158	jupiter.cherry.org	Redhat 5.0	kernel 2.0.32
196.227.33.189	saturn.kiwi.org	Solaris 2.5.1	
197.182.91.233	mars.avocado.net	SunOS 4.1.4	
197.218.177.69	pluto.plum.net	Redhat 5.0	kernel 2.0.32
192.168.1.30	monitor.af.mil	MacOS	AF SNMP monitor
192.168.1.10	calvin.world.net		Outside gateway
192.168.1.20	aesop.world.net		Outside Web Server
192.168.1.90	solomon.world.net		Not Part of Simulation

Routers & Hubs

IP Address	Hostname	Operating System	Notes
172.16.112.1	loud.world.net		Cisco 2514 Router
192.168.1.1	loud.world.net		Cisco 2514 Router
172.16.112.5	None.		Hewlett-Packard EtherTwist Hub
192.168.1.2	None.		Hewlett-Packard EtherTwist Hub

Inside Hosts

IP Address	Hostname	Operating System	Notes
172.16.12.10	plato.eyrie.af.mil	Solaris 2.6	Not part of simulation
172.16.112.10	locke.eyrie.af.mil	Solaris 2.6	inside sniffer
172.16.112.20	hobbes.eyrie.af.mil	Redhat 5.0	Inside gateway, kernel 2.0.32
172.16.112.50	pascal.eyrie.af.mil	Solaris 2.5.1	
172.16.112.100	hume.eyrie.af.mil	Windows NT 4.0	Build 1381, Service Pack 1
172.16.112.149	eagle.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.112.194	falcon.eyrie.af.mil	Solaris 2.5.1	
172.16.112.207	robin.eyrie.af.mil	SunOS 4.1.4	
172.16.113.50	zeno.eyrie.af.mil	SunOS 4.1.4	
172.16.113.84	duck.eyrie.af.mil	SunOS 4.1.4	

172.16.113.105	swallow.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.113.204	goose.eyrie.af.mil	Solaris 2.5.1	
172.16.114.50	marx.eyrie.af.mil	Redhat 4.2	kernel 2.0.27
172.16.114.148	crow.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.114.168	finch.eyrie.af.mil	SunOS 4.1.4	
172.16.114.169	swan.eyrie.af.mil	Solaris 2.5.1	
172.16.114.207	pigeon.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.115.5	pc1.eyrie.af.mil	Windows 95	
172.16.115.87	pc2.eyrie.af.mil	Windows 95	
172.16.115.234	pc0.eyrie.af.mil	Window NT 4.0	Build 1381, Service Pack 1
172.16.116.44	pc5.eyrie.af.mil	Windows 3.1	
172.16.116.194	pc3.eyrie.af.mil	Windows 95	
172.16.116.201	pc4.eyrie.af.mil	Windows 95	
172.16.117.52	pc7.eyrie.af.mil	Windows 3.1	
172.16.117.103	pc9.eyrie.af.mil	MacOS	
172.16.117.111	pc8.eyrie.af.mil	MacOS	
172.16.117.132	pc6.eyrie.af.mil	Windows 3.1	
172.16.118.10	linux1.eyrie.af.mil	Redhat 5.2	kernel 2.0.36
172.16.118.20	linux2.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.30	linux3.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.40	linux4.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.50	linux5.eyrie.af.mil	Redhat 5.0	kernel 2.0.32

172.16.118.60	linux6.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.70	linux7.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.80	linux8.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.90	linux9.eyrie.af.mil	Redhat 5.0	kernel 2.0.32
172.16.118.100	linux10.eyrie.af.mil	Redhat 5.0	kernel 2.0.32

Appendix 'C'

Detections List : Training DARPA dataset

ID	Date	Start_Time	Destination	Score	Name
1	03/08/1999	08:01:01	hume.eyrie.af.mil	1	NTinfoscan
2	03/08/1999	08:50:15	zeno.eyrie.af.mil	1	pod
3	03/08/1999	09:39:16	marx.eyrie.af.mil	1	back
4	03/08/1999	12:09:18	pascal.eyrie.af.mil	1	httptunnel
5	03/08/1999	15:57:15	pascal.eyrie.af.mil	1	land
6	03/08/1999	17:27:13	marx.eyrie.af.mil	1	secret
7	03/08/1999	19:09:17	pascal.eyrie.af.mil	1	ps attack
8	03/09/1999	08:44:17	marx.eyrie.af.mil	1	portsweep
9	03/09/1999	09:43:51	pascal.eyrie.af.mil	1	eject
10	03/09/1999	10:06:43	marx.eyrie.af.mil	1	back
11	03/09/1999	10:54:19	zeno.eyrie.af.mil	1	loadmodule
12	03/09/1999	11:49:13	pascal.eyrie.af.mil	1	secret
13	03/09/1999	14:25:16	pascal.eyrie.af.mil	1	mailbomb
14	03/09/1999	13:05:10	172.016.112.001-114.254	1	ipsweep
15	03/09/1999	16:11:15	marx.eyrie.af.mil	1	phf
16	03/09/1999	18:06:17	pascal.eyrie.af.mil	1	httptunnel
17	03/10/1999	12:02:13	marx.eyrie.af.mil	1	satan
18	03/10/1999	13:44:18	pascal.eyrie.af.mil	1	mailbomb
19	03/10/1999	15:25:18	marx.eyrie.af.mil	1	perl (Failed)

20	03/10/1999	20:17:10	172.016.112.001-114.254	1	ipsweep
21	03/10/1999	23:23:00	pascal.eyrie.af.mil	1	eject (console)
22	03/10/1999	23:56:14	hume.eyrie.af.mil	1	crashiis
23	03/11/1999	08:04:17	hume.eyrie.af.mil	1	crashiis
24	03/11/1999	09:33:17	marx.eyrie.af.mil	1	satan
25	03/11/1999	10:50:11	marx.eyrie.af.mil	1	portsweep
26	03/11/1999	11:04:16	pigeon.eyrie.af.mil	1	neptune
27	03/11/1999	12:57:13	marx.eyrie.af.mil	1	secret
28	03/11/1999	14:25:17	marx.eyrie.af.mil	1	perl
29	03/11/1999	15:47:15	pascal.eyrie.af.mil	1	land
30	03/11/1999	16:36:10	172.016.112.001-254	1	ipsweep
31	03/11/1999	19:16:18	pascal.eyrie.af.mil	1	ftp-write
32	03/12/1999	08:07:17	marx.eyrie.af.mil	1	phf
33	03/12/1999	08:10:40	marx.eyrie.af.mil	1	perl (console)
34	03/12/1999	08:16:46	pascal.eyrie.af.mil	1	ps (console)
35	03/12/1999	09:18:15	duck.eyrie.af.mil	1	pod
36	03/12/1999	11:20:15	marx.eyrie.af.mil	1	neptune
37	03/12/1999	12:40:12	hume.eyrie.af.mil	1	crashiis
38	03/12/1999	13:12:17	zeno.eyrie.af.mil	1	loadmodule
39	03/12/1999	14:06:17	marx.eyrie.af.mil	1	perl (Failed)
40	03/12/1999	14:24:18	pascal.eyrie.af.mil	1	ps
41	03/12/1999	15:24:16	pascal.eyrie.af.mil	1	eject
42	03/12/1999	17:13:10	pascal.eyrie.af.mil	1	portsweep

43	03/12/1999	17:43:18	pascal.eyrie.af.mil	1	ftp-write

Appendix 'D'

Effect of increasing number of hidden layers in neural network training

(Figure 7.2)

Epochs	1 layer	2 layer	3 layers
0	0.002401	0.049361	0.680279
5	0.000165	0.000116	0.000231
10	0.000121	7.18E-07	0.000231
15	3.93E-05	5.49E-07	0.000231
20	3.91E-05	5.18E-07	0.000231
25	3.90E-05	4.66E-07	0.00023
30	3.90E-05	3.25E-07	0.00023
35	3.89E-05	2.82E-07	0.00023
40	3.89E-05	2.48E-07	0.00023
45	3.88E-05	1.59E-07	0.00023
50	3.87E-05	9.09E-08	0.00023
55	3.87E-05	5.86E-08	0.00023
60	3.87E-05	4.93E-08	0.000229
65	3.87E-05	3.71E-08	0.000229
70	3.87E-05	2.36E-08	0.000229
75	3.87E-05	1.66E-08	0.000229
80	3.87E-05	1.37E-08	0.000229
85	3.87E-05	6.43E-09	0.000229
90	3.87E-05	3.51E-09	0.000229
95	3.87E-05	2.34E-09	0.000228
100	3.87E-05	1.74E-09	0.000228
105	3.87E-05	5.44E-10	0.000228
110	3.87E-05	2.82E-10	0.000226
115	3.87E-05	1.87E-10	0.000222
120	3.87E-05	1.39E-10	0.000206
125	3.87E-05	6.81E-11	0.000205

Note : The effect of varying the number of hidden layers in MLFFBPNN is observed in the table above, Column 1 shows the number of epochs and 2,3,4 M.S.E rate when the number of layers varied as 1,2,3 respectively.

Appendix 'E'

Effect of varying number of nodes in each hidden layer

(Figure 7.1)

Epochs	6 nodes	4 nodes	8 nodes	3 nodes	10 nodes
0	0.049361	0.0102666	0.47036	0.00033	0.772442
5	0.000115821	0.00	0.00023	0.00014	0.0002008
10	7.18E-07	7.81E-05	0.00021	0.00011	0.0001817
15	5.49E-07	2.60E-05	0.00021	8.88E-05	1.82E-04
20	5.18E-07	1.97E-06	0.00021	2.64E-05	1.82E-04
25	4.66E-07	2.36E-07	0.00021	2.54E-05	1.82E-04
30	3.25E-07	1.84E-07	0.00021	2.52E-05	1.82E-04
35	2.82E-07	1.54E-07	0.00021	2.50E-05	1.82E-04
40	2.48E-07	9.80E-08	0.00018	2.50E-05	1.82E-04
45	1.59E-07	5.90E-08	0.00014	2.50E-05	1.82E-04
50	9.09E-08	3.84E-08	0.00014307	2.50E-05	1.82E-04
55	5.86E-08	1.48E-08	0.00014	2.50E-05	1.82E-04
60	4.93E-08	4.42E-09	0.00014	2.50E-05	1.82E-04
65	3.71E-08	2.76E-09	0.00014	2.50E-05	1.82E-04
70	2.36E-08	4.69E-10	0.00014	2.50E-05	1.82E-04
75	1.66E-08	2.70E-10	0.00014	2.50E-05	1.82E-04
80	1.37E-08	1.98E-10	0.00014	2.50E-05	1.82E-04
85	6.43E-09	7.47E-11	0.00014	2.50E-05	1.82E-04
90	3.51E-09	7.47E-11	0.0001	2.49E-05	1.82E-04
95	2.34E-09	7.47E-11	0.0001	2.49E-05	1.82E-04
100	1.74E-09	7.47E-11	0.0001	2.49E-05	1.82E-04
105	5.44E-10	7.47E-11	0.0001	2.49E-05	1.82E-04
110	2.82E-10	7.4664E-11	0.0001	2.49E-05	1.82E-04
115	1.87E-10	7.47E-11	0.0001	2.49E-05	1.82E-04
120	1.39E-10	7.47E-11	0.0001	2.49E-05	1.82E-04
125	6.81E-11	7.47E-11	0.0001	2.49E-05	1.82E-04

Note : Table shows the effect of varying the number of nodes in the two layers employed by the MLFFBPNN, column 1 indicates the number of epochs and column 2,3,4,5,6 their respective M.S.E rate.

Appendix 'F'

Comparison of various backpropogation techniques for neural network training

(Figure 7.4)

No. of Epochs	steepest descent with momentum	conjugate gradient	Quassi Newton	Levenberg Marquart
0	0.893889	0.680279	0.000337088	0.049361
5	0.878886	0.000230781	0.000229374	0.000115821
10	0.861514	0.000230742	0.000227556	7.18E-07
15	0.84092	0.000230657	0.00022753	5.49E-07
20	0.816229	0.000230533	0.000223532	5.18E-07
25	0.786577	0.000230364	0.000177734	4.66E-07
30	0.751248	0.000230144	0.000168121	3.25E-07
35	0.709929	0.000229952	0.000164455	2.82E-07
40	0.663089	0.000229836	0.00016444	2.48E-07
45	0.612289	0.000229761	0.000160058	1.59E-07
50	0.560131	0.000229629	0.000154173	9.09E-08
55	0.509577	0.00022954	0.000153896	5.86E-08
60	0.462915	0.000229423	0.000153861	4.93E-08
65	0.421049	0.000229304	0.000153791	3.71E-08
70	0.383604	0.000229178	0.000153775	2.36E-08
75	0.349554	0.000229042	0.000153649	1.66E-08
80	0.31783	0.000228892	0.000153365	1.37E-08
85	0.287657	0.000228727	0.000153365	6.43E-09
90	0.258631	0.00022854	0.000153365	3.51E-09
95	0.230686	0.000228295	0.000153365	2.34E-09
100	0.204006	0.000228026	0.000153365	1.74E-09
105	0.178918	0.000227741	0.000153365	5.44E-10
110	0.155788	0.000226366	0.000153365	2.82E-10
115	0.134926	0.000222266	0.000153365	1.87E-10
120	0.116522	0.000205547	0.000153365	1.39E-10
125	0.100614	0.000205309	0.000153365	6.81E-11

Note : Column 1 indicates the number of epochs and column 2,3,4,5 the corresponding M.S.E rate when various back propagation techniques were used by the neural network

Appendix 'G'

Effects of datamining skills on neural network convergence

(Figure 7.6)

Epochs	post datamining	pre datamining
0	0.049361	0.065554
5	0.000116	0.011383
10	7.18E-07	2.35E-03
15	5.49E-07	1.85E-03
20	5.18E-07	1.27E-03
25	4.66E-07	8.91E-04
30	3.25E-07	6.77E-04
35	2.82E-07	5.15E-04
40	2.48E-07	5.10E-04
45	1.59E-07	4.99E-04
50	9.09E-08	4.34E-04
55	5.86E-08	3.26E-04
60	4.93E-08	2.62E-04
65	3.71E-08	1.70E-04
70	2.36E-08	1.49E-04
75	1.66E-08	1.42E-04
80	1.37E-08	1.40E-04
85	6.43E-09	1.37E-04
90	3.51E-09	1.35E-04
95	2.34E-09	1.31E-04
100	1.74E-09	1.22E-04
105	5.44E-10	9.66E-05
110	2.82E-10	9.24E-05
115	1.87E-10	9.17E-05
120	1.39E-10	9.14E-05
125	6.81E-11	9.14E-05

Note : Column 1 indicates the number of epochs and column 2 reflects the corresponding error rate after datamining technique was applied where as column 3 indicates M.S.E rate without the datamining skills applied

Appendix 'H'

Study on effect of weighted features on clustering mechanism

(Figure 6.1)

Serial No.	Weighted features for clustering					No. of Clusters formed	% detection
	Port Activity	Invalid ports	TTL	Same address	key		
1	1	1	1	1	1	53	10
2	1	40	1	40	40	25	98
3	1	10	1	10	10	43	60
4	1	100	1	100	100	22	98.3
5	20	40	20	40	40	49	16

Note : Column 2 indicates all the ports being monitored have been given equal weights, where as column 3 indicates weight accorded to feature indicating number of inactive ports accessed by the source machine, TTL feature takes a value 1 if a packet within the time window of a source has an abnormal TTL field value, Key is a feature which takes a numeric value in the range [0-1] based on the presence of attack signature.

Appendix 'J'

VALID_PORT.TXT

(The file holds the valid or active ports on the victim machine against whom the intrusion is being monitored by preparing a list of active ports on the machine based on nonattack tcpdump file, a sample of some of those portsare listed below)

```
23    0.000000
22    0.000000
25    0.000000
123   0.000000
32787 0.000000
32810 0.000000
32838 0.000000
32848 0.000000
21    0.000000
32833 0.000000
32825 0.000000
32846 0.000000
32780 0.000000
32832 0.000000
32842 0.000000
32808 0.000000
32809 0.000000
32804 0.000000
32805 0.000000
32807 0.000000
32844 0.000000
32845 0.000000
32795 0.000000
32806 0.000000
32775 0.000000
32820 0.000000
32824 0.000000
0     0.000000
32803 0.000000
32794 0.000000
32776 0.000000
32823 0.000000
32828 0.000000
32779 0.000000
32792 0.000000
32827 0.000000
32829 0.000000
32843 0.000000
32777 0.000000
32851 0.000000
32790 0.000000
32778 0.000000
20    0.000000
32821 0.000000
32836 0.000000
32837 0.000000
32783 0.000000
```

Appendix 'K'

CLUSTER.TXT

(Note that the first five columns reflect the port activity where as the next four columns show the values of other features, column 11 indicate the number of times the cluster was labeled as an attack in the complete training period , column 12 indicates the number of time the cluster center was selected during the training period)

```
1     0.000000     -0.130121    -0.171973    -0.007347    1.355499
      0.000000     0.000000     0.000000     0.000000     0.000000
      2730.000000 0.000000

2     -0.000494    -0.294508    -0.186172    -0.069040    -0.426699
      0.000000     0.000000     0.000000     0.000000     8.000000
      2371.000000 16.000000

3     0.000000     -0.167857    1.038775     -0.014935    -0.611762
      0.000000     0.000000     0.000000     0.000000     0.000000
      712.000000   0.000000

4     0.000000     -0.316465    -0.206052    1.820978     -0.413980
      0.000000     0.000000     0.000000     0.000000     1.000000
      299.000000   20.000000

5     0.000000     1.202854     -0.171130    -0.039892    -0.503391
      0.000000     0.000000     0.000000     0.000000     1.000000
      792.000000   20.000000

6     0.000000     -0.357167    -0.243810    -0.111109    -0.461362
      0.000000     0.000000     0.000000     1.500000     3.000000
      895.000000   20.000000

7     0.000000     -0.492279    -0.256090    -0.174318    -0.369589
      0.000000     0.000000     0.000000     2.474874     1.000000
      328.000000   20.000000

8     0.000000     1.549690     -0.330869    -0.039806    -0.444999
      0.000000     0.000000     0.000000     1.788854     1.000000
      29.000000    20.000000

9     0.000000     -0.558844    2.277098     0.000000     -0.382929
      0.000000     0.000000     0.000000     2.032660     0.000000
      4.000000     0.000000

10    -0.000387    -0.395002    -0.231951    -0.127416    1.828020
      0.000000     0.000000     0.000000     -0.796108    4.000000
      1109.000000 16.000000

11    0.000000     -0.371564    1.984264     -0.120016    -0.422800
      0.000000     0.000000     0.000000     -0.796108    4.000000
      573.000000   20.000000
```

```
12      0.000000     1.719902     -0.224051     -0.115796     -0.414455
        0.000000     0.000000     0.000000      -0.796108     1.000000
        883.000000   20.000000

13      0.000000     3.175427     -0.303693     0.000000      -0.288675
        0.000000     0.000000     0.000000      0.684167      0.000000
        2.000000     0.000000

14      0.000000     -0.171106    -0.129329     -0.134884     -0.357966
        0.000000     0.000000     0.000000      -1.368335     1.000000
        53.000000    20.000000

15      0.000000     1.667806     -0.292541     1.953339      -0.413501
        0.000000     0.000000     0.000000      -0.550482     0.000000
        16.000000    0.000000

16      0.000000     -0.081650    -0.650339     -0.081650     -0.512590
        0.000000     0.700000     0.000000      1.500000      1.000000
        4.000000     16.000000

17      0.000000     -0.183067    0.193357      -0.100504     -0.398646
        0.000000     0.700000     0.000000      -0.303457     0.000000
        2.000000     0.000000

18      0.000000     -0.466072    -0.408248     0.000000      -0.408248
        0.000000     0.000000     1.000000      -0.461069     1.000000
        1.000000     20.000000

19      0.000000     -0.359000    1.950601      1.984485      -0.419940
        0.000000     0.000000     0.000000      -0.579239     0.000000
        6.000000     0.000000

20      0.000000     -0.444754    2.623821      -0.047246     -0.338415
        0.000000     0.000000     0.000000      0.408576      0.000000
        34.000000    0.000000

21      0.000000     3.474396     -0.375556     0.000000      -0.267261
        0.000000     0.000000     0.000000      -1.557695     0.000000
        1.000000     0.000000

22      0.000000     -0.247164    -0.218603     -0.148018     3.496258
        0.000000     0.000000     0.000000      -1.557695     0.000000
        5.000000     0.000000

23      -0.027217    1.645849     2.037323      -0.046462     -0.429477
        0.000000     0.000000     0.000000      -0.408248     2.000000
        14.000000    16.000000

24      0.000000     1.663361     -0.291403     2.091417      -0.402848
        0.000000     0.000000     0.000000      1.207778      0.000000
        3.000000     0.000000

25      0.000000     -0.542964    2.761858      2.242673      -0.335310
        0.000000     0.000000     0.000000      0.672842      0.000000
        2.000000     0.000000
```

```
26    0.000000     -0.452044    -0.190488    2.119653     -0.371446
      0.000000     0.000000     0.000000     1.788854     0.000000
      8.000000     0.000000

27    0.000000     -0.480465    0.000000     -0.223607    -0.223607
      0.000000     0.000000     0.000000     4.006649     0.000000
      1.000000     0.000000

28    0.000000     -0.491428    -0.239774    3.114036     -0.297775
      0.000000     0.000000     0.000000     0.049979     0.000000
      44.000000    0.000000

29    0.000000     0.000000     -0.101778    -0.454854    2.708391
      0.000000     0.000000     0.000000     -2.428707    0.000000
      3.000000     0.000000

30    0.000000     -0.397873    -0.333333    2.666667     -0.333333
      0.000000     0.000000     1.000000     -0.755929    1.000000
      1.000000     20.000000

31    0.000000     0.000000     3.170790     -0.288675    -0.288675
      0.000000     0.000000     0.000000     -2.106059    0.000000
      3.000000     0.000000

32    2.041241     -0.366296    -0.276158    2.041241     1.290994
      0.000000     0.700000     0.000000     2.040196     1.000000
      1.000000     16.000000

33    0.000000     0.000000     0.000000     0.000000     0.000000
      0.000000     0.000000     0.000000     0.000000     0.500000
      1.000000     0.000000
```

Appendix 'L'

LABELS2.TXT

(Note : The detection list is converted by the program into labels2.txt file which is then loaded in program data structure using load_detectlist function, Column 2, 3 gives out the date and time on which the victim machine givenin column 4 is attacked , the type of attack is identified in column 6)

```
00000003 03/08/1999 09:39:16 172.16.114.50      1 back
00000005 03/08/1999 15:57:15 172.16.112.50      1 land
00000008 03/09/1999 08:44:17 172.16.114.50      1 portsweep
00000010 03/09/1999 10:06:43 172.16.114.50      1 back
00000022 03/10/1999 23:56:14 172.16.112.100     1 crashiis
00000023 03/11/1999 08:04:17 172.16.112.100     1 crashiis
00000025 03/11/1999 10:50:11 172.16.114.50      1 portsweep
00000029 03/11/1999 15:47:15 172.16.112.50      1 land
00000037 03/12/1999 12:40:12 172.16.112.100     1 crashiis
00000042 03/12/1999 17:13:10 172.16.112.50      1 portsweep
```

Appendix 'M'

WEIGHTS.TXT

(The weights learned by the neural network are stored in this file, the first column identifies the layer whose weight matrix is given in the respective column for eg. 1 indicates input layer, 2 indicates first hidden layer, 11 indicates the biases for the activation function for the first layer and so on)

```
1       12.124458 1.336097  4.189026  6.794621  -10.694255 -14.406758 3.915025  -14.322686
1       4.094130  -2.701712 3.770890  1.586957  -5.579520 12.831892 3.727951  7.102176
1       5.986238  5.154681  -3.738653 -3.059633 0.793220  5.544315  2.526078  8.003633
1       7.266730  -0.707502 -3.184903 -1.816282 4.052572  -3.813072 11.603840 1.590209
1       5.346167  3.254396  5.188834  -0.289592 0.275908  1.443059  1.664576  0.662656
1       -0.402044 -0.813780 -0.888795 -1.648982 -1.191605 0.080371  0.660984  0.097620
1       -28.985300 -32.817535 -21.243147 -41.157098 -33.861924 -3.830380 25.419094 -13.387460
1       -17.197579 26.661211 -26.331027 2.002557 1.340583 9.265058  -10.933915 -16.537451
1       2.123473  0.736094  -0.835305 6.112051  7.614748  3.765511  -16.330341 -1.774928
2       -1.588189 0.880498  -6.735077 2.338000  -3.991837 -0.231353 1.615264  6.885277
2       -1.100372 2.710463  3.063509  -4.355818 1.348661  9.724065  0.266114  22.523344
2       -0.815352 2.760505  -0.043067 -0.399806 3.293720  0.899433  -2.046620 0.198607
2       3.494991  2.047174  -3.858157 -0.965705 1.780292  -0.649249 3.441759  0.361174
2       -0.157083 -4.922811 7.265902  -3.733304 9.365573  1.230118  0.500251  -8.766551
2       -3.378566 1.299354  5.746967  5.581333  3.194082  -3.507980 -3.540452 1.309580
2       2.884412  4.150403  8.315664  -0.023191 -3.722160 2.400542  -1.561797 4.834430
2       -2.459903 3.753446  -11.513946 -1.307424 4.232545 -9.212719 0.151397  -1.627768
3       2.517360  -5.947798
3       -3.875303 -3.074685
3       21.020371 8.742780
3       3.898470  0.653328
3       -5.921271 -4.453188
3       10.322303 24.829320
3       0.920767  -6.245233
3       -12.252769 2.196610
11      2.000225
11      -1.332123
11      11.828607
11      -0.251997
11      16.501632
11      -6.677321
11      -8.822067
11      13.463149
22      8.598995
22      -6.349320
22      -1.708481
22      0.630150
22      1.317005
22      -2.740570
22      7.829945
22      1.658313
33      -3.207665
33      -6.511979
```

Appendix 'N'

SOM_CLUSTER.TXT

(This file displays the classification of the clusters learned during clustering by the statistical computation process)

```
learning  on file :in21.tcpdump
** timestamp cluster 03/08/1999 15:56:16   timestamp label 03/08/1999
15:57:15
 Attacker :195.73.151.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :11   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.371564
 (22) : 1.984264
 (25) : -0.120016
 (123) : -0.422800
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/08/1999 15:56:16   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.114.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :6    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.357167
 (22) : -0.243810
 (25) : -0.111109
 (123) : -0.461362
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 1.500000
** timestamp cluster 03/08/1999 15:56:48   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.112.10       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :10   name of attack:land
       20.000000
 (80) : -0.000387
 (23) : -0.395002
 (22) : -0.231951
 (25) : -0.127416
 (123) : 1.828020
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
```

```
** timestamp cluster 03/08/1999 15:56:52   timestamp label 03/08/1999
15:57:15
 Attacker :195.73.151.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :11   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.371564
 (22) : 1.984264
 (25) : -0.120016
 (123) : -0.422800
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/08/1999 15:56:52   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.114.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :7    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.492279
 (22) : -0.256090
 (25) : -0.174318
 (123) : -0.369589
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 2.474874
** timestamp cluster 03/08/1999 15:57:07   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.112.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :18   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.466072
 (22) : -0.408248
 (25) : 0.000000
 (123) : -0.408248
 key : 0.000000
 TTL : 0.000000
 same addr : 1.000000
 invalid port : -0.461069
** timestamp cluster 03/08/1999 15:58:12   timestamp label 03/08/1999
15:57:15
 Attacker :172.16.113.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:land
       20.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
learning  on file :in22.tcpdump
learning  on file :in23.tcpdump
learning  on file :in24.tcpdump
** timestamp cluster 03/11/1999 15:46:26   timestamp label 03/11/1999
15:47:15
 Attacker :195.73.151.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :5    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : 1.202854
 (22) : -0.171130
 (25) : -0.039892
 (123) : -0.503391
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/11/1999 15:46:53   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.114.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :8    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : 1.549690
 (22) : -0.330869
 (25) : -0.039806
 (123) : -0.444999
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 1.788854
** timestamp cluster 03/11/1999 15:46:53   timestamp label 03/11/1999
15:47:15
 Attacker :197.218.177.69       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :11   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.371564
 (22) : 1.984264
 (25) : -0.120016
 (123) : -0.422800
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/11/1999 15:46:24   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.112.10        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :10   name of attack:land
       20.000000
 (80) : -0.000387
 (23) : -0.395002
 (22) : -0.231951
 (25) : -0.127416
 (123) : 1.828020
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/11/1999 15:46:41   timestamp label 03/11/1999
15:47:15
 Attacker :196.227.33.189      Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :4    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.316465
 (22) : -0.206052
 (25) : 1.820978
 (123) : -0.413980
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/11/1999 15:46:41   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.112.20       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:land
       20.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/11/1999 15:47:02   timestamp label 03/11/1999
15:47:15
 Attacker :195.73.151.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :14   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.171106
 (22) : -0.129329
 (25) : -0.134884
 (123) : -0.357966
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -1.368335
** timestamp cluster 03/11/1999 15:47:03   timestamp label 03/11/1999
15:47:15
```

```
 Attacker :197.218.177.69       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :11   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.371564
 (22) : 1.984264
 (25) : -0.120016
 (123) : -0.422800
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/11/1999 15:47:03   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.114.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :6    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.357167
 (22) : -0.243810
 (25) : -0.111109
 (123) : -0.461362
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 1.500000
** timestamp cluster 03/11/1999 15:47:03   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.113.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:land
       20.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/11/1999 15:47:07   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.112.50        Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :30   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.397873
 (22) : -0.333333
 (25) : 2.666667
 (123) : -0.333333
 key : 0.000000
 TTL : 0.000000
 same addr : 1.000000
 invalid port : -0.755929
** timestamp cluster 03/11/1999 15:47:28   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.112.10       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :10   name of attack:land
       20.000000
 (80) : -0.000387
 (23) : -0.395002
 (22) : -0.231951
 (25) : -0.127416
 (123) : 1.828020
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/11/1999 15:47:41   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.114.168      Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :12   name of attack:land
       20.000000
 (80) : 0.000000
 (23) : 1.719902
 (22) : -0.224051
 (25) : -0.115796
 (123) : -0.414455
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/11/1999 15:47:41   timestamp label 03/11/1999
15:47:15
 Attacker :172.16.112.20       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :6    name of attack:land
       20.000000
 (80) : 0.000000
 (23) : -0.357167
 (22) : -0.243810
 (25) : -0.111109
 (123) : -0.461362
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 1.500000
learning  on file :in25.tcpdump
** timestamp cluster 03/12/1999 17:12:12   timestamp label 03/12/1999
17:13:10
 Attacker :195.73.151.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :23   name of attack:portsweep
       16.000000
 (80) : -0.027217
 (23) : 1.645849
 (22) : 2.037323
 (25) : -0.046462
 (123) : -0.429477
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.408248
** timestamp cluster 03/12/1999 17:12:13   timestamp label 03/12/1999
17:13:10
 Attacker :172.16.113.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:portsweep
       16.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/12/1999 17:12:20   timestamp label 03/12/1999
17:13:10
 Attacker :172.16.114.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:portsweep
       16.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/12/1999 17:12:29   timestamp label 03/12/1999
17:13:10
 Attacker :195.73.151.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :23   name of attack:portsweep
       16.000000
 (80) : -0.027217
 (23) : 1.645849
 (22) : 2.037323
 (25) : -0.046462
 (123) : -0.429477
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.408248
** timestamp cluster 03/12/1999 17:12:42   timestamp label 03/12/1999
17:13:10
 Attacker :172.16.114.50       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:portsweep
       16.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/12/1999 17:13:02  timestamp label 03/12/1999
17:13:10
 Attacker :209.167.99.71       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :32   name of attack:portsweep
       16.000000
 (80) : 2.041241
 (23) : -0.366296
 (22) : -0.276158
 (25) : 2.041241
 (123) : 1.290994
 key : 0.000000
 TTL : 0.700000
 same addr : 0.000000
 invalid port : 2.040196
** timestamp cluster 03/12/1999 17:13:05  timestamp label 03/12/1999
17:13:10
 Attacker :172.16.112.20       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :2    name of attack:portsweep
       16.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/12/1999 17:13:13  timestamp label 03/12/1999
17:13:10
 Attacker :172.16.112.10       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :10   name of attack:portsweep
       16.000000
 (80) : -0.000387
 (23) : -0.395002
 (22) : -0.231951
 (25) : -0.127416
 (123) : 1.828020
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : -0.796108
** timestamp cluster 03/12/1999 17:13:21  timestamp label 03/12/1999
17:13:10
 Attacker :196.37.75.158       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
```

```
cluster No. :2    name of attack:portsweep
       16.000000
 (80) : -0.000494
 (23) : -0.294508
 (22) : -0.186172
 (25) : -0.069040
 (123) : -0.426699
 key : 0.000000
 TTL : 0.000000
 same addr : 0.000000
 invalid port : 0.000000
** timestamp cluster 03/12/1999 17:14:06   timestamp label 03/12/1999
17:13:10
 Attacker :209.167.99.71       Victim in label: 172.16.112.50
      victim  : 172.16.112.50
cluster No. :16   name of attack:portsweep
       16.000000
 (80) : 0.000000
 (23) : -0.081650
 (22) : -0.650339
 (25) : -0.081650
 (123) : -0.512590
 key : 0.000000
 TTL : 0.700000
 same addr : 0.000000
 invalid port : 1.500000



***********************************************************************
```