CHAPTER 1

LITERATURE SURVEY

1.1 Literature Survey about Computer Networks

Andrew S. Tanenbaum (2003) defines "Computer Network" as a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection can be wired which could be a copper wire or through fiber optics etc, The interconnection can also be done wireless such as microwaves, infrared, or communication satellites. Andrew S. Tanenbaum and Marteen Van Steen have also discussed the clear distinction between the distributed system and Computer Networks.

While dealing with the design and organization of the computer networks, managing the assets of computer network will be of major concern. The assets of computer network includes Software, Hardware, Data, People and Important Documents. Applications on the Computer Networks are built based on either ISO-OSI (ISO-Open Systems Interconnection, developed by Day and Zimmermann ,1983) or TCP/IP ( Transmission Control Protocol and Internet Protocol, proposed by , submitted by Cerf. V and R. Kahn as RFC793 and RFC791 in 1981) architectures. ISO-OSI or TCP / IP defines several layers of protocol stack which helps in development of inter-operable communication standards and applications. ISO-OSI specifies seven layers ( Application-layer, Presentation-layer, Session-layer,Transport-layer, Internet-layer, Data-link Layer and Physical Layer) whereas TCP/IP defines only five layers (Application-layer, Transport-layer, Network-layer, Network Access Layer and Physical Layer) in their protocol architecture. The purpose of each layer, their tasks and functionalities are explained by William Stallings. John Naughton has weaved his own personal experience about the way computer
networks and internet evolved in his book. He has also reported in detail about the creation of first computer network that was initiated by ARPA and the exponential growth of computer networks till 2000 in his book. During 1950 The United States Department of Defense (DoD) wanted to have a command-and-control network that could survive a possible nuclear war. The entire military communication system was using the public telephone network at that time, which was vulnerable as it would not survive the heavy bombings. During 1960 Advanced Research Projects Agency decided to build its own network and funded lot of agencies. In 1967 Larry Roberts with the help of Weseley Clark started to build a packet-switched network with each host having its own router. During 1969 First message was passed in the packet switched subnet.

Cameron Chapman in his article has written about how Computer Networks evolved and other developments associated with the computer networks. During 1970 there was ARPANET network and at the same time American Research Project Agency (ARPA) project encouraged universities and researchers to join their network and contribute towards creating standard protocols and applications. With the Inception of TCP/IP, many more types of networks joined the ARPANET and by 1973 trans-atlantic connection was done. Applications like email gained lot of popularity by that time. Once ARPANET computers switched over to TCP/IP in the year 1983 the entire architecture and way in which the computer network was viewed also changed. Slowly the internet was evolving in an exponential way. Internet got the world wide accepted definition as a network of networks and not just a network. Very soon thousands of domains were created, people started creating lot of applications based on the TCP/IP stack. Services over the network gained lot of popularity. During the decade 1970 and 1980 NSFNET replaced ARPANET and similarly other continents also created their own networks and finally the entire architecture of internet took a new structure. To have some control over
the standard way of communicating there were several agencies and organizations setup. During 1984 DNS (Domain Naming System) came into existence to manage the domain space. During 1989 and 1991 world wide web evolved which enabled the commercialization of the computer networks by offering services over internet.

Andrew S Tanenbaum (2003) tells us how the computer networks were used to develop applications which are developed based on every layer of the protocol architecture stack. With the commercialization of the computer networks and the introduction of world wide web(www) people started developing business applications which led to the use of client-server model for communication between the computer machines. Computer Networks were used to access remote information, communicate between person to person, entertainment and also for e-commerce and other related tasks.

1.2 Literature survey about the evolution of Internet and its security

With the intention of creating a backup communication network for US department of Defense, ARPANET was started. But by 1992 the whole architecture took a from with focus on non military purpose. DoD decided to not to fund further the domain registration and manage the address databases services. Finally DoD finally handed over to InterNiC. In 1998 IANA and InterNIC were reorganized under the control of ICANN. The DNS system operations were privatized and opened up to competition.

The merging of computers and communications had heavy influence on the way computer systems are organized. The combination of mobile computing and wireless networks increased the usages of the computer networks. Currently computers and computer networks, especially internet has become a basic need for most of the day to day computing.
“Disruption is a feature, Not a Bug” says John Naughton (2010) in his latest article which has given a very detailed view about all that any one should know about the computer networks. Disruption can be in any form which could include creation of innovative applications over the network protocol architectures. Some of the applications would help in making use of the computer network for a good cause like communication between person to person over e-mail or file transfer applications that helps business organization or governments with e-commerce. Also innovative applications can cause problems for computer networks or the resources on the computer networks which include software, hardware, data, people and documentations.

Most of the applications today are built based on TCP/IP architecture. The Biggest challenge is to secure the computer networks. Securing the Computer Networks mean securing the entire resources on the Computer Networks and not just one or few of its components. Andrew S Tanenbaum explains what are the tasks involved in securing the computer networks at various layers of the TCP/IP.

Even before looking at different ways of securing a computer network, it is important to understand the main cause behind the problems that have arose on computer networks. As John Naughton (2010) reports the way Vint Cerf and Robert Kahn created the computer networks, it was the design decisions made by Cerf and Kahn to address the need of a system which is future-proof that allow linking of lot of networks. Vint Cerf and Kahn decided to adopt two simple principles – first one is to design the system which does not have any central control where no one organization would decide who will join the network and secondly, design the system so that no one should decide what the network could be used for. So the system was designed and made so simple that it would just take data at one end and do the best thing to deliver at that other end being neutral about what data is sent across the network.
The works of Cerf and Kahn led to many innovations which includes the Tim Berners-Lee who put the code on an internet server to demonstrate web during 1991. During 1999 napster was introduced by a student and attracted more than 50 million users. At the same time there were other programmers who were looking at the TCP/IP and the flaws in the designs which were rather introduced as features started exploiting the loop holes and created disruptive applications like virus, worms, trojans, malware, botnets and other security exploits. Mark A. Ludwig explains various forms of virus. He also explains the choice of operating system also play a vital role while managing the computer and the Computer Networks. Operating system design flaws also counts during the implementation of computer network security measures.

Today computer network security cannot be addressed at one step or even cannot be defined in one single way. As Computer Networks are kept exposed and have to be exposed to all the users, there is always a possibility of threat coming from unknown and as well as known locations of a computer network Thus it is essential to take maximum care to secure the computer networks.

Potential problems in the design of TCP/IP architecture have given many opportunities for the attackers or problem creators to affect computers networks at all the levels of the protocol stack. Since there are now thousands of applications running on various ports as listed by IANA it is clear that addressing the security problems is the most challenging task.

TCP/IP has became victim of its own popularity. There were several flaws and vulnerabilities reported related to the design of TCP/IP which have given chances for the exploiters to cause problems on computer networks. TCP/IP wasn't built with security in mind, problems are caused due the vulnerabilities in applications with a weak secure endpoint communication says
Kelly Jackson Higgins in his article about blaming TCP/IP, DNS cache-poisoning flaw, man-in-the-middle Internet routing attack and denial-of-service attack using the TCP/IP. As mentioned by Andrew S Tannenbaum care has to be taken at every layer of the protocol stack while securing the computer network.

Andrew S Tannenbaum (2003) explains that security of computer network involves confidentiality, authentication, non-repudiation and integrity control. He also points our the resources of security threats. The security threats or attacks can come from a student who just want to snoop in to another persons email system, Cracker who would be trying to steal another person data, A Spy or a terrorist who would be willing to learn enemies military secrets or send threatening messages. Even from an internal person to the network who might just want to create unwanted problems for the computer network threats can be launched. Also problems can be created by a previous employee who knew details about the internals of an organization. With all these types of scenarios it is essential to take measures at every layer of the protocol stack.

Andrew S Tanenbaum suggested that in the physical layer, wiretapping can be avoided by covering communication lines in sealed tubes with gas at high pressure in case of wired networks. If some one tries to drill into the communication link, tube will release gas that reduces the pressure and raises an alarm. Many Defense systems use this technique.

At the data link layer, packets on a point-to-point line encryption could be implemented as they leave a machine and decryption can be done as they enter another machine. Every implementation can be done at a particular layer without worrying about the top layer. This solution is vulnerable to attacks from within the router and may not allow some sessions to be protected (e.g.,
those involving on-line purchases by credit card) and others not. But Link encryption still remains to be effective.

In the network layer, firewalls can be installed to apply filter so that good packets and bad packets could be distinguished at the router level. IP security also functions in this layer.

At transport layer, end to end, that is, process to process encryption can be done. For maximum security, end-to-end security is required. Finally, issues such as user authentication and non-repudiation have to be handled at the application layer.

Wllian Stallings (2004) has dealt the network security in more detail and explain the implementations of security services which include Authentication, Access Control, Data Confidentiality, Data Integrity and Non Repudiation. Stallings points at X.800 specifications for implementing security services.

Chris Chambers, Justin Dolske and Jayaraman Iyer have explored the flaws in TCP/IP, They pointed at the implementations of SYN flooding, IP Spoofing, Connection Hijacking, etc. They have also mentioned that the non focus on security has lead directly to the exploitation of TCP/IP's weaknesses by writing destructive applications. Also they suggested various ways of fixing some of the flaws (designing add-ons like TCP Wrappers, Kerberos, and SKIP), but the fixes may not be used by every one. Success always depends the willingness to implement the mechanisms. They have hinted that migrating to Ipv6 protocol suit, would be fixing the flaws in TCP/IP.

1.3 Literature survey about computer attacks / intrusions / virus /worms

Thomas Chen, Jean-Marc Robert,Fred Cohen, George Smith and many others have reported with complete details about how and why computer virus
were written and their impact on the computer networks. An article on wikipedia lists the entire time-line of computer virus. During 1966 John Von Neumann work was published titled “Theory of self-reproducing automata”. Based on his lectures given during 1949 on “Theory and Organization of Complicated Automata” enough evidence came were there by which it was possible to write self-replicating program that could affect the computer and their networks.

During 1970 Bob Thomas wrote Creeper, A fork bomber called wabbit was found during 1974. During 1975 John Walker wrote ANIMAL Which was the first Trojan. Richard Skenta wrote first large scale computer malware “Elk Cloner” for Apple II. Fredrick Cohen coined the term “Computer Virus” in 1983 after being able to write a program which was able to replicate itself and was also able to affect other programs on different computers running VAX11 and over the computer network. In the same year Ken Thompson wrote a first backdoor for Unix running computer.

In 1986 Basid Farooq Alvi and Amjad Farooq Alvi wrote first boot sector virus for the IBM PC. In 1987 Viena, Jerusalem, SCA, Emigas and other virus were reported. Morris and Ghostball virus were reported during 1989. During 1990's Happy99, melisa and variation of earlier worms, trojans and virus were reported with much more devastating effect over the computer networks.

During 2001 Barok created the “I LOVE YOU” worm. Later email virus Annan Kournikova was reported followed by sircam, codered and nimda were developed.

From 2002 onwards attackers started to write virus for the most vulnerable applications that runs on unsecured operating systems like windows...
which had lot of flaws in it and allowed to access hardware through its lower level functionalities. Beast was one such backdoor Trojan written only for windows running machines and similarly many other virus programs were written like Blaster, Sobber, witty, Sasser worms.

In 2005 Sammy, Bitook were reported and later Nixem and Startion were found. From 2005 onwards thousands of intrusive programs, virus, trojans, worms and other malicious code started to take different forms as many of the counter measures were able to identify the known attacks.

Counter measures included implementing decoying mechanisms and strong authentication procedures certain problems were avoided. After 2005 computer networks have grown with a complex structure and heterogeneous products which communicate with one another in a standardized way. The complexity involved in the process of one computer talking to another was so simplified that the user is almost not bothered about how the connection establishment and data transfer is taking place. Many Operating Systems included lot of security modules that takes care of majority of the tasks related to security. Many network interface cards were used to take care about the connection between one another and the network software usually that would be present on the chip of the card or as a module within the operating systems. Many times various versions of Unix and Unix like operating system like GNU / Linux included the network software that is TCP/IP stack within the operating system.

Today the whole architecture of internet has changed. Networks security could be classified into several categories like prevention of an attack, detection of an attack, protection / response for an attack. Computer Network term itself has become a confusing as the term distributed system was also
existing with few similarities. Andrew S Tannenbaum has clearly distinguished these terms.

For any computer network architect, design, organization and maintenance of computer network is a very challenging task in the current world. Initially during the creation of ARPANET the messages between host computers were communicated using packet switching technique. During early stages very few computers were connected and the communication lines were of very low speed. RFC508 describes the configuration of the ARPANET. ARPANET was the first store-and-forward packet-switching network. The subnet was a datagram subnet with minicomputers called IMPs (Interface Message Processors) connected by 56000 bits per second lines. Since Reliability was a high priority, each IMP host was connected to at least two other IMPs, so that messages could be automatically rerouted along alternative paths.

ARPANET networked four universities by December 1969. it grew to eight during July 1970 and then to nineteen universities by march 1971. By April 1972 more than thirty university joined the ARPANET which later grew to forty five universities by September 1972. With the adoption of TC/IP ARPANET in 1983 the size of the network grew exponentially. The number of users also grew more than 10,000. This gave lot of opportunity for the users on the network to try and experiment their programs or applications.

IANA lists well known port numbers which are used to run different types of services and applications based on TCP/IP or UDP. The number of applications that were developed was in such a huge number that it created lot of vulnerabilities and these vulnerabilities were exploited by the attackers to create problems for the computers and computer networks.
Users on one network in an Internet (network of networks) communicated with other users on different network by various means. Some of the users used emails, few used plain text, few used file sharing methodologies, few others used encoded documents, As and when the communication technologies also developed the methods of communication also changed which led to the creation of even more vulnerable situation of the network. Few malicious users started to create problem by writing malicious programs and include them in the documents which they use to communicate with the other users. And at the other end when the document was decoded, the malicious code was also getting executed. This kind of activities started to cause disaster as no one would know whether the document received was a legitimate or not. Computer Network which were designed initially for military purpose were slowly built and designed to address the non-military needs.

During the time of change in the communication technologies and computer hardwares, Software to manage the hardware also played a vital role in deciding the configurations and structures of the computer networks. With the changes in the processing capacities of the nodes on the networks more and more powerful devices were introduced and the driving softwares also were getting developed in parallel.

More than 100 Operating systems were developed during 1950 and 2000. Currently very few are being maintained and have stabilized. Among them Windows, MAC, BSD, RedHat, GNU/Linux are most popular. Other operating systems also exist for smaller, embedded devices. From the year 2000 very few operating system exist with regular updates.

Computers are now used for almost every day to day computing. Word Processors, Games, Music Players, Programming and Browsers are the most common softwares that are used on the Operating systems. Computer networks
are mainly used for e-commerce and information retrieval. Governments also are encouraging the usage of services on top of the Computer Network to reach the citizens for providing various facilities. Also the economics have changed a lot due to the entry of computer and usage of computer networks in every form of business. Since there are billions of users and thousands of different types of applications running on computer networks, IETF and other organization focused on creating variety of protocols for standardizing the processes of communication and computation. Standard Protocols for communication were developed, Standards for encoding documents evolved and also standard protocols for communication between users were developed.

So the growth of ARPANET in specific and Computer Networks in general, Development of Applications that run on the network, growth in the processing power of computers and developments in the area of communication which include data encoding standards and over all this having no one person or organization control over the computer networks or internet gave huge opportunity for every type of people to explore and exhibit their talents in developing their own applications of interested areas. Some of the people made the best use of the computer networks and developed advanced technologies covering variety of domains and some of the people took advantage of the uncontrolled environment, the various flaws that existed in the applications which were running on the computer networks, the flaws in the operating systems and tried creating problematic programs that affect either the user or the software or hardware on the computer network. All these developments led to the inclusion of Security Measures in every computer network. Today top priority is give for securing the resources of the computer network.

From various experiences, studies, research and experiments it is now clear that the combination of choosing the communication technologies,
computer hardware, software applications, encoding standards for the documents and over all the above the choice of operating system all together contribute to the efforts involved in securing the computer network. Flaws, Vulnerabilities and Weakness in any one of the above could lead to an opportunity for an attacker who will find all possibilities of attacking the computer network.

Gert De Laet has done a comprehensive work related to network security and also defined some of the computer networks terminologies with more clarity. He states that External and internal weaknesses have to be considering while dealing with the task of securing the computer networks. External weaknesses can be malware, spyware, cracker, executable scripts and other similar programs. Malware can be a virus, worm, trojan etc. Internal weaknesses could be from authorized users, unregistered programs and from non patched programs or non updated operating systems.

Fred Cohen from 1987 till 1994 has reported, described and demonstrated in his series of books the way to create malicious programs and the technical reasons about how it would be possible to write such kind of malicious content. He has covered a wide variety of different ways in which the computers and the networks could be affected. During 1994 George Smith had conducted many experiments in his Virus Creation labs and also published their signatures, behavior so that security experts would take appropriate steps in defending them. Many Operating System developers also used his information and patched their modules wherever necessary.

1.4 Literature survey about classification of attacks / intrusions

Kendall, K.R and others (1999 - 2000) while evaluating offline intrusion detection system for the DARPA data, have discussed in detail about the type of attacks and also the exploits that are possible to run on a computer or over a
network. N. Paulauskas and E.Garsva, in 2006 have made a very detailed classification of computer attacks and the technical reason about how it was possible to generate any of those attacks.

The current situation of network security is completely different. The computer users or a computer network administrators have to look for an alternative way of securing the networks as whatever the measures are taken intrusions are likely to happen either from inside or from outside due to the external and internal weaknesses and vulnerability which will exist always. Cyber Security Laws have evolved to bring some control over the computer networks or Internet. In order to legally bind the activities over the computer networks even the governments in many countries have setup laws. Many Companies are making business by providing services related to computer and the network security. Many volunteer community have initiated to write the malicious programs signature and programs that detect and prevent the attacks on computers and to collaboratively fight against the attackers. But at the other end many countries governments are encouraging their scientists and experts to steal other countries sensitive information like defense related documents which indirectly creating a competition to create problems for computer networks. In addition, governments in many countries are playing significant role in security by enforcing laws covering emerging technologies to build a legal system. So when any organization or an individual should abide by the legal requirements when making decisions to secure their resources on computer networks. Some of these legal requirements which are already in law are HIPAA (Health Insurance Portability and Accountability Act) GLB (Gramm – Leach - Bliley Act) and ECPA (The Electronic Communications Privacy Act) which mainly focus on privacy and security of some of the resources of the computer network. Other similar Acts exist in many countries.
Carl Endorf, during 2004 defines an Attack / Intrusion as an active sequence of related events that deliberately try to cause harm, such as rendering a system unusable, accessing unauthorized information, or manipulating such information. He refers to both successful and unsuccessful attempts while defining.

Attacks were grouped into five major categories by Lippman during the year 2000. Based on the know attack pattern he classified the attacks as Probe, Scan, Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), Data. Computer Emergency Response Team (CERT), based on the degree of success in obtaining access suggested that classification of attacks to be done based on account break-in, access attempts and as unauthorized use. DOS (Denial of Service) and Corruption of Data or Informational stealing were some of the examples.

N. Paulauskas, E. Garsv (1999) has suggested a detailed classification about the computer attacks. He also states that every attack will have one or all the fourteen features listed in his work. Attacks can be classified based on an attempt to break any of the security policy violation like executing malicious code, gaining access to user accounts or privileges, information integrity, confidentiality violations or denying a service.

Attacks can further be classified based on effects over user application like executable code detection, probe, scan, non-standardized protocol usage and malicious content executables.

Attacks can be classified based on the target of the attacker, like hardware access or administration privilege gains. Attacks can also be classified based on the type of service the attacker is targeting. There are more
than 10,000 different types of standard protocols that exist today, Attackers will be looking for standard services like Routing (RIP, OSPF, BGP), File transfers (FTP, SMB), Mail services (SMTP, POP3, IMAP), Web access (HTTP), remote access (SSH, TELNET, RDP) and encryption services (SSL) based on the target location which can be on a Local Network, Local Computer, Wired or a Wireless Network, Global Network or P2P network.

Attacks can be classified based on the type of operating system the attacker is targeting like Solaris, MacOS, BSD, GNU/Linux, Windows and others.

Also, they are classified the protocol layer the attacker is targeting any one of the TCI/IP or ISO-OSI protocol architecture like Application, Presentation, Session, Transport, Network, Data Link and Physical Layers.

Attacks can also be classified whether the attack is automated or manual and then further they can be classified based on whether the attack was based on the attack source which is a one to one or many to one or many to many and also Attacks could be classified based on the Connection quality which could be single or multiple. The terms Attacks or Intrusion with respect to a computer and a computer network could be interchangeably used as most of the researchers and security experts have used these terms to mean the same.

1.5 Literature survey about security mechanisms

By learning all the details about Attacks or Intrusions, with the knowledge about the reasons behind the problems that we face due to attacks, security policies have to be designed. It is the responsibility of each individual and as well as network administrator and the management to adopt security policies and also abide the government's security policies in order to have
secured computer, computer network and computing facilities at the individual, corporate, nation and global level.

The countermeasures or the security policies that were adopted in securing the computer networks main goal was to follow the standardized procedures like mentioned in X.800 standard which address authentication, authorization, data confidentiality, data integrity and non repudiation.

But Network security largely constitutes the mechanisms that addresses the Security of the computer network resources. Any Computer Network administrator would and should have implemented Preventive, Detective and Protective / Responsive mechanisms that will address all the standard security policies that would be adopted in an organization or at corporate level or even by the governments to maintain integrity.

US-CERT releases Technical Cyber Security Alerts, Weekly Summary of Vulnerabilities regularly on their websites and also Cyber Security Tips to help the computer users to keep counter measures updated against any new Attacks and take corrective steps to avoid spreading it. Also be aware about latest threats and attacks to a computer network. From the reports it is clear that majority of the threats are targeting Operating Systems or applications running on the Operating systems. Lot of works, experiments, research, standardization have taken place since many decades to solve the problem of attacks or intrusions. Majority of the works were mainly focusing to re-engineer and find a solution to the problem looking at the impact of an attack. Many companies now regularly keeps updating their subscribers with the latest update for any new attacks and will also suggest the users to take precautionary measures to fight against a possible attack / Intrusion.

Some of the most common precautionary measures involve preventive mechanisms. Preventive mechanisms ensures proper user authentication.
William Stallings explain the requirements and mechanisms for implementing the information security over a computer network. Encryption, Digital Signatures and Authentication Exchange were used to implement Peer entity authentication and Data origin authentication. Access Controlling mechanism are required to implement the access control service. Encryption and Routing Control is required to provide Confidentiality, along with this traffic padding will support Traffic Flow confidentiality. Digital Signatures play an important role in implementing the Data Integrity and Non-Repudiation. Availability of information is always possible with a proper Authentication Exchange and Data Integrity mechanisms.

Preventive mechanism were later concentrated not only to the information but also to the hardware and other resources on the computer network. To prevent the unauthorized users accessing resources on the computer network, proper authentication mechanisms have to be implemented and supported by the Operating system or the applications that are developed above the operating systems. The design flaws in applications will make the application more vulnerable for attacks and these vulnerabilities will be exploited by the crackers to write malicious code and affect the networks. Also to prevent even more sophisticated crackers, measures have to be taken to secure the applications that we use on a network or those installed on the operating systems. This decision will lead us to a wise selection of the operating system and the applications to be used in the computer network.

Preventive mechanisms would be useful only in the case of known attacks, but would not be sufficient enough to handle new type of attacks or intrusion and even the threats from the internal source cannot be identified. For situations like this and as well as during a situation where there are suspicious events needs detective mechanisms. Detective mechanism will help in customizing the security mechanism to detect new type of intrusions. But detective mechanism
will take lot of computing power. Response / Protection mechanism are those actions that would be taken after an attack is identified and the source and reason behind that attacks is found. Responsive / Protective mechanisms have to be very fast as these mechanism's decision will have impact over the future attacks that counts in rating the damages done to a computer network.

1.6 Literature survey about preventive security mechanisms

Firewalls were designed to implement preventive mechanisms like blocking unauthorized access of resources of a computer network through services. Firewalls will be built inside some of the operating systems. Firewalls could be implemented on a chip or even a set of devices can be designed as firewall. Firewalls usually deny data transmission over a network based upon the security policies adopted by an individual or an organization. Firewalls could be implemented both using a software or a hardware. Combination of software and hardware is also possible.

Firewall techniques include cover packet filtering, application gateways, circuit level gateways and proxy server for communication and as well as applications. In 1988 the first paper on firewall was published by Digital Equipment Corporation (DEC) soon after the detection of morris worm which affected even the NASA computer networks. The firewalls designed at that time were doing mainly packet filtering done by IP routers. Bill Cheswick and Steve Bellovin at AT & T Bell Labs designed another firewall. The first generation firewalls were termed as Packet filtering firewalls that mainly worked with the protocol and the port number combination. Depending on type of the protocol and the port numbers the security policies would be designed which will be fed as an input to the firewalls. Tis (1993), as part of DARPA contract, developed the Firewall Toolkit (FWTK).
In the second generation, focus was on application level protocol. Since application layer can filter more than what packet filtering does it gained more acceptance. There were two types of application-layer firewalls called host based and network based firewalls. Web application Firewalls and database firewalls also emerged during this time.

Third generation firewalls were “Stateful Firewalls”. From 1989-1990 three colleagues Dave Presetto, Janardan Sharma, and Kshitij Nigam, from AT&T Bell Laboratories introduced circuit level firewalls. Third-generation firewalls were maintaining records of all connections which was configured with firewall. They were also capable of deciding whether the packet is valid or invalid packet by keeping track of connection status. State-full Firewalls later proved to be inefficient during Denial-of-service attacks. Subsequent development focused on network layers largely and then application layer. Proxy concept gained lot of acceptance and network address translations (NAT) also played a very important role while designing the firewalls.

Several research works have been done related to firewall design and implementation. Christoph L. Schuba during 1997 proposed a firewall framework which could be used to design network level firewalls. Wolfgang Weber described several terminologies related to firewall and have also discussed the implementation details of firewalls relevant to telecommunication networks in 1999. In 2000 Ehab S. Al-Shaer has made several policy advises that had helped in detecting anomalies over computer network. Luciano Pascal Gaspary in 2004 for first time using the events and logs of Firewall had done behavioral study on computer networks.

In 2005 Quan Huang introduced an embedded firewall which was based on the network processor. Mohamed G. Gouda and Alex X. Liu implemented a state-full firewall in a private network in 2005. As the communication link
speed increased packet filtering and state-full filtering based firewalls were inefficient since many packets were overflown at the buffer due to the slow processing speed. Errin W. Fulp and Ryan J. Farley designed and demonstrated a parallel firewall for a high speed network and similar works were done by Errin W. Fulp too during 2006. J. Lane Thames and Randal Abler have done a study on the need of collaborative effort of many security mechanisms to fight against intrusions or attacks in a distributed environment during 2007. Shen Li and other during 2007 proposed their work on inclusion of dynamism in the Firewalls. The situation during 2007 was so that the impact of the attacks were so immense that the decisions were suppose to be done immediately and learn the behavior of the attacks to update the attack database. In 2008 Senda hammouda and others proposed an integrated model while implementing security measures. They have proposed to integrate the functionalities of NIDS and firewall to find the anomalies over TCP/IP channels.

As the number of intrusions and other malicious activities increased to a big number preventing all them was needed huge computing power. Ryma Abassi and Sihem Guemara El Fatmi gave an important suggestion that the Firewall policies validations have to be automated. Their work focused on automation of firewall policy generation and their validation.

In 2009 several experiments were conducted to integrate variety of applications and technologies as a single collaboration to fight against distributed threats. Bo Wang and other designed a honeynet-based firewalls scheme as an alternate way of implementing network security preventive mechanism. Hamed Salehi and others also proposed integrating signature based NIDS with packet-filtering firewall to increase the overall security of the computer network. In 2010 and current work mainly focuses on integrating various technologies and trying to find out the optimal combination which could be implemented in a distributed way.
From earlier research work related to firewall and other preventive measures related to network security have shown and convinced the experts and research community that preventive mechanisms alone are inefficient integration with detective mechanisms in order to address the threats / attacks / intrusions from the inside computer users of a network is required.

Since the number of types of attacker’s are increasing day by day and the technologies are growing in all areas like communication link, processing power, memory devices it is a real hard and challenging task to implement all security mechanism that are recommended by standard bodies. Many private organization have evolved since the past few years stating that they would provide complete security solutions. but the entire control of the computer and the processing power, memory usage, control over the devices attached to a computer or within a network will have to be handed over to the service providers which make the computer network even vulnerable, as a general user or a owner of the computer network will not have any idea about what the software or the hardware solution provided by the companies will do.

1.7 Literature survey about detective security mechanisms

D. E. Denning and P. G. Neumann designed the first Intrusion Detection Expert System in 1985 and deployed in the SRI International, Computer Science Lab. In 1987 they published their work and described a generic intrusion detection model which was capable of detecting computer abuse like penetration, break-ins. Their model depended mainly on the system audit records that helps in detecting the abnormalities. Statistical models were used to define the deviations and study the behaviors. Similar work and study on the Intrusion Detection Expert System (IDES) and a progress report was submitted in 1990 by Teresa F. Lunt. More neural network models were used to understand the behavior of the users, groups and other traffic flows over the
computer network. Teresa F. Lunt, R.Jagannathan, Rosanna Lee, Alan Whitehurst developed a knowledge-based Intrusion Detection System. Statistical models and rule based anomaly detection was implemented which they claimed to give a comprehensive security at that time. In 1991 L.Todd Heberlein and other also designed a security monitor which was similar to SRI-IDES but they experimented on the single broadcast LAN such as Ethernet.

Based on the profile of the usage of network abnormalities were identified and reported. They also focused on integrating Host based IDES with their monitor which gave significant improvements in detecting the deviations. Carl Endorf reports that L.Todd Heberlein had built Network System Monitor (NSM) during 1990-1991. NSM was different from IDES and DIDS in that it would analyze network traffic rather than system logs. This created more interest among the public towards the NIDS. Qiang Xue and others designed a distributed NIDS which was based on an intelligent agent.

During 1996 N. Paulauskas, and E. Garsva classification of attacks on computer systems gives a complete overview of different ways in which a computer or a computer network could be attacked. Their classification covers most of the attacks which are known attacks. Their work also involves the calculation of severity of each type of attack. In 1997 Kristopher Kendall in his thesis “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems” gave a real insight into the way the security mechanism have to be designed. His work entirely was focused on the Defense Advanced Research Project Agency 1998(DARPA98) data provided by Massachusetts Institute of Technology. Although the work of Kendall focused on off-line analysis even today his work is widely referred while working on classification of attacks on computer networks and especially with the study on Network Intrusion Detection systems. The work was tested with the standard data set DARPA98 for reference.
Sean Convery (1997) describes the Network Authentication, Authorization, and Accounting (AAA) Preventive mechanisms which could be used in preventing known attacks over a network or a single system. AAA mechanism existed even before Internet's Existence. Authentication process checks who is trying to access the system or what program is trying to get access to network or system. Authorization process will check about what users are allowed to do after getting into the system. Finally, Accounting will check for what the users have done and when was the event done. These fundamental security building blocks are being used in variety of ways for preventing unauthorized access to the system. Instead of simple user name and passwords, technologies like graphical passwords, Two-factor Authentication, Multi-factor Authentication, Biometrics, Single sign-On, Smart Cards, Digital Certificates and Public Key Authentication Infrastructure could be used along with other advanced authorization mechanism to prevent the unauthorized access to networks. Accountability will also help in keeping track each and every activity and helps in verifying any deviations other than what was allowed to be done and decide the consequences for those.

Detective mechanisms are the most costly mechanisms. Dorothy E. Denning in 1998 introduced a general purpose expert-intrusion detection system which is system-independent, application independent for real time implementation. Their proposed model is based on the hypothesis that exploitation of a system's vulnerabilities involves abnormal use of the system, which could be detected by the abnormal patterns of usage of system. Some of the ways to obtain the abnormal pattern of usage are by looking into the different types of logs like kernel log, application log, usage of CPU time, usage of memory while a program is running or over a time, statistics about the traffic in a network over a time, type of traffic, Audit Records and Anomaly records could be referred to analyze the abnormal pattern and use the statistical
models to calculate the deviation from the normal behavior and confirm the same.

Despite having (AAA) mechanism attacks were still happening over the network and also anomalous, malicious behavior and activities could not be prevented. Variety of techniques and methodologies evolved, During 1991 Harold S. Javitz and Alfonso Valdes introduced several statistical ways to identify anomalous users on a network.

Wenke Lee, Salvatore J. Stolfo (1998) have discussed about developing methods for intrusion detection using data mining techniques to discover consistent and useful patterns of system features that describe program and user behavior, by using the set of relevant system features to compute classifiers that can recognize anomalies and known intrusions. The association rule algorithm and the frequent episodes algorithm were used. These algorithms can be used to compute the intra and inter audit record patterns, which are essential in describing program or user behavior.

In 2003 there were few efforts in designing additional modules that support the NIDS like Decision Support systems designed by Yang Hongyu which was doing both detection and as well as responding to the intrusions. Their hierarchical NIDS works at network layer and application layer by monitoring the payloads at network data and applies statistical models like Naive Bayes analysis for classification of the attacks as anomalies. In 2003 Kaining Lu and other used the sequences of system calls information and designed a real time IDS, Their work focused on scenarios after an attack has happened and was an example of a collaborative model that focused on reducing the false alarm rates in IDS. Wu N and Zhang.J introduced factor analysis and Mahalanobis distance based Network Intrusion detection system in 2003 which reduced the false alarms significantly.
Carl Endorf, during 2004 defines an Attack / Intrusion as an active sequence of related events that deliberately try to cause harm, such as rendering a system unusable, accessing unauthorized information, or manipulating such information. He refers to both successful and unsuccessful attempts while defining the term attack.

Gert Schauwers in 2004 has introduced several categories of intrusion detection systems. He categorizes intrusion detection systems into signature based, anomaly-based, host based, network based, host based, protocol based, and traffic based. More classification can also be done based on the deployment level and the application level of the IDS.

Yu-Sung Wu et al in 2003 proposed a Collaborative intrusion detection system (CIDS) framework for aggregating the alarms from the different detectors to provide a combined alarm for an intrusion.

Min Qin and Kai Hwang in 2004 introduced a new Internet trace technique for generating frequent episode rules to characterize Internet traffic events. These episode rules are used to distinguish anomalous sequences of TCP, UDP, or ICMP connections from normal traffic episodes. Their anomaly detection scheme had success a rate of 47% for DoS, R2L, and port-scanning attacks. Which was an average of 51% improvement over the use of association rules. Their anomaly detection scheme can be used jointly with signature-based IDS to achieve even higher detection efficiency.

Steve Purser in 2004 wrote about his experience and has done lot of very practical recommendation to Information security managers, Security Consultants, Internal Auditors, Risk Managers, System Administrators, Business Managers and to other general users. In his experience he found that extracting information from the network is a biggest challenge so as to classify what is genuine and what is not. Classification of the traffic or data over
network requires lot of computing facility as the network may be huge and also
the communication link speed may also be very fast. With this complexity
scanning each and every packet flown over the network would just waste time
and energy and gives chance for a sophisticated attacker to attack the network
even before some body is thinking about securing the system. Steve Purser and
other introduced lot of techniques, tools, rules, standard practices and policies
that helped in securing the systems. It is always a big challenge to always
monitor the network for any threats. Many non-conventional ways of
addressing this problem were to be designed and the research community along
with the usage of the conventional approaches non-conventional techniques
were used to address the same.

Gert De Laet, Chengchen Hu and others in 2005 applied fuzzy
optimization models to improves the efficiency of a distributed NIDS. They
have found that the deployment of a network monitor in a computer network is
key task for the success of the NIDS. Fabrice Gadaud in 2005 proposed a new
NIDS architecture with a focus on clusters. Efficiency, Scalability and
Reliability were the main focus of their work. They concentrated on the process
level load balancing. Packet dispatch were done at network layer based on the
streams.

Joseph Migga Kizza in 2005 states that network security involves
elements like confidentiality, integrity and availability. Based on these
elements security can be categorized some time in to Physical and
psychological. Physical security is assured if deterrence, prevention, detection
and protection mechanisms are guaranteed in any network. Deterrence
mechanism involves regular announcements and imposing laws against
security threats or violations. Preventive mechanism involves authorization
techniques like biometric authentication, user accounts and other similar
techniques like firewalls as either hardware or software mechanism can be used
to block ports on a machine. Detective mechanism will help in detecting the possible threats over a network or a successful attack on a machine or over a network. Detective mechanisms can be applied for a known type of attack where the signature of the attack or the problem is already known and action against them can also be taken, typically detective mechanism will just inform the concerned person who has applied the detective mechanism in his / her network by raising an alarm. The alarms can be true or false. Reducing these alarms will be the main concern of the administrator of the network. Detective mechanisms should also have to be applied for the unknown type of attacks or threats. When none of the Deterrence, Preventive and Detection mechanisms work Response will be the only mechanism that have to be applied to avoid further damages that could be caused to the failure of the the three mechanisms. Joseph Migga Kizza called Psychological security as Pseudo-security. Pseudo-security is security through obscurity. This way of security believes in keeping the security mechanisms secret. Several times this kind of security depends entirely on how much the insiders are believed as safe. By implementing security mechanism like this can lead to a situation where the entire systems vulnerabilities and weakness could also go hidden which could be later exploited by some one and make the whole system weak.

Massachusetts Institute of Technology (MIT) also has Attack database which helps to work with intrusion detection. MIT has described seventeen types of Denial of Service Attacks, Twelve types of User to Root Attacks, Fifteen types of Remote to Local Attacks. Ten types of probes attacks and Secret Data attack types. MIT also has kept the simulated network data along with the attacks so as to help the research community to test for this standard data for their work. In the database that is described by MIT has the signature and the behavior of the above types of attacks in detail. But again any new type of attack may or may not fall into any of the above four in that case a very different approaches have to be explored.
A. Qayyum et al (2005) presented a guideline for Statistical based anomaly detection techniques. They gave a theoretical guideline about using variety of statistical models depending upon the action that has to be taken during an event of anomaly in a network. IDS will have sensor, analysis and response modules. A Qayyum et al have presented a very detailed study about variety of anomaly detection techniques and especially statistical techniques like operational model, markov model, statistical moments, multivariate models and time series models that could be used depending upon the situation that arises on a network during abnormalities or anomalous behavior.

Tich Phuoc Tran et al in 2006 found a technique with dynamic approach that could handle complex attacks on distributed computer systems where the complete process could be automated. They proposed an innovative Machine Learning algorithm called Boosted Modified Probabilistic Neural Network (BMPNN) which utilizes semi-parametric learning model and Adaptive boosting techniques to reduce learning bias and generalization variance in difficult classification.

Hao Wang and others (2006) designed a product called Netspy which will help in generating signatures for NIDS. The signatures were targeted at network level. Netspy was capable of generating signatures for the malicious activities which are used as NIDS signatures.

Shufen Liu et al (2006) designed a NIDS based on firewall on GNU/Linux, they used IPTABLES as the softwares. Simkhada, K et al (2006) proposed an efficient signature-based method for automatic detection of worms over large-scale networks. In their system, detection is performed in a hierarchical manner. Security managers in collaboration with high-hierarchy metropolitan managers and global managers will collectively work together to identify the worms with high detection rates and low false alarm rates.
Miyuki Hanaoka and others in 2007 worked towards improving the efficiency of the NIDS / NIPS which has layer seven awareness. Their design did not have the traditional dependency of having the full information about TCP/IP reassembly. They implemented their idea on GNU /Linux version.

Using Automation transition matrix in 2007 Ne-FeHuang and others designed a cons-effective string matching algorithm for NIDS. Similarly many algorithms were getting developed to improve the performance and efficiency of the NIDS. Cheng Zong in 2007 proposed a string matching algorithm using Chinese remainder theorem that help in processing network traffic data.

Mohsen Beheshti and others during 2008 worked on Snort and other free software with data mining and machine learning algorithms. The main focus was on collection of snort data and transformation of the same to their SeeS data mining system. Zing-king-hua and others (2008) introduced a self adaptive NIDS which were desperately needed by the security experts as monitoring the enormous alerts generated during a small change in the structure of the network was a very tedious task.

Abhishek Das (2008) and others developed a FPGA based architecture for NIDS. Once the speed of communication network crossed few Gigabits the existing NIDS engines were not able to do complete trace of the network traffic which reduces the efficiency of the NIDS. Instead of collecting the entire information , only the feature of the network traffic were extracted and were used at final stages while taking decisions. Ryan Proudfoot (2008) and others designed a flexible software and hardware based NIDS which was capable of handling upto 1.7 GB/sec communication link and was scalable.

Di He, and Henry Leung (2008) did a deep study on the way the alerts were generated by a NIDS and from that information itself they were able to compare the constant false alarm rate (CFAR) with any abnormalities and
report the same. They have used statistical modules for predicting the behavior of the network traffic. Khosravifar B and Bentahar J (2008) proposed an architecture based on honeypots to detect the attacks and also to reduce the false alarms using distributed agents. Using their honeypot based scheme.

WenJie Tian and Ji Cheng Liu in 2009 overcame the deficiencies of low accuracy and high false alarm rate in network intrusion detection system by designing an integrated Intrusion detection model based on support vector regression (SVR) and principal components analysis (PCA). Zhihua Zhou Shifu Chen (2009) proposed a neural network based classifier which used Adaptive Resonance Theory and Field Theory. Their experiments have shown that the learning speed was much higher than other neural network models and also the predictive accuracy was much larger than other techniques.

Lot of researchers focused on adopting neural network concepts to determine the pattern of the intrusions in a computer network. One such research was conducted by Jingwen Tian and Meijuan Gao in 2009 who adopted genetic algorithm neural network along combination of adaptive and floating-point code genetic algorithm with BP network. The results showed that their approach had higher accuracy and faster convergence speed with rapid intrusion detection capabilities.

Miyuki Hanaoka and others (2009) came up with a collaboration model that will have many NIDS/NIPS working together to detect intrusions. Redundant rules were reduced due to these kind of approaches. Sheng Sun and YuanZhen Wang (2009) used A Weighted Support Vector Clustering Algorithm in Network Intrusion Detection on KD99 Cup, Lin Chin (2009) came up with a Framework that was based on Game Theory to take decisions in an Intrusion Detection System for a Heterogeneous Networks. Lin Chen (2009) developed a Network Intrusion Detection System with rules combing
fuzzy set theory and genetic network programming to detect intrusion. Their work demonstrated that the logic could be applied to both misuse and anomaly detection. Desheng Fu and others (2009) implemented a distributed NIDS based on data mining principles. We Zang (2009) and others proposed a new NIDS based on hardware-based multiple regular expressions matching architecture, called MRM.

Their solution utilized RAM modules and register's signals to take decisions and their model worked fine for communication link upto 2.8Gbps. Few experts and researchers were concentrating on using the existing COTS solutions for implementing the NIDS and improve the rule generation mechanisms. Raghavan Muthuregunathan and others (2009) using parallel clustering techniques and evolutionary computing techniques like genetic algorithms and hill climbing which optimizes the clusters and later analyzing each clusters rules were generated. Jianchao Han and others worked on adopting COTS solutions for NIDS to detect and prevent malicious and misuse on a computer network.

In 2010 Salvatore Pontarelli and others proposed a new architecture for NIDS which was based on the FPGA. Since the hardware-based solutions for NIDS would face the problem of fixed rules and updating would be tough. FPGA based solution would give the freedom to program the Chips. Their solution also helps in providing dynamic responses to the intrusions.

Bin Zeng and others (2010) published a different way of implementing the NIDS, they used snooping agents that collect information from the main packed demultiplexer and then detect and infer about the intrusion.

Monis Akhlaq and others (2010) developed a faster NIDS using dynamic cluster and comparator logic. Each computer on a private computer network were used to create a cluster which helped in implementing security
policies. They designed a load balancer which was distributing the intrusion-detection tasks among the nodes of the cluster and comparator logic was used to get back the lost information about any abnormal or regular activity.

Michele Colajanni and others (2010) implemented an early threat detection algorithm for NIDS in a large distributed network. Combines peer to peer and hierarchical NIDS cooperation schemes with distributed ranking in order to focus on the intrusion classification.

Guangming Yang and others (2010) developed a NIDS based on the vulnerabilities over a network. Their proposed idea involved keeping track of the known vulnerabilities and focus on the relevant according to the network instead of the complete trace of the network data. They evaluated each host on the network and decide which are safe and which are not and then consider only those host that are vulnerable. In this way certain amount of computing was reduced to detect the intrusions.

1.8 Literature survey about honeypots

Lance Spitzner (2002) defines HoneyPots as a security resource whose value lies in being probed, attacked, or compromised. During 1990/1991 First public works documenting honeypot concepts were done by Clifford Stolls. In 1997 Version 0.1 of Fred Cohen's Deception Toolkit was released, which was one of the first honeypot solutions made available to the network security community. In 1998 development began on CyberCop Sting, one of the first commercial honeypots sold to the public. CyberCop Sting introduces the concept of multiple, virtual systems bound to a single honeypot. During 1998 Marty Roesch and GTE Internetworking begin development on a honeypot solution that eventually becomes NetFacade. This work marked the beginning of the concept of Snort. In the same year BackOffice Friendly was released as a free, simple-to-use Windows-based honeypot. In 1999 Formation of the
Honeynet Project and publication of the "Know Your Enemy" series of papers was done. This work helped increasing awareness and validate the value of honeypots and honeypot technologies. During 2000/2001 use of honeypots to capture and study worm activity Initiated . More organizations started adopting honeypots for both detecting attacks and for researching new threats. In the year 2002 a honeypot was used to detect and capture in the wild, a new and unknown attack, specifically the Solaris dtsped exploit. In 2007 Neil C. Rowe and Han C. Goh demonstrated their research work that included deceptive techniques which will make the attackers feel that the inconsistency information they are getting from the system would be of no use and will not try intrude the computer network and create problems. They used snort for their experiments.

S. K. Gupta et. al. (2007) designed a framework to induce a suspected user in the honeypot in order to trap the actual intruder. By collecting information from the induced user about the activities of the others who communicate with them. These kind of techniques were used in most of the honeypot based solutions later. Katsumi Ono et al (2007) reported the trends of botnet and have discussed the methodologies to detect botnets and capture malware. They have adopted the strategy of observing the fixed point over the internet. Mohamed Nassar designd a SIP protocol specific Honeypot. Their inference mechanisms were based on the received messages. Remi Badonnel and others (2007) designed a honeypot based NIDS specifically for the P2P networks to trap the cyber predators over P2P networks as there are always chances to pretend the file sharing process sand do anomalous activities.

Chung-Huang Yang, Chao-Hsi Yeh Designed and Implemented a Honeypot Systems Based on Open-Source Software in 2008 gave ample opportunities for many researchers and organization to study more and use more honeypot based solutions to secure their networks.
Jungsuk Song and Hiroki Takkura (2008) proposed two types of honeypot to collect unforeseen exploit codes automatically decoying malicious attackers and cooperation based active honeypot and self-protection type honeypot.

False alarms are one of the major constraints that NIDS has. The quality of the NIDS is evaluated based on the number of false alarms that it generates. Towards minimum false alarm generation, Babak Khosravifar and others have proposed a NIDS based on honey pot which raises less false alarm rates than the conventional models. Anjali Sardana and R.C. Joshi (2008) proposed the automatic generation of adequate server nodes to service client requests and honeypots to interact with attackers in contained manner. During DDOS attacks it would be impossible to recover from the effects of the attack, during such situations mixture of servers and honeypots in DMZ at different time intervals was used. This approach will keep the network stable even during attacks. They also have worked on mitigating DDOS attacks on servers at ISP levels using a honeypot based routing.

S. Almotairi and others (2008) used principal component analysis to characterize attacker's activities in the honeypot which reduced the efforts in separating out the latent groups of activities. Once the honeypots are detected then it would be tough for the security experts or the administrators of computer network to again use honeypots to detect the intrusions as the intruder will learn about the honeypot, stops all subsequent interactions. In order to take countermeasure steps against detection of honeypots it is essential that certain precautions have to be taken. One such precautionary measure was proposed by Lai-Ming Shiu and Shang-Juh Kao (2008).

The procedure or technique to collect information about Internet malwares on a computer network based on the client side honeypot was
proposed by Xiaoyan Sun and others in 2008. Extracting information from the activities over a network, inferring from the information collected and also designing rules based on the information collected were becoming key tasks in a NIDS as every day there were new threats coming and also the total number of already known attacks were huge in number for which computing power required to scan every attack or intrusion was merely impossible. S.Antonatos and others in 2008 proposed an idea for large-scale threat monitoring and they named it as Honey@home. Another such similar large-scale deployment was done by HoneyLab (2009) which was proposed by W.Y.Chin. They invited other organizations, researchers, security firms and others to deploy their own honeypot services and later collectively learn against any new type of intrusion.

Vini V Das (2009) proposed an honeypot scheme for a DDOS attacks. Their idea of opening a virtual communication channel for an authorized client of a server and plater provide an Active server and also having the server for honeypot gave better results in trapping and classifying the intrusions. Honeypots were gaining huge popularity by 2009. Honeypots were also implemented in sensor networks. Rajani Muraleedharan, and Lisa Ann Osadciw developed a framework which included honeypot concepts and swarm intelligence. Hogyun Lee developed a P2P honeypot to prevent illegal or harmful contents form spreading in P2P network. Lin Chen and others (2009) have done research work on dynamic forensics where they used honeypots as intrusion tolerance mechanism. Their work mainly focused on collecting more information about the occurrence of an intrusion while the honeypot mechanisms were running to show the tolerance towards intrusions. Detecting malicious web pages on a network is a tedious task. High-interaction client honeypots will not be able to completely detect malicious web pages, There will be many web pages carrying rootkit that contains executables which will harm several processes or modify certain entries in system files and also try to
access ports. To detect these kind of malicious web pages it requires a different approach.

Hengya Liu and others (2010) designed a high-interaction honeypot that detects the rootkits in the web pages. Chang and others also worked on a collaborative approach to improve the honeypot architecture. They integrated several existing honeypot solutions and created a new virtual honeypot architecture for conducting customized implementation of the algorithms that were implemented in each of the honeypots and fight in collaboration against intrusions. Shimoda (2010) developed and proposed an untraceable honeypot for large-scale applications. Shimoda and others used virtualization technologies to make the honeypot untraceable. They proposed a system called DarkPots, which had lot of virtualized honeypots with unused and nonconsecutive IP addresses in a production network. Since the probes were inefficient on a production network, it was difficult to detect the honeypot. Their solution worked fine without affecting the network traffic from 500 Mbps to 1Gbps.

A Comparative Study of Network Intrusion Detection Systems was done (2010) by Moses Garuba, Chunmei Liu, and Duane Frates. They have conducted an extensive study on the different Intrusion techniques and they also demonstrated that NIDS alone cannot handle both internal and external threats to computers. They proposed that Heuristic Based solutions are better than signature based solutions. Self Adaptivity and Dynamic analysis are the key features that have to be there in any NIDS as the responsiveness for any NIDS was determined by these properties.

The importance of dynamic behavior of the NIDS was demonstrated (2010) by Zang Qing Hua, Fu Yu Zhen and Xu Bu-gong. Luis Carlos Caruso and others have submitted their proof of concept on huge computing power requirement for signature based NIDS called SPP-NIDS. The limitations as
mentioned in Snort and findings by Miyuki Hanaoka, after a certain communication link speed NIDS will fail to perform due to the increased load and softwares like SNORT require a huge computing capability to handle communication line greater then 100Mbps. Miyuki Hanaoka and others have discussed the importance of collaboration between the security mechanisms and specifically the collaboration between many NIDS. They also demonstrated that redundant rules could be eliminated between the NIDS with a collaborative model. NIDS alone is not sufficient to handle entire range of threats and attacks on the computer networks.

Intrusion Preventive mechanisms will also help significantly in reducing the effect of an attack over a computer network. Network Intrusion Preventive mechanisms like traditional firewall along with strong authenticating procedures in collaboration with NIDS will make a computer network more secured. Simon P. Chung has experiment(2008) in their works. Softwares like IPTABLES on GNU/Linux can be used in setting up a firewall on an operating system with Linux as a kernel, where the version is higher than 2.4. With iptables and NIDS a variety of security related mechanisms are implemented in psad. Firewall play a vital role in NIPS, Despite taking all these precautions attacks still happen and the computer security system still fails to secure the computer networks in case of new type of attacks.

A mechanism where it would be possible for the attackers to get trapped unknowingly and avoid computer networks getting infected is essential. HoneyPots can be used to secure the computer network along with NIDS and NIPS. Honeyd is a small daemon that creates virtual hosts on a network. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. Honeyd enables a single host to claim multiple addresses. Honeyd improves network security by providing mechanisms for threat detection and assessment. It also
deters adversaries by hiding real systems in the middle of virtual systems. The Collaboration of NIDS system like SNORT, NIPS mechanism like IPTABLES and Honeypot like honeyd will make the Intrusion Collaborative Security System more powerful and robust. Knowing the enemies for a computer network with more practical details is demonstrated in the white papers of Honeynet project.

Every day computer networks are growing in a very large scale and there are more and more people trying to attack the networks. Hence securing the computer networks demands a distributed solution. Security system for each network will finally create a bigger system of security system which will be deployed in a distributed manner.

Jamshidi, M, Crossley, W.A, Sage, A.P, Keating, Kotov, Carlock, and Manthorpe gives the definition of System of Systems and explain the emerging characteristics of System of Systems where Internet is an example of such system. But integrating NIDS, NIPS (IPTABLES) with Honeypot and deploying these mechanism considering Security System as a System of Systems will yield better results. Though the security system is setup in each network it is very important to deploy the mechanisms at proper places.

A formal approach of Systems-of-Systems towards effective deployment of Intrusion Collaborative system in a distributed manner. Several experiments have been conducted by Leila Rikhtechi Afshin, Li Tian, You Yang and Jua Mi, Haifeng Wang and Qingkui Chen where they have discusses major challenges and issues associated with the distributed deployment of Intrusion Detection Systems in a large-scale and distributed networks.

Most of the network sniffers are based on the libpcap library which was used to develop the one time most widely used software called tcpdump. Tcpdump can be used for extracting simple information about the network
traffic. There were wrappers written to libpcap for other languages like java and named them as jpcap.

Snort is a very famous NIDS and almost used as a defacto software for reference. From the experimental level till the fullest deployment the snort software is used. The snort rules are written by the community and they are made available 15 days late to the public than to a subscriber.

Wireshark is the most popular sniffing tool used to study the flow of the network traffic. Along with the sniffing wireshark is capable of identifying the type of protocols and also provides lot of statistics about the connections during a session and provides exporting network data facility in several formats which helps in directly using for analysis in variety of ways with different programming languages.

Tcpdstat and tcptrace are other softwares which will help in generating some basic statistics about the network traffic which would be useful in classification of network traffic. Etherape is another simple tool which helps in viewing the network traffic graphically. Gawk and sed can also be used to extract statistical information from the tcpdump network data.

Mysql is a free ans open source database engine used by many organization to maintain their database could be used to log the network traffic and keep the information for further study or live analysis of the network traffic which would help identifying the threats or attacks over a network.

C,C++, perl, python, shell scripts are very powerful ways to program several programming languages programs can be combined to form an effective way to address the network security problem.
CHAPTER 2

Motivation and Objective of Present Work

Despite having variety of authentication, authorization and accounting mechanisms, still intrusions and attacks are happening over a computer network. Though powerful computing facility is there it has been impossible to trace each and every activity over the network due to the high increase in speed of communication lines and vulnerabilities in the application that are run on the networks. Tracing every packet is beyond the capacity of the computing facilities typically available in an organization with normal security principles applied. Understanding the behavior of the users over a period of time in a network is crucial to trace the attacks that could happen from inside the network. To trace the abnormal or anomalous or malicious activities in a network, it is very important to first have the proper database or signatures of the existing pattern of threats or attacks. Extracting key features from the network data is the most essential information to take any decision or set a rule or a policy to secure a network. Traditional approaches of tracing each and every packet will take enormous computing time so non-conventional approaches are required to classify the attacks which could be incorporated in any network intrusion detection and prevention system. To make any inference, efficient information extracting techniques have to be developed.

Understanding the behavior or the pattern of the network is the most demanding task for the security managers. As once the pattern of the network is understood, recognizing anomalous behavior of a user or malicious activities or any other types of attacks can easily be done with less computing facility. Also since the conventional approaches will not help in making the efficient security policies concepts like honeypots are looked at. Honeypots gives a different approach for solving security problems. Instead of securing the network completely, several resources could be compromised and the internal
and external threats are traced. This does not require huge computing facility and the whole network need not be compromised.

After having understood the pattern of the network, efficient information extraction techniques have to be devised using non conventional ways of solving the security problems which is not just enough. Over a network it also matters where and how the security principles and policies are applied for which efficient deployment of security mechanisms have to be done. Efficient Information extraction using non conventional approaches and effectively deploying the security mechanisms is the motivation behind the research.

Objectives of the work are to design a Attack classifier for a Honeypot based Intrusion Collaborative System . The Attack Classifier will not just be a traffic analyzer and not just classify the traffic or detect the known attacks but is also a dynamic rule creator for network intrusion detection system that will have a continuous learning model based on neural network mechanisms which will be able to predict new attacks based on the change in the behavioral pattern of the network traffic due to new attacks.

Another major work that is part of the research was to device an aggregated attack generator to test the stability and correctness of the proposed solution. An Attack aggregation of various attacks is designed.

The Research work also focuses on creating dynamic rules for Network Intrusion Detection or Prevention systems that will help in blacklisting the computers dynamically in a network which we have embedded in the Intrusion Collaboration System.

An important objective of the research work is to also device an efficient deployment mechanism of the security measures. Having a centralized mechanism will always make the security system vulnerable. Rather, a distributed way of deploying the security measures will always make the network more secured and robust.
The merging of computers and communications had heavy influence on the way computer systems are organized. The combination of mobile computing and wireless networks increased the usages of the computer networks. Currently computers and computer networks, especially internet has become a basic need for most of the day to day computing.

Preventive mechanism were later concentrated not only to the information but also to the hardware and other resources on the computer network. To prevent the unauthorized users accessing resources on the computer network, proper authentication mechanisms have to be implemented and supported by the Operating system or the applications that are developed above the operating systems. The design flaws in applications will make the application more vulnerable for attacks and these vulnerabilities will be exploited by the crackers to write malicious code and affect the networks. Also to prevent even more sophisticated crackers, measures have to be taken to secure the applications that we use on a network or those installed on the operating systems. This decision will lead us to a wise selection of the operating system and the applications to be used in the computer network.

Preventive mechanisms would be useful only in the case of known attacks, but would not be sufficient enough to handle new type of attacks or intrusion and even the threats from the internal source cannot be identified. For situations like this and as well as during a situation where there are suspicious events needs detective mechanisms. Detective mechanism will help in customizing the security mechanism to detect new type of intrusions. But detective mechanism will take lot of computing power. Response / Protection mechanism are those actions that would be taken after an attack is identified and the source and reason behind that attacks is found. Responsive / Protective mechanisms have to be very fast as these mechanism's decision will have impact over the future attacks that counts in rating the damages done to a computer network.
CHAPTER 3

Efficient Information Extraction for Traffic Anomaly Detection

In this chapter we are presenting an efficient way of extracting information from the network data. We are focusing on the libpcap compatible dumped data. Our experiments included the DARPA98 dataset. Information extraction techniques are discussed in the context of Traffic anomaly detection. The anomalies in the traffic are identified by using statistical models.

We have experimented extracting information by using scripting languages and DDL, DML statements after exporting the data from the libpcap based network data to plain text and as well as CSV files. Also combination of scripting languages and DDL, DML for extracting information is presented.

3.1 Terminology

**Intrusion**: In a computer network or network of networks Intrusion means the process of violating the security policies of the computer network. The Intrusions usually are attempts to sneak into a computer and try to harm the system.

**IDS**: IDS is Intrusion Detection System, which is one of the detective mechanism in Network Security. Intrusion detection mechanism are implemented to detect intrusions from outside and as well as inside the computer network. Intrusion Detection Systems could be implemented both on hardware, software and also as a combination of both. IDS main challenges are to detect the attacks with faster rate, detect new attacks.

**N-IDS**: Network Intrusion Detection System are those type of systems which will monitor the whole network and detect any intrusions. N-IDS will take care
of generating alerts in case of any intrusion or attacks from both external and internal resources.

**H-IDS:** Host Intrusion Detection System takes care of detecting intrusions or attacks on a host. Malicious programs like virus, worms and Trojans target individual systems and then try multiplying over a network. Each host will have to take precautionary measures to counter malicious programs.

**Hybrid-IDS:** Hybrid-IDS are mix of both N-IDS and H-IDS

**Attack:** Attack is an active sequence of related events that deliberately try to cause harm, such as rendering a system unusable, accessing unauthorized information, or manipulating such information. Both successful and unsuccessful attempts are considered.

**Alarm:** An alarm is a message about possible intrusion in a computer network. Alarm can be in the form of a mail to the computer network administrator. Alarm can be also in the form of a short message on a communication device. Depending on the layer at which the attack is done accordingly the alert would be generated.

**Libpcap:** Libpcap is a portable C/C++ library for network traffic capture. Using the applications program interfaces provided by the libpcap many applications are written which include tcpdump.

**Sed:** Sed is a stream editor used to perform basic text transformations on an input stream (a file or input from a pipeline). Sed works by making only one pass over the input(s). Sed's is capable of filtering text in a pipeline.
**Awk:** Awk is a filter and report writer. AWK is a tool used for processing rows and columns format of text based data. AWK has string manipulation functions, which could be used to search strings and modify the output. AWK has associative arrays which helps in faster processing. The name awk comes from the initials of its designers: Alfred V. Aho, Peter J. Weinberger and Brian W. Kernighan. Paul. Rubin wrote the GNU implementation, gawk.

**Shell:** A Unix shell is a command line interpreter for unix and unix like operating systems. Shell allows to do command line substitution, and also supports minimal programming constructs like variables, control structures and wildcards. Most popular shell are Bourne Shell and C Shell.

**Shell Script:** A shell script is a file which has sequence of commands that are executable over a shell.

**Tcpdump:** Tcpdump is a command-line packet analyzer on computer network. Tcpdump uses libpcap library to sniff the traffic on a network.

**Ethereal:** Ethereal is a network packet analyzer. Ethereal can capture network packets and can display that packet data as detailed as possible. A network packet analyzer is like a measuring device used to examine what's going on inside a network cable. Ethereal can be used to troubleshoot computer network, learn protocol implementation and network protocol.

**Etherape:** Etherape is another network monitoring tool which gives statistics about the traffic over network visually. But etherape doesn't allow to log the traffic to a file but only gives a pictorial representation of the traffic flow over a network.
**Tcptrace:** Tcptrace is another program which will sniff network and monitor data and report the events in an unorganized way.

**Sniffer:** A sniffer is a tool used to capture packets off the wire.

**Wireshark:** Wireshark is similar to ethereal in functionality and has much mode advanced features. Wire shark does deeper network protocol analysis and also network traffic analysis.

**Tcpdstat:** Program Written by Dave Dittrich, tcpdstat is a powerful tool that performs an in-depth protocol breakdown by bytes and packets. It further displays average and maximum transfer rates, IP flow information, and packet size distribution

**Snort:** Snort is a light weight free and open source network intrusion prevention and detection system (IDS/IPS). Combining the benefits of signature, protocol, and anomaly-based inspection, Snort is the most widely deployed IDS/IPS technology worldwide.

**Perl:** Perl is a general purpose high-level interpreted programming language.

**DDL:** Data Definition Language (DDL) statements are used to define the database structure or schema. CREATE - creates objects in the database, ALTER - alters the structure of the database, DROP - deletes objects from the database, TRUNCATE - removes all records from a table, including all spaces allocated for the records are removed, COMMENT - adds comments to data dictionary RENAME - rename an object are some of the DDL statements.
**DML:** Data Manipulation Language (DML) statements are used for managing data within schema objects. SELECT - retrieves data from the a database, INSERT - inserts data into a table, UPDATE - updates existing data within a table, DELETE - deletes all records from a table, the space for the records remain, CALL - call a PL/SQL or Java subprogram, LOCK TABLE - control concurrency are some of the DML statements.

**DCL:** Data Control Language (DCL) statements. GRANT - gives user's access privileges to database and REVOKE - withdraw access privileges given with the GRANT command are some of the examples of DCL statements.

**TCL:** Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions. COMMIT - saves work done, SAVEPOINT - identifies a point in a transaction to which a rollback can be done later, ROLLBACK - restores database to original since the last COMMIT, SET TRANSACTION - Changes transaction options like isolation level and rollback segment.

**Authentication:** Authentication is the process of confirming a claimed identity. All forms of authentication are based on something already know, something already we have, or something we are. Authentication on a computer network can be implemented in many ways. User name and passwords are one of the examples.

**Authorization:** Authorization is a process of granting permissions for someone or something to conduct an act. Even when identity and authentication have indicated who someone is, authorization is needed to establish what a user is allowed to do over a network or on a host.
**Cryptography:** Cryptography is the science of writing in secret code. Cryptography is one of the preventive mechanism in network security. Many cryptographic algorithms are used to secure the information over the network.

**Anomaly:** Deviation from the normal or common order or form or rule. Over a computer network the traffic flow would be of some magnitude, Sudden deviations from the normal traffic in a network would lead to a Anomalous Situation. Anomalies can be of many types. There can be traffic anomaly. There can be protocol anomaly where the packets would be malformed and sent intentionally to create problems for the computer network resources.

**Traffic:** Flow of Data over a computer network.

**Protocol:** Set of rules and regulations which govern the communication between two computers or nodes or any devices across a computer network. For example HTTP is protocol to transfer the information from one machine to another.

**Firewall:** Firewall is also one of the preventive mechanism. Firewall usually will be placed behind the main gateway of the computer network. Firewall prevents unauthorized access to the resources of a computer network.

**Router:** It is a combination of hardware and software which routes the packets at network layer of the TCP/IP or ISO-OSI architecture. Router will be capable of storing and forwarding the packets and as well as take decisions about where the packets have to be sent from an existing point.
3.2 Generic model of Intrusion Detection System

Intrusion Detection Systems detect attacks or intrusions or threats or abnormal or malicious activities in a computer network. A generic Intrusion detection systems will have modules to extract information, analysis module, a knowledge base about the previous attack and an optional response model which will take necessary actions against intrusions. As given in Figure 3.2.1 most of the standard IDS or NIDS implementations will have a module which will collect information about the computer network through various techniques.

![Figure 3.2.1 : Generic model of an Intrusion Detection System](image)

3.3 Structure of Libpcap captured data

Most of the information extractor on any network will have to sniff the communication link. Sniffers use pcap library like libpcap or wrappers of the pcap for their respective programming languages like jpcap for java. Libpcap library based applications gives applications will sniff the network and report the traffic flow to the detection engine. The format in which the sniffers
sniff the traffic through the interfaces and generate the output will not always be in a format by which necessary responses could be given. Also the logs generated by kernel, user applications, firewall and other sources also differ in providing information about the activities or events generated on a host or a server. Intrusion detection systems which are capable of detecting anomalies in traffic or protocol will need information about the traffic flow in the computer network in such a format that will favor raising alarms about the anomalies in the traffic. Libpcap based captured data will have the information organized in to several blocks. It will have Section Header Block which defines the most important characteristics of the capture file.

There will be Interface Description Block which defines the interfaces used for capturing traffic. Packet Block will contain actual data or part of the data. Simple Packet Block will have information only about one captured packet. Name Resolution Block defines mapping information. Capture Statistics Block will provide basic statistics about the reasons behind capturing the packets. Table 3.3.1 defines the Basic Block Structure of Libpcap based captured Data. Table 3.3.2 describes format of the Section Header Block. Table3.3.3 describes the Interface Description Block. Table 3.3.4 describes the structure of Packet Block, Table 3.3.5 describes the structure of Simple Packet Block and Table 3.3.6 describes the Name Resolution Block.

**Table 3.3.1 Basic Block Structure of Libpcap based captured data.**

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Type</td>
<td>Block Total Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Body [ variable length]</td>
<td>Block Total Length ( for backward file navigation.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Table 3.3.2 Section Header Block format.**

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte-Order Magic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Major Version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minor Version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Section Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (variable) (Hardware, OS, User Application)</td>
</tr>
</tbody>
</table>

**Table 3.3.3 Interface Description Block Structure**

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LinkType</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SnapLen</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (variable) [if_name, if_description, if_IPv4addr, if_IPv6addr, if_MACaddr, if_EUIaddr, if_speed, if_tsaccur, if_tzone, if_filter, if_os, if_fcslen]</td>
</tr>
</tbody>
</table>

**Table 3.3.4 Packet Block format Structure**

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Interface ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Drops Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Timestamp (High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Timestamp (Low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Captured Len</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Packet Len</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Packet Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (variable) (pack_flags, pack_hash)</td>
</tr>
</tbody>
</table>
Table 3.3.5 Name Resolution Block format

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Type (nres_endofrecord, nres_ip4record and nres_ip6record)</td>
<td>Record Length</td>
<td>Record Value</td>
<td>Other records</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Record Type = End of Records</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Captured Len</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (variable)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Packet Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (variable) (pack_flags, pack_hash)</td>
</tr>
</tbody>
</table>

Table 3.3.6 Simple Packet Block format

<table>
<thead>
<tr>
<th>First Byte</th>
<th>Second Byte</th>
<th>Third Byte</th>
<th>Fourth Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Length</td>
<td></td>
<td></td>
<td>Packet Data (Variable)</td>
</tr>
</tbody>
</table>

Interface Statistics Block which is optional will provide Interface Id Details, Time Stamps and few optional information. Interface Statistics Block will help in knowing few statistics about the type of the traffic and time stamps.

3.4 Information Formats Generated by Different Network Sniffers and Monitoring Tools

3.4.1 Pictorial information generated by EtherApe

EtherApe is a graphical network monitor for Unix and Unix Like Operating System. EtherApe was modeled after etherman. EtherApe works with Data link layer, IP and TCP modes, it displays network activity graphically. Hosts and links change in size with traffic. Color coded protocols display. It supports Ethernet, FDDI, Token Ring, ISDN, PPP and SLIP devices.
EtherApe can do both offline and online information extraction. EtherApe is capable of reading libpcap based data and provide online scanned information of the network. Computer Network Administrators will only be able to get minimal information from EtherApe and cannot be a major source of knowledge to take decisions about intrusions. EtherApe will only report about the activities over a computer network and is not capable of taking any decisions. EtherApe will be useful to find out the most active host or a node by which some inferences could be drawn about the behavior of a particular node.

Figure 3.4.1: Pictorial Data of Computer Network Activity generated by EtherApe

3.4.2 Sample Statistics Generated By Tcpdstat

Tcpdstat performs an in-depth protocol breakdown by bytes and packets. It further displays average and maximum transfer rates, IP flow
information and packet size distribution. The statistics generated are for each of the source machine. Figure 4.4.2.1 shows some of the statistics generated by tcpstat program.

**Table 3.4.2.1 shows a sample of statistics generated by tcpdstat program.**

<table>
<thead>
<tr>
<th>Port</th>
<th>Usage</th>
<th>IP</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>33.80%</td>
<td>131.243.89.214</td>
<td>8.50%</td>
</tr>
<tr>
<td>22</td>
<td>16.70%</td>
<td>128.3.2.102</td>
<td>6.20%</td>
</tr>
<tr>
<td>11001</td>
<td>12.40%</td>
<td>204.116.120.26</td>
<td>4.80%</td>
</tr>
<tr>
<td>2049</td>
<td>10.70%</td>
<td>128.3.161.32</td>
<td>3.60%</td>
</tr>
<tr>
<td>1023</td>
<td>10.60%</td>
<td>131.243.89.4</td>
<td>3.50%</td>
</tr>
<tr>
<td>993</td>
<td>8.20%</td>
<td>128.3.164.194</td>
<td>2.70%</td>
</tr>
<tr>
<td>1049</td>
<td>8.10%</td>
<td>128.3.164.15</td>
<td>2.40%</td>
</tr>
<tr>
<td>524</td>
<td>6.60%</td>
<td>128.55.82.146</td>
<td>2.40%</td>
</tr>
<tr>
<td>33305</td>
<td>4.50%</td>
<td>131.243.88.227</td>
<td>2.30%</td>
</tr>
</tbody>
</table>

**3.4.3 Sample of Statistics generated by tcptrace**

| DumpFile: trace.pcap          | IP flow (unique src/dst pair) Information                      |
|                               | # of flows: 1612801 (avg. 83.30 pkts/flow)                     |
| FileSize: 98876.89MB          | Top 10 big flow size (bytes/total in %):                        |
| Id: 200703011241              | 33.6% 3.2% 2.2% 1.5% 1.4%                                      |
| StartTime: (anonymized)       | 1.0% 1.0% 0.9% 0.8% 0.8%                                       |
| EndTime: (anonymized)         | ### IP address Information ###                                |
| TotalTime: 7216.13 seconds    | # of IPv4 addresses: 480065                                    |
| TotalCapSize: 96826.91MB      | Top 10 bandwidth usage (bytes/total in %):                     |
| CapLen: 1514 bytes            | 34.4% 34.4% 3.3% 3.3% 3.0%                                     |
| # of packets: 134347439 (96826.91MB) | 2.7% 2.3% 1.8% 1.5% 1.5%                                   |
| AvgRate: 113.10Mbps           | ### Packet Size Distribution                                  |
| stddev:47.96M                | (including MAC headers) ###                                  |
| PeakRate:260.92Mbps           | [ 32- 63]: 20839652                                          |
|                               | [ 64- 127]: 38798140                                         |

**Figure 3.4.3 : Sample of Statistics generated by tcptrace**
3.4.4 Sample of Statistics generated by Wireshark

Other than capturing the network data in libpcap compatible form wireshark has a feature to export the data in C array format, CSV format and few other formats customised for specific devices. Wireshark also does give minimum statistics, traffic flow analysis and connection informations. Wireshark is capable of providing the captured information both offline and online. The major limitation wireshark has is its inability to handle huge data files.

<table>
<thead>
<tr>
<th>Absolute Date,Time</th>
<th>Circuit ID</th>
<th>Delta Time</th>
<th>Destination Address</th>
<th>H/W Source address</th>
<th>H/W Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informatio</td>
<td>Protocol</td>
<td>Packet Length</td>
<td>Packet No</td>
<td>Net Src Address</td>
<td>Net Dest Address</td>
</tr>
<tr>
<td>Relative Time</td>
<td>Src Addr (Resolved)</td>
<td>Router ID</td>
<td>Src Address (IP)</td>
<td>Dest Address (IP)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4.4.1 Live Information Generated By Wireshark
3.5 Information Format required for Statistical Traffic Anomaly Detection

Intrusion Detection Systems will not have any statistical profiler for attacks or any traffic classifier. Based on the header information in each packet statistics will be generated. Most of the logic will be built in a per-processor. Usual practice would be to include the protocol decoder and have per-processor to customize the need for a specific application. Free and open source software like Snort which is most widely used NIDS encourages the users to add customised rules and also develop per-processors for their own need. To make statistical inferences to identify the anomalies it requires a well formatted information. Table 4.5.1 shows the format of the information required by a Statistical Traffic Anomaly Detection.

Table 3.5.1 Format of the information required by a Statistical Traffic Anomaly Detection

<table>
<thead>
<tr>
<th>Protocol</th>
<th>T1</th>
<th>T1 + 10 secs</th>
<th>T1 + 20 secs</th>
<th>T1 + 30 secs</th>
<th>T1 + 40 secs</th>
<th>T1 + 50 secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop</td>
<td>57</td>
<td>7</td>
<td>12</td>
<td>23</td>
<td>56</td>
<td>33</td>
</tr>
<tr>
<td>llc</td>
<td>112</td>
<td>233</td>
<td>66</td>
<td>12</td>
<td>235</td>
<td>23</td>
</tr>
<tr>
<td>ntp</td>
<td>232</td>
<td>333</td>
<td>643</td>
<td>435</td>
<td>676</td>
<td>132</td>
</tr>
<tr>
<td>arp</td>
<td>112</td>
<td>321</td>
<td>345</td>
<td>454</td>
<td>8</td>
<td>214</td>
</tr>
<tr>
<td>icmp</td>
<td>123</td>
<td>54</td>
<td>34</td>
<td>656</td>
<td>99</td>
<td>31</td>
</tr>
<tr>
<td>dns</td>
<td>232</td>
<td>45</td>
<td>556</td>
<td>5</td>
<td>98</td>
<td>334</td>
</tr>
<tr>
<td>tcp</td>
<td>223</td>
<td>665</td>
<td>56</td>
<td>567</td>
<td>654</td>
<td>433</td>
</tr>
<tr>
<td>Ssl</td>
<td>2</td>
<td>57</td>
<td>88</td>
<td>668</td>
<td>23</td>
<td>46</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
<td>68</td>
<td>89</td>
<td>78</td>
<td>34</td>
<td>44</td>
</tr>
</tbody>
</table>

T1:2010-06-05 17:20:32.068 - UTC format

The information given in the Table 3.5.1 would be required for each of the Ip address and in a particular point in time. The detailed information that would be required by a Traffic Anomaly Detector is as given in the Figure 3.4.5. To
apply any of the statistical techniques (Operational model, Markov model, Statistical moments Multi-Variate and TimeSeries model) for traffic anomaly detection, it is essential to provide the information in the format where applying statistical formulas will be easy. With the libpcap based application's captured data it is not directly possible to extract information and apply the statistical formulas directly.

![Diagram of traffic anomaly detection]

Figure 3.5.1 Detailed Statistical Information required by Traffic Anomaly Detector

The information given in the Table 3.5.1 would be required for each of the Ip address and in a particular point in time. The detailed information that would be required by a Traffic Anomaly Detector is as given in the Figure 3.5.1 To apply any of the statistical techniques (Operational model, Markov model, Statistical moments Multi-Variate and TimeSeries model) for traffic anomaly detection, it is essential to provide the information in the format where applying statistical formulas will be easy. With the libpcap based application's captured data it is not directly possible to extract information and apply the statistical formulas directly.

For Example: At 2010-06-05 17:20:32.068, if we want to know the number of packets of all possible types of protocol (like TCP, HTTP, FTP,
TELNET, SMTP, TCP etc) then it requires keeping counts for each and every packet type. Having the same information in a particular time range, that is between 2010-06-05 17:20:32.068 and 2010-06-05 17:25:32.068 is an another essential requirement. As shown in the Table 3.5.1, information provided between a time range has to be generated for each system on the network with all possible protocols that are allowed according to the security policies of an organization.

3.6 DARPA98 and DARPA99 DataSet

The standard corpora for evaluation of computer network intrusion detection systems was collected and distributed for the first time by Information Systems Technology Group (IST) of MIT Lincoln Laboratory, under Defense Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS) sponsorship termed as the DARPA98 and DARPA99 Dataset in the year 1998 and 1999 respectively.

The DARPA98 dataset consisted of the simulated network data. Dataset include a four hour subset of the training data, where file system dump of /root, /usr, /home and /opt was given along with Tcpump, DSM and ASCII BSM Data. The DataSet includes seven weeks of training data and two weeks of testing data which has network-based attacks in the middle of the normal background data.

Most of the research works related to evaluation of network intrusion detection systems will usually be run on the DARPA dataset. Extracting information from these dataset and generating information which suits anomaly detection system has to be done based on the libpcap data structures as the entire sample data is captured using libpcap based application.
3.7 Information Extraction using GAWK/AWK/SED from DARPA98 DataSet

CSV file has to be given as input to the information extracting program written using scripts.

3.7.1 Algorithm to extract information in the format given in Table 3.5.1 and Figure 3.5.1 using scripts.

#1 load the file for extracting information.
#2 converting time to second
#3 function for converting date to a integer number
#4 read starting date and end date for information extraction
#5 split starting date and ending date using split()
#6 convert the date to integer format
#7 read starting time and ending time of analysis
#8 split starting time
#9 read stating time and ending time of analysis
#10 read time gap in hr
#11 read time gap in min
#12 read time gap in sec
#13 split of record date in month day and year(which is 2nd field)
#14 record date in days from 1st year
#15 split of record time in hour minute second(which is 3rd field)
#16 record time in seconds
#17 record date-time in seconds
#18 if record date-time is between starting date-time and ending date-time
#19 if record date-time is between starting date-time and time gap
#20 increment the protocol count for each type of protocol
#21 declare associative array to count number of packet from source ip(4thfield) to protocol(6th field) for given time gap
#22 increment the time count according to the time gap mentioned
#23 count no of packet for a given time gap
#24 increment of starting date-time
#25 populate the array ipproto and sort
#26 populate the array protocol and sort
#27 calculation of starting time in hh:mm:ss format
#28 calculation of starting time +gap in hh:mm:ss format
#29 get file name according to timegap and date (if extraction is repeated)
#30 redirection of output to file in a CSV or TAB separated format.
Above algorithm provides the steps to extract information from the DARPA98 dataset. When the training data of a particular day in a week is fed to the scripts, scripts will ask for the date and time range for the analysis and also asks for the time gap in which the statistics are required. The DARPA98 dataset will have to be first loaded by a tool like wireshark and the required information can be exported to a CSV format file. CSV format file will contain only relevant fields of interest. Wireshark can generate up to 25 types of data from the traffic.

3.8 Information Extraction using DDL, DML procedures from DARPA98 DataSet

To get the list of protocols (should be stored in query file)
select distinct NIDS.Master.protocol from NIDS.Master order by NIDS.Master.protocol;

To Select Number of Protocols B/w a Specified Interval
mysql> select Atime,Atime + INTERVAL 5 MINUTE ,protocol, count(*) from Master where Atime Between 'Start_time' AND 'End_time' + INTERVAL 50 MINUTE group by protocol

Procedure to generate statistics
delimiter /
create procedure t2t_data (s_time datatime,e_time datatime,t_int int )
begin
while timestampdiff(frac_second,s_time,e_time) > t_int*60
  do
  select protocol,count(*) from master where atime between s_time and s_time + interval t_int minute group by protocol;
  if (select count(*) from master where atime between s_time and s_time + interval t_int minute group by protocol) then
select s_time, s_time + interval t_int minute as e_time;
end if;
set s_time = s_time + interval t_int minute;
end while;
end;

3.9 Information Extraction using combination of DDL, DML and Scripts

Steps to create a perl script that generates the statistical information as given in Table 3.5.1 and Figure 3.5.1 are given below and an example perl script for writing subroutines is also given

#1 Declare variables for Database, Table name, Database password and Query
#2 Generate error if source file not provided.
#3 Open outfile to write the Query
#4 Assign the argument to the query variable, database name and password.
#5 Extract the Protocols from the database
   E.g $cmd="echo \"select distinct protocol from NIDS\Master order by protocol\" >/query; #6 Send the query results to the database
   E.g mysql -u root -D NIDS -p".$DBPASS." </query >/parameters".
#7 Create Table with captured protocols;
#8 let $csv holds the Query to create table and generate CSV file.
#9 Query Results will be generated as CSV File.
E.g $cmd="mysql -H -u root -D NIDS -p".$DBPASS." </query |sed -f SedInfile | cut -d \" \" -f3->/Res_val".
#10 Print Extracting Data...."; &Insert_to_table($DB,$TABLE).
#11 Define a subroutine to be called for inserting data to table.
An Example of creating subroutine

sub Insert_to_table
{
    my($DBNAME, $TABLENAME)=@_;
    open(FILEIN, "/Res_val")||die("Unable to Open Source File...");
    open(FILEOUT, ">/Ins_query")||die("Unable to Create output File...");

    while(<FILEIN>)
    {
        @array_val=split(\,,$_);

        $protocol=" (`StartTime`, `EndTime`, ");
        $values=" (".$array_val[@array_val-3].",".
        $array_val[@array_val-2].",";
        $extval="";

        for ($i=0;$i<@array_val-3;++$i)
        {
            if ( ($i % 2) == 0 )
            {
                $protocol=$protocol."`
                $protocol=$protocol.
                $extval=$extval.$array_val[$i]." ";
            } else
            {
                $values=$values.$array_val[$i]." ";
                $extval=$extval.$array_val[$i]." ";
            }
        } #End of for loop

        print "\n".$extval;
        $extval="";
        chop($protocol);
        chop($values);
    }
}
3.10 Results and Discussions

3.10.1 Extraction information using DDL, DML

![Graph showing the relationship between time and number of records inserted into the database.](image)

**Figure 3.10.1 : Time taken for inserting records into the database**

While extracting the information using DDL, DML statements on database to generate statistics about the traffic and also for generating the statistics about the user behavior or any other program behavior is very time consuming. Since the engines of the Database Management Systems will have limitations with the number of inserts. After inserting the records into the
database, retrieval is not an issue but finding the patterns and creating views of
the patterns in the databases will take huge amount of time and memory.
For example if the database has m records and n columns and to generate
statistics as given in the Figure 3.10.1 requires m-1 x n-1 update query
executions, which works fine with less number of records. But the network data
is a very large database ranging from at least few GB to several hundred of GB
per day in a network of more than 100 machines. Figure 3.10.2 explains more
about the time taken for updating many rows and only one column which
reveals that it is inefficient to use databases to create views or create statistics
from which any intrusion detection could be carried out. Our approach has an
advantage over using just SQL (DDL and DDL or procedural) statements to
generate statistics

![Figure 3.10.2: Time taken for updating tables recursively for many rows and one column](image)

3.10.2 Information extraction and statistics generation using gawk / awk / sed

While extracting the information using gawk, awk / sed scripts alone for extracting information from the network data which are exported to a plain text space separated files using wireshark will yield faster results only up to certain number of records. After a certain number of records and depending upon the main memory and the processor speed, scripts written in gawk and sed will not give the results faster. Table 3.5.1. and Figure 3.10.2 gives an idea about the time taken to generate statistics from network data for traffic anomaly detection.

When the number of records crosses 25,00,000 records CPU and memory requirement will increase too much which is the major drawback of using scripts written in gawk /sed.

Table 3.10.1 : Time taken to generate statistics by gawk scripts with more than one time slots (columns)

<table>
<thead>
<tr>
<th>No of rows</th>
<th>T=15 secs</th>
<th>T=10 Secs</th>
<th>T=5 Secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.01</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.02</td>
<td>0.02</td>
<td>0.2</td>
</tr>
<tr>
<td>10</td>
<td>0.03</td>
<td>0.03</td>
<td>0.3</td>
</tr>
<tr>
<td>25</td>
<td>0.04</td>
<td>0.04</td>
<td>0.4</td>
</tr>
<tr>
<td>50</td>
<td>0.1</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0.4</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>250</td>
<td>3.5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>500</td>
<td>8.5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>1000</td>
<td>15</td>
<td>23</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 3.10.1 and Figure 3.10.5 tells about the time taken in case of only one slot of 10 mins, which is only one column and many rows. When the number of columns increases, creating views, generating statistics as given in Table3.5.1 and Figure 3.10.1 will take lot of time and after a certain limit the time taken is so huge that even high speed computers will fail to complete the tasks.
To generate similar statistics as given in Table 3.5.1 and Figure 3.5.1 using gawk / awk / sed will take less time compared to DDL or DML statements. But major disadvantage of using gawk / awk / sed is that for more than 2500000 records and for time slots more than 250 it will be very slow and many times processor will become completely busy and system hangs even on a system with more than 4 GB RAM.

![Information Extraction](image)

**Figure 3.10.3**: Time take to extract information using DDL/DML for more than one time slots

But when perl / gawk / sed / awk are mixed with pager / procedures, DDL/DML statements using database then the statistics will be generated much efficiently than just using DDL/DML / perl / gawk / awk / sed alone. From the Figure 3.10.4 and comparing this with Figure 3.10.2, Figure 3.10.3 Figure 3.10.4 and Figure 3.10.5 it implies that our approach using scripts along with sql statements have given better and efficient results.
Figure 3.10.4: Statistics generation time by gawk scripts with more than one time slots.

Figure 3.10.5: Time taken to extract information using DDL /DML procedures and perl/gawk /sed scripts
CHAPTER 4  
Design of Hybrid Framework Based on Honeypot For Dynamic 
Blacklisting of IP

In this chapter we present a framework written in java which allows to 
 integrate several other softwares for information extraction, classification and 
 trapping the malicious programs or users who would try to intrude the 
 computer network. The aim of developing this framework is to dynamically 
 blacklist those computers who does suspicious activity over the network.

4.1 Terminology

**Honeypot**: “Honeypot is a security resource whose value lies in being probed, 
attacked or compromised” , this is the most widely used definition. Honeypots 
are decoys designed to trap, delay, and gather information about attackers. A 
Honeypot can take forms, such as files or data records, or even unused IP 
address space

**Apache2**: 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

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

**Land**: 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

**Mailbomb:** 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.

**SYN Flood:** A SYN Flood is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree).

**Ping of Death:** The Ping of Death is a denial of service attack that affects many older operating systems.

**Process Table:** The Process Table attack is a novel denial-of-service attack that was specifically created for this evaluation.

**Teardrop:** The selfping attack is a denial of service attack in which a normal user can remotely reboot a machine with a single ping command

**Udpstorm:** A Udpstorm attack is a denial of service attack that causes network congestion and slowdown. When a connection is established between two UDP services, each of which produces output, these two services can produce a very high number of packets that can lead to a denial of service on the machine(s) where the services are offered.

**Smurf:** 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.
**Syslogd:** The Syslogd exploit is a denial of service attack that allows an attacker to remotely kill the syslogd service.

**Dictionary:** The Dictionary attack is a Remote to Local User attack in which an attacker tries to gain access to some machine by making repeated guesses at possible usernames and passwords.

**Ftp-write:** The Ftp-write attack is a Remote to Local User attack that takes advantage of a common anonymous ftp misconfiguration.

**Guest:** Similar to guest attack without passwords.

**Imap:** The Imap attack exploits a buffer overflow in the Imap server of Redhat Linux 4.2 that allows remote attackers to execute arbitrary instructions with root privileges.

**Phf:** The Phf attack abuses a badly written CGI script to execute commands with the privilege level of the http server.

**Java:** Java is a platform independent, pure object programming Language.

**Xlock:** In the Xlock attack, a remote attacker gains local access by fooling a legitimate user who has left their X console unprotected, into revealing their password.

**Ipsweep:** 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.
**Mscan:** Mscan is a probing tool that uses both DNS zone transfers and/or brute force scanning of IP addresses to locate machines, and test them for vulnerabilities.

**Nmap:** Nmap is a general-purpose tool for performing network scans. Nmap supports many different types of portscans—options include SYN, FIN and ACK scanning with both TCP and UDP, as well as ICMP (Ping) scanning.

**Saint:** SAINT is the Security Administrator's Integrated Network Tool. In its simplest mode, it gathers as much information about remote hosts and networks as possible by examining such network services as finger, NFS, NIS, ftp and tftp, rexd, statd, and other services.

**Satan:** SATAN is an early predecessor of the SAINT scanning program described in the last section. While SAINT and SATAN are quite similar in purpose and design, the particular vulnerabilities that each tools checks for are slightly different.

**Blacklisting:** Marking the IP addresses or computers which are trying to attack the computer network. Any traffic from such systems (IP addresses) will be blocked.

**4.2 Introduction to Honeypots**

Honeypots are decoys designed to trap, delay, and gather information about attackers. All the previous work in the field was related mainly to intrusion detection system, but in this research work, the highlight is more focused on the novel approach of creation of a Honeypot schema which is powered by intelligence along with the design of classifier. The output generated by the classifier generates a dynamic list of attacks, which are then
queued in the proposed Honeypot architecture built with neural network to understand various approaches and patterns of the attacker. The network administrator collects all such relevant information over the network itself allowing the inbound network connection from the attacker.

Network administrators usually use a firewall and an intrusion detection system (IDS) to protect their network. The firewall can control the inbound and outbound traffic according to the type of service requested, the user name, and the IP address of packets. The NIDS can be deployed between the local area network and the Internet or any other important gateway for detecting suspicious packets. The anamoly based NIDS usually will have a high false-positive ratio. The use of a honeypot can overcome the inherent deficiencies of the IDS and firewall. If a honeypot is deployed in front of a firewall, it is an early-warning system. If we deploy it behind the firewall, it serves as a defense-in-depth system and can be used to detect attackers who bypass the firewall and IDS or threats from insiders.

4.3 Hybrid Framework for blacklisting IP

Diagram5.3.1 describes the framework for blacklisting IP. We developed a framework using java for extracting information from the network, classify attacks which uses the neural network based libraries, continuous learning module about the behavior of the network in order to predict any possible intrusions. Application programming interfaces are provided for including the behavior of the network in honeypot systems and generate profile of intrusions. A web server is used to serve the logs to the attack classifier or inviting the cracker to try and attack it. Our framework gives option to generate a rule in case of any possible intrusion and the same rule or signature will be updated in the IPTABLES firewall and then also train the neural network classifier.
4.4 Neural Network-based Attack Classifier

In this section an introduction to neural networks is provided and then signatures of different types are explained. Also the method of how signatures of attacks are fed to the neural network modules is explained.

4.4.1 Introduction to Neural Networks

Brains are highly complex information processing systems. An Artificial Neural Network (ANN) is an information processing system based on mathematical and computational models, whose design is inspired by the structure and/or functional aspects of biological neural networks. Artificial Neural Network System are composed of a large number of highly interconnected processing nodes (neurones) working collectively to solve problems. ANNs gets knowledge from the previous incidents or examples. An
ANN focuses on a specific area, such as pattern identification or information classification.

Learning in biological systems involves a very complex and highly understanding process where their connections between the neurones play a key role. This is true of ANNs as well.

Neural networks, have unique and remarkable capacity to make inferences, take impulsive decisions looking at the complicated or even incomplete data. Neural networks could be used to extract patterns and detect trends. A trained neural network is like a "specialist" or “expert” in the area of information it has been given for analysis. This specialist or expert module can then be used to provide projections or extrapolations or predictions given new situations relevant to a particular topic of interest and answer variety of queries.

ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

There are three main types of training patterns to train the ANN. They are supervise, unsupervised and reinforcement learning.

Since network security is a complex and distributed task it is better to use unsupervised learning methodologies to train the ANN about the possible scenarios of intrusion over a computer network.

We implemented Kohonen's map to train the neural network classifier about the attack patterns, Kohonen maps are one of the Artificial Neural Network that gets trained using unsupervised learning technique.
4.4.2 Introduction to different types of attack signatures

Kristopher Kendall has given a detailed description of many attacks that were simulated and added in the DARPA98 dataset. Currently there are lot of variations of these type of attacks. In this section properties of different types of attacks and their signatures are explained.

Table 4.4.2.1 Summary of Denial of Service Attacks

<table>
<thead>
<tr>
<th>Name</th>
<th>Service</th>
<th>Vulnerable Platforms</th>
<th>Mechanism</th>
<th>Time to Implement</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache2</td>
<td>http</td>
<td>Any Apache</td>
<td>Abuse</td>
<td>Short</td>
<td>Crash httpd</td>
</tr>
<tr>
<td>Back</td>
<td>http</td>
<td>Any Apache</td>
<td>Abuse/Bug</td>
<td>Short</td>
<td>Slow server response</td>
</tr>
<tr>
<td>Land</td>
<td>N/A</td>
<td>SunOS</td>
<td>Bug</td>
<td>Short</td>
<td>Freeze Machine</td>
</tr>
<tr>
<td>Mailbomb</td>
<td>smtp</td>
<td>All</td>
<td>Abuse</td>
<td>Short</td>
<td>Annoyance</td>
</tr>
<tr>
<td>SYN Flood</td>
<td>Any TCP</td>
<td>All</td>
<td>Abuse</td>
<td>Short</td>
<td>Deny Service</td>
</tr>
<tr>
<td>Ping of Death</td>
<td>icmp</td>
<td>None</td>
<td>Bug</td>
<td>Short</td>
<td>None</td>
</tr>
<tr>
<td>Process Table</td>
<td>Any TCP</td>
<td>All</td>
<td>Abuse</td>
<td>Moderate</td>
<td>Deny new process</td>
</tr>
<tr>
<td>Teardrop</td>
<td>icmp</td>
<td>All</td>
<td>Abuse</td>
<td>Moderate / Long</td>
<td>Network slowdown</td>
</tr>
<tr>
<td>Udpstorm</td>
<td>syslog</td>
<td>Solaris</td>
<td>Bug</td>
<td>Short</td>
<td>Kill Syslogd</td>
</tr>
<tr>
<td>Smurf</td>
<td>N/A</td>
<td>Linux</td>
<td>Bug</td>
<td>Short</td>
<td>Reboot Machine</td>
</tr>
<tr>
<td>Syslogd</td>
<td>echo/</td>
<td>All</td>
<td>Abuse</td>
<td>Short</td>
<td>Network Slowdown</td>
</tr>
</tbody>
</table>
### Table 4.4.2.2 Summary of User to Root Attacks

<table>
<thead>
<tr>
<th>Name</th>
<th>Service</th>
<th>Vulnerable Platforms</th>
<th>Mechanism</th>
<th>Time to Implement</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eject</td>
<td>Any user session</td>
<td>Solaris</td>
<td>Buffer Overflow</td>
<td>Medium</td>
<td>RootShell</td>
</tr>
<tr>
<td>Ffbconfig</td>
<td>Any user session</td>
<td>Solaris</td>
<td>Buffer Overflow</td>
<td>Medium</td>
<td>RootShell</td>
</tr>
<tr>
<td>Fdformat</td>
<td>Any user session</td>
<td>Solaris</td>
<td>Buffer Overflow</td>
<td>Medium</td>
<td>RootShell</td>
</tr>
<tr>
<td>Load module</td>
<td>Any user session</td>
<td>SunOS</td>
<td>Poor Environment Sanitation</td>
<td>Short</td>
<td>RootShell</td>
</tr>
<tr>
<td>Perl</td>
<td>Any user session</td>
<td>Linux</td>
<td>Poor Environment Sanitation</td>
<td>Short</td>
<td>RootShell</td>
</tr>
<tr>
<td>Ps</td>
<td>Any user session</td>
<td>Solaris</td>
<td>Poor Temp File Management</td>
<td>Short</td>
<td>RootShell</td>
</tr>
<tr>
<td>Xterm</td>
<td>Any user session</td>
<td>Linux</td>
<td>Buffer Overflow</td>
<td>Short</td>
<td>RootShell</td>
</tr>
</tbody>
</table>

### Table 4.4.2.3 Summary of Probes

<table>
<thead>
<tr>
<th>Name</th>
<th>Service</th>
<th>Vulnerable Platforms</th>
<th>Mechanism</th>
<th>Time to Implement</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ipsweep</td>
<td>ICMP</td>
<td>All</td>
<td>Abuse of Feature</td>
<td>Short</td>
<td>Finds active machines</td>
</tr>
<tr>
<td>Mscan</td>
<td>Many</td>
<td>All</td>
<td>Abuse of Feature</td>
<td>Short</td>
<td>Looks for known vulnerabilities</td>
</tr>
<tr>
<td>Nmap</td>
<td>Many</td>
<td>All</td>
<td>Abuse of Feature</td>
<td>Short</td>
<td>Finds active ports on a machine</td>
</tr>
<tr>
<td>Saint</td>
<td>Many</td>
<td>All</td>
<td>Abuse of Feature</td>
<td>Short</td>
<td>Looks for known vulnerabilities</td>
</tr>
</tbody>
</table>
Table 4.4.2.4 Summary of Remote to Local Access

<table>
<thead>
<tr>
<th>Name</th>
<th>Service</th>
<th>Vulnerable Platforms</th>
<th>Mechanism</th>
<th>Time to Implement</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>Telnet, rlogin, pop, imap, ftp</td>
<td>All</td>
<td>Abuse of Feature</td>
<td>Medium</td>
<td>User-Level Access</td>
</tr>
<tr>
<td>Ftp-write</td>
<td>ftp</td>
<td>All</td>
<td>Misconfiguration</td>
<td>Short</td>
<td>User-Level Access</td>
</tr>
<tr>
<td>Guest</td>
<td>Telnet, rlogin</td>
<td>All</td>
<td>Misconfiguration</td>
<td>Short</td>
<td>User-Level Access</td>
</tr>
<tr>
<td>Imap</td>
<td>nmap</td>
<td>Linux</td>
<td>Bug</td>
<td>Short</td>
<td>Root Shell</td>
</tr>
<tr>
<td>Named</td>
<td>dns</td>
<td>Linux</td>
<td>Bug</td>
<td>Short</td>
<td>Root Shell</td>
</tr>
<tr>
<td>Phf</td>
<td>http</td>
<td>All</td>
<td>Bug</td>
<td>Short</td>
<td>Execute commands as user http</td>
</tr>
<tr>
<td>Sendmail</td>
<td>smtp</td>
<td>Linux</td>
<td>Bug</td>
<td>Long</td>
<td>Execute commands as user root</td>
</tr>
<tr>
<td>Xlock</td>
<td>X</td>
<td>All</td>
<td>Misconfiguration</td>
<td>Medium</td>
<td>Spoof user to obtain Password</td>
</tr>
</tbody>
</table>

4.4.3 Training Neural Network Module About Attack Signatures

The knowledge about attack signatures can be obtained from many location and resources on a computer network. Some of the scenarios over a computer network have to be understood by the neural network module. C++ neural network libraries were used to feed the following scenarios in order to train the classifier.

- immediate rise in no of mails sent from a machine
- immediate rise in no of mails sent to a machine
• No of SYN packets targeted for a particular machine that come from an unreachable host.
• ICMP packets > 64000 bytes
• ping the broadcast interface just before machine dies, the malicious broadcast would have put this task in the cronjob.
• sudden increase in the ssh connections
• request for repeated TCP RESET connection following a TCP CONNECT request
• inconsistent logins to a machine -- Authentication logs will revel that
• attempts to connect periodically on non well-known ports
• request to a name server with byte size large than 4096.
• execution of REGEDIT followed with execution of winlog.exe on windows machine
• ping packets for every possible machine on a network

The above knowledge and similar other information will be fed into the neurons.

4.4.4 Experimental Setup

The Framework was developed using Java on computers with GNU /Linux operating system. Experiments were conducted in four slots of seven weeks each. A local area network with 120 – 150 machines was used for experimentation. There was a dedicated web server set for collecting information about the possible IP for blacklisting and also for collecting and serving information from the proxy router. Several workgroup and subnetworks were created for experimenting with different type of intrusions. Each computer on the network was having 2GB RAM, 200 GB Hard Disk, Intel Dual Core Processor with the link speed of 100 Mbps. There were several servers, Main server was equipped with 10GB RAM, 500 Gb HDD and 2 quad core processors.
4.4.5 Blacklisting IP

Blacklisting IP involves blocking certain computer machines doing any interactions with other machines on the network, blocking certain services for a particular computer and blocking some computers for suspicious activities.

Blocking or Blacklisting IP is done after repeated anomalies reported by the classifier. Several attacks were simulated and the neural network module was trained for known intrusions for which it will raise an alarm and based on this information the IP was blacklisted. There are several ways to blacklist the IP. The IP behaving abnormally could be broadcasted to every other IP.

The IP could be entered in the gateway of the main firewall for denying service. Depending upon the type of intrusion, intrusions can be classified. For example, mail spam list, DNSBL (DNS-Based Block list).
4.5 Results and Discussions

The Framework developed will allow to implement several functionalities in a single window. The architecture of the framework provide flexibility in implementation of servers for serving the alerts and statistics about the traffic.

The hybrid framework will allow to implement both preventive and protective mechanisms.

Framework also includes honeypot schemes where the attackers could be trapped. The trapping modules mimic the existence of the actual machine and give opportunity for the intruder to interact with it and compromises its security level and encourages to exploit the weakness it will expose. The interactions will be continuously sent to the central location with a web server. Finally decision will be taken based on the type of interaction.

The hybrid framework has a continuous learning module which continuously learn about the activity over the computer network and finally provides a knowledge base about the intrusions.

Framework has option to setup Honeypot with NIDS or Honeypot with Firewall. NIDS rules like rules of snort can be implemented inside the honeypot. Firewall rules could be customized to monitor the every incoming outgoing traffic. Framework provides option to have customized log viewers or a pictorial representation of the computer network.

Framework has the option of implementing multiple web servers for viewing the alerts like using BASE or honeyview to view the SNORT alerts or the honeypot logs.
Applications like BASE, network monitors like etherape, honeyd logviewers and statistics generators can be instantiated with the help of the framework developed. The framework includes a module to monitor the local computer network which keeps track of all the computers on the network and will graphically display whether the computers are offline or online. The monitoring module also generates graphs about the traffic flow.

BASE helps in analyzing the alerts coming from a SNORT IDS system. BASE provides minimum statistics and also provides a web interface to directly work with the configuration files of Snort NIDS. Our framework will facilitate to run the snort NIDS at the background and allow the administrator to use tools like BASE or similar customized applications to work with the alerts generated.
Figure 4.5.2 Statistics generated by a log analyzers showing the summary of the most active and hour wise statistics about most active computer machines (IP)

As shown in the Figure above the framework will also facilitate to generate statistics by creating instances of the log analyzers like “honeyview” for viewing the honeyd servers.
Figure 4.5.3 Statistics generated by a log analyzers showing most active ports as summary and also hour wise.

Figure 4.5.4 Monitoring the local computer network using java based network monitoring module
CHAPTER 5
Design of Aggregated DoS Attack Generator

5.1 Introduction

Research work related to NIDS and NIPS always involves a deep understanding of a variety of attacks on Computer Networks. The most challenging task is to study the scenario during attacks. As the attacks on networks happen randomly or abnormally, it is not possible to guess the timings of attack and study their behavior. Our research focus is to understand the scenario during several types of Denial of Service (DoS) attacks so that the pattern of that behavior or attack is recorded and is further used to train the classifier based on neural networks which classifies the attacks that helps the honeypot based NIDS / NIPS in blacklisting the IP dynamically.

In this chapter the easiest technique to generate DoS attacks on single machine and as well as over a network using Java, C and lex programs is presented and also the process of extracting the pattern of a DoS attacks, training the neural network classifier with the extracted pattern of attacks is demonstrated. We focused our work to generate attack precisely on apache2 kind and then extended the same for other types like arp spoofing, nuke attacks. Instead of taking the attack individually we tried to aggregate and generate the DoS attacks by creating the same effects as that were done individually.

5.2 Terminology

**DoS** : Denial of Service is one of the attacks on a computer. Typically web servers will be targeted by the attackers. A single machine would be attacked in such a way that it would keep on responding to the requests

**DDoS**: Distributed Denial of Service conducted by many computers at the same time
IP Spoofing: Technique to conceal the identity a cracking computer will Masquerade as a trusted host. Variety of techniques exist to do IP spoofing, DOS attacks can also be generated using Spoofing as one of the techniques.

5.3 Generating DOS attacks using Java

We used Java.net package and jpcap library which is based on libpcap library. Using Java we hard coded the actual IP into the program and try to do the DoS attack on a server which was running web application server apache tomcat. Also at runtime IP address were given and simple request for connections were sent using sockets. This task was continuously done and we were able to see that the web application server stopped responding within a minute as it exceeded the number threads it could create. Within this program with minor modifications we were able to construct packets with different protocols and construct Ethernet frames and tried attacks of nuke and land types.

5.4 Algorithm for generating DoS attacks using Java

1. Retrieve the number of interfaces on the machine where the attack is generated from.
2. Open the required network interface to send the packet.
3. Create a TCP packet with specified port numbers, flags, and other parameters using TCP Packet class.
4. Specify the IPV4 header parameters.
5. Set the data fields of the packet.
7. Set frame type as IP.
8. Set source and destination MAC addresses, with Hexadecimal to decimal conversions.
9. Set the data-link frame as ether & send the packet.
5.5 Sample code for DoS Attack by Spoofing IP using java

//open a network interface to send a packet to
JpcapSender sender=JpcapSender.openDevice(devices[index]);

//create a TCP packet with specified port numbers, flags, and other parameters
TCP Packet p=new
TCP Packet(12,34,56,78,false,false,false,false,true,true,true,true,10,10);

//specify IPv4 header parameters
p.setIPv4Parameter(0,false,false,false,0,false,false,false,0,1010101,100,IPPacket.IPPROTO_TCP,
InetAddress.getByName("Mach1"),InetAddress.getByName("Mach2");

//set the data field of the packet
p.data=("data").getBytes();

//create an Ethernet packet (frame)
Ethernet Packet ether=new Ethernet Packet();

//set frame type as IP
ether.frametype=EthernetPacket.ETHERTYPE_IP;

//set source and destination MAC addresses
ether.src_mac=new byte[]{0,1,2,3,4,5};
ether.dst_mac=new byte[]{0,1,2,3,4,5};
for(byte b : ether.dst_mac)
System.out.print(Integer.toHexString(b&0xff)+ ":");

//set the datalink frame of the packet p as ether
p.datalink=ether;

//send the packet p
sender.sendPacket(p);

sender.close();
5.6 The algorithm for generating DoS attacks using Lex and C

1. Read destination-MAC, destination-IP
2. Read source-MAC, source-IP, Generate randomly.
3. Establish a connection with target and key busy as follows (multi-thread if necessary)
   a) Create spoofed packet with intended payload and send
   b) Sniff for response and reply
4. Creation and sending of spoofed packet
   a) Allocate memory for packet +headers
   b) Copy payload into packet
   c) Initialize the TCP header
   d) Initialize the IP header
   e) Initialize Link Ethernet Layer Header
   f) Set the checksum values
   g) Send packet
5. Sniffing for Response and Replying
   a) Create a raw socket in promiscuous mode
   b) Analyze each packet to determine the one with chosen MAC and IP.
   c) Parse payload
   d) Create a response using spoofed packet and send it.

5.7 The Pseudo-code for IP Spoofing is as follows

1. target_ip <- argv[1]
2. Perform steps 3 to 9 till program is interrupted / terminated
3. ip_rand <- rand()
4. a <- ip_rand & 0xFF
5. b <- (ip_rand >> 8) & 0xFF
6. c <- (ip_rand >> 16) & 0xFF
7. d <- (ip_rand >> 24) & 0xFF
8. Create command string with appropriate data Payload, data.txt (contains random info), TCP header info, randomly spoofed source_ip (a.b.c.d), dest_ip (target_ip), IP header info, source_port (random_int), dest_port (80)
9. Flush temporary memory spaces used
5.8 Sample code of the Lex and C programs to generate DoS attacks

for(;;)
{
    j=rand();
    a=j&0x000000FF;
    b=(j>>8)&0x000000FF;
    c=(j>>16)&0x000000FF;
    d=(j>>24)&0x000000FF;
    sprintf(command,"sendip -f data.txt -p ipv4 -is %d.%d.%d.%d -id %s -p tcp -ts 80-td 80 %s",a,b,c,d,dest,dest);
    system(command);
}

5.9 Experimental Setup

![Diagram of network setup]

Figure 5.9.1 : Generating DoS attacks

5.9.1 The experimental setup has the following configuration.

A LAN with a wired / wireless router
Several Servers (web, mail etc)
Attacking computer was having Core2 Duo processor and 4GB RAM and 150 GB hard disk.
All other configurations include standard computer Lab setup with necessary equipments switches and cabling for networking is done with the standard requirement to setup a LAN connection.
5.10 Results and Discussion

Figure 5.10.1 Sample output of a live log of DoS attacks

In the Figure 5.10.1 we could notice a DoS attack been generated. A datafile called data.txt will have a garbage data or any file could be used. This data file is for only to fill the payload part of a packet while transmission. We used the sendip package to build the packet and flood the same packet over the computer network. The number of IPS that were mimicked ranged from 50 to 500 per second depending on the requirement that is the number of spoofed Ips created depended upon the person who was generating attacks. The protocol of the packets were also randomly changed using a function to generate random port number while building the packet. So multiple types of attacks were also been able to generate with simple changes in the packet headers.
5.10.2 Wireshark Showing the DoS attacks using spoofing

The DoS attacks generated for internal testing of the computer network could be viewed by one of the popular tools like wireshark. In Figure 5.10.2 we could notice the DoS attack being generated on a computer with IP 192.168.1.1 using a simple TCP packet which are dummy.

Attack generator that is developed and presented above provides a very high level of flexibility to have customized intrusions for a particular scenario in a computer network. The attack generator designed was for testing the responsiveness of the security system implemented in the computer network limited to our organization.

Attacks generated were initially of Apache2 type and later we included the other type of DoS and DDoS attacks like SYN Flood, Ping of Death, Smurf attacks with a very few changes in the code.

Figure 5.10.2 : Wireshark showing the DoS attacks using spoofing
Attack generator presented here allows to do bit and byte level modification in the packet. With this kind of customization it is easy to include many type of attacks within the same period. The attack generator developed using java allows to test even at the bit level intrusions.

The effect or the impact over the computer network that is generated is like a benchmark for developing the countermeasure to tackle the intrusions / attacks. The computer network was clogged as given in the Table 6.9.1.

**Table 5.10.2 : Impact of DoS attacks over computer network**

<table>
<thead>
<tr>
<th>Number of Computers</th>
<th>Time taken for network clog</th>
<th>Recovery Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2 to 3 seconds</td>
<td>An external system has to be plugged in and trace the root of the attack if the recovery mechanism were not implemented at all</td>
</tr>
<tr>
<td>30</td>
<td>5 to 7 seconds</td>
<td>Firewall rules have to be written to decide what packets should flow in and out of every machine.</td>
</tr>
<tr>
<td>60</td>
<td>14 to 17 seconds</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>30 to 40 seconds</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>45 to 70 seconds</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>75 to 100 seconds</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>140 to 170 seconds</td>
<td></td>
</tr>
<tr>
<td>700 – 1000</td>
<td>240 to 300 seconds</td>
<td>Restarting DHCP server and Routers are essential</td>
</tr>
<tr>
<td>1000- 1300</td>
<td>350 to 500 seconds</td>
<td></td>
</tr>
</tbody>
</table>

Attacks generator were used for training the neural network based solution for the network intrusions. Every type of intrusions can be hard coded and the scenario can be trained to a neural network module which helps in generating the dynamic rules in case of any other abnormality in the network behavior.

Another attack generator developed using lex and C gives even more flexibility to include variety of attacks with a period of time. Also system level functions were used to generate attacks with deep impact. The Ip Spoofing
techniques used in our attack generator will create maximum load on the computer network resources.

For every second depending upon the processor speed and the network speed at least 150 IP spoofed packets are built. While testing the security measures adopted in the computer network we found that the GNU/Linux operating system has inbuilt capability to learn about the spoofing mechanism's and ignore those such spoofed packets. Apart from computers systems other network resources like routers, switches get clogged and if the flow of unwanted traffic is not controlled. Within a few minutes the network, switches and routers will be completely filled with the dummy packets.
CHAPTER 6
Design and Efficient Deployment of Intrusion Collaborative System

6.1 Design of Network Intrusion Collaborative System in a subnetwork

Intrusion Collaborative system is a combination of intrusion detection and intrusion prevention mechanisms. Iptables are used along with the Honeypots where the snort rules are embedded within the firewall to identify the know threats and Honeypots are deployed at appropriate location to trap the intruders and logs are taken for every abnormal activity and the probable threat is identified.

Figure 6.1: Design of Network Intrusion Collaborative System in a Subnetwork.
Above Figure 6.1 shows the components of the Network Intrusion Collaborative System. The NICS system includes combination of several Free and Open Source Softwares to implement preventive and detective mechanisms to secure the computer network. Figure 6.1 shows the setup of NICS in one of the subnets of a larger computer network. Each network will have firewall setup at the gateway using IPTABLES on GNU / Linux Operating System. To detect the known standard attacks and the internal attacks SNORT will be deployed in every subnet. One or many network resources like a computer machine or a combination of a computer, switch and a software like honeyd will be deployed as a Honeypot in each subnetwork to detect any anomalous behavior over the computer network. Each subnetwork will be connected to the universal gateway of the main network. A web server will be installed in each of the subnetwork to serve the necessary logs to any of the clients who want to interact and learn about the anomalous activities in a subnetwork. Every subnetwork will have a classifier which will classify the network activities according to their behavior which will help in predicting any future threats. There will be a proxy router set in each subnetwork to take care about connections with the universal gateway and in turn with the Internet. Mail server will help in sending messages to the administration department about any anomalous activities in the computer network.

6.2 Design of Network Intrusion Collaborative System in a complete network.

We experimented with a network of 1500 to 1600 computers with 16 class C Subnetworks. Each Subnetwork was configured with a separate NIDS (SNORT) and a firewall(IPTABLES – F1,F2,F3 and F4 in Diagram1) to detect and prevent any intrusions. Honeypot (H1,H2,H3 and H4 in Diagram6.2) was introduced in each of the network and information was extracted from its log continuously to detect any abnormal behavior in the network. Few
Subnetworks installed Honeypots on the virtual machines to hide from honeypot detectors. Snort rules and as well as customized rules were written and fed to the Honeypot. Firewall keeps updating the information about the anomalous or unusual activities. Firewall was implemented using IPTABLES provided by GNU/Linux Distribution. Debian Squeeze distribution was used to setup the firewall in each subnetwork. On some machines Fedora11 was used. Each Subnetwork had computers ranging from 75 to 100 with different operating systems running on them. Each Subnetwork had both wired and wireless switches. Only one DHCP server was used for the entire network. Each Subnetwork had the freedom to setup their own proxies (PR1, PR2, PR3 and PR4) and filter traffic according to their need. Figure 6.2 describes the setup and deployment of NICS for a complete network in an organization level.

![Design and deployment of Network Intrusion Collaborative System](image)

**Figure 6.2 : Design and deployment of Network Intrusion Collaborative System**
As described in the Diagram 6.2, Snort will detect all the known attacks based on its signatures. Firewall will prevent unauthorized activities. In the event of malicious or anomalous behavior statistical information can be extracted by the scripts and immediately reported to the classifier (C1,C2,C3 and C4 in Diagram 6.2) which will continuously keep creating a knowledge base (D1,D2,D3 and D4) of the behavior of the entire subnetwork which will be finally sent to a centralized server (D-main in Diagram 6.2) that keeps track of the activity of the entire network and helps the administrator in taking decisions. Administrator will decide on blacklisting IP in case of any attacks, threats or anomalous behavior based on the information obtained from attack classifier. NIDS was also deployed at appropriate locations in each sub network to check the internal attacks in every subnetwork. The information about the activity of the subnetwork within the network was always collected and sent to the central server to take corrective measures to avoid threats from internal resources. An aggregate DDOS attack pattern generator was used to test the capability of the collaborative system in tracing the internal intrusions and attacks. Also other attacks were generated using software tools like metasploit and nmap to test NICS.

Leila Rikhtechi Afshin and Rezakhani Roozbahani, Li Tian, You Yang and Jua Mi and Haifeng Wang have demonstrated in their research papers that when the size of the computer network scales up the deployment of any security mechanisms will become more and more complex. Lot of challenges will have to be faced and many issues have to be addressed. The Foremost task is the reliability of the information, hiding the honeypot, collecting the right information from the right resource in right time and the counter measures that have to be taken during any threat or intrusions and similar such activities. Also anomalous behavior have to be tracked and continuously keep learning about the network behavior and build knowledge base which will be further shared

We consider and propose that Intrusion Collaboration Systems should also be treated as a System of NIDS and NIPS and Honeypot and many other complex systems including the Subnetworks, These System of Security Systems(SoSS) in each subnetwork must collaboratively work together to fight against intrusions. SoSS will eventually become operationally independent of one another, managerially independent of each System, deployable in an evolutionary manner, emergent, distributed geographically, and heterogeneous while networking with other Systems. With such a complexity involved to automate the entire process a formal approach would be ideal to deploy a network intrusion collaborative system.

6.3 Deployment of Network Intrusion Collaborative System.

In this section system of system engineering concepts are introduced and then a formal approach towards deployment of NICS based on system of system concepts is proposed. The structure of logs from various resources is described. An algorithm to collect logs from different network resources and the control flow of the NICS is explained. Also few attacks generated for testing the NICS are described.

6.3.1 Introduction to System of System Engineering.

Mayer.M has defined architectural principles for System of System based on heuristics and through communication standards. He defines a simple taxonomy which distinguishes the normal systems and System of System with characteristics associated by the operational independence of the elements,
managerial independence of the elements, evolutionary development, emergent behavior, and geographic distribution. And also he categorized System of System based on the managerial control, Collaborative, Directed and Virtual System of System. He mentioned about the architectural principles telling about the stable intermediate forms of the systems and giving more important to the interface design as interface would be the most crucial part of the System of System where all communications and dependencies are exhibited. Also in the architectural principles he has mentioned about the importance of ensuring about the cooperation between the components. Mayer .M has given more importance about the communication substrates which defines how the components within and as a whole should communicate so as to exhibit the behaviors of the individual and as well as the System of System behaviors.

Manthrope, W.H, defines systems of systems in relation to the joint war fighting, System of System is concerned with interoperability and synergism of command, control, computers, communications, information(C4I) and Intelligence, surveillance, and Reconnaissance(ISR) Systems. Manthrope primary focus was on Information superiority with respect to military applications.

Pie defines system of system integration is a method to pursue development, integration, interoperability, and optimization of system to enhance performance in future battlefield scenarios. Here the primary focus was Information Intensive system integration with respect to military applications.

Kotov describes SoS as large scale concurrent and distributed systems that are comprised of complex systems. here the primary focus was on Information system of Private Enterprises. During 2004-2005 CMMI came up
with few suggestions about having a clear distinction between a system and System of Systems as to improve the success of System of System. The differences are given in the table below.

**Table 6.3.1 : 2004-2005 CMMI suggestions**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Single System</th>
<th>System of System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constituents</td>
<td>All constituents are known and visible</td>
<td>Changing and potentially unknown constituents entity assembling a system of systems may not know constituents until runtime. Constituent may not know it is part of a system of systems.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Predetermined by system owner and conveyed to constituents</td>
<td>Continuously evolving, cooperatively determined, and may or may not be known by systems participating in systems of systems</td>
</tr>
<tr>
<td>Control</td>
<td>Hierarchically structured with central control by system owner</td>
<td>System owners participating in a system of systems may have control over their systems, but they do not control how and when their systems are used in the system of systems. Entity assembling a system of systems has control over assembly but not over the participating systems.</td>
</tr>
<tr>
<td>Requirements</td>
<td>Defined and managed by system owner</td>
<td>Systems participating in the system of systems often have to anticipate how their system will be used.</td>
</tr>
<tr>
<td>Ownership</td>
<td>Pieces developed are owned, maintained, and evolved by system owner.</td>
<td>Constituent systems are independently owned, developed, maintained, and evolved.</td>
</tr>
<tr>
<td>Boundaries</td>
<td>Closed with clearly defined boundaries</td>
<td>In general, unbounded and part of larger systems of systems</td>
</tr>
<tr>
<td>Visibility</td>
<td>All aspects can be seen, understood, and controlled.</td>
<td>Components and process aspects are beyond control and visibility of developers, users, and</td>
</tr>
</tbody>
</table>

Carlock and Fenton describe Enterprise system of system engineering is focused on coupling traditional system engineering activities with enterprise activities of strategic planning and investment analysis. Here the primary focus was on Information Intensive systems with respect to private enterprises.

We consider and propose that Intrusion Collaboration Systems should also be treated as a System of NIDS and NIPS and Honeypot and many other complex systems including the Subnetworks, These System of Security Systems (SoSS) in each subnetwork must collaboratively work together to fight against intrusions. SoSS will eventually become Operationally independent of one another, Managerially independent of each System, Deployable in an Evolutionary manner, Emergent, Distributed Geographically, and Heterogeneous while Networking with Systems. With such a complexity involved to automate the entire process a formal approach would be ideal to deploy live network intrusion collaboration system.

6.3.2 System of System approach towards deployment of NICS: Formal Approach

G1: Closure Axiom
∀ a1, a2∈group G, a1 Bin a2 ∈ G

G2: Associative Axiom
∀ SoS1, SoS2∈SoSS, SoS1 Bin SoS2 ∈ SoSS

G2: Associative axiom or Associative Law
∀ a, b, c∈ group Ga∗b∗c = (a∗b)∗c
For a System of System to be called as group, when the elements try to establish relations with one another under a particular binary operation it should adhere to the associative axiom which states that
\[ \forall SoS1, SoS2, SoS3 \in SoSS, SoS1 \text{ Bin } SoS2 \text{ Bin } SoS3 = \{ SoS1 \text{ Bin } SoS2 \} \text{ Bin } SoS3 \]  

(iv)

G3: Identity Axiom
G contains an element ‘e’ such that
\[ \forall a \in Ga \ast e = e \ast a = a; \text{ ‘e’ is an identity element} \]  

(v)

System of Systems contains an element called as identity element ‘Se’ such that
\[ \forall SoSa \in SoSS, SoSa \text{ Bin } SoSe = SoSe \text{ Bin } SoSa = SoSa \]  

(vi)

G4: Inverse axiom
\[ \forall a \in G \exists a^{-1} \in G || a \ast a^{-1} = a^{-1} \ast a = e, \text{where } a^{-1} \text{ is called as inverse of } a. \]  

(vii)

With respect to SoS
\[ \forall SoSa \in G \exists SoSa^{-1} \in G || SoSa \ast SoSa^{-1} = SoSa^{-1} \ast a = e \]  

(viii), where SoSa^{-1} is called as inverse of SoSa.

So any non empty set that satisfies all the above four axioms will be called as Group. There will be scenarios where the sequence of occurrences of the elements is not important i.e S1, S2, S3 could be operated with the binary operation Bin in any order but still the solution is obtained. That means even though the binary operation is applied with the different combinations of the S1, S2, S3 would not matter and what matters is the only value that would be obtained after the application of the binary operation over S1,S2 and S3. this is formally derived in the axiom G5 called commutative law as stated below.
G5: commutative law :
\[ \forall a,b \in G, a \cdot b = b \cdot a \]  

(ix)
is also satisfied, then G is called as an Abelian Group with respect to \( \ast \) or commutative group with respect to \( \ast \).

With respect to System of System the commutative axiom states that
\[ \forall SoS1, SoS2 \in SoSS, SoS1 \text{ Bin } SoS2 = SoS2 \text{ Bin } SoS1 \]  

(x)
is satisfied then such a System of Security System should be defined as an Infinite Abelian Group. This definition will helpful when defining such a type of systems where there is flexibility in the order of joining of the elements into a group but the order of application of binary operations should not be changed.

6.3.3 Different Types Of Logs Used By The Classifier

Table 6.3.3.1 shows some of the log files from a typical Unix or Unix like Operating System. System Registry information and application logs will be collected similarly from other operating system. The log files are depended on the type of the operating system and the applications that run them. It is essential to configure a information extractor in such a way that it will automatically collect the information from the logs generated by any of the network resources. The log files will partly help in understand how a process /user/computer interacts with other process/user/computer over a network.
Table 6.3.3.1 Log files and their location required for classifier

<table>
<thead>
<tr>
<th>Log Type</th>
<th>Location of the Log file</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication logs</td>
<td>/var/log/auth.log/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Crond logs (cron job)</td>
<td>/var/log/cron.log/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Mail server logs</td>
<td>/var/log/maillog/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Qmail log directory</td>
<td>/var/log/qmail/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Apache access and error logs directory</td>
<td>/var/log/httpd/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Lighttpd access and error logs directory</td>
<td>/var/log/lighttpd/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>System boot log</td>
<td>/var/log/boot.log/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>MySQL database server log file</td>
<td>/var/log/mysqld.log/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Authentication log</td>
<td>/var/log/secure/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Login records file</td>
<td>/var/log/utmp or /var/log/wtmp</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Yum log files</td>
<td>/var/log/yum.log/</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>Firewall Logs (Iptables logs)</td>
<td>/var/log/iptables.log</td>
<td>Unix / GNU Linux</td>
</tr>
<tr>
<td>System Registry Information</td>
<td>C:\Windows\System32\</td>
<td>Windows</td>
</tr>
</tbody>
</table>

Other logs used by classifier includes information provided by honeypot machines and other alerts. Figure 6.3.3.1 shows an example of honeyd log during ICMP replies.

Depending upon the severity of the attack the alert messages could be ranked in such a way that the responses can be given either automatically or manually. Messages can be ranked as Urgent (URG), Alerts (ALRT), Severe (SEVER), Errors (ERR), Warning (WARN), Notify (NOTE), Information (INFO) and Bug (BUG#).
Depending upon the location and information extracted using logs will help in cross verifying the severity of the attack with the messages so as to take responsive action with more confidence and responses will not lead to another problem on the computer network.

Figure 6.3.3.1: Honeyd log during reply for ICMP requests under IP Spoofing

6.4 Attacks generated for internal testing of NICS

During setup and deployment of the NICS we tested the capabilities by generating few attacks internally. The attacks were generated using softwares like nmap, metasploit and few customised scripts to poof the IP and generate DDOS attacks.

6.4.1 Attacks generated using nmap

Following are some of the few sample of attacks done on the computer network to do a internal pilot testing. customized scripts were used for DDOS attacks.

```
#nmap -v -O --osscan-guess 192.168.1.1 | egrep 'MAC Address:|Devicetype:| Running:OS details:|Uptime guess:|Network Distance' >> systemdetails1.txt
```
Metasploit was used to write few exploits. Custom-built attacks were generated to test the capabilities of Network Intrusion Collaborative System like

### 6.4.2 Pseudo-code for IP-Spoofing is as follows

1. Define a variable called arget_ip and assign argv[1] to it.
2. Perform steps 3 to 9 till program is interrupted / terminated
3. Define a variable called ip_rand using any rand() method, not exceeding the network limit
4. Assign a variable a with and IP after ip_rand & 0xFF
5. Assign a variable b with and IP b after ( ip_rand >> 8 ) & 0xFF
6. Assign a Variable c after ( ip_rand >> 16 ) & 0xFF
7. Assigna Variable d after ( ip_rand >> 24 ) & 0xF
8. Create command string with appropriate data Payload , data.txt ( contains randominfo ) , TCP header info , randomly spoofed source_ip ( a.b.c.d ) , dest_ip ( target_ip ) , IP header info, Source_port ( random_int ) , dest_port ( 80 ).
9. system(command)

### 6.5 Results and Discussions

With the introduction of the NIDS, Firewall, Classifier and Honeypot in each SOSS , The security level of each subnetwork is increased by 50 %. Also the efforts in tracing the abnormal activities is minimized exponentially since most of the traffic filtering can be done at the firewall and NIDS. In Subnetwork firewall and NIDS takes care of detecting major known anomalous behavior using signatures and concentration will be on the new type of attacks which will be easily traced with the honeypot and reported to the classifier and again reported to the main server to record the abnormal activity
to take corrective measures by the administrator. This mechanism also reduced
the number of alarms usually raised by the NIDS upto 70%. Honeypots and
NIDS detect the abnormal behavior during an internal and external DDOS
attack within 5 seconds and were able to take corrective measures within 7
seconds whereas a network without Honeypot was clogged within 12 seconds
and the switches were completely non functional and entire system was
supposed to be shut down. With the introduction of the firewall and a proxy
router at each subnetwork, traffic filtering task was simplified. Universal
Gateway had the major responsibility of deciding the genuineness of an
activity. Always it is possible to sneak into the network but the intruder or
attacker will always look for the compromised system and Honeypots will be
able to easily trap them and report their interactions to the classifier to
dynamically either blacklist those machines or prevent them from doing further
damage, System of Systems approach will help in automating the process of
information collection from the classifiers, several scripts were written with
rssh commands to collect the logs from each of the Honeypots and also the
classifiers were automated which had helped in implementing the live Network
intrusion Collaboration Systems-of-Systems.

Care should be taken while deciding the number of systems to be at
compromised state. The major problem in creating a honeypot is that they take
more resources and also measures have to be taken to hide the honeypot.
Otherwise once the honeypots are detected the attacker will come to know
about it and will not try to do further interactions. Installing honeypots on
virtual machines will make the intruder tough to detect the honeypots. Services
like ssh should be run in nonstandard ports so that the intruder will be blocked
from trying standard ports for attacks. In each of the machine and as well at
the proxy routers and firewalls of the the subnetworks care has to be taken
about the traffic that each machine is allowed to generate. This will prevent
Table 6.5.1: Advantages of Network Intrusion Collaboration System

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources Used (CPU + memory)</td>
<td>2.00%</td>
<td>5.00%</td>
<td>15 – 20 %</td>
<td>20 – 25 %</td>
<td>&gt; 25 %</td>
</tr>
<tr>
<td>Internal Intrusion /Attacks / Threats</td>
<td>Can not Detect</td>
<td>Can Detect, @ each SoS</td>
<td>Can be trapped</td>
<td>Trapped + Responded</td>
<td>[4] + Live Protection.</td>
</tr>
<tr>
<td>GUI for results maintenance</td>
<td>Required [BASE]</td>
<td>[1] + custom</td>
<td>Custom Scripts</td>
<td>Custom-built Scripts</td>
<td>Custom-built Scripts</td>
</tr>
<tr>
<td>Max Link Speed Handled</td>
<td>N/W fails in 20 sec</td>
<td>N/W Failure within 1 minute</td>
<td>N/W failures are less</td>
<td>N/W failures are &lt; [3]</td>
<td>Network failures are Negligible</td>
</tr>
<tr>
<td>Effect of DDoS attacks</td>
<td>N/W clogs in 5 secs</td>
<td>N/W clogs in 2 minutes</td>
<td>N/W wont clog, logs generated</td>
<td>N/W wont be clogged, logs generated</td>
<td>No n/w clogs, very specific alerts raised</td>
</tr>
<tr>
<td>Effects of IP Spoofing</td>
<td>N/W clogs in 5 secs</td>
<td>N/W clogs in 2 minutes</td>
<td>N/W clogs in 2 minutes</td>
<td>N/W clogs, logs generated</td>
<td>No n/w clogs, very specific alerts raised</td>
</tr>
<tr>
<td>Intrusion Detection Time</td>
<td>3-5 sec for known attacks</td>
<td>10%-20% &gt; [1] 26% &gt; [1] can also detect new attacks</td>
<td>40% &gt; [1], can detect new attacks</td>
<td>50% prevented and Live Intrusion Detection</td>
<td></td>
</tr>
</tbody>
</table>

around 30 % to 50% of the attacks that could be possible from an insider. Guest Attacks, like when a new person enters the network with a laptop or any other device at a compromised state, then there are always chances of attacks getting multiplied. During this situation appropriate permissions and network policies have to be set for the guests otherwise it will be a very hard task to recover from an attack once they start multiplying.
Individual queries for detailed analysis in each subnetwork during portscan

```sql
mysql> select s.sig_name, count(*) as count from event e, signatures where e.signature=s.sig_id group by e.signature order by count desc;
```

<table>
<thead>
<tr>
<th>sig_name</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(portscan) Open Port</td>
<td>188</td>
</tr>
<tr>
<td>COMMUNITY SIP TCP/IP message flooding to SIP proxy</td>
<td>144</td>
</tr>
<tr>
<td>(snort decoder) Bad Traffic Same Src/Dst IP</td>
<td>115</td>
</tr>
<tr>
<td>BAD-TRAFFIC same SRC/DST</td>
<td>115</td>
</tr>
<tr>
<td>(portscan) TCP Portscan</td>
<td>33</td>
</tr>
<tr>
<td>(portscan) TCP Portsweep</td>
<td>5</td>
</tr>
<tr>
<td>(spo_bo) Back Orifice Snort buffer attack</td>
<td>4</td>
</tr>
<tr>
<td>(spo_bo) Back Orifice Traffic detected</td>
<td>4</td>
</tr>
<tr>
<td>(http_inspect) BARE BYTE UNICODE ENCODING</td>
<td>1</td>
</tr>
<tr>
<td>(spp_ssh) Protocol mismatch</td>
<td>1</td>
</tr>
<tr>
<td>COMMUNITY SIP DNS No such name threshold - Abnormally high count of No such name responses</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 6.5.1 : Deep attack analysis**
CHAPTER 7

Summary and Conclusions

This chapter summarizes the results and experiments described in the thesis; as well as discusses overall solutions which are related to network security in general, network intrusion detection in particular.

My research in network security area was motivated by questions like:

1. How to detect any new attack / intrusion in a computer network?
2. How to efficiently extract information from the computer network traffic and understand the behavior of computer network in order to suspect or predict a possible attack or intrusion?
3. How to fight against threats / intrusions / malicious activities within the network and from out side the network , collectively in a computer network , in a distributive and collective manner?

The broad objective of this work was to design a security system for a computer network which reduces the number of false alarms raised in a typical network intrusion detection system. The objective of this work also focuses on generating dynamic rules during any new intrusion or abnormal activity over a network. The work identifies the significance of collaborative effort to collectively fight against intrusion in a distributive way.

The objective also include automating the process of extracting appropriate information from the network to take decisions against malicious, abnormal and suspicious activities by using less resources . Another generic objective of our work is efficient deployment of security mechanisms. In a computer network whose size and complexity are growing continuously needs continuous monitoring and also security mechanisms have to be deployed at appropriate locations. An efficient deployment of security mechanism inline with the System of System engineering concepts is also one of the main objective of our research work.
The specific objective of this work were outlined as follows:

**Efficient Information Extraction for Traffic Anomaly Detection**

In chapter 3 a generic model of NIDS is introduced and most essential structure (libpcap) of the information extracted from the communication link of a computer network is also introduced. The libpcap structure is the key knowledge required to work with information extraction from the network data. Few information extractors and their output formats are introduced in this chapter like etherape, tcptrace, tcpdstat and wireshark.

The importance of the format of the information that is required to draw conclusions about any possible attacks / intrusion or to detect traffic anomalies is raised in this chapter. An efficient information extractor to detect traffic anomalies was designed.

Several different tools and techniques were experimented and among them we found a technique which was more efficient in extracting information from the libpcap based network data. The information extraction technique we have introduced uses a combination of scripting languages like sed, awk, perl and DDL DML statements.

We found that using database alone and with DDL and DML statements for a NIDS like Snort or any libpcap format data dumped to a RDBMS will not give faster results while creating views or inserting records or retrieving. Also just using the scripting languages like sed / awk/ perl also will not yield faster results due to the limitation with the size of the associative arrays.

The combination that we experimented gave 60% to 70% better results than other methods alone. In this chapter it is also taken care to provide information in a time range of seconds with protocol that will help a lot in identifying traffic anomalies.
Design of Hybrid Framework based on Honeypot and dynamic rules for Dynamic Blacklisting of IP

In chapter 4 a framework is presented which was designed mainly using java technologies that provides many application program interfaces to instantiate other softwares to collaboratively work towards blacklisting computer machines rather their IP that are involved in malicious / abnormal activity over a computer network.

The Honeypot concepts are introduced in this chapter which are a kind of trapping modules that makes the intruder believe that there is a actual machine interacting with it in a compromised state of network security.

Honeypot based schema is implemented to blacklist the malicious / abnormal computers. Honeypot schema helps in collecting the information from other modules that generate logs about activities over the network.

A neural network based classifier will be continuously learning about the network activities and also help in classification of the network intrusions.

The Framework developed can run on any platform which has jre and provides ready to use api to instantiate variety of standard tools for monitoring alerts, generating statistics and classifying attack. The framework also allows to customize the modules according to the need of the administrator. Framework proposed will reduce 30% to 40% of time and cost involved in implementing the modules for the tasks mentioned above.

Design of Aggregated DoS Attack Generator

In chapter 5 a design of computer network attack generator is presented. The attack generator was designed to generate majority of Denial of Service attacks and its variants. The attack generator was designed only to test the capabilities of the security mechanisms of the the organization where the experiments were conducted. The results obtained show encourages to use the attack generator as
a benchmarking tool while testing any security mechanisms to fight against DOS attacks. The computer network consisting of computers with number varying from 10 till 1300 were able to be brought down in a very short time ranging from 5 to 500 seconds.

Efforts involved in design of the DOS attack generator over a computer network is simplified by making use of the packet building utilities and C programming language. The attack generator will also allow to extend the program to generate other type of attack with simple modifications.

The major advantage of the attack generator presented is that it will give an aggregated effect of all variants of DOS attacks.

**Design and Efficient Deployment of Intrusion Collaborative System**

In chapter 6 a collaborative system of intrusion prevention and detection modules is designed and also an efficient way of deploying the collaborative system is presented. System of system engineering concepts are used to design the collaborative system. Every network will have a network intrusion collaborative system and each of the network intrusion collaborative system will communicate and collaborate to collectively fight against the intrusions. Systems like SNORT, neural network based attack classifier, log collector, IPTABLES and Honeypot are used to design the collaborative system.

A formal approach in deploying the network intrusion collaborative system that helps in automating the process of securing the computer network is also proposed.

Since each of the subsystems are operationally and managerially independent, will pose lot of challenges to build a security system which will have emergent nature and distributed geographically. Many challenging issues were addressed during implementation and the whole system well tested using customized attack generation.
Results shows that the computer networks could be secured upto 95% using NICS approach. Another 5% is always difficult due to flaws in design and threats generated internally. The intrusion collaborate system will use 25% less processing power than other systems proposed by many researchers. Live protection can be provided with the help of NICS. Major problem of false-negative alarms is addressed. Upto 60% of the false negative alarms are reduced. This reduction in false-negative alarms was because of using honeypots. The computer networks going down are also very less when the security mechanisms are implemented in a NICS manner.

**Future Work:**

The Information extraction techniques proposed are currently functioning to a limited speed of 10 MBPS communication link. In future we are migrating our information extraction techniques to heterogeneous environment, which includes developing algorithms in languages like OpenCL that will help in processing the data in a communication link with a speed of 10 GBPS. Also the algorithms and techniques that emerge out of these will be made GPU compliant.

Traffic anomalies, protocol anomalies, abnormal and malicious activities could be efficiently traced and response mechanism can be implemented if there are stronger deployment techniques of security mechanisms over a network in a distributed way. In future we will be improving the honeypot based deployment mechanisms that will reduce the efforts in detecting the attacks.
REFERENCES


Analysis, WenJie Tian; JiCheng Liu, Intrusion Detection Model Based on Support Vector Regression and Principal Components. ( 2009 ), WNIS '09. International Conference on Wireless Networks and Information Systems, Page(s): 129 – 132


Bo Wang, Ping Zhu, Qiaoyan Wen, Xiaojun Yu, “A Honeynet-Based Firewall Scheme with initiative security strategies” (2009), IEEE 978-1-4244-5273-6/09


Chris Chambers, Justin Dolske and Jayaraman Iyer, (2004), “TCP/IP Security”, Department of Computer and Information Science, Ohio State University, Columbus, Ohio 43210
Christoph L. Schuba and Eugene H. Spafford, “A Reference Model for Firewall Technology”, (1997), COAST Laboratory, Department of Computer Sciences, Purdue University.


Desheng Fu, Shu Zhou, Ping Guo, “The Design and Implementation of a Distributed Network Intrusion Detection System Based on Data Mining”, (2009), World congress on software engineering, DOI 10.1109/WCSE.2009.225, PP 446-450


Ehab S. Al-Shaer and Hazem H. Hamed, “Firewall Policy Advisor for Anamoly Discovery and editing”, MultimediaNetworking Research Laboratory, (2003), School of Computer Science, Telecommunications and Information Systems, DePaul Universiv, Chicago, USA

Errin W. Fulp, “Parallel Firewall Designs for High-Speed Networks”
Department of Computer Science, (2006), Wake Forest University, Winston-Salem


Fabrice Gadaud, “NIDS Architecture for Clusters” , (2005), IEEE-0-7695-2387-0/05/

Firewall Setup on Gnu/Linux, IPTABLES,
http://www.aboutdebian.com/firewall.htm

Firewall_(computing) , wikipedia resource
http://en.wikipedia.org/wiki/Firewall_(computing)


Guangming Yang, Jian Xu, Dongming Chen and Zhiliang Zhu, Research of Intrusion Detection System Based on Vulnerability Scanner, (2010), IEEE, 978-1-4244-5848-6/10/

Haifeng Wang and Qingkui Chen “Design of Cooperative Deployment in Distributed Honeynet System”, (2010), 14th International Conference on Computer Supported Cooperative Work in Design, Pages 711-716

Hamed Salehi, Hossein Shirazi, Reza Askari Moghadam, “Increasing overall network security by integrating Signature-Based NIDS with Packet Filtering Firewall”, (2009), International Joint Conference on Artificial Intelligence, IEEE, DOI 10.1109/JCAI.2009.12


Honeyd, http://www.honeyd.org/


http://packetstormsecurity.org/0310-exploits/php67.txt

http://packetstormsecurity.org/advisories/iss/iss.summary.6.6

http://packetstormsecurity.org/advisories/iss/summary/iss.summary.5.8

http://packetstormsecurity.org/DoS/IGMPNukeV1_0.zip

http://packetstormsecurity.org/DoS/killwin.c

http://packetstormsecurity.org/DoS/targa.c

http://packetstormsecurity.org/poisonpen/u_utils/winnuke.c

http://packetstormsecurity.org/Win/porting.zcools.zip

http://packetstormsecurity.org/Win/STC3.zip

http://project.honeynet.org/book/

http://sixrevisions.com/resources/the-history-of-the-internet-in-a-nutshell/

http://www.computerhope.com/history/internet.html (latest)
http://www.cs.ucl.ac.uk/staff/mrogers/cache/ipspoof1.txt

http://www.iana.org/assignments/port-numbers

http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/attackDB.html

http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/docs/attackDB.html

http://www.napster.com/


http://www.snort.org


IANA http://www.iana.org/

IEEE Deception Toolkit http://www.all.net/dtk


John Von Neumann, “Thoery of Self-Reproducing Automata”, (1996), University of Illinois Press,


Kelly Jackson Higgins http://www.darkreading.com/security/vulnerabilities/


Luis Carlos Caruso, Guilherme Guindani, Hugo Schmitt, Ney Calazans, Fernando Moraes, (2007) SPP-NIDS-A Sea of Processor Platform for Network Intrusion Detection, 18th IEE International Workshop on Rapid System Prototyping,

Mantrap Honeypot http://www.mantrap.com


Metasploit, http://www.metasploit.com


Min Qin; Kai Hwang; Frequent episode rules for Internet anomaly detection, Proceedings of 3rd IEEE International Symposium on Network Computing and Applications, (2004), Page(s): 161 - 168


Mysql, http://www.mysql.com/


Mohamed G. Gouda and Alex X. Liu “A Model of Stateful Firewalls and its Properties”, (2005), DSN'05, IEEE -0-7695-2282-3/05


N. Paulauskas and E. Garsva, “Computer System Attack Classification”, (2006), Vilnius Gediminas Technical University, Department of Computer Engineering, ISSN 1392 – 1215


Nmap, port scanning tool developed by Fyodor http://www.insecure.org/nmap


Quan Huang, Shengke Qiu and Shicun Qin, Cheng Cao, “An Embedded Firewall Based on Network Processor” (2005), ICESS-05, 2005, 0-7695-2512-1/05


RFC: 793 http://www.faqs.org/rfcs/rfc793.html  transmission control protocol


sed, http://www.gnu.org/software/sed/

Senda HAMMOUDA, Lilia MAALEJ and Zouheir TRABELSI, “Towards Optimized TCP/IP Covert Channels Detection, IDS and Firewall Integration”, (2008), IEE 978-2-9532443-0-4


Simkhada K, Taleb T, Yuji Waizumi, Jamalipour, A. Nei Kato and Yoshiaki Nemoto, “An Efficient Signature-Based Approach for Automatic Detection of Internet Worms over Large-Scale Networks”, (2006), ICC ’06. IEEE International Conference on Communications Vol5, 06, Page(s): 2364 – 2369

Simon P. Chung and Aloysius K., Mok phchung, mok , (2005), Collaborative Intrusion Prevention, N00014-03-1-0705

Specter Honeypot http://www.specter.com


Tcpdump http://www.tcpdump.org/

Tcptrace, http://www.tcptrace.org/


Tich Phuoc Tran, Jan, T, “Boosted Modified Probabilistic Neural Network (BMPNN) for Network Intrusion Detection”, (2006), IJCNN '06. International Joint Conference on Computer and Information, Page(s): 2354 - 2361


Vinu V Das, “Honeypot Scheme for Distributed Denial-of-Service Attack “, (2009), International Conference on Advanced Computer Control, PP 497-452


Well known Ports, http://www.iana.org/assignments/port-numbers


Wireshark, http://www.wireshark.org/


Xiaoyan Sun, Yang Wang, Jie Ren, Yuefei Zhu, & Shengli Liu, “Collecting Internet Malware Based on Client-side Honeypot”, (2009), The 9th International Conference for Young Computer Scientists, PP 1493 -1499


LIST OF PUBLISHED PAPERS

I. National Conference

NIL

II International Conference


III International Journal


VITAE

Mr Renuka Prasad B obtained bachelor degree in computer science from S.J.R.C., Bengaluru, Karnataka and obtained Master of Computer Applications from R.V.C.E, Bengaluru, Karnataka. He is currently working in the Department of Master of Computer Applications of R.V. College of Engineering, Bengaluru as Assistant Professor. He is interested in the area of network security and System of Security Systems. He has implemented several security mechanisms for the institution computer network which include Honeypot based Network Intrusion Detection Systems. Mr Renuka Prasad B is also interested in FOSS (Free and Open Source Software) solutions related to network security. He has implemented and customized several existing FOSS solutions related to network Security.