Intrusion Prevention Scripts to Detect Malicious Traffic using Bro

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

By:
Navdeep Singh
(Roll No. 80631012)

Under the supervision of:
Dr. Maninder Singh
Assistant Professor, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JUNE 2008

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Intrusion Prevention Scripts to detect Malicious Traffic using Bro"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr Maninder Singh. The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Navdeep Singh)

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

(**Dr. Maninder Singh**)
Supervisor
CSED Department
Thapar University
Patiala

Countersigned by

Acknowledgement

Many people have been a part of my post- graduate education as teachers and friends. Here, I want to take the opportunity to thank them all.

First of all, I am deeply indebted to my supervisor Dr. Maninder Singh, Assistant Professor, Computer Science & Engineering Department, Thapar University, Patiala whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

I also want to extend my thanks to Dr. Seema Bawa, Professor & Head, Computer Science and Engineering Department, Thapar University, Patiala and the entire faculty members for their excellent guidance and encouragement right from the beginning of this course.

I would also like to gratefully acknowledge the support of my classmates Maninder Singh and Sanmeet Kaur. I want to thank them for all their help, support, interest and valuable hints.

Most of all, I would like to thank my family, and especially my parents, for their endless love and support.

Navdeep Singh
(80631012)

Abstract

Computers have virtually revolutionized every sphere of our life. The rapid growth in the development of computers focused primarily on making the computer easy to use i.e. usability. The idea was to make computer easy to use for all sections of society. The rapid growth did not emphasize much on the security of the Computer system thereby rendering system as vulnerable to attacks. Had security been considered earlier in the development of computer system, our systems would have been more secure these days. Thus the preference of usability over security has made system more prone to attacks. Network security is an ongoing process that helps keep unauthorized parties from gaining access to the network. Securing any Network infrastructure is like securing possible entry points of attack. Any networked system where security or privacy protection of assets is valued needs security experts to protect and control it.

Further, Internet has made hacking much easier. The skill level required for hacking has gone down considerably. One can easily get exploits for latest vulnerabilities and threats. There are many tools and techniques available for securing networks like Firewalls, IDS etc. and until now they are used very frequently by nearly all the organizations to safeguard information and other critical data but these are not sufficient for implementing complete security because the intruders have become more smarter. In order to access information they use various alternatives like if they are not allowed access by certain means like blocking IP address, then they use some other way to access information like they use certain concepts like http tunneling etc. In this way various protocols get hidden under other protocols and then the information can be easily accessed by bypassing the security system because the security system checks just the header information and not the content hidden inside the packet and because of that these type of attacks often go undetected .So there is a need of detecting these type of intrusions which is solved because of the evolution of a new concept called deep

packet inspection and this is done using a new scripting language called Bro. Bro act as an IDS but the specialty of Bro is that it can be customized according to ones need thus making it more flexible implementing security besides being the fact that it can be used on any Linux system with minimum ease and very low cost. It provides the security analyst or the administrator to build custom policy scripts according to the need thus ensuring network security.

Intrusion prevention system (IPS) is another way of implementing security. An IPS makes use of rules that governs which packet to accept and which one to reject. IPS is not a substitute for a Firewall. Instead an IPS works along with the firewall. Firewall basically deploys rules for incoming and outgoing packets. IPS can further investigate the packets for Intrusions. In this work, development of IPS script is carried out using Bro which is able to detect malicious traffic.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Information Security

1.1.1 Complexity of Networks

The world has become interconnected in many different ways. Now organizations can easily spread across the globe, and can now have inter-office collaboration on a daily basis. But all of that interconnectedness relies on the ability to protect the networks that create those connections. Unfortunately, the last five years have seen hackers and criminals become increasingly effective at compromising these networks, as they have quickly developed new and even more malicious threats to network security. The more the threats arise the more innovative the security technology becomes and more innovative the security technology becomes, the more innovative becomes the intruders or hackers. These newly created threats have been so successful in creating an image in the mind of users that they always have a vague conception that these new threats exist even when very good security measures are in place and this thinking is right to certain extent because despite having very good security measures sometimes attacks occurs and the security personnel have no answer to them or even if they have, it comes after certain time and until then the intruder or hacker performs whatever the hacker wants to do thus making the information very insecure.

1.1.2 Security Action

Security is the state of well being of information and infrastructure in which the possibility of successful yet undetected theft, tamper with, and disruption of information and services is kept low. Nowadays as the intruders have become more innovative the organizations face many more threats from intruders than sometime back. Security Attack is an action that threatens the state of well being.

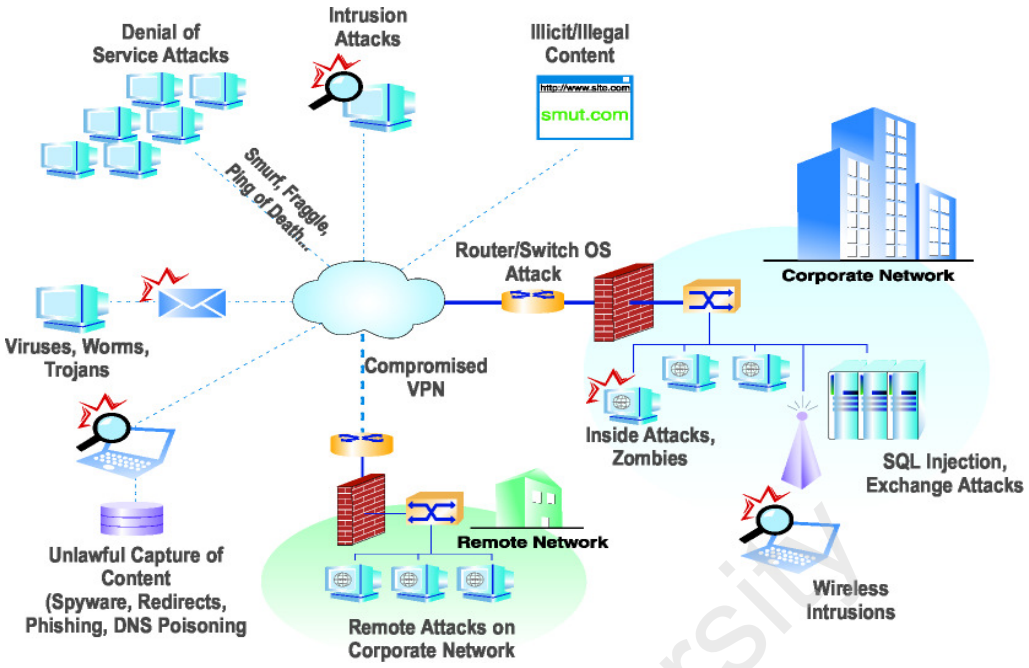

Figure 1.1 Prevalent threat vectors in today's networking environment[1]

There are many types of threats but the most prevalent are as follows:

1.1.2.1 Viruses and Worms

The term virus has long been used generically to describe any computer threat, but in actuality it refers specifically to malware that inserts malicious code into existing documents or programs, and then spreads itself by various means. The reason people often call every computer threat a "virus", is because viruses are the original type of malware, actually predating the public Internet. Viruses have become an accepted part of computing in the modern world. A risk of doing business in a networked environment. It's never nice finding any computer infected with a virus or worm, and yet powering up the virus scanner usually takes care of the problem. The virus problem is one that is increasing, and is not likely to go away in the near future. Understanding the common virus can help combat and defeat even the nastiest of viruses. Today, viruses are still by far the most common type of network security threat, and over 90 percent of viruses are spread through attachments on emails. Often the attacker will combine a virus with a "zombie" attack so that one will receive an email with an attachment from a friend that actually contains a virus[36].

1.1.2.2 Trojan Horses

Trojan Horses hide themselves in seemingly harmless programs, until a certain condition causes it to open. They do not however replicate themselves, so also cannot be considered to be true viruses. A Trojan horse is a malware attack that disguises itself as something innocent, such as a greeting card etc. A recent example of a devastating Trojan horse used an email with a link that supposedly connected the reader to a greeting card but actually it inserts the client code on the client side and then the server side applicant can easily use the client code to have a look into the information on the client side. To prevent Trojan Horses one method is to block users from downloading freeware, blocking links embedded in emails, and using a white list to create a list of approved websites that employees may visit. Because Trojans are much easier to prevent than they are to cure, with an infected computer sometimes requiring a complete reformatting of the hard drive, taking these drastic preventative measures may be warranted for some organizations[36].

1.1.2.3 Spam

Spamming is the use of any electronic communication medium to send unsolicited messages in bulk with many copies of the same message, in an attempt to force the message on people who would not otherwise choose to receive it. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. It costs the sender very little to send. Most of the costs are paid for by the recipient or the carriers rather than by the sender. As such, spamming can occur through email, instant messaging, mobile phone messaging, newsgroups etc. Much of the confusion surrounding it is derived from the lack of knowledge that everyday email users have regarding the methods in which others obtain their lists. List generation can generally be categorized into four methods: email harvesting, database sharing, single opt-in registration and double opt-in registration.

1.1.2.4 Spyware

Spyware, is not a virus and is generally not designed to damage computer. Spyware is broadly defined as any program that gets into the system without permission and hides in the background while it makes unwanted changes to user experience. The damage it does is more a by-product of its main mission, which is to serve the

targeted advertisements or make the browser display certain sites or search results. It can also be said that spyware is a program that is put in someone's computer to secretly gather information about the user and relay it to advertisers or other interested parties. Even though the name may indicate so, spyware is not an illegal type of software in any way. However there are certain issues that a privacy oriented user may object to and therefore prefer not to use the product. This usually involves the tracking and sending of data and statistics via a server installed on the user's PC and the use of internet connection in the background.

1.1.2.5 Phishing

It is the second most important type of spam. In it some link is provided to the user in the email and the user is asked to follow it to connect to certain bank or organization to verify their account but this link is the link of a spoofed website which is used just to collect the user information like username, passwords or account numbers and then the exploiters use this information to empty the users accounts or for some other uses which are beneficial to them. It is always recommended to open the particular website in a new browser window rather than following the link to avoid phishing attacks. Also the users can check for exact website through security certificates etc. These measures can be of very good help in preventing the users from being exploited.

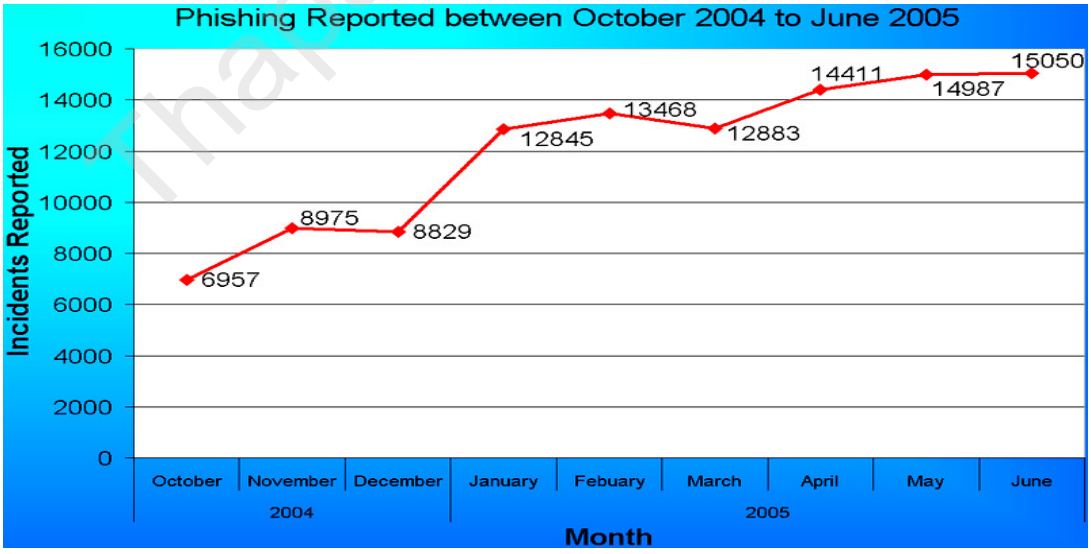

Figure 1.2 Phishing Trends[2]

As these threats arise on continuous basis there arise the need for network security and if complete network security is not implemented now then definitely the organizations are going to have a tough time facing these threats and a lots of money will be wasted just to encounter these threats but then also they won't feel free from security havocs. So, now is the time to have a look at Network Security and start finding some alternatives so that security can be implemented in such a way that it is implemented with minimum cost and also the level of security should be such that the organizations should feel free after implementing the security into their networks.

1.1.3 Network Security

Network Security can be stated as the adoption of the rigorous policies by the administrator so as to prevent the unauthorized access to network accessories by illegitimate users or better to say intruders or hackers. The level to which these security policies are being adopted depend on the administrator of the network. The past few years have seen a radical evolution in the nature and requirements of network security. There are many factors contributing to these changes, the most important of which are content-based threats such as Viruses, Worms, Trojans, Spyware and Phishing that can spread quickly and indiscriminately, and require sophisticated levels of intelligence to detect[1]. The technology required to prevent against these threats and this technology is "DPI" also called as "Deep Packet Inspection".

As technology improves so does its exploitation. Past few years have seen some good use of technology as well as its use for some illegal purposes. So as technology increased so does threats such as denial of service attacks, viruses etc. These threats are not only from the outside world but also from inside the network and these threats need to be taken care off in order to have smooth communication without the fear of anything. So in order to safeguard the networks from these threats it became imperative to have some good security policies in place and these policies forms the basis of Network Security which can be stated as safeguarding the sensitive information from unauthorized access. Till now Network Security is mostly implemented using Firewalls which are still preferred in most organizations. Firewalls work according to the rules which are defined by the administrator. Decisions on whether or not to accept the packets are based upon policies established

by the network administrator related to which senders are allowed to reach designated computing systems on their internal network, and which protocols they can use to exchange information. While this filtering is useful, it is not adequate to determine the difference between, say, a valid mail message or a message infected with a virus, because Stateful Inspection does not check the actual contents of the packets to distinguish malicious content from valid content[1]. It only checks the header part of the packet and not the data part and as one know that header part contains source and destination address based on the layer like whether it is a transport layer or network layer. In case of Transport layer there is source and destination port address where as in case of Network layer there is source and destination IP address. Now when the firewall encounters these addresses it checks its table and makes the decision whether to allow these addresses or block these addresses. If ever it permits the address to allow exchanging of data packets then may be it can happen that the permitted packet contains some unauthorized or illegal content which is also allowed to pass through the firewall thus jeopardizing the security. So today the level of threats has reached such a state that only using predefined rules will not suffice. Security can be better implemented if these rules can be made more flexible so that administrator can have more grasp on each and every bit that is passing through the network and this can be implemented by the upcoming technology called “Deep Packet Inspection”.

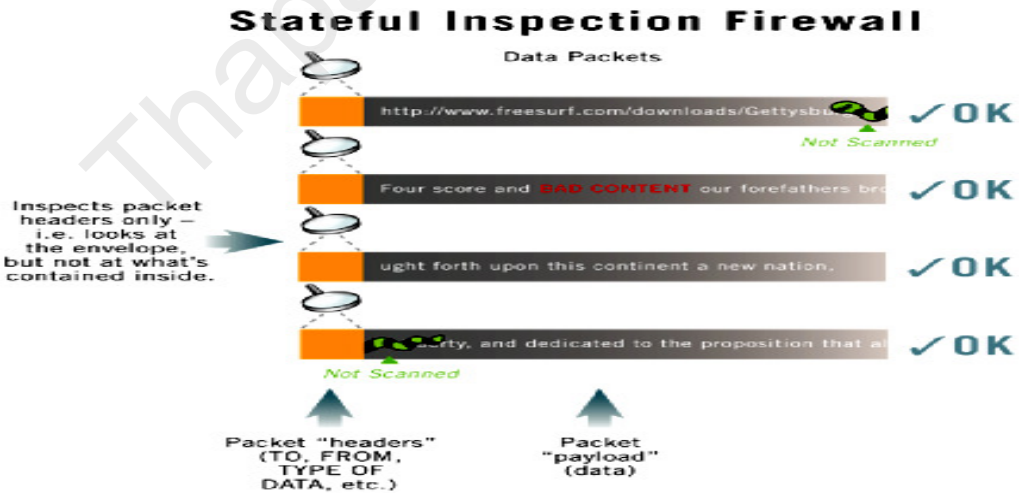

Figure 1.3 Representation of Stateful Inspection[1]

1.1.4 Deep Packet Inspection

The most prevalent problem today the security analysts are facing is not the security but the depth at which the security is implemented. *Deep Packet Inspection* is a technology that not only looks at the packet headers like other appliances like firewalls but at every bit of payload itself, often across multiple thousand of packets to detect threats and based on the content it directs the firewall whether to allow the packet to go through or not thus making security more powerful[1]. In other words, it can be said that "Deep Inspection" is a combination of Stateful packet inspection firewall with some IDS signatures and some application protocol anomaly detection rules[4]. DI firewalls use an Attack Object Database to store protocol anomalies and attack patterns (sometimes referred to as signatures), grouping them by protocol and security level (severity).

Deep packet inspection directs, persists, filters and logs IP-based applications and Web services traffic based on content encapsulated in a packet's header or payload, regardless of the protocol or application type. With deep packet inspection in place through a single intelligent network device, companies can boost performance without buying expensive servers or additional security products. Deep Packet Inspection can be seen as the integration of Intrusion Detection (IDS) and Intrusion Prevention (IPS) capabilities with traditional stateful firewall technology. Traditional networks have a defined boundary demarcated by a firewall with an IDS sensor sitting behind it. The engine that drives deep packet inspection typically includes a combination of signature-matching technology along with heuristic analysis of the data in order to determine the impact of that communication stream but deep packet inspection is bit difficult to implement. Deep Packet Inspection provides robustness and dynamic protection to the network. The inspection engine must use a combination of signature-based analysis techniques as well as statistical, or anomaly analysis, techniques. Both of these are borrowed directly from intrusion detection technologies. In order to identify traffic at the speeds necessary to provide sufficient performance newer ASICs will have to be incorporated into existing firewall designs. These ASICs, or Network Processors Units (NPUs), provide for fast discrimination of content within packets while also allowing for data classification[5]. Deep Packet Inspection essentially collapses Intrusion Detection (IDS) functionality into the

firewall appliance so that both a firewall and an in-line IDS are implemented on the same device. Deep Packet Inspection is sometimes called as Complete packet inspection because in it both the header and data part of the packet are completely checked. This is in contrast to Shallow packet Inspection in which only the header part of the packet is checked. Thus one can say that Deep Packet Inspection offers a more deep and rigorous inspection of the data and having good analysis can help one find the anomalous or illegal data easily. Analysis of packet headers can be done economically since the locations of packet header fields are restricted by protocol standards. However, the payload contents are, for the most part, unconstrained. Therefore, searching through the payload for multiple string patterns within the data stream is a computationally expensive task.

One of the most significant aspects of DPI is that it is a service-based technology. Unless the security appliance knows what threat signatures or anomalies it is looking for, it is helpless. IPS provides the security appliance with a frequently updated library of threat signatures, heuristic instructions etc., in order to insure it is protecting the network from current threats thus making the security appliance dynamic[1].

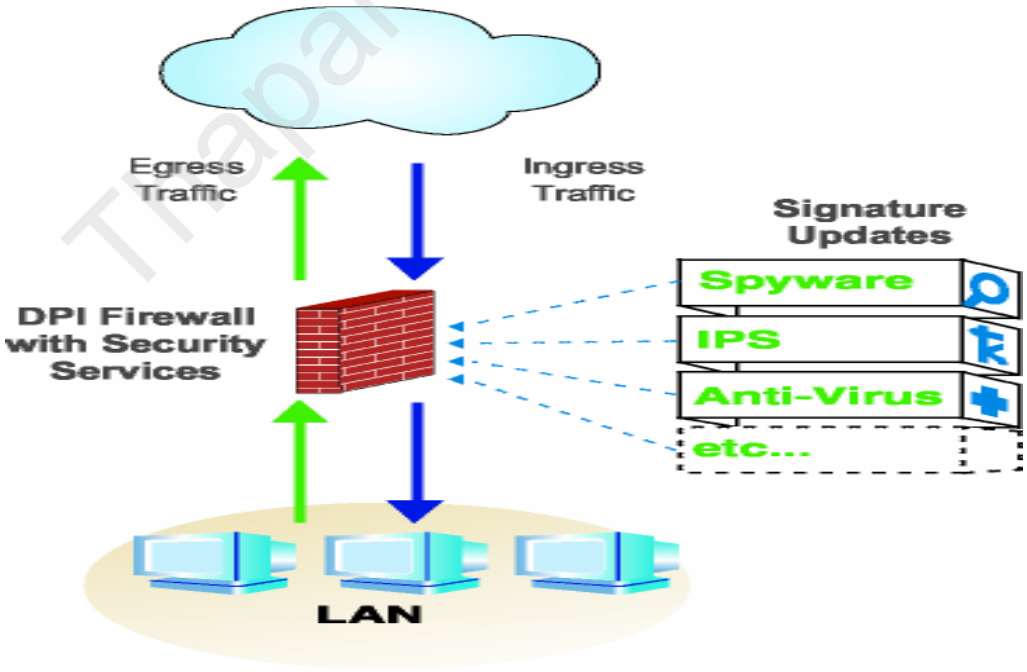

Figure 1.4 The security appliance is now a dynamic system that requires regular signature updates[1]

Chapter 2

Literature Survey

Within the last five years, businesses have begun to need to share data across wide areas. This has prompted efforts to convert principally LAN-based protocols into WAN-friendly protocols. The result has spawned an entire industry of consultants who know how to manipulate routers, gateways and networks to force principally broadcast protocols across point-to-point links (two very different methods of transmitting packets across networks). Recently (within the last 2 or 3 years) more and more companies have realized that they need to settle on a common networking protocol. Frequently the protocol of choice has been TCP/IP, which is also the primary protocol run on the Internet. The emergence of TCP/IP allows companies to interconnect with each other via private networks as well as through public networks. While reality is rapidly approaching this utopian picture, several relatively minor issues have changed status from low priority to extreme importance. Security is probably the most well known of these problems. When businesses send private information across the net, they place a high value on it getting to its destination intact and without being intercepted by someone other than the intended recipient. Individuals sending private communications obviously desire secure communications. Finally, connecting a system to a network can open the system itself up to attacks. If a system is compromised, the risk of data loss is high.

It can be useful to break network security into two general classes:

- methods used to secure data as it transits a network
- methods which regulate what packets may transit the network

While both significantly effect the traffic going to and from a site, their objectives are quite different[35].

There are various types of tools and techniques for ensuring Network Security and they are discussed as follows:

2.1 Using Firewall for Network Security

A firewall is a system or group of systems that enforces an access control policy between two or more networks. The actual means by which this is accomplished varies widely, but in principle, the firewall can be thought of as a pair of mechanisms: one which exists to block traffic, and the other which exists to permit traffic. Some firewalls place a greater emphasis on blocking traffic, while others emphasize permitting traffic. Probably the most important thing to recognize about a firewall is that it implements an access control policy. It's also important to recognize the firewall's configuration, because it is a mechanism for enforcing policy, imposes its policy on everything behind it. Administrators for firewalls managing the connectivity for a large number of hosts therefore have a heavy responsibility [27]. .

2.1.1 Need of Firewall

The Internet, like any other society, is plagued with the kind of jerks who enjoy the electronic equivalent of writing on other people's walls with spray paint, tearing their mailboxes off, or just sitting in the street blowing their car horns. Some people try to get real work done over the Internet, and others have sensitive or proprietary data they must protect. Usually, a firewall's purpose is to protect a private network from the security threats that may come form the other external public networks. Generally firewalls are combination of hardware and software. Many traditional-style corporations and data centers have computing security policies and practices that must be followed. In a case where a company's policies dictate how data must be protected, a firewall is very important, since it is the embodiment of the corporate policy. Frequently, the hardest part of hooking to the Internet, if it is a large company, is not justifying the expense or effort, but convincing management that it's safe to do so. A firewall provides not only real security--it often plays an important role as a security blanket for management. Firewall is used to protect private internal network from offensive Web sites and potential hackers.

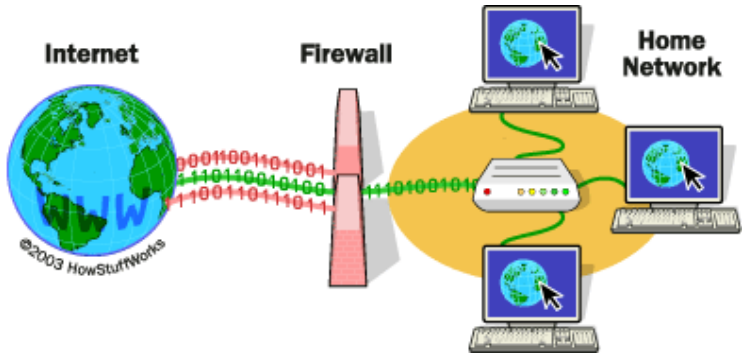

Figure 2.1 Firewall [27]

Basically, a firewall is a barrier to keep destructive forces away from the property. In fact, that's why it is called a firewall. Its job is similar to a physical firewall that keeps a fire from spreading from one area to the next. Some firewalls permit only email traffic through them, thereby protecting the network against any attacks other than attacks against the email service. Other firewalls provide less strict protections, and block services that are known to be problems. Generally, firewalls are configured to protect against unauthenticated interactive logins from the external world. More elaborate firewalls block traffic from the outside to the inside, but permit users on the inside to communicate freely with the outside. The firewall can protect a private network against most of the network-borne attack if you unplug it [27].

2.1.2 Basic types of Firewall

Conceptually, there are three types of firewalls:

1. Network layer
2. Application layer
3. Hybrid

It depends on network administrators which type of firewall they want to use. It depends on what mechanisms the firewall uses to pass traffic from one security zone to another. The International Standards Organization (ISO) Open Systems Interconnect (OSI) model for networking defines seven layers, where each layer provides services that higher-level layers depend on. In order from the bottom, these layers are physical, data link, network, transport, session, presentation, application. The important thing to recognize is that the lower-level the forwarding mechanism, the less examination the firewall can perform. Generally speaking, lower-level

firewalls are faster, but are easier to fool into doing the wrong thing. These days, most firewalls fall into the hybrid category, which do network filtering as well as some amount of application inspection. The amount changes depending on the vendor, product, protocol and version, so some level of digging and/or testing is often necessary[23].

- **Network Layer Firewall**

These generally make their decisions based on the source, destination addresses and ports in individual IP packets. A simple router is the traditional network layer firewall, since it is not able to make particularly sophisticated decisions about what a packet is actually talking to or where it actually came from. Modern network layer firewalls have become increasingly sophisticated, and now maintain internal information about the state of connections passing through them, the contents of some of the data streams, and so on. One thing that's an important distinction about many network layer firewalls is that they route traffic directly though them, so to use them, one either need to have a validly assigned IP address block or to use a private internet address block. Network layer firewalls tend to be very fast and tend to be very transparent to users [27].

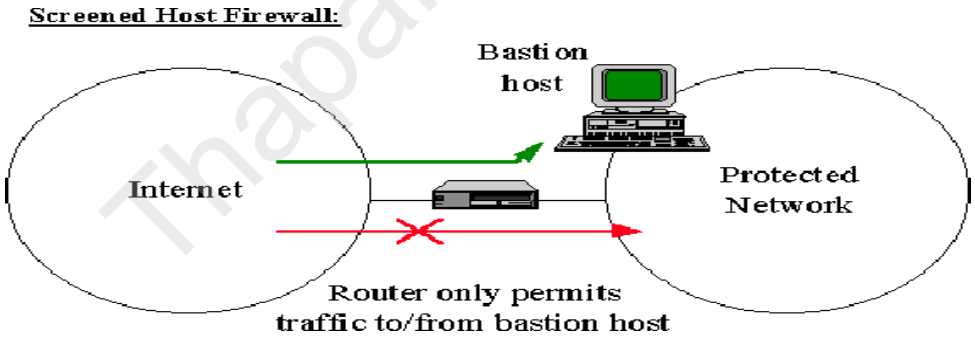

Figure 2.2 Screened Host Firewall[27]

In the above Figure 2.2, a network layer firewall called a Screened Host Firewall is represented. In a screened host firewall, access to and from a single host is controlled by means of a router operating at a network layer. The single host is a bastion host; a highly-defended and secured strong-point that can resist attack.

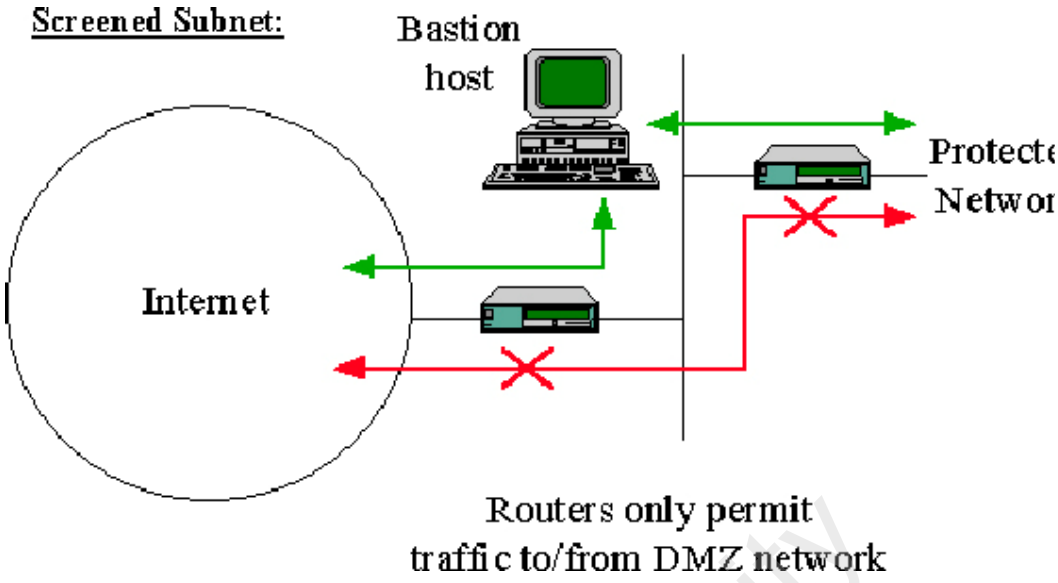

Figure 2.3 Screened Subnet Firewall [27]

In Figure 2.3, a network layer firewall called a screened subnet firewall is represented. In a screened subnet firewall, access to and from a whole network is controlled by means of a router operating at a network layer [27]. It is similar to a screened host, except that it is, effectively, a network of screened hosts.

- **Application Layer Firewalls**

These generally are hosts running proxy servers, which permit no traffic directly between networks, and which perform elaborate logging and auditing of traffic passing through them. Since the proxy applications are software components running on the firewall, it is a good place to do lots of logging and access control. Application layer firewalls can be used as network address translators, since traffic goes in one side and out the other, after having passed through an application that effectively masks the origin of the initiating connection. Having an application in the way in some cases may impact performance and may make the firewall less transparent.

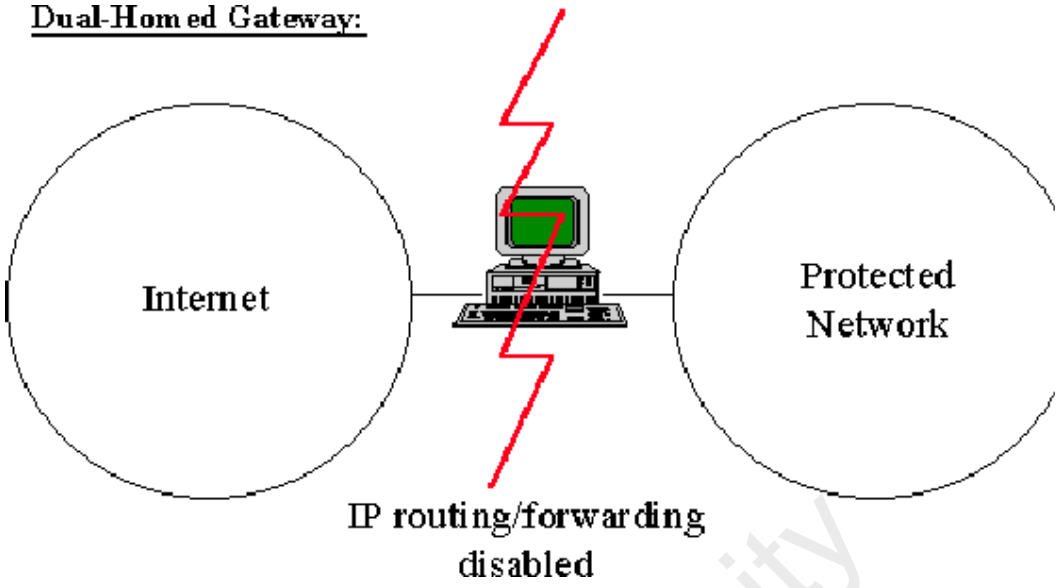

Figure 2.4 Dual Homed Gateway[27]

Early application layer firewalls are not particularly transparent to end users and may require some training. Modern application layer firewalls are often fully transparent. Application layer firewalls tend to provide more detailed audit reports and tend to enforce more conservative security models than network layer firewalls. In Figure 2.4 an application layer firewall called a dual homed gateway is represented. A dual homed gateway is a highly secured host that runs proxy software. It has two network interfaces, one on each network, and blocks all traffic passing through it.

- **Hybrid Firewall**

In Hybrid combination of both network layer firewall and application layer firewall is used. These hybrid firewalls now lie some place between network layer firewalls and application layer firewalls. As expected, network layer firewalls have become increasingly aware of the information going through them, and application layer firewalls have become increasingly low level and transparent. The end result is that now there are fast packet-screening systems that log and audit data as they pass through the system. Increasingly, firewalls (network and application layer) incorporate encryption so that they may protect traffic passing between them over the Internet.

- **Shortcomings of Firewalls**

- Firewalls cannot enforce strong access policies with router access lists. Users can easily install backdoors to their systems to get over no incoming telnet or no X11 rules. Also crackers install telnet backdoors on systems where they break in.

- It is not sure that what services are listening for connections on high port numbers.

- Checking the source port on incoming FTP data connections is a weak security method. It also breaks access to some FTP sites. It makes use of the service more difficult for users without preventing bad guys from scanning target hosts.

2.2 IDS (Intrusion Detection System)

Firewall helps a lot in ensuring Network Security but they have certain limitations as discussed above. Those tools and techniques which generates alerts only after something malicious activity has happened, comes under the category of reactive tool and techniques category. One of the common reactive approaches for network security is called IDS. **An intrusion detection system** (**IDS**) generally detects unwanted manipulations of computer systems. An intrusion detection system is used to detect several types of malicious behaviors that can compromise the security and trust of a computer system. An intrusion detection system is used to detect several types of malicious behaviors that can compromise the security and trust of a computer system. This includes network attacks against vulnerable services, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins etc. An intrusion detection system (IDS) monitors network traffic and monitors for suspicious activity and alerts the system or network administrator. There are many types of IDS and each one of them approaches the goal of detecting suspicious traffic in different ways. There are network based (NIDS) and host based (HIDS) intrusion detection systems [3]. With the number of intrusion and hacking incidents around the world on the rise, the importance of having dependable intrusion detection systems in place is greater than ever. IDS just detect intrusion and record those intrusions or whatever changes are made by the intruder and then sends these information to reactive system like IPS.

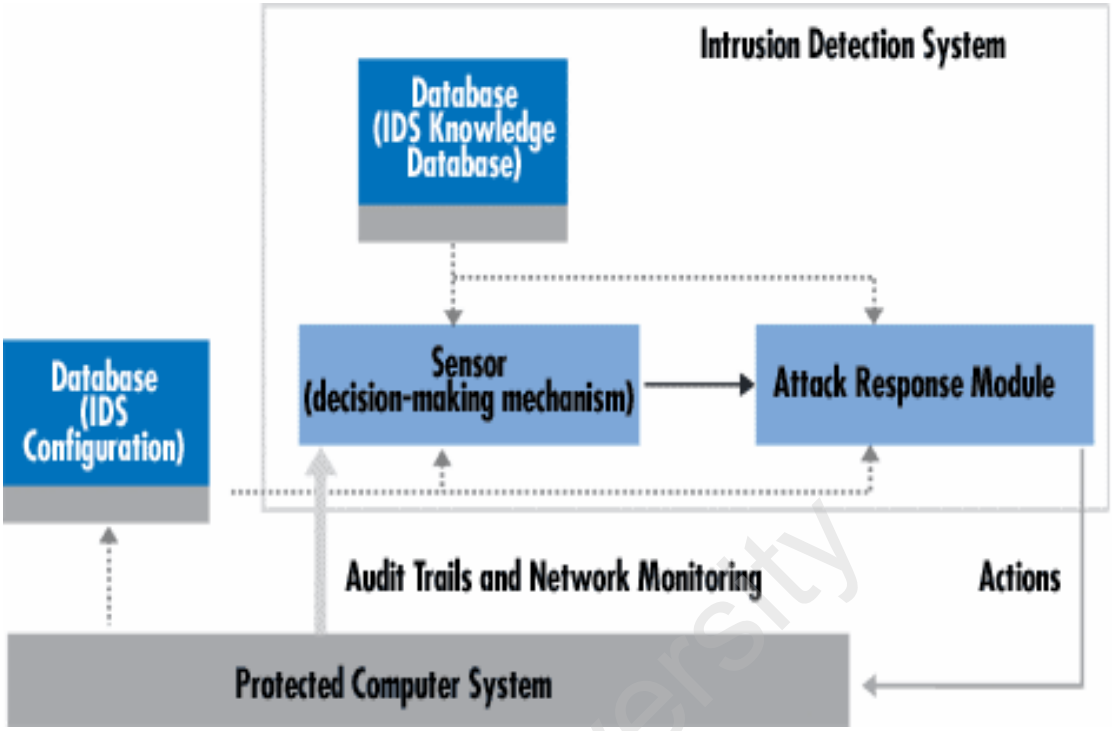

Figure 2.5 A sample IDS. The arrow width is proportional to the amount of information flowing between system components [9]

Here in Figure 2.5, Sensor is the most important component because it checks for intrusions. It receives data from Audit Trails and IDS knowledge database apart from logs etc. This information is used for making further decisions. The sensor gets the data from event generator which captures data and when data is captured it generate events corresponding to the information found in traffic. These events are then passed on to sensor which keeps the relevant data with it and discard irrelevant data and based on this relevant data it detects anomalies and takes decisions.

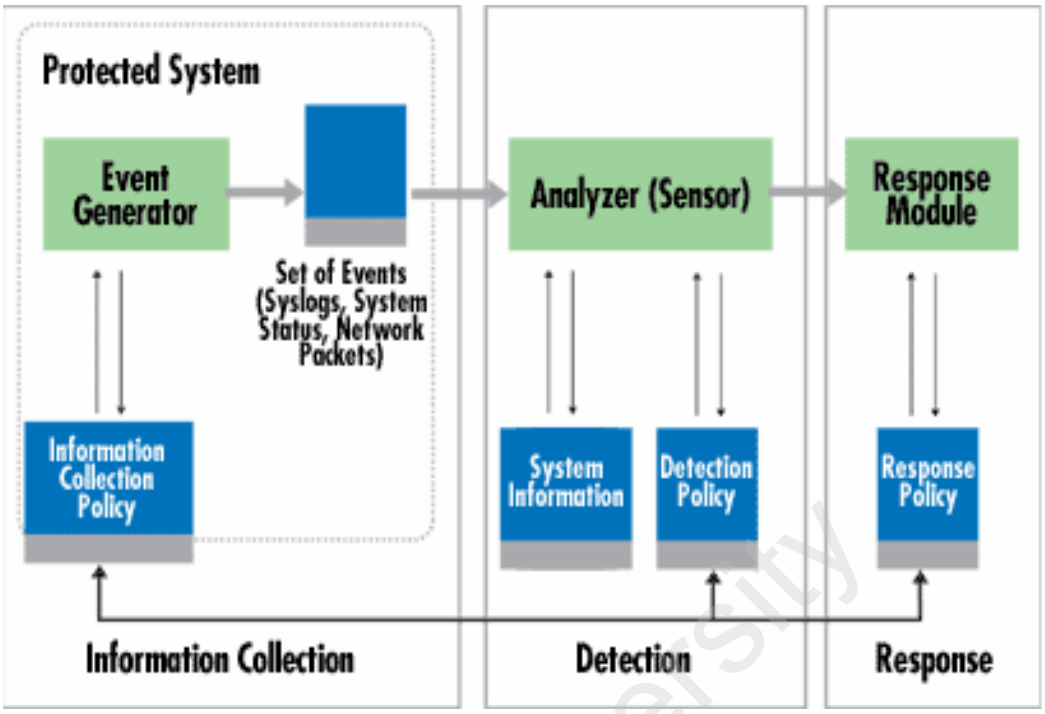

Fig 2.6 IDS components [10]

Intrusion detection systems can be arranged as either centralized (for example, physically integrated within a firewall) or distributed. A distributed IDS consists of multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other. More sophisticated systems follow an agent structure principle where small autonomous modules are organized on a per-host basis across the protected network [11]. The role of the agent is to monitor and filter all activities within the protected area and depending on the approach adopted make an initial analysis and even undertake a response action. The cooperative agent network that reports to the central analysis server is one of the most important components of intrusion detection systems. DIDS can employ more sophisticated analysis tools, particularly connected with the detection of distributed attacks [12]. Another separate role of the agent is associated with its mobility and roaming across multiple physical locations. In addition, agents can be specifically devoted to detect certain known attack signatures. This is a decisive factor when introducing protection means associated with new types of attacks [13]. IDS agent-based solutions also use less sophisticated mechanisms for response policy updating [14].

2.2.1 Types of IDS

There are various types of IDS. They are discussed as follows:

2.2.1.1 NIDS (Network Intrusion Detection System)

In NIDS the IDS is placed at main points of the network and thus it analyzes whatever traffic flows through the network. Network Intrusion Detection Systems gain access to network traffic by connecting to a hub, network switch. It watches live network packets and looks for signs of computer crime, network attacks, network misuse and anomalies. The NIDS does this by reading all the incoming packets and trying to find suspicious patterns. NIDS not only analyzes the incoming traffic but also the outgoing traffic as well because most of the attacks are done from inside the network as well. IDS tools are capable of distinguishing between insider attacks originating from inside the organization (coming from own employees or customers) and external ones (attacks and the thread posed by hackers). Because they are responsible for monitoring a network, rather than a single host, Network-based intrusion detection systems (NIDS) tend to be more distributed than host-based IDS. Software, or appliance hardware in some cases, resides in one or more systems connected to a network, and is used to analyze data such as network packets.

2.2.1.2 HIDS

Host based Intrusion Detection System works similarly as NIDS but here rather than sitting on the choke point or main point of the network HIDS sits on each system and monitors the traffic that is going in and out of that particular system and not of entire network. The difference between host-based and network-based intrusion detection is that NIDS deals with data transmitted from host to host while HIDS is concerned with what occurs on the hosts themselves. Host-based intrusion detection is best suited to combat internal threats because of its ability to monitor and respond to specific user actions and file accesses on the host. The majority of computer threats come from within organizations, from many different sources [6]. A HIDS will usually make its best efforts to prevent the object-database, checksum-database and its reports from any form of tampering because if intruders succeed in modifying any of the objects the HIDS monitors, nothing can stop such intruders from modifying the HIDS itself unless security administrators take appropriate precautions. A host system looks at system logs for evidence of malicious or suspicious application activity in real time. It also monitors key system files for evidence of tampering. Careful consideration is

required in this area to ensure that performance is not degraded. The main thing here is that whether it is NIDS or HIDS both look for specific patterns in the traffic that indicate malicious content. When an IDS looks for these patterns in network traffic, it's network-based. When an IDS looks for attack signatures in log files, it's host-based. Both NIDS and HIDS have some weaknesses also if they have strengths but a truly secure network is one which employ both NIDS and HIDS.

NIDS vs HIDS

There are some points on which one can ponder in case of both NIDS and HIDS and based on that one can decide whether NIDS is needed or HIDS or both.

Advantages of NIDS are:

- **Cost :**

In case of NIDS, as it is known that intrusion detection is done at main points of the network , so security appliances need to be deployed at these main points only and not on every system so less cost is needed to deploy them than HIDS because in HIDS one need to deploy security appliances on each and every host.

- **Difficult to remove evidence:**

In NIDS, as the traffic is captured from network and not by the particular host, so it becomes difficult for the hacker to remove his tracks from NIDS.

- **Can also use headers:**

NIDS also works by opening the headers which may contain some malicious information but HIDS don't work using the headers so they cannot detect these type of attacks. DoS is an attack which can only be identified by inspecting the headers but if we use only HIDS then this attack cannot be detected.

- **Real time detection:**

NIDS detect the attacks in real time before they can do some damage but HIDS can only take detect after some damage has been done and log entry has been written. So NIDS provide better option.

- **Platform Independence:**

NIDS works independently of platform whereas HIDS are very platform dependent and particular HIDS works only on particular platform. Thus, NIDS offer more flexibility than HIDS.

Advantages of HIDS are:

- **Less Cost:**

HIDS offers very less cost of implementation on particular host than NIDS because each NIDS is very costly than HIDS.

- **Real Time Detection:**

Even though HIDS cannot perform detection in real time as fast as NIDS but then also they can detect intrusions in real time and can make alerts based on that.

- **Detect certain attacks:**

HIDS detect some attacks which cannot be seen by the NIDS.

- **Verifies success :**

NIDS can perform real time detection and can alert the administrator about the attack based on which certain action can be taken but whether the action taken has been success or not can be verified from the HIDS which monitors the logs and based on that it can determine whether the attack occurred or it is prevented.

- **Detect on greater level:**

HIDS can even detect when the intruder logged in and logged off and what was performed but NIDS cannot perform detection to such a level.

NIDS or HIDS

As discussed above both NIDS and HIDS offers certain advantages that the other does not offer. So the answer is HIDS for a complete solution and NIDS for a LAN solution complimenting HIDS. Similarly when an installation of antivirus software is done not only is the software installed on main servers, but it is also installed on all clients as well. There is no reason why both NIDS and HIDS can not be used in conjunction as a strong IDS complimentary strategy. It is perceived that NIDS are easier to disable from an intruders perspective and everyone tend to agree with this notion [14]. So in order to have good security in place both NIDS and HIDS should be used.

2.2.1.3 Protocol Based Intrusion Detection System

A protocol based intrusion detection system consists of a system or agent that would typically sit at the front of a server, monitoring and analyzing the communication protocol between a connected device (a user/PC or system). For a web server this would typically monitor the HTTPS protocol stream and understand the HTTP protocol relative to the web server/system it is trying to protect whereas HTTPS is in use then this system would need to reside in the "shim" or interface between where HTTPS is un-encrypted and immediately prior to it entering the Web presentation layer[15]. In protocol Analysis each packet is wrapped in predefined layers of different protocols. IDS authors, recognizing this, implemented engines that unwrap and inspect these layers, according to the protocol standards or RFC. Each wrapper has several fields with expected or normal values. Anything that violates or is outside of these standards is likely malicious. The IDS inspects each field of the different protocols of an incoming packet: IP, TCP, and UDP. If something violates a protocol rule, for instance, if it contains an unexpected value, an alert is generated[16].

2.2.1.4 Signature Based

A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. This is similar to the way most antivirus software detects malware. The issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to your IDS. During that lag time IDS would be unable to detect the new threat [17] Most intrusion detection systems (IDS) are what is known as signature-based. This means that they operate in much the same way as a virus scanner, by searching for a known identity or signature for each specific intrusion event. And, while signature-based IDS is very efficient at sniffing out known attacks, it does, like anti-virus software, depend on receiving regular signature updates, to keep in touch with variations in hacker technique. In other words, signature-based IDS is only as good as its database of stored signatures [18].

2.2.1.5 Application protocol based Intrusion Detection System

An Application protocol based intrusion detection system consists of a system or agent that would typically sit within a group of servers, monitoring and analyzing the communication on application specific protocols. For example in web server with the

database this would monitor the SQL protocol specific to the middleware/business login as it transacts with the database[19].

2.2.1.6 Hybrid Intrusion Detection System

A Hybrid Intrusion detection system combines two or more approaches. Host agent data is combined with the network information to form a comprehensive view of the network.

2.3 IPS (Intrusion Prevention System)

IPS is any device (hardware or software) that has the ability to detect attacks, both known and unknown, and prevent the attack from being successful. When the blocking capabilities of a firewall are combined with the deep packet inspection of an IDS, intrusion prevention systems or IPS is obtained .

2.3.1 Types of IPS

The Intrusion Prevention Systems could be of the following types and their combinations could also be used differing from organization to organization. Some of those are as follows:

2.3.1.1 Host Based Intrusion Prevention System

In this IPS is installed on a specific host or on a single computer. These are also known as HIPS. As with host IDS systems, the host IPS relies on agents installed directly on the system being protected. It binds closely with the operating system kernel and services, monitoring and interpreting system calls to kernel or APIs in order to prevent attacks as well as log them. It may also monitor data streams and the environment specific to a particular application (file locations and registry settings for a web server for example) in order to protect that application from generic attacks for which no signature yet exists. One potential disadvantage with this approach is that given the necessary tight interaction with the host operating system, future operating system upgrades could cause problems. Since a host IPS agent intercepts all requests to the system it protects, it has certain prerequisites –it must be very reliable, must not negatively impact performance and must not block legitimate traffic. Any HIPS that does not meet these minimum requirements should never be installed in a host, no matter how effectively it blocks attacks [20]

2.3.1.2 Network based Intrusion Prevention System

A Network based IPS is one where the application/hardware and any actions taken to prevent an intrusion on a specific network host(s) is done from a host with another IP address on the network (This could be on a front end firewall appliance). Network Intrusion Prevention Systems (NIPS) are purposely built hardware/software on their combination that are designed to analyze, detect and report on security related events and even to take proper actions. NIPS are designed to inspect traffic and based on their configuration or their security policy they can drop malicious traffic. The network IPS combines features of standard IDS, an IPS and a firewall and is sometimes known as an In Line IDS or Gateway IDS(GIDS)[21].The next generation firewall, the deep packet inspection firewall, also exhibits a similar feature set thought the researchers do not believe that the deep packet inspection is ready for mainstream deployment yet. As with typical firewall, the NIPS has at least two network interfaces, one designated as internal and one as external. As packets appear at the either interface, they are passed to the detection engine at which points the IPS device functions much as any IDS would in determining whether or not the packet being examined poses a threat. However if it should detect a malicious packet in addition to raising an alert it will discard the packet and mark that flow as bad [22].

2.3.1.3 Content Based Intrusion Prevention System

A Content based IPS (CBIPS) inspects the content of network packets for unique sequences, called signatures, to detect and hopefully prevent known types of attacks as worm infections and hacks.

2.3.1.4 Protocol Based Intrusion Prevention System

A key development in IDS/IPS technologies is the use of protocol analyzers. Protocol analyzers can natively decode application layer network protocols like HTTP, FTP. Once the protocols are fully decoded the IPS analysis engine can evaluate different parts of the protocol for anomalous behavior or exploits. For example, the existence of a large binary file in the User Agent field of an HTTP request would be very unusual and likely an intrusion. A protocol analyzer could detect this anomalous behavior and instructs IPS engine to drop the offending packets. Not all IPS/IDS engines are full protocol analyzers. Some products rely on simple pattern recognition techniques to look for known patterns. While this can be sufficient in many cases, it

creates an overall weakness in the detection capabilities. Since many vulnerabilities have dozens or even hundreds of exploit variants, pattern recognition based IPS/IDS engines can be evaded. For example some pattern recognition engines require hundreds of different signatures to protect against a single vulnerability. This is because they must have a different pattern for each exploit variant. Protocol analysis based products can often block exploits with a single signature that monitors for the specific vulnerability in the network communication [19, 23].

2.3.1.5 Rate based Intrusion Prevention System

Rate based IPS (RBIPS) are primarily intended to prevent denial of service and distributed denial of service attacks. They work by monitoring and learning normal network behaviors. Through real time traffic monitoring and comparison with stored statistics, RBIPS can identify abnormal rates for certain types of traffic eg. TCP, UDP or ARP packets, connections per second, packets per connection, packets to specific ports etc. Attacks are detected when threshold are exceeded. The thresholds are dynamically adjusted based on the time of day, day of week etc, drawing on stored traffic statistics. Unusual but legitimate network traffic patterns may create false alarms. The system's effectiveness is related to granularity of the RBIPS rule base and the quality of stored statistics. Once an attack is detected, various prevention techniques may be used such as rate limiting specific attack related traffic types, source or connection tracking and source-address, port or protocol filtering (black listing) or validation (white testing)[23].

Host Based vs Network Based Intrusion Prevention System

HIPS can handle encrypted and unencrypted traffic equally, because it can analyze the data after it has been decrypted on the host. NIPS does not use processor and memory on computer hosts but uses its own CPU and memory. NIPS is a single point of failure, which is considered a disadvantage, however this property also makes it simpler to maintain. A Bypass Switch can be implemented to alleviate the single point of failure disadvantage though. This also allows the NIPS appliance to move and take offline for maintenance when needed. NIPS can detect events scattered over the network and can react whereas with a HIPS only the hosts data itself is available to take a decision, respectively it would take too much time to report it to a central decision making engine and report back to block[23].

2.4 Deep Packet Inspection (DPI)

DPI is the foremost technology for identifying and authenticating protocols and applications conveyed by IP. The standard packet inspection process extracts basic protocol information such as IP addresses (source, destination) and the other low-level connection states. This information typically resides in the packet header itself and consequently reveals the principal communication intent. The inspection level in the shallow inspection process is insufficient to reach any application-related deductions. DPI, on the other hand, provides application awareness. This is achieved by analyzing the content in both the packet header and the payload over a series of packet transactions [24]. Deep packet inspection has recently gained popularity as it provides the capability to accurately classify and control traffic in terms of content, applications, and individual subscribers [8]. It is becoming increasingly important to verify traffic based on content and not only based on the header.

Network threats have evolved from relatively simple, connection-based attacks to more complex content-based attacks such as viruses, worms, and Trojans. At the same time, organizations are struggling to cope with other content-based threats, such as email spam and inappropriate web content that reduce productivity and expose them to substantial liability. These new content-based attacks are not detected or stopped by the Stateful Inspection firewalls that have been deployed by many companies, causing a search for newer, more effective technologies. Recently, many firewall vendors have been touting the benefits of a technology called Deep Packet Inspection, promising better protection against content-based threats. While Deep Packet Inspection is more effective than Stateful Inspection for certain types of attacks, it falls far short of being a complete solution for protecting network and computing systems. Specifically, Deep Packet Inspection cannot detect a substantial portion of active viruses, Trojans and worms, and is completely ineffective for dealing with inappropriate Web content and email spam [34].

To be more specific DPI has following three limitations:

- While effective against buffer overflow attacks, denial of service attacks and certain types of malware, DPI can also be exploited to facilitate attacks in those same categories.
- DPI adds to the complexity and unwieldy nature of existing firewalls and other security-related software. DPI requires its own periodic updates and revisions to remain optimally effective.

- DPI can reduce computer speed because it increases the burden on the processor. Despite these limitations, many network administrators have embraced DPI technology in an attempt to cope with a perceived increase in the complexity and widespread nature of Internet-related perils.

These limitations encountered in DPI can be removed using Bro language which is very powerful and helps a great deal in ensuring security.

2.5 BRO

Bro is an open-source, Unix-based Network Intrusion Detection System (NIDS) that passively monitors network traffic and looks for suspicious activity. Bro detects intrusions by first parsing network traffic to extract is application-level semantics and then executing event-oriented analyzers that compare the activity with patterns deemed troublesome. Its analysis includes detection of specific attacks (including those defined by signatures, but also those defined in terms of events) and unusual activities (e.g., certain hosts connecting to certain services, or patterns of failed connection attempts)[25]. If BRO encounters something unusual it alerts the administrator or it logs whatever it see as unusual apart from executing some operating system command. Bro is intended for use by sites requiring flexible, highly customizable intrusion detection. It is important to understand that Bro has been developed primarily as a research platform for intrusion detection and traffic analysis. It is not intended for someone seeking an "out of the box" solution. Bro is designed for use by Unix experts who place a premium on the ability to extend an intrusion detection system with new functionality as needed, which can greatly aid with tracking evolving attacker techniques as well as inevitable changes to a site's environment and security policy requirements [25].

The Bro language is strongly typed and includes a bunch of types designed to aid analyzing network traffic. It also supports implicit typing, meaning that often it don't need to explicitly indicate a variable's type because Bro can figure it out from context. This feature makes the strong typing a bit less of a pain, while retaining its bug-finding benefits. For high performance, Bro relies on use of an efficient packet filter to capture only a (hopefully small) subset of the traffic that transits the link it monitors. Related to this, Bro comes with a set of analyzers, that is, scripts for analyzing different protocols and different types of activity. In general you can pick and choose among these for which types of analysis you want to enable, and Bro will

only capture traffic relating to the analyzers you choose. Thus, one can control how much work Bro has to do by the analyzers designated to it, a potentially major consideration if the monitored link has a high volume of traffic [26].

2.5.1 Working of Bro

First of all BRO captures the traffic and then it discards the unwanted traffic and keep only the important data with it and then based on the data in the traffic it generates events which are handled by the event handlers. The resulting filtered packet stream is then handed up to the next layer, the Bro ``event engine.'' This layer first performs several integrity checks to assure that the packet headers are well-formed, including verifying the IP header checksum. If these checks fail, then Bro generates an event indicating the problem and discards the packet. It is also at this point that Bro reassembles IP fragments so it can then analyze complete IP datagrams. If the checks succeed, then the event engine looks up the connection state associated with the tuple of the two IP addresses and the two TCP or UDP port numbers, creating new state if none already exists. It then dispatches the packet to a handler for the corresponding connection. Bro maintains a `tcpdump` trace file associated with the traffic it sees. The connection handler indicates upon return whether the engine should record the entire packet to the trace file, just its header, or nothing at all. This triage trades off the completeness of the traffic trace versus its size and time spent generating the trace. Generally, Bro records full packets if it analyzed the entire packet; just the header if it only analyzed the packet for SYN/FIN/RST computations; and skips recording the packet if it did not do any processing on it. After processing all the packets the event engine checks whether some events got generated from the traffic or not and if the events are generated then they are kept in queue and processed one after the other by applying policies on them and in this was intrusions are detected.

2.5.2 Detection using BRO

The most important question that arise is that how BRO detects intrusions and this is made very simple by BRO. Actually what BRO does is that it matches the traffic pattern with the already stored patterns in the database and from there it can make a guess about what is wrong and what is right. Now the other important thing is that how BRO analyzes traffic and this is done as first Bro filters the traffic, discarding elements of minimal important to its analysis. The remaining information is sent to its

"event" engine, where Bro interprets the structure of the network packets and abstracts them into higher-level events describing the activity. Finally, Bro executes policy scripts against the stream of events, looking for activity that the rules indicate should generate alerts or actions, such as possible intrusions. The term mentioned as event is nothing but action that take place in a network like connection established or connection terminated etc and the term policy script is just a program that is written in BRO language and is just used to make rules of our own. The most important thing here to note is that BRO cannot take action of its own, it must be guided to take action like what to do when some anomalous or confusing pattern is found. For example, Bro can detect a scan and send the attacking IP address to an external program that can, in turn, insert an access control block into a router, thus stopping the attacking IP from further scanning.

Intrusions are devised and executed by intelligent people who are often actively trying to avoid detection. A skilled attacker might be using little known techniques that are very difficult to detect. However, experience has shown that most intrusions are attempted by intruders who use standard, well-know techniques that they have learned from others. Usually Bro can detect these. Even expert attackers are susceptible to eventual mistakes leading to tell-tell signatures that Bro will detect. In addition, rather than relying solely on unique signature identification, Bro's more advanced features can often discern network anomalies that are caused by hostile activity. Even if previously-unknown attack techniques are use, they can often be detected by observing that the corresponding network activity that violates the rules of expected traffic[25].

2.5.3 BRO vs Snort

Snort is solely signature based, meaning that it looks for very specific content in the network stream and reports each instance of a particular signature. Bro can analyze network traffic at a much higher-level of abstraction, and has powerful facilities for storing information about past activity and incorporating it into analyses of new activity. Bro also provides a signature mechanism similar to Snort's [25]. BRO uses regular expressions as patterns to be matched with the traffic whereas Snort does not use regular expressions but instead use fixed string. On the basis of this one can be of the opinion that BRO offers much more flexibility than Snort.

Chapter 3

Problem Formulation

As per literature review carried out everything needed to ensure security was based on the information contained in header part of each packet but this was not sufficient to ensure complete protection because in order to have complete protection from intruders, one not only needs to know about header information but also about **Content Information**. In network security, the technology to perform network policy decisions based on the traffic content is referred to as Deep Packet Inspection(DPI)(in some flavors known as layer 7 firewalling).

Administrator of an organization see some sites as offensive or bad, so the administrator wants that no one can access these sites .The easiest way that can be adopted is to use a dedicated firewall and block all those IP addresses that are frequent users of these sites but the users nowadays are very intelligent so they can find some alternative like they can use some proxy services and can access those sites without the administrator's knowledge. So the Administrator really want to be able to block access to these by using URLs embedded inside HTTP requests. This can only be done using DPI.

The other thing of interest is the use of JPEG's. Everyone is very interested in opening and seeing whatever JPEG is sent to them even by an unknown user. These JPEG's are sent to them either through emails and when the user clicks them some backdoor gets installed without their knowledge. There are large number of patches available for these exploits but it is virtually impossible to install patches on each and every system when the number of users is very large. The only way left with the Administrator is to stop these JPEG's from coming to the network or more specifically going to each user.

P2P file sharing applications are bane for users but they steal much more bandwidth. So the administrator sees the traffic and notice that P2P is offering some problems to other users and because of that the Administrator really wants to stop P2P file sharing or transmission of P2P traffic but how can he do so using a firewall because a firewall can only work by blocking IP addresses. As a result of which the administrator wants some alternative that can classify and differentiate P2P traffic and non-P2P traffic.

Traditional firewalls fall short in these tasks and some of the advanced firewalls will be too rigid. They won't let do custom processing. Thus the emphasis is on openness. DPI is very crucial and this thesis has IPS as part of DPI. Work carried out is based on open technologies, will be able to keep a great degree of control on how IPS scripts build while keeping it highly customizable. Last but not the least such solutions need to work in a way as to have minimal effect on the throughput and latency of the network.

Chapter 4
Implementation

IDS/IPS systems go for layer 3, 4 and higher. They are specifically designed to detect intrusions into the network using services that are allowed on the network but impossible to detect by just IP addresses and TCP/UDP ports. Specifically they look for patterns in layer 5 and above and try to match them with a known set of malicious pattern rules-signatures. Any matches trigger actions such as alerting the administrator or dropping the flow altogether .The solution mentioned above do something useful but it fell short when an administrator tries to tackle issues related to today's network. As indicated above some of the problems today require deeper understanding of protocols residing on top of TCP and UDP.IDS/IPS systems come closest in that respect. They look for patterns in the content and make decisions based on those but that's not sufficient most of the times. Pattern matches are good at finding worm like content in the traffic where the pattern in question is a known set of strings. Pattern matches fall short when decisions are based on programmable logic or a state machine. Also some requirements dictate that more than one packet be looked at to make decisions.

A DPI based system would let the administrator write network policies based on not only the network and transport layer but also the content encompassed by the application layer. Firewalls can't do that kind of stuff .They may be changed to build application layer intelligence. No doubt about that but there's a formidable impediment. Firewall policies definition primitives will need to change too. A firewall policy is defined in terms of 5 tuples or parts thereof. For example allow traffic from subnet 192.168.1.0/24 to any host with the TCP destination port 80 and drop rest of the traffic. Now imagine when a firewall need to be extended to write policies with rules that cover application layers. The range of possibilities in application layer is virtually limitless. A firewall with such support would either cover only a limited functionality or would never be in a finished state. DPI firewalls need to be really flexible, in terms of letting users very finely specify what they are looking for. A DPI firewall which presents the user with the most flexible way of specifying interesting traffic, will be the one that will be most useful and powerful.

Using BRO helps a great deal in finding intrusions thus acting as IDS but it also offers power which is far more than just IDS because they not only use a fixed set of patterns but instead they are based on some state based patterns which can keep an eye on future attacks also. Bro is intended for use by sites requiring flexible, highly customizable intrusion detection. It is important to understand that Bro has been developed primarily as a research platform for intrusion detection and traffic analysis. It is not intended for someone seeking an "out of the box" solution. Bro is designed for use by Unix experts who place a premium on the ability to extend an intrusion detection system with new functionality as needed, which can greatly aid with tracking evolving attacker techniques as well as inevitable changes to a site's environment and security policy requirements. Since Bro is open source and runs on commodity PC hardware, it provides a low-cost means to experiment with alternative techniques. Some sites may wish to run a commercial IDS as their front-line of defense, and then also run Bro as a way to:

- Verify the results of the commercial IDS / defense-in-depth
- Attain richer forensics capabilities
- Provide policy-checking capabilities not facilitated by the commercial IDS
- Experiment with new approaches and incorporate leading-edge research

Thus one can say that BRO not only act as IDS but can also perform the function of IPS and that too with much more flexibility. Here are some of the examples of BRO about what we can do with BRO and these examples also show how we can customize Bro. First of all one need to capture packets i.e traffic need to be captured .This traffic is then analyzed for patterns. One can capture the traffic by either using Wireshark or Tcpdump or the traffic can also be captured live. Once the packet is captured using Wireshark or Tcpdump then it is analyzed offline because the traffic captured through Wireshark or Tcpdump can only be analyzed offline.

This section discusses how to run it interactively (mostly useful for building up familiarity with the policy language, not for traffic analysis), running it on live and recorded network traffic, modifying Bro policy scripts, and the different run-time flags.

The semantics of certain files which are very important are as follows:

`@load`

The `@load` directive can be used in a policy script to indicate that Bro should at that point process another policy script (like C's include directive). So one could have in *my-policy*:

```
@load my-additional-policy
```

bro.bif.bro

It contains all the built in functions. This file is automatically loaded in each and every bro file. The functions which are not built in, are used through namespaces like For example :

module http;

Note that namespaces are used here just for functions.

/usr/local/bro/policy/bro.init

This file contains all the data structures used in bro i.e all data structures are defined here.

/usr/local/bro/policy/event.bro.bif

This file contains all the event handlers i.e all the event handlers are declared here in this file.Bro gets installed by a script.

During installation if it asks for libtermcap during ./configure do the following:

dpkg –L libncurses *

sudo apt-get install libncurses-dev

sudo apt-get install libssl-dev

make

make install

make install-brolite

Setting Path:

export BROHOME=/usr/local/bro

export BROPATH=$BROHOME/site:$BROHOME/policy

export PATH=/usr/local/bro/bin:$PATH

Running BRO

Inorder to run Bro first of all the user should be in /usr/local/bro directory and then run bro as following:
/usr/local/bro/etc/bro.rc –start

This starts bro and if any one want to check the status or want to stop one should simply use

/usr/local/bro/etc/bro.rc –checkpoint
/usr/local/bro/etc/bro.rc –stop

Now to run Bro on particular bro file first of all the traffic should be captured and it is captured as

tcpdump –s 0 –w trace.out (ctrl+c)
and then run Bro on this captured traffic file trace.out as
bro –r trace.out hostfile.bro

4.1 Customizing Bro

Bro is very customizable, and there are several ways to modify Bro to suit user's environment. One can write one's own policy analyzers using the Bro language. Most sites will likely just want to do minor customizations, such as changing the level of an alert from "notice" to "alarm", or turning on or off particular analyzers. The default policy scripts for Bro are all in $BROHOME/policy. These files should never be

edited, as user's edits will be lost when anyone upgrade Bro. To customize Bro for one's site, one should make all the changes in $BROHOME/site. The default policy scripts for Bro are all in $BROHOME/policy. Remember that these files should never be edited, as all the edits will be lost when one upgrade Bro. [25]To customize Bro for any site, one should make all changes in $BROHOME/site. Many simple changes just require you to *redefine* (using the redef operator, a Bro constant from a standard policy script with your own custom value. One can also write own custom script to do whatever you want. Many simple changes just require one to *redefine* (using the redef operator, a Bro constant from a standard policy script with one's own custom value)[25]. The Administrator can also write his own custom script to do whatever is wanted.

For example, to add "guest" to the list of forbidden_ids (user names that generate a login alarm), you do this:

```
redef forbidden_ids += { "guest", };
```

4.2 Bro Policy Files

Bro *policy* script is the basic analyzer used by Bro to determine what network events are alarm worthy. A policy can also specify what actions to take and how to report activities, as well as determine what activities to scrutinize. Bro uses policies to determine what activities to classify as *hot*, or questionable in intent. These hot network sessions can then be flagged, watched, or responded to via other policies or applications determined to be necessary, such as calling `rst` to reset a connection on the local side, or to add an IP address block to a main router's ACL (Access Control List). Policy files are loaded using an `@load` command. The semantics of `@load` are "load in this script if it hasn't already been loaded", so there is no harm in loading something in multiple policy scripts[25].

4.3 Capturing Network Traffic

In order to detect network security threats, the first step is to capture the network traffic and then do it's analysis. For capturing the packets from the network traffic flowing in the traffic various tools can be used like tcpdump, ethereal, wireshark. These tools can capture the data packets and these captured packets can be used for active analysis or passive analysis. In active analysis the data packets at the same time when they are captured, on the other hand in passive analysis packets once

captured, they are stored and analyzed later in future. So both active as well as passive network traffic analysis is important [29].Some of the tools which can capture the network traffic or packets flowing in the network are as tcpdump, ethereal, snort. All of these tools are developed by using popular programming library called libpcap that provides a high level interface to packet capture [29,30].

4.3.1 Using tcpdump for Network Traffic Capturing

Tcpdump prints out a description of the contents of packets on a network interface that match the Boolean expression. It can also be run with the **-w** flag, which causes it to save the packet data to a file for later analysis, and/or with the **-r** flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match expression will be processed by tcpdump [31]. When tcpdump finishes capturing packets, it will report counts of: packets captured, Packets received by filter (the meaning of this depends on the operating system on which tcpdump is running, and possibly on the way the Operating system was configured - if a filter was specified on the command line, on some Operating system it counts packets regardless of whether they were matched by the filter expression and, even if they were matched by the filter expression, regardless of whether tcpdump has read and processed them yet, on other Operating systems it counts only packets that were matched by the filter expression regardless of whether tcpdump has read and processed them yet, and on other Operating systems it counts only packets that were matched by the filter expression and were processed by tcpdump), packets dropped by kernel[29].

4.3.2 Using Ethereal for Network Traffic Capturing

Ethereal is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. Ethereal is a very powerful well featured packet analyzer. It captures packets and decodes them into their component parts for analysis. The range of decodes is very large, if we've heard of a protocol there's a good chance that Ethereal has a decoder for it, and if you haven't there's still likely to be a decoder for it. Ethereal is available for use on UNIX systems and for Microsoft Windows also. We can think of a network packet analyzer as a measuring device used to examine what's going on inside a network cable [32].

➢ Some Intended purposes to use Ethereal

Here are some examples people use Ethereal for:

• network administrators use it to troubleshoot network problems

• network security engineers use it to examine security problems

• developers use it to debug protocol implementations

• people use it to learn network protocol internals

➢ Packet Capturing using Ethereal from many different media

Despite its name, Ethereal can capture traffic from network media other than Ethernet. Ethereal can save packets captured in a large number of formats of other capture programs. Ethereal is an open source software project, and is released under the GNU General Public License.

➢ Start Packet Capturing

Ethereal can be used in two ways to start capturing packets with Ethereal:

1. From the command line using the following:

ethereal -i eth0 –k

This will start Ethereal capturing on interface eth0.

2. By starting Ethereal and then selecting Start... from the Capture menu, this brings up the Capture Options dialog box.

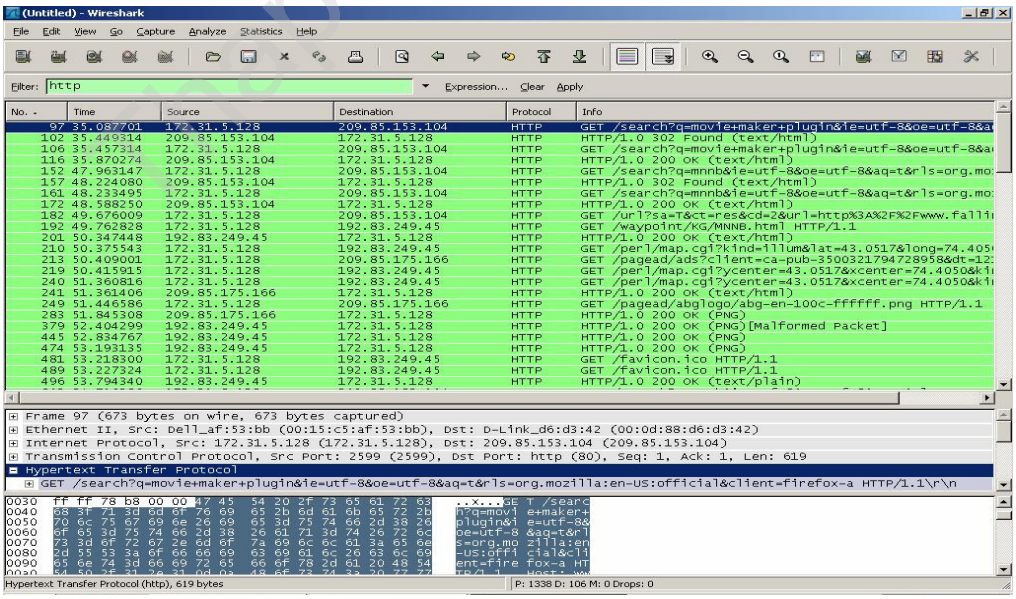

Figure 4.1 Packet capturing using Ethereal

The syntax for capturing the traffic offline by using Tcpdump is as follows:

tcpdump -s 0 -w trace.out

This captures the traffic and create a file named trace.out which can then be further analyzed. We can further customize it like we can capture only smtp or http data then we can use the following syntax:

tcpdump -s 0 -w trace.out port smtp or port http

Kill off the tcpdump after capturing traffic for a few minutes (use ctrl-C). We can also capture traffic using Wireshark.

4.3.3 Capturing live traffic

The traffic can also be analyzed live rather than offline as follows:

```
#bro –i eth1 –i eth2 myhost.mysite.org.bro
```

The traffic is captured live using interface eth1 and a bro policy file ipno.bro is run against it. The following is the output of the captured live traffic.

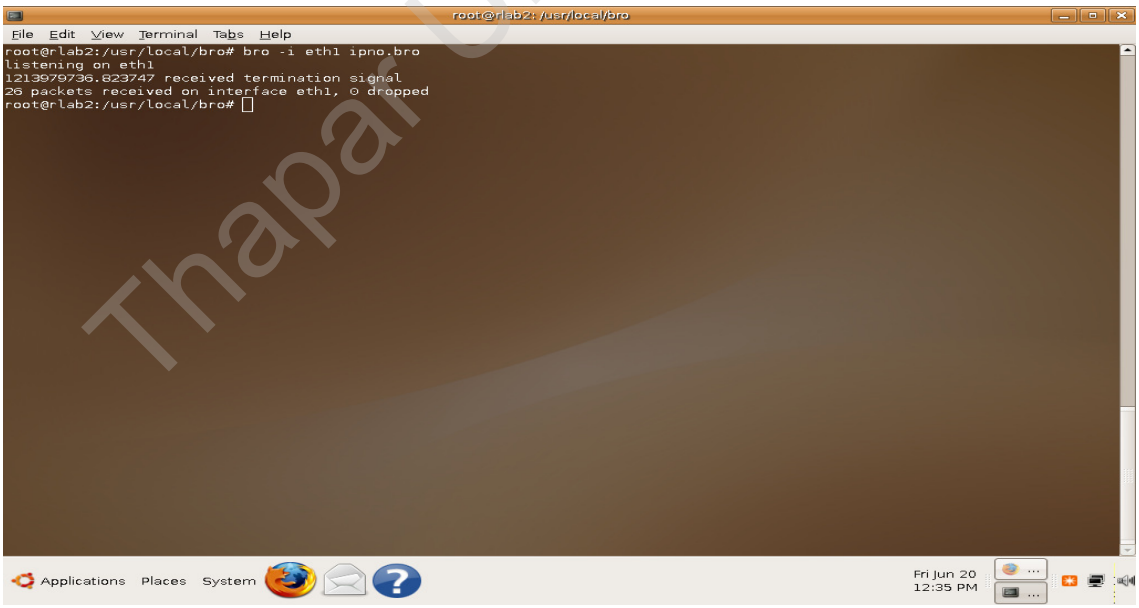

Figure 4.2 Live Traffic Capture

```
# export BROHOME= /usr/local/bro
# export BROPATH= $BROHOME/site:$BROHOME/policy
#export PATH=/usr/local/bro/bin:$PATH
```

bro -r trace.out hostname.bro

Here first of all various PATHS are set and then the captured file trace.out is run against a .bro file named as hostname.bro which contain various policies. Bro policy script is the basic analyzer used by Bro to determine what network events are alarm worthy. A policy can also specify what actions to take and how to report activities, as well as determine what activities to scrutinize. Bro uses policies to determine what activities to classify as hot, or questionable in intent. Policy files are loaded using an @load command. The semantics of @load are "load in this script if it hasn't already been loaded", so there is no harm in loading something in multiple policy scripts.

The following policy scripts are included with Bro:

Site	Weird	Smtp
Alarm	Conn	Http
Tcp	Hot	Http-request
Login	Ftp	Http-reply

These policy scripts are added by default to each bro file but some are explicitly loaded like:

Drop	Dns	Backdoor
Icmp	Ssl	Analy
Dns	Passwords	Ident

These policy scripts contains the rules or policies which are applied on the traffic and based on that malicious patterns are recognized. For example backdoor.bro file discussed above contains the code for a file into which backdoor servers () are written. There is another policy file smtp.bro. This is used to detect relaying. Similarly other policy files also contains rules that are run against the captured traffic.

Now the packets have been captured ,so just to begin with let us have a brief idea about what code is used internally to capture the packet and for that purposes we need to look into the pcap library because it contains the required functions for fetching the packets. The function that is used in the beginning is pcap_lookupdev(errbuf) function and it is used internally to set the interface that will be used for sniffing the packet. In the event that the command fails, the function will populate the string errbuf with a description of the error. In this case, if pcap_lookupdev() fails, it will store an error message in errbuf. Now after the device

has been set up the device is opened for sniffing and for that the function which is used is pcap_t *pcap_open_live (char *device, int snaplen, int promisc, int to_ms, char *ebuf). snaplen is an integer which defines the maximum number of bytes to be captured by pcap. promisc, when set to true, brings the interface into promiscuous mode (however, even if it is set to false, it is possible under specific cases for the interface to be in promiscuous mode, anyway). to_ms is the read time out in milliseconds (a value of 0 sniffs until an error occurs; -1 sniffs indefinitely). Lastly, ebuf is a string we can store any error messages within (as we did above with errbuf). The function returns our session handler. After the device has been set up the next thing done is filtering specific traffic i.e what traffic need to be analyzed and for that the function that runs in the background is int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask). The first argument is our session handle (pcap_t *handle). Following that is a reference to the place we will store the compiled version of our filter. Then comes the expression itself, in regular string format. Next is an integer that decides if the expression should be "optimized" or not (0 is false, 1 is true. Standard stuff.) Finally, we must specify the net mask of the network the filter applies to. The function returns -1 on failure; all other values imply success.

Now as some limitations are being set now comes the turn to capture packets and for this the function used is u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h). The first argument is our session handler. The second argument is a pointer to a structure that holds general information about the packet, specifically the time in which it was sniffed, the length of this packet, and the length of his specific portion (incase it is fragmented, for example.) pcap_next() returns a u_char pointer to the packet. If one just wants to capture only one packet the pcap_next() is sufficient but inorder to capture multiple packets pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user) or pcap_dispatch() functions are used[28].Here in pcap_loop() notice the last argument i.e *user of the type u_char .This variable points to the entire packet itself but how can one know which part of the packet to access. For that purposes lets first of all understand the structure of a packet. A packet consists of Ethernet header,IP header and TCP header. In order to access these headers we need to typecast *user variable.

The structure of these headers is as follows::

```
/* Ethernet header */
struct sniff_ethernet {
    u_char  ether_dhost[ETHER_ADDR_LEN];  /*  Destination  host  address
*/
    u_char ether_shost[ETHER_ADDR_LEN]; /* Source host address */
    u_short ether_type; /* IP? ARP? RARP? etc */
};

/* IP header */
struct sniff_ip {
    #if BYTE_ORDER == LITTLE_ENDIAN
    u_int ip_hl:4, /* header length */
    ip_v:4; /* version */
    #if BYTE_ORDER == BIG_ENDIAN
    u_int ip_v:4, /* version */
    ip_hl:4; /* header length */
    #endif
    #endif /* not _IP_VHL */
    u_char ip_tos; /* type of service */
    u_short ip_len; /* total length */
    u_short ip_id; /* identification */
    u_short ip_off; /* fragment offset field */
    #define IP_RF 0x8000 /* reserved fragment flag */
    #define IP_DF 0x4000 /* dont fragment flag */
    #define IP_MF 0x2000 /* more fragments flag */

    #define IP_OFFMASK 0x1fff /* mask for fragmenting bits */
    u_char ip_ttl; /* time to live */
    u_char ip_p; /* protocol */
    u_short ip_sum; /* checksum */
    struct in_addr ip_src,ip_dst; /* source and dest address */
};
/* TCP header */
struct sniff_tcp {
    u_short th_sport; /* source port */
    u_short th_dport; /* destination port */
    tcp_seq th_seq; /* sequence number */
    tcp_seq th_ack; /* acknowledgement number */
```

```
    #if BYTE_ORDER == LITTLE_ENDIAN
    u_int th_x2:4, /* (unused) */
    th_off:4; /* data offset */
    #endif
    #if BYTE_ORDER == BIG_ENDIAN
    u_int th_off:4, /* data offset */
    th_x2:4; /* (unused) */
    #endif

    u_char th_flags;
    #define TH_FIN 0x01
    #define TH_SYN 0x02
    #define TH_RST 0x04
    #define TH_PUSH 0x08
    #define TH_ACK 0x10
    #define TH_URG 0x20
    #define TH_ECE 0x40
    #define TH_CWR 0x80
    #define                                                    TH_FLAGS
(TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
    u_short th_win; /* window */
    u_short th_sum; /* checksum */
    u_short th_urp; /* urgent pointer */
};
```

4.4 Extracting IP addresses and Port numbers

As the basics of pcap library are understood we can use this knowledge to set device and find IP addresses embedded in the packet. Once we understand how to find IP addresses it will become easier to understand the basics of packet structure. The program that can be used for finding these is:

```
int main()
{
char *dev,errbuf[PCAP_ERRBUF_SIZE];
pcap_t *des;
bpf_u_int32 mask;
bpf_u_int32 net;
const u_char *packet;
struct pcap_pkthdr hdr;
```

```
struct ether_header *eptr;

const struct ethhdr *ethernet;
const struct iphdr *ip;
const struct tcphdr *tcp;
const char *payload;

int size_ethernet = 14;
int size_ip=sizeof(struct iphdr);
int size_tcp;
int size_payload;
struct in_addr myaddr;

dev=pcap_lookupdev(errbuf);
pcap_lookupnet(dev, &net, &mask, errbuf);
myaddr..s_addr=net;

printf("\nDEV: %s\n",dev);
printf("\nNET: %s\n",(char*)inet_ntoa(myaddr));

des=pcap_open_live(dev,BUFSIZ,0,-1,errbuf);
packet = pcap_next(des,&hdr);

ip = (struct iphdr*)(packet + size_ethernet);
myaddr.s_addr=ip->saddr;

printf("\nSRCIP: %s\n",(char*)inet_ntoa(myaddr));

myaddr.s_addr=ip->daddr;

printf("\nDESTIP: %s\n",(char*)inet_ntoa(myaddr));

tcp=(struct tcphdr*)(packet + size_ethernet + size_ip);

printf("\nSRCPORT: %d\n",ntohs(tcp->source));

printf("\nDESTPORT: %d\n",ntohs(tcp->dest));
}
```

Inorder to compile this program(lib.c) we need to compile it using gcc lib.c –lpcap inorder to have smooth run of it otherwise some errors might occur like:

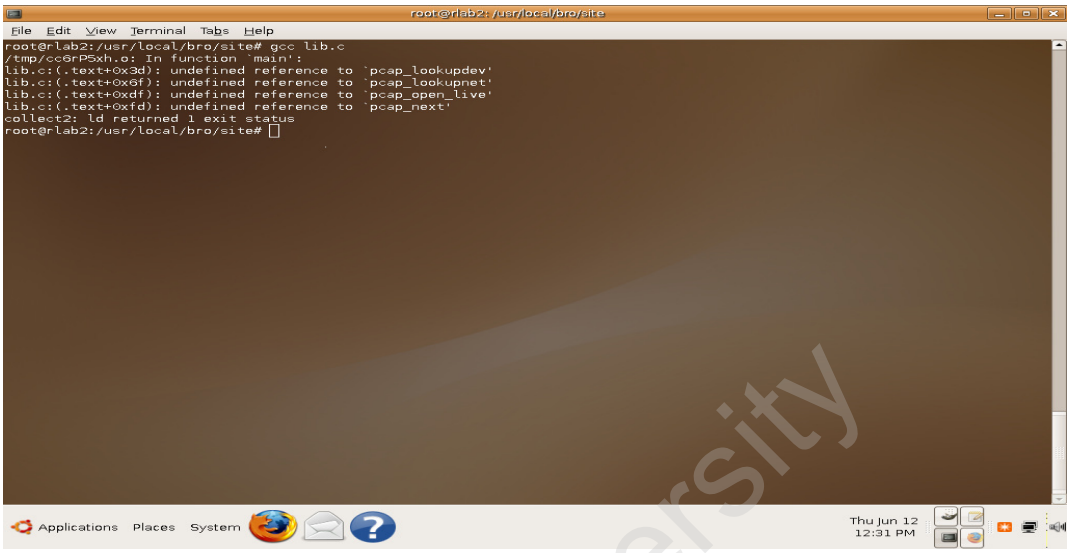

Figure 4.3 Errors without lpcap

So in order to run this program smoothly we need to compile it using argument –lpcap and then run it .The output is shown as:

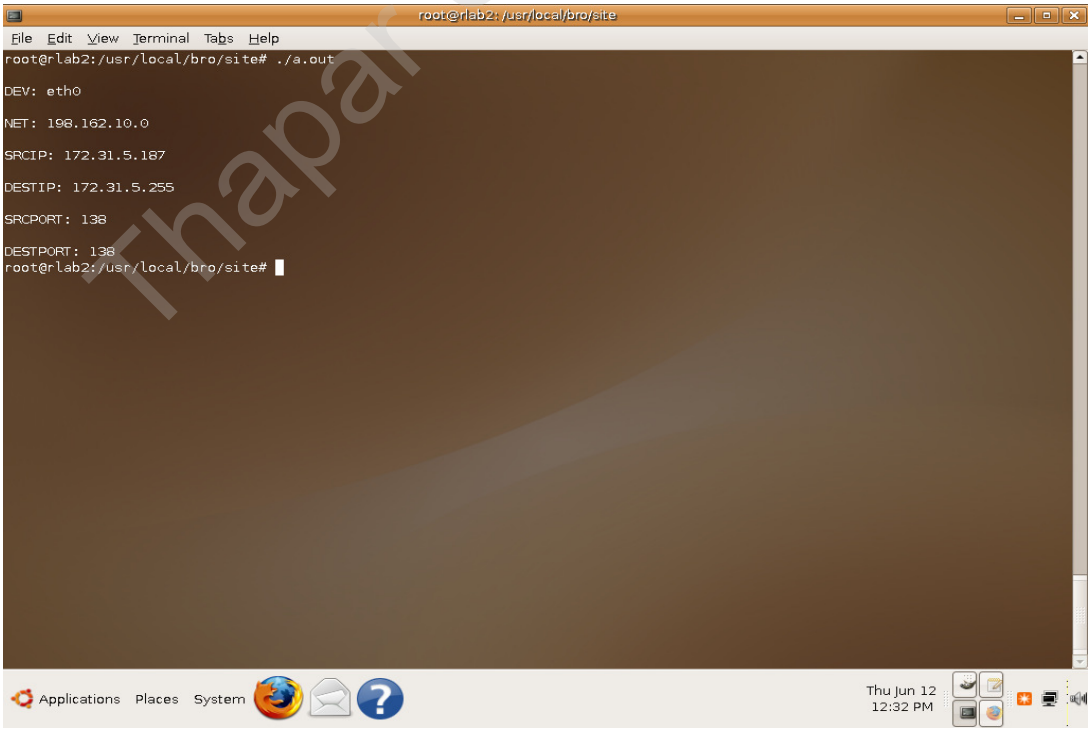

Figure 4.4 Output showing IP addresses and port numbers using pcap library

This output was given by directly interacting with the pcap libraray. Now the same thing is to be done using Bro IDS.

4.5 Extracting IP addresses using BRO

Using Bro one can write policy for extracting the IP address of source and destination. This program will certainly give an idea about how to write custom policy scripts. The program finds out the IP addresses and this is how it is done:

```
@load http
@load weird
redef ignore_checksums=T;
redef weird_action += { ["non_IPv4_packet"] = WEIRD_IGNORE, };
event http_request(c: connection, method: string, original_URI: string,
  unescaped_URI: string, version: string)
{
local id = c$id;
print fmt("%s%s %s %s %s%s %s %s",id$orig_h,":",id$orig_p,"to",id$resp_h,":",id$resp_p,method,original_URI);
}
```

The `@load` directive can be used in a policy script to indicate that Bro should at that point process another policy script called http policy script.Similarly @load weird is used to load the weird policy file. Weird policy script initialize generic mechanism for detecting unusual events. Here in this program if some weird event occurs like some non IPv4 packets also come across like other packets then they should not create weird event instead these weird events are ignored ,so when any non IPv4 packet come across no weird event props up.

The output of the above program is as follows:

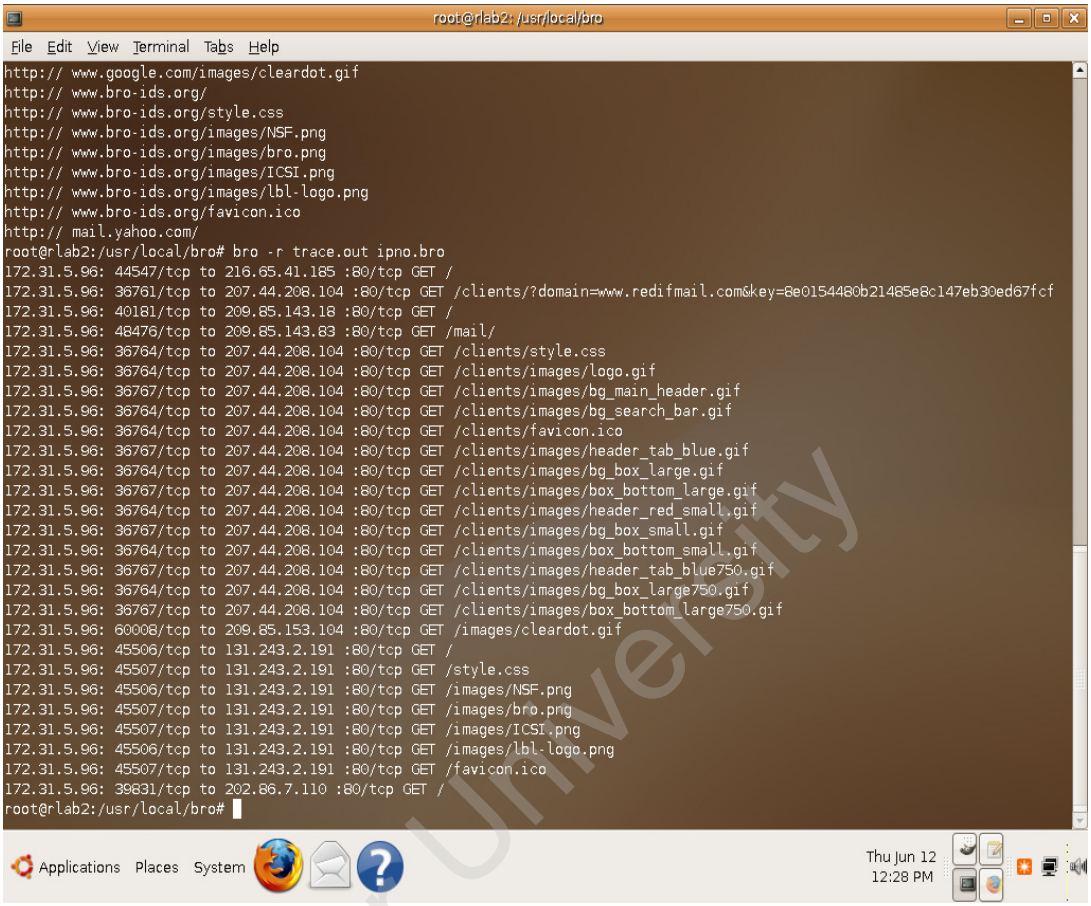

Figure 4.5 IP addresses and port numbers using Bro scripts

4.6 Extracting URLs

This is sufficient to become familiar with capturing and extracting packets. Now let us extract full URLs embedded in the packets and not only the URIs.

This can be achieved as follows:

```
@load http
redef ignore_checksums= T;
global path: string;
event http_request(c: connection, method: string, original_URI: string, unescape
d_URI: string, version: string)
{
path = original_URI;
}
```

```
event http_header(c: connection, is_orig: bool,name: string, value: string)
{
local s=lookup_http_request_stream(c);
if(name=="HOST")
print fmt("http://%s%s",value,path );
}
```

While trying to capture URLs sometimes checksum errors may also creep up which effect the output ,so in order to get the correct output it must be stopped and for that one has to redefine ignore_checksum variable to TRUE. Note that here two event handlers are to be used and not only one because http_header cant work alone. In order for it to work one must use http_request event handler because only when http_request event occurs only then one can access its header and in the http_header one can print both the original_URI and its value thus getting the full URL.

The output of the program for fetching the entire URL is as follows:

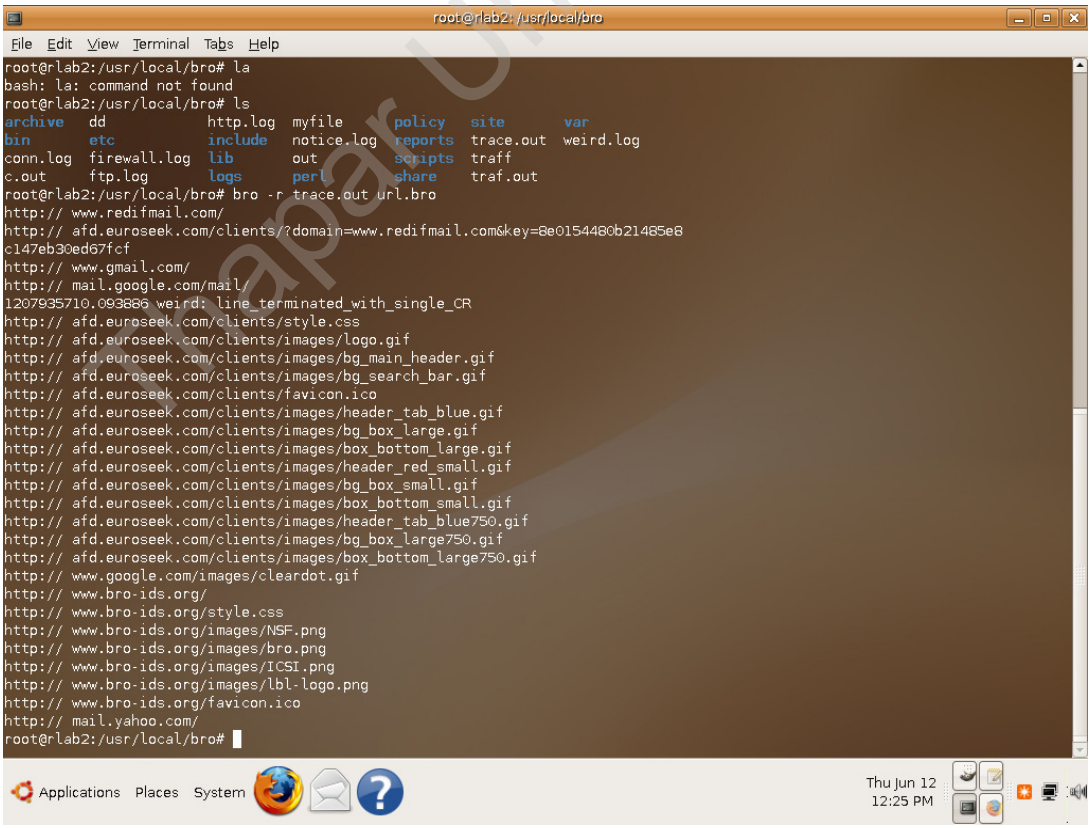

Figure 4.6 Extracting URLs using Bro Script

Till now Bro is used for intrusion detection only but is intrusion detection only sufficient. Now a question arises that if one is aware that something is going to happen in network because of which network is going to be down but it cannot be stopped then what is the benefit of intrusion detection. Bro not only provides intrusion detection but also it can be customized for intrusion prevention to take action on what is seen as malicious. For example one can block an IP which is doing something illegal but before doing that one must understand about iptables. iptables is a user space application program that allows a system administrator to configure the Netfilter tables, chains, and rules. Because iptables requires elevated privileges to operate, it must be executed by user root, otherwise it fails to function.

On most Linux systems, iptables is installed as `/usr/sbin/iptables`. Linux comes with advanced tools for packet filtering, the process of controlling network packets as they enter, move through, and exit the network stack within the kernel. Traffic moves through a network in packets. A network packet is collection of data in a specific size and format. In order to transmit a file over a network, the sending computer must first break the file into packets using the rules of the network protocol. Each of these packets holds a small part of the file data. Upon receiving the transmission, the target computer reassembles the packets into the file[33].These packets are further processed by tables or chains.

The Linux kernel contains the built-in ability to filter packets, allowing some of them into the system while stopping others. The 2.4 kernel's netfilter has three built-in tables or rules lists. They are as follows:

- `filter` — This is the default table for handling network packets.
- `nat` — This table used to alter packets that create a new connection.
- `mangle` — This table is used for specific types of packet alteration.

Each of these tables in turn have a group of built-in chains which correspond to the actions performed on the packet by the netfilter.

The built-in chains for the `filter` table are as follows:

- *INPUT* — This chain applies to packets received via a network interface.
- *OUTPUT* — This chain applies to packets sent out via the same network interface which received the packets.
- *FORWARD* — This chain applies to packets received on one network interface and sent out on another.

The built-in chains for the nat table are as follows:

- *PREROUTING* — This chain alters packets received via a network interface when they arrive.
- *OUTPUT* — This chain alters locally-generated packets before they are routed via a network interface.
- *POSTROUTING* — This chain alters packets before they are sent out via a network interface[33].

4.7 Using IPtables in BRO

The following script shows how to do it.

```
redef ignore_checksums=T;
event connection_established(c: connection)
{
system(fmt("%s" ," iptables -A FORWARD -s 198.162.10.6 -j DROP -p tcp --destinat
ion-port http"));
}
```

In this program what is being done is when an event occurs like connection is established, an event handler for this particular event is called from the site directory and this event handler call a function called system which further use iptable command to DROP all the packets coming from IP address 198.162.10.6 to any of the destination port. There are many of the other powerful features of iptables which can be used with Bro like one can ACCEPT the incoming packets, BLOCK them, or FORWARD them to the destination port. Apart from this one can minutely specify, that the incoming packet should be accepted at which port and which interface and whether one should be allowed to send ICMP data or not. So, when the power of iptables is combined with Bro, the security system becomes very powerful and can be used for prevention purposes, so that the network remains protected from the intruders.

The output of the above program is as follows:

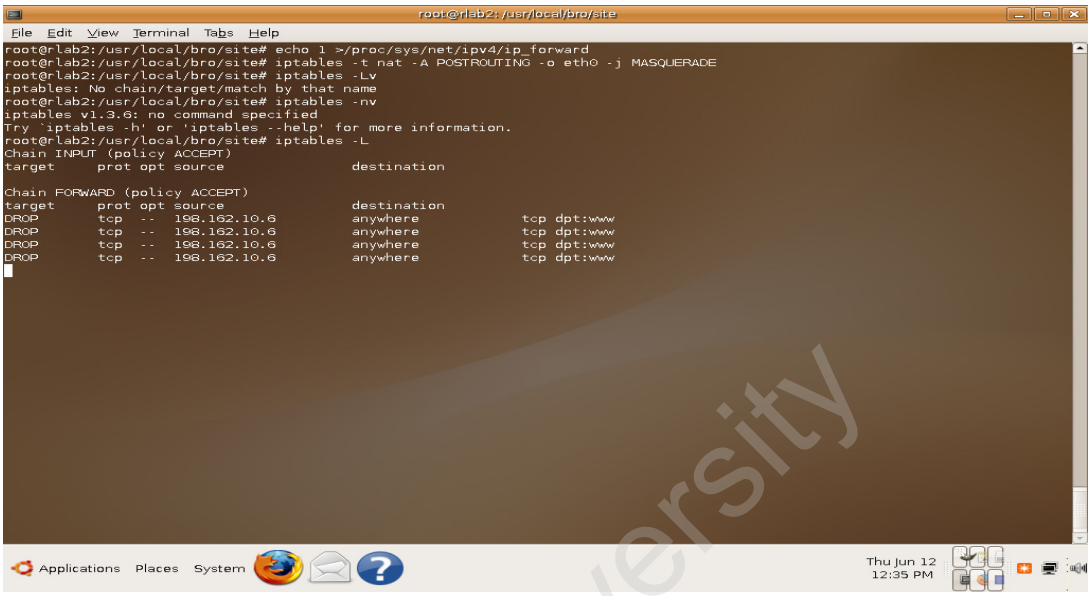

Figure 4.7 Prevention using Iptables and Bro

Chapter 5

Conclusions and Future Work

5.1 Conclusion

There is a huge disparity between the number of open-source IDSs available as compared to IDSs. The same holds true for availability of these systems on Linux. Apart from this there are many type of tools available in the market like firewalls etc besides IDS but all these are commercial products and once build they are very difficult to customize but on the other hand Bro can be easily customized and whatever policies or rules an administrator wants to have on the network can be easily done using Bro. In this work it is found that using Bro one can build own security system which can inspect the packets flowing through the network and extract the URL's and other important information like what protocol is being used or what method is being used to send data and based on that one can take actions and make the network secure. Thus Bro provides us with a complete information about the malicious content and this output is then used by the Bro engine to take action for ensuring network security thus acting as IPS and that too at very low cost because commercial applications are very costly but Bro helps us build our own IDS/IPS, we just need a system with Linux and Bro running on it.

5.2 Future Work

Bro is relatively a very new scripting language and is just in its beginning phase and very little work has been done. In this work basically the work is done on how to extract URLs and IP addresses, port numbers apart from iptables, so one is able to detect intrusions and can prevent it but the work mainly revolved around built in scripts besides building a few new ones also but still rather than using iptables, one can develop own code also .So this feature of Bro also need to be worked upon .The other thing to work upon is the use of signatures that are built in and are used for detecting viruses etc. Also rather than using built in signatures which are of fixed type one can use Bro to build signatures based on regular expressions which can not only detect already available viruses but that also that may cause havoc in future and its quite sure that if work is done in this direction attacks can be prevented thus making our networks very secure.

References

[1] Modern Network Security: The Migration to Deep Packet Inspection
White Paper – The Migration to Deep Packet Inspection

[2] http://en.wikipedia.org/wiki/Image:Phishing_chart.png

[3] Intro to IDS by Tony Bradley

[4] http://www.ranum.com/security/computer_security/editorials/deepinspect/

[5] http://www.securityfocus.com

[6] The Evolution of Intrusion Detection System by Paul Innella, Tetrad Digital Integrity, LLC

[7] Security Focus by Thomas Porter, Ph.D., CISSP

[8] Algorithms to Accelerate Multiple Regular Expressions Matching for Deep packet Inspection by Sailesh Kumar, Sarang Dharmapurikar, Fang Yu
Patrick Crowley, Jonathan Turner, Washington University
Computer Science and Engineering

[9] H. Debar, M. Dacier, A. Wespi, Towards a taxonomy of intrusion-detection systems, Computer Networks 31, 1999, pages 805-822

[10] E. Lundin, E. Jonsson, Survey of research in the intrusion detection area, Technical report 02-04, Department of Computer Engineering, Chalmers University of Technology, Göteborg January 2002, http://www.ce.chalmers.se/staff/emilie/papers/Lundin_survey02.pdf

[11] C. Krügel, T. Toth, Applying Mobile Agent Technology to Intrusion Detection, ICSE Workshop on Software Engineering and Mobility, Toronto May 2001

[12] C. Krügel, T. Toth, Distributed Pattern Detection for Intrusion Detection, Conference Proceedings of the Network and Distributed System Security Symposium NDSS '02, 2002

[13] 13 J.S. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isaco, E. Spafford, D. Zamboni, An Architecture for Intrusion Detection using Autonomous Agents, 14th IEEE Computer Security Applications Conference ACSAC '98, December 1998, pages 13-24

[14] 14 D.J. Ragsdale, C.A. Carver, J.W. Humphries, U.W. Pooh, Adaptation techniques for intrusion detection and intrusion response systems, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2000, pages 2344-2349

[15] Theuns Verwoerd, "Active Network Security",Honours Report 5 November 1999.

[16] The Great IDS Debate : Signature Analysis Versus Protocol Analysis by Matt Tanase

[17] Introduction to Intrusion Detection Systems (IDS) from Tony Bradley

[18] Signature-Based or Anomaly-Based Intrusion Detection: The Practice and Pitfalls by Arnt Brox

[19] Rafeeq Ur Rehman, "Intrusion Detection System with Snort",New Jersey

[20] IBM Internet Security Systems

URL:http://www.iss.net/blackice/

[21] URL:http://www.nss.co.uk

[22] Bruce Schneier,"Secrets and Lies",John Wiley & Sons,2000

[23] http://www.auditmypc.com/freescan/readingroom/

[24] Allot Communications,"Digging Deeper into Deep Packet Inspection(DPI)",white paper.

[25] http://bro-ids.org/

[26] http://www.bro-ids.org/wiki/index.php/Reference_Manual:_Introduction

[27] Paul D. Robertson, Matt Curtin, Marcus J. Ranum

URL:http://www.interhack.net/pubs/fwfaq/firewalls-faq.html

[28] Programming with LibPcap: A Pcap Tutorial by Tim Carstens

[29] Mark E. Crovella, Member, IEEE, and Azer Bestavros, Member, IEEE "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes" IEEE/ACM Transactions On Networking, VOL. 5, NO. 6, December 1999.

[30] Luca Deri "Improving Passive Packet Capture: Beyond Device Polling"
URL: http://luca.ntop.org/

[31] Van Jacobson, Craig Leres and Steven McCanne "tcpdump Manual"
URL: http://www.tcpdump.org/

[32] Richard Sharpe "Ethereal User's Guide V2.00 for Ethereal 0.10.5"

[33] Red Hat Linux 8.0: The Official Red Hat Linux Reference Guide

[34] Addressing the Limitations of Deep Packet Inspection with Complete Content Protection

[35] Network Security, Filters and Firewalls by Darren Bolding

[36] http://www.itsecurity.com/features/network-security-threats-011707/

List of Papers Published/Communicated

Navdeep Singh, Maninder Singh, "Intrusion Prevention Scripts to Detect Malicious Traffic using BRO", 16th IEEE INTERNATIONAL CONFERENCE ON NETWORKS(ICON 2008), 12-14th DECEMBER, 08, NEW DELHI, INDIA (Communicated).