Deep Packet Inspection in Linux Kernel Firewall

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

Master of Engineering
in
Computer Science & Engineering

By:
Vaibhav Bhadade
(80732004)

Under the supervision of:

Dr. Maninder Singh
Associate professor

Dr. V. P. Singh
Assistant professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
JULY 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Deep Packet Inspection in Linux Kernel Firewall"**, in partial fulfilment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala is an authentic record of my own research work carried out under the supervision of Dr. Maninder Singh and Dr. V. P. Singh, it refers to other researcher's works, also which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Bhadade Vaibhav

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

(Dr. Maninder Singh)
Computer Science and
Engineering Department
Thapar University, Patiala.

(Dr. V. P. Singh)
Computer Science and
Engineering Deparment
Thapar University, Patiala.

Countersigned by

(Dr. Rajesh Bhatia) 15/7/09.
Assistant Professor & Head
Computer Science & Engineering Department
Thapar University Patiala.

16/7
(R. K. SHARMA)
Dean (Academic Affairs)
Thapar University,
Patiala

Acknowledgment

I wish to express my deep gratitude to Dr. Maninder Singh , Associate Professor, and Dr. V. P. Singh, Assistant Professor Computer Science & Engineering Department, TU, Patiala for providing there uncanny guidance and support throughout the thesis. I sincerely thank Mr. Maninder singh, Software Engineer, CSE for his constant support and guidance during my thesis work. To him, I owe a great debt of gratitude for his patience and inspiration.

I am thankful to Dr. Rajesh Bhatia, Computer Science & Engineering Department, TU, Patiala, for the motivation and inspiration that triggered me for the thesis work. I would also like to thank all the staff members who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of the thesis.

Last but not the least, I express my heartfelt thanks to my parents and all of my friends for encouraging me and providing me useful information during my work.

Finally, my special thanks go to authors whose works I have consulted and quoted in this work.

Bhadade Vaibhav

Abstract

In the early days of the Internet, non-discrimination was easy because it was not technologically feasible for service providers to inspect messages and evaluate their content in real time. But recently, electronics manufacturers have developed so-called Deep Packet Inspection technology capable of tracking Internet communications in real time, monitoring the content, and deciding which messages or applications will get through the fastest.

There are some Deep Packet Inspection solution available in the market, but each of these have some limitation in one or other aspect.

This thesis work discuss Linux based network security concepts, infrastructure, tools and applications and leverages these to build a new user space Application Protocol Interface (API) for Deep Packet Inspection which will provide a consistent, standard and highly flexible packet control solution for implementing firewalls.

Contents

List of Figures

Computer Science and Engineering Department TU Patiala

Chapter 1

Introduction

As people use insurance to protect their valuables from fire or theft. Businesses protect themselves from intellectual theft through patents and trademarks. Similarly , as the use of global networking has increased the information flow and dependence upon our computing technology, Information System Managers have realized the need to protect their computing systems, networks, and information from damage and theft. Although there are several ways this can be achieved, the most prevalent is the use of a firewall .This chapter focuses on the increasing significance of computer networks and evolving network security issue in the world it also introduce concept of Deep packet Inspection.

1.1 Significance of Computer Networks Security

"Security against defeat implies defensive tactics; ability to defeat the enemy means taking the offensive." Sun Tzu [36]

Even though these words were written by a general who lived in China around 500 B.C., the words are still alive [1]. It is necessary to take the offensive action by installing strong defenses on the network , using all available tools and by studying security methodology to know where network is weak in security . If administrator take an best efforts to keep network and data secure with newest security measures , then there will be better chance of victory over the intruders and hackers , then administrator can deny victory of black hats .But it doesn't mean that it will restrict the black hat hackers completely. There is no hundred percent foolproof security, aside from turning off the system. And even that is not one hundred percent as someone could physically walk off with the device [3]. All administrator can hope to accomplish to force the black hats to go bother someone else who less prepared than him.

While using the Internet, along with the convenience and speed of access to information arises new risks. Among them are the risks that valuable information will be lost, stolen, corrupted, or misused and that the computer systems will be corrupted. If information is recorded electronically and is available on networked

Computer Science and Engineering Department TU Patiala

computers, it is more vulnerable than if the same information is printed on paper and locked in a file cabinet. Intruders do not need to enter an office or home, and may not even be in the same country. They can steal or tamper with information without touching a piece of paper or a photocopier [4].

1.1.1 Basic Internet security concepts.

The three basic security concepts important to information on the Internet are:

- Confidentiality.
- Integrity.
- Availability.

When information is read or copied by someone not authorized to do so, the result is known as loss of confidentiality. For some types of information, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications, and corporate investment strategies. In some locations, there may be a legal obligation to protect the privacy of individuals. This is particularly true for most banks and loan companies, debt collecting agencies, businesses that offer credit to their customers or issue credit cards, hospitals, doctors' offices, and medical testing laboratories, individuals or agencies that offer services such as psychological counseling or drug treatment and agencies that collect any form of taxes [8].

Information can be corrupted when it is available on an insecure network. When information is modified in unexpected ways, the result is known as loss of integrity. This means that unauthorized changes are made to information, whether by human error or intentional tampering. Integrity is particularly important for critical safety and financial data used for activities such as electronic funds transfers, air traffic control, and financial accounting [18].

Information can be erased or become inaccessible, resulting in loss of availability. This means that people who are authorized to get information cannot get what they need. Availability is often the most important attribute in service-oriented businesses that depend on information examples are airline schedules and online inventory

Computer Science and Engineering Department TU Patiala

systems. Availability of the network itself is important to anyone whose business or education relies on a network connection. When a user cannot get access to the network or specific services provided on the network, they experience a denial of service [8].

Authentication is proving that a user is whom he or she claims to be. That proof may involve something the user knows such as a password, something the user has such as a smartcard, or something about the user that proves the person's identity such as a fingerprint. Authorization is the act of determining whether a particular user has the right to carry out a certain activity, such as reading a file or running a program. Authentication and authorization go hand in hand. Users must be authenticated before carrying out the activity they are authorized to perform. Security is strong when the means of authentication cannot later be refuted the user cannot later deny that he or she performed the activity. This is known as no repudiation.

It is easy to gain unauthorized access to information in an insecure networked environment, and it is hard to catch the intruders. Even if users have nothing stored on their computer that they consider important, that computer can be a "weak link", allowing unauthorized access to the organization's systems and information. Seemingly innocuous information can expose a computer system to compromise. Information that intruders find useful includes which hardware and software are being used, system configuration, type of network connections, phone numbers, and access and authentication procedures. Security-related information can enable unauthorized individuals to get access to important files and programs, thus compromising the security of the whole system. Examples of important information are passwords, access control files and keys, personnel information, and encryption algorithms [3].

Internet security abuse is often reported in the media. Nobody on the Internet is fully or completely immune to a security breach. Those mostly affected are banks and financial companies, insurance companies, brokerage houses, consultants, government contractors, government agencies, hospitals and medical laboratories, network service providers [4].

Computer Science and Engineering Department TU Patiala

The threats in security cover a broad range of possibilities: a minor loss of time in recovering from the problem, a decrease in productivity, a loss of money, a loss of credibility or market opportunity, a business no longer able to compete, legal liability, and the loss of life.

Internal network security is very often underestimated by its administrators. In fact, in certain environments such security does not even exist, allowing one user to easily access another user's computer using well-known exploits, trust relationships and default settings. Most of these attacks require little or no skill, putting the integrity of a network at stake. A user within the company already has access to many internal resources without needing to bypass firewalls or other security mechanisms. In fact, these security measures are generally used to prevent non-trusted external sources, such as Internet users, from accessing the internal network. However, most threats come from internal users. An internal user, equipped with hacking skills, can successfully penetrate and achieve administrative network rights while ensuring that their abuse is hard to identify or even detect.

The Computer Crime and Security Survey compiled in 2003 by the Computer Security Institute and the FBI discovered that approximately 65% of respondents reported at least one security incident involving an insider. Poor network security may also allow malicious users that break into a network system to access the rest of the internal network more easily. This would enable a sophisticated attacker to read and possibly leak confidential emails and documents, delete data and damage computers leading to loss of important information and more [2]. While using the Internet, along with the convenience and speed of access to information come new risks. Among them are the risks that valuable information will be lost, stolen, corrupted, or misused and that the computer systems will be corrupted.

Computer Science and Engineering Department TU Patiala

1.2 Security Threats and Prevention Measures

1. Viruses and Worms

The term virus has long been used generically to describe any computer threat, but in actuality it refers to malware that inserts malicious code into existing documents or programs, and 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. 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 received email with an attachment from a friend that may contains a virus [13].

Prevention : Virus require a user action to insert themselves onto your computer. So, never open an email attachment that they weren't expecting, no matter who the sender is, this will keep network free of viruses. But this will do little to stop worms from infecting network. That is because although worms are also often initially delivered in email, they don't need a host file and they can propagate themselves. Worms, unlike viruses, spread on their own. So once a computer is infected, the worm can often make quick copies of itself and infect an entire network within a few hours. Because of this unique opportunity to multiply themselves quickly across a network, worms are responsible for a good number of companies' widespread network failures. Both viruses and worms often work to open up new holes in network security in order to allow even more dangerous security threats to infect network. It should be an essential priority of every company and individual to use virus protection software to limit the incoming malware, and then to educate employees to make sure those worms and viruses that slip through never get opened [13] .

2. Trojan Horses

A Trojan horse is a malware attack that disguises itself as something innocent, such as a computer game, or a YouTube search results page. As an example, Trojan horse used an email with a link that supposedly connected the reader to a video of the Saddam Hussein hanging, but instead just infected them with malware. Once installed on a computer, the 'Saddam' Trojan horse then downloaded and installed a keylogger

Computer Science and Engineering Department TU Patiala

onto the infected computer. This keylogger was used to record every keystroke by a computer's user, thus stealing financial account information and passwords Trojans are dangerous because they all appear so innocuous on the surface. Often trojans imbed themselves on a particular website hide in downloaded free software, a person might be infected by clicking on a link sent to them in email [13].

Prevention : As hackers are so creative in coming up with new and different types of Trojan horses everyday, administrator may want to consider blocking users from downloading freeware, blocking links imbedded in emails, and using a white list to create a list of approved websites that you may visit. As 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.

3. Spam

Spam email takes a variety of forms, ranging from unsolicited emails promoting products like Viagra, to coordinated spam attacks designed to take up so much bandwidth on a network so as to cause it to crash. A more recent trend is image spam, which eats up even more bandwidth than its textual cousin, and often circumvents contextual spam filters which analyze the message text to look for indications that the email is spam. Another technique that spammers are using is called "news service" spam, which uses legitimate headlines such as "Vaibhav you have Earns $83M Bonus" or "Lucky chance to win" to trick recipients into opening spam emails that are filled with spammed drug advertisements. These and other new spam trends constantly threaten the productivity of email and the security of IT networks [13] .

Prevention : When it comes to fighting spam, this can be filtered out by a good email filter. And much of what slips through can be avoided by staying current on the latest techniques that spammers use. In addition, however, administrator should protect your network from email spam by requiring his employees to use separate accounts for their personal internet use, and demand that accounts not be used to sign up for any online service or freebie.

Computer Science and Engineering Department TU Patiala

4. Phishing

Anyone who has ever used PayPal or does their banking online has probably received dozens of emails with titles such as, "URGENT: Update Account Status". These emails are all attempts by a spammer to "phish" your account information. Phishing refers to spam emails designed to trick recipients into clicking on a link to an insecure website. Typically, phishing attempts are executed to steal account information for e-commerce sites such as eBay, payments processors such as PayPal, or regular financial institutions' websites [24]. A phishing email supplies you with a link to click on, which will open a page where user can re-enter all your account details, including credit card number and/or passwords Some criminals are also using VoIP or VoIM software to send vishing messages. These try to confuse people into calling the provided number usually an automated VoIP Call In number outgoing call number and revealing credit card details, which are recorded in audio form [34].

Prevention : Phishing in all its varieties is a huge and growing problem for network security managers and business owners. As we all become more interconnected and access more and more personal information through networks, there become more and more opportunities for phishers to attack. To protect one's network, it is becoming increasingly vital that you educate users about the most common ways in which hackers try to phish account information. Even though simplistic phishing attempts like the PayPal scam now seem obvious to regular internet users, a single phishing attack can compromise an entire network's security if the user is tricked into giving his network account information [33].

5. Packet Sniffers

Packet sniffers capture data streams over a network, thus allowing for the capture of sensitive data like usernames, passwords and credit card numbers. The result, unsurprisingly, is the loss of data, trade secrets, or online account balances. For network managers specifically, even bigger losses can come from lawsuits due to noncompliance of data protection regulations. While Packet sniffers have been used in rather harmless ways, such as by law enforcement and by corporations for data protection compliance purposes the real concern for network owners is packet sniffers

Computer Science and Engineering Department TU Patiala

more malicious forms. Packet sniffers work by monitoring and recording all the information that comes from and goes to your computer over a compromised network. So in order to be effective, the packet sniffer must first have access to the network you are using. The most common way to do this, is through using something called honeypots. Honeypots are simply unsecured wifi access points that hackers setup and trap people into using them. These honeypots are setup in public places such as airports, and the wifi network is titled like "Free Public Wi-Fi". Unsuspecting individuals then sign onto the corrupted network and the packet sniffer then grabs their personal information when they enter things like their credit card info into a site [35] .

Prevention : Education is simply the best policy to deal with the threat of packet sniffers. Once user know to never access the internet through an unsecured connection, and are made aware of the fact that packet sniffers exist, they are much less likely to fall victim to this hacking technique. Because a single victim of packet sniffing among any user can compromise sensitive network data, it is important that everyone learn how to identify honeypots and how to secure their own home wifi networks. In addition, make sure that your employees use a variety of different sign on names and passwords to access various levels of network security. That way, if login information is compromised, the damage can at least be limited in scope [27] .

6. Maliciously-Coded Web sites

Maliciously-coded Web sites can take many different forms, from installing Trojan horses to redirecting you to an unrequested site. But one of the most threatening forms of maliciously-coded websites, those that are designed to steal passwords, are on the rise [4]. A very common form of these Web sites takes advantage of human's charitable instincts by setting up traps in what appear to be sites that allow you to make donations to victims of natural disasters such as Hurricane Katrina. Hackers set up a fake sign-in page, and then encourage unsuspecting victims to enter their credit card number and other personal information. In addition to stealing personal information, maliciously-coded websites are also often designed for the following purposes:

Computer Science and Engineering Department TU Patiala

- installation of keyloggers
- adware/ spyware/ reading cookies
- drive-by downloads
- XSS - cross--site scripting to utilize web browser flaws for other intentions.

Prevention : In order to protect network, administrator should encourage user to purchase information only from security certified sites, and to use PayPal instead of a credit card whenever possible, since by doing so they will not have to reveal their credit card information to another site. In addition to limiting the number of times credit card information is typed into a website, paying by PayPal is also helpful because maliciously-coded sites are less likely to accept PayPal payments since the owners of that PayPal account are easier to trace to an address or bank account [33].

7. Password Attacks

A 'Password Attack' is a general term that describes a variety of techniques used to steal passwords to accounts.

- Brute-force : One of the most labor intensive and unsophisticated methods hackers use to steal passwords is to try to guess a password by repeatedly entering in new combinations of words and phrases compiled from a dictionary. This 'dictionary attack' can also be used to try to guess usernames as well, so developing difficult to guess usernames and passwords is increasingly vital to network security.
- Packet sniffers : Packet Sniffers glean data electronically from a compromised network.
- IP-spoofing : Similar to 'Honeypots', this attack involves the interception of data packets by a computer successfully pretending to be a trusted server/ resource.
- Trojans : Trojans are actually invasive, and they are the most likely to be successful, especially if they install keyloggers.

Prevention : Automated testing ,human behavior , and other security flaws make it easier for password attackers to succeed. Unfortunately, there is no one single method to prevent against password attacks, though combining network traffic analysis along with the old stalwarts of email scanning, virus protection, firewalls and an educated work force can all together form a strong defense for any network [33] .

Computer Science and Engineering Department TU Patiala

1.3 Deep Packet Inspection

Deep packet inspection refers to the fact that it don't simply look at the header information as packets pass through them. Rather, they move beyond the IP and TCP header information to look at the payload of the packet. The goal is to identify the applications being used on the network, but some of these devices can go much further; those from a company like Narus, for instance, can look inside all traffic from a specific IP address, pick out the HTTP traffic, then drill even further down to capture only traffic headed to and from Gmail, and can even reassemble e-mails as they are typed out by the user [25].

1.3.1 DPI's potential uses

DPI technology is unique in that as of now it's the only way to accomplish certain governmental security directives. DPI also has the potential to do a great deal of good.

- Network security: DPI's ability to inspect data streams at such a granular level will prevent viruses and spyware from either gaining entrance to a network or leaving it.
- Network access: DPI create s conditions where network access rules are easy to enforce due to the deep inspection of packets.
- CALEA compliance: DPI technology augments traffic access points (TAP) technology used initially for governmental surveillance equipment.
- SLA enforcement: ISPs can use DPI to ensure that their acceptable use policy is enforced. For example, DPI can locate illegal content or abnormal bandwidth usage.
- QoS: P2P traffic gives ISPs a great deal of trouble. DPI would allow the ISP to instigate traffic control and bandwidth allocation [29].

Computer Science and Engineering Department TU Patiala

1.4 Current DPI Solutions

The various DPI solutions are based on signature matching or anomaly detection. Currently available popular DPI solutions include Snort, snort-Inline and BRO.

1.4.1 Snort

Snort is an open source IDS (Intrusion detection system) written by Martin Roesch. It was bought by the commercial company SourceFire which was bought itself by the FireWall Giant CheckPoint in 2005. Like Tcpdump, Snort uses the libpcap library to capture packets.

Snort can be runned in 4 modes:

- sniffer mode: snort will read the network traffic and print them to the screen.
- packet logger mode: snort will record the network traffic on a file
- IDS mode: network traffic matching security rules will be recorded
- IPS mode: also known as snort-inline (IPS = Intrusion prevention system)

Snort is a very powerful tool and is known to be one of the best IDS on the market even when compared to commercial IDS [28].

1.4.2 Bro

A Unix based network based Intrusion Detection System (IDS) developed by Vern Paxson at Lawrence Berkeley National Lab and the International Computer Science Institute. Bro analyzes network traffic against rules describing what sort of activity is deemed troublesome. These rules might describe restrictions on activity e.g., only certain hosts can connect to certain services, policies regarding what activity is worth alerting e.g., attempts to a given number of different hosts constitutes a "scan", or signatures describing known attacks or access to known vulnerabilities. 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

Computer Science and Engineering Department TU Patiala

activity that the rules indicate should generate alerts or actions, such as possible intrusions [12].

1.4.3 Snort-inline

The Snort_inline IPS is a modified version of the famous Snort IDS.It receives packets sent from the Netfilter firewall with the help of the lipipq library, compares them with Snort signature rules and tags them as drop if they match a rule, then finally sends them back to Netfilter where the Snort_Inline tagged packets are dropped. An IDS logs an alert when a packet matches a signature rule but does not discard or even modify it. This is different with an IPS where a packet matching a signature rule is blocked or modified. You must be extremely careful with the "false positive" alarms (packets matching a signature rule but being in fact harmless) on an IPS because this can hurt the good behavior of the communications between your systems by blocking required links for the business [28].

The solution available in the market are useful but there are some issues. The network are much more complex to handle using this solution . This solution require the deep understanding also the good knowledge of pattern matching and application layer protocol etc. The attackers are using more sophisticated and advanced methods of sending malicious information across the network e.g. ICMP tunneling, P2P, HTTPS tunneling using which the information to be sent is hidden under the shield of another legitimate protocol. Even if we open the packets and see the contents we will not be able to know what information is being passed in them. So, we need more comprehensible solution. Moreover, the Packet control that NIDS like Snort with snort-Inline patch provides is limited to DROP or PASS the packet as it travels through the path.

Computer Science and Engineering Department TU Patiala

Chapter 2

Literature Survey

2.1 TCP/IP Protocols

TCP/IP stands for Transmission Control Protocol/Internet Protocol, it is not just one or two protocols. TCP/IP is a suite of protocols**.** Like most network protocols, TCP/IP is a layered protocol. Each layer builds upon the layer below it, adding new functionality. The lowest level protocol is concerned purely with the business of sending and receiving data any data using specific network hardware. At the top are protocols designed specifically for tasks like transferring files or delivering email. In between are levels concerned with things like routing and reliability.The benefit that the layered protocol stack gives you is that, if you invent a new network application or a new type of hardware, you only need to create a protocol for that application or that hardware: you don't have to rewrite the whole stack**.**

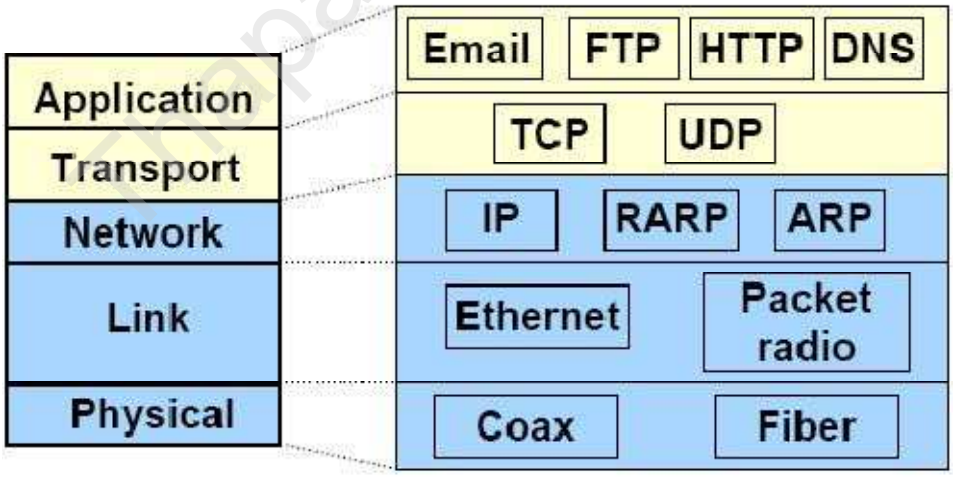


Figure 2.1: TCP –IP reference model [37].

Computer Science and Engineering Department TU Patiala

2.1.1 Internet Protocol

IP is the bedrock protocol of TCP/IP. Every message and every piece of data sent over any TCP/IP network is sent as an IP packet. IP's job is to enable data to be transmitted across and between networks. Hence the name internet protocol. In a small LAN, it adds little to what could be achieved if the network applications talked directly to Ethernet. If every computer is connected to the same Ethernet cable, every message could be sent directly to the destination computer. In connecting networks together, direct Ethernet communication becomes impractical. Application level address a message to a computer on the far side of the world, but Ethernet card can't communicate with the Ethernet card on that computer. Physical Ethernet limitations would prevent it, for a start. It would, in any case, be undesirable for every computer in the world to be connected to one big network. Every message sent would have to be heard by every computer, which would be bedlam. Instead, inter-net communications take place using one or more "hops". Ethernet card will communicate with another Ethernet device on the route to the final destination. Routing is the important capability that IP adds to a hardware network protocol. Before we come to it, we will look at some other features of IP. IP is a connectionless protocol. This means that it has no concept of a job or a session. Each packet is treated as an entity in itself. IP is rather like a postal worker sorting letters. IP is not concerned with whether a packet is one of a batch. IP simply routes packets, one at a time, to the next location on the delivery route. IP is also unconcerned with whether a packet reaches its eventual destination, or whether packets arrive in the original order. There is no information in a packet to identify it as part of a sequence or as belonging to a particular job. Consequently, IP cannot tell if packets were lost or whether they were received out of order. IP is an unreliable protocol. Any mechanisms for ensuring that data sent arrives correct and intact are provided by the higher-level protocols in the suite [37].

2.1.2 TCP

TCP receiving data from the Application layer and then handing it off to the Network layer. TCP creates segments or user datagrams by taking the information from the Application layer and adding a header to it. Since TCP is responsible for Port to Port

Computer Science and Engineering Department TU Patiala

addressing, it uses a 16bit process port to identify who it wants to talk to. This means that TCP deals with program to program, not machine to machine. It works by opening up a stream or virtual circuit between the two ports, which begins by alerting the receiver to expect information and ends by an explicit termination signal. Since every segment received is answered with an acknowledge, TCP is a reliable stream delivery service. This just means that the information is guaranteed to arrive, or an error will be returned [38].

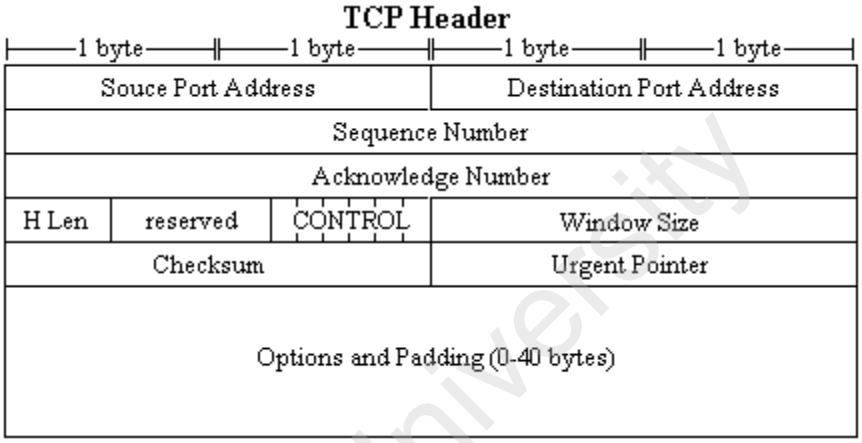


Figure 2.2: TCP Packet Header [21]

A description of each of the header components of the TCP segment and their size in bits follows:

Source Port Address (16)-	The address of the application that is generating the segment.
Destination Port Address (16)-	The address of the application that will receive the segment.
Sequence Number (32)-	Position of data in the original data stream (if it has been split).
Acknowledge Number (32)-	Acknowledges the acceptance of data from the other device.
H Len (4)-	Header Length. Number of 4 byte words used in header (0-60 bytes).
reserved (6)-	--reserved for future use--

Computer Science and Engineering Department TU Patiala

2.1.3 UDP

User Datagram Format (UDP) is simpler, faster and cheaper than TCP. UDP is connectionless and unreliable which means that it does not establish a virtual circuit like TCP, nor does it demand an acknowledge. It merely sends out the message. UDP headers are all 8 bytes, while TCP headers can be 20-60 bytes long. Like TCP, UDP provides delivery of segments using IP. Below is a diagram of the UDP header. UDP is used for things like FINGER and other programs which prioritize speed over reliability, and whose data is not worth the overhead that establishing a TCP connection demands. UDP is neither intelligent enough to compensate for congestion and receiver's speed, nor is it complex enough to care. There is no guarantee that the segments will arrive in order, or at all. It's also possible, if a long enough delay causes a resend, to receive duplicate segments [39].

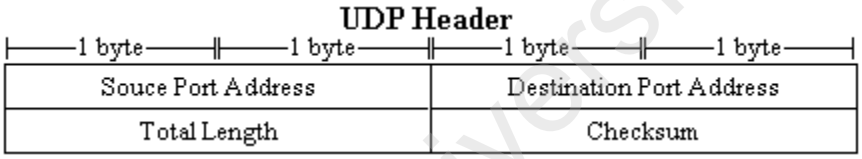


Figure 2.3: UDP Packet Header[21]

A description of each of the header components and their size in bits follows:

Source Port Address (16)-	The address of the application that is generating the user datagram.
Destination Port Address (16)-	The address of the application that will receive the user datagram.
Total Length (16)-	Total length of the user datagram in bytes
Checksum (16)-	Error Detection

Computer Science and Engineering Department TU Patiala

2.2 Linux Kernel Network Subsystem

The Linux kernel is the core of a large and complex operating system, and while it's huge, it is well organized in terms of subsystems and layers. GNU/Linux operating system architecture is as per below:

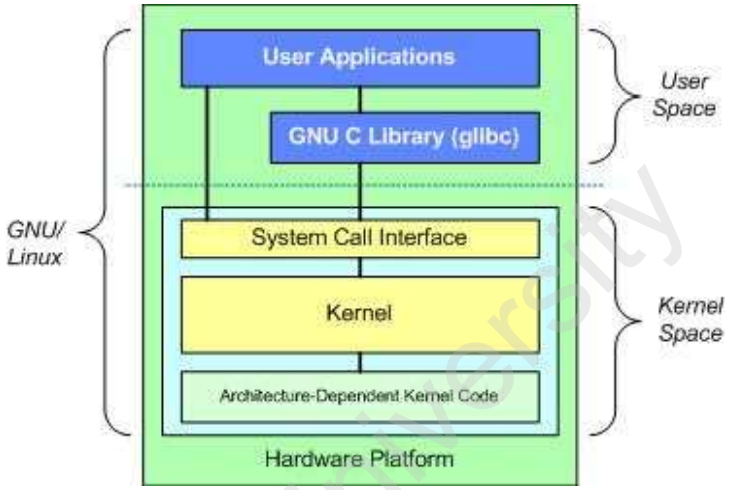


Figure 2.3 : The fundamental architecture of the GNU/Linux operating system [11].

Kernel of an operating system usually remains hidden beyond common user space pro- grams and system utilities and performs various critical duties like management of the memory, processes, file System, synchronization, and any other resources attached. In Linux, external kernel modules can be dynamically plugged into the core kernel. Device drivers can be compiled statically into the kernel or dynamically loaded later. Kernel comprises of various subsystems; the code is split into subsystems logically based on the function provided

Major subsystems of the Linux kernel:

Computer Science and Engineering Department TU Patiala

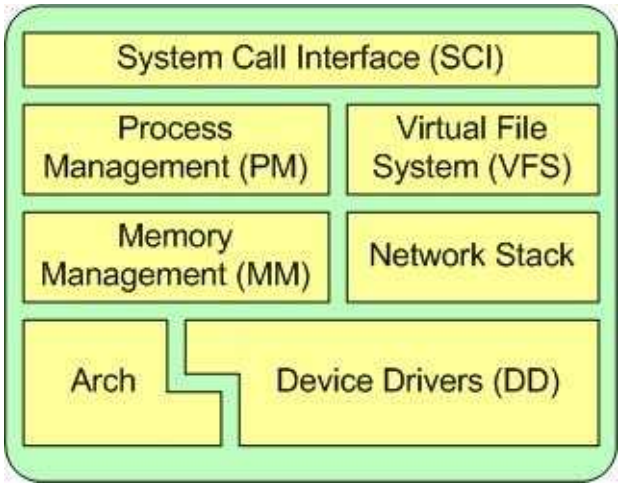


Figure 2.4 : Major Subsystem of Linux kernel [11].

The main concern in my thesis is of networking subsystem of the Linux kernel. The strength of the Linux kernel network subsystem lies in its routing, firewalling and shaping capabilities**.** The kernel starts with three lists of rules; these lists are called firewall chains or just chains. The three chains are called INPUT, OUTPUT and FORWARD When IP forwarding is enabled, the routing is done inside the kernel: packet arrives from one network interface adapter and kernel decides whether it belongs to thiscomputer. In TCP/IP networks, this is done by comparing the IP address, but there is also an exception: Network Address Translation. If the packet does not belong to this computer, it could be forwarded (based on the settings) to the other network. TCP/IP implementation relies on IP routing table, which is maintained from the user mode, but routing is done in the kernel. Packet filter decides what to do by looking inside the packet and to the known state of the connection. These decision rules are stored in chains; packet traverses the chain from beginning until it is matched by some rule or list finishes. Common targets are ACCEPT, REJECT or DROP. Packet filter implementation in Linux is called Netfilter which we will see in the next few sections. The two most important data structure of Linux kernel network layer are sk buff (defined in include/linux/skbuff.h) and netdevice (defined in include/linux/netdevice.h) [11].

Computer Science and Engineering Department TU Patiala

2.3 Main Data Structures

The networking part of the kernel uses mainly two data structures. One to keep the state of a connection called sock and other to keep the status of each packet called sk_buff.

2.3.1 Skbuff Data Structure

Skbuffs are the buffers in which the Linux kernel handles network packets. The packet is received by the network card, put into a variable (normally, skb) of sk buff type and then passed to the network stack, which uses the skb all the time. The skb's contain pointer and length fields that allow each protocol layer to manipulate the application data via standard functions. Changing packet fields is achieved by changing its fields. The first fields are general ones. A pointer to the next and previous skbs in the list (packets are frequently put in lists or queues). The socket that owns the packet is stored in sk (note that if the packet is arriving from the network only at a later stage the socket owner is known). The time of arrival is stored in tstamp. The dev field stores the device the packet arrived and when and if the device to be used for transmission is known (for example by inspection of the routing table) the dev field is updated correspondingly. The following is a snippet from the skbuff.h which shows the partial list of fields contained in the sk buff structure [32].

```
struct sk_buff { struct sk_buff *next;
 struct sk_buff *prev;
 struct sock *sk;
ktime_t tstamp;
struct net_device *dev;
int iif; struct dst_entry *dst;
 struct sec_path *sp; char cb[48];
unsigned int len,
          data_len,
         mac_len;


union { __wsum csum; struct
```

Computer Science and Engineering Department TU Patiala

```
        {
         __u16      csum_start;
          __u16     csum_offset;
         }; };

           __u32          priority;

           __u8           local_df:1,
cloned:1,
ip_summed:2,
nohdr:1, nfctinfo:3;
__u8 pkt_type:3,
fclone:2,
ipvs_property:1;
__be16 protocol;
 __u32 mark;
sk_buff_data_t transport_header;
 sk_buff_data_t network_header;
sk_buff_data_t mac_header;
sk_buff_data_t tail;
 sk_buff_data_t end;
unsigned char *head, *data;
 unsigned int truesize;
atomic_t users;
};
```

The *dst member is the generic route for the packet. It tells us how to get the packet to it's destination. We store the security path traversed by the packet, if any in the *sp field; for example, on input IPSEC records each transformation which has been applied to the packet by a decapsulator. The array cb is the control buffer. It is free to use for every layer. All the private variables are put into this buffer. If we want to keep these variables across layers we have to do a skb clone() first. The three length members are pretty straight-forward. The total number of bytes in the packet is 'len'. SKBs are composed of a linear data buffer, and optionally a set of 1 or more page buffers. If there are page buffers, the total number of bytes in the page buffer area is

Computer Science and Engineering Department TU Patiala

'data len'. Therefore the number of bytes in the linear buffer is 'len - data len'. There is a shorthand function for this in 'skb headlen()'. The 'mac len' holds the length of the MAC header. Finally, 'csum' holds the checksum of the packet. When building send packets, we copy the data in from userspace and calculate the 16-bit two's complement sum in parallel for performance. This sum is accumulated in 'csum'. This helps us compute the final checksum stored in the protocol packet header checksum field. This field can end up being ignored if, for example, the device will checksum the packet for us. The 'priority' field is used in the implement of QoS. The packet's value of this field can be determined by, for example, the TOS field setting in the IPV4 header. Then, the packet scheduler and classifier layer can key off of this SKB priority value to schedule or classify the packet, as configured by the administrator. The 'local df' field is used by the IPV4 protocol, and when set allows us to locally fragment frames which have already been fragmented. This situation can arise, for example, with IPSEC. In order to make quick references to SKB data, Linux has the concept of SKB clones. When a clone of an SKB is made, all of the 'struct sk buff' structure members of the clone are private to the clone. The data, however, is shared between the primary SKB and it's clone. When an SKB is cloned, the 'cloned' field will be set in both the primary and clone SKB. Otherwise is will be zero [32].

2.3.2 Sock Data Structure

The sock data structure keeps the state of a specific connection. When a socket is created in user space a sock structure is allocated.

The fields daddrs , dport, rec_saddr contain source and destination addresses and ports.

```
struct sock {
 __u32 daddr; /* Foreign IPv4 addr */
 __u32 rcv_saddr; /* Bound local IPv4 addr */
 __u16 dport; /* Destination port */
unsigned short num; /* Local port */ int bound };
```

Computer Science and Engineering Department TU Patiala

Among many fields the sock structure contains protocol specific information.

```
union {
 struct ipv6_pinfo af_inet6;
 }
net_pinfo; union
{
struct tcp_opt af_tcp;
 struct raw_opt tp_raw4;
struct raw6_opt tp_raw;
struct spx_opt af_spx;
} tp_pinfo;
};
```

2.4 Journey of a packet through Linux Kernel

In this section we will see the paths through the kernel followed by IP packets when they are received or transmitted from a host.

2.4.1 NAPI

NAPI stands for New Application Programming Interface .NAPI is a modification to the device driver packet processing framework, which is designed to improve the performance of high-speed networking. NAPI works through:

Interrupt mitigation

High-speed networking can create thousands of interrupts per second, all of which tell the system something it already knew: it has lots of packets to process. NAPI allows drivers to run with (some) interrupts disabled during times of high traffic, with a corresponding decrease in system load.

Packet throttling

When the system is overwhelmed and must drop packets, it's better if those packets are disposed of before much effort goes into processing them. NAPI-compliant drivers can often cause packets to be dropped in the network adaptor itself, before the kernel sees them at all.

Computer Science and Engineering Department TU Patiala

NAPI was first incorporated in the 2.5/2.6 kernel but was also backported to the 2.4.20 kernel.

When the system boots up, the network device is by default in the closed state. As soon as the device is administratively brought up, it moves into the poll off state. In this state, the network device has its interrupts enabled. An arriving packet causes the receive interrupt status to be enabled and the interrupt handler is invoked. The interrupt handler reads the status registers to find out what caused interrupts. It then disables the interrupts, schedules for the device to be polled, and moves the device into poll on state. During this state, incoming packets are deposited onto the DMA receive ring upon arrival. In the poll on state, Linux is aware that the device has packets that typically are sitting at the DMA ring, waiting to be processed. If the CPU is busy elsewhere during the poll on state, arriving packets continue to get deposited at the DMA ring without bothering the CPU from what it is doing. When the ring fills up, any new incoming packets are dropped. When the system is ready, it schedules the driver to enter the poll receive state. In this state, the driver is told the maximum number of packets-known as the budget-it can push up to the stack. The design of NAPI is to treat all network devices fairly and to this end the Deficit Round Robin algorithm (DRR) is used to give equal opportunity to push packets up the stack. In poll receive state, the driver removes a packet from the ring pushes it up the stack where it is processed to completion [41]. As an example, if the packet is to be forwarded, it will go all the way to be deposited either on the DMA ring of the egress device or the schedule queue for the egress device. The driver will push up to a maximum number of packets that is asked of it as long as it does not exceed its allocated budget. If at the end of its run the network interface has no more packets on its ring, it is moved to the poll off state and interrupts are re-enabled (an IO write). If, on the other hand, it still had packets left, it is moved back to the poll on state and as a result rescheduled for a future poll. It should be noted that when the system processes packets, more space becomes available for new packets. This is a very desirable feature because it allows the CPU to process packets proportional to its processing capacity and runtime load. The network device oscillates between poll receive and poll on states if there is a heavy network load. When the system perpetually oscillates between these two states, it is because it has reached its maximum processing capacity. This capacity is referred to as the Maximum Loss Free Rate (MLFR). In the

Computer Science and Engineering Department TU Patiala

poll receive state, it is possible to go into the OOM state, if the driver is unable to allocate a buffer to send up the stack. The driver starts a timer in the OOM state. When the timer expires, the driver retries to allocate a buffer. Upon success, the driver will reschedule itself and move to the poll on state. After a maximum number of retry failures, the driver bails out and enters the Reset state. In the Reset state, the driver frees all its resources, such as buffers stuck in either transmit or receive rings. On completion of a reset, the driver moves itself onto the poll off state. It should be noted that most drivers do not implement the OOM and Reset state8 and, in fact, go back to the poll on state from poll receive state upon failure to allocate buffers [32].

2.4.2 Journey of packet in Linux Kernel 2.6

The reception and transmission of the frames from/to the medium are handled by the code mostly in these files:

- include/linux/netdevice.h
- net/core/skbuff.c
- net/core/dev.c
- arch/i386/irq.c
- drivers/net/net-init.c
- net/sched/sch generic.c

When the driver of Network Interface card (NIC) is loaded, it sets up the packet descrip- tors and organizes them as ring buffers. The ring buffers are arrays of skbuffs which are allocated on reception of frames and deallocated on transmission of frames. The ring buffer for transmission of packets is tx ring and for reception is rx ring.

When a packet is received the following steps are followed:

- Packets are first received by the card. They are put in the rx_ring using DMA for recent cards. The rx_ring is a ring in the kernel memory where the card DMAs the incoming packets for the driver.

- The card interrupts the CPU, which then jumps to the driver ISR code, which called netif_rx_schedule() function which is inside include/linux/netdevice.h. This function puts a reference of the device (NIC) which received this packet, into a queue attatched to the interrupted CPU known as poll list.

Computer Science and Engineering Department TU Patiala

-For NAPI, the CPU polls the devices present in his poll_list to get all the received packets from their rx_ring. The poll method of any device calls, for each received packet, netif_receive_skb()which roughly calls ip_rcv().

When a packet is created the following steps are followed*:*

- All the IP packets are built using the arp_constructor() method. Each packet contains a dst field, that provides the destination computed by the routing algorithm. The dst field provides an output method, which is dev_queue_xmit() for IP packets;
- For each packet to transmit from the IP layer, the dev_queue_xmit() procedure (net/core/dev.c) is called. It queues a packet in the qdisc associated to the output interface. Then, if the device is not stopped (link failure, tx_ring full), all packets present in the qdisc are handled by qdisc_restart() (net/sched/sch_generic.c);
- The hard_start_xmit() virtual method is then called. This method is implemented in the driver code. The packet is placed in the tx_ring and the to schedule a softirq for later deallocating the memory associated with these packets.driver tells the card there are some packets to send;
- Once the card has sent a packet or a group of packets, it communicates to the CPU that packets have been sent out. The CPU uses this information (net_tx_action(), net/core/dev.c) to put the packets into a completion_queue [40].

2. 5 IPTABLES

Iptables is the native packet filtering mechanism for the 2.4 kernel series. We can use it to filter packets, implement network address translation and mangle packets. There are three default chains for filtering packets: INPUT, OUTPUT and FORWARD, and two default chains for network address translation: PREROUTING and POSTROUTING, and three tables: filter, nat and mangle. You can add your own chains.

2.5.1 Filter Table

Filter table is used to filter packets, which looks at the header of packets as they pass through, and decides the fate of the entire packet. It might decide to DROP the packet, ACCEPT the packet, or something more complicated. The iptables tool inserts and deletes rules from the kernel's packet filtering table. The kernel starts with three lists

Computer Science and Engineering Department TU Patiala

of rules in the `filter' table,these lists are called firewall chains or just chains. The three chains are called INPUT, OUTPUT and FORWARD.

- When a packet comes in (say, through the Ethernet card) the kernel first looks at the destination of the packet: this is called `routing'.
- If it's destined for this box, the packet passes to the INPUT chain. If it passes this, any processes waiting for that packet will receive it.
- Otherwise, if the kernel does not have forwarding enabled, or it doesn't know how to forward the packet, the packet is dropped. If forwarding is enabled, and the packet is destined for another network interface (if you have another one), then the packet goes to the FORWARD chain. If it is ACCEPTed, it will be sent out.
- Finally, a program running on the box can send network packets. These packets pass through the OUTPUT chain immediately: if it says ACCEPT, then the packet continues out to whatever interface it is destined for [10].

2.5.2 NAT Table

The table of NAT rules contains three lists called `chains': each rule is examined in order until one matches. The two chains are called PREROUTING (for Destination NAT, as packets first come in), and POSTROUTING (for Source NAT, as packets leave). The third (OUTPUT) will be ignored here.

The following diagram would illustrate NAT :

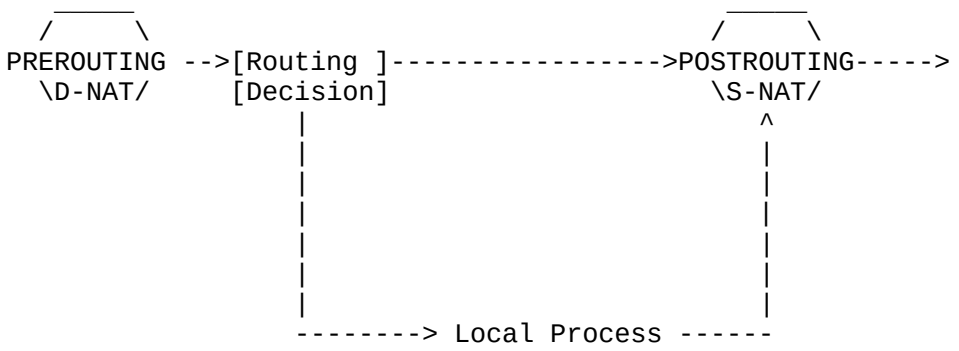


Figure 2.5: NAT Process [9] .

Computer Science and Engineering Department TU Patiala

NAT is divided into two different types: Source NAT (SNAT) and Destination NAT (DNAT).

Source NAT is when you alter the source address of the first packet: i.e. we are changing where the connection is coming from. Source NAT is always done postrouting, just before the packet goes out onto the wire. Destination NAT is when you alter the destination address of the first packet [9].

2.5.3 Mangle Table

Netfilter provides more means to manipulate packets. The built-in mangle table can be used in many ways to control and even alter packets passing through our packet filter. For the time being, we will pick the FORWARD chain of the mangle table, which is not only different from the FORWARD chain of the filter table, but also gets examined before. Later on we will discuss how to setup the assignments to our quality of service policy in the POSTROUTING chain of the mangle table.

Mangle table is used for mangling packets. There are three targets in this table: TOS, TTL and MARK. The TOS target is used to set and/or change the Type of Service field in the packet. The TTL target is used to change the TTL (Time To Live) field of the packet. The MARK target is used to set special mark values to the packet [7].

2.6 Packet Inspection Techniques

2.6.1 Packet Filter Firewalls

Fundamental type of firewall is called a packet filter. Packet filter firewalls are routing devices that include access control functionality for system addresses and communication sessions. The access control functionality of a packet filter firewall is governed by a set of directives collectively referred to as a rule. In their most basic form, packet filters operate at Layer 3 (Network) of the OSI model. This basic functionality is designed to provide network access control based upon several pieces of information contained in a network packet

Computer Science and Engineering Department TU Patiala

2.6.2 Application-Proxy Gateway Firewalls

Application-Proxy Gateway firewalls are advanced firewalls that combine lower layer access control with upper layer (Layer 7 – Application Layer) functionality. Application-proxy gateway firewalls do not require a Layer 3 (Network Layer) route between the inside and outside interfaces of the firewall; the firewall software performs the routing. In the event the application-proxy gateway software ceases to function, the fire-wall system is unable to pass network packets through the firewall system. All network packets that traverse the firewall must do so under software (application-proxy) control [8].

2.6.3 Dedicated Proxy Servers

Dedicated proxy servers differ from application-proxy gateway firewalls in that they retain proxy control of traffic but they do not contain firewall capability. They are typically deployed behind traditional firewall platforms for this reason. In typical use, a main firewall might accept inbound traffic, determine which application is being targeted, and then hand off the traffic to the appropriate proxy server, e.g., an email proxy server. The proxy server typically would perform filtering or logging operations on the traffic and then for-ward it to internal systems. A proxy server could also accept outbound traffic directly from internal systems, filter or log the traffic, and then pass it to the firewall for outbound delivery. An example of this would be an HTTP proxy deployed behind the firewall; users would need to connect to this proxy en route to connecting to external web servers. Typically, dedicated proxy servers are used to decrease the work load on the firewall and to perform more specialized filtering and logging that otherwise might be difficult to per-form on the firewall itself [20].

2.6.4 Hybrid Firewall Technologies

Nearly all major firewall vendors have introduced hybridization into their products in some way, shape, or form, so it is not always a simple matter to decide which specific firewall product is the most suitable for a given application or enterprise infrastructure. Hybridization of firewall platforms makes the pre-purchase product

Computer Science and Engineering Department TU Patiala

evaluation phase of a firewall project important. Supported feature sets, rather than firewall product classifica-tion, should drive the product selection [20].

2.6.5 Shallow Packet Inspection Firewall

In the 1990's, Stateful Inspection became the industry standard for Firewalls to address protect against undesired access and other malicious behaviour including protection against low-level DoS attacks. As well as examining header information, Stateful Inspection can examine the contents of a packet to determine more context about the packet beyond its source and destination information. In addition, Stateful Inspection monitors the state of a connection and compiles historic information in a state table. As a result, dynamic filtering decisions can be expanded beyond administrator-defined rules that simply block known IP addresses or TCP ports to take into account the context of a packet that has been established by packets that passed through the firewall earlier. In essence, stateful inspection firewalls add Layer 4 awareness to the standard packet filter architecture. Stateful inspection firewalls share the strengths and weaknesses of packet filter firewalls, but due to the state table implementation, stateful inspection firewalls are generally considered to be more secure than packet filter firewalls [27].

Stateful inspection uses information that utilizes layers 3-7 of the OSI model to obtain knowledge such as allowing traffic sessions to pass over specific ports dynamically. By combining information from various layers ,an IPS is able to better understand the protocol that it is inspecting and make more intelligent decisions on whether packets are legitimate or not.

2.6.6.Deep Packet Inspection in Firewall

Deep packet inspection (or DPI) is a powerful way to protect not just SIP traffic, but also the network. DPI is a form of computer network packet filtering that examines the data (or datagram) and UDP/TCP header part of a packet as it passes through an Ingate SIParator or Firewall. The Ingate is searching for non-protocol compliance, viruses, spam, intrusions or predefined criteria to decide if the packet can pass or if it needs to be routed to a different destination, or for the purpose of collecting statistical information. This is in contrast to shallow packet inspection (usually called just

Computer Science and Engineering Department TU Patiala

packet inspection) which only checks the UDP/TCP header portion of a packet. Deep Packet Inspection is packet filtering that involves examination of data further intothe packet than a 'traditional' firewall would look. Using DPI we look for data that might be indication of viruses or intrusion attempts, for instance. The line between the firewall and IDS becomes blurred in Linux iptables where many modules exist that implement functionality that is not normally associated with a firewall. This brings up the consideration of what unctions to leave to the IDS, and which might be moved to the firewall Looking deep into the packet, iptables can decide what to do with it, depending on its contents. Whereas an IDS might notice the packet and log it, iptables can react to it by dropping it, forwarding it, sending it to a tarpit, etc.

We can, for instance, tell the difference between HTTP traffic on port 80, and other traffic that is using that port (peer-to-peer, VoIP, etc.) We may be able to tell the difference between HTTP traffic that we want (website traffic), and HTTP traffic that we don't want (XML-RPC, SOAP)

DPI's potential uses : DPI technology is unique in that as of now it's the only way to accomplish certain governmental security directives. DPI also has the potential to do a great deal of good. For example, DDoS attacks are virtually impossible to thwart. Conceivably if DPI were in place and configured correctly it would detect the DDoS packets and filter them out. Some more potential uses are listed below:

- Network security: DPI's ability to inspect data streams at such a granular level will prevent viruses and spyware from either gaining entrance to a network or leaving it.
- Network access: DPI creates conditions where network access rules are easy to enforce due to the deep inspection of packets.
- CALEA compliance: DPI technology augments traffic access points (TAP) technology used initially for governmental surveillance equipment.
- SLA enforcement: ISPs can use DPI to ensure that their acceptable use policy is enforced. For example, DPI can locate illegal content or abnormal bandwidth usage.

Computer Science and Engineering Department TU Patiala

- QoS: P2P traffic gives ISPs a great deal of trouble. DPI would allow the ISP to instigate traffic control and bandwidth allocation.

- Tailored service: DPI allows ISPs to create different services plans, which means users would pay for a certain amount of bandwidth and traffic priority. This one is controversial and affects Net Neutrality.

- DRM enforcement: DPI has the ability to filter traffic to remove copyrighted material. There's immense pressure from the music and movie industries to make ISPs responsible for curtailing illegal distribution of copyrighted material [42].

2.7 Packet Filtering Option in Linux

Packet control refers to what can be done with packets entering and leaving the system. The stress is given on "through the system" because Linux is being looked here from the point of a networking device operating system rather than a host operating system. A networking device's main function is processing packets on the network which are mostly not meant for the device itself, but other hosts on the network. Sometimes, the hosts on the network are not even aware of the presence of the networking device. The following explain the various well-known packet control primitives:

Routing

Linux based routers are pretty common and easy to build too. Routing refers to taking packets from one network and putting them on another one towards the cause of making the packet reach its destination. A routing table is at the heart of such a decision mak- ing. Linux offers a very advanced routing infrastructure. In some cases even better than some of the popular specialized network devices. A lot of this functionality is available in the kernel. Some of the more advanced routing protocols may be instantiated using open-source software. An important point to note is that the routing operation modifies the packets. Thus routing is generally less efficient than other packet control methods

Computer Science and Engineering Department TU Patiala

Bridging

Bridging works on layer 2 as compared to routing which works on layer 3. Bridging packets is in some ways the same as routing except that in bridging, packets are not modified. Another requirement for bridging is that networks joined using bridging are on the same subnet.

Firewalling

Firewalling refers to the basic operations of letting either through the device (on paths determined by other parts of the system like routing, bridging etc.) or dropping them based on some known rules. A set of such rules is known as firewall policy. Linux implements firewalling using iptables. iptables tools are available on all common Linux systems. Firewalling is a very common and important operation in what we are trying to accomplish.

Quality of Service (QoS)

Sometimes dropping a packet or passing it is not enough. You may want to control what gets queued and how. Additionally you may want to prioritize some traffic over others. Metrics like bandwidth, throughput and latency etc. are important in context of a network. You may also want to control how much of these is allocated/permissible to a particular set of traffic. QoS infrastructure lets you do all of that. It is also referred to as Traffic Control (TC) or Traffic Control Next Generation (TCNG) in the Linux world. Packet Modification Packet modification or mangling is an interesting concept. By modifying the packets as they are passing through the device, one could solve interesting problems.

2.8 **Firewall**

Network firewalls are devices or systems that control the flow of network traffic between networks employing differing security postures. In most modern applications, firewalls and firewall environments are discussed in the context of Internet connectivity and the TCP/IP protocol suite. However, firewalls have applicability in network environments that do not include or require Internet connectivity. For example, many corporate enter-prise networks employ firewalls to restrict connectivity to and from internal networks ser-vicing more sensitive functions, such as the accounting or personnel department. By employing firewalls to

Computer Science and Engineering Department TU Patiala

control connectivity to these areas, an organization can prevent unauthorized access to the respective systems and resources within the more sensitive areas. The inclusion of a proper firewall or firewall environment can therefore provide an additional layer of security that would not otherwise be available [20]. There are several types of firewall platforms currently available from vendors. One way of comparing the capabilities of the firewall platforms is by examining the aspects of the Open Systems Interconnect (OSI) model that each given firewall platform is able to function with and can make use of. The OSI model is an abstraction of network communications between computer systems and network devices. The exact details of the OSI model are outside the scope of this document, but those layers relevant to the firewall topic are addressed. Layer 1 represents the actual physical communication hardware and media such as Ethernet. Layer 2 represents the layer at which network traffic delivery on Local Area Networks (LANs) occurs. Layer 2 is also the first layer that contains addressing that can identify a single specific machine. The addresses are assigned to network interfaces and are referred to as MAC, or Media Access Control addresses. An Ethernet address belonging to an Ethernet card is an example of a Layer 2 MAC address [22].

Layer 3 is the layer that accomplishes delivery of network traffic on Wide Area Networks (WANs). On the Internet, Layer 3 addresses are referred to as Internet Protocol (IP) ad-dresses; the addresses are normally unique but in circumstances involving Network Ad-dress Translation (NAT), it is possible that multiple physical systems are represented by a single Layer 3 IP address. Layer 4 identifies specific network applications and communi-cation sessions as opposed to network addresses; a system may have any number of Layer 4 sessions with other systems on the same network. Terminology associated with the TCP/IP protocol suite includes the notion of ports, which can be viewed as end points for sessions: a source port number identifies the communication session on the originating system; a destination port identifies the communication session of the destination system. The upper layers (5, 6, and 7) representing end-user applications and systems.

Computer Science and Engineering Department TU Patiala

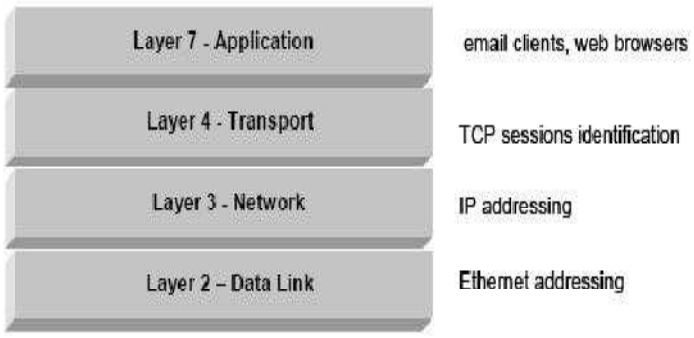


Figure 2.6 : Firewall operational layer [25]

Basic firewalls will operate on a smaller number of layers; more advanced firewalls will cover a larger number of layers. In terms of functionality, firewalls capable of examining a larger number of layers are more thorough and effective. Additional layer coverage also increases the configuration granularity present in the firewall; adding layer awareness al-lows the firewall to accommodate advanced applications and protocols. Increasing the layers a firewall can examine also allows the firewall to provide services that are very user-oriented, such as user authentication. A firewall that function with layers 2 and 3 only does not usually deal with specific users, but a higher end application-proxy gateway firewall can enforce user authentication as well as logging events to specific users.

Independent of firewall architecture, there can be many add-on services. Some of these services include Network Address Translation (NAT), Dynamic Host Configuration Pro-tocol (DHCP), encryption functionality such as Virtual Private Networks (VPNs), and application content filtering [7].

Computer Science and Engineering Department TU Patiala

Chapter 3

Problem statement

3.1 The Current Picture

Traditionally the security of network is managed by an device such as:

- Switches
- Routers
- Firewalls
- Intrustion prevention system
- Intrustion detection system

Switches operate at layer 2 or layer 3 and it offer very basic security features. They provide some access control rules , through which they decide which services should be offered or which user should be permitted to enter in the network. As they provide Access control list through which would administrator control what goes through the network and what doesn't based on the layer 2 or 3 identification of hosts. Also the working of router is based on the same principal of acces control list. Both of this may have an capability to provide other mechanisms like Virtual LAN based authentication using an external server, but again those apply to only a few special scenarios. Though they provide such an capability they don't understand the concept of a flow which is required for a end-to-end host communication. Firewalls have such capability to understand the concept of flows and provide an control on them by either specifying the IP address or the TCP or UDP port information. They also understand the concept of subnets and provide an capability to group users. It's easy to build rules in firewalls that block a known set of IP addresses and TCP/UDP ports, allow only outbound connections and do a bit of stateful analysis. As example allowing only outbound FTP .

IDS/IPS systems go for layers 3, 4 and higher. IDS, analyze network traffic and generate alerts when malicious activity is discovered. They are generally able to reset TCP connections by issuing specially crafted packets after an attack begins and some are even able to interface with firewall systems to re-write firewall rulesets onthefly. The limitation of Intrusion Detection Systems is that they cannot preempt network attacks because IDS sensors are based on packet sniffing technologies that only watch

Computer Science and Engineering Department TU Patiala

network traffic as it passes by. Intrusion Prevention Systems, IPS, perform the same analysis as Intrusion Detection Systems but, because they are inserted in-line, between other network components, they can pre-empt malicious activity. In contrast to IDS sensors, network traffic flows through an IPS sensor not past it so the IPS sensor can pull or drop traffic from the wire.

3.2 The Missing Part

The current packet filter solutions available in the market such as intrusion detection system work on the network and above layer . They detect the intrusions using the services available on this layer such as port number , ip addresses etc. , 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.Most of the packet control solution doesn't support all the packet filter options that are available in the IP tables .

Many of the solutions available in the market implement the firewall in the linux kernel . They implement the API in the user space and force to add the rule in the kernel space from user space , it make the kernel vulnerable . This implementation is also not as per the standardisation rule.

3.3 The Solution

The solution to above problem can be a DPI firewall which will present the user with the most flexible way of specifying "interesting" traffic, where the filtering rules will not just be based on 5-tuple (Source IP, Source Port, Destination IP, Destination Port, Control code) or parts thereof but will have an extended capability to accept better policies with rules that cover application layer. The range of possibilities in application layers is virtually limitless. A firewall with such a support would either cover only a limited functionality or would never be in a finished state. Secondly, the proposed solution would provide the maximum possible number of packet control options. Even new options can be developed by combining the ones already ex- isting. A consistent kernel API is to be developed and used at the core of a user space administrative part of the proposed kind of DPI firewall. The API would be used by most of the security applications without modification, thus ensuring consistency. Moreover, rather than directly implementing the firewall in kernel and pushing the

Computer Science and Engineering Department TU Patiala

policy from the user-space directly into the kernel thereby making the system vulnerable to bugs in the kernel firewall code, the vendors would architect their system differently. They will make use of this new standard API which will have a plethora of possibilities of different criteria to form the rules and would allow the customization (not actually customization but not even less than that because of numerous rule formation options) from the user space only, thus avoiding the possible vulnerability.

3.4 Objective

The goal of this thesis work is to study and understand the packet control mechanisms and then design and develop an Application protocol Interface which is capable of enforce packet control policies from the user-space. The following considerations are to be taken into account:

• The API should allow exhaustive control Most kinds of the packet control mechanisms should be supported

• The API should be easy to use. A variety of applications should be able to use it without substantial changes thus providing consistency.

3.5 The Platform

The setup used to carry out this thesis work involved Fedora Core 6 with Kernel 2.6.22. The system required two interfaces so as to make the system a routing device or IP forwarder. Suffice it to say that such solutions would generally not be available as ASIC or other hardware based systems, the power and flexibility of software has been used in this work to achieve goals.

Computer Science and Engineering Department TU Patiala

Chapter 4

Design and Solution

4.1 Conceptual architecture of the proposed solution

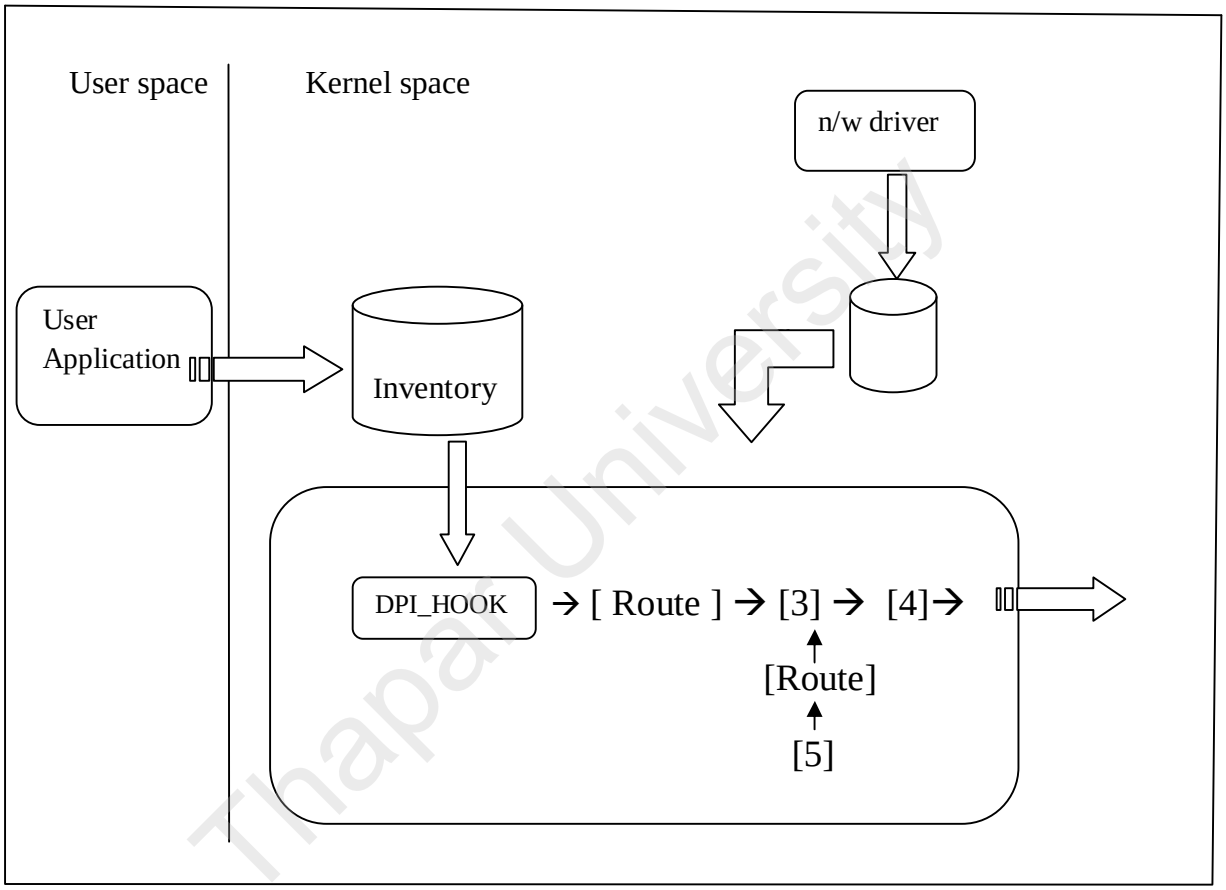


Figure 4.1 : The Conceptual architecture of the solution

1=NF_IP_PRE_ROUTING

2=NF_IP_LOCAL_IN

3=NF_IP_FORWARD

4=NF_IP_POST_ROUTING

The proposed API is to be implemented consist of three major part .

1. User Application Programme
2. Rule Inventory
3. Packet Inspection Hook

Computer Science and Engineering Department TU Patiala

The kernel part of the desired product which accept rules from a user space program is contain an ioctl function call which act like a communicator between the user space and the kernel space. Second part is an rule inventory which is an use to store the iptables rule which is used to compare all the incoming packets headers and payload for inspection . netfilter module will read the rules and match them against the packets passing through the network subsystem and third part which will use the stored rules . Now this is the point where capability of the Linux network subsystem can be harnessed and its true value can be realized. Figure 4.1 explains the architecture.

4.1.1 User Application Programme

The user space program is simple module which use the ioctl system call to communicate with the kernel . The user space program when requires issuing a new filter rule, it is made to call the kernel API and results in ioctl Module being invoked, accepting the rule, and getting it stored in a rule repository

4.1.2 Rule Inventory

This is an repository of which store the rules. All the rules are stored in the repository as they are accepted by the ioctl function . It will not be responsible for storing the rules permanently between the system-down-events. That job is assumed to be handled by the application program itself in the user space.

4.1.3 Packet Filter Hook

Packet filter hook is an modules which is placed in the NF_IP_PRE_ROUTING module of an netfilter . This module sits in between the path of the packet through the network subsystem of kernel. Each forwarded packet, is passes through the netfilter. As packet filter hook is palced in Netfilter packet passing through the netfilter also passess through the user defined hook , it matches its parameters with those stored inside each of the rules. If a rule matches, the specified target action given in the rule is taken with the packet otherwise the packet continues its normal journey through the kernel.

Computer Science and Engineering Department TU Patiala

4.2 Developing Linux Kernel Modules

Linux operates in two modes the Kernel mode (kernel space) and the User mode (user space). The kernel works in the highest level (also called supervisor mode) where it has all the authority, while the applications work in the lowest level where direct access to hardware and memory are prohibited. Linux transfers the execution from user space to the kernel space through system calls and the hardware interrupts. The Kernel code executing the system call works in the context of the process, which invokes the system call. As it operates on behalf of the calling process, it can access the data in the processes address space. The kernel code that handles interrupts, works to the processes and related to any particular process.

4.2.1 Linux Kernel Modules

The Linux kernel is a monolithic kernel i.e. it is one single large program where all the functional components of the kernel have access to all of its internal data structures and routines. The alternative to this is the micro kernel structure where the functional pieces of the kernel are broken out into units with strict communication mechanism between them. This makes adding new components into the kernel, via the configuration process. The best and the robust alternative is the ability to dynamically load and unload the components of the operating system using Linux Kernel Modules. The Linux kernel modules are piece of codes, which can be dynamically linked to the kernel , even after the system bootup. They can be unlinked from the kernel and removed when they are no longer needed. Mostly the Linux kernel modules are used for device drivers or pseudo-device drivers such as network drivers or file system. When a Linux kernel module is loaded, it becomes a part of the Linux kernel as the normal kernel code and functionality and it posses the same rights and responsibilities as the kernel code [32].

4.2.2 Life cycle of Linux kernel module

The life cycle of a module starts with the init_module(). The task of init_module is to prepare the module for later invocation. The module is registered to the kernel and attaches its data-structures and functionality to the kernel. The kernel-defined external functions are also resolved. The life cycle of the module ends with cleanup_module().

Computer Science and Engineering Department TU Patiala

It `unregisters' the module functionality from the kernel. The init_module is called when the module is inserted into the kernel where as the cleanup_module is called just before removing it from the kernel. In the following program, the init_module and the cleanup_module functions are demonstrated.

```
#include
#include
#if CONFIG_MODVERSIONS==1
#define MODVERSINS
#include
#endif
 int init_module()    /* initialise the module*/
{
printk("init_module invoked  \n");
printk("the message is printed from the kernel spacen");

return 0;
}
void cleanup_module()  /*end of module life cycle */
{
printk("cleanup_module invokedn");
printk("module is now going to be unloaded from the kerneln");
}
```

Compile the above program using the following:

#gcc -Wall -DMODULE -D__KERNEL__ -DLINUX -O -c simpelModule.c

Run the compiled module using the following:

#insmod simple Module.o

Now to check the status of the module run

#lsmod

Then remove the module using

#rmmod simple Module

Computer Science and Engineering Department TU Patiala

If the non zero value is returned, then it means that the init_module failed and the kernel module can't be loaded If you have not seen any of the module-initiated console printing (implemented using `printk') about the status of the module, use the following command to see the kernel messages.

#dmesg | less

In the above, commands insmod, lsmod and rmmod are used to load and unload modules to the Linux kernel [14].

4.2.3 Loading modules

Insmod loads the 'loadable kernel modules' in the running kernel. Insmod tries to link a module into the running kernel by resolving all the symbols from the kernel's exported `symbol table'. When the Linux kernel discovers the need for a module, the kernel requests to the kernel daemon (kerneld) to load the appropriate module. The kerneld is the normal user process having exclusive superuser privileges. At the time of booting, kerneld opens the IPC channel to the kernel and uses it for transferring messages to and from the kernel. While loading the module, the kerneld calls modprobe and insmod to load the required module. The insmod utility should able to access the requested module. The demand loadable kernel modules are usually located at /lib/module/ directory as the object files linked as relocatable images. The insmod depends on some critical system calls to load the module to the kernel. It uses the sys_create_module to allocate kernel memory to hold module. It uses get_kernel_syms system call to get the kernel symbol table in order to link the module. It then calls the sys_init_module system call to copy the relocatable object code of the module to the kernel space. And soon after this, insmod calls the initialization function of the concerned module i.e. init_module. All of these system calls can be found in kernel/module.c [16].

4.2.4 Unloading modules

The modules can be unloaded using rmmod command. While removing modules, rmmod ensures the restriction that the modules are not in use and they are not referred by any other module or the part of the kernel. The demand loaded modules are automatically removed from the system by `kerneld' when they are no longer used. Every time it is idle, timer expires and kerneld makes a system call requesting for all

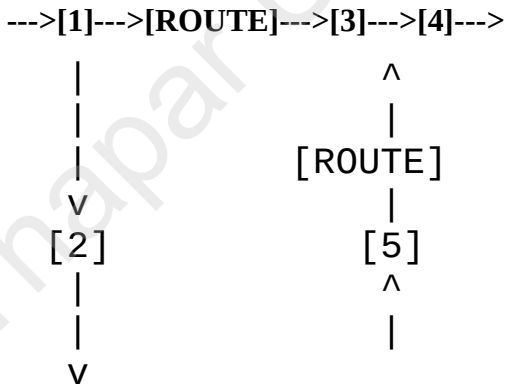
Computer Science and Engineering Department TU Patiala

the demand loaded kernels, which are not busy to be removed. Assuming that the module can be unloaded, the cleanup_module function of the concerned module is called to freeup the kernel resources it has allocated. After the successful execution of the cleanup_module, the module datastructure is marked DELETED and it is unlinked from the kernel and unlisted from the list of kernel modules. The reference list of the modules on which it is dependent is modified and dependency is released. The kernel memory allocated to the concerned module is deallocated and returned to the kernel memory spool [14].

4.2.5 Netfilter Hooks

When a packet is passed to the netfilter framework, it checks at each hook to see if any function has been registered for that protocol and hook. If so, the packet then travels through that registered hook function. So, each hook function has a chance to mangle or do what it wishes with the packets but it eventually has to return a Netfilter code which specifies what to do with the packet next. A hook function is created and registered with a particular hook in the protocol stack.

Netfilter architecture in IPv4



1=NF_IP_PRE_ROUTING

2=NF_IP_LOCAL_IN

3=NF_IP_FORWARD

4=NF_IP_POST_ROUTING

Figure 4.2: Netfilter Architecture [26].

4.2.6 The hook function

Hook function can be created with any name but it has to be registerd with the same name in the entry function . Also it has to be registered , once registered, this function

Computer Science and Engineering Department TU Patiala

would be automatically called as soon as packet reaches the specified hook at which it is registered. All is needed is registering this function. The hook function takes five parameters:

hook number Hook identification number.

Skb Pointer to sk buff structure of the packet to be handled.

Indev Pointer to the net device structure

Outdev Pointer to the net device structure of the network device that should be used by the packet to leave the local computer.

okfn() This function is invoked when all filter functions registered with this hook returned NF_ACCEPT.

The function return an unsigned integer value which specify the what should happen to the packet and packet can be in one of the following states.

- NF DROP (0) : This state specify that the active rules list processing is stopped, and the packet is dropped.
- NF ACCEPT (1): This state specify that the packet is passed to the next packet filter function in the rules list and packet continues its normal journey through the kernel.
- NF STOLEN (2): This state specify that the packet filter function withholds the packet for further pro- cessing, so that the active rules list processing is stopped. In contrast to NF DROP, however, the packet does not have to be explicitly dropped.
- NF QUEUE (3) : The packet is sent to a user space program for modification. This user space program is called queue handler. The queue handler, after making required modifications, reinjects the packet into the network stack and packet continues its journey through kernel.
- NF REPEAT (4): In contrast to NF ACCEPT, rather than a continuation of pro- cessing at the next packet-filter function, the current filter function is invoked again.

4.2.6 Registering the hook function:

Computer Science and Engineering Department TU Patiala

The task of registration is done within entry function of the module. To register the hook function, a structure variable of type nf hook ops is declared, initialized with all the information required and then registered by calling the function nf register hook() and passing the address of this variable as an argument to it.

The hook function is unregistered in the exit function of the module by calling nf unregister hook() and passing the address of the structure variable as an argument to it. Plugging the module into kernel First, a Makefile is created for compiling the module and then it is compiled using the same procedure as that of earlier helloworld program. It is inserted into kernel by using insmod command. As soon as insert this module is inserted into the kernel, the entry function is called which registers the hook function and each packet then goes through our function and the function logs the given message into the system log once for each packet received. We can unload the module from the kernel by issuing rmmod command. Similarly, we can write functions to register at various hooks as per requirement. All is needed is to write the function, declare a nf hook ops structure variable, initialize it with. appropriate values and register it using nf register hook() function [16].

4.3 Communication between user space and kernel space

The address space is managed by Linux dividing it into two distinct regions:

- kernel space
- user space.

Core process of the operating system runs in the kernel space .The memory assigned to all the process are in the kernel space . The kernel modules which we have learnt to create are loaded into the kernel address space only. The user process and the memory assigned to them are reside in the user space only . The kernel is responsible for managing the user processes within this space and prevents them from interfering with each other. However, kernel space can be accessed by user processes only through the use of system calls. IOCTLs and shared memory mechanism are two ways to achieve this communication. we will use IOCTL for the communication .

4.3.1 IOCTL

The IOCTL stands for Input/Output Controls. The IOCTL is used for the mean of a communication between the driver and application . IOCTL also used for reading or

Computer Science and Engineering Department TU Patiala

writing data to and from the user application . The driver exports a number of IOCTLs and defines data structures that would be used in this communication . The device driver modules have a function in them which interprets IOCTL commands sent by the user space program using a predefined function. So, the module to demonstrate IOCTL mechanism will look like a device driver only. There is a common concept about Linux is that linux recognize each thing as a file . Directories, pipes and even hardware devices are files. This concept allows a consistent way of communication with software and hardware constructs . In this work, a header file is created. To use IOCTL mechanism, include the header linux/ioctl.h. A dummy character device (named as 'dpifile' in this work) is created with some major number [14].

4.4 Rules

As DPI is more interesting to look at. Linux, using iptables modules, can check network traffic against fields or conditions deep within a packet. A basic example is *'string'*, which matches packets based on a string located anywhere in the packet:

iptables -A INPUT -m string –string "something bad" -j DROP

Rule to matches packet with content containing 'exe'

iptables -A INPUT -p tcp -m string --algo bm --string 'exe'-j drop

Rule to matches packet with length between 10 and 100 bytes

iptables -A INPUT -p tcp -m length --length 10:100- j drop

In this conceptual solution the objects in the inventory is an rule which will be used to decide what should happen with the each packet transfers through the deep packet inspection hook . The structure of the rule based on the advance filtering capability of the iptable. Such as the filtering of the packet on the basis of malicious content which can be detected with the matching string . The matching string is can be the well known pattern of the well known attacks, name of the viruses,etc. Iptables also provide the capability to filter the packet on basis of the group owner id and process id .

This rule is handled under following conditions:

Computer Science and Engineering Department TU Patiala

1. If there is no rule in list yet, add the new rule only if it is of DROP type. An ACCEPT type of rule simply implies to remove corresponding DROP rule but since list is empty at this point, so ACCEPT type of rule is meaningless here.

2. If already list contains some rule, then the rule is either added to list as new rule; or it may be a duplicate rule to be discarded, or user might have wanted to remove an already existing rule .

4.5 Results

The given conceptual solution is designed to implement in Linux kernel firewall module, In which the packet inspection is handled by the hook name as DPI hook which contains an function (IOCTL) to get packet filtering rules from the user space . The function name as DPI hook is an function which will actually filter the packets based on the rules given and stored in a inventory of the rule set. The hook implemented in the pre routing module, first the address of network layer header and address of transport layer header are retrieved from the current socket buffer which stores the packet for which the hook has been called. Then the packet is matched with each of the rules present in linked list and it will check weather the incoming packet contains the malicious word or the well known hazardous programs . If the packet matches any of the rules, module will DROP this packet otherwise ACCEPT. As The linked list stores only DROP kind of rules. Only the packet which will pass all the rules will be ACCETED. The rules Implemented in the rule set is a 6-tuple i.e. Source and Destination addresses ,Source and Destination ports and the strings . Any of them if zero in some rule in the list, means that field wont be included in the match. It is a kind of wildcard. Thus, if all the fields are zero in the rule, it means all the will pass this rules and so, the packets would be accepted if it also pass the other rules in the rule set . In the block which forms the ioctl function which is associated with a character device, We provide only one ioctl command for the user space program to issue i.e. IOCTL SET MSG. It is because user space program needs to send only one item which is an address of a data structure which contains the rule which is a 6-tuple. So, when this command is invoked, a new memory block is reserved in kernel space and the rule is copied from the user-space-address pointed to by ioctl parameter into the kernel-space- address.

Computer Science and Engineering Department TU Patiala

This function is accomplished by checking if ports and ip addresses match with an already existing entry in the list. If none matches, it would indicate normal insertion of the new rule. If one matches, then target operation value (ACCEPT or DROP) is checked. If these are same (both values would be DROP in this case because list is storing only DROP kind of rules) then it is just a duplicate rule. If operation values differ, it means in the list it is obviously of DROP type and in new rule, it is ACCEPT. So, this rule simply implies remove an already existing rule in the list. So, the user space program only needs to fill up the different fields of a data structure which forms the rule and then call the appropriate ioctl command passing a pointer to that data structure as an argument to the call. The user has nothing to worry about everything else. The rule will automatically be added to the list.

When control pass to the DPI hook function it print the text “We are in DPI Hook function” in the system log files. It shows the packet transfer from the DPI Hook function .

DPI Hook function actually filter the packet on the basis of rules available in the rule set and return the integer value. Some of the cases and the action taken by the hook is as per below.

Case 1: when the packet containing the string “something bad” is pass through the hook ,it matches the string and drop the packet. The DPI hook function return the 0 value.

iptables -A INPUT -m string –string “something bad” -j DROP

Case 2: when the packet containing the .exe file is pass through the hook ,it matches the .exe name and drop the packet. The DPI hook function return the 0 value.

iptables -A INPUT -p tcp -m string --algo bm --string ‘exe’-j drop

Case 3: when the packet containing the string “something bad” is pass through the hook ,it matches the string and drop the packet. The DPI hook function return the 0 value.

Computer Science and Engineering Department TU Patiala

Case 4: when the packet which doesn't contain any maching string is pass through the hook ,then DPI hook just allow it to pass through the next filter and return the value 1.

Computer Science and Engineering Department TU Patiala

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

DPI is next-generation technology that's capable of inspecting every byte of every packet that passes through the DPI device, that means packet headers, types of applications, and actual packet content. Up until now, this wasn't possible with IDS/IPS systems or stateful firewalls. The difference being, DPI has the ability to inspect traffic at layers 2 through 7, hence the "deep" in DPI. A simple analogy would be that of snail mail. IDS/IPS firewalls would be the mail sorters who just read the letter's address, knowing nothing about the letter's content. Inspecting Internet traffic from layers 2 through 7 would correspond to the person who actually reads the letter and understands the contents.

The Linux iptables / netfilter firewall system is flexible, powerful, and extensible. DPI with iptables allows us to very closely monitor and control how data enters and leaves our network. The Linux Operating System is rich in its networking capabilities and overall infrastructure that can be leveraged to provide a more comprehensive DPI solution with support of a number of packet control options. A number of powerful network administration tools like Wireshark, tcpdump and IPRoute are available that make it easy. The packet capturing libraries are easy to use to make own sniffer. The communication between user space and kernel space is done by using IOCTLs or shared memory . Third party modules extend iptables to include functionality that goes beyond conventional firewall capabilities.

The proposed solution provides a standard way for the security companies to implement firewall in the kernel. Moreover, it is a consistent API which can be used by any security application. It provides exhaustive control as it implements maximum possible number of packet control mechanisms and provides the user flexibility to specify the non- rigid set of rules for filtering and controlling the network traffic.

Computer Science and Engineering Department TU Patiala

5.2 Future Scope

A key issue with doing DPI is that it's very computationally intensive . This puts a high stake on the performance of the system to make it a viable solution. Thus the solution needs to work in a way as to have minimal effect on the throughput and latency on the network. The proposed API can be optimized to improve on this part. The proposed solution can also be optimised using the better string matching algorithm. Thus, any new custom packet control mechanisms may be built using a combination of the above.

Computer Science and Engineering Department TU Patiala

References

[1] Hacking firewall and network how to hack remote computer, by Derek Atkins, published on 14 oct,2006. www.katzforums.com/showthread.php?t=262689.

[2] computer/network security how to. Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc.
www.interhack.net/pubs/network-security/.

[3] Jeremy Bennett Brian Hernacki and James Hoagland. An overview of network evasion methods. Technical Report 140e149, 2005. Information Security Technical Report .

[4] Computer Network Security Theory and Practice Wang, Jie Jointly published with Higher Education Press 2009.

[5] Computer and Network security Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc.
http://www.interhack.net/pubs/network-security/network security.html#SECTION000

[6]Packet filtering firewall .
http://www.radagast.bglug.ca/linux/packet_filtering_firewall.pdf

[7] Linux advance routing and packet control how to,by Gregory Maxwell,burt hurbert, April 2001.
www.ibiblio.org/pub/Linux/docs/.../Adv-Routing-HOWTO.pdf.

[8] Firewall Evolution - Deep Packet Inspection, by Ido Dubrawsky, 2003-07
www.securityfocus.com/infocus/1716.

[9] Roll your own firewall with netfilter (Linux Journal), Posted October 13, 2003 by ris
www.linuxjournal.com/artical/7184.

[10] Oskar Andreasson, " Iptables tutorial 1.2.2" ,May 2006.
http://iptables-tutorial.frozentux.net/iptables- tutorial.html.

[11]Driver development part 2: introduction to implementing ioctl. By Toby Opferman, 5 May 2005. http://www.codeproject.com/KB/system/driverdev2.aspx

[12] Bro intrusion detection system, national science foundation , Version 1.0.1 - Last published Feb 14, 2009. http://www.bro-ids.org/.

Computer Science and Engineering Department TU Patiala

[13] Computer/network security services. Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc .http://www.satoriservices.com/security/index.html.

[14] "Driver development part 2: Introduction to implementing ioctls". By Toby Opferman ,5 May 2005 http://www.codeproject.com/KB/system/driverdev2.aspx.

[15] A. V. Aho, M. J. Corasick, "Efficient string matching: An aid to bibliographic search," Comm. of ACM, 18(6):333–340, 1975.

[16] "The Linux Kernel Module Programming Guide", Peter Jay Salzman ,Ori Pomerantz, 2003-04-04 ver 2.4.0.

[17] "How to write simple make file",oreilly publication, 25 march 2005. http://www.hsrl.rutgers.edu/ug/make help.html.

[18] Rose Mathew. Careers in information security course. http://www.bharatbhasha.com/careers.php/49365.

[19] "Deep Packet Inspection: Next Phase of Firewall Evolution" , R. Stiennon, 21 November 2002 http://www.dpacket.org/?q=node/7.

[20] 'Introduction to firewalls". http://netsecurity.about.com/od/hackertools/a/aa072004.htm.

[21] Introduction to iptables. http://www.ks.uni-freiburg.de/download/ inetworkSS04/practical/iptables-intro-short.pdf.

[22] "Chapter 2. Introduction to Firewalls", by Dameon D. Welch-Abernathy – 2004. http://etutorials.org/Networking/Router+firewall+security/Part+I+Security+Overview+and+Firewalls/Chapter+2.+Introduction+to+Firewalls/

[23] Iptables tutorial 1.2.2. Mon, 20 Nov 2006 http://iptables-tutorial.frozentux.net/iptables-tutorial.html.

[24] Admin Digest: The Basics of Linux Network Security Introduction, Rob Reilly Monday, January 6, 2003

Computer Science and Engineering Department TU Patiala

[25] J. Moscola, et al, "Implementation of a content-scanning module for an internet firewall," IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, USA, April 2003.

[26] Roll your own firewall with netfilter. Posted October 13, 2003 by ris . www.linuxjournal.com/artical/7184.

[27] Greg Kroah-Hartman Jonathan Corbet, Alessandro Rubini. Linux Device Drivers. OREILLY, third edition.

[28] Snort. www.snort.org.

[29] Digging deeper into deep packet inspection.Allot communication ,2007 http://www.getadvanced.net/learning/whitepapers/networkmanagement/ Deep%20Packet%20Inspection White Paper.pdf.

[30] Addressing the limitations of deep packet inspection with complete content protection. 2008.

[31] Suminder Ahuja. Introduction to the make utility. August 2002. http://developers.sun.com/solaris/articles/make utility.html#3bb.

[32] Christian Benvenuti. Understanding Linux Network internals. ORIELLY Publication ,second edition.

[33] IT Security Editors on January 17, 2007 . www.itsecurity.com/features/network-security-threats-011707/.

[34] Jeremy Bennett Brian Hernacki and James Hoagland. An overview of network evasion methods. Technical Report 140e149, 2005. Information Security Technical Report.

[35] Neywork security Threats .www.cgisecurity.com/.../top-9-network-security-threats-in-2009.html Tim Carstens. Programming with pcap. http://www.tcpdump.org/pcap.htm.

[35] Microsoft SME Business Center. Technology basics. http://www.microsoft.com/ malaysia/smallbusiness/issues/technology/basics/networks.mspx.

[36] Sun Tzu and the Art of Defensive Tactics . "How is a sixth century BC Chinese military treatise relevant to modern law enforcement?" July 8th, 2008 http://www.officer.com/web/online/On-the-Street/Sun-Tzu-and-the-Art-of-Defensive-Tactics/21$38403.

Computer Science and Engineering Department TU Patiala

[37] "Understanding TCP/IP" by Julian Moss , PC Network Advisor Issue 87 , September 1997 . http://www.techsupportalert.com/pdf/c04100.pdf

[38] An overview of the TCP/IP Protocol Suite by Jason Yanowitz ,September 1997. http://www.acm.org/crossroads/xrds1-1/tcpjmy.html

[39] Understanding the UDP Protocol, by Don Parker , September 20, 2005 . http://www.windowsnetworking.com/articles_tutorials/Understanding-UDP-Protocol.html

[40] "Design and Development of a Powerful Easy-to-use Packet Control API for LINUX" Maninder Singh,24-july-2008.

[41] "Introduction to NAPI" by Alexey Kuznetsov, Robert Olsson , January 12/2002 revision 0.1. http://www.cookinglinux.org/pub/netdev_docs/napi-howto.php3.html

[42] "Behavioral targeting: FTC still prefers self-regulation" by Michael Kassner , February 18th, 2009. http://blogs.techrepublic.com.com/networking/?cat=160

Computer Science and Engineering Department TU Patiala

List of Papers Published/Communicated

[1] Bhadade vaibhav , Maninder Singh, V.P. Singh "Deep Packet Inspection In Linux kernel Firewall" in ISAN 2009 Chitakara Campus . [Status : communicated]

[2] Bhadade vaibhav , Maninder Singh, V.P. Singh "Deep Packet Inspection In Linux kernel Firewall" In JETWI 2009 (Journal of Emerging Technology In Web Intelligence) [Status: Selected for Publication]

References

[1] Hacking firewall and network how to hack remote computer .by Derek Atkins, published on 14 oct,2006 www.katzforums.com/showthread.php?t=262689

[2] computer/network security how to . Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc.
www.interhack.net/pubs/network-security/

[3] Jeremy Bennett Brian Hernacki and James Hoagland. An overview of network evasion methods. Technical Report 140e149, 2005. Information Security Technical Report .

[4] Computer Network Security Theory and Practice Wang, Jie Jointly published with Higher Education Press 2009.

[5] Computer and Network security Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc.
http://www.interhack.net/pubs/network-security/network security.html#SECTION0005200.

[6]Packet filtering firewall .
http://www.radagast.bglug.ca/linux/packet_filtering_firewall.pdf

[7] Linux advance routing and packet control how to ,by Gregory Maxwell,burt hurbert, April 2001.
www.ibiblio.org/pub/Linux/docs/.../Adv-Routing-HOWTO.pdf

[8] Firewall Evolution - Deep Packet Inspection,by Ido Dubrawsky ,2003-07
www.securityfocus.com/infocus/1716.

[9] Roll your own firewall with netfilter (Linux Journal) ,Posted October 13, 2003 by ris
www.linuxjournal.com/artical/7184.

[10] Oskar Andreasson, " Iptables tutorial 1.2.2" ,May 2006.
http://iptables-tutorial.frozentux.net/iptables- tutorial.html.

[11]Driver development part 2: introduction to implementing ioctl. By Toby Opferman ,5 May 2005. http://www.codeproject.com/KB/system/driverdev2.aspx

[12] Bro intrusion detection system, national science foundation , Version 1.0.1 - Last published Feb 14, 2009. http://www.bro-ids.org/.

[13] Computer/network security services. Matt Curtin ,March 1997. Reprinted with the permission of Kent Information Services, Inc
.http://www.satoriservices.com/security/index.html.

[14] "Driver development part 2: Introduction to implementing ioctls". By Toby Opferman ,5 May 2005 http://www.codeproject.com/KB/system/driverdev2.aspx.

[15] A. V. Aho, M. J. Corasick, "Efficient string matching: An aid to bibliographic search," Comm. of ACM, 18(6):333–340, 1975.

[16] "The Linux Kernel Module Programming Guide", Peter Jay Salzman ,Ori Pomerantz, 2003-04-04 ver 2.4.0.

[17] "How to write simple make file",oreilly publication,25 march 2005. http://www.hsrl.rutgers.edu/ug/make help.html.

[18] Rose Mathew. Careers in information security course. http://www.bharatbhasha.com/careers.php/49365.

[19] "Deep Packet Inspection: Next Phase of Firewall Evolution" , R. Stiennon, 21 November 2002 http://www.dpacket.org/?q=node/7.

[20] 'Introduction to firewalls". http://netsecurity.about.com/od/hackertools/a/aa072004.htm.

[21] Introduction to iptables. http://www.ks.uni-freiburg.de/download/ inetworkSS04/practical/iptables-intro-short.pdf.

[22] "Chapter 2. Introduction to Firewalls", by Dameon D. Welch-Abernathy – 2004. http://etutorials.org/Networking/Router+firewall+security/Part+I+Security+Overview+and+Firewalls/Chapter+2.+Introduction+to+Firewalls/

[23] Iptables tutorial 1.2.2. Mon, 20 Nov 2006 http://iptables-tutorial.frozentux.net/iptables-tutorial.html.

[24] Admin Digest: The Basics of Linux Network Security Introduction, Rob Reilly Monday, January 6, 2003

[25] J. Moscola, et al, "Implementation of a content-scanning module for an internet firewall," IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, USA, April 2003.

[26] Roll your own firewall with netfilter. Posted October 13, 2003 by ris . www.linuxjournal.com/artical/7184.

.

[27] Greg Kroah-Hartman Jonathan Corbet, Alessandro Rubini. Linux Device Drivers. OREILLY, third edition.

[28] Snort. www.snort.org.

[29] Digging deeper into deep packet inspection.Allot communication ,2007 http://www.getadvanced.net/learning/whitepapers/networkmanagement/ Deep%20Packet%20Inspection White Paper.pdf.

[30] Addressing the limitations of deep packet inspection with complete content protection. 2008.

[31] Suminder Ahuja. Introduction to the make utility. August 2002. http://developers.sun.com/solaris/articles/make utility.html#3bb.

[32] Christian Benvenuti. Understanding Linux Network internals. ORIELLY Publication ,second edition.

[33] IT Security Editors on January 17, 2007 . www.itsecurity.com/features/network-security-threats-011707/.

[34] Jeremy Bennett Brian Hernacki and James Hoagland. An overview of network evasion methods. Technical Report 140e149, 2005. Information Security Technical Report.

[35] Neywork security Threats .www.cgisecurity.com/.../top-9-network-security-threats-in-2009.html Tim Carstens. Programming with pcap. http://www.tcpdump.org/pcap.htm.

[35] Microsoft SME Business Center. Technology basics. http://www.microsoft.com/ malaysia/smallbusiness/issues/technology/basics/networks.mspx.

[36] Sun Tzu and the Art of Defensive Tactics . “How is a sixth century BC Chinese military treatise relevant to modern law enforcement?” July 8th, 2008 http://www.officer.com/web/online/On-the-Street/Sun-Tzu-and-the-Art-of-Defensive-Tactics/21$38403.

[37] “Understanding TCP/IP” by Julian Moss , PC Network Advisor Issue 87 , September 1997 . http://www.techsupportalert.com/pdf/c04100.pdf

[38] An overview of the TCP/IP Protocol Suite by Jason Yanowitz ,September 1997. http://www.acm.org/crossroads/xrds1-1/tcpjmy.html

[39] Understanding the UDP Protocol, by Don Parker , September 20, 2005 . http://www.windowsnetworking.com/articles_tutorials/Understanding-UDP-Protocol.html

[40] “Design and Development of a Powerful Easy-to-use Packet Control API for LINUX” Maninder Singh,24-july-2008.

[41] “Introduction to NAPI” by Alexey Kuznetsov, Robert Olsson , January 12/2002 revision 0.1. http://www.cookinglinux.org/pub/netdev_docs/napi-howto.php3.html

[42] “Behavioral targeting: FTC still prefers self-regulation” by Michael Kassner , February 18th, 2009. http://blogs.techrepublic.com.com/networking/?cat=160