

Implementation of an intrusion detection system

BY

E.M.S.A. GUNASINGHE

00/AS/028

This thesis is submitted in partial fulfillment of the report for the degree of

Bachelor of Science

in

Physical Sciences major in Computer Science

Department of Physical Sciences

Faculty of Applied Sciences

Sabaragamuwa University Of Sri Lanka

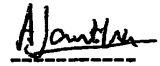
Buttala

March 2004

DECLARATION

I certify that this dissertation does not incorporate without acknowledgement any material Previously submitted for degree or diploma in any university and to the best of my knowledge and belief it does not contain any material previously published or written or orally communicated by another person except where due to reference is made in the text .

E.M.S.A. Gunasinghe.

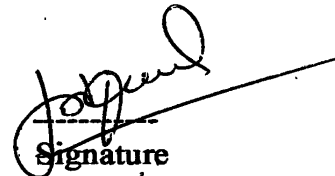


Signature

To best of my knowledge the above particulars are correct

External Supervisor

Dr. K.M.Liyanage
Senior lecturer. Dept EEE.
Faculty of engineering
University of Peradeniya
Tel-0812 385184
email liya@cc.pdn.ac.lk



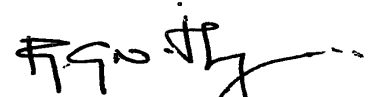
Signature

21/5/14

Date

Internal Supervisor

Dr .R.G.N. Meegama
Co-ordinator
Center for computer studies
Sabaragamuwa university of sirilanka
Belihuloya.
Tel-0452 280049



Signature

13/05/2014

Date

Head of the Department

Dr. Nirmalee Wickramaratne
Head/Dept Physical Sciences
Faculty of Applied Sciences
Sabaragamuwa University of Sri Lanka.



Signature

26/05/2014

Date

ACKNOWLEDGEMENTS

First of all I wish to thanks to my internal supervisor Dr R.G.N: Meegama, Co-ordinator, Center for computer studies, Sabaragamuwa University of SriLanka for his assistance, encouragement, guidance. I specially thanks him for his effort towards securing the opportunity of training at Computing cente, Faculty of Engineering, University of Peradeniya.

I like to express my sincere gratitude to my external supervisor Dr K.M: Liyanage, Senior Lecturer, Faculty of Engineering, University of Peradeniya, Peradeniya, who kindly offered me the industrial placement with all the facilities.

Then my deepest gratitude is express to the Dr D.B.M: Wickramarathna the Dean, Faculty of Applied sciences Sabaragamuwa University of SriLanka, and Dr Nirmalee Wickramarathna Head of the Department of physical sciences Faculty of Applied sciences Sabaragamuwa University of Sri Lanka for guiding me towards a successful completion.

I am heavily thanks to Mr.Manjula Gunaratna. Instructor, Computing center, Faculty of Engineering, Peradeniya, who gave me the invaluable support thought my training. I very much like to thanks to all the staff of Computing center, Faculty of Engineering, University of Peradeniya giving me the help during the training period.

ABSTRACT

Internet was the most widely used information carrier and the service provider for web based information. The Internet servers had to face major problem with illegal access. Generally, when a person made a request to a particular web server, that message proceeded to the web server through the permitted logical ports (in the case of standard http request, the port number 80). If a hacker tried to access the web server through ports that not opened to the outside world (eg. Port 25, 21 that did not permit http request), then the system detected the information about such unusual requests and wrote the relevant information to a file, which would be read by security experts or system administrators.

The IDS program developed to capture the data packets at the network interface before it entering to the firewall, and captured data packet not reassembled and read their header data such as source IP, source port, protocol etc. The program wrote in C language for the programmable Ethernet card, and the program could run in Linux environment.

The program successfully detected unusual traffic other than http request made to a web server. As a future development, source IP address could be converted to FQDN, that identified domain name and time where unusual request made.

CONTENTS

	PAGE NO.
ABSTRACT	I
ACKNOWLEDGEMENTS	II
LIST OF CONTENTS	III
LIST OF FIGURES	V
LIST OF TABLE	VI
Chapter 1	
1. Introduction	
1.1 Intrusion detection	1
1.2 Objectives	5
Chapter 2	
2. Theoretical background	
2.1 Various type of internet packets	6
2.1.1 Considering the packet issues	6
2.1.2 The raw packets	9
2.1.2 IP control and error message (ICMP)	10
2.1.3 User datagram protocol (UDP)	11
2.1.4 Transmission control protocol (TCP)	13
2.1.5 How to IP protocol fit together	18
2.2 Data packet and layers	19
2.2.1 Packet filtering firewall and layers	20
2.3 Firewall	21
2.3.1 Packet filtering firewall	22
2.3.2 Proxy servers	23
2.4 What is a socket	24
2.5 Web server	26
2.6 Internet layer addressing	27
2.7 Network intrusion detection system	30
2.7.1 How do intruder get in to system	31
2.7.3 Why can intruder get in to system	32
2.7.4 System configuration	33
2.7.5 Password cracking	34

2.7.7 Sniffing unsecured traffic	34
2.7.8 Design flaws	35
2.7.9 How do intruders get password	35
2.7.10 What is a typical intrusion scenario	37
2.7.11 What are some common intrusion signatures	38
2.7.12 What are some common exploits	38
2.7.13 Web server attracts	38
2.7.12 Web browser attracts	38
2.7.13 SMTP (send mail) attracts	39
2.7.14 DNS poisoning through sequence prediction	41
2.7.15 What are some common reconnaissance scan	42
2.7.16 What are some common DOS (Denial of services)	43
2.7.17 How are intrusion detected	44
2.7.18 How does a NIDS match signatures with incoming traffic	44
2.7.19 What happens after a NIDS detects an attract	45
2.7.20 What other countermeasures	46
2.7.21 Where do I put IDS systems on my network	46
2.7.22 How does IDS fit with rest of my security frameworks	47
2.7.23 how can I detect if someone is running a NIDS	48
Chapter 3	
3. Methodology	49
Chapter 4	
4.1 Results	51
4.2 Discussion	51
Chapter 5	
5.1 Conclusion	53
5.2 Recommendation	53
Chapter 6	
6. References	54

LIST OF FIGURE

1.1 Usual access for web server	5
1.2 Unusual access for web server	5
2.1 Structure of icmp header	11
2.2 Structure of udp header	13
2.3 Structure of ip header	16
2.4 How data packet physically fit together	18
2.5 Structure of ip header	19
2.6 different packet and different layers	20
2.7 Packet filtering firewall	21
2.8 Transmit the data between two computer using socket	24
2.9 Internet protocol	25

LIST OF TABLE

2.1 Packet type benefits	6
2.2 The three-way handshake	17
2.4 TCP connection close	17

CHAPTER 1

1 Introduction

1.1 Intrusion detection

A computer system should provide confidentiality, integrity and assurance against intrusion attempts. However, due to increased connectivity on the Internet, more and more systems are subject to attack by intruders. Intrusion Detection Systems (IDS) are used by organizations to extend their security infrastructure by detecting and responding to unauthorized access of resources in real time. This chapter discusses what an intrusion Detection system, model is and main techniques.

What is an IDS? ID stands for Intrusion Detection, which is the art of detecting inappropriate, incorrect, or anomalous activity. An Intrusion Detection System (IDS) analyzes a system for file system changes or traffic on the network, this system, learns what normal traffic looks like, then notes changes to the norm that would suggest an intrusion or otherwise suspicious traffic. So an IDS protects a system from attack, misuse, and compromise. It can also monitor network activity, audit network and system configurations for vulnerabilities, analyze data integrity, and more. Depending on the detection methods someone chooses to deploy.

There are basically 3 main types of IDS used: Network based (a packet monitor), Host based (looking for instance at system logs for evidence of malicious or suspicious application activity in real time), and Application Based IDS (monitor only specific applications).

Host-Based IDS (HIDS). Host-based systems were the first type of IDS to be developed and implemented. These systems collect and analyze data that originate on a computer that hosts a service, such as a Web server. Once this data is aggregated for a given computer, it can either be analyzed locally or sent to a separate or central analysis machine. One example of a host-based system is programs that operate on a system and receive application or operating system audit logs. These programs are highly effective for detecting insider abuses. On the down side, host-based systems can get unwieldy. With several thousand possible endpoints on a large network,

can get unwieldy. With several thousand possible endpoints on a large network, collecting and aggregating separate specific computer information for each individual machine may prove inefficient and ineffective.

Possible host-based IDS implementations include Windows NT/2000 Security Event Logs, RDMS audit sources, Enterprise Management systems audit data (such as Tivoli), and UNIX Syslog in their raw forms or in their secure forms such as Solaris' BSM; host-based commercial products include Real Secure, ITA, Squire, and Entercept etc.

Network based IDS(NIDS). NIDS are used to monitoring the activities that take place on a particular network, Network-based intrusion detection analyzes data packets that travel over the actual network. These packets are examined and sometimes compared with empirical data to verify their nature: malicious or benign. They have network interface in promiscuous mode. Because they are responsible for monitoring a network, rather than a single host, Network-based intrusion detection systems (NIDS) tend to be more distributed than host-based IDS. Instead of analyzing information that originates and resides on a computer, network-based IDS uses techniques like "packet-sniffing" to pull data from TCP/IP or other protocol packets traveling along the network.

This surveillance of the connections between computers makes network-based IDS great at detecting access attempts from outside the trusted network. In general, network-based systems are best at detecting the following activities.

Unauthorized outsider access: When an unauthorized user logs in successfully, or attempts to log in, they are best tracked with host-based IDS. However, detecting the unauthorized user before their log on attempt is best accomplished with network-based IDS.

Bandwidth theft/denial of service: These attacks from outside the network single out network resources for abuse or overload. The packets that initiate carry these attacks can best be noticed with use of network-based IDS.

Some possible downsides to network-based IDS include encrypted packet payloads and high-speed networks, both of which inhibit the effectiveness of packet interception and detect packet interpretation.

Application Based IDS: Application Based IDS monitor only specific applications such as database management systems, content management systems, accounting systems etc. They often detect attacks through analysis of application log files and can usually identify many types of attack or suspicious activity. Sometimes application-based IDS can even track unauthorized activity from individual users. They can also work with encrypted data, using application-based encryption or decryption services.

There are two IDS Models used. knowledge-based IDS (signature-based model) Which alert administrators before an intrusion occurs using a database of common attacks. Behavioral IDS(anomaly model) That tracks all resource usage for anomalies & malicious activity. Some IDSes are standalone services that work in the background and passively listen for activity, logging any suspicious packets from the outside. Others combine standard system tools, modified configurations, and verbose logging.

knowledge-based IDS: Knowledge based systems use signatures about attacks to detect instances of these attacks. Knowledge based systems is the most-used IDS model. Signatures are patterns that identify attacks by checking various options in the packet, like source address, destination address, source and destination ports, flags, payload and other options. The collection of these signatures composes a knowledge base that is used by the IDS to compare all packet options that pass by and check if they match a known pattern. Signatures have the same limitations as a patch - it is not possible to write the signature until the hack has materialized.

Behavioral IDS: Behavior based systems use a reference rule of normal behavior and flag deviations from this model as anomalous and potentially intrusive. A behavioral rule aims to define a profile of legitimate activity. Any activity that does not match the profile, including new types of attack, is considered anomalous. As rules are not specific to a particular type of attack, forensic information is not normally very detailed. However, rules can identify malicious behavior without

having to recognize the specific attack used. This approach offers unparalleled protection against new attacks ahead of any knowledge being available in the security community. The disadvantage of this model is that it macausea high number of false positive alerts.

False positive A report of an attack or attempted attack when no vulnerability existed or no compromise occurred. *False negative* The failure of an IDS to report an instance in which an attacker successfully compromises a host or network. **Sensor** : The computer that monitors the network for intrusion attempts. Sensors usually run in promiscuous mode, often without an IP address.

IDS application for web server

Normally, when a person makes a request to a particular web server (usual access fig:1.1), that message proceeds to the web server through the permitted logical ports (in the case of standard http requests, that port number is 80).. If a hacker (or an intruder or an unusual access fig:1.2) tries to access the web server through ports that are not open to the outside world (eg. Port 25, 21 that do not permit http requests), then the system should be able to detect the following information about such unusual requests and write the relevant information to a file which can be read by security experts or system administrators. Information contained in this file can be used to implement the necessary security policies to the server. The format of the file that contains information about outside computers that tries to gain illegal access to the web server is as follows example.

Source ETH ID	:=20.12.02.01
Destination ETH ID	:=ff.ff.ff.ff.ff
Source IP address	:=192.168.10.182
Destination IP Address	:=192.168.20.10
Protocol	:=TCP
Packet length	:=1024
Feld	:=4
Version	:=IPv6

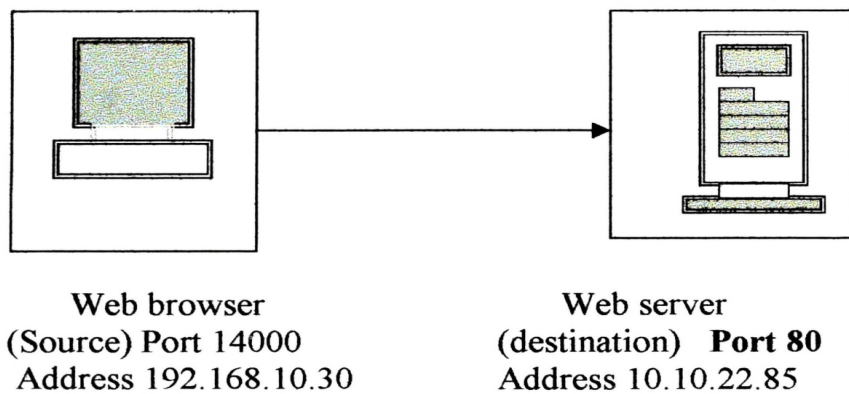


figure 1.1 usual access for web server.

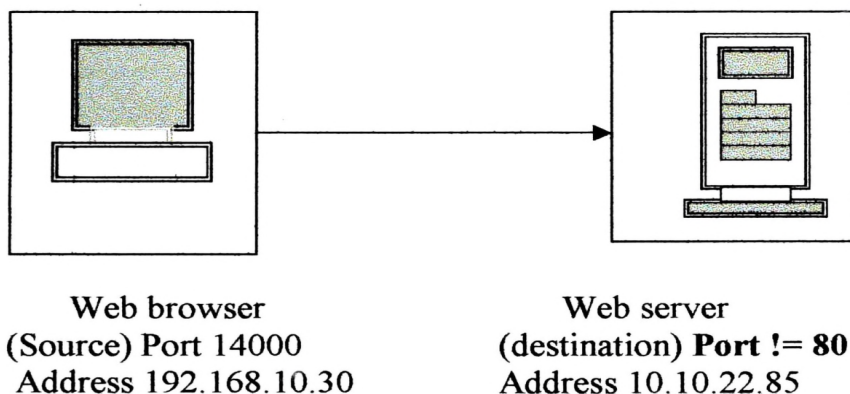


figure 1.2 unusual access for web server.

1.2 Objectives

Overall objectives: Development of security of the web server.

A computer system should provide confidentiality, integrity and assurance against intrusion attempts. However, due to increased connectivity on the Internet, more and more systems are subject to attack by intruders. Intrusion Detection Systems (IDS) are used by organizations to extend their security infrastructure by detecting and responding to unauthorized access of *resources in real time.*

Specific objectives: Identify unusual requests made to the web server from outside computer

CHAPTER 2

2.1 Various types of internet Packets

The Internet Protocol offers several packet protocols that range from very fast to very reliable. All of them rest on the lowest layer—the basic IP packet. However, each layer has evolved to solve specific problems. To select the correct packet type, you must know about what you're transmitting. The packet types most likely to be of interest are TCP, UDP, ICMP, and raw. Knowing the advantages and disadvantages of each type can help you choose the most appropriate for your application. Each packet type has different benefits, as summarized in Table 2.1

Table 2.1 Packet Type Benefits.

	RAW	ICMP	UDP	TCP
Overhead(bytes)	20-60	20-60+[4]	20-60+[8]	20-60+[20-60]
Message size(bytes)	65,535	65,535	65,535	unlimited
Reliability	Low	Low	Low	High
Message type	Datagram	Datagram	Datagram	Stream
Throughput	High	High	Medium	Low
Data integrity	Low	Low	Medium	High
Fragmentation	Yes	Yes	Yes	Low

In this table, notice that each packet type contains comparisons. A reliability of Low value only means that you cannot rely on the protocol to help reliability. While the differences may seem extreme, remember that they are merely comparisons.

2.1.1 Considering the Packet's Issues

Each protocol addresses issues in the transmission. The following sections define each issue and associated category from Table 2.2. This information can help you see why certain protocols implement some features and skip others.

Protocol Overhead: Protocol overhead includes both the header size in bytes and the amount of interaction the protocol requires. High packet overhead can reduce throughput, because the network has to spend more time moving headers and less time reading data. Strong protocol

synchronization and handshaking increase interaction overhead. This is more expensive on WANs because of the propagation delays. Table 3.2 does not include this measurement.

Protocol Message Size: To calculate network throughput, you need to know the packet size and the protocol's overhead. The transmission size gives you the maximum size of a sent message. Since all but TCP use a single-shot message, this limitation is typically due to the limits of IP packet (65,535 bytes). The amount of data your program transmits per packet is the transmission size less the headers.

Protocol Reliability: Part of the problem with networks is the possibility of lost messages. A message could be corrupted or dropped as it moves from one host or router to another, or the host or router could crash or fail. In each case, a message may simply be lost, and your program may need to follow up. Also, you may need to make sure that the destination processes the packets in the correct order. For example, you may compose a message that does not fit in one packet. If the second packet arrives before the first, the receiver must know how to recognize and correct the problem. However, the order is not important when each message is independent and self-contained.

The packet's reliability indicates the certainty of safe arrival of messages and their order. Low reliability means that the protocol can't guarantee that the packet gets to the destination or that the packets are in order.

Protocol Message Type: Some messages are self-contained and independent from other messages. Pictures, documents, email messages, and so on are a few examples that may fit the size of the packet. Others are more in the form of a flowing stream, such as Telnet sessions, HTTP's open channel [RFC2616], large documents, pictures, or files. The message type defines which style best fits each protocol. To get the best performance, you need to know the throughput. Often, the bits-per-second is a small part

HTTP's Protocol: HTTP 1.0 could effectively use UDP for transferring messages instead of TCP. The client simply sends the request for a specific document, and the server replies with the file. Effectively, no conversation occurs between client and server.

Protocol Throughput: The most noticeable aspect of data transmission is network throughput. Getting the most out of your network is the best way to make your users happy of

the whole equation; it only indicates how the network could perform under ideal circumstances.

The protocol throughput measures how much real data the originator can send to the destination within a period of time. If the headers are large and the data small, the result is low throughput. Requiring acknowledgment for each message dramatically reduces throughput. By default, high reliability and integrity result in low throughput and vice versa.

Protocol Data Integrity: The networking technology currently has a lot of safeguards for data integrity. Some network interfaces include a checksum or cyclical redundancy check (CRC) for each low-level message. They also include special hardware technology that can filter out noise and get to the real message. Additionally, each protocol includes measures to detect errors in the data. The importance of data integrity depends on the data. That is, some data requires very careful oversight, while less important data is less critical. Here are some types of data.

Fault-Intolerant—Life-critical data. Anything that can affect public or private health/life. For example, life signs and vital signs from medical equipment and missile launch commands.

- **Critical—Important and reliable data.** Data that if out of sequence or faulty can cause harm to property or security. For example, financial transactions, credit cards, PIN numbers, digital signatures, electronic money, trade secrets, virus scanner updates, and product updates.
- **Important—Data that requires proper functionality.** Any loss can cause malfunction. For example, X11 connections, FTP downloads, Web pages, server/router addresses, and Telnet connections.
- **Informational—Data that can be less than 100% reliable for proper functionality.** For example, email, news feeds, advertisements, and Web pages.
- **Temporal—Data that is date/time bound.** Unless the program uses the information within a specific time, its importance lessens. For example, weather data, surveillance data, and time.
- **Lossy—Data that can degrade without loss of usefulness.** These are typically audio or visual. For example, movies, audio files, photos, and spam (of course).

Prior to choosing the packet type or protocol, try to categorize data according to this list. Also include the additional (or external) constraints of the program. These may be regulatory constraints as well.

Protocol Fragmentation: Large messages on slow networks can frustrate other users. All networks place a maximum frame size so those large messages don't dominate the network. Keep in mind that the routing host may still carve up, or fragment, large messages that go through a constricted network. Each protocol has a different likelihood of fragmentation. Since reassembling fragmented messages is part of IP, the reassembly may be transparent to the higher protocols. Certain circumstances, however, may require the packet's wholeness. This is particularly important for network performance. When routers carve up the packet into smaller chunks, the router has to take the time to chop up the message, and the resulting packet overhead increases. By blocking fragmentation, the network drops the packet and returns a message-too-big error to your program.

Packet Types: The following sections describe each packet, showing its statistics and header definition (if there is one). Each section uses a quick-reference style to help you quickly see the features of each protocol. Use this style to help you choose the right packet for your applications.

2.1.1 The Raw Packet

A raw packet has direct access to an IP packet and header. It is useful in writing special or custom protocols. Linux provides the option to work with different layers in the Internet Protocol stack most basic TCP/IP message is the raw IP message. It has no information other than the most basic. You can use the IP packet itself to create the most basic layer to create your own custom protocols. Access the IP packet by selecting `SOCK_RAW` in the `socket()` system. For security, you must have root privileges to run a raw socket program.

The raw socket lets you play with the guts of the IP packet. You can configure the socket to work on two levels of detail: data only or data and header manipulation. Data manipulation is like UDP data transfers but does not support ports. In contrast, header manipulation lets you set the header fields directly. Using this message has both advantages and disadvantages. As a datagram message, it offers no guarantees of arrival or data integrity. However, you can send and receive messages nearly at network speed.

2.1.2 IP Control and Error Messaging (ICMP)

The Internet Control Message Protocol (ICMP) is one of the layers built on top of the basic IP packet. All Internet-connected computers (hosts, clients, servers, and routers) use ICMP for control or error messages. It is used for sending error or control messages. Some user programs also employ this protocol, such as traceroute and ping. You can reuse your socket to send messages to different hosts without reopening the socket if you employ the ICMP in your own program. Send messages using the `sendmsg()` or `sendto()` system call (described in the next chapter). These calls require an address of the destination. With a single socket, you can send messages to as many peers as you want.

The advantages and disadvantages of an ICMP packet are essentially the same as raw IP (and other datagrams). However, the packet includes a checksum for data validation. Also, the likelihood that the network may fragment an ICMP packet is very small. The reason is because of the nature of ICMP messages: They are for statuses, errors, or control. The message is not going to be very large, so it may never require reassembly.

While you can use the ICMP for your own messages, it is usually for error messages and control. All networking errors travel the network through an ICMP packet. The packet has a header that holds the error codes, and the data part may contain a more specific message describing the error. Part of the IP protocol, ICMP gets an IP header and adds its own header. Following Listing shows a definition of the structure and structure of ip header figure 2.1

ICMP Structure Definition

Formal definition in `netinet/ip_icmp.h`

```
typedef unsigned char ui8;
typedef unsigned short int ui16;
struct ICMP_header {
    ui8 type;           ( Error type )
    ui8 code;          ( Error code )
    ui16 checksum;     ( Message checksum )
    uchar msg[ ];     ( Additional data description )
};
```

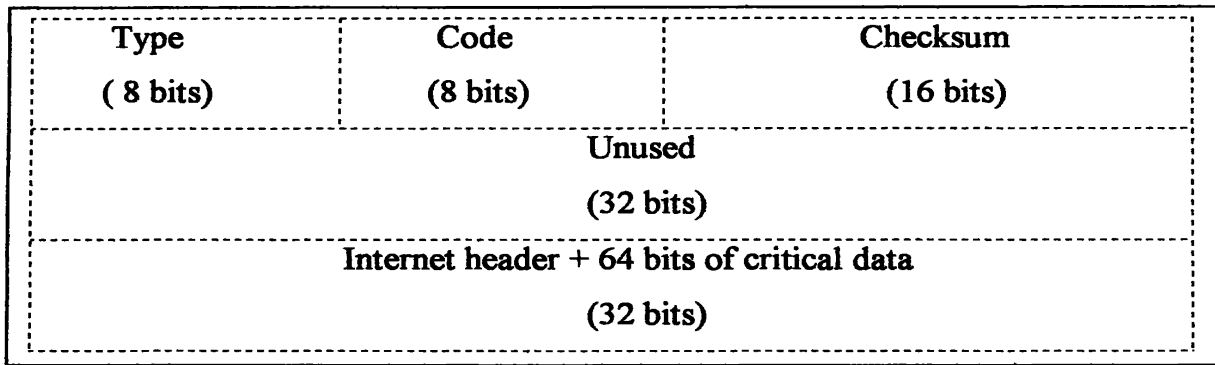


Figure 2.1 structure of ip header.

Type and code define what error occurred. msg can be any additional information to help detail what went wrong.

2.1.3 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is used mostly for connectionless (independent messages) communications. It can send messages to different destinations without re-creating new sockets and is currently the most common connectionless protocol. Each layer up the IP stack provides more focus on data and less on the network. UDP hides some of the details about error messages and how the kernel transmits messages. Also, it reassembles a fragmented message.

A message you send via UDP is like an email message: The destination, origin, and data are all the information it needs. The kernel takes the message and drops it on the network but does not verify its arrival. As with the ICMP packet, you can send to multiple destinations from a single socket, using different send system calls. However, without the verification, you can experience near-maximum throughput.

Without arrival verification, the network can lose data reliability. The network can lose packets or fragments and corrupt the message. Programs that use UDP either track the message themselves or don't care if something gets lost or corrupted. Of the different data types (previously defined), Informational, Temporal, and Lossy best fit the UDP services. The primary reason is their tolerance for loss. If your Web camera fails to update every client, the end user is unlikely to either notice or care. Another possible use is a correct time service. Because correct time is Temporal, a host may drop a couple of clock ticks without losing

integrity.UDP offers the advantage of high speed. Moreover, you can increase its reliability yourself in the following ways:

- Break up large packets. Take each message and divide it into portions and assign a number (such as 2 of 5). The peer on the other end reassembles the message. Bear in mind that more overhead and less sent data decrease throughput.
- Track each packet. Assign a unique number to each packet. Force the peer to acknowledge each packet because, without an acknowledgment, your program resends the last message. If the peer does not get an expected packet, it requests a resend with the last message number or sends a restart message.
- Add a checksum or CRC. Verify the data of each packet with a data summation. A CRC is more reliable than a checksum, but the checksum is easier to calculate. If the peer discovers that data is corrupted, it asks your program to resend the message.
- Use timeouts. You can assume that an expired timeout means failure. Your originator could retransmit the message, and your receiver could send a reminder to the sender.

The Critical and Important data types require the reliability found in TCP or better. Fault-Intolerant requires much more than any of these protocols offer. These outlined steps mimic the reliability of TCP.UDP relies on IP's features and services. Each UDP datagram packet receives an IP and a UDP header. Following Listing defines how the UDP structure appears.

UDP Structure Definition

(Formal definition in netinet/udp.h)

```
typedef unsigned short int ui16;
struct UDP_header {
    ui16 src_port;      (Source port number)
    ui16 dst_port;     (Destination port number)
    ui16 length;       (Message length)
    ui16 checksum;     ~(Message checksum)
    uchar data[ ];     (Data message)
};
```

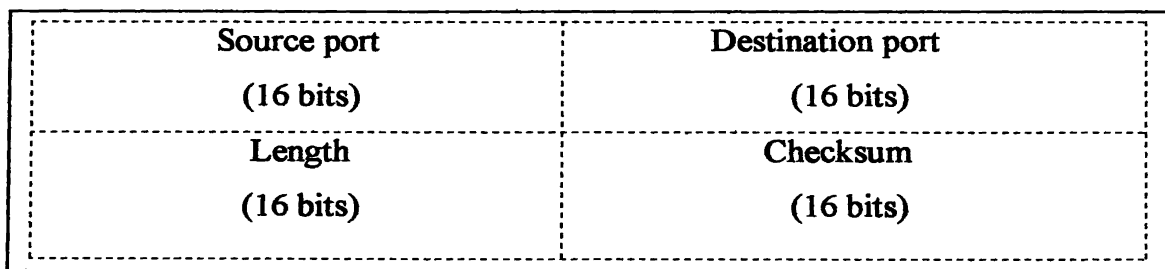


Figure 2.2 structure of UDP header

UDP creates a virtual network receptacle for each message in the form of ports. With the port, IP can rapidly shuffle the messages to the correct owner. Even if you don't define a port with `bind()`, the IP subsystem creates a temporary.

2.1.4 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is the most common socket protocol used on the Internet. It can use `read()` and `write()` and requires re-creating a socket for each connection. Taking reliability one step further requires ensuring that the destination gets the exact message the originator sent. UDP has the speed but does not have the reliability that many programs require. TCP solves the reliability problem. The network, however, has several fundamental problems that make it unreliable. These problems are not limitations. Instead, they are inherent in the design of the network. To get reliable, streamable messages through the tangled Web, TCP/IP has to incorporate many of the ideas to improve reliability suggested in the section on UDP. The Internet has three hurdles: dynamic connections, data loss, and constricted paths, as discussed in the following sections.

Dynamic Connections: One host sends a message to another host. That message travels the networks, going through various routers and gateways. Each message sent may use a different path. Networking segments (connections between computers) often appear and disappear as servers come up and go down. The power of the Internet is its capability to adapt to these changes and route the information accordingly.

Adaptability is one of the driving forces behind the Internet. Your computer can make a request, and the network tries possible avenues to fill the order. Unfortunately, this advantage means that the path between your computer and the server or peer can change, lengthening and shortening the distance. As the path lengthens, propagation times increase. This means

that your program could send successive messages and many would arrive at different times, often out of order.

TCP ensures that the destination has correctly received the last message before it sends the next message. Compare this to a series of numbered messages (this is really how TCP works). Your program may send 10 messages in succession. TCP takes each message, attaches a unique number, and sends it off. The destination accepts the message and replies with an acknowledgment. Upon receiving the acknowledgment, TCP lets your program send the next message.

Sliding Window Protocol: TCP uses a better technique than the send/wait (or ACK/NACK) protocol, which is too slow for anyone's patience. Instead, it uses a sliding window: It gauges when and how often to reply with an ACK. Slower or dirtier connections may increase the acknowledge messages. Connections that are faster and lose less allow more messages to ship before expecting acknowledgments. This is part of the Nagle Algorithm.

Data Loss: When the destination gets your message, it determines the integrity of the data. The data may travel along less-than-optimal communication paths that may drop or corrupt message bits. Remember that the network sends every message one bit at a time. TCP sends with the message a checksum to verify the data. TCP is the last layer that can detect and remedy corrupted data.

If the destination detects any errors, it sends back an error, requesting a retransmittal from your program. Likewise, if your computer does not get an acknowledgment within a specific amount of time, the TCP subsystem automatically resends the message without your program's intervention.

Constricted Paths: Going back to the single message sent to a particular host, suppose that the message is too long for the intervening segments. The problem that the packet encounters as it passes through the network is the different technologies and transmission carriers. Some networked computers permit lengthy packets; others place limits on the size.

UDP tries to send the largest message that it can. This can be a problem with the constricted data paths. The IP algorithms anticipate that the routers may fragment data. Likewise, IP expects that it has to reassemble the incoming message.

TCP, on the other hand, limits every packet to small chunks. TCP breaks up longer messages, before the network has the chance to touch them. The size TCP chooses is one that a majority of networks can accept intact. By default, TCP uses 536 bytes and typically negotiates up to 1,500. To increase that size manually, set the MSS (maximum segment size) TCP socket option

The receiver may find that the message's packets are out of order. TCP reorders them before handing the message to your program. Solving all these network problems adds protocol and header overhead to TCP's algorithm. Of course, the added overhead of all TCP's techniques slows performance noticeably. TCP had to add a lot of information to its header to support all the features that it offers you. The size, in bytes, of the TCP header is about three times that of the UDP header. See following Listing for a definition of the structure.

TCP Structure Definition

(Formal definition in `netinet/tcp.h`)

```
typedef unsigned char ui8;
typedef unsigned short int ui16;
typedef unsigned int ui32;
typedef unsigned int uint;
struct TCP_header {
    ui16 src_port;    (Originator's port number)
    ui16 dst_port;    (Destination's port number)
    ui32 seq_num;     (Sequence number)
    ui32 ack_num;     (Acknowledgment number)
    uint data_off:4;  (Data offset)
    uint __res:6;     (reserved)
    uint urg_flag:1;  (Urgent, out-of-band message)
    uint ack_flag:1;  (Acknowledgment field valid)
    uint psh_flag:1;  (Immediately push message to process)
    uint rst_flag:1;  (Reset connection due to errors)
    uint syn_flag:1;  (Open virtual connection (pipe))
    uint fin_flag:1;  (Close connection)
    ui16 window;     (How many bytes receiver allows)
    ui16 checksum;   (Message checksum)
}
```

```

ui16 urg_pos;    ( Last byte of an urgent message )
ui8  options[ ]; ( TCP options)
ui8  __padding[ ]; (Needed for aligning data[ ])
uchar data[ ];   ( Data message )
};

```

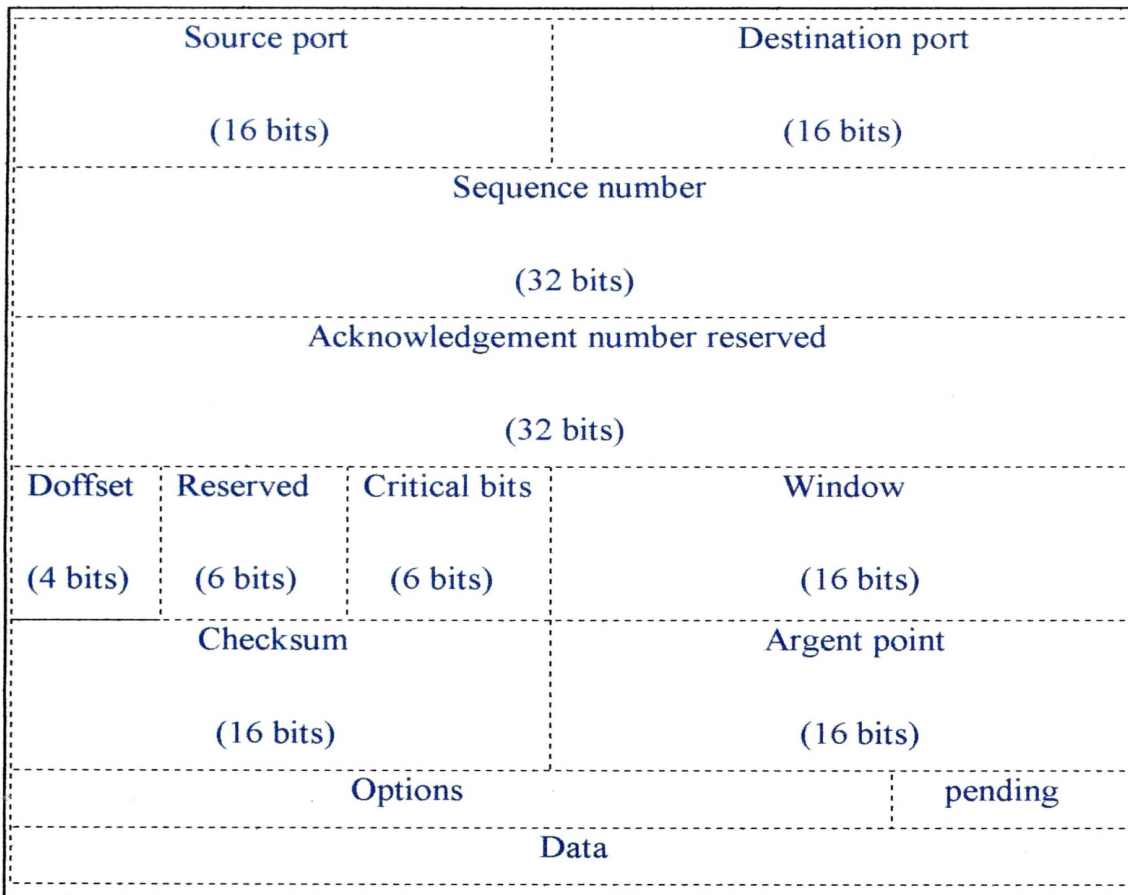


figure 2.3 structure of TCP header

The header may have a variable size, so the `data_off` field points to the beginning of the data. To save header space, this field acts like the IP's `header_len` field: It gives the count of 32-bit words that physically precede your data.

TCP uses some of the fields exclusively for opening a connection, flow control, and connection closure. During a communication session, some of the header is empty. The following paragraphs describe a few interesting fields. The TCP header uses the same port number found in UDP. But `seq_num` and `ack_num` provide traceability to the stream. When you send a message, the IP subsystem attaches a sequence number (`seq_num`). The receiver

replies that it got the message with an acknowledgment number (ack_num) that is 1 greater than the sequence number. This feature lets acknowledgment packets carry data as well.

Looking at the TCP Interactions: When you open a streaming connection, your program and server exchange the messages listed in Table 2.4

Table 2.4 The Three-Way Handshake

Client sends	Server sends	Description
SYN=1(syn flag)	SYN=1(syn flag)	Request a virtual connection(pipe)
ACK=0(ack flag)		Set sequence number
		Permit and acknowledge virtual connection
ACK=1(ack flag)		
SYN=0(syn flag)		
ACK=1(ack flag)		establish a virtual connection

This is called the *three-way handshake*. During the transfers, the client and server specify the buffer size of their receiving buffers (windows).

On the other hand, closing a connection is not as simple as it may appear, because there may be data in transit. When your client closes a connection, the interaction shown in Table 2.5 may occur.

Table 2.5 TCP Connection Closure

Client	Server	Description
FIN=1(fin flag)	Transmit and Receives data	Client request close
ACK=1	Transmit more Receives more	Server channels flushed
ACK=1	FYN=1	Close accepted server close and a waits ACK
ACK=1		Client closes its side

Closing the TCP connection makes it impossible to reuse the socket for other connections. For example, if you connect to a server, the only way to sever the connection is to close the channel, which closes the socket as well. If you then want to connect to another server, you must create a new socket. The other protocols do not have that limitation.

2.1.5 How the IP Protocols Fit Together

While interfacing with the network, you may wonder how all these protocols fit together. In some cases, it may seem that they don't fit together at all. They do use some of each other's features, but they really don't work so closely together that they are inseparable.

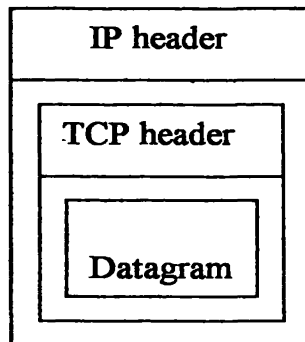


figure 2.4 how data packet physically fit together.

The raw IP, ICMP, UDP, and TCP protocols fulfill specific roles. You can use these roles to fit your needs when you design your network application. Of course, while TCP has more reliability and features than the other protocols, you cannot replace ICMP with TCP. Because the Linux subsystems require different features from TCP/IP, each packet type is important for your system to work correctly. The ICMP, UDP, and TCP packets physically rely on the raw IP packet (figure 2.4). Their headers and data reside in the IP's data section, following the IP header (figure 2.5).

IP header definition

```
typedef unsigned int uint;
```

```
typedef unsigned char uchar;
```

```
struct ip_header {
```

```
    uint header_len:4;           (header length in words in 32bit words)
```

```
    uint version:4;             (4-bit version)
```

```
    uint serve_type:8;          (how to service packet)
```

```
    uint packet_len:16;         (total size of packet in bytes)
```

```
    uint ID:16;                 (fragment ID)
```

```

uint frag_offset:13;      ( to help reassembly)
uint more_frags:1;      (flag for "more frags to follow")
uint dont_frag:1;      (flag to permit fragmentation )
uint __reserved:1;      (always zero )
uint time_to_live:8;    ( maximum router hop count )
uint protocol:8;        (ICMP, UDP, TCP )
uint hdr_chksum:16;     ( ones-comp. checksum of header)
uchar IPv4_src[IP_SIZE]; (IP address of originator )
uchar IPv4_dst[IP_SIZE]; ( IP address of destination)
uchar options[0];      (up to 40 bytes )
uchar data[0];         (message data up to 64KB)
};

```

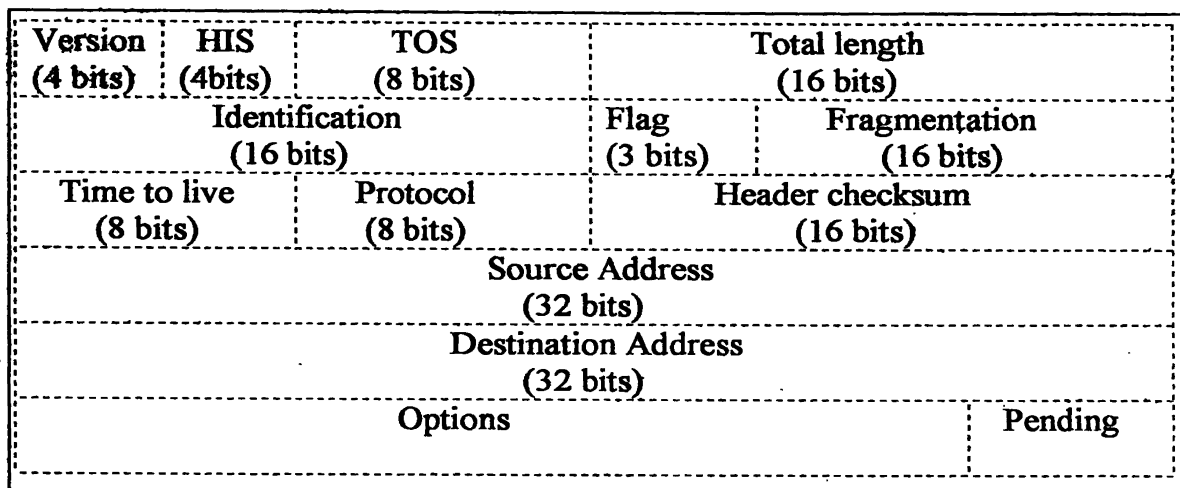


figure 2.5 structure of IP header.

2.2 Data packets and layers.

The application layer represents communication between two programs. The transport layer represents how this communication is delivered between the two programs. Programs are identified by numbers called service ports. The network layer represents how this communication is carried between the two end computers, or their individual network interface cards, are identified by numbers called IP addresses. the subnet layer represents how

this communication carried between each individual along the way. On the Ethernet network, these computer network interfaces are identified by numbers called Ethernet addresses which you are probably familiar with as your network card's burned-in hardware MAC address.

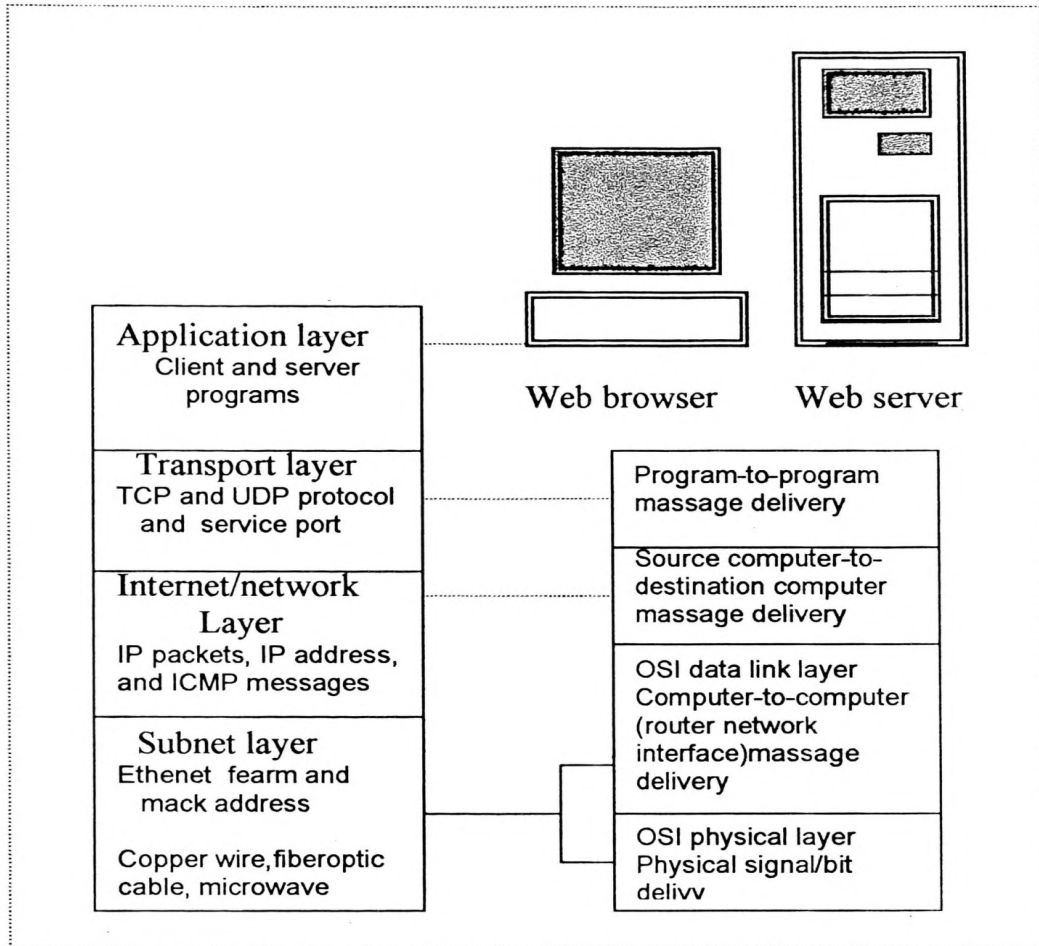


figure 2.6 different packet and different layers.

2.2.1 Packet filtering firewall and layers

Three of the most common meanings refer to a packet-filtering firewall. An application gateway also called a screened-host firewall, and an application-level circuit gateway also called a proxy firewall.

A packet filtering firewall is normally implemented within the operating system and operates at the IP network and transport protocol layers. it protects the system by making routing decisions after filtering packets based on information in the IP packet header.

An application gateway is implemented at the network architecture and system configuration levels. Network traffic is never passed through the application gateway machine. External access is allowed only to the gateway machine. Local user must log In to the gateway machine and access the internet from there. Additionally, the gateway Machine may be protected by packet-filtering firewalls on both its external and internal interfaces.

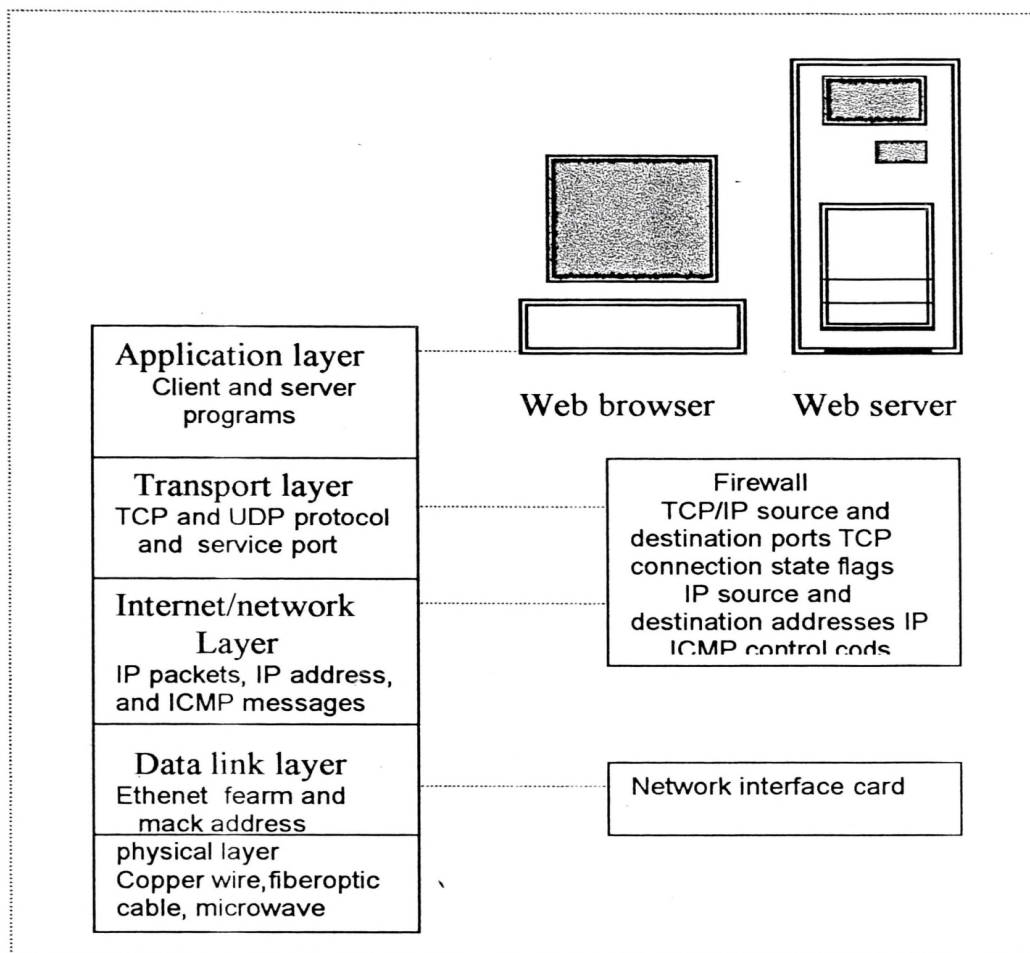


figure 2.7 packet filtering firewall.

2.3 Firewalls

A firewall is a structure intended to keep a fire from spreading. Building have firewalls made of brick walls completely dividing sections of the building. In a car a firewall is the metal wall separating the engine and passenger compartments. Internet firewalls are intended to keep the flames of Internet hell out of your private LAN. Or, to keep the members of your LAN pure and chaste by denying them access the all the evil Internet temptations.

The first computer firewall was a non-routing Unix host with connections to two different networks. One network card connected to the Internet and the other to the private LAN. To reach the Internet from the private network, you had to logon to the firewall (Unix) server. You then used the resources of the system to access the Internet. For example, you could use X-windows to run Netscape's browser on the firewall system and have the display on your work station. With the browser running on the firewall it has access to both networks.

This sort of dual homed system (a system with two network connections) is great if you can TRUST ALL of your users. You can simple setup a Linux system and give an account accounts on it to everyone needing Internet access. With this setup, the only computer on your private network that knows anything about the outside world is the firewall. No one can download to their personal workstations. They must first download a file to the firewall and then download the file from the firewall to their workstation. A firewall machine is all you need. *Set policies first.*

Types of Firewalls: There are two types of firewalls.

1. Filtering Firewalls - that block selected network packets.
2. Proxy Servers (sometimes called firewalls) - that make network connections for you.

2.3.1 Packet Filtering Firewalls

Packet Filtering is the type of firewall built into the Linux kernel. A filtering firewall works at the network level. Data is only allowed to leave the system if the firewall rules allow it. As packets arrive they are filtered by their type, source address, destination address, and port information contained in each packet.

Many network routers have the ability to perform some firewall services. Filtering firewalls can be thought of as a type of router. Because of this you need a deep understanding of IP packet structure to work with one. Because very little data is analyzed and logged, filtering firewalls take less CPU and create less latency in your network.

Filtering firewalls do not provide for password controls. User can not identify themselves. The only identity a user has is the IP number assigned to their workstation. This can be a problem if you are going to use DHCP (Dynamic IP assignments). This is because rules are based on IP numbers you will have to adjust the rules as new IP numbers are assigned.

Filtering firewalls are more transparent to the user. The user does not have to setup rules in their applications to use the Internet. With most proxy servers this is not true.

2.3.2 Proxy Servers

Proxies are mostly used to control, or monitor, outbound traffic. Some application proxies cache the requested data. This lowers bandwidth requirements and decreases the access the same data for the next user. It also gives unquestionable evidence of what was transferred. There are two types of proxy servers.

1. Application Proxies - that do the work for you.
2. SOCKS Proxies - that cross wire ports.

Application Proxy

The best example is a person telneting to another computer and then telneting from there to the outside world. With a application proxy server the process is automated. As you telnet to the outside world the client send you to the proxy first. The proxy then connects to the server you requested (the outside world) and returns the data to you.

Because proxy servers are handling all the communications, they can log everything they (you) do. For HTTP (web) proxies this includes very URL they you see. For FTP proxies this includes every file you download. They can even filter out "inappropriate" words from the sites you visit or scan for viruses.

Application proxy servers can authenticate users. Before a connection to the outside is made, the server can ask the user to login first. To a web user this would make every site look like it required a login.

SOCKS Proxy

A SOCKS server is a lot like an old switch board. It simply cross wires your connection through the system to another outside connection.

Most SOCKS server only work with TCP type connections. And like filtering firewalls they don't provide for user authentication. They can however record where each user connected to.

2.4 What is a socket

A socket is a connection between two computers in a TCP/IP network. Sockets are used to exchange information such as files, mails, websites etc. Many computer games also use sockets for multiplayer games. (figure 2.7).

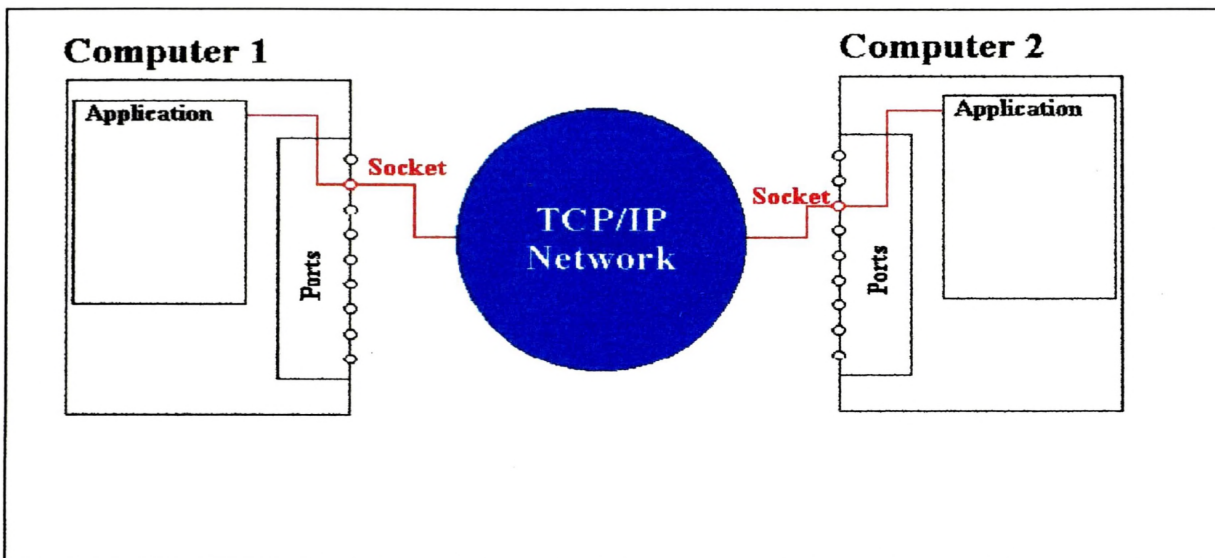


figure 2.7 transmit the data between two computer using socket

There are two types of sockets, TCP (Transmission Control Protocol) sockets and UDP (User Datagram Protocol) sockets. As you can see, these two types are called protocols. A protocol is a set of rules used to determine the way information is received and sent in a network. Other protocols are often used on top of TCP and UDP, for example, FTP is a protocol for file transfer which uses TCP to communicate with the file server. TCP and UDP themselves are built on top of the IP (Internet Protocol). The difference between these types is that in a TCP socket, the information sent is guaranteed to reach the recipient, this is not the case for UDP sockets, in this case, the information is not guaranteed to reach the recipient at all. Because of this, many UDP applications that uses datagram sockets to send information waits for the recipient to send a reply to check whether the transfer succeeded or not.

To connect a socket to another computer, you need to know its IP-address and which port you are going to connect to. An IP-address consists of four numbers between 0 and 255 (although 0 and 255 are reserved) separated by dots. For example, 195.125.13.54 Each computer has 65536 different ports. Some ports are reserved for different protocols. For example, port

number 80 is reserved for the HTTP (HyperText Transfer Protocol) protocol and 21 is reserved for FTP (File Transfer Protocol). All the ports under 1024 are reserved for specific protocols and should not be used by other protocols.

The next thing we do is to create a socket using the socket function. This function returns a SOCKET handle, which is used to refer to our socket when using other functions, the three arguments supplied are the address family, the socket type and the protocol used. The address family is AF_INET which means that we will use TCP or UDP, protocols of the AF_INET (or IP) address family. For AF_INET, there are three socket types, SOCK_STREAM, SOCK_DGRAM and SOCK_RAW, for this particular address family, SOCK_STREAM is TCP, SOCK_DGRAM is UDP and SOCK_RAW is used to send raw ip packets. For each of these socket types, you can choose which protocol you want to use for the socket (although you have to choose IPPROTO_UDP for SOCK_DGRAM and IPPROTO_IP for SOCK_STREAM).(diagram 2.1)

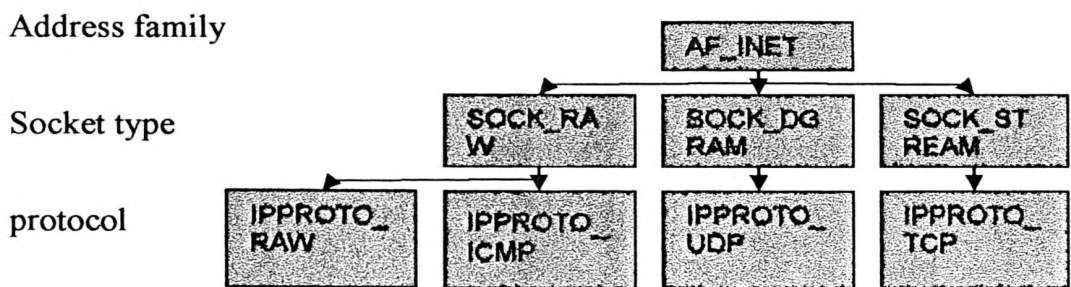


Diagram 2.1 internet protocol.

The UDP protocol is somewhat different from the TCP protocol, the most important difference is that it's message-based, that is you don't establish any "connections" to the receiver, you only send messages. The second difference is that UDP is an unreliable protocol, unlike TCP, UDP doesn't guarantee that your information reaches its destination. You can compare UDP with the postal service, you aren't guaranteed that the mail will reach its destination (although it often does). TCP however can be compared with a phone call, you exchange information simultaneously and you are guaranteed that the information is delivered.

2.5 Web server

First part looks at what a Web server is and how Web communication takes place. Basically for communication where there is a client-server flavor, the server process creates a socket and the client socket accesses the server through client socket techniques.

Socket: A socket is fundamentally nothing but an end point of communication. It can be of two types: Physical socket and Logical socket. In Logical socket operating system has its system calls, which creates them. Now for client-server access the socket needs three things to provide service or ask for service.

- 1) Service name (example: telnet)
- 2) Protocol (TCP-stream)
- 3) Port no (23)

The service uses protocol and protocol uses port number to provide service at server end and to get service at client end. Ultimately we find that the port number is mainly responsible for a client server communication. The protocols supported by Linux is shown by /etc/protocols and the services can be seen in /etc/services. Let's take few more examples then start with Web server.

1. telnet service uses TCP/IP protocol and communicate through port no. 23
2. ftp service uses TCP/IP protocol and communicate through 20,21 port numbers
3. www service uses http protocol and communicate through port no 80.

Web communication : Web communication deals with a browser type of client process and Web server type of server process. What actually happens when a user writes `http://www.yahoo.com`? Well, the browser transfers the URL to current machine's operating system with a destination address' operating system, which is responsible for extracting protocol i.e. "http" from the client socket (browsers) and then it packets data using layer software and over the packet it attaches the header http. This enables the remote machine to hand over the request to Web server of remote machine. Why so? Because there can be many a server running on the same machine so the particular services are distinguished by their protocol.

But how should we explain when telnet and ftp both are using same protocol but have different server Processes? The answer is that they are distinguished by their port numbers.

After this the operating system throws the data to network interface card through the ram and then network interface card gives it to nearest gateway, which sends the data to the server machine at server end.

The network card gives a signal back to operating system that a data enclosed with http header using TCP/IP header has arrived. One's operating system checks that data has http wrapper and searches for Web server on that machine. When it finds, it hands over the data and pays attention to other processes.

Before the Web server processes the data, it goes through a filtration by the gateway process implemented on the Web server, which actually filters the raw data. This concept implemented is called as common gateway interface that has the Web server environment variables, which stores the data in different variable. When the user asks for some unnecessary data, headers also get attached with data and so the need for filtration.

2.6 Internet layer addressing.

In TCP/IP terms, a network consists of host that are connected on a local network and do not need to use a router or gateway. Routers are used to forward packet from one network to another, connected on an internetwork.

Internet layer addresses represented by 32-bit IP addresses, which contain two parts, a network ID and host ID, separated in to four 8-bit logical section or octets, for example.

10100101 01101111 010101100 11000101

IP addresses are commonly represented in a dotted decimal notation, rather than as 32-bit address. For example, the preceding IP address may be more conveniently remembered as;

165.111.172.197

The assignment of octets to networked and hosted varies according to the class of IP addresses. The network field specifies which network the host is part of, whereas the host id field is used to allocate a unique identifier to each host on a given networks. five main classes exist, from A to E.

class	IP address range	PU/UNIX ask	IBM mask	NO of stations per network
A	1.x.x.x to 127.x.x.x	255.0.0.0	0.255.255.0	16 million+
B	128.x.x.x to 191.x.x.x	255.255.0.0	0.0.255.0	65000+
C	192.x.x.x to 254.x.x.x	255.255.255.0	0	253

Class D is used for multicasting by special protocol to transmit messages to a select groups of nodes, and class E is reserved for future use.

Special IP numbers certain IP numbers are reserved a hostid of all 0's or all 1's is never assigned to an individual host because a hostid of 0 refers to the network itself; for example, IP address 165.111.0.0 refer to the class B network 165.111

The netid 127 is used to loopback address. IP address 127.0.0.1 will be used test for the configuration of TCP/IP 127.0.0.1 simply refers to the local host, the most machine itself. Hostid of 255 are not permitted and are reserved for broadcast. A message sent on 165.111.225.225 is sent to every host on the network 165.111

Subnet masking: as already mentioned, a general IP address consists of two parts, a netid and hostid. If subnetting were not used, several netid would be required to identify different subnetworks. Instead, through the use of a subnet mask, the IP address can be divided into three parts: netid, subnetid, and hostid. A subnet mask is a 32-bit number in which a 1 is in the mask indicates that the corresponding bit in the IP address is part of the netid; a 0 indicates the bit as part of the hostid. In this way, subnetting uses sum of the bits in the hostid octets as a subnetid.

Subnet Masking on an octet Boundary: the subnet Mask 255.0.0.0, 255.255.0.0 and 255.255.255.0 Mask at octet boundaries, whereas the Mask 255.255.248.0 mask at a nonboundart. Consider the following example, which shows how the mask 255.255.255.0 cuts the IP address 165.111.172.197

IP address 10100101 01101111 10101100 11000101
Subnet Mask 11111111 11111111 11111111 00000000
Netid 10100101 01101111
Subnetid 10101100
Hostid 11000101

Subnet masking on a non-octet boundary subnet mask generally consist of a set of adjacent 1's followed by all 0's. Although subnet masks can consist of noncontiguous bits, this is rare and provides no advantage over adjacent bits. The eight most common subnet masks are 0,128,192,224,240,258,253,254,and 255.

For wxample, the subnet mask 255.255.248.0 cuts the IP address 165.111.172.197 at a non-octet boundary, as shown in the following example:

IP address :10100101 01101111 10101100 11000101
Subnet mask :11111111 11111111 11111000 00000000
Netid :10100101 01101111
Subnetid : 10101
Hostid : 100 11000101

2.7 Network intrusion detection system

An intrusion is somebody (A.K.A. "hacker" or "cracker") attempting to break into or misuse your system. The word "misuse" is broad, and can reflect something severe as stealing confidential data to something minor such as misusing your email system for spam (though for many of us, that is a major issue!). An "Intrusion Detection System (IDS)" is a system for detecting such intrusions. For the purposes of this FAQ, IDS can be broken down into the following categories:

network intrusion detection systems (NIDS): monitors packets on the network wire and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine, thus discovering if someone is attempting a TCP port scan. A NIDS may run either on the target machine who watches its own traffic (usually integrated with the stack and services themselves), or on an independent machine promiscuously watching all network traffic (hub, router, probe). Note that a "network" IDS monitors many machines, whereas the others monitor only a single machine (the one they are installed on).

system integrity verifiers (SIV): monitors system files to find when a intruder changes them (thereby leaving behind a backdoor). The most famous of such systems is "Tripwire". A SIV may watch other components as well, such as the Windows registry and chron configuration, in order to find well known signatures. It may also detect when a normal user somehow acquires root/administrator level privileges. Many existing products in this area should be considered more "tools" than complete "systems": i.e. something like "Tripwire" detects changes in critical system components, but doesn't generate real-time alerts upon an intrusion.

log file monitors (LFM): monitor log files generated by network services. In a similar manner to NIDS, these systems look for patterns in the log files that suggest an intruder is attacking. A typical example would be a parser for HTTP server log files that looking for intruders who try well-known security holes, such as the "phf" attack. Intruders can be classified into two categories.

Outsiders: Intruders from outside your network, and who may attack you external presence (deface web servers, forward spam through e-mail servers, etc.). They may also attempt to go around the firewall to attack machines on the internal network. Outside intruders may come from the Internet, dial-up lines, physical break-ins, or from partner (vendor, customer, reseller, etc.) network that is linked to your corporate network.

Insiders: Intruders that legitimately use your internal network. These include users who misuse privileges (such as the Social Security employee who marked someone as being dead because they didn't like that person) or who impersonate higher privileged users (such as using someone else's terminal). A frequently quoted statistic is that 80% of security breaches are committed by insiders.

2.7.1 How do intruders get into system

The primary ways a intruder can get into a system:

Physical Intrusion If a intruders have physical access to a machine (i.e. they can use the keyboard or take apart the system), they will be able to get in. Techniques range from special privileges the console has, to the ability to physically take apart the system and remove the disk drive (and read/write it on another machine). Even BIOS protection is easy to bypass: virtually all BIOSes have backdoor passwords.

System Intrusion This type of hacking assumes the intruder already has a low-privilege user account on the system. If the system doesn't have the latest security patches, there is a good chance the intruder will be able to use a known exploit in order to gain additional administrative privileges.

Remote Intrusion This type of hacking involves a intruder who attempts to penetrate a system remotely across the network. The intruder begins with no special privileges. There are several forms of this hacking. For example, a intruder has a much more difficult time if there exists a firewall on between him/her and the victim machine.

Note that Network Intrusion Detection Systems are primarily concerned with Remote Intrusion.

2.7.3 Why can intruders get into systems

Software always has bugs. System Administrators and Programmers can never track down and eliminate all possible holes. Intruders have only to find one hole to break in.

Software bugs: Software bugs are exploited in the server daemons, the client applications, the operating system, and the network stack. Software bugs can be classified in the following manner:

Buffer overflows: Almost all the security holes you read about in the press are due to this problem. A typical example is a programmer who sets aside 256 characters to hold a login username. Surely, the programmer thinks, nobody will ever have a name longer than that. But a hacker thinks, what happens if I enter in a false username longer than that? Where do the additional characters go? If they hackers do the job just right, they can send 300 characters, including code that will be executed by the server, and voila, they've broken in. Hackers find these bugs in several ways. First of all, the source code for a lot of services is available on the net. Hackers routinely look through this code searching for programs that have buffer overflow problems. Secondly, hackers may look at the programs themselves to see if such a problem exists, though reading assembly output is really difficult. Thirdly, hackers will examine every place the program has input and try to overflow it with random data. If the program crashes, there is a good chance that carefully constructed input will allow the hacker to break in. Note that this problem is common in programs written in C/C++, but rare in programs written in Java.

Unexpected combinations: Programs are usually constructed using many layers of code, including the underlying operating system as the bottom most layer. Intruders can often send input that is meaningless to one layer, but meaningful to another layer. The most common language for processing user input on the web is PERL. Programs written in PERL will usually send this input to other programs for further evaluation. A common hacking technique would be to enter something like `"| mail < /etc/passwd"`. This gets executed because PERL asks the operating system to launch an additional program with that input. However, the operating system intercepts the pipe '|' character and launches the 'mail' program as well, which causes the password file to be emailed to the intruder.

Unhandled input: Most programs are written to handle valid input. Most programmers do not consider what happens when somebody enters input that doesn't match the specification.

Race conditions: Most systems today are "multitasking/multithreaded". This means that they can execute more than one program at a time. There is a danger if two programs need to access the same data at the same time. Imagine two programs, A and B, who need to modify the same file. In order to modify a file, each program must first read the file into memory, change the contents in memory, then copy the memory back out into the file. The race condition occurs when program A reads the file into memory, then makes the change. However, before A gets to write the file, program B steps in and does the full read/modify/write on the file. Now program A writes its copy back out to the file. Since program A started with a copy before B made its changes, all of B's changes will be lost. Since you need to get the sequence of events in just the right order, race conditions are very rare. Intruders usually have to try thousands of times before they get it right, and hack into the system.

2.7.4. System configuration

System configuration bugs can be classified in the following manner:

Default configurations: Most systems are shipped to customers with default, easy-to-use configurations. Unfortunately, "easy-to-use" means "easy-to-break-in". Almost any UNIX or WinNT machine shipped to you can be hacked in easily.

Lazy administrators: A surprising number of machines are configured with an empty root/administrator password. This is because the administrator is too lazy to configure one right now and wants to get the machine up and running quickly with minimal fuss. Unfortunately, they never get around to fixing the password later, allowing intruders easy access. One of the first things an intruder will do on a network is to scan all machines for empty passwords.

Hole creation: Virtually all programs can be configured to run in a non-secure mode. Sometimes administrators will inadvertently open a hole on a machine. Most administration guides will suggest that administrators turn off everything that doesn't absolutely positively

need to run on a machine in order to avoid accidental holes. Note that security auditing packages can usually find these holes and notify the administrator.

Trust relationships: Intruders often "island hop" through the network exploiting trust relationships. A network of machines trusting each other is only as secure as its weakest link.

2.7.5 Password cracking

This is a special category all to itself.

Really weak passwords: Most people use the names of themselves, their children, spouse/SO, pet, or car model as their password. Then there are the users who choose "password" or simply nothing. This gives a list of less than 30 possibilities that an intruder can type in for themselves.

Dictionary attacks: Failing the above attack, the intruder can next try a "dictionary attack". In this attack, the intruder will use a program that will try every possible word in the dictionary. Dictionary attacks can be done either by repeatedly logging into systems, or by collecting encrypted passwords and attempting to find a match by similarly encrypting all the passwords in the dictionary. Intruders usually have a copy of the English dictionary as well as foreign language dictionaries for this purpose. They all use additional dictionary-like databases, such as names (see above) and lists of common passwords.

Brute force attacks: Similar to a Dictionary attack, an intruder may try all possible combinations of characters. A short 4-letter password consisting of lower-case letters can be cracked in just a few minutes (roughly, half a million possible combinations). A long 7-character password consisting of upper and lower case, as well as numbers and punctuation (10 trillion combinations) can take months to crack assuming you can try a million combinations a second (in practice, a thousand combinations per second is more likely for a single machine).

2.7.7 Sniffing unsecured traffic

Shared medium: On traditional Ethernet, all you have to do is put a Sniffer on the wire to see all the traffic on a segment. This is getting more difficult now that most corporations are transitioning to switched Ethernet.

Server sniffing: However, on switched networks, if you can install a sniffing program on a server (especially one acting as a router), you can probably use that information to break into client machines and trusted machines as well. For example, you might not know a user's password, but sniffing a Telnet session when they log in will give you that password.

Remote sniffing: A large number of boxes come with RMON enabled and public community strings. While the bandwidth is really low (you can't sniff all the traffic), it presents interesting possibilities.

2.7.8 Design flaws

Even if a software implementation is completely correct according to the design, there still may be bugs in the design itself that leads to intrusions.

TCP/IP protocol flaws: The TCP/IP protocol was designed before we had much experience with the wide-scale hacking we see today. As a result, there are a number of design flaws that lead to possible security problems. Some examples include smurf attacks, ICMP Unreachable disconnects, IP spoofing, and SYN floods. The biggest problem is that the IP protocol itself is very "trusting": hackers are free to forge and change IP data with impunity. IPsec (IP security) has been designed to overcome many of these flaws, but it is not yet widely used.

UNIX design flaws: There are number of inherent flaws in the UNIX operating system that frequently lead to intrusions. The chief problem is the access control system, where only 'root' is granted administrative rights. As a result,

2.7.9 How do intruders get password

Intruders get passwords in the following ways:

Clear-text sniffing: A number of protocols (Telnet, FTP, HTTP Basic) use clear-text passwords, meaning that they are not encrypted as they go over the wire between the client and the server. An intruder with a protocol analyzer can watch the wire looking for such passwords. No further effort is needed; the intruder can start immediately using those passwords to log in.

Encrypted sniffing: Most protocols, however, use some sort of encryption on the passwords. In these cases, the intruder will need to carry out a Dictionary or Brute Force attack on the password in order to attempt decryption. Note that you still don't know about the intruder's presence, as he/she has been completely passive and has not transmitted anything on the wire. Password cracking does not require anything to be sent on the wire as intruder's own machine is being used to authenticate your password.

Replay attack: In some cases, intruders do not need to decrypt the password. They can use the encrypted form instead in order to login to systems. This usually requires reprogramming their client software in order to make use of the encrypted password.

Password file stealing: The entire user database is usually stored in a single file on the disk. In UNIX, this file is `/etc/passwd` (or some mirror of that file), and under WinNT, this is the SAM file. Either way, once a intruder gets hold of this file, he/she can run cracking programs (described above) in order to find some weak passwords within the file.

Observation: One of the traditional problems in password security is that passwords must be long and difficult to guess (in order to make Dictionary and Brute Force cracks unreasonably difficult). However, such passwords are often difficult to remember, so users write them down somewhere. Intruders can often search a persons work site in order to find passwords written on little pieces of paper (usually under the keyboard). Intruders can also train themselves to watch typed in passwords behind a user's back.

Social Engineering: A common (successful) technique is to simply call the user and say "Hi, this is Bob from MIS. We're trying to track down some problems on the network and they appear to be coming from your machine. What password are you using?" Many users will give up their password in this situation. (Most corporations have a policy where they tell users to never give out their password, even to their own MIS departments, but this technique is still successful. One easy way around this is for MIS to call the new employee 6-months have being hired and ask for their password, then criticize them for giving it to them in a manner they will not forget :-)

2.7.10 What is a typical intrusion scenario

A typical scenario might be:

Step 1: outside reconnaissance The intruder will find out as much as possible without actually giving themselves away. They will do this by finding public information or appearing as a normal user. In this stage, you really can't detect them. The intruder will do a 'whois' lookup to find as much information as possible about your network as registered along with your Domain Name (such as `foobar.com`). The intruder might walk through your DNS tables (using 'nslookup', 'dig', or other utilities to do domain transfers) to find the names of your machines. The intruder will browse other public information, such as your public web sites and anonymous FTP sites. The intruder might search news articles and press releases about your company.

Step 2: inside reconnaissance The intruder uses more invasive techniques to scan for information, but still doesn't do anything harmful. They might walk through all your web pages and look for CGI scripts (CGI scripts are often easily hacked). They might do a 'ping' sweep in order to see which machines are alive. They might do a UDP/TCP scan/strobe on target machines in order to see what services are available. They'll run utilities like 'rcpinfo', 'showmount', 'snmpwalk', etc. in order to see what's available. At this point, the intruder has done 'normal' activity on the network and has not done anything that can be classified as an intrusion. At this point, a NIDS will be able to tell you that "somebody is checking door handles", but nobody has actually tried to open a door yet.

Step 3: exploit The intruder crosses the line and starts exploiting possible holes in the target machines. The intruder may attempt to compromise a CGI script by sending shell commands in input fields. The intruder might attempt to exploit well-known buffer-overflow holes by sending large amounts of data. The intruder may start checking for login accounts with easily guessable (or empty) passwords. The hacker may go through several stages of exploits. For example, if the hacker was able to access a user account, they will now attempt further exploits in order to get root/admin access.

Step 4: foot hold At this stage, the hacker has successfully gained a foot hold in your network by hacking into a machine. The intruder's main goal is to hide evidence of the attacks (doctoring the audit trail and log files) and make sure they can get back in again. They may

install 'toolkits' that give them access, replace existing services with their own Trojan horses that have backdoor passwords, or create their own user accounts. System Integrity Verifiers (SIVs) can often detect an intruder at this point by noting the changed system files. The hacker will then use the system as a stepping stone to other systems, since most networks have fewer defenses from inside attacks.

Step 5: profit The intruder takes advantage of their status to steal confidential data, misuse system resources (i.e. stage attacks at other sites from your site), or deface web pages.

Another scenario starts differently. Rather than attack a specific site, an intruder might simply scan random internet addresses looking for a specific hole. For example, an intruder may attempt to scan the entire Internet for machines that have the SendMail DEBUG hole. They simply exploit such machines that they find. They don't target you directly, and they really won't even know who you are. (This is known as a 'birthday attack'; given a list of well-known security holes and a list of IP addresses, there is a good chance that there exists some machine somewhere that has one of those holes).

2.7.11 What are some common intrusion signatures

There are three types of attacks:

reconnaissance These include ping sweeps, DNS zone transfers, e-mail recons, TCP or UDP port scans, and possibly indexing of public web servers to find cgi holes.

exploits Intruders will take advantage of hidden features or bugs to gain access to the system.

denial-of-service (DoS) attacks Where the intruder attempts to crash a service (or the machine), overload network links, overload the CPU, or fill up the disk. The intruder is not trying to gain information, but to simply act as a vandal to prevent you from making use of your machine.

2.7.12 What are some common exploits

CGI scripts: CGI programs are notoriously insecure. Typical security holes include passing tainted input directly to the command shell via the use of shell metacharacters, using hidden variables specifying any filename on the system, and otherwise revealing more about the

system than is good. The most well-known CGI bug is the 'phf' library shipped with NCSA httpd. The 'phf' library is supposed to allow server-parsed HTML, but can be exploited to give back any file. Other well-known CGI scripts that an intruder might attempt to exploit are: TextCounter, GuestBook, EWS, info2www, Count.cgi, handler, webdist.cgi, php.cgi, files.pl, nph-test-cgi, nph-publish, AnyForm, FormMail. If you see somebody trying to access one or all of these CGI scripts (and you don't use them), then it is clear indication of an intrusion attempt (assuming you don't have a version installed that you actually want to use).

2.7.13 Web server attacks

Beyond the execution of CGI programs, web servers have other possible holes. A large number of self-written web servers (include IIS 1.0 and NetWare 2.x) have hole whereby a file name can include a series of "../" in the path name to move elsewhere in the file system, getting any file. Another common bug is buffer overflow in the request field or in one of the other HTTP fields.

Web server often have bugs related to their interaction with the underlying operating system. An old hole in Microsoft IIS have been dealing with the fact that files have two names, a long filename and a short 8.3 hashed equivalent that could sometimes be accessed bypassing permissions. NTFS (the new file system) has a feature called "alternate data streams" that is similar to the Macintosh data and resource forks. You could access the file through its stream name by appending "::\$DATA" in order to see a script rather than run it.

Servers have long had problems with URLs. For example, the "death by a thousand slashes" problem in older Apache would cause huge CPU loads as it tried to process each directory in a thousand slash URL.

Web browser attacks

It seems that all of Microsoft's and Netscape's web browsers have security holes (though, of course, the latest ones never have any that we know about -- yet). This includes both URL, HTTP, HTML, JavaScript, Frames, Java, and ActiveX attacks.

URL fields can cause a buffer overflow condition, either as it is parsed in the HTTP header, as it is displayed on the screen, or processed in some form (such as saved in the cache

history). Also, an old bug with Internet Explorer allowed interaction with a bug whereby the browser would execute .LNK or .URL commands.

HTTP headers can be used to exploit bugs because some fields are passed to functions that expect only certain information.

HTML can be often exploited, such as the MIME-type overflow in Netscape Communicator's <EMBED> command.

JavaScript is a perennial favorite, and usually tries to exploit the "file upload" function by generating a filename and automatically hidden the "SUBMIT" button. There have been many variations of this bug fixed, then new ways found to circumvent the fixes.

Frames are often used as part of a JavaScript or Java hack (for example, hiding web-pages in 1px by 1px sized screens), but they present special problems. For example, I can include a link to a trustworthy site that uses frames, then replace some of those frames with web pages from my own site, and they will appear to you to be part of that remote site.

Java has a robust security model, but that model has proven to have the occasional bug (though compared to everything else, it has proven to be one of the most secure elements of the whole system). Moreover, its robust security may be its undoing: Normal Java applets have no access to the local system, but sometimes they would be more useful if they did have local access. Thus, the implementation of "trust" models that can more easily be hacked.

ActiveX is even more dangerous than Java as it works purely from a trust model and runs native code. You can even inadvertently catch a virus that was accidentally imbedded in some vendor's code.

SMTP (SendMail) attacks

SendMail is an extremely complicated and widely used program, and as a consequence, has been the frequent source of security holes. In the old days (of the '88 Morris Worm), hackers would take advantage of a hole in the DEBUG command or the hidden WIZ feature to break into SMTP. These days, they often try buffer overruns. SMTP also can be exploited in reconnaissance attacks, such as using the VRFY command to find user names.

Access Failed login attempts, failed file access attempts, password cracking, administrative powers abuse

IMAP Users retrieve e-mail from servers via the IMAP protocol (in contrast, SMTP transfers e-mail between servers). Hackers have found a number of bugs in several popular IMAP servers.

IP spoofing There is a range of attacks that take advantage of the ability to forge (or 'spoof') your IP address. While a source address is sent along with every IP packet, it isn't actually used for routing. This means an intruder can pretend to be you when talking to a server. The intruder never sees the response packets (although your machine does, but throws them away because they don't match any requests you've sent). The intruder won't get data back this way, but can still send commands to the server pretending to be you. IP spoofing is frequently used as part of other attacks:

SMURF Where the source address of a broadcast ping is forged so that a huge number of machines respond back to victim indicated by the address, overloading it (or its link).

TCP sequence number prediction

In the startup of a TCP connection, you must choose a sequence number for your end, and the server must choose a sequence number for its end. Older TCP stacks choose predictable sequence numbers, allowing intruders to create TCP connections from a forged IP address (for which they will never see the response packets) that presumably will bypass security.

2.7.14 DNS poisoning through sequence prediction

DNS servers will "recursively" resolve DNS names. Thus, the DNS server that satisfies a client request will become itself a client to the next server in the recursive chain. The sequence numbers it uses are predictable. Thus, an intruder can send a request to the DNS server and a response to the server forged to be from the next server in the chain. It will then believe the forged response, and use that to satisfy other clients.

Buffer overflow Some other buffer overflow attacks are:

DNS overflow Where an overly long DNS name is sent to a server. DNS names are limited to 64-bytes per subcomponent and 256-bytes overall.

statd overflow where an overly long filename is provided

DNS attacks DNS is a prime target because if you can corrupt the DNS server, you can take advantage of trust relationships.

DNS cache poisoning Every DNS packet contains a "Question" section and "Answer" section. Vulnerable servers will believe (and cache) Answers that you send along with Questions. Most, but not all, DNS servers have been patched as of November, 1998.

2.7.15 What are some common reconnaissance scans

Ping sweeps This simple scan simply pings a range of IP addresses to find which machines are alive. Note that more sophisticated scanners will use other protocols (such as an SNMP sweep) to do the same thing.

TCP scans Probes for open (listening) TCP ports looking for services the intruder can exploit. Scans can use normal TCP connections or stealth scans that use half-open connections (to prevent them from being logged) or FIN scans (never opens a port, but tests if someone's listening). Scans can be either sequential, randomized, or configured lists of ports.

UDP scans These scans are a little bit more difficult because UDP is a connectionless protocol. The technique is to send a garbage UDP packet to the desired port. Most machines will respond with an ICMP "destination port unreachable" message, indicating that no service is listening at that port. However, many machines throttle ICMP messages, so you can't do this very fast.

OS identification By sending illegal (or strange) ICMP or TCP packets, an intruder can identify the operating system. Standards usually state how machines should respond to legal packets, so machines tend to be uniform in their response to valid input. However, standards omit (usually intentionally) the response to invalid input. Thus, each operating system's

unique responses to invalid inputs forms a signature that hackers can use to figure out what the target machine is. This type of activity occurs at a low level (like stealth TCP scans) that systems do not log.

Account scans Tries to log on with accounts

- Accounts with no passwords
- Accounts with password same as username, or "password".
- Default accounts that were shipped with the product (a common problem on SGI, done to make setup easier)
- Accounts installed with software products (common on Microsoft as well as Unix, caused by products that run under their own special user account).
- Anonymous FTP problems (CWD ~root)
- Scan for rlogin/rsh/rexec ports, that may supported trusted logins.

2.7.16 What are some common DoS (Denial of Service) attacks

Ping-of-Death Sends an invalid fragment, which starts before the end of packet, but extends past the end of the packet.

SYN Flood Sends TCP SYN packet (which start connections) very fast, leaving the victim waiting to complete a huge number of connections, causing it to run out of resources and dropping legitimate connections. A new defense against this are "SYN cookies". Each side of a connection has its own sequence-number. In response to a SYN, the attacked machine creates a special sequence number that is a "cookie" of the connection then forgets everything it knows about the connection. It can then recreate the forgotten information about the connection when the next packets come in from a legitimate connection.

Land/Latierra Sends forged SYN packet with identical source/destination address/port so that system goes into infinite loop trying to complete the TCP connection.

WinNuke Sends OOB/URG data on a TCP connection to port 139 (NetBIOS Session/SMB), which cause the Windows system to hang.

2.7.17 How are intrusions detected

Anomaly detection

The most common way people approach network intrusion detection is to detect statistical anomalies. The idea behind this approach is to measure a "baseline" of such stats as CPU utilization, disk activity, user logins, file activity, and so forth. Then, the system can trigger when there is a deviation from this baseline.

The benefit of this approach is that it can detect the anomalies without having to understand the underlying cause behind the anomalies. For example, let's say that you monitor the traffic from individual workstations. Then, the system notes that at 2am, a lot of these workstations start logging into the servers and carrying out tasks. This is something interesting to note and possibly take action on.

Signature recognition

The majority of commercial products are based upon examining the traffic looking for well-known patterns of attack. This means that for every hacker technique, the engineers code something into the system for that technique.

This can be as simple as a pattern match. The classic example is to examine every packet on the wire for the pattern `"/cgi-bin/phf?"`, which might indicate somebody attempting to access this vulnerable CGI script on a web-server. Some IDS systems are built from large databases that contain hundreds (or thousands) of such strings. They just plug into the wire and trigger on every packet they see that contains one of these strings.

2.7.18 How does a NIDS match signatures with incoming traffic

Traffic consists of IP datagrams flowing across a network. A NIDS is able to capture those packets as they flow by on the wire. A NIDS consists of a special TCP/IP stack that reassembles IP datagrams and TCP streams. It then applies some of the following techniques:

Protocol stack verification A number of intrusions, such as "Ping-O-Death" and "TCP Stealth Scanning" use violations of the underlying IP, TCP, UDP, and ICMP protocols in order to attack the machine. A simple verification system can flag invalid packets. This can include valid, but suspicious, behavior such as severally fragmented IP packets.

Application protocol verification A number of intrusions use invalid protocol behavior, such as "WinNuke", which uses invalid NetBIOS protocol (adding OOB data) or DNS cache poisoning, which has a valid, but unusually signature. In order to effectively detect these intrusions, a NIDS must re-implement a wide variety of application-layer protocols in order to detect suspicious or invalid behavior.

Creating new loggable events A NIDS can be used to extend the auditing capabilities of your network management software. For example, a NIDS can simply log all the application layer protocols used on a machine. Downstream event log systems (WinNT Event, UNIX syslog, SNMP TRAPS, etc.) can then correlate these extended events with other events on the network.

2.7.19 What happens after a NIDS detects an attack

Reconfigure firewall: Configure the firewall to filter out the IP address of the intruder. However, this still allows the intruder to attack from other addresses. Checkpoint firewall's support a "Suspicious Activity Monitoring Protocol (SAMP)" for configuring firewalls. Checkpoint has their "OPSEC" standard for re-configuring firewalls to block the offending IP address.

Chime Beep or play a .WAV file. For example, you might hear a recording "You are under attack".

SNMP Trap Send an SNMP Trap datagram to a management console like HP OpenView, Tivoli, Cabletron Spectrum, etc.

DT Event Send an event to the WinNT event log.

Syslog Send an event to the UNIX syslog event system.

send e-mail Send e-mail to an administrator to notify of the attack.

Page Page (using normal pagers) the system administrator.

Log the attack Save the attack information (timestamp, intruder IP address, victim IP Address/port, protocol information).

Save evidence Save a tracefile of the raw packets for later analysis.

Launch program Launch a separate program to handle the event.

Terminate the TCP session Forge a TCP FIN packet to force a connection to terminate.

2.7.20 What other countermeasures besides IDS are there

Firewalls: Most people think of the firewall as their first line of defense. This means if intruders figure out how to bypass it (easy, especially since most intrusions are committed by employees inside the firewall), they will have free run of the network. A better approach is to think of it as the *last* line of defense: you should be pretty sure machines are configured right and intrusion detection is operating, and then place the firewall up just to avoid the wannabe script-kiddies. Note that almost any router these days can be configured with some firewall filtering. While firewalls protect external access, they leave the network unprotected from internal intrusions. It has been estimated that 80% of losses due to "hackers" have been internal attacks.

Authentication: You should run scanners that automated the finding of open accounts. You should enforce automatically strict policies for passwords (7 character minimum, including numbers, dual-case, and punctuation) using crack or built in policy checkers (WinNT native, add-on for UNIX). You can also consider single-sign on products and integrating as many password systems as you can, such as RADIUS/TACACS integration with UNIX or NT (for dial-up style login), integrating UNIX *and* WinNT authentication (with existing tools are the new Kerberos in Windows 2000). These authentication systems will help you also remove "clear-text" passwords from protocols such as Telnet, FTP, IMAP, POP, etc.

2.7.21 Where do I put IDS systems on my network

Network hosts: Even though network intrusion detection systems have traditionally been used as probes, they can also be placed on hosts (in non-promiscuous mode). Take for example a switched network where an employee is on the same switch as the CEO, who runs

Win98. The windows machine is completely defenseless, and has no logging capabilities that could be fed to a traditional host-based intrusion detection system. The employee could run a network-based password cracker for months without fear of being caught. A NIDS installed like virus scanning software is the most effective way to detect such intrusions.

network perimeter: IDS is most effective on the network perimeter, such as on both sides of the firewall, near the dial-up server, and on links to partner networks. These links tend to be low-bandwidth (T1 speeds) such that an IDS can keep up with the traffic.

WAN backbone : Another high-value point is the corporate WAN backbone. A frequent problem is hacking from "outlying" areas to the main corporate network. Since WAN links tend to be low bandwidth, IDS systems can keep up.

server farms : Servers are often placed on their own network, connected to switches. The problem these servers have, though, is that IDS systems cannot keep up with high-volume traffic. For extremely important servers, you may be able to install dedicated IDS systems that monitor just the individual server's link. Also, application servers tend to have lower traffic than file servers, so they are better targets for IDS systems.

LAN backbones: IDS systems are impractical for LAN backbones, because of their high traffic requirements. Some vendors are incorporating IDS detection into switches. A full IDS system that must reassemble packets is unlikely to keep up. A scaled-down system that detects simpler attacks but can keep up is likely to be a better choice.

2.7.22 How does IDS fit with the rest of my security framework

1. Put firewalls between areas of the network with different security requirements (i.e. between internet-localnet, between users-servers, between company-partners, etc).
2. Use network vulnerability scanners to double check firewalls and to find holes that intruders can exploit.
3. Use host policy scanners to make sure they conform to accepted practices (i.e. latest patches).
4. Use Network intrusion detection systems and other packet sniffing utilities to see what is actually going on.

5. Use host-based intrusion detection systems and virus scanners to flag successful intrusions.
6. Create an easy to follow policy that clearly states the response to intrusions.

2.7.22 How can I detect if someone is running a NIDS

A NIDS is essentially a sniffer .An example would be to do a traceroute against the victim. This will often generate a low-level event in the IDS. Traceroutes are harmless and frequent on the net, so they don't indicate an attack. However, since many attacks are preceded by traceroutes, IDSs will log them anyway. As part of the logging system, it will usually do a reverse-DNS lookup. Therefore, if you run your own DNS server, then you can detect when somebody is doing a reverse-DNS lookup on your IP address in response to your traceroute.

CHAPTER 3

Methodology

This program used to capture Internet packets network interface. Data packet not reassembled read their header data before arriving the firewall. The program was implemented in C language according to

1. Capture all packet from Ethernet interface.
2. IP filtering step. If local IP equal to destination IP from data packet go to next.
3. protocol filtering step. If protocol equal to TCP, dump TCP packet protocol equal to UDP, dump UDP and get destination port, else go to next
4. Port filtering step. If destination port does not equal to port number 80, print information about that data packet.
5. Write relevant data about intruder.

1. Dump data packet and get header data

```
void DumpPacket(char *buffer, int len) {
    struct ip_packet *ip=(void*)buffer;
    struct tcp_header*tcp=(void*)buffer;
    struct udp_header*udp=(void*)buffer;
    dump(buffer, len);
}
```

2. IP filtering step

```
char dst_ip[20];
strcpy(dst_ip,inet_pton(*(struct in_addr*)&ip->IPv4_dst));
Char localip[20]={'1','9','2','.','1','6','8','.','1','0','.','1','8','2'};
int i=0;
while((localip[i]==dst_ip[i])&&(i<15)){
    i++; } if(i>=14) {
```

3. Protocol filtering step

```
int A,B;
if(6==(ip->protocol))
    (A=ntohs(tcp->src_port))&&(B=ntohs(tcp->dst_port));
else if(17==(ip->protocol))
    (A=ntohs(udp->src_port))&&(B=ntohs(udp->dst_port));
else (A=0)&&(B=0);
```

4. port filtering step

```
if(80!=B) {
    printf("destination port =%d\n",B);
    printf("source port =%d\n",B);
}
```

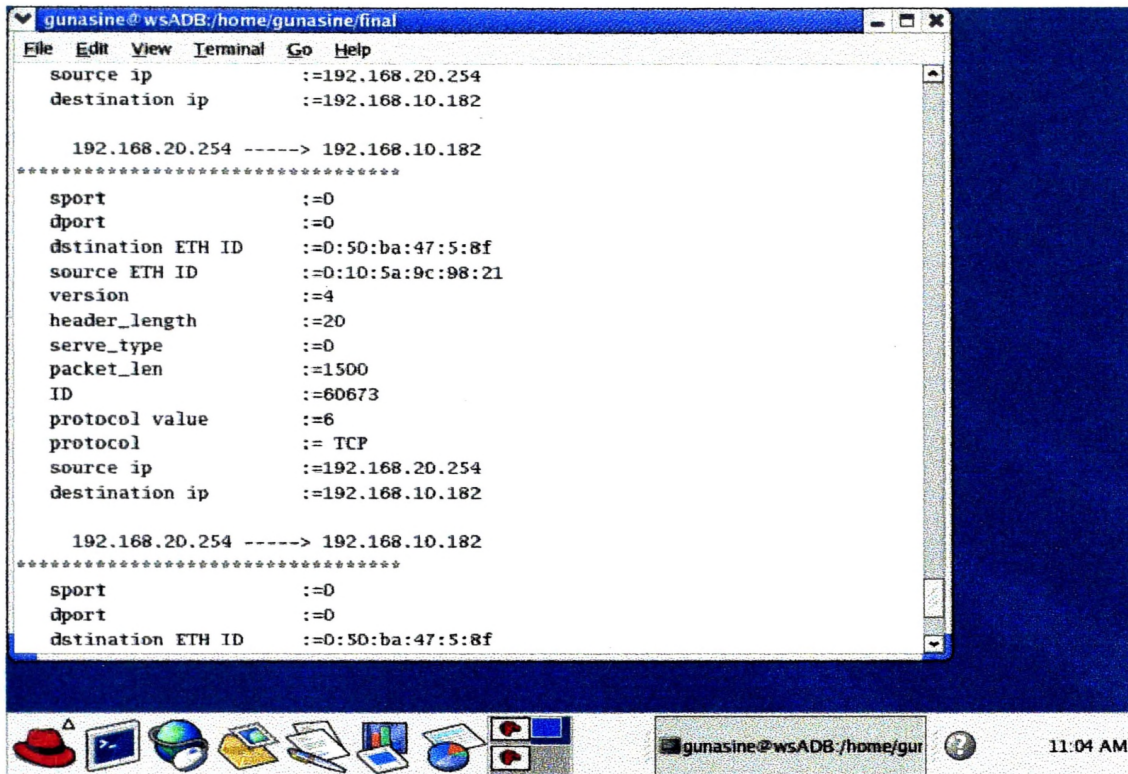
5. Write relevant data about intruder.

```
printf("checksum=%d, ", ntohs(ip->hdr_chksum));
PrintAddr("source=", ip->IPv4_src, eIP_ADDR);
PrintAddr(", destination=", ip->IPv4_dst, eIP_ADDR);
Char src_ip[20];
strcpy(src_ip,inet_nton(*(struct in_addr*)&ip->IPv4_src));
printf(" %s ----->%s",src_ip,dst_ip);
printf("\n");  fflush(stdout);
```

CHAPTER 4

4.1 Results

The program successfully identified unusual requested relevant data such as source IP address, source port etc. The output of the tested program shows in the following screen printout.



```
gunasine@ wsADB:/home/gunasine/final
File Edit View Terminal Go Help
source ip      :=192.168.20.254
destination ip :=192.168.10.182

192.168.20.254 -----> 192.168.10.182
*****
sport          :=0
dport          :=0
destination ETH ID :=0:50:ba:47:5:8f
source ETH ID  :=0:10:5a:9c:98:21
version        :=4
header_length  :=20
serve_type     :=0
packet_len     :=1500
ID             :=60673
protocol value :=6
protocol       := TCP
source ip      :=192.168.20.254
destination ip :=192.168.10.182

192.168.20.254 -----> 192.168.10.182
*****
sport          :=0
dport          :=0
destination ETH ID :=0:50:ba:47:5:8f
```

4.2 Discussion.

The type of Ethernet card had to be considered at beginning of coding the program. Programmable Ethernet IDs Some OEMs (original equipment manufacturers) offer their network interface cards (either PCI or PCMCIA) that support a programmable MAC address (or Ethernet ID). This makes mass production possible for a few card manufacturers while serving several hundred name-brand companies. Unfortunately, you may get a card that has a bogus ID, because the name-brand company did not program it correctly. This error can make your card non-unique on the network.

The following aspects have been used for the algorithm

1. **The roles of different types of data packets through the internet.**
2. **The role of IP addresses port numbers and socket.**
3. **Client server network programming.**
4. **The security of the web servers and other information servers.**
5. **The role of firewall and proxy servers.**
6. **How to works on the Linux operating system.**

CHAPTER 5

5.1 Conclusion

1. The developed system is a Network Monitoring Tool for Internet servers
2. It successfully detected unusual traffic other than http requests made to a web server
3. The system was successfully tested at www.sab.ac.lk (192.248.87.3) web server

5.2 Recommendation

Future development as follows

1. Implement a method to convert IP address to FQDN.
2. Implement a method to identify the unusual requested geographical area.
3. Implement a method to identify the time of unusual requested.

CHAPTER 6

References.

Neil, M. and Richard, S. 1999. Beginning Linux programming. Shroff publishers & distributors PVT LTD Mumbai Calcutta.

Tim, P. 1998. Teach yourself TCP/IP in 14 days. Techmedia munish plaza 20 Ansari road Darya Ganj, New Delhi-2.

Joe, C and Bob, W. 1999. SAMS Teach yourself TCP/IP in 24 hours. Techmedia munish plaza 20 Ansari road Darya Ganj. New Delhi-110 001.

Donglas, C.E and David, S.L. 2000. Internetworking with TCP/IP client server programming and application. Techmedia munish plaza 20 Ansari road Darya Ganj. New Delhi-110 001.

Robert, Z.L. 2000. Linux firewalls. Techmedia munish plaza 20 Ansari road Darya Ganj. New Delhi-110 002.

Richard, S.W. 2000. Unix Network programming. Addison Wesley Longman pte.Ltd.vol.1.

Steve, W. 1997. Building internets with internet information server and front page. Galgatio publications pvt.ltd. 5 Ansari road Darya Ganj. New Delhi-110 002.

Anone. 2000. <http://www.Ebcvg.com>. 25th November 2003.

Anone. 1999. <http://www.freeOS.com> 20th November 2003

National Digitization Project

National Science Foundation

Institute : Sabaragamuwa University of Sri Lanka

1. Place of Scanning : Sabaragamuwa University of Sri Lanka, Belihuloya

2. Date Scanned : ..2017-09-25.....

3. Name of Digitizing Company : Sanje (Private) Ltd, No 435/16, Kottawa Rd,
Hokandara North, Arangala, Hokandara

4. Scanning Officer

Name : ..B.A.C. Badarwan.....

Signature : .......

Certification of Scanning

I hereby certify that the scanning of this document was carried out under my supervision, according to the norms and standards of digital scanning accurately, also keeping with the originality of the original document to be accepted in a court of law.

Certifying Officer

Designation : ..Librarian.....

Name : ..T. N. Neerghsoorei.....

Signature : .......

Date : ..2017-09-25.....

Mrs. T. N. NEERGHSOOREI
(MSSc, PhD, ASLA, BA)
Librarian
Sabaragamuwa University of Sri Lanka
P.O. Box 02 Belihuloya, Sri Lanka
Tele: 094 45 2280045
Fax: 094 45 2280045

"This document/publication was digitized under National Digitization Project of the National Science Foundation, Sri Lanka"