

**STATISTICAL BASED DETECTION OF DOS
(DENIAL OF SERVICE) ATTACKS**

TARIQ. ABED. MOHAMAD

SEPTEMBER, 2012

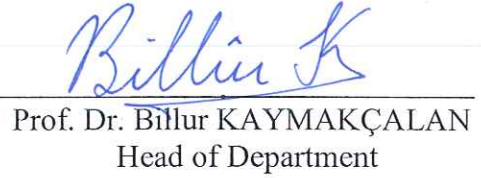
Title of Thesis : **Statistical Based Detection Of Dos (Denial Of Service) Attacks**

Submitted : **Tariq. Abed. Mohamad.**

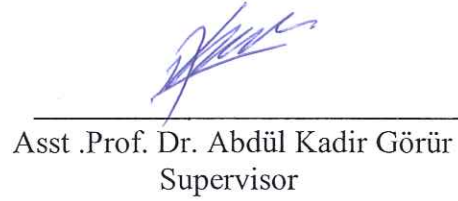
Approval of the Graduate School of Natural and Applied Sciences, Cankaya University


Prof. Dr Taner ALTUNOK
Director

I certify that this thesis satisfies all requirements as a thesis for the degree of Master of Science.


Prof. Dr. Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.


Asst. Prof. Dr. Abdül Kadir Görür
Supervisor

Examination Data: 17.09.2012

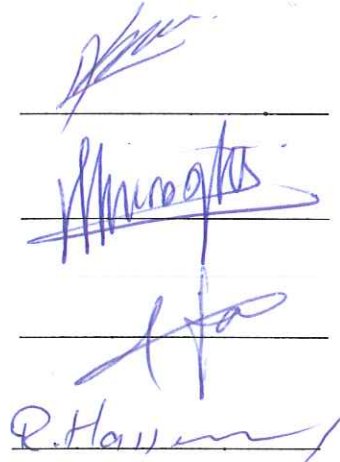
Examining Committee Members

Asst. Prof. Dr. Abdül Kadir Görür (Cankaya Univ.)

Asst. Prof. Dr. Bülent Gürsel Emiroğlu (Baskent Univ.)

Asst. Prof. Dr. Fahd JARAD (Cankaya Univ.)

Asst. Prof. Dr. Reza Zare Hassanpour (Cankaya Univ.)


R. Hassanpour

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and Presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work .

Name, Last Name: Tariq .Abed Mohamad

Signature

:



Date

:

17/09/2012

ABSTRACT
STATISTICAL BASED DETECTION OF DOS
(DENIAL OF SERVICE) ATTACKS

Tariq. Abed. Mohamad

M.Sc., Department of Mathematics and Computer Science

Supervisor: Asst. Prof. Dr. Abdul Kadir GORUR

September 2012, 112 Pages

Automated detection of anomalies in network traffic is an important and challenging task. Anomalies and intrusion are the main factors that affect the network performance. Detecting intruders that are aiming in degrading and preventing access to certain web pages or network address is the aim of many research works. In this work we propose an automated system to detect Denial of Service (DoS) attacks by using statistical methods. Related research work in this field will be surveyed and the aim will be the design of an enhanced method. Visual Basic .Net will be used in designing the programs. The proposed program will be tested on a real environment and its effectiveness will be verified.

Keywords: Statistical base. Denial of Service (DoS), (DDoS), IP Address, network traffic, IP information.

ÖZET
DOS İSTATİSTİKSEL TABANLI TESPİTİ
(Denial of Service) SALDIRILAR

Tariq. Abed. Mohamad

Yuksek Lisans, Matematik - Bilgisayar Bolumu

Danışman: Asst. Prof. Dr. Abdul Kadir GORUR

Eylül 2012, 112 Sayfa

Ağ trafiğindeki anomalilerin otomasyonlu tespiti önemli ve zorlayıcı bir görevdir. Ağın performansını etkileyen ana etkenler anomaliler ve izinsiz girişlerdir. Belirli internet sayfalarına ve ağ adreslerine erişimi azaltmayı ve engellemeyi amaçlayan izinsiz giriş yapanların tespiti birçok araştırma çalışmasının amacı olmuştur. Bu çalışmada, istatistiksel yöntemler kullanarak Hizmet Engelleme (DoS) saldırılarını tespit etmeye yönelik otomasyonlu bir sistem önermekteyiz. Bu alandaki ilgili araştırma çalışmaları gözden geçirilecektir ve amaç gelişmiş bir yöntemi tasarlamaktır. Programların tasarımında Visual Basic .Net kullanılacaktır. Önerilen program gerçek ortamda test edilecektir ve etkinliği doğrulanacaktır..

Anahtar Kelimeler: İstatistiksel taban. Hizmet Engelleme (DoS), (DDoS), IP Adresi, ağ trafiği, IP bilgileri.

ACKNOWLEDGEMENTS

Firstly the author wishes to express his deepest gratitude to my thesis supervisor Asst. Prof. Dr. Abdul Kadir GORUR and without his guidance and helps this thesis would not have been completed.

I would like to express my gratitude and appreciation to my Thesis Co-supervisor supervisor Asst. Prof. Dr. Reza Hassan pour for wisdom, foresight and the guidance he provided throughout the thesis completion process.

And, I would like to extend my thank to my special gratitude is to my (Mother) for their eternal love, support and trust in me. Without them I would never come up to this stage.

Finally, I offer my sincere thanks to my family and special my wife for their understanding and Support they provided during this study.

TABLE OF CONTENTS

STATISTICAL BASED DETECTION OF DOS ATTACKS.....	iii
ABSTRACT	vii
ÖZET	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
LIST OF SYMBOLS / ABBREVIATIONS.....	xii
CHAPTERS:	
1.1. Introduction.....	1
1.2. Definition of a Denial of Service Attack.....	2
1.3. What are the sorts of DOS attacks	4
1.3.1. Apache 2 Attack.....	4
1.3.2. Back Attack	4
1.3.3. Dosnuke Attack	4
1.3.4. Land Attack:	5
1.3.5. Mail bomb Attack.....	5
1.3.6. Neptune (SYN-Flood) Attack.....	5
1.3.7. Ping of Dead Attack.....	6
1.3.8. Process Table Attack.....	7
1.3.9. Smurf Attack.....	7
1.3.10. Syslogd Attack.....	7
1.3.11. Tcpreset Attack.....	8
1.3.12. Teardrop Attack.....	8
1.3.13. UdpstormAtta.....	8
1.3.14. Unintentional denial of service	9

1.4 Why should we care.....	10
1.5. Some Fast facts	10
1.6. DoS Shortfalls	11
2. Introduction.....	14
2.1. Background.....	14
2.2. Denial-of-Service Level II.....	17
2.3 Incidents and historical	17
2.4. Previous works and researches conducted on DOS similar methods and available solutions.....	21
2.5. Dos attacks: detection and prevention	32
2.5.1. DoS Attacks.....	33
2.5.2. Detection Approaches.....	35
5.2.1. ICMP Traceback.....	36
5.2.2. Packet Marking.....	37
2.5.3. Prevention Approaches.....	39
5.3.1. Ingress Filtering.....	39
5.3.2. Route-based Filtering.....	40
2.6. IP Addressing Scheme	41
2.7. Parts of an IP Address	41
2.7.1. Network Part.....	42
2.7.2. Host Part.....	42
2.7.3. Subnet Number (optional).....	42
2.8. Network Classes	43
2.8.1. Class A Network Numbers.....	43
2.8.2. Class B Network Numbers.....	44
2.8.3. Class C Network Numbers.....	44
2.9. Administering Network Numbers	45
2.10. Designing You IP Addressing Scheme	45
2.11. How IP Addresses Apply to Network Interfaces	47
2.12. Internet Protocol (IP)	47

2.13. IPv6.....	49
2.14. Private IP Addresses.....	51
2.15. Subnet Masks.....	54
3. The laws and theory are used.....	55
3.1. Introduction	55
3.2. The theory of the user in the system.	58
3.2.1. Theory I.....	58
3.2.2. Theory II.....	61
3.2.3. Theory III.....	63
3.2.4. Theory V	63
3.3. Definitions.....	65
3.3.1. Classes.....	65
3.3.2. Subnet mask default.....	66
3.3.3. Broadcast address of this subnet.....	66
3.4. Explain Planned IP information.....	67
4. Results, Applications And Software Code Application of program statistics of base attacks detection.....	69
4.1. INTRODACION TO PROGRAM	69
4.2. STATISTICAL BASED DETECTION OF DOS ATTACKS.....	72
4.2.1. Check IP-Address.....	73
4.2.2. View IP –Address Attack	84
4.2.3. Search Valid IP – Address	87
4.2.4. View Valid IP –Address.....	90
4.2.5. Network Information.....	92
5. CONCLUSIONS AND FUTURE WORK.....	108
5.1. CONCLUSIONS.....	108
5.2. FUTURE WORK	109
REFERENCES	110
APPENDIX	112

LIST OF TABLES

Table 1 Classification of DoS Attacks.....	12
Table 2 Countermeasures for DoS Attacks.....	13
Table 3 IPv6 Packet Header Forma.....	50
Table 4 Default Subnet Mas	54
Table 5 Class A network	65

LIST OF FIGURES

Figure 1 DoS attack graphlets.....	22
Figure 2 The framework of our Intrusion Detection System.....	23
Figure 3 AIDS system architecture.....	25
Figure 4 AIDS system architecture Simulation model in ns-2.....	27
Figure 5 Illustration of SIP signaling.....	28
Figure 6 UMTS Network Architecture.....	30
Figure 7 our experimental network environment for data collection.....	31
Figure 8 Different scenarios for DoS attacks. Attacker A1 launches an attack on the victim V. A1 spoofs IP address of host H5 from domain D5. Another attacker A3 uses host H3 as a reflector to attack V.....	34
Figure 9 IP headers.....	48
Figure 10 Design of System Public.....	57
Figure 11 Design of IP CHIK	60
Figure 12 Design of IP Time.....	62
Figure 13 Design of system STATISTICAL BASED daily.....	63
Figure 14 Design of system IP information	64
Figure 15 login the security a window.....	69
Figure 16 used a window adjustment.....	70
Figure 17 The Main Window Program.....	72
Figure 18 Check IP _Address.....	74
Figure 19 Views IP –Address Attack.....	85
Figure 20 IP ADDRESS CHART.....	87
Figure 21 Searches Valid IP – Address.....	88
Figure 22 View Table Valid IP –Address.....	90
Figure 23 Network Information Attack IP.....	92
Figure 24 Network Information Valid IP.....	100

LIST OF SYMBOLS / ABBREVIATIONS

3G	Third Generation
ACK	Acknowledgment
ARP	Address Resolution Protocol
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FP	False-Positive
FTP	File Transfer Protocol
HIDS	Host Based Intrusion Detection System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IOS	Internet Operating System
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Control address
NAT	Network Address Translation
NGMN	Next Generation Mobile Network
NIDS	Network Based Intrusion Detection System
NIC	Network interface card
OS	Operating System
POD	Ping Of Death
PPM	Probabilistic Packet Marking
RST	Reset Packet
SIP	Session Initiation Protocol

SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SPIE	Source Path Isolation Engine
SYN	Synchronize
TCP	Transmission Control Protocol
TELNET	Teletype Network
TF	True Positive
TTL	Time-to-Live
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System

CHAPTER I

1.1. INTRODUCTION

Nothing is more hindering your business than a network outage, and no one is protected to them. They lead to practical obstacles, and it takes time to deal with them. Thousands of dollars are lost every second your services are unavailable to your customers. You may have invested in the personnel or infrastructure to respond to an equipment failure, but what do you do when you see yourself a victim of a Denial of Service (DoS) or a Distributed DoS (DDoS) attack? Can you afford to be paralyzed by hackers? DoS and DDoS attacks deny legal users access to critical network services. Hackers conduct this by launching attacks that consume extreme network bandwidth, host processing cycles, or other network infrastructure resources. DoS attacks have resulted some of the world .s biggest companies to fail customers and investors as their Web sites became inaccessible to customers, partners, and users sometimes for up to twenty-four hours. For the victim, the influence can be comprehensive. Tools that empower DoS attacks are maturing to the point that even uncomplicated intruders could make risky damage. [4]

Instead, they merely use a massive amount of Internet resources, i.e., compromised hosts located between themselves and the victim (or victims), to send out a huge amount of useless packets toward the victim (or victims) simultaneously. The magnitude of the combined traffic is commonly sufficiently huge to jam, or even crash, the victim (system resource exhaustion), or its Internet links (bandwidth exhaustion), or both, then taking the victim off the Internet Often hackers of DoS attacks spoofed their attack packets' source addresses, and in DDoS attacks, each one sends a specific amount of packets to a victim or victims. Both make it very complicated to trace to the factual attackers. Based on a 2007 CSI Computer Crime and Security Survey, DoS attacks were in the top 5 amongst all attack kinds. 25 percent of respondents' computers had recognized DoS attacks. many companies'

loss were caused by DoS is \$2,888,600 which were ranked the top 7 among all attack categories. These show the severity of information security. [2]

1.2. DEFINITION OF A DENIAL OF SERVICE ATTACK:

Denial of service attacks is not modern, but the network made it lethal. The rudiment of this approach is not complicated and the attacker got rid of the devices with a stream of requests and orders that exceed the ability of the device provided on the treatment. It can be precise and uncomplicated examples of this approach when we continue to press the ENTER button on the input terminal (Terminal) have not yet logged on to the Network Log In, but linked to a specific type of hardware or workstations. The reason is that this method can be categorized into methods of denial of service attacks that the input button is in most circumstances; the start routine is to recognize the tool within the operating system, usually the routine implementation of high priority. And continued pressure on this button creates a high demand for the treatment process which is needed to identify the tool (keyboard of this case), leading to consumption of 100% of processor power and make it unable to receive requests for additional treatment. This drives to paralysis in the operating system, which usually does not have the intelligence to recognize between legitimate requests for access, and entry of harmful applications. In this situation there is no mechanism can correspond to this attack. And other approaches for this type of attack are to target resources in other fixed infrastructure, and examples of attacks that dump SYN. Within the meetings of the network (the Internet), the usual process like a handshake between systems, where the systems are to issue a demand to connect to another system using the package SYN (SYNchronization).

The host system in this case by issuing a package SYN-ACK, can respond to the demand coming from a particular IP address, and log this address in a particular table, and identify a specific period to disconnect if no response to this package occurs, and this must be in the formation of packet ACK issued by the first system. In the attacks of dumping, the attacker sends the biggest possible amount of packets SYN using IP addresses, fake, and the host system log responses to the packages SYN-ACK in the table, which remains there because the attacker does not

send packets ACK required, leading to full scale applications and its inability to receive any new connection requests.

Despite the damage that can inflict this kind of attack, the remedy can be implemented in two steps; first is to enlarge the size of the table that receives connection requests, and the second - a step is inherent to the first - to decreased the time required to respond to connection requests in order to omit the input is used more quickly. There is another kind of denial of service attacks, where the attacker uses a program to go into the test user accounts within a specific service through the experience of all user names, passwords and the wrong or deliberate use.

When using this software, some servers, if there is a specific delay between trials to enter, you prevent legitimate users to access the system. There is also another method of attack is called "packets watery Teardrop" where the attacker sends an assertive distorted order that leads to collapse of the processing of IP addresses on the device supplier. Similarly, there is a method of dumping the treatment process itself in the operating system by sending commands handling or the introduction of long (much longer than it is permitted by the operating system or application) Buffer Overflow, not the processing of inputs within the operating system Besdha (a loophole exploited by the authors of virus code Red Code Red in the Microsoft servers and operating systems) lead to the collapse of the system. [12]

Denial of service (DoS) is a type of attack in which a hacker issues a big amount of packets to freeze specific servers' services, consequently blocking legal users from normal access to the services. Distributed DoS (DDoS) attacks are another forms of DoS attacks in which a host or hosts complain from receiving a huge amount of packets issued by zombies. DDoS attacks often do not depend on particular network protocols or system weaknesses [2]

1.3. WHAT ARE THE SORTS OF DOS ATTACKS?

1.3.1. Apache2 Attack:

The apache2 attack is a denial of service attack against an apache web server. In this attack, plenty of http requests with http headers are sent to the web server. Since the server receives many of these http requests, it slows down, and in the long run it might crash a typical http request involves twenty or fewer headers. However, every http request which is submitted as part of this exploit involves many http headers.

despite the exact number and value of these headers can be varied between attack instances, commonly the number of headers is 500 times greater than the number of headers in a normal request. Moreover, the real component of the header is not important for the exploit, the exploit only depends on the fact that http request consists of many headers [8-16]

1.3.2. Back Attack:

Back attack is a denial of service attack against the apache web server. This attack Makes malformed web requests to port 80 with the payload, “\Get //// ...” followed by many slashes Since the server tries to process these requests, it will slow down and becomes unable to process other requests. An intrusion detection system can detect back any attack by checking the payloads of network packets. hence, the requests which are carrying more number of front slashes in payload should be considered an attack [8-16].

1.3.3. DoSnuke Attack:

DoSnuke is a denial service attack against NetBIOS. In this attack, “out-of-band” the (MSG_OOB) data that is delivered with higher priority than ordinary data is sent to NetBIOS port of the victim machine. These packets which are being sent by the attacking machines are flagged "urg". When a Windows system receives a packet with the "urg" flag set, it expects the data that will follow that flag. The exploit consists of setting the flag"urg", but not does not follow it with data. It can be detected by searching the sniffed data for a NetBIOS handshake followed by NetBIOS packets with the "urg" flag set. [8-19]

1.3.4. Land Attack:

The Land attack is a denial of service attack that is effective against some older TCP/IP implementations. The Land attack takes place when an attacker sends a spoofed TCP SYN packet that contains same IP address as the source and the destination addresses. Such a packet locks the victim system completely. However, it is quite noticeable, because IP packets with identical source and destination addresses should never exist on a properly working network. [8-16].

1.3.5. Mail bomb Attack:

Mail bomb is an attack in where the attacker sends many messages to a server, overflowing that server's mail queue and might possibly causing system failure. An intrusion detection system that is looking for a mail bomb attack can look for thousands of mail messages coming from or sent to a particular user within a short period of time. However, the aim of this type of attack is not to generate a high traffic volume, but to create a large size of messages in the server's queue. Therefore, via looking only at the IP header, a normal host might exhibit behavior similar to that of an attacker since it could send mail with large attachments to the SMTP server [8-16].

1.3.6. Neptune (SYN-Flood) Attack:

In order to establish a TCP connection between a client and server systems, a set sequenced of messages is required to be exchanged. Firstly, the client system sends a SYN message to the server to request a TCP connection. The server then acknowledges the SYN message by sending SYN-ACK message to the client. The client then completes establishing the connection by responding with an ACK message. The connection between the client and the server at that time is open, and the service-specific data can be exchanged between the client and the server the potential for abuse arises at the point where the server system has sent an

acknowledgment (SYN-ACK) back to client but has not received the ACK message yet. This is what we mean by half-open connection.

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

A Neptune attack can be distinguished from normal network traffic by looking seeking high number of simultaneous SYN packets determined for a particular machine that are coming from an unreachable host [8-23-16].

1.3.7. Ping Of Dead Attack:

A ping of death (abbreviated "POD") is a denial service attack, in which attacker sends a malformed or a malicious ping to a victim system. A ping is normally 64 bytes in size and carried by the ICMP protocol. Many computer systems do not have the ability to handle a ping larger than the maximum IP packet size, namely 65,535 bytes. Sending a 65,536 byte ping packet is illegitimate according to networking protocol, but a packet of such a size can be sent if it is fragmented; when the target computer recollects the packet, a buffer overflow can take place. Possible reactions include crashing, freezing, and rebooting . An attempted ping of death can be

identified by examining the size of all ICMP packets and flagging those that are longer than 64000 bytes [8-20-16].

1.3.8. Process Table Attack:

Process table attack is a type of DoS attack, which exploits the feature of some network services to generate a new process each time a new TCP/IP connection is set installed. The attacker attempts to make as many as possible uncompleted connections to the victim in order to oblige the victim's system to generate ample processes. Hence, because the number of processes that are running on the system cannot be unlimitedly large, the attack renders the victim would be unable to serve any other request. An intrusion detection system that is trying to detect a process table attack will need to use somewhat subjective criteria for identifying the attack, since this attack comprises of abuse of a perfectly legal action [8-19-16].

1.3.9. Smurf Attack:

Smurf attack is a denial-of-service attack, in which ICMP echo request packets are exploited. The attacker sends ICMP echo request packets to the broadcast address of many subnets with the source address spoofed to the intended victim. Any machines that are listening to these subnets will respond by sending ICMP "echo reply" packets to the victim. So, this attack results in a large ongoing flow of "ECHO" replies that flood the victim. The surf attack can be identified by checking if there is a large number of "echo replies" being sent to a particular victim machine from many various places, but no "echo requests" originating from the victim machine [8-16].

1.3.10. Syslogd Attack:

The Syslogd exploit is a denial-of-service attack that lets an attacker remotely eliminate the sys-logged service on a Solaris server. Under Solaris, if syslogd receives a packet containing an irresolvable IP address, it will crash with a

segmentation fault This attack can be recognized with a network-monitoring intrusion detection system, Which examines packets destined for the sys-log port whether the source address is Irresolvable or not, in the fact, it may not be authentic for an intrusion detection system to check every packet destined for the syslog[8-16].

1.3.11. Tcpreset Attack:

Tcpreset attack is a denial-of-service attack, in which the attacker intends to eradicate all TCP connections to a host. The network is monitored for TCP connection requests to the victim. As soon as such a request is discovered, the malevolent attacker sends a spoofed TCP RESET packet to the victim and forces it to terminate the TCP connection [8].

In order to detect this attack, TCP session/takedown rate can be checked and then especially concentrated on cases in which RESET packets appear to come from the machine that had initially attempted to start the connection [8-19].

1.3.12. Teardrop Attack:

In this attack, IP fragments with overlapping and oversized payloads are sent to the target machine. A bug in the TCP/IP fragmentation re-assembly code of various operating systems led to the fragments to be inappropriately controlled, crashing them as a result of this [8-21].

1.3.13. Udpstorm Attack:

An Udpstorm attack is a denial of service attack exploits two victim machines that Unconsciously attack each other. Firstly, the attacker forges a single packet that has been spoofed to seem that it is coming from the echo port on the first victim machine and sends it to the second victim. The echo service aimlessly responds to any request it receives by simply echoing the data of the request back to the machine and port that sent the echo request, so when the victim receives this spoofed packet it sends a response to the echo port of the second victim. This second victim responds

identically, and the loop of traffic persists. therefore, network congestion and slowdown occur. Once the loop of network traffic is initiated, an intrusion detection system that can notice network traffic on the inside of the network can note that traffic is being sent from the chargen or echo port of one machine to the chargen or echo port of another [8-16].

1.3.14. Unintentional Denial Of Service:

This describes a situation where a website ends up denied, not due to a deliberate attack by a single individual or group of individuals, but simply due to a sudden enormous spike in popularity. This can take place when an extremely popular website posts a prominent link to a second, less well-prepared site, for instance, as part of a news story. The upshot is that a significant proportion of the primary site's regular users — potentially hundreds of thousands of people — click that link in the space of a few hours, having the same impact on the target website as a DDoS attack. A VIPDoS is the same, but specifically when the link was posted by a celebrity.

An example of this occurred when Michael Jackson died in 2009. Websites such as Google and Twitter slowed down or even crashed.¹ Many sites' servers thought the requests were from a virus or spyware attempting to cause a Denial of Service attack, warning users that their queries looked like "automated requests from a computer virus or spyware application". [22]

1.4.WHY SHOULD WE CARE?

As per 2006 CSI/FBI Computer Crime and Security Survey

- 25% of respondents faced some form of DoS attacks in previous 12 months. This value varied from 25% to 40% over the course of time.
- DoS attacks are the 5th most costly form of attacks A DoS attack is not just missing out on the latest sports scores or Tweets or weather reports Internet is now a critical resource whose disruption has financial implications, or even dire consequences on human safety.
- Cybercrime and cyber warfare might use of DoS or DDoS as a potential weapon to disrupt or degrade critical infrastructure.
- DDoS attacks are major threats to the firmness of the Internet. [5]

1.5.SOME FAST FACTS ?

- In Feb 2000, series of massive DoS attacks incapacitated several high-visibility Internet e-commerce sites, including Yahoo, EBay and E*TRADE.
- In Jan 2001, Microsoft's name servers infrastructure disabled 98% and the legitimate users could not get access any Microsoft's servers.
- In Sept 2001, an attack by a UK-based teenager on the port of Houston's Web server, made weather and scheduling information unavailable, as a result No ships could dock at the *world's 8th busiest maritime* facility due to lack of weather and scheduling information Entire network performance was affected.
- In Oct 2002, all Domain Name System servers were attacked, and the Attack lasted only an hour 9 of the 13 servers were seriously affected.
- In Aug 2009, the attack on Twitter and Facebook. [5]

1.6.DOS SHORTFALLS:

- DoS attacks are unable to attack large bandwidth websites – one upstream client cannot produce enough bandwidth to cripple major megabit websites.
- New distributed server architecture makes it more difficult for one DoS to take down an entire site.
- New software protections neutralize existing DoS attacks quickly
- Service Providers know how to avoid these attacks from effecting their networks.
- “Old” Internet Technology – something new needs to take it’s place (Hackers want the challenge of a new technology). [10]

Table 1 Classification of DoS Attacks ^{[10][14]}

Attack	Affected Area	Example	Description
Network Level Device	Routers, IP Switches, Firewalls	Ascend Kill II, “Christmas Tree Packets”	Attack tries to exhaust hardware resources using multiple duplicate packets or a software bug.
OS Level	Equipment Vendor OS, End-User Equipment.	Ping of Death, ICMP Echo Attacks, Teardrop	Attack takes advantage of the way operating systems implement protocols.
Application Level Attacks	Finger Bomb	Finger Bomb, Windows NT RealServer G2 6.0	Attack a service or machine by using an application attack to exhaust resources.
Data Flood (Amplification, Oscillation, Simple Flooding)	Host computer or network	Smurf Attack (amplifier attack) UDP Echo (oscillation attack)	Attack in which massive quantities of data are sent to a target with the determination of using up bandwidth/processing resources.
Protocol Feature Attacks	Servers, Client PC, DNS Servers	SYN (connection depletion)	Attack in which “bugs” in protocol are utilized to take down network resources. Methods of attack include: IP address spoofing, and corrupting DNS server cache.

Table 2 Countermeasures for DoS Attacks^{[10][14]}

Attack	Countermeasure Options	Example	Description
Network Level Device	Software patches, packet filtering	Ingress and Egress Filtering	Software upgrades can repair known bugs and packet filtering can avoid attacking traffic from entering a network.
OS Level	SYN Cookies, drop backlog connections, shorten timeout of the time	SYN Cookies	Shortening the backlog time and dropping backlog connections will free up resources. SYN cookies proactively prevent attacks.
Application Level Attacks	Intrusion Detection System	GuardDog, other vendors.	Software used to detect illicit activity.
Data Flood (Amplification, Oscillation, Simple Flooding)	Replication and Load Balancing	Akami/Digital Island provide content distribution.	Extend the volume of content under attack makes it more complex and more difficult for attackers to identify services to attack and accomplish attacks completely.
Protocol Feature Attacks	Extend protocols to support security.	ITEF standard for itrace, DNSSEC	Trace source/destination packets by means other than the IP address (blocks against IP address spoofing). DNSSEC would provide authorization and authentication on DNS information.

CHAPTER II

2. INTRODUCTION

2.1. BACKGROUND

Feinstein et al. [14] used chi-square formula to detect DoS attacks. The authors classify network connections into six groups, called previous groups, according to packets' receiving frequencies, and calculate the total frequency of normal connections for each group. When detecting attacks, the detector again classifies frequencies of current connections, and classify them into six groups, called current groups. It then compares the frequency of previous group i and that of current group I with chi-square statistic method to see whether or not the difference is significant.

If so, we can suspect that there is a DoS or DDoS attack. However, several problems exist in this system. The first is that there are only six groups. For a huge institute or organization, the number should be increased. Otherwise, too many IPs are in a group, resulting in higher false positives. The second, there is no way to identify who the backers are once a DoS/DDoS is discovered. To solve this problem, we create a mechanism to do this. The third, if hackers issue a DoS/DDoS attack with a specific protocol, e.g., ICMP flooding, due to only occupying a portion of total packets, the attack is hard to be detected. The fourth is before calculating chi-square statistics, the authors grouped IPs based on their current connection frequencies rather than following the classification of previous groups. That is, a packet which is classified into previous group i may be classified into current group j , $1 \leq i, j \leq 6, i \neq j$, again resulting in the fact that we only know that there is a DoS/DDoS attack, but do not know who is issuing the attack. The final problem is the authors failed to consider

network consumption attacks. They only considered resource consumption attack. In fact, network consumption attack can be detected by the same method. [2]

Modern DoS attacks employ many advanced and sophisticated techniques to amplify the damage and elude detections or mitigations of countermeasures. IP spoofing is widely adopted by hackers to mask the real source of attacks, or launch reflective DoS attacks; Distributed DoS is used to initiate attacks from multi-source; low-rate pulsing method is utilized to reduce average packet rate and evade network monitors. Based on a header analysis, frequency domain characteristics are studied to improve the IDS performance [12][19], a ramp-up behavior is also considered as a way to distinguish between single- or multi-source attacks. In [20][22], authors propose to take a spectral analysis to detect shrew attacks which consist of short time bursts repeating at a maliciously chosen low frequency. This kind of low-rate attack sends out packets at certain fixed intervals, to intentionally reduce the average packet rate, rendering the IDS unable to discover undergoing attacks. To defend against IP spoofings, various off-line IP trace-back techniques are proposed to pinpoint the real origin of DoS attack [7][17], some on-line countermeasures are also developed to filter out those spoofed packets, help sustain service availability during attacks [23] presents a Hop-Count Filtering scheme to utilize the Time-to-Live(TTL) value in the IP header to filter out spoofed IP packets. Recent work on intrusion countermeasures include machine learning IDS techniques, alert correlation, alert fusion and feature analysis. *Machine learning techniques*, such as decision tree, neural network, Bayesian network, are applied to detect network intrusions. *Alert correlation* attempts to correlate IDS alerts based on the similarity between alert attributes, previously known attack scenarios, or prerequisites and consequences of known attacks [13].

Alert fusion combines detection outputs of the same attack from different independent detectors. *Feature Analysis* tries to optimize the information gained from multiple dimensional features through feature bagging, relevance and redundancy analysis, and feature weight classification [5][14][1][3].

In the HIDS literature, various techniques utilizing system call tracking and auditing trails are proposed. System call arguments are integrated to capture data-flow

behaviors of programs, and improve attack detections in HIDS [2]. A policy-driven solution is presented in [4] to define and enforce process behavior rules controlling processes' access to system resources. All system behaviors are monitored in real-time by a modified kernel. Basically, research works investigating DoS attack utilize sniffer-based methodologies. They only rely on analyzing network traffic information at the application level.

These network-based schemes suffer from fast traffic, switched network, information encryption, and most importantly, they have little knowledge of what is really going on in the victim machine. Significant useful information on the victim host is neglected. HIDS against DoS attacks are not widely researched since it is difficult to find a generic and low-cost way to defend against such attacks. We propose to utilize the strong correlation of architectural behaviors with DoS attacks, and employ multi-layer features to construct an IDS model. Close to our work, Woo and Lee [11] have observed performance degradation of multi-threaded workload under architectural DoS attacks. However, they do not further study the correlation of architectural behavior and DoS attacks and apply into an IDS in identifying and preventing such attacks. In our work, we are exploring architecture features to enrich the existing feature set used for intrusion detection research and demonstrate its effectiveness in a systematic approach.[1] The "Session-Expires" extension header is proposed in RFC 4028 [2] as a keep-alive mechanism for SIP. In the "security considerations" section of the RFC, only attacks through setting very small session timers are addressed, where an attacker may force compliant user agents to frequently send session refreshes at a rapid rate. The RFC proposed a 422 (Session Interval Too Small) response to reject the attacker's request if the timer is smaller than the value specified in the "Min-SE" header. However, there is no enforcement of how large the session timer can be and we identify a novel resource drained attack by utilizing this fact.

Also, the victim of our identified attack is the SIP proxy server rather than the user agents as considered in the RFC. Surveys of the DoS attacks in VoIP can be found in [19]–[20]. The SIP flooding attack is one of the major threats because it is easy to

launch and capable of quickly overloading both the network and nodes. However, existing work such as [21], [22] can efficiently detect the attack.

The attack utilizing the open-ended implementation of SIP is another major problem on VoIP networks [20]. Attackers modify SIP header values and excessively consume processing capability of SIP nodes. The attack identified in this paper belongs to this category. We propose a detection scheme based on the statistical Anderson-Darling test to deal with the attack.[6]

2.2.DENIAL-OF-SERVICE LEVEL II

The goal of DoS L2 (possibly DDoS) attack is to cause a launching of a defense mechanism which blocks the network segment from which the attack originated. In case of distributed attack or IP header modification (that depends on the kind of security behavior) it will fully block the attacked network from Internet, but without system crash.

2.3.INCIDENTS AND HISTORICAL

- The first major attack involving DNS servers as reflectors occurred in January 2001. The target was Register.com.[9] This attack, which forged requests for the MX records of AOL.com (to amplify the attack) lasted about a week before it could be traced back to all attacking hosts and shut off. It used a list of tens of thousands of DNS records that were a year old at the time of the attack.
- In February 2001, the Irish Government's Department of Finance server was hit by a denial of service attack carried out as part of a student campaign from NUI Maynooth. The Department officially complained to the University authorities and a number of students were disciplined.
- In July 2002, the Honeynet Project Reverse Challenge was issued. The binary that was analyzed turned out to be yet another DDoS agent, which implemented several DNS related attacks, including an optimized form of a reflection attack.
- On two occasions to date, attackers have performed DNS Backbone DDoS Attacks on the DNS root servers. Since these machines are intended to provide service to all

Internet users, these two denial of service attacks might be classified as attempts to take down the entire Internet, though it is unclear what the attackers' true motivations were. The first occurred in October 2002 and disrupted service at 9 of the 13 root servers. The second occurred in February 2007 and caused disruptions at two of the root servers.

■ In February 2007, more than 10,000 online game servers in games such as Return to Castle Wolfenstein, Halo, Counter-Strike and many others were attacked by the hacker group RUS. The DDoS attack was made from more than a thousand computer units located in the republics of the former Soviet Union, mostly from Russia, Uzbekistan and Belarus. Minor attacks are still continuing to be made today. *[citation needed]*

■ In the weeks leading up to the five-day 2008 South Ossetia war, a DDoS attack directed at Georgian government sites containing the message: "win+love+in+Russia" effectively overloaded and shut down multiple Georgian servers. Websites targeted included the Web site of the Georgian president, Mikhail Saakashvili, rendered inoperable for 24 hours, and the National Bank of Georgia. While heavy suspicion was placed on Russia for orchestrating the attack through a proxy, the St. Petersburg-based criminal gang known as the Russian Business Network, or R.B.N, the Russian government denied the allegations, stating that it was possible that individuals in Russia or elsewhere had taken it upon themselves to start the attacks.

■ During the 2009 Iranian election protests, foreign activists seeking to help the opposition engaged in DDoS attacks against Iran's government. The official website of the Iranian government (ahmedinejad.ir (<http://www.ahmadinejad.ir/>)) was rendered inaccessible on several occasions. Critics claimed that the DDoS attacks also cut off internet access for protesters inside Iran; activists countered that, while this may have been true, the attacks still hindered President Mahmoud Ahmadinejad's government enough to aid the opposition.

■ On June 25, 2009, the day Michael Jackson died, the spike in searches related to Michael Jackson was so big that Google News initially mistook it for an automated attack. As a result, for about 25 minutes, when some people searched Google News they saw a "We're sorry" page before finding the articles they were looking for.

- June 2009 the P2P site The Pirate Bay was rendered inaccessible due to a DDoS attack. This was most likely provoked by the recent sellout to Global Gaming Factory X AB, which was seen as a "take the money and run" solution to the website's legal issues.[30] In the end, due to the buyers' financial troubles, the site was not sold.
- Multiple waves of July 2009 cyber-attacks targeted a number of major websites in South Korea and the United States. The attacker used botnet and file update through internet is known to assist its spread. As it turns out, a computer trojan was coded to scan for existing My Doom bots. My Doom was a worm in 2004, and in July around 20,000-50,000 were present. My Doom has a backdoor, which the DDoS bot could exploit. Since then, the DDoS bot removed itself, and completely formatted the hard drives. Most of the bots originated from China, and North Korea.
- On August 6, 2009 several social networking sites, including Twitter, Facebook, Livejournal, and Google blogging pages were hit by DDoS attacks, apparently aimed at Georgian blogger "Cyxymu". Although Google came through with only minor setbacks, these attacks left Twitter crippled for hours and Facebook did eventually restore service although some users still experienced trouble. Twitter's Site latency has continued to improve, however some web requests continue to fail.
- In July and August 2010, the Irish Central Applications Office server was hit by a denial of service attack on four separate occasions, causing difficulties for thousands of Second Level students who are required to use the CAO to apply for University and College places. The attack is currently subject to a Garda investigation.
- On November 28, 2010, whistle blower site wikileaks.org experienced a DDoS attack. This was presumably related to the pending release of many thousands of secret diplomatic cables.
- On December 8, 2010, a group calling themselves "Anonymous" launched orchestrated DDoS attacks on organizations such as Mastercard.com, PayPal, Visa.com and PostFinance; as part of the ongoing "Operation Payback" campaign, which originally targeted anti-piracy organisations, in support of the Whistleblowing site Wikileaks and its founder, Julian Assange. The attack brought down the Mastercard, PostFinance, and Visa websites successfully. PostFinance, the bank that

had frozen Julian Assange's account, was brought down for more than 16 hours due to the attacks. However, in denial of the fact that it was taken down by a bunch of notorious internet users, the bank issued a statement that the outage was caused by an overload of inquiries: "Access to www.postfinance.ch and thus also e-finance is currently overloaded owing to a multitude of online enquiries. The security of customer data is not affected.[13]

2.4. PREVIOUS WORKS AND RESEARCHES CONDUCTED ON DOS SIMILAR METHODS AND AVAILABLE SOLUTIONS :

Based on the study and search I conducted on DOS, I have quoted some ideas belong to some researchers worked in this regard and made similar experiments on DOS. The following includes some quotations with their explanations:

- 1- Denials of service (DoS) attacks have become ongoing to evolve and influence availability of the internet infrastructure. a large number of the researchers in the field of network security and system survivability have been installing mechanisms to find DoS attacks. by conducting such action they hope to maximize precise detections (true-positive) and minimize non-justified detections (false-positive). this research suggests a lightweight approach to recognize DoS attacks by analyzing host behaviors. our approach relies on the term of blind classification or blink: no access to packet payload, no information of port numbers, and no extra information other than what current flow collectors supply. rather than using pre-identified signatures or rudiments as in typical intrusion detection systems, blank maps flows into graph lets of each attack sample. in this work we produce three types of graph lets for the following DoS attack patterns: SYN flood, ICMP flood, and host scan. Upshots suggest that our methodology can introduce all events and all hosts associated accompanied by attack activities, with a low percentage of false positive

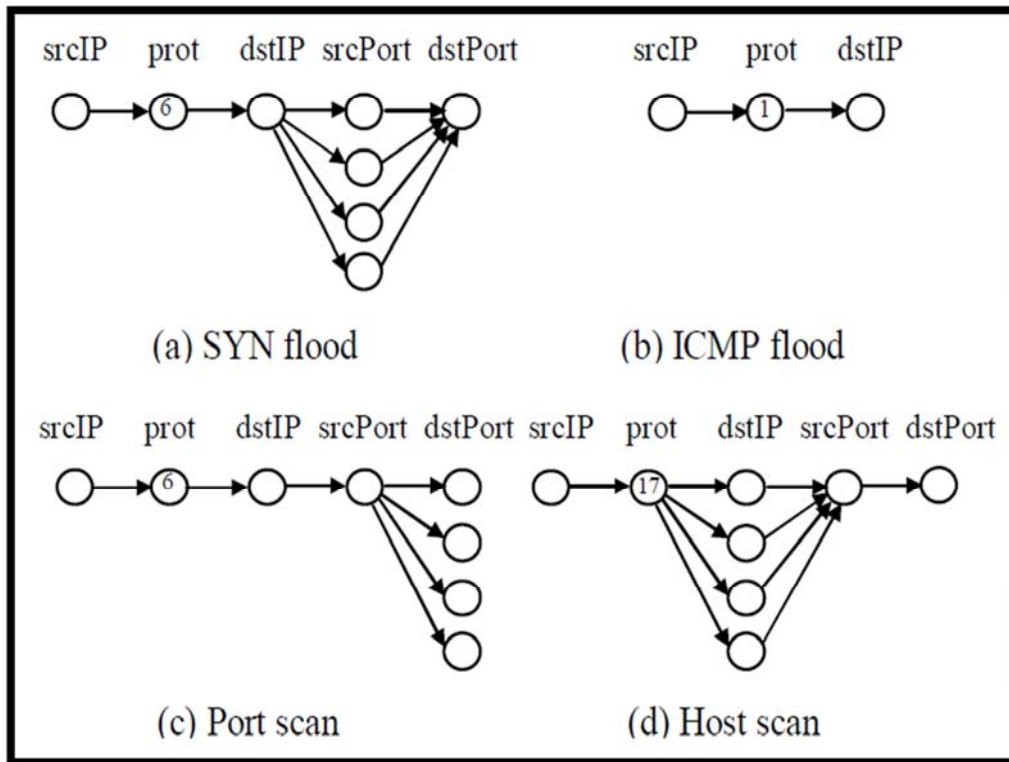


Figure 1 DoS attack graphlets

We propose a lightweight approach to recognize DoS attacks and their onsets. Our approach can identify SYN flood, ICMP flood, port scan, and host scan, according to the concept of BLINC's host behavior analysis. The procedure consists of two phases. First we produce attack graph-lets by checking unique flow behaviors. Secondly, we identify an attack flow by comparing flow records with the previously defined graph-lets. The advantage of our approach is that it is able to identify all events and all hosts linked with attack activities without depending on packet payload, packet inter-arrival time, or size of individual packets. Moreover, it can significantly detect anomalous behaviors in the network if the flow behaviors which are similar to DoS attacks. moreover addition, our approach can perform near real-time detection, within one minute interval, with low possibility of false alarms.[17]

2. Application features such as port numbers are Utilized by Network-based Intrusion Detection Systems (NIDSs) to Find attacks flowing from networks. System calls and the operating System relevant information are used by Host-based Intrusion Detection Systems (HIDSs) to detect intrusions towards a host. However, the connection between hardware Architecture occurrences and Denial-of-Service (DoS) attacks has not been well expressed. When increasingly complicated intrusions emerge, some attacks will be able to bypass both the application and the operating system level feature monitors. hence, more influential solution is needed to support existing HIDSs. In this paper, we identify the following hardware architecture aspects: *Instruction Count, Cache Miss, Bus Traffic* and merge them into an HIDS framework based on a contemporary statistical Gradient supporting model. Through the integration of application, operating system and architecture level aspects, our proposed HIDS shows a significant improvement of the detection rate in terms of sophisticated DoS intrusions

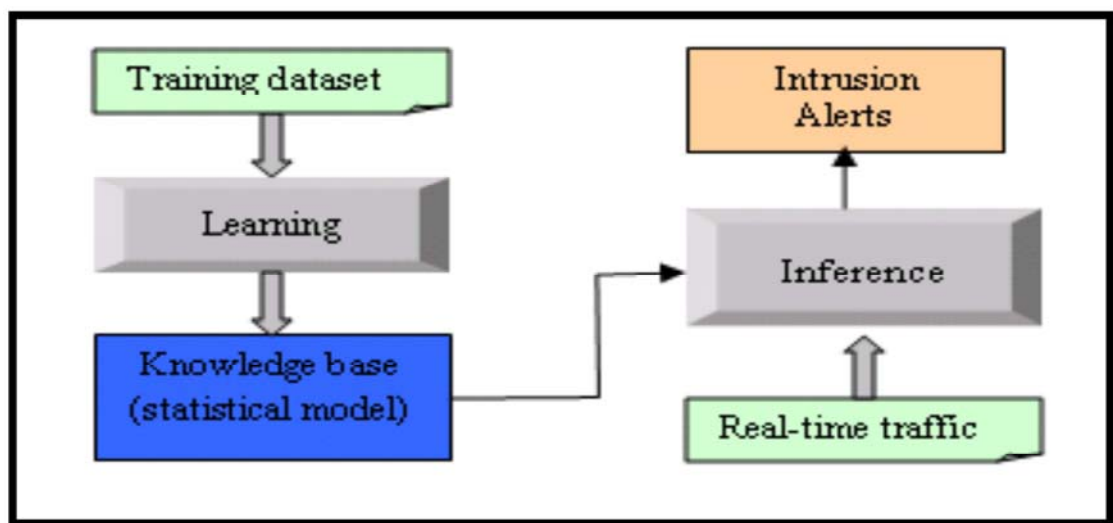


Figure 2 The framework of our Intrusion Detection System

We have made many experiments to show that an only IDS using application features didn't succeed to detect complicated DoS attacks because these attacks demonstrate normally if their behaviors are only watched by the application feature

set. In order to detect the missed DoS attacks, we use a combination of application and architecture feature set. The upshots of our experiments showed improved IDS performance. In brief, we propose the concept that if crackers use complicated systems to evade defense, the architectural level behavior gives us valuable information to develop the IDS against such DoS attacks. [1]

3-These days, users can easily access and download network attack tools, which often provide friendly relationship and easily operated features, from the Internet. Therefore, even a non-skilled hacker can also launch a large scale DoS or DDoS attack to hinder a system, i.e., the victim, from accessing Internet services.

In this part, we suggest an agent based intrusion detection architecture, which is a sprat detection system, to detect DoS/DDoS attacks by a protest of statistic approach that compares source IP addresses' normal and current packet statistics to discriminate whether there is a DoS/DDoS attack. It first gathers all resource IPs' packet stats so as to make their normal packet distribution. When some IPs' current packet distribution abruptly changes, this can be an attack. Experimental upshots show that this method is able to effectively detect DoS/DDoS attacks .

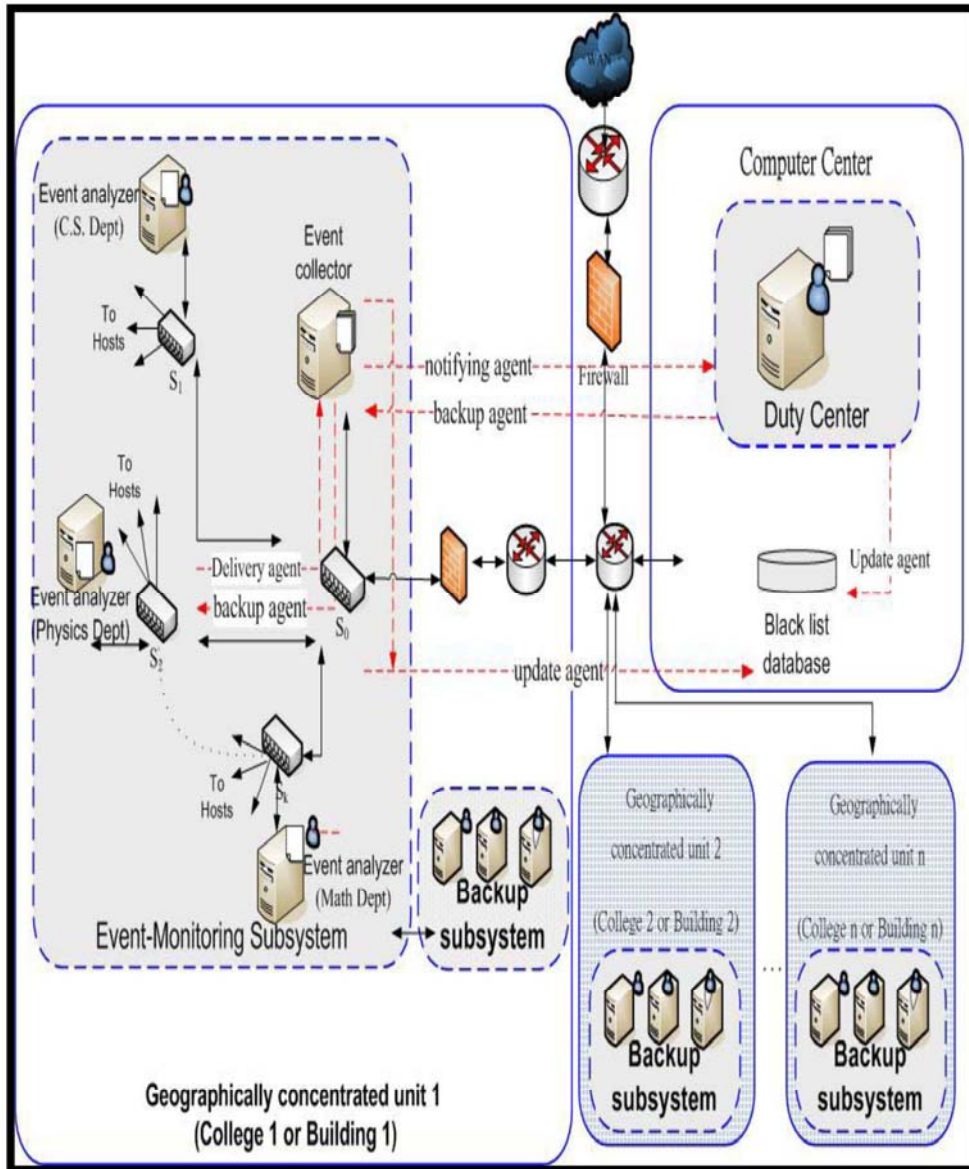


Figure 3 AIDS system architecture

we proposed distributed detection architecture Known as agent based intrusion detection system (AIDS), which uses Quality of fit test of chi-square test to detect DoS/DDoS attacks. It analyzes amount and variation of source address that send packets to us, and statistics of IP address distribution. If hackers utilize attack tools, for example, “Stacheldraht” to generate a big amount of packets of random source IP addresses. We check to see whether its chi-square value exceed threshold. Experimental results show that this method can influentially detect DoS/DDoS attacks.[2]

4 -confirming security of the infrastructure against outer attacks across network borders constitutes one of the main attributes as well as defiances of the next generation mobile network (NGMN). To alleviate the possibility of such attacks emancipating the NGMN architecture, it is required to identify the attack categories. However, detection of the attack types from different traffic flows (as is the case in network links) and their subsequent classification can be a very complicated task, particularly when both the attack and the legitimate traffic demonstrate similar statistical properties (such as denial-of-service (DoS) and distributed DoS (DDoS)). Furthermore, the attacker's capacity to spoof and forge the packet header information (including IP address) makes the detection process even more compound. Conventional anomaly based attack detection mechanisms have been found wanting in such cases.

detection algorithm that recognizes and characterizes network traffic by seeking the frequency spectrum distribution. The Lomb period gram is used to specify the power spectrum of the monitored traffic whereupon two *deviation score* parameters are employed to disconnect the anomaly traffic flows from legitimate ones in a two-step method. For simple purposes, the efficiency of such classification effort is demonstrated for DoS and DDoS attacks only (for their statistical similarity to normal traffic).

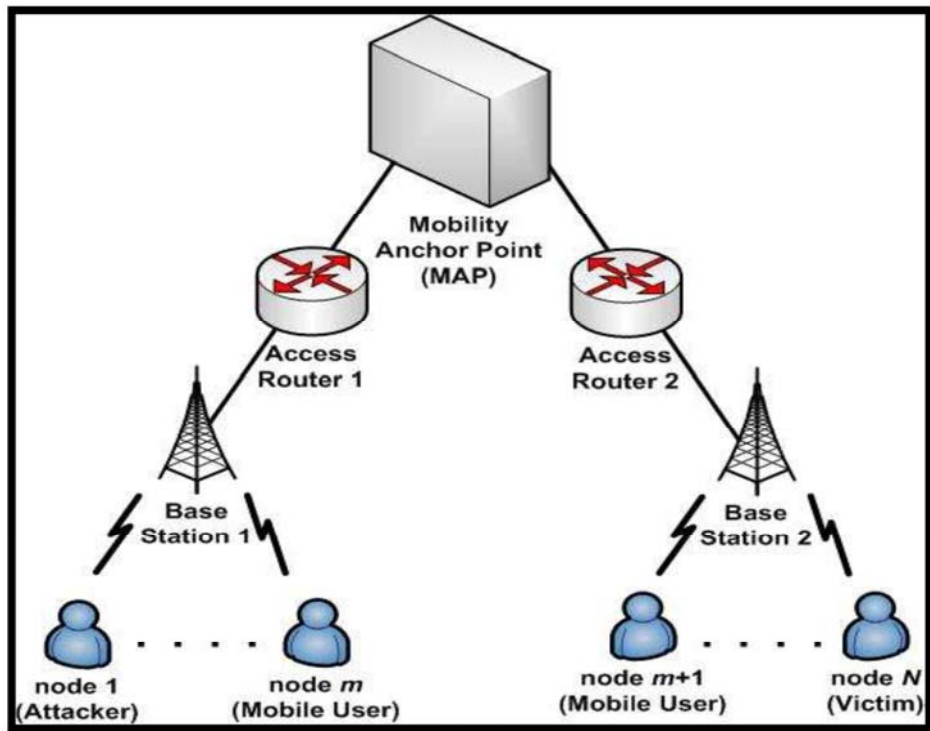


Figure 4 AIDS system architecture Simulation model in ns-2

One of the main goals of NGMN architecture is to avoid spoofed sources from attacking the infrastructure and individual nodes (in the form of DoS and DDoS), as well as to prevent attack migration across heterogeneous network borders. Since conventional anomaly based detection mechanism (using packet-level analysis) are unable of addressing such security cases in expanded networks, in this paper we proposed an NGMN detection mechanism that can precisely classify traffic into normal, DoS and DDoS. We used frequency domain analysis to specify the power spectrum of network traffic. Two deviation score parameters were identified for network anomaly detection and their subsequent classification. The initiative simulation results indicate that the frequency domain analysis is active and has the ability to detect and characterize attacks through different conditions. With such promising results, determining a generic expression that can identify the threshold values, as well as raise the level the segregation of different attack traffic such as worms and flash crowd events will form the future direction of our research.[24]

5- The Session Initiation Protocol (SIP) has been generally used in VoIP for session control and management. As the basic SIP specifications do not need the proxy servers to trace the states of founded sessions, an extension header field “Session-Expires” has been suggested for SIP to permit the proxy server to hold resources for established sessions just within the determined intervals. In this paper, we identify a denial of service (DoS) attack utilizing this SIP extension to exhaust resources of the proxy servers in wireless VoIP. In particular, by planned setting a large value of the “Session-Expires” header and then physically disconnecting from the wireless network, attackers can frequently hold resources of the proxy server as long as they are willing to do so. Also, the low-volume nature of the attack permits it to prevent being detected by existing volume-based intrusion detection systems. As a counter-measure, we propose a concrete detection scheme based on the statistical Anderson-Darling test. The key insight that leads to the scheme is the changed statistical property of the header values caused by the attack. We check the performance through computer simulation. The scheme shows its capacity to detect the attack and is even more effective when used against the distributed denial of service (DDoS) attack.

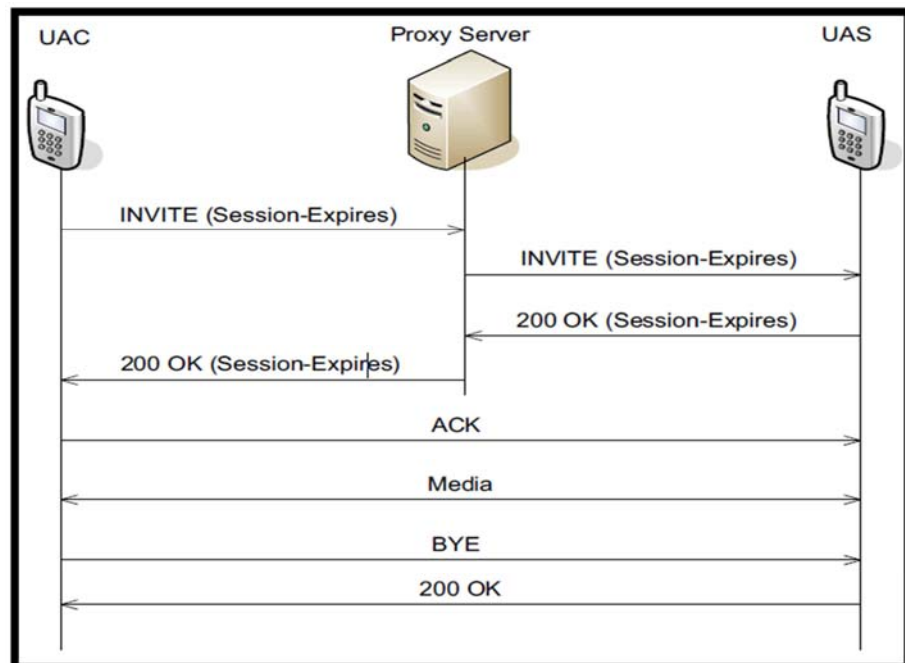


Figure 5 Illustration of SIP signaling

we identify a novel resource-drained denial Of service attack which is directed to SIP-based wireless VoIP networks. The attack works by using vulnerabilities of one SIP protocol extension and wireless networks. The “Session- Expires” header, or the session timer, is basically suggested as a keep-alive mechanism for SIP. However, it provides attackers with chances to reserve resources on the SIP proxy servers as long as they want. Also, wireless networks let attackers to easily disconnect from the network and the disconnection does not release the resources on the proxy servers until the session timer expires. Attackers can carry multiple resources through repeating reservations and disconnections, and bigger damages can be resulted if cooperating attacks are initiated from distributed attackers. Moreover, the low-volume nature of the attack permits it to prevent from being detected by existing volume-based intrusion detection systems. As a counter-measure, we propose a robust

detection scheme for the resource-drained attack based on the statistical Anderson-Darling test through investigating the characteristics of both the normal and attack behaviors. The scheme utilizes the changed statistical property of the session timers induced by the attack as the key insight that leads to detection. Through computer simulation, we show that besides the capability to detect the basic one attacker resource-drained attack, the scheme is even more effective when dealing with the DDoS attack. As future work, we will investigate and quantify the damage level caused by the attack through measures such as call dropping rate and SIP proxy server overload. Also, better statistical measures will be identified to more accurately and realistically model the distribution of the session timer values. [6]

6 - Third Generation (3G) wireless networks based on the CDMA2000 and UMTS standards are now increasingly being deployed throughout the world. Because of their complex signaling and relatively limited bandwidth, these 3G networks are generally more vulnerable than their wire line counterparts, thus making them fertile ground for new attacks. In this paper, we identify and study a novel Denial of Service

(DoS) attack, called *signaling attack*, that exploits the unique vulnerabilities of the signaling/control plane in 3G wireless networks. Using simulations driven by real traces, we are able to demonstrate the impact of a signaling attack. Specifically, we show how a well-timed low-volume signaling attack can potentially overload the control plane and detrimentally affect the key elements in a 3G wireless infrastructure. The low-volume nature of the signaling attack allows it to avoid detection by existing intrusion detection algorithms, which are often signature or volume-based. As a counter-measure, we present and evaluate an online early detection algorithm based on the statistical CUSUM method. Through the use of extensive trace-driven simulations, we demonstrate that the algorithm is robust and can identify an attack in its inception, before significant damage is done.

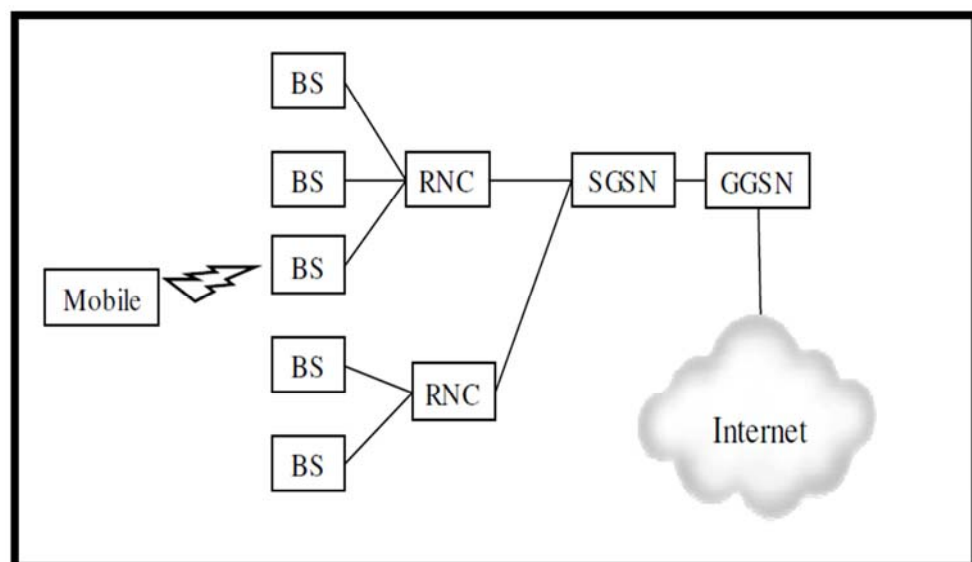


Figure 6 UMTS Network Architecture

We have demonstrated a new DoS attack, known as signaling attack, which targets 3G wireless networks. This attack works by using the heavy-weight nature of signaling in 3G wireless networks. We have demonstrated via trace-driven simulation that the signaling attack can substantially burden a wireless infrastructure using only minimal traffic. Because Due to its low-rate, low-volume property, the signaling attack can evade the detection of traditional counter-DoS systems. In view

of in this regard, we have proposed a statistical CUSUM-based detection mechanism to defend against the signaling attack. Using real-world tracks, we have shown that our detection mechanism can recognize the source of a signaling attack in a timely manner before the damage becomes aggravated, while producing very few false positives. In addition, our detection mechanism is strong as it relies solely on the extra signaling load and based on any assumed attack strategy. [18]

7 - DoS (Denial of Service) attacks, an access filtering mechanism is used in the firewall. The difficulty to define the filtering rules lies where ordinary and anomaly packets have to be different in the incoming packets. The objective behind our research is to explore the early detective approach for anomaly accesses based on statistical analysis. In this paper, we defined the chi-square method, and then conducted analyses the all amount of incoming packets to our College. As the results, we debriefed the following aspects. Firstly, the chi-square analysis based on the destination port number is more sensitive to the DDoS attacks and IP scan than that based on the destination IP address. Secondly, DoS attacks lift up the chi-square value up based on the analysis of the destination IP address. Finally, the multiplexing DoS attacks tend to decrease the chi-square values in both analyses.

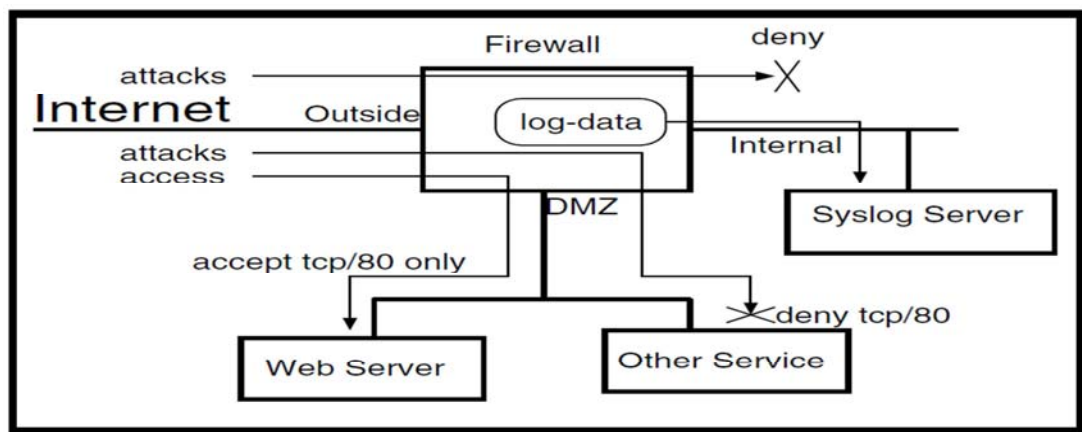


Figure 7 Our experimental network environment for data collection

We watched all amount of incoming packets, and conducted analyses based according to both the source IP address and the destination port number. As a result, we extracted the following aspects. Firstly, the chi-square analysis based on the destination port number is more sensitive to the DDoS attacks and IP scan than that based on the destination IP address. Secondly, DoS attacks raise the chi-square value up based on the analysis of the destination IP address. Finally, The multiplexing DoS attacks tend to reduce the chi-square values in both analyses. In the future, analysis based on the number of bytes of received packets and the duration of packet receiving are to be planned. Combined analysis will create the new features Of different of attacks. [11]

2.5. DOS ATTACKS: DETECTION AND PREVENTION

In the literature, there are several approaches to deal with denial of service (DoS) attacks. In this section, we provide an approximate taxonomy of these approaches. In Addition, we briefly describe the main features of each approach and highlight the strengths and weaknesses of it. We divide the approaches for dealing with DoS attacks into two main categories: *detection* and *prevention* approaches. The detection approaches capitalize on the fact that appropriately punishing wrong doers (attackers) will deter them from re-attacking again, and will scare others to do similar acts. The detection process has two phases:

Detecting the attack and identifying the attacker. To identify an attacker, several *trace back* methods can be used, as explained later in this section. The obvious way to detect an attack is just waiting till the system performance decreases sharply or even the whole system collapses. We propose a more effective method for detecting attacks before they severely harm the system. We propose to use monitoring for *early* detection of DoS attacks. The details are given in Section 3. The prevention approaches, on the other hand, try to thwart attacks before they harm the system. Filtering is the main strategy used in the prevention approaches. To clarify the presentation, we use the hypothetical network topology shown in Figure 8 to demonstrate several scenarios for DoS attacks and how the different

approaches react to them. The figure shows several hosts (denoted by H_s) connected to four domains D_1 , D_2 , D_3 , and D_4 ; which are interconnected through the Internet cloud. In the figure, A_i represents an attacker i while V represents a victim.[7]

2.5.1. DoS Attacks

The aim of a DoS attack is to consume the resources of a victim or the resources on the way to communicate with a victim. By wasting the victim's resources, the attacker disallows it from serving legitimate customers. A victim can be a host, server, router, or any computing entity connected to the network. Inevitable human errors during software development, configuration, and installation open several unseen doors for these type of attacks. Several DoS attacks are known and documented in the literature,

Flooding a victim with an overwhelming amount of traffic is the most common. This unusual traffic clogs the communication links and thwarts all connections among the legitimate users, which may result in shutting down an entire site or a branch of the network. This happened in February of 2000 for the popular web sites Yahoo, E*trade, EBay, and CNN for several hours.

TCP SYN flooding is an instance of the flooding attacks, Under this attack, the victim is a host and usually runs a Web server. A regular client opens a connection with the server by sending a TCP SYN segment. The server allocates buffer for the expected connection and replies with a TCP ACK segment. The connection remains half-open (backlogged) till the client acknowledges the ACK of the server and moves the connection to the established state. If the client does not send the ACK, the buffer will be deallocated after an expiration of a timer. The server can only have a specific number of half-open connections after which all requests will be refused. The attacker sends a TCP SYN segment pretending a desire to establish a connection and making the server reserves buffer for it. The attacker does not complete the connection. Instead, it issues more TCP SYNs, which lead

the server to waste its memory and reach its limit for the backlogged connections. Sending such SYN requests with a high rate keeps the server unable to satisfy connection requests from legitimate users. Scuba *et al.* developed a tool to alleviate the SYN flooding attack. The tool watches for SYN segments coming from spoofed IP addresses and sends TCP RST segments to the server. The RST segments terminate the half-open connections and free their associated buffers. Other types of flooding attacks include TCP ACK and RST flooding, ICMP and UDP echo-request flooding, and DNS request flooding.[7]

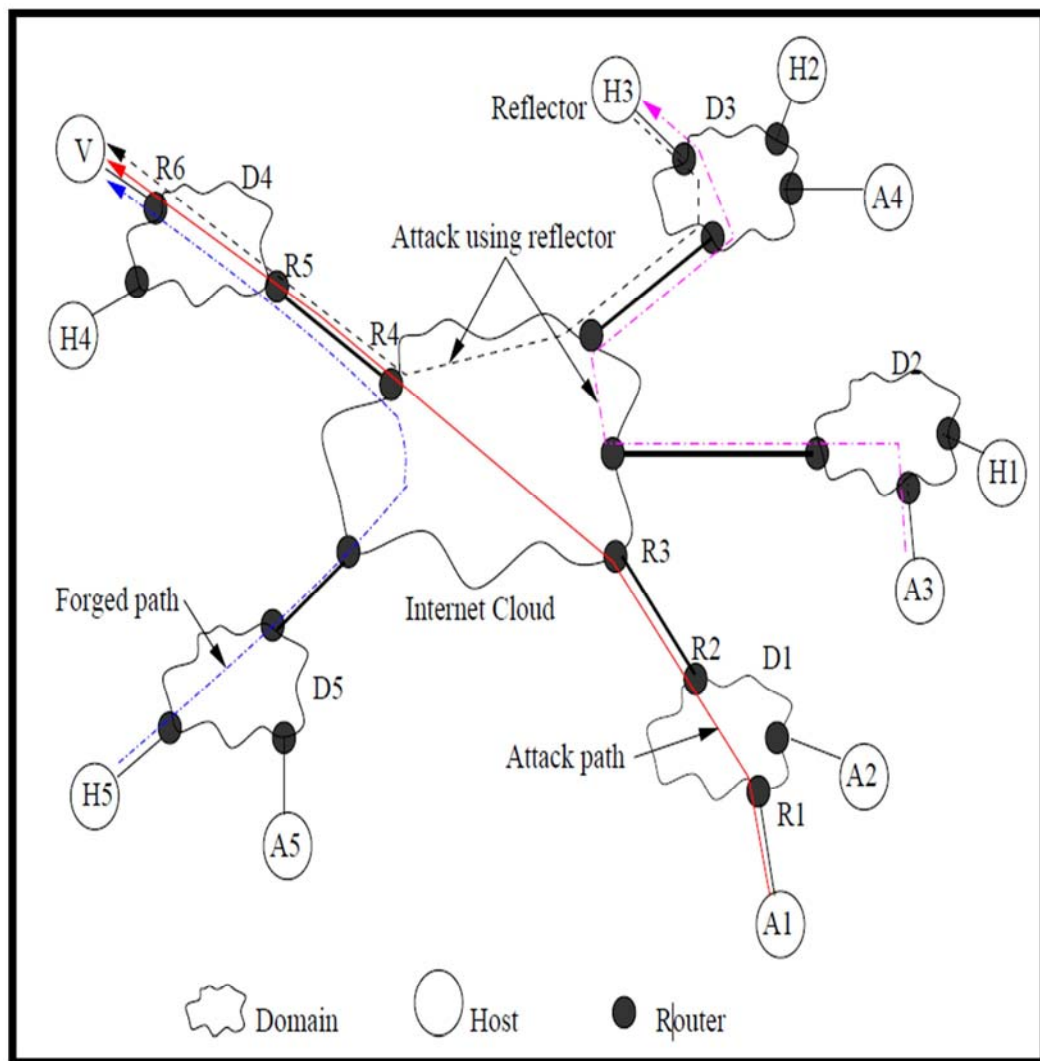


Figure 8 Different scenarios for DoS attacks. Attacker A1 launches an attack on the victim V . A1 spoofs IP address of host H5 from domain D5. Another attackerA3 uses host H3 as a reflector to attack V. [7]

This list is by no means exhaustive. A DoS attack can be more severe when an attacker uses multiple hosts over the Internet to storm a victim. To achieve this, the attacker compromises many hosts and deploys attacking agents on them. The attacker signals all agents to simultaneously launch an attack on a victim. Barros shows that DDoS attack can reach a high level of sophistication by using *reflectors*. A reflector is like a mirror that reflects light. In the Internet, many hosts such as Web servers, DNS servers, and routers can be used as reflectors because they always reply to (or reflect) specific type of packets. Web servers reply to SYN requests, DNS servers reply to queries, and routers send ICMP packets (time exceeded or host unreachable) in response to particular IP packets. The attackers can abuse these reflectors to launch DDoS attacks. For example, an attacking agent sends a SYN request to a reflector specifying the victim's IP address as the source address of the agent. The reflector will send a SYN ACK to the victim. There are millions of reflectors in the Internet and the attacker can use these reflectors to flood the victim's network by sending a large amount of packets. Paxson analyzes several Internet protocols and applications and concludes that DNS servers, Gnutella servers, and TCP-based servers are potential reflectors.[7]

2.5.2. Detection Approaches

The detection approaches rely on finding the malicious party who launched a DoS attack and consequently hold him liable for the damage he has caused. However, pinning the real attacker down is not a straightforward task. One reason is that the attacker spoofs the source IP address of the attacking packets. Another reason is that the Internet is *stateless*, which means, whenever a packet passes through a router, the router does not store any information (or traces) about that packet. Therefore, mechanisms such as ICMP trace back and packet marking is devised to figure out the real

attacker. In this subsection, we describe several techniques to identify the attacker after the attack took place. We defer the issue of early detection of an attack till Section 3. [7]

5.2.1. ICMP Traceback :

Bellovin proposes the idea of ICMP traceback messages, where every router samples the forwarded packets with a very low probability (e.g., 1 out of 20,000) and sends an ICMP Traceback message to the destination. An ICMP Traceback message contains the previous and next hop addresses of the router, timestamp, portion of the traced packet, and authentication information. In Figure 8, while packets are traversing the network path from the attacker A1 to the victim V; the intermediate routers (R1;R2;R3;R4;R5; and R6) sample some of these packets and send ICMP Traceback messages to the destination V:With enough messages, the victim can trace the network path A1 ! V: The pitfall of this approach is that the attacker can send many false ICMP Traceback messages to confuse the victim. To address Distributed DoS (DDoS) attacks by reflectors, Barros proposes a modification to the ICMP Traceback messages. In his refinement, routers sometimes send ICMP Traceback messages to the *source*. In Figure 8, A3 launches a DDoS attack by sending TCP SYN segments to the reflector H3 specifying V as the source address. H3, in turn, sends SYN ACK segments to the victim V: According to the modification, routers on the path A3 ! H3 will send ICMP messages to the source, i.e., to V: This *reverse trace* enables the victim to identify the attacking agent from these trace packets. The *reverse trace* mechanism depends only on the number of attacking agents, and not on the number of reflectors. This achieves scalability because the number of available reflectors is much higher than the number of attacking agents on the Internet. Snoeren *et al.* propose an attractive hashed-based system that can trace the origin of a single IP packet delivered by a network in the recent past. The system is called source path isolation engine (SPIE). The SPIE uses an efficient method to store information about packets

traversing through a particular router. The method uses n bits of the hashed value of the packet to set an index of a 2^n -bit *digest table*. When a victim detects an attack, a query is sent to SPIE, which queries routers for packet digests of the relevant time periods. Topology information is then used to construct the attack graph from which the source of the attack is determined. [7]

5.2.2. Packet Marking:

Instead of having routers send separate messages for the sampled packets, Burch and Cheswick propose to inscribe some path information into the header of the packets themselves. This marking can be deterministic or probabilistic. In the deterministic marking, every router marks all packets. The obvious drawback of the deterministic packet marking is that the packet header grows as the number of hops increases on the path. Moreover, significant overhead will be imposed on routers to mark every packet.

The probabilistic packet marking (PPM) encodes the path information into a small fraction of the packets. The assumption is that during a flooding attack, a huge amount of traffic travels towards the victim. Therefore, there is a great chance that many of these packets will be marked at routers throughout their journey from the source to the victim. It is likely that the marked packets will give enough information to trace the network path from the victim to the source of the attack. Savage *et al.* describe efficient mechanisms to encode the path information into packets. This information contains the XOR (exclusive OR) of two IP addresses and a distance metric. The two IP addresses are for the start and the end routers of the link. The distance metric represents the number of hops between the attacker and the victim. To illustrate the idea, consider the attacker A1 and the victim V in Figure 8. Assume there is only one hop between routers R3 and R4: If Router R1 marks a packet, it will encode the XOR of R1 and R2 addresses into the packet and sets the distance metric to zero, that is, it will encode the tuple $\langle R1 _ R2; 0 \rangle$. Other routers on the path just increase the distance

metric of this packet, if they don't decide to mark it again. When this packet reaches the victim, it provides the tuple $\langle R1_R2; 5 \rangle$. Similarly, some packets may get marked at routers $R2;R3;R4;R5$; and $R6$ and they will provide the tuples $\langle R2_R3; 4 \rangle; \langle R3_R4; 3 \rangle; \langle R4_R5; 2 \rangle; \langle R5_R6; 1 \rangle; \langle R6; 0 \rangle$; respectively, when they reach the victim. The victim can retrieve all routers on the path by XORing the collected messages sorted by distance. (Recall that $Rx_Ry_Rx = Ry$.) This approach can reconstruct most network paths with 95% certainty if there are about 2,000 marked packets available and even the longest path can be resolved with 4,000 packets. For DoS attacks, this amount of packets is clearly obtainable because the attacker needs to flood the network to cause a DoS attack. (Moore *et al.* report that some severe DoS attacks had a rate of thousands of packets per second.) The authors describe ways to reduce the required space and suggest to use the identification field (currently used for IP fragmentation) of IP header to store the encoding of the path information. They also propose solutions to handle the co-existence of marking and fragmentation of IP packets. The main limitation of the PPM approaches stems from the fact that, nothing prevents the attacker from marking packets. If a packet marked by the attacker does not get re-marked by any intermediate router, it will confuse the victim and make it harder to trace the real attacker. Park and Lee show that for single-source DoS attacks, PPM can identify a small set of sources as potential candidates for a DoS attack. For DDoS attacks, however, the attacker can increase the uncertainty in localizing the attacker. Therefore, PPM is vulnerable to distributed DoS attacks [7].

2.5.3. Prevention Approaches:

Preventive approaches try to stop a DoS attack by identifying the attack packets and discarding them before reaching the victim. We summarize several packet filtering techniques that achieve this goal. [7]

5.3.1. Ingress Filtering:

Incoming packets to a network domain can be filtered by ingress routers. These filters verify the identity of packets entering into the domain, like an immigration security system at the airport. Ingress filtering, proposed by Ferguson and Senile, is a restrictive mechanism that drops traffic with IP address that does not match a domain prefix connected to the ingress router. As an example, in Figure 8, the attacker A1 resides in domain D1 with the network prefix a.b.c.0/24. The attacker wants to launch a DoS attack to the victim V that is connected to domain D4. If the attacker spoofs the IP address of host H5 in domain D5, which has the network prefix x.y.z.0/24, an input traffic filter on the ingress link of R1 will thwart this spoofing. R1 only allows traffic originating from source addresses within the a.b.c.0/24 prefix. Thus, the filter prohibits an attacker from using *spoofed* source addresses from outside of the prefix range. Similarly, filtering foils DDoS attacks that employ reflectors. In Figure 8, ingress filter of D2 will discard packets destined to the reflector H3 and specifying V's address in the source address field. Thus, these packets will not be able to reach the reflector. Ingress filtering can drastically reduce the DoS attack by IP spoofing if *all* domains use it. It is hard, though, to deploy ingress filters in *all* Internet domains. If there are some unchecked points, it is possible to launch DoS attacks from that points. Unlike ingress filters, egress filters reside at the exit points of a network domain and checks whether the source address of exiting packets belong to this domain. Aside from the placement issue, both ingress and egress filters have similar behavior. [7]

5.3.2. Route-Based Filtering:

Park and Lee propose route-based distributed packet filtering, which rely on route information to filter out spoofed IP packets. For instance, suppose that A1 belongs to domain D1 and is attempting a DoS attack on V that belongs to domain D4. If A1 uses the spoofed address H5 that belongs to domain D5, the filter at domain D1 would recognize that a packet originated from domain D5 and destined to V should not travel through domain D1. Then, the filter at D1 will discard the packet. Routebased filters do not use/store individual host addresses for filtering, rather, they use the topology information of Autonomous Systems (ASes). The authors of show that with partial deployment of route-based filters, about 20% in the Internet AS topologies, it is possible to achieve a good filtering effect that prevents spoofed IP flows reaching other ASes. These filters need to build route information by consulting BGP routers of different ASes. Since routes on the Internet change with time, it is a challenge for route-based filters to be updated in real time. Finally, all filters proposed in the literature so far fall short to detect IP address spoofing from the domain in which the attacker resides.

For example, in Figure8, if A1 uses some unused IP addresses of domain D1; the filters will not be able to stop such forged packets to reach the victim V. [7]

2.6.IP ADDRESSING SCHEME

. OVERVIEW :

The number of machines in your network and need to support, will affect several decisions you will need to make. Some organizations require only a small network of serveral dozen standalone machines located on one floor of a single building. In other organizations, you may need to set up a network with more than 1000 hosts spanning several buildings. In cases where you will need to support a large number of hosts, it may require you to further divide your network into subdivisions called subnets.[3] The size of your prospective network will affect the:

- Network class you apply for
- Network number you receive
- IP addressing scheme you use for your network

2.7.PARTS OF AN IP ADDRESS :

Any TCP/IP network will require a unique network number and every host on a TCP/IP network will require a unique IP address. Before registering your networking and obtaining a network number, it is critical that you understand how IP address are constructed.

An IP address is a 32-bit number that uniquely identifies a network interface on a machine. IP addresses are typically written in decimal digits, formatted as four 8-bit fields separated by periods. Each 8-bit field represents a byte of the IP address. This form of representing the bytes of an IP address is often referred to as the dotted-decimal format.

The bytes of an IP address can be further classified into two parts: the network part and the host part. [3] The example below shows the components of the Class B network 192.168.1.100.

192.168.1.100

| |___ (host part)

|

|___ (network part)

2.7.1. Network Part

This part specifies the unique number assigned to your particular network. It is also the part that identifies the class of network assigned. In the above example, the network part takes up two bytes of the IP address, namely 192.168. [3]

2.7.2. Host Part

This is the part of the IP address that you assign to each host, and uniquely identifies each host on your network. Note that for each host on your network, the network part of the address will be the same, but the host part must be different. [3]

2.7.3. Subnet Number Optional

Many local area networks (LANs) with a large number of hosts will be divided into *subnets*. If you choose to divide your network into subnets, you need to assign a

subnet number for the subnet. You can maximize the efficiency of IP address space by using some of the bits from the host number part of the IP address as a network identifier. When used as a network identifier, the specified part of the address becomes the subnet number. You create a subnet number using a subnet mask, which is a bit mask that selects the network and subnet parts of an IP address. [3]

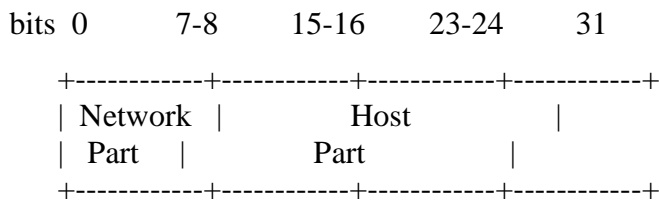
2.8. NETWORK CLASSES:

The first step in planning for IP addressing on your network is to determine which network class is appropriate for your network. After you have done this, you can take the critical second step - obtaining the network number from the inter NIC addressing authority.

Currently, there are three classes of TCP/IP networks. Each class uses the 32-bit IP address space differently, providing more or fewer bits for the network part of the address. These classes are Class A, B and Class C. [3]

2.8.1. Class A Network Numbers:

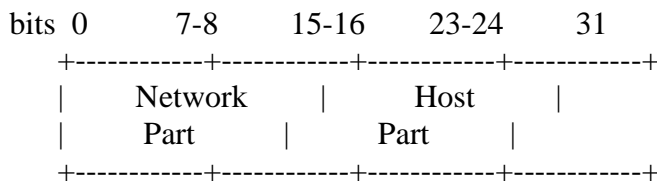
A Class A network number uses the first 8 bits of the IP address as its "network part". The remaining 24 bits comprise the host part of the IP address. (See below)



The values assigned to the first byte of Class A network numbers fall within the range 0-127. Consider for example the IP address 68.8.1.100. The value 68 in the first byte indicates that the host is on a Class A network. The Inter NIC assigns only the first byte of a Class A number. Use of the remaining three bytes is left to the discretion of the owner of network number. Only 127 Class A networks can exist. Each one of these numbers can accommodate up to 16,777,214 hosts. [3]

2.8.2. Class B Network Numbers :

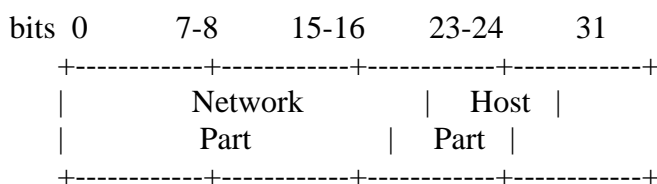
A Class B network number consists of the first 16 bits for the network number and 16 bits for host numbers. The first byte of a Class B network number is in the range 128-191. Take for example the IP address 132.168.1.100, the first two bytes, 132.168, are assigned by the Inter NIC, and comprise the network address. The last two bytes, 1.100, make up the host part of the address, and is assigned at the discretion of the owner of the network number. (See below)



Class B is typically assigned to organizations with many hosts on their network.[3]

2.8.3. Class C Network Numbers:

A Class C network uses 24 bits for the network part and 8 bits for the host part. Class C network numbers are appropriate for networks with few hosts - the maximum being 254. A Class C network number occupies the first three bytes of an IP address. Only the fourth byte is assigned at the discretion of the network number owner. (See below)



The first bytes of a class C network number covers the range 192-223. The second and third each cover the range 1-255. A typical Class C address might be 192.98.1.100. The first three bytes, 192.98.1, form the network number. The final byte in this example, 100, is the host number. [3]

2.9. ADMINISTERING NETWORK NUMBERS :

If your organization has been assigned more than one network number, or uses subnets, appoint a centralized authority within your organization to assign network numbers. That authority should maintain control of a pool of assigned network numbers, assigning network, subnet, and host numbers as required. To prevent problems, make sure that duplicates or random network numbers do not exist in your organization.[3]

2.10. DESIGNING YOUR IP ADDRESSING SCHEME :

After receiving your network number, you can then start planning how you will assign the host parts of the IP address. The table below (*Division of IP Address Space*) shows the division of the IP address space into network and host address spaces. For each Class, "range" specifies the range of decimal values for the first byte of the network number. "Network Address" indicates the number of bytes of the IP address that are dedicated to the network part of the address, with each byte represented by xxx. "Host Address" indicates the number of bytes dedicated to the host part of the address. For example, in a Class A network address, the first byte is dedicated to the network, and the last three are dedicated to the host. The opposite is true for a Class C network. [3]

Division of IP Address Space

Class	Range	Network Address	Host Address
A	0-127	xxx	xxx.xxx.xxx
B	128-191	xxx.xxx	xxx.xxx
C	192-223	xxx.xxx.xxx	xxx

The numbers in the first byte of the IP address define whether the network is Class A, B, or C and are always assigned by the Inter NIC. The remaining three bytes have a range from 0-255. The numbers 0 and 255 are reserved; you can assign the numbers 1-254 to each byte depending on the network number assigned to you. The following table shows which bytes of the IP address are assigned to you and the range of numbers within each byte that are available for you to assign to your hosts.

Range of Available Numbers

Class	Byte 1 Range	Byte 2 Range	Byte 3 Range	Byte 4 Range
A	0-127	1-254	1-254	1-254
B	128-191	Pre-assigned by Internet	1-254	1-254
C	192-223	Pre-assigned by Internet	Pre-assigned by Internet	1-254

2.11. How IP Addresses Apply to Network Interfaces :

Before connecting a host to the network, a computer must have at least one network interface. Each network interface must have its own IP address. The IP address that you give to a host is assigned to its network interface, sometimes referred to as the *primary network interface*. If you add a second network interface to the machine, it must have its own unique IP address. Adding a second network interface changes the function of a machine from a host to a router. If you add a second network interface to a host and disable routing, the host is then considered a multihomed host. Each network interface has a device name, device driver, and associated device file in the `/devices` directory. The network interface might have a device name such as `le0` or `smc0`, device names for two commonly used Ethernet interfaces [3]

Internet Protocol (IP)

Internet Protocol (IP) essentially is the Internet layer. The other protocols found here merely exist to support it. IP holds the big picture and could be said to “see all,” in that it’s aware of all the interconnected networks. It can do this because all the machines on the network have a software, or logical, address called an IP address, which I’ll cover more thoroughly later in this chapter. IP looks at each packet’s address. Then, using a routing table, it decides where a packet is to be sent next, choosing the best path. The protocols of the Network Access layer at the bottom of the DoD model don’t possess IP’s enlightened scope of the entire network; they deal only with physical links (local networks). Identifying devices on networks requires answering these two questions: Which network is it on? And what is its ID on that network? The first answer is the *software address*, or *logical address* (the correct street). The second answer is the hardware address (the correct mailbox). All hosts on a network have a logical ID called an IP address. This is the software, or logical, address and contains valuable encoded information, greatly simplifying the complex

task of routing. (IP is discussed in RFC 791.) IP receives segments from the Host-to-Host layer and fragments them into datagrams (packets) if necessary. IP then reassembles datagrams back into segments on the receiving side. Each datagram is assigned the IP address of the sender and of the recipient. Each router (layer 3 device) that receives a datagram makes routing decisions based on the packet's destination IP address. Figure 2.6 shows an IP header. This will give you an idea of what the IP protocol has to go through every time user data is sent from the upper layers and is to be sent to a remote network.[3]

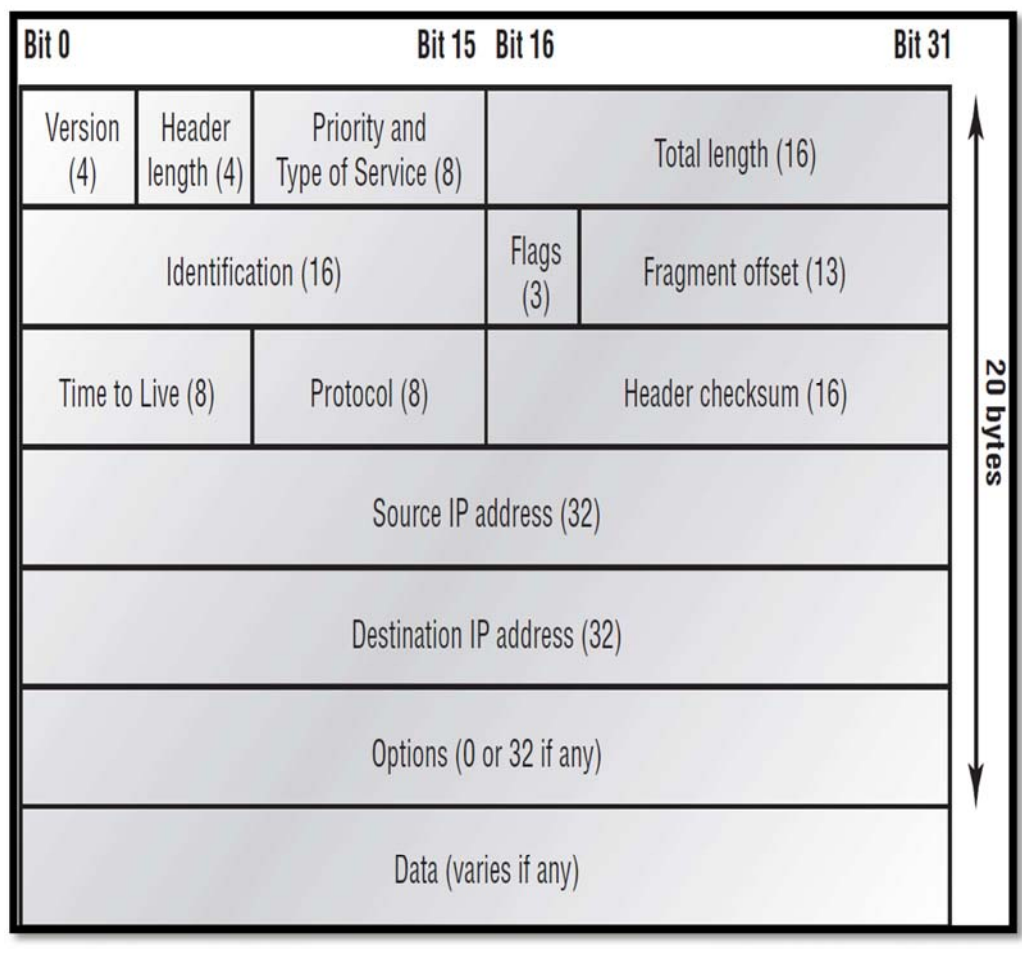


Figure 9 IP header [3]

2.12. IPV6 :

One of the newest major standards on the horizon is IPv6. Although IPv6 has not officially become a standard, it is worth some overview. It is very possible that this information will change as we move closer to IPv6 as a standard, so you should use this as a guide into IPv6, not the definitive information.

A number of books are now being published that cover in detail this emerging standard; if you are looking for more details you should refer to these books. All the RFCs available on the Internet have the raw details on how this standard is developing. However, these documents are difficult to interpret at first glance and require some commitment to going through any number of RFCs pertaining to many subjects all related to IPv6 development.

Internet Protocol Version 4 is the most popular protocol in use today (see Chapter 31, “Internet Protocols”), although there are some questions about its capability to serve the Internet community much longer. IPv4 was finished in the 1970s and has started to show its age. The main issue surrounding IPv6 is addressing—or, the lack of addressing—because many experts believe that we are nearly out of the four billion addresses available in IPv4. Although this seems like a very large number of addresses, multiple large blocks are given to government agencies and large organizations. IPv6 could be the solution to many problems, but it is still not fully developed and is not a standard—yet! Many of the finest developers and engineering minds have been working on IPv6 since the early 1990s.

Hundreds of RFCs have been written and have detailed some major areas, including expanded addressing, simplified header format, flow labeling, authentication, and privacy. Expanded addressing moves us from 32-bit address to a 128-bit addressing method. It also provides newer unicast and broadcasting methods, injects hexadecimal into the IP address, and moves from using “.” to using “:” as delimiters.[3]

Table 3 IPv6 Packet Header Format

4 bits	4 bits	24 bits
Version	version	Flow label

Some of the benefits of IPv6 seem obvious: greater addressing space, built-in QoS, and better routing performance and services. However, a number of barriers must be overcome before the implementation of IPv6. The biggest question for most of us will be what the business need is for moving from current IPv4 to IPv6. The killer app has not appeared yet, but it may be closer than we think. The second consideration is the cost—it may not have much to do with hardware replacement cost. All the larger routers have upgradable OSs IOS; the only necessity is the commitment to upgrading IOS. More likely to do with training and support of minor IP devices such as printers and network faxes, they will support the new address space. IPv6 has schemes to support old and new, however, so this may not even be a barrier. The last issue to consider is training: This will need to happen sooner or later because we all need to start thinking about 128-bit addressing based on MAC addresses in HEX. This involves all new ways of addressing and will be an uncomfortable change for many people.

This conclusion may seem negative, but the greater good will overpower all the up-front issues. The issue is not whether you will have to move to IPv6, but when! We all need IPv6; the increased address space is needed for the growth of IP appliances that we are starting to hear about weekly. IP-ready cars are already shipping today. This requires mobility, which is addressed in IPv6.

Of course, a number of very important features have not been discussed in this section, including QoS, mobile IP, autoconfiguration, and security. All these areas are extremely important, and until IPv6 is finished, you should keep referring to the IETF Web site for the most current information. Several new books on IPv6 also are

starting to show up on bookstore shelves and should provide the deeper technical detail on address headers and full packet details.[3]

2.14.Private IP Addresses :

The people who created the IP addressing scheme also created what we call private IP addresses .These addresses can be used on a private network, but they're not routable through the Internet. This is designed for the purpose of creating a measure of well-needed security, but it also conveniently saves valuable IP address space. If every host on every network had to have real routable IP addresses, we would have run out of IP addresses to hand out years ago. But by using private IP addresses, ISPs, corporations, and home users only need a relatively tiny group of bona fide IP addresses to connect their networks to the Internet. This is economical because they can use private IP addresses on their inside networks and get along just fine. To accomplish this task, the ISP and the corporation—the end user, no matter who they are—need to use something called Network Address Translation (NAT), which basically takes a private IP address and converts it for use on the Internet. (NAT is covered in Chapter 10, “Network Address Translation.”) Many people can use the same real IP address to transmit out onto the Internet. Doing things this way saves megatons of address space—good for us all!

Broadcast Addresses Most people use the term *broadcast* as a generic term, and most of the time, we understand what they mean. But not always. For example, you might say, “The host broadcasted through a router to a DHCP server,” but, well, it's pretty unlikely that this would ever really happen. What you probably mean—using the correct technical jargon—is, “The host broadcasted for an IP address; a router then forwarded this as a unicast packet to the DHCP server.” Oh, and remember that with IPv4, broadcasts are pretty important, but with IPv6, there aren't any broadcasts sent at all—now there's something to get you excited about when you get to Chapter 13! Okay, I've referred to broadcast addresses throughout Chapters 1 and 2, and even showed you some examples. But I really haven't gone into the different terms and uses associated with them yet, and it's about time I did. So here are the four different broadcast (generic term *broadcast*) types that I'd like to define for you:

Layer 2 broadcasts These are sent to all nodes on a LAN. Broadcasts (layer 3) These are sent to all nodes on the network. Unicast These are sent to a single destination host. Multicast These are packets sent from a single source and transmitted to many devices on different networks. First, understand that layer 2 broadcasts are also known as hardware broadcasts—they only go out on a LAN, and they don't go past the LAN boundary (router). The typical hardware address is 6 bytes (48 bits) and looks something like 0c.43.a4.f3.12.c2. The broadcast would be all 1s in binary, which would be all *F*s in hexadecimal, as in FF.FF.FF.FF.FF.FF. Then there's the plain old broadcast addresses at layer 3. Broadcast messages are meant to reach all hosts on a broadcast domain. These are the network broadcasts that have all host bits on. Here's an example that you're already familiar with: The network address of 172.16.0.0 255.255.0.0 would have a broadcast address of 172.16.255.255—all host bits on. Broadcasts can also be “all networks and all hosts,” as indicated by 255.255.255.255. A good example of a broadcast message is an Address Resolution Protocol (ARP) request. When a host has a packet, it knows the logical address (IP) of the destination. To get the packet to the destination, the host needs to forward the packet to a default gateway if the destination resides on a different IP network. If the destination is on the local network, the source will forward the packet directly to the destination. Because the source doesn't have the MAC address to which it needs to forward the frame, it sends out a broadcast, something that every device in the local broadcast domain will listen to. This broadcast says, in essence, “If you are the owner of IP address 192.168.2.3, please forward your MAC address to me,” with the source giving the appropriate information.[3]

A unicast is different because it's a broadcast packet that goes from 255.255.255.255 to an actual destination IP address—in other words, it's directed to a specific host. A DHCP client request is a good example of how a unicast works. Here's an example: Your host on a LAN sends out an FF.FF.FF.FF.FF.FF layer 2 broadcast and 255.255.255.255 layer 3 destination broadcast looking for a DHCP server on the LAN. The router will see that this is a broadcast meant for the DHCP server because it has a destination port number of 67 (BootP server) and will forward the request to the IP address of the DHCP server on another

LAN. So, basically, if your DHCP server IP address is 172.16.10.1, your host just sends out a 255.255.255.255 DHCP client broadcast request, and the router changes that broadcast to the specific destination address of 172.16.10.1. (In order for the router to provide this service, you need to configure the interfaces with the ip helper-address command—this is not a default service). Multicast is a different beast entirely. At first glance, it appears to be a hybrid of unicast and broadcast communication, but that isn't quite the case. Multicast does allow point-to multipoint communication,

which is similar to broadcasts, but it happens in a different manner. The crux of multicast is that it enables multiple recipients to receive messages without flooding the messages to all hosts on a broadcast domain. Multicast works by sending messages or data to IP multicast group addresses. Routers then forward copies (unlike broadcasts, which are not forwarded) of the packet out every interface that has hosts subscribed to that group address. This is where multicast differs from broadcast messages—with multicast communication, copies of packets, in theory, are sent only to subscribed hosts. When I say in theory, this means that the hosts will receive, for example, a multicast packet destined for 224.0.0.9 (this is an EIGRP packet and only a router running the EIGRP protocol will read these).

All hosts on the broadcast LAN (Ethernet is a broadcast multi-access LAN technology) will pick up the frame, read the destination address, and immediately discard the frame, unless they are in the multicast group. This saves PC processing, not LAN bandwidth. Multicasting can cause severe LAN congestion, in some instances, if not implemented carefully.

There are several different groups that users or applications can subscribe to. The range of multicast addresses starts with 224.0.0.0 and goes through 239.255.255.255. As you can see, this range of addresses falls within IP Class D address space based on classful IP assignment.[3]

2.15. SUBNET MASKS

For the subnet address scheme to work, every machine on the network must know which part of the host address will be used as the subnet address. This is accomplished by assigning a *subnet mask* to each machine. A subnet mask is a 32-bit value that allows the recipient of IP packets to distinguish the network ID portion of the IP address from the host ID portion of the IP address. The network administrator creates a 32-bit subnet mask composed of 1s and 0s.

The 1s in the subnet mask represent the positions that refer to the network or subnet addresses. Not all networks need subnets, meaning they use the default subnet mask. This is basically the same as saying that a network doesn't have a subnet address. Table 4 shows the default subnet masks for Classes A, B, and C.

These default masks cannot change. In other words, you can't make a Class B subnet mask read 255.0.0.0. If you try, the host will read that address as invalid and usually won't even let you type it in. For a Class A network, you can't change the first byte in a subnet mask; it must read 255.0.0.0 at a minimum. Similarly, you cannot assign 255.255.255.255, as this is all 1s—a broadcast address. A Class B address must start with 255.255.0.0, and a Class C has to start with 255.255.255.0. [3]

Table 4 Default Subnet Mask [3]

Class	Format	Default Subnet Mask
A	<i>network.node.node.node</i>	255.0.0.0
B	<i>network.network.node.node</i>	255.255.0.0
C	<i>network.network.network.node</i>	255.255.255.0

CHAPTER III

3. THE LAWS AND THEORY ARE USED

3.1. INTRODUCTION:

In this section I will fully explain the main objective of my project research about statistic base DOS. First of all, I established a multi-programs system (visual basic edition 6, fox-pro and Microsoft access) and gathered them to be unified program and to be set up on Windows XP system regardless the edition of XP. This system creates a multi-task server and this program has such a characteristic that enable us to reach the basic information of every IP that enters the server and via this information we can recognize the positive and negative IPs in addition to the time (i.e. hour, minute and second) and the time of entering this IP also the basic information of the IP as follows:

- Classes
- Subnet mask default
- Subnet address of this subnet or wire
- The number of subnet
- The maximum number of subnet
- Ordinal number
- Range form first host to lost host
- Broadcast address of this subnet

Moreover, we can get the database of the system, the diagram, search engine and use it practically through setting it up on the main network or the computer of main server and then controlling the strange IPs. We can also operate this program in private sector companies or institutions at the medium level. By using this program we can control even partially the daily coming attacks on the main services and it is worth to be mentioned that they can deprive many users to enter the websites in minutes.

hence, I will try to explain all the program steps in addition to the theories I relied on to recognize these facts about daily statistic databases and the saved in databases of the system in addition to all the types of IPs. We will provide the structure of the program. I would like to confirm that this was an overview about the program generally and section 3 particularly . first of all we will explain the diagram of the structure in figure 10.

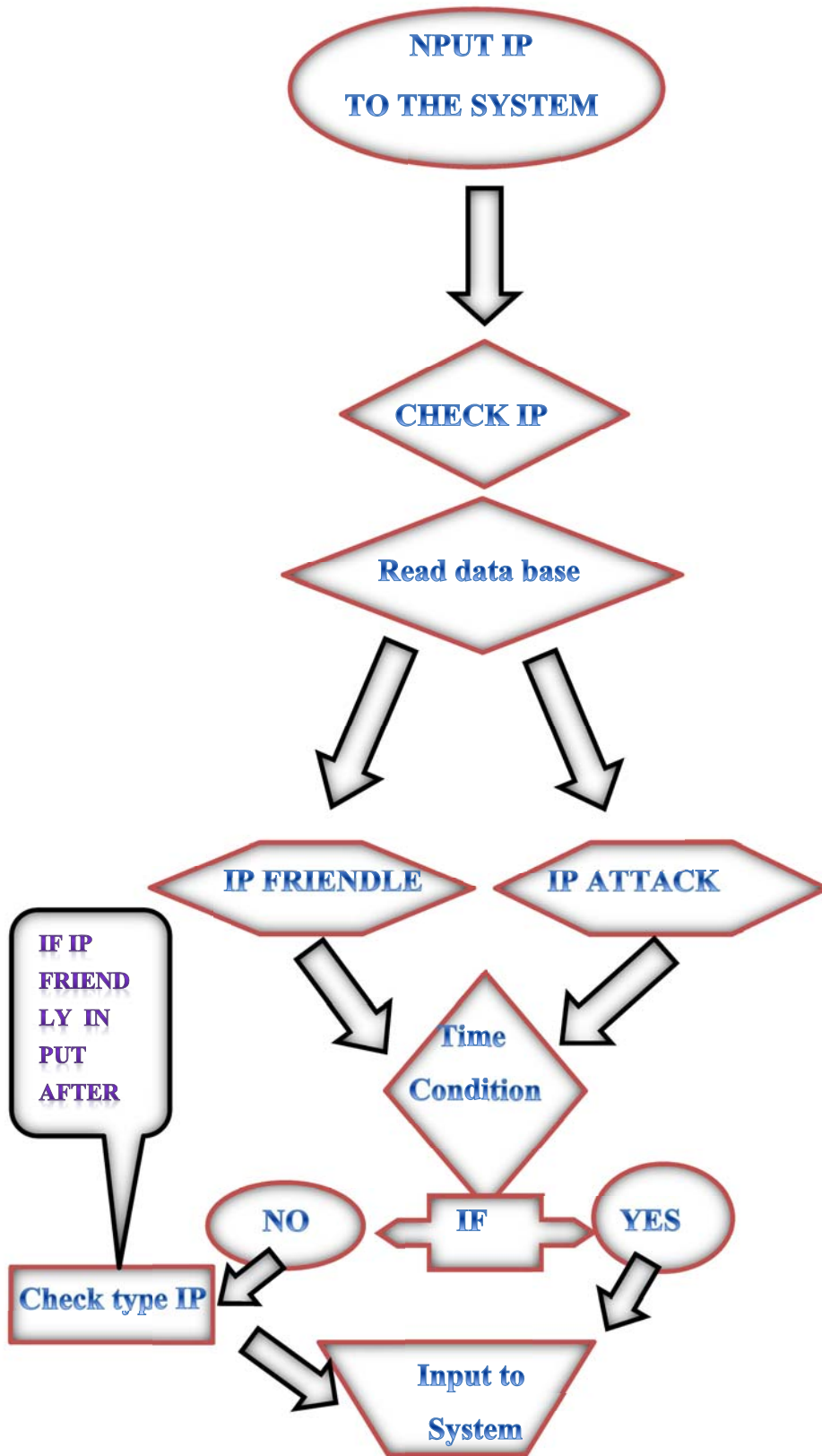


Figure 10 Design of System Public

3.2. THE THEORY OF THE USER IN THE SYSTEM.

3.2.1.Theory I

The relationship between the known and anonymous IPs

We know that here is something is called IP which represents everything in the computer except visual things. This can be specified through the way of dividing the information into smaller parts called packets. The sender sends packets to another device (i.e. router) on the network which uses the same protocol. The second device sends the packets to another device in the same way. This procedure is possible to be repeated until the packets reaches the to receiver which the packets sent to.

This edition that the world deals with. This consists of 32 bits and it can distribute 296.967.294.4 IPs in the world. But as long as this 4 billions will not be sufficient in the future due to the rapid increase of the computer users, for this reason the six edition of IPV6 has been developed which can distribute 6 trillion IPs in the world. The huge difference between two editions of IP structure can be expressed that the sixth edition consists of 128 bits and six decimal numbers instead of the numbers which are existed in the fourth edition.

An example of IP is as follows.

- 10.0.0.0/8
- 14.0.0.0/8
- 127.0.0.0/8
- 128.0.0.0/16
- 169.254.0.0/16
- 172.16.0.0/12
- 191.255.0.0/16
- 192.0.0.0/24
- 192.0.2.0/24
- 192.88.99.0/24
- 192.168.0.0/16
- 198.18.0.0/15
- 223.255.255.0/24
- 224.0.0.0/4
- 240.0.0.0/4
- 255.255.255.255

This license which is provided to all the communication network users particularly in the internet. The data and the information which is sent from a person to another would be missing or random if the IP is not existed. Initially when this program is set up and when an IP is entered, this program checks the IP directly and this check will be recorded in the database. Then there will be confirmation of this IP and automatically imposes the second term and condition of the process that consider the time theory of entering this.

In case the IP is unrecorded in the database, the system puts a sign indicates that this IP is odd and unrecorded. The unrecorded one will be able to enter the system but

will be moved to the second phase (time theory) which accounts the last time that it entered the system. See Figure 11.

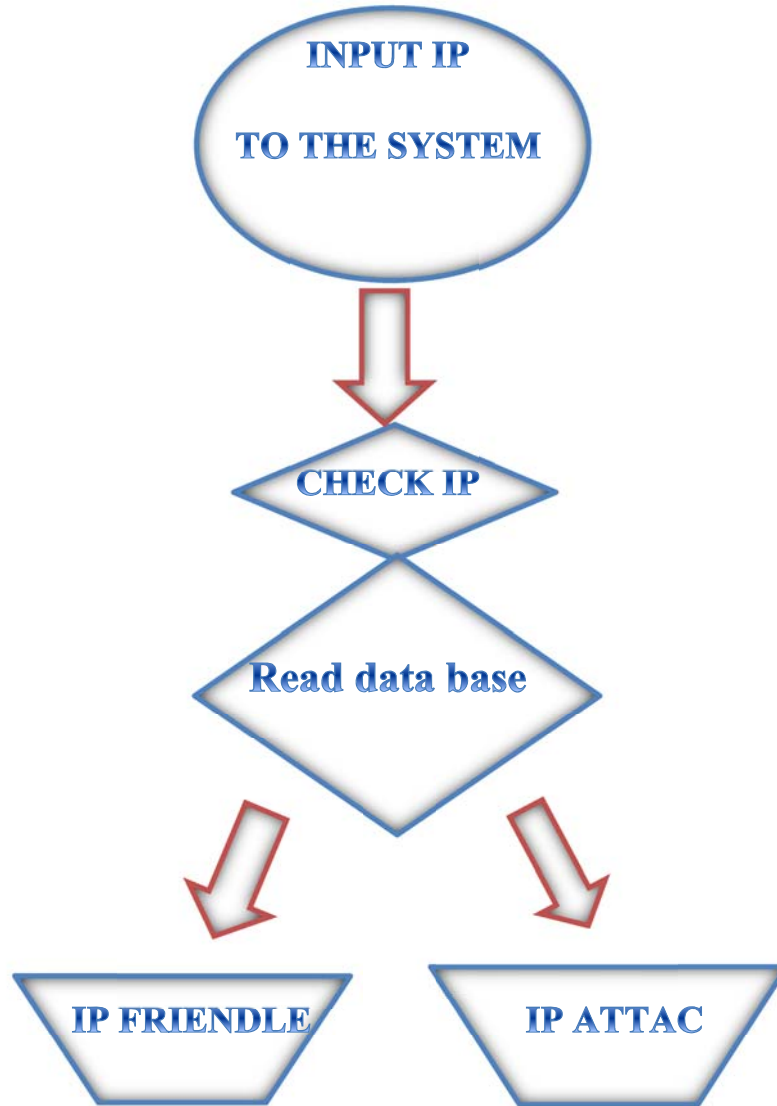


Figure 11 Design of IP CHIK

3.2.2. Theory II

Relationship between the IP and time:

In this method, we will conduct the theoretical timing procedure of the IP regardless whether it the IP is positive or negative. Within 4 seconds it checks all the entering IP into the system. This is the main goal of the program because the time protects the program and the users from deprivation.

Based on this, if the IP entry occurs for the second time is less than 4 seconds then this IP will be hindered to enter the system, otherwise, it can enter if the time elapses within more than 4 seconds.

There are two conditions for the IP to enter the system if it is not according to the specified time:

- ❖ If the IP is positive and recorded in the database, it will be stopped for 5 minutes, and in case the IP tries to enter the system in less than the specified time 4 seconds and within 5 minutes, then it will be blocked and a message will be sent to in not to make the third trial because it is going to be blocked temporarily for 24 hours. If the trail is wrongly continues then the same process will be repeated. This is called friendly attacks.

- ❖ If it is not in accordance with the timing condition and the IP is not recorded in the databases (i.e. ordinary customer) then it will be blocked and not permitted to enter the system for 24 hours. The procedure will be repeated if there is another trial after 24 hours and details of the IP to be sent to the database. If it happens for the third time the IP will be blocked for ever . Refer to the coming Figure 12.

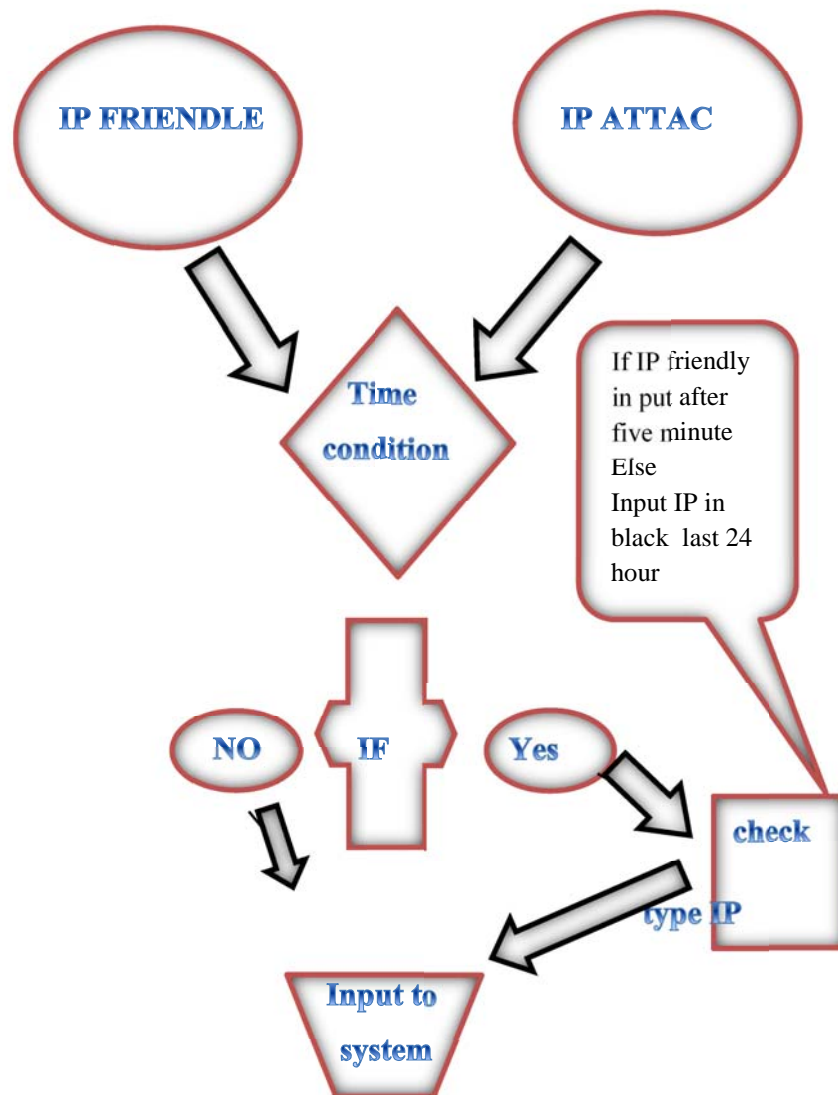


Figure 12 Design of IP Time

3.2.3. Theory III

The rule of accounting the attacks on the system for 24 hours.

This is a statistical system condition for the details of gathering information which enters the system which we can have a special database used to reach any negative IP. This daily statistical system can be considered different from the other systems because it forms daily regular statistical database for unlimited period as explained in the statistical table (3-4).

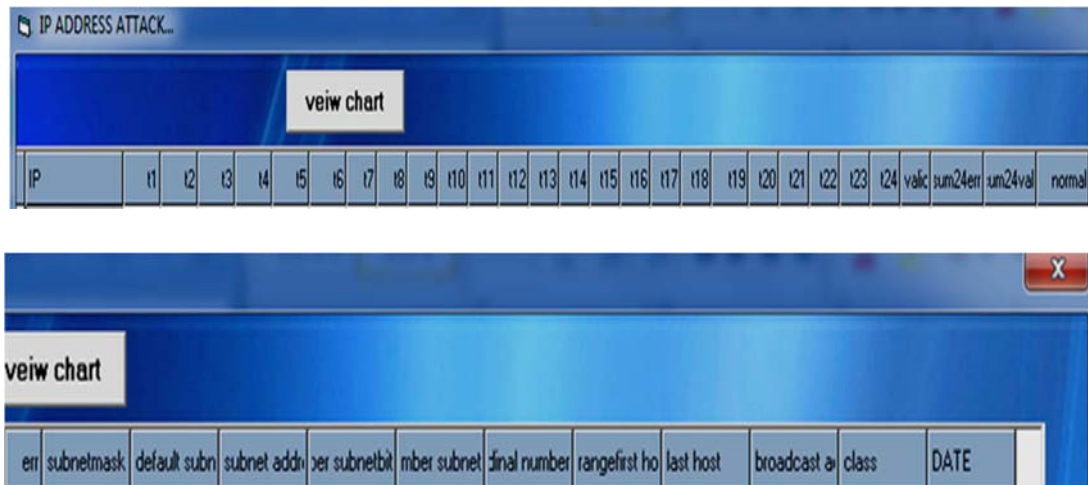


Figure 13 Design of system STATISTICAL BASED daily

3.2.4. Theory V

Information bases on IP :

In this phase, we prepare the important information which the system needs automatically in order to obtain more knowledge about these types of IPs and this important information feeds databases in the system about the entering IPs regardless whether it was positive or negative.

algorithms about the IP information will be setup and benefits to be obtained regarding the identification of positive and negative attacks and this can be a special characteristic of this system. We relied on resources including (the book of Sykes) from Cisco company for communication and network technology. We have quoted

the most important mathematical rules which enabled us to conclude these laws and bases. The most important information which we can recognize when an IP enters is through the explanation of the content of the next informational system. At the beginning we show the next diagram (3-5) and then we provide explanation of some theories and rules.

The screenshot shows a software application window titled "NETWORK INFORMATION". It features a blue background with a light blue abstract graphic on the left. The interface includes the following elements:

- IP ADDRESS:** Four input boxes containing the values 171, 68, 99, and 71.
- subnet mask:** Four input boxes containing the values 255, 255, 240, and 0.
- Action Buttons:** Two blue buttons with white text: "Find information / attack IP" and "Find information / valid IP".
- Result Section:** A blue header bar labeled "Result" followed by several rows of input fields:
 - CLASSES:** One input box.
 - SUBNET MASK DEFAULT:** Four input boxes.
 - The subnet address of this subnet or wire:** Four input boxes.
 - The number of Subnet bit:** One input box.
 - The maximum number of subnet:** One input box.
 - ordinal number:** One input box.
 - Range from first host to last host:** Two rows of four input boxes each.
 - Broadcast address of this subnet:** Four input boxes.

Figure 14 Design of system IP information

3.3.DEFINITIONS :

3.3.1.Classes :

The designers of the Internet determined to create sorts of networks relying on network size. For the small number of networks possessing a very huge number of nodes, they produced the rank *Class A network*. At the other extreme is the *Class C network*, which is reserved for the ample networks with a small number of nodes. The class distinction for networks between very large and very small is expectedly known as the *Class B network*.

Class A: network Part of the Internet Protocol hierarchical addressing scheme. Class A networks have only 8 bits for defining networks and 24 bits for defining hosts and subnets on each network.

Class B: network Part of the Internet Protocol hierarchical addressing scheme. Class B networks have 16 bits for defining networks and 16 bits for defining hosts and subnets on each network.

Class C: network Part of the Internet Protocol hierarchical addressing scheme. Class C networks have 24 bits for defining networks and only 8 bits for defining hosts and subnets on each network.[3]

Table 5 Class A network [3]

A	<i>network. host . host</i>	255.0.0.0
B	<i>network. network. host host</i>	255.255.0.0
C	<i>network. network. network. host</i>	255.255.255.0

3.3.2. Subnet mask default:

In order the subnet address scheme to work, every machine on the network must have an idea which part of the host address will be utilized as the subnet address. This is done by supporting a Subnet mask to each machine. A subnet mask is a 32-bit value that permits the recipient of IP packets to distinguish the network ID portion of the IP address from the host ID portion of the IP address. The network administrator produces a 32-bit subnet mask composed involves 1s and 0s. The 1s in the subnet mask represent the positions that point to the network or subnet addresses. Not all networks need subnets, this mean that they use the default subnet mask. This is basically the same as stating that a network lacks a subnet address. [3]

3.3.3. Broadcast address of this subnet:

Most the majority of people use the term *broadcast* as a generic term, and most of the time, we understand what they mean. But not always. For example, you might say, “The host broadcasted through a router to a DHCP server,” but, well, it’s pretty unlikely that this would ever really happen. What you probably mean—using the correct technical jargon—is, “The host broadcasted for an IP address; a router then forwarded this as a unicast packet to the DHCP server.” Oh, and don’t forget that with IPv4, broadcasts are very important, but with IPv6, there aren’t any broadcasts sent at all—now there’s something to get you excited about when you get to Chapter 13! Okay, I’ve referred to broadcast addresses throughout Chapters 1 and 2, and even showed you some examples. But I really haven’t gone into the different terms and

uses associated with them yet, and it's about time I did. So here are the four different broadcasts (generic term *broadcast*) types that I'd like to define introduce to you:

Layer 2 broadcasts These are sent to all nodes on a LAN.

Broadcasts (layer 3) These are sent to all nodes on the network.

Unicast These are sent to a single destination host.

Multicast These are packets sent from a single source and transmitted to many devices on different networks.[3]

3.4 EXPLAIN PLANNED IP INFORMATION :

At this in this regard, it's important that you both understood and memorized your powers of 2. Please refer to the sidebar "Understanding the Powers of 2" earlier in this chapter if you need some support. Here's how you get the answers to those five big questions:

_ How many subnets?

$2^x =$ number of subnets. x is the number of masked bits, or the 1s. For example, in 11000000, the number of 1s gives us 22 subnets. In this example, there are 4 subnets.

_ How many hosts per subnet?

$2^y - 2 =$ number of hosts per subnet. y is the number of unmasked bits, or the 0s. For example, in 11000000, the number of 0s gives us $26 - 2$ hosts. In this example, there are 62 hosts per subnet. You need to subtract 2 for the subnet address and the broadcast address, which are not valid hosts.

_ What are the valid subnets?

$256 - \text{subnet mask} =$ block size, or increment number. An example would be $256 - 192 = 64$. The block size of a 192 mask is always 64. Start counting at zero in blocks

of 64 until you reach the subnet mask value and these are your subnets. 0, 64, 128, 192. Easy, huh?

_ What's the broadcast address for each subnet? Now here's the really easy part. Since we counted our subnets in the last section as 0, 64, 128, and 192, the broadcast address is always the number right before the next subnet. For example, the 0 subnet has a broadcast address of 63 because the next subnet is 64. The 64 subnet has a broadcast address of 127 because the next subnet is 128. And so on. And remember, the broadcast address of the last subnet is mostly 255.

_ What are the valid hosts?

Valid hosts are the numbers between the subnets, deleting the all 0s and all 1s. For example, if 64 is the subnet number and 127 is the broadcast address, then 65–126 is the valid host range—it's always the numbers between the subnet address and the broadcast address. [3]

CHAPTER IV

3. RESULTS, APPLICATIONS AND SOFTWARE CODE APPLICATION OF PROGRAM STATISTICS OF BASE ATTACKS DETECTION

4.1. INTRODACTION TO PROGRAM

Design and programming of the system base statistics of detecting attacks program in Microsoft Visual Basic 6.0, for ease of use for everyone, as well as a universal language with a good level in the application and implementation.

We ordered at the beginning and at the entrance to the Execution of the program window where it appears to us first, as in Figure (15), a security system if the window is the application user name and password so as to protect software from unauthorized use of this system. And In this window, is an edit and delete users and add both cases and new to the program.



Figure 15 login the security a window

In this window, as in Figure (16) is an edit and delete users and add both cases and new to the program.



Figure 16 used a window adjustments

In this step we are no special Login code security programmatic for the program, as follows:

```

Private Sub Command1_Click()
Data1.Recordset.AddNew
Data1.Recordset.Fields(0) = t1.Text
Data1.Recordset.Fields(1) = t2.Text

End Sub
Private Sub Command2_Click()
n = MsgBox("are you sure you want to delete this record", vbOKCancel)
If n = vbOK Then
Data1.Recordset.Delete
End If
End Sub

Private Sub Command3_Click()
Dim x, n As String
x = InputBox("Enter user name")
n = "nnn1 like" & x & ""
Data1.Recordset.FindFirst n
t1.Text = Data1.Recordset.Fields(0)
t2.Text = Data1.Recordset.Fields(1)

If Data1.Recordset.NoMatch Then
MsgBox "the record not found", vbOKOnly + vbCritical, "the user name search"
End If

End Sub

Private Sub Command4_Click()
Data1.RecordSource = ("select * from tab2 where nnn1 = " & t1.Text & " and nnn2 =" & t2.Text & "")
Data1.Refresh

```

```

If Data1.Recordset.RecordCount = 0 Then

    MsgBox ("Invalid uername or passward")
Else
Command1.Enabled = True
Command2.Enabled = True
Command3.Enabled = True
Command6.Enabled = True
Command7.Enabled = True
End If
t1.Text = ""
t2.Text = ""
End Sub

Private Sub Command5_Click()
End
End Sub

Private Sub Command6_Click()
Data1.Recordset.Fields(0) = t1.Text
Data1.Recordset.Fields(1) = t2.Text

Data1.Recordset.Update
Data1.Refresh
MsgBox ("the new data is saved")
Data1.Refresh
End Sub

Private Sub Command7_Click()
Data1.Recordset.Edit
Data1.Recordset.Fields(0) = t1.Text
Data1.Recordset.Fields(1) = t2.Text
Data1.Recordset.Update
MsgBox ("data is saved")
End Sub

Private Sub Data1_Validate(Action As Integer, Save As Integer)

End Sub

Private Sub Form_Load()
Data1.DatabaseName = App.Path & "\valip.mdb"
Data1.RecordSource = "tab2"
End Sub

```

4.2. STATISTICAL BASED DETECTION OF DOS ATTACKS :

Having been to enter the program seems to us the main window of the program through which applications are identified on the programming of the system and also in the following figure (17) .

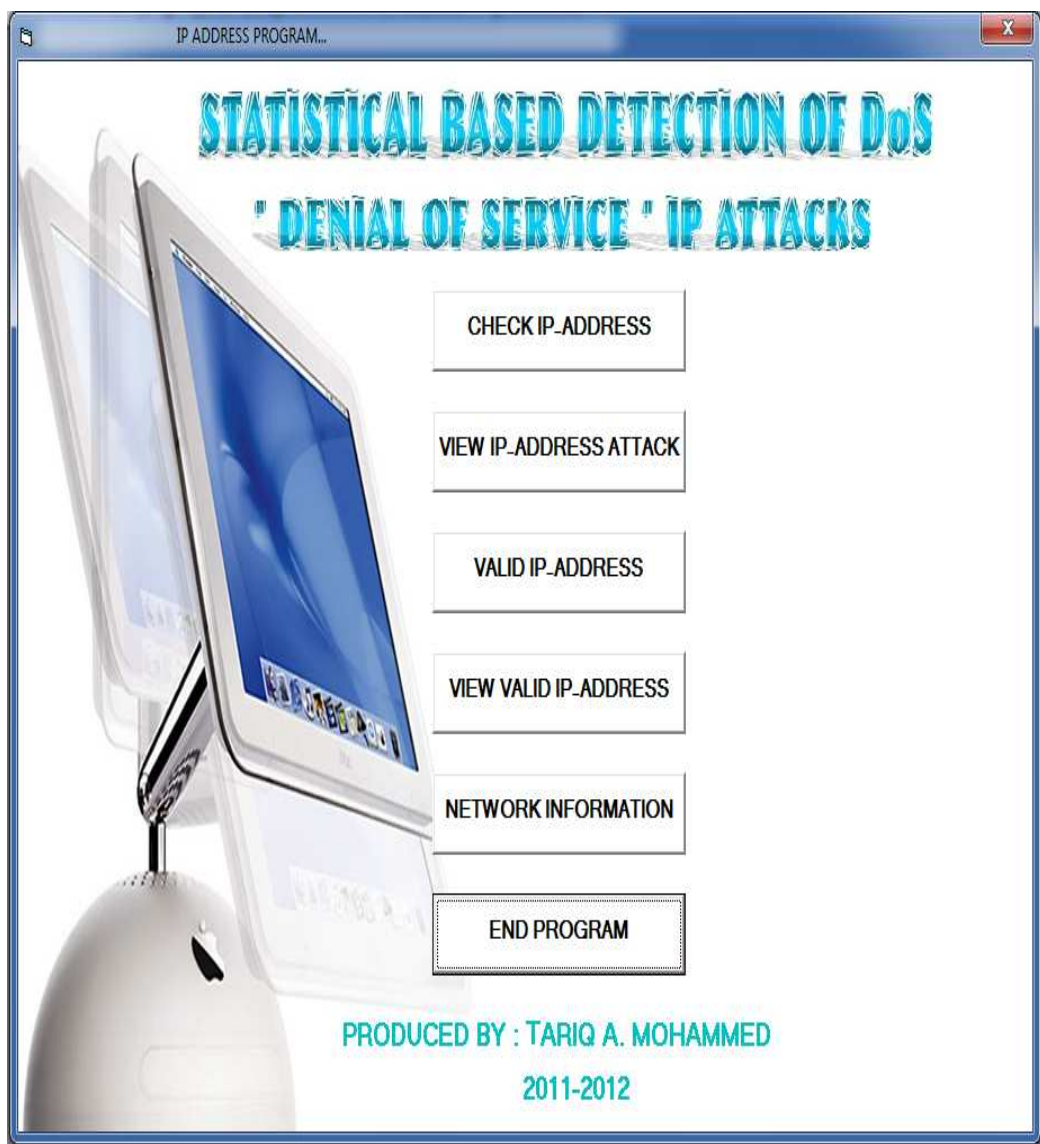


Figure 17 The Main Window Program

Options of this display in the main window as shown above. These are :

- **Check IP _Address .**
- **View IP –Address Attack .**
- **Valid IP – Address .**
- **View Valid IP –Address .**
- **Network Information .**
- **End Program .**

4.2.1. Check IP _Address:

In this window to As in Figure (18) has a program button on enter IP and time and history bound with the current time of the calculator and check and test the IP and know what type of IP is home of the IP friend or enemy, is enter the values IP and find out the results in the same window or in the Applied window for the table .

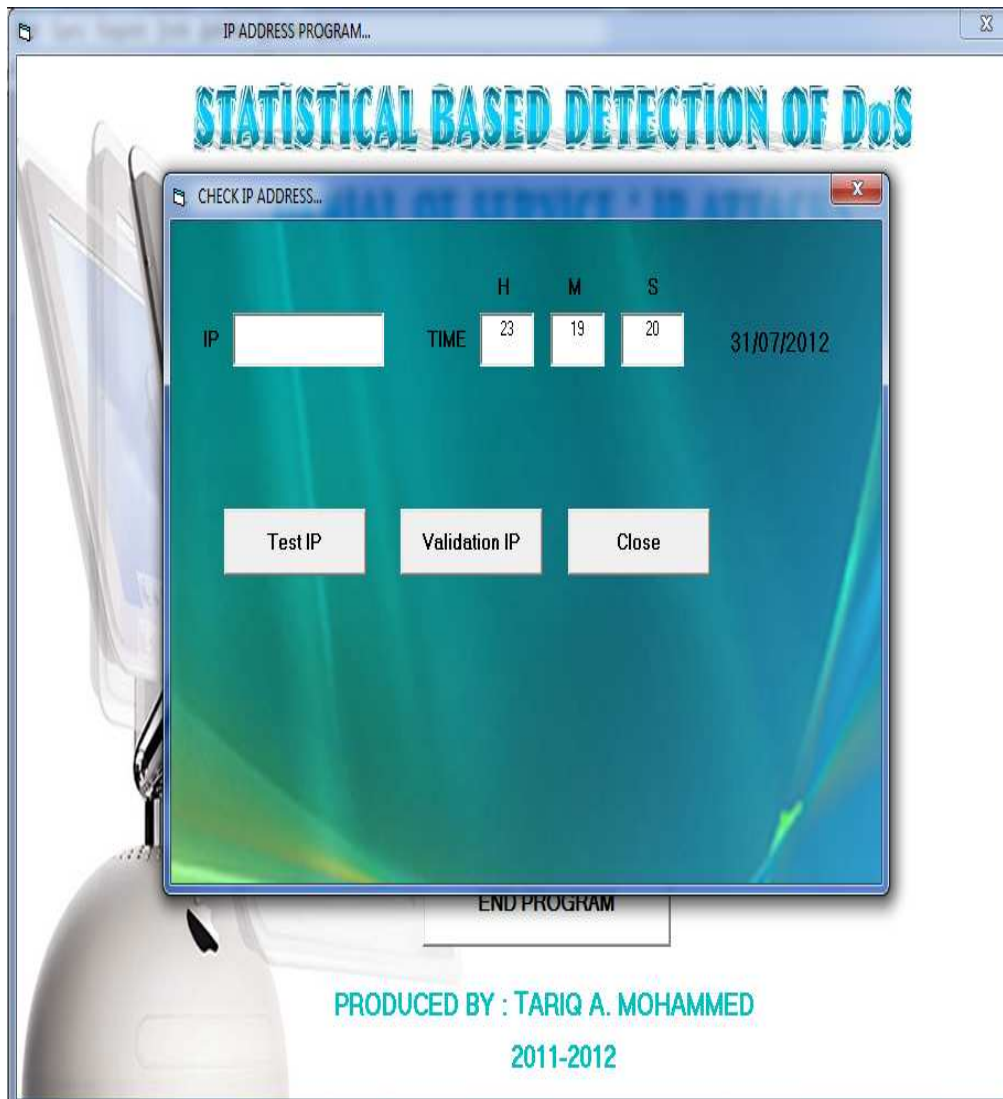


Figure 18 Check IP _Address

In this step, we write your code programmatic (Check IP _Address) and as follows:

```
Public Function Valid_IP(IP As String) As Boolean
    Dim i As Integer
    Dim n1 As Integer
    Dim n2 As String
    Dim n3
    IP = Trim$(IP)
    If Len(IP) < 4 Then
        Valid_IP = False
```



```

MsgBox IP & " IS INVALID", , "IP Validator"
Text1.Text = ""
Exit Function
End If
i = 1
n1 = 0
For i = 1 To Len(IP)
    If Mid$(IP, i, 1) = "." Then
        n1 = n1 + 1
        n2 = ""
        If i = Len(IP) Then
            Valid_IP = False
            MsgBox IP & " IS INVALID", , "IP Validator"
            Text1.Text = ""
            Exit Function
        End If
    Else
        n2 = n2 & Mid$(IP, i, 1)
        On Error Resume Next
        n3 = CByte(n2)
        If (Err) Then
            Valid_IP = False
            MsgBox IP & " IS INVALID", , "IP Validator"
            Text1.Text = ""
            Exit Function
        End If
    End If
Next i
If n1 <> 3 Then
    Valid_IP = False
    MsgBox IP & " IS INVALID", , "IP Validator"
    Text1.Text = ""
    Exit Function
End If
Valid_IP = True
MsgBox IP & " is Valid", , "IP Validator"
End Function
Private Sub Command3_Click()
    If Len(Text1) = 0 Then
        MsgBox "Please type an IP Address in the textbox.", , "IP Validator"
    Else
        Valid_IP Text1
    End If
End Sub
Private Sub Data1_Validate(Action As Integer, Save As Integer)
End Sub
Private Sub Form_Load()
Label6.Caption = Date
Text2.Text = Hour(Time)
Text3.Text = Minute(Now)
Text4.Text = Second(Now)
Data2.DatabaseName = App.Path & "\attack.mdb"
Data2.RecordSource = "tab1"
Data1.DatabaseName = App.Path & "\valip.mdb"

```

```

Data1.RecordSource = "tab1"
End Sub
Private Sub Command1_Click()
Label6.Caption = Date
If Len(Text1) = 0 Then
    MsgBox "Please type an IP Address in the textbox.", , "IP Validator"
Else
    Valid_IP Text1
End If
nv = 0: v = 0: f = 0: nvno = 0
Data1.Recordset.MoveFirst
Do While Data1.Recordset.EOF = False
    x = Data1.Recordset.Fields("IP").Value
    If x = Text1.Text Then
        h1 = Data1.Recordset.Fields("hour").Value
        m1 = Data1.Recordset.Fields("minute").Value
        s1 = Data1.Recordset.Fields("second").Value
        If h1 = Text2 And m1 = Text3 And (Text4 - s1) <= 4 Then
            nv = 1: nvno = Data1.Recordset.Fields("count").Value
        Else
            v = 1
            a1 = Data1.Recordset.Fields("count").Value + 1
            Data1.Recordset.Edit
            Data1.Recordset!Count = a1
            Data1.Recordset!Hour = Text2
            Data1.Recordset!Minute = Text3
            Data1.Recordset!Second = Text4
            Data1.Recordset.Update
        End If
    End If
    Data1.Recordset.MoveNext
Loop
Data1.Recordset.MoveFirst
'-----
If v = 0 Then
On Error Resume Next
Data2.Recordset.MoveFirst
Do While Data2.Recordset.EOF = False
    x = Data2.Recordset.Fields("IP").Value
    If x = Text1 Then
        f = 1
        h2 = Text2
        Select Case h2
        Case 1
            Y = Data2.Recordset.Fields("t1").Value
            Y = Y + 1
            Data2.Recordset.Edit
            Data2.Recordset!t1 = Y
        Data2.Recordset.Fields("subnetmask").Value = "255.255.255.255"
            Data2.Recordset.Update
        Case 2
            Y = Data2.Recordset.Fields("t2").Value
            Y = Y + 1
            Data2.Recordset.Edit

```

```

    Data2.Recordset!t2 = Y
Data2.Recordset!subnetmask = "255.255.255.255"
    Data2.Recordset.Update
Case 3
    Y = Data2.Recordset.Fields("t3").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t3 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 4
    Y = Data2.Recordset.Fields("t4").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t4 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 5
    Y = Data2.Recordset.Fields("t5").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t5 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 6
    Y = Data2.Recordset.Fields("t6").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t6 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 7
    Y = Data2.Recordset.Fields("t7").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t7 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 8
    Y = Data2.Recordset.Fields("t8").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t8 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 9
    Y = Data2.Recordset.Fields("t9").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t9 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 10
    Y = Data2.Recordset.Fields("t10").Value

```

```

    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t10 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 11
    Y = Data2.Recordset.Fields("t11").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t11 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 12
    Y = Data2.Recordset.Fields("t12").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t12 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 13
    Y = Data2.Recordset.Fields("t13").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t13 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 14
    Y = Data2.Recordset.Fields("t14").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t14 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 15
    Y = Data2.Recordset.Fields("t15").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t15 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 16
    Y = Data2.Recordset.Fields("t16").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t16 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
Case 17
    Y = Data2.Recordset.Fields("t17").Value
    Y = Y + 1
    Data2.Recordset.Edit
    Data2.Recordset!t17 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update

```

```

Case 18
  Y = Data2.Recordset.Fields("t18").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t18 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 19
  Y = Data2.Recordset.Fields("t19").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t19 = Y
Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 20
  Y = Data2.Recordset.Fields("t20").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t20 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 21
  Y = Data2.Recordset.Fields("t21").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t21 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 22
  Y = Data2.Recordset.Fields("t22").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t22 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 23
  Y = Data2.Recordset.Fields("t23").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t23 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
Case 24
  Y = Data2.Recordset.Fields("t24").Value
  Y = Y + 1
  Data2.Recordset.Edit
  Data2.Recordset!t24 = Y
  Data2.Recordset.Fields(30) = "255.255.255.255"
  Data2.Recordset.Update
End Select
'-----
      Data2.Recordset!Date = Text7.Text
s1 = Data2.Recordset.Fields("t1").Value + Data2.Recordset.Fields("t2").Value +
Data2.Recordset.Fields("t3").Value

```

```

s2 = Data2.Recordset.Fields("t4").Value + Data2.Recordset.Fields("t5").Value +
Data2.Recordset.Fields("t6").Value
s3 = Data2.Recordset.Fields("t7").Value + Data2.Recordset.Fields("t8").Value +
Data2.Recordset.Fields("t9").Value
s4 = Data2.Recordset.Fields("t10").Value + Data2.Recordset.Fields("t11").Value +
Data2.Recordset.Fields("t12").Value
s5 = Data2.Recordset.Fields("t13").Value + Data2.Recordset.Fields("t14").Value +
Data2.Recordset.Fields("t15").Value
s6 = Data2.Recordset.Fields("t16").Value + Data2.Recordset.Fields("t17").Value +
Data2.Recordset.Fields("t18").Value
s7 = Data2.Recordset.Fields("t19").Value + Data2.Recordset.Fields("t20").Value +
Data2.Recordset.Fields("t21").Value
s8 = Data2.Recordset.Fields("t22").Value + Data2.Recordset.Fields("t23").Value +
Data2.Recordset.Fields("t24").Value
Sum = s1 + s2 + s3 + s4 + s5 + s6 + s7 + s8
Data2.Recordset.Edit
Data2.Recordset!sum24err = Sum
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update
If nv = 1 Then
Data2.Recordset.Edit
Data2.Recordset!valid = "yes"
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update
Else
Data2.Recordset.Edit
Data2.Recordset!valid = "no"
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update
End If
Normal1 = nvno * 100 / (nvno + Sum)
Err1 = Sum * 100 / (nvno + Sum)
Data2.Recordset.Edit
Data2.Recordset!Normal = Normal1
Data2.Recordset!Err = Err1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update
End If
Data2.Recordset.MoveNext
Loop
If f = 0 Then
Data2.Recordset.MoveLast
Text5.Text = Text1.Text
Text7.Text = Label6.Caption
Data2.Recordset.AddNew
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update
Data2.Refresh
Data2.Recordset.MoveFirst
Do While Data2.Recordset.EOF = False
x = Data1.Recordset.Fields("IP").Value
If x = Text1.Text Then
Select Case h2
Case 1

```

Data2.Recordset.Edit
Data2.Recordset!t1 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 2
Data2.Recordset.Edit
Data2.Recordset!t2 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 3
Data2.Recordset.Edit
Data2.Recordset!t3 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 4
Data2.Recordset.Edit
Data2.Recordset!t4 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 5
Data2.Recordset.Edit
Data2.Recordset!t5 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 6
Data2.Recordset.Edit
Data2.Recordset!t6 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 7
Data2.Recordset.Edit
Data2.Recordset!t7 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 8
Data2.Recordset.Edit
Data2.Recordset!t8 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 9
Data2.Recordset.Edit
Data2.Recordset!t9 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 10
Data2.Recordset.Edit
Data2.Recordset!t10 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 11
Data2.Recordset.Edit
Data2.Recordset!t11 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 12
Data2.Recordset.Edit
Data2.Recordset!t12 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 13
Data2.Recordset.Edit
Data2.Recordset!t13 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 14
Data2.Recordset.Edit
Data2.Recordset!t14 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 15
Data2.Recordset.Edit
Data2.Recordset!t15 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 16
Data2.Recordset.Edit
Data2.Recordset!t16 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 17
Data2.Recordset.Edit
Data2.Recordset!t17 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 18
Data2.Recordset.Edit
Data2.Recordset!t18 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 19
Data2.Recordset.Edit
Data2.Recordset!t19 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 20
Data2.Recordset.Edit
Data2.Recordset!t20 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 21
Data2.Recordset.Edit
Data2.Recordset!t21 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"
Data2.Recordset.Update

Case 22
Data2.Recordset.Edit
Data2.Recordset!t22 = 1
Data2.Recordset.Fields(30) = "255.255.255.255"


```

        Data2.Recordset.Update
    Case 23
        Data2.Recordset.Edit
        Data2.Recordset!t23 = 1
        Data2.Recordset.Fields(30) = "255.255.255.255"
        Data2.Recordset.Update
    Case 24
        Data2.Recordset.Edit
        Data2.Recordset!t24 = 1
        Data2.Recordset.Fields(30) = "255.255.255.255"
        Data2.Recordset.Update
End Select
'-----
s1 = Data2.Recordset.Fields("t1").Value + Data2.Recordset.Fields("t2").Value +
Data2.Recordset.Fields("t3").Value
s2 = Data2.Recordset.Fields("t4").Value + Data2.Recordset.Fields("t5").Value +
Data2.Recordset.Fields("t6").Value
s3 = Data2.Recordset.Fields("t7").Value + Data2.Recordset.Fields("t8").Value +
Data2.Recordset.Fields("t9").Value
s4 = Data2.Recordset.Fields("t10").Value + Data2.Recordset.Fields("t11").Value +
Data2.Recordset.Fields("t12").Value
s5 = Data2.Recordset.Fields("t13").Value + Data2.Recordset.Fields("t14").Value +
Data2.Recordset.Fields("t15").Value
s6 = Data2.Recordset.Fields("t16").Value + Data2.Recordset.Fields("t17").Value +
Data2.Recordset.Fields("t18").Value
s7 = Data2.Recordset.Fields("t19").Value + Data2.Recordset.Fields("t20").Value +
Data2.Recordset.Fields("t21").Value
s8 = Data2.Recordset.Fields("t22").Value + Data2.Recordset.Fields("t23").Value +
Data2.Recordset.Fields("t24").Value
Sum = s1 + s2 + s3 + s4 + s5 + s6 + s7 + s8
    Data2.Recordset.Edit
    Data2.Recordset!sum24err = Sum
    Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
If nv = 1 Then
    Data2.Recordset.Edit
    Data2.Recordset!valid = "yes"
    Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
End If
Normal1 = nvno * 100 / (nvno + Sum)
Err1 = Sum * 100 / (nvno + Sum)
    Data2.Recordset.Edit
    Data2.Recordset!Normal = Normal1
    Data2.Recordset!Err = Err1
    Data2.Recordset.Fields(30) = "255.255.255.255"
    Data2.Recordset.Update
End If
Data2.Recordset.MoveNext
Loop
'-----
End If
End If
End Sub

```

```
Private Sub Command2_Click()  
Form1.Hide  
End Sub
```

4.2.2. View IP –Address Attack:

In this window applied in Figure (19) have been clarified details of the rule STATISTICS daily for 24 continuous hours on the results of entering the ip to the system and periodically if it is to know login ip to the system at any time of the day as well as at any date and know what type of ip and the total entry and the number of errors that were committed as well as knowledge and data, including:

- Classes
- Subnet mask default
- Subnet address of this subnet or wire
- The number of subnet
- The maximum number of subnet
- Ordinal number
- Range from first host to lost host
- Broadcast address of this subnet

IP	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19	t20	t21	t22	t23	t24	valid	sum24err	sum24val	normal	err	subn
133.144.12.12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	255.
169.124.102.112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	yes	2	0	33 67 255.
169.128.103.132	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	no	8	0	0 100 255.
169.126.104.182	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	no	4	0	0 100 255.
169.128.103.152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	no	3	0	0 100 255.
169.128.103.123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	no	3	0	0 100 255.
169.124.102.122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	yes	2	0	33 67 255.
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 255.

Figure 19 View IP –Address Attack

In this step, we write your code programmatic (View IP-Address Attack) and as follows:

```
Private Sub Data1_Validate(Action As Integer, Save As Integer)
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Data1.DatabaseName = App.Path & "\attack.mdb"
```

```
Data1.RecordSource = "tab1"
```

```
MSFlexGrid1.ColWidth(0) = 200
```

```
MSFlexGrid1.ColWidth(1) = 1300
```

```
MSFlexGrid1.ColWidth(2) = 400
```

```
MSFlexGrid1.ColWidth(3) = 400
```

```
MSFlexGrid1.ColWidth(4) = 400
```

```
MSFlexGrid1.ColWidth(5) = 400
```

```
MSFlexGrid1.ColWidth(6) = 400
```

```
MSFlexGrid1.ColWidth(7) = 400
```

```
MSFlexGrid1.ColWidth(8) = 400
```

```
MSFlexGrid1.ColWidth(9) = 400
```

```
MSFlexGrid1.ColWidth(10) = 400
```

```
MSFlexGrid1.ColWidth(11) = 400
```

```
MSFlexGrid1.ColWidth(12) = 400
MSFlexGrid1.ColWidth(13) = 400
MSFlexGrid1.ColWidth(14) = 400
MSFlexGrid1.ColWidth(15) = 400
MSFlexGrid1.ColWidth(16) = 400
MSFlexGrid1.ColWidth(17) = 400
MSFlexGrid1.ColWidth(18) = 400
MSFlexGrid1.ColWidth(19) = 400
MSFlexGrid1.ColWidth(20) = 500
MSFlexGrid1.ColWidth(21) = 400
MSFlexGrid1.ColWidth(22) = 400
MSFlexGrid1.ColWidth(23) = 400
MSFlexGrid1.ColWidth(24) = 400
MSFlexGrid1.ColWidth(25) = 400
MSFlexGrid1.ColWidth(26) = 400
MSFlexGrid1.ColWidth(27) = 700
MSFlexGrid1.ColWidth(28) = 700
MSFlexGrid1.ColWidth(29) = 700
MSFlexGrid1.ColWidth(30) = 400
```

```
MSFlexGrid1.ColWidth(7) = 500
MSFlexGrid1.ColWidth(2) = 500
MSFlexGrid1.ColWidth(3) = 500
MSFlexGrid1.ColWidth(4) = 500
MSFlexGrid1.ColWidth(5) = 500
MSFlexGrid1.ColWidth(6) = 500
MSFlexGrid1.ColWidth(7) = 500
MSFlexGrid1.CellAlignment = vbCenter
```

```
MSFlexGrid1.RowHeight(0) = 400
```

End Sub

In the following window is identified on the illustrative scheme for IP ADDRESS CHART As in Figure (20) Shown in each of :

- SUM 24 ERR
- SUM 24 VAL
- NORMAL
- ERR

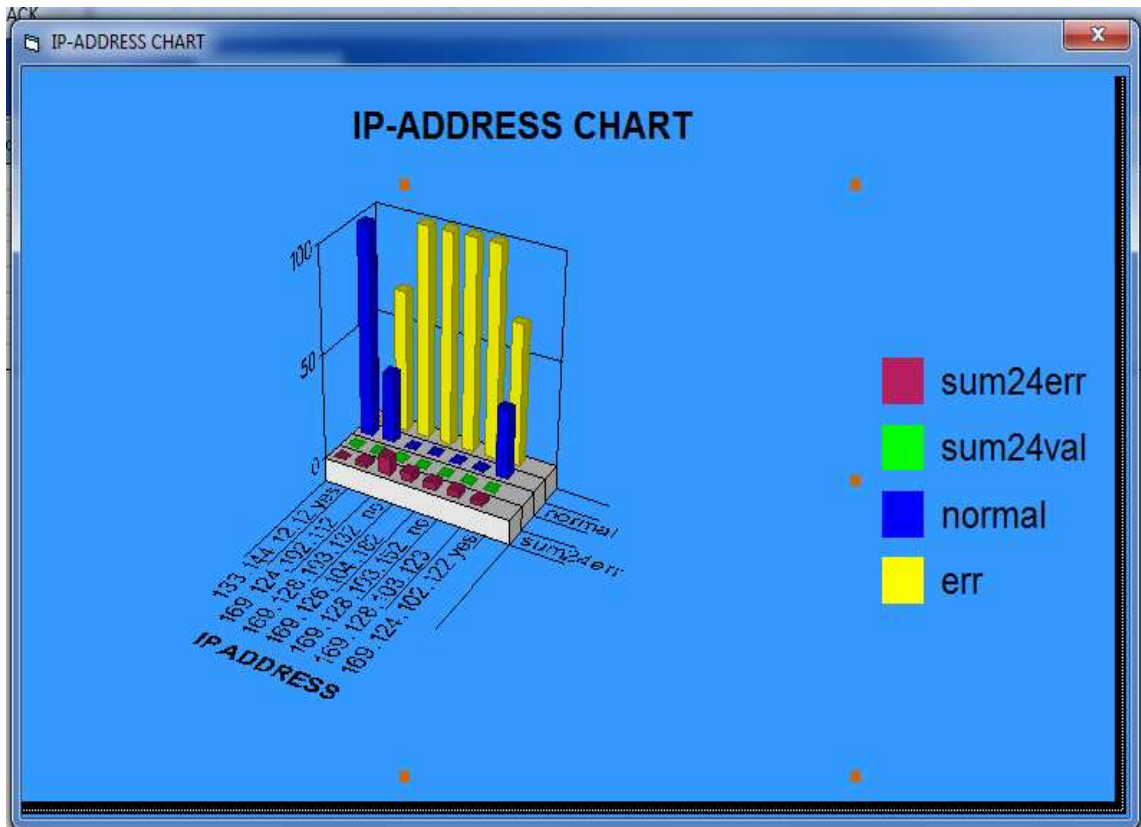


Figure 20 IP ADDRESS CHART

4.2.3. Search Valid IP – Address :

In this window, as in Figure (21) we will conduct the search for a specific IP and specific friend and knowledge at the base of data on the time history of entry and the number of times access to the system since the first recorded in the system and can also be an amendment to the IP through the addition or delete



Figure 21 Search Valid IP – Address

In this step, we write your code programmatic (Search Valid IP – Address) and as follows:

```
Private Sub Command10_Click()
Dim x, n As String
x = InputBox(" Enter IP ADDRESS ")
n = "IP like" & x & ""
Data1.Recordset.FindFirst n
If Data1.Recordset.NoMatch Then
MsgBox "Record not found ", vbOKOnly + vbCritical, "Find Record"
End If
End Sub
Private Sub Command13_Click()
Unload Me
End Sub
Private Sub Command5_Click()
Data1.Recordset.AddNew
End Sub
Private Sub Command6_Click()
x = MsgBox("ää ÊÑĩ İĐÝ ÇáÓİá", vbOKCancel)
If x = vbOK Then
Data1.Recordset.Delete
End If
End Sub
Private Sub Command7_Click()
Data1.Recordset.Update
MsgBox ("Êã ÍÝÙ ÇáÈíÇäÇÊ")
End Sub
Private Sub Data1_Validate(Action As Integer, Save As Integer)
End Sub
Private Sub Form_Load()
Data1.DatabaseName = App.Path & "\valip.mdb"
Data1.RecordSource = "tab1"
End Sub
Private Sub Text1_Change()
End Sub
Private Sub Text7_Change()
End Sub
```

4.2.4. View Valid IP –Address :

In the following window as in Figure (22) is where the supply of all information on IP knowledge by the time the hour, minute and second and the total entry of the IP and known for Classes, Subnet mask default, Subnet address, The number of subnet, The maximum number of subnet, Ordinal number, Range from first host to lost host, Broadcast address of this subnet .

IP	hour	minute	second	count	subnet mask	default subnetma	subnet addr	per subnetbit	mber subnet	final number	range	first ho	le
169.124.102.112	22	36	58	1									
169.124.102.122	22	52	55	1									
169.124.102.132	0		0	0									
169.124.102.142	0		0	0									
169.124.102.152	0		0	0									
169.124.102.162	0		0	0									
169.124.102.172	0		0	0									
196.124.102.182	0		0	0									
169.124.102.192	0		0	0									
169.124.102.102	0		0	0									
169.128.103.113	0		0	0									
169.128.103.123	0		0	0									
169.128.103.133	0		0	0									
169.128.103.143	0		0	0									
169.128.103.153	0		0	0									
169.128.103.163	0		0	0									
169.128.103.173	0		0	0									
169.128.103.183	0		0	0									
169.128.103.193	0		0	0									
169.128.103.103	0		0	0									

Figure 22 View Table Valid IP –Address

In this step, we write your code programmatic (View Table Valid IP –Address) and as follows:

```

Dim DB As ADODB.Connection
Dim RS As ADODB.Recordset
Private Sub Adodc1_WillMove(ByVal adReason As ADODB.EventReasonEnum, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
End Sub
Private Sub Combo1_click()
On Error Resume Next
On Error Resume Next
'Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;data source=" & App.Path &
"\valip.mdb;persist security info=false"
'Adodc1.RecordSource = "select * from tab1"
'Adodc1.CommandType = adCmdText
'Adodc1.Refresh
If Combo1.Text = "by hour only" Then
Adodc1.RecordSource = "select * from tab1 where ( hour like '%' & Text2.Text & '%' ) "
Adodc1.CommandType = adCmdText
Adodc1.Refresh
Elseif Combo1.Text = "by hour & minute only" Then
Adodc1.RecordSource = "select * from tab1 where ( hour like '%' & Text2.Text & '%' ) and ( minute like '%' &
Text2.Text & '%' ) "
Adodc1.CommandType = adCmdText
Adodc1.Refresh
Elseif Combo1.Text = "by hour & minute & second" Then
Adodc1.RecordSource = "select * from tab1 where ( hour like '%' & Text2.Text & '%' ) and ( minute like '%' &
Text3.Text & '%' )and ( second like '%' & Text4.Text & '%' ) "
Adodc1.CommandType = adCmdText
Adodc1.Refresh
Else
Adodc1.RecordSource = "select * from tab1 "
Adodc1.CommandType = adCmdText
Adodc1.Refresh
End If
Adodc1.Refresh
Adodc1.Recordset.Update
End Sub
Private Sub Form_Load()
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;data source=" & App.Path &
"\valip.mdb;persist security info=false"
Adodc1.CommandType = adCmdTable
Adodc1.RecordSource = "tab1"
Adodc1.Refresh
MSHFlexGrid1.ColWidth(0) = 200
MSHFlexGrid1.ColWidth(1) = 1700
MSHFlexGrid1.ColWidth(2) = 1400
MSHFlexGrid1.ColWidth(3) = 1400
MSHFlexGrid1.ColWidth(4) = 1700
MSHFlexGrid1.ColWidth(5) = 1700
MSHFlexGrid1.ColWidth(6) = 1700
MSHFlexGrid1.ColWidth(7) = 1300
MSHFlexGrid1.RowHeight(0) = 400
End Sub

```

4.2.5. Network Information :

In this phase we will conduct two operations and two to find information on IP in detail and every individual, as follows:

1 - we will find information on the process of Attack IP of others known and if the enemy is the introduction of IP are obtained information about it in detail, as in Figure (23).

2 - We will find information on the process of Valid IP-known and friendly if it is the introduction of IP are obtained information about it in detail, as in Figure (24).

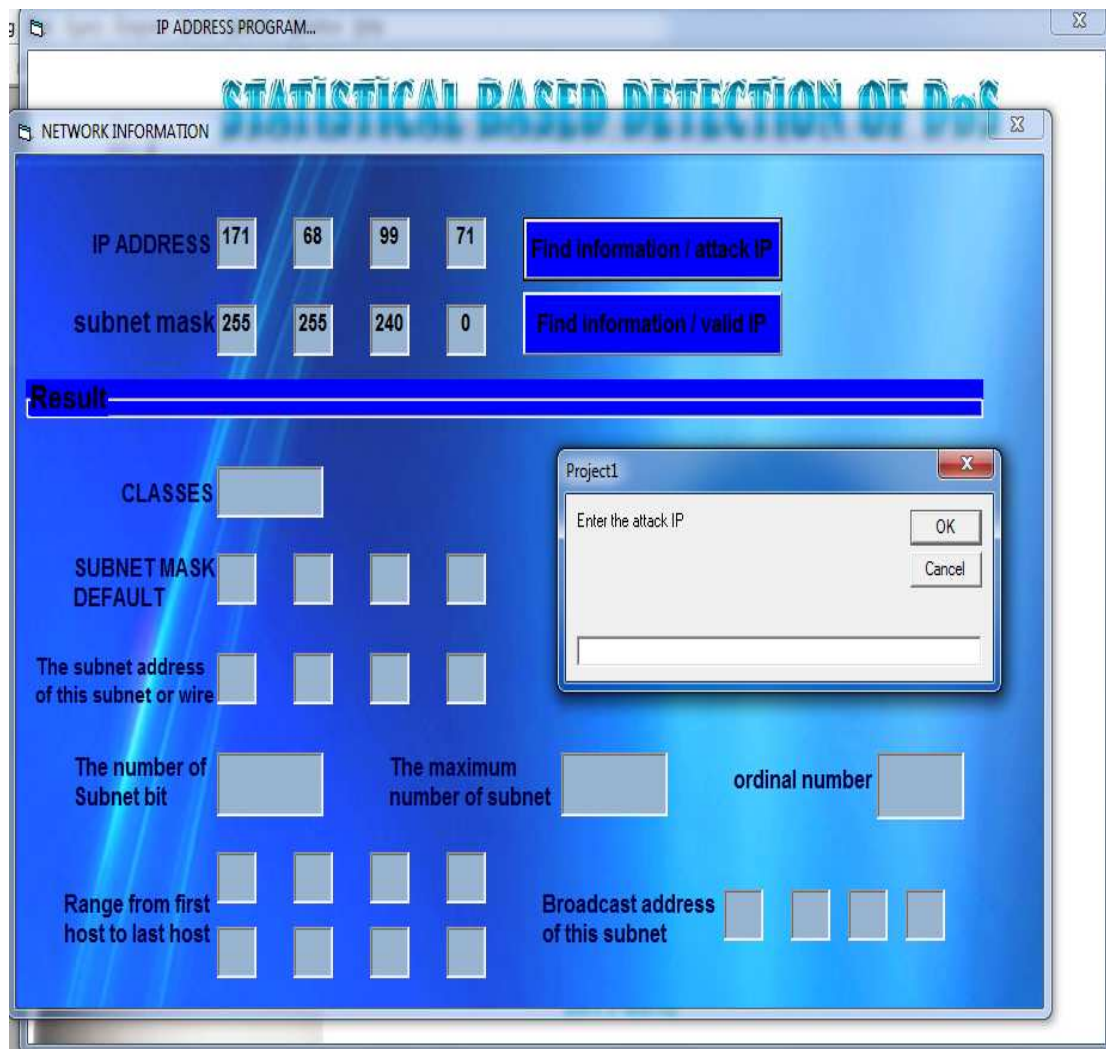


Figure 23 Network Information Attack IP

In this step, we write your code programmatic (Network Information Attack IP) and as follows :

```
Private Sub Command3_Click()
Dim x11 As String
Dim x12 As String
Dim x13 As String
x12 = "find IP"
x13 = InputBox("Enter the attack IP")
qry = "ip=" & x13 & ""
Data2.Recordset.FindFirst qry
If Data2.Recordset.NoMatch Then
MsgBox "the IP attack not found", vbOKOnly + vbCritical, " the attack IP search"
End If
' attack ip address
Dim str As String
Dim nad11() As String
Dim str1 As String
Dim nad10() As String
str = Text45.Text
nad11() = Split(str, ".")
Text1.Text = nad11(0)
Text2.Text = nad11(1)
Text3.Text = nad11(2)
Text4.Text = nad11(3)
'subnet mask
str1 = Text44.Text
nad10() = Split(str1, ".")
Text10.Text = nad10(0)
Text11.Text = nad10(1)
Text12.Text = nad10(2)
Text13.Text = nad10(3)
Dim msg As String
Dim i As Double
'Dim x As Integer, y As Integer, z As Integer
i = Text1.Text
h = Text2.Text
o = Text3.Text
c = Text4.Text
d = Text10.Text
e = Text11.Text
f = Text12.Text
g = Text13.Text
Dim no1 As Byte, no2 As Byte, no3 As Byte, no4 As Byte, no5 As Byte, no6 As Byte, no7 As Byte, no8 As Byte
If ((i >= 1 And i <= 127) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <= 255)
And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <= 255))
Then
Text9.Text = "A"
Text5.Text = 255
Text6.Text = 0
Text7.Text = 0
```

```

Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text
Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text
Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
w = 0
w = w + Val(Text14.Text) \ 256
w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
n = 0
n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128
b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64) \ 32

```

```

b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
If b = 0 And m <> 0 Then
ord = (Text20.Text * 256) \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text19.Text * (2 ^ m))
Text32.Text = ord
Elseif m = 0 And b = 0 Then
ord = Text19.Text \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text19.Text * (2 ^ (m + b)))
Text32.Text = ord
Elseif b <> 0 And m <> 0 Then
ord = Text21.Text \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text20.Text * (2 ^ m))
ord = ord + (Text19.Text * (2 ^ (m + b)))
Text32.Text = ord
End If
Text30.Text = ((2 ^ (16 - Text22.Text)) - 1) + Text20.Text
Text31.Text = 254
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Elseif ((i >= 128 And i <= 191) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <= 255) And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <= 255)) Then
Text9.Text = "B"
Text5.Text = 255
Text6.Text = 255
Text7.Text = 0
Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text
Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text

```

```

Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
'b = 0
b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128
b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
'x = b
'MsgBox x
'm = 0
m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
w = w + Val(Text14.Text) \ 256
w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
If b <> 0 And m <> 0 Then
ord = Text21.Text \ 2 ^ (16 - (b + m))
ord = ord + ((Text20.Text) * 2 ^ b)
Text32.Text = ord
Elseif b = 0 And m <> 0 Then
ord = Text20.Text \ (2 ^ (8 - (b + m)))
Text32.Text = ord
End If
xxx = (2 ^ (16 - Text22.Text)) - 2 + Text21.Text
yyy = (xxx / 255)

```

```

yy = Int(yyy)
If xxx < 255 Then
Text30.Text = Text20.Text
Text31.Text = xxx
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Else
'xxx > 255 Then
Text30.Text = yy + Text20.Text - 1
Text31.Text = 254
Text29.Text = Text19.Text
Text28.Text = Text18.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
End If
Elseif ((i >= 192 And i <= 255) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <=
255) And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <=
255)) Then
Text9.Text = "C"
Text5.Text = 255
Text6.Text = 255
Text7.Text = 255
Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text
Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text
Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
w = 0
w = w + Val(Text14.Text) \ 256
w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32

```

```

w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
n = 0
n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2

m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128
b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64) \ 32
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
'If b <> 0 Then
ord = Text21.Text \ (2 ^ (8 - (b)))
Text32.Text = ord
'End If
Text31.Text = ((2 ^ (8 - Val(Text22.Text))) - 2) + Text21.Text
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text30.Text = Text20.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Else:
msg = MsgBox("the number is error", vbCritical, "wrong")
Text9.Text = " "
Text5.Text = " "
Text6.Text = " "
Text7.Text = " "

```



```
Text8.Text = " "  
Text18.Text = " "  
Text19.Text = " "  
Text20.Text = " "  
Text21.Text = " "  
Text22.Text = " "  
Text23.Text = " "  
Text24.Text = " "  
Text25.Text = " "  
Text26.Text = " "  
Text27.Text = " "  
Text28.Text = " "  
Text29.Text = " "  
Text30.Text = " "  
Text31.Text = " "  
Text32.Text = " "  
Text33.Text = " "  
Text34.Text = " "  
Text35.Text = " "  
Text36.Text = " "  
End If  
Text47.Text = Text18.Text & "." & Text19.Text & "." & Text20.Text & "." & Text21.Text  
Text46.Text = Text5.Text & "." & Text6.Text & "." & Text7.Text & "." & Text8.Text  
Text48.Text = Text24.Text & "." & Text25.Text & "." & Text26.Text & "." & Text27.Text  
Text49.Text = Text28.Text & "." & Text29.Text & "." & Text30.Text & "." & Text31.Text  
Text50.Text = Text33.Text & "." & Text34.Text & "." & Text35.Text & "." & Text36.Text  
Text52.Text = Text9.Text  
Text54.Text = Text22.Text  
Text55.Text = Text23.Text  
Text57.Text = Text32.Text  
Data2.Refresh  
End Sub
```

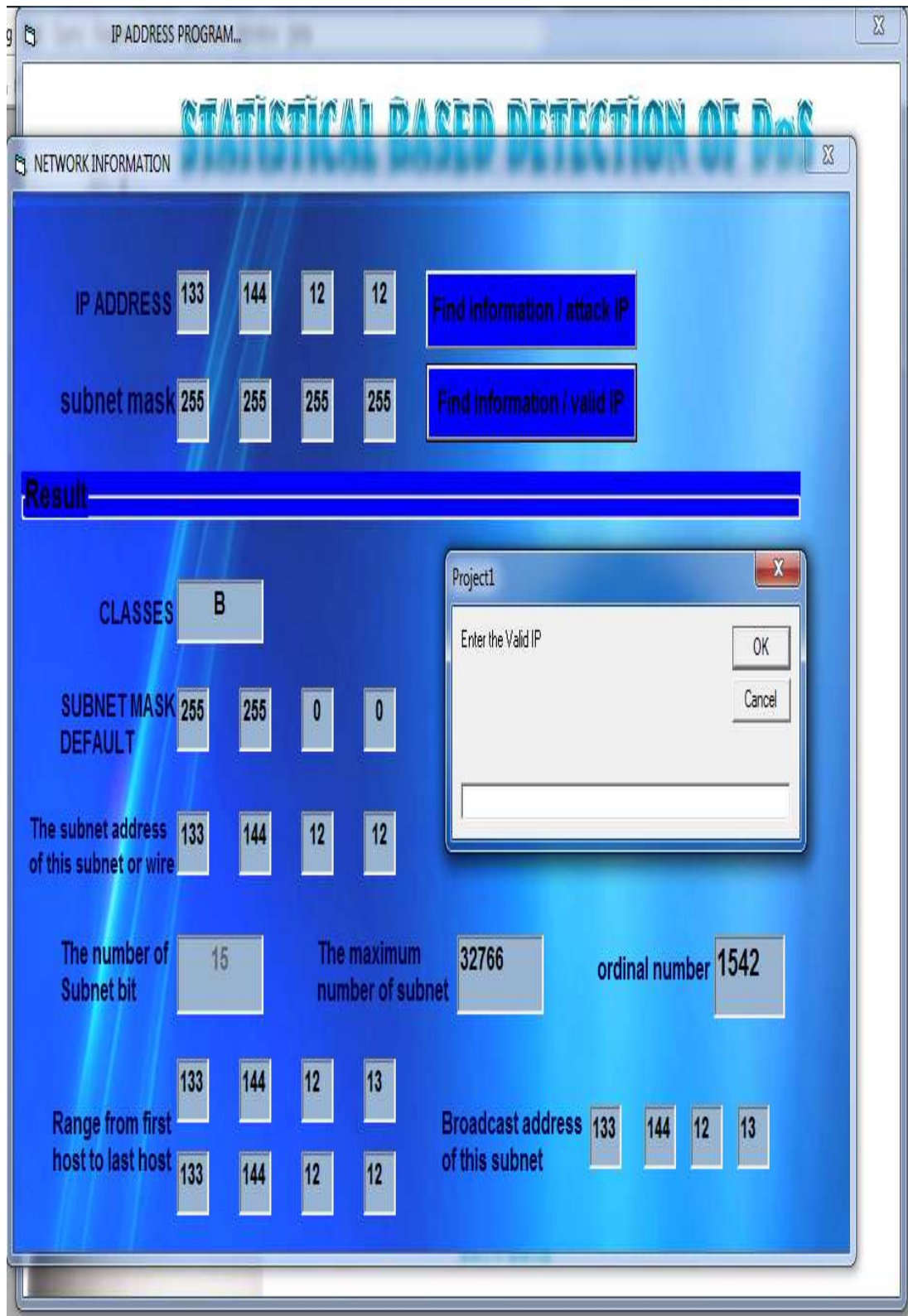


Figure 24 Network Information Valid IP

In this step, we write your code programmatic (Network Information Valid IP) and as follows :

```

:

Private Sub Command2_Click()
Dim x11 As String
Dim x12 As String
Dim x13 As String
x12 = "ÉÏË Úä ip"
x13 = InputBox("Enter the Valid IP")
qry = "ip=" & x13 & ""
Data1.Recordset.FindFirst qry
If Data1.Recordset.NoMatch Then
MsgBox "the IP attack not found", vbOKOnly + vbCritical, "the Valid IP search"
End If
' valid ip address
Dim str As String
Dim nad11() As String
Dim str1 As String
Dim nad10() As String
str = Text37.Text
nad11() = Split(str, ".")
Text1.Text = nad11(0)
Text2.Text = nad11(1)
Text3.Text = nad11(2)
Text4.Text = nad11(3)
'subnet mask
str1 = Text38.Text
nad10() = Split(str1, ".")
Text10.Text = nad10(0)
Text11.Text = nad10(1)
Text12.Text = nad10(2)
Text13.Text = nad10(3)
Dim msg As String
Dim i As Double
'Dim x As Integer, y As Integer, z As Integer
i = Text1.Text
h = Text2.Text
o = Text3.Text
c = Text4.Text
d = Text10.Text
e = Text11.Text
f = Text12.Text
g = Text13.Text
Dim no1 As Byte, no2 As Byte, no3 As Byte, no4 As Byte, no5 As Byte, no6 As Byte, no7 As Byte, no8 As Byte
If ((i >= 1 And i <= 127) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <= 255)
And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <= 255))
Then
Text9.Text = "A"
Text5.Text = 255
Text6.Text = 0

```

```

Text7.Text = 0
Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text
Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text
Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
w = 0
w = w + Val(Text14.Text) \ 256
w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
n = 0
n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2

b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128

```

```

b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64) \ 32
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
If b = 0 And m <> 0 Then
ord = (Text20.Text * 256) \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text19.Text * (2 ^ m))
Text32.Text = ord
Elseif m = 0 And b = 0 Then
ord = Text19.Text \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text19.Text * (2 ^ (m + b)))
Text32.Text = ord
Elseif b <> 0 And m <> 0 Then
ord = Text21.Text \ (2 ^ (24 - (b + m + n)))
ord = ord + (Text20.Text * (2 ^ m))
ord = ord + (Text19.Text * (2 ^ (m + b)))
Text32.Text = ord
End If
Text30.Text = ((2 ^ (16 - Text22.Text)) - 1) + Text20.Text
Text31.Text = 254
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Elseif ((i >= 128 And i <= 191) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <=
255) And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <=
255)) Then
Text9.Text = "B"
Text5.Text = 255
Text6.Text = 255
Text7.Text = 0
Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text

```

```

Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text
Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
'b = 0
b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128
b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
'x = b
'MsgBox x
'm = 0
m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2

n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
w = w + Val(Text14.Text) \ 256
w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
If b <> 0 And m <> 0 Then
ord = Text21.Text \ 2 ^ (16 - (b + m))
ord = ord + ((Text20.Text) * 2 ^ b)
Text32.Text = ord
Elseif b = 0 And m <> 0 Then
ord = Text20.Text \ (2 ^ (8 - (b + m)))
Text32.Text = ord

```

```

End If
xxx = (2 ^ (16 - Text22.Text)) - 2 + Text21.Text
yyy = (xxx / 255)
yy = Int(yyy)
If xxx < 255 Then
Text30.Text = Text20.Text
Text31.Text = xxx
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Else
'xxx > 255 Then
Text30.Text = yy + Text20.Text - 1
Text31.Text = 254
Text29.Text = Text19.Text
Text28.Text = Text18.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
End If
Elseif ((i >= 192 And i <= 255) And (h >= 0 And h <= 255) And (o >= 0 And o <= 255) And (c >= 0 And c <=
255) And (d >= 0 And d <= 255) And (e >= 0 And e <= 255) And (f >= 0 And f <= 255) And (g >= 0 And g <=
255)) Then
Text9.Text = "C"
Text5.Text = 255
Text6.Text = 255
Text7.Text = 255
Text8.Text = 0
no1 = Text10.Text
no2 = Text5.Text
no3 = Text11.Text
no4 = Text6.Text
no5 = Text12.Text
no6 = Text7.Text
no7 = Text13.Text
no8 = Text8.Text
Text14.Text = no1 Xor no2
Text15.Text = no3 Xor no4
Text16.Text = no5 Xor no6
Text17.Text = no7 Xor no8
Text18.Text = Text1.Text And Text10.Text
Text19.Text = Text2.Text And Text11.Text
Text20.Text = Text3.Text And Text12.Text
Text21.Text = Text4.Text And Text13.Text
Text24.Text = Text18.Text
Text25.Text = Text19.Text
Text26.Text = Text20.Text
Text27.Text = Text21.Text + 1
w = 0
w = w + Val(Text14.Text) \ 256

```

```

w = w + (Val(Text14.Text) Mod 256) \ 128
w = w + (Val(Text14.Text) Mod 256 Mod 128) \ 64
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64) \ 32
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
w = w + (Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
w = w + Val(Text14.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
n = 0
n = n + Val(Text15.Text) \ 256
n = n + (Val(Text15.Text) Mod 256) \ 128
n = n + (Val(Text15.Text) Mod 256 Mod 128) \ 64
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64) \ 32
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
n = n + (Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
n = n + Val(Text15.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2

m = m + Val(Text16.Text) \ 256
m = m + (Val(Text16.Text) Mod 256) \ 128
m = m + (Val(Text16.Text) Mod 256 Mod 128) \ 64
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64) \ 32
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
m = m + (Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
m = m + Val(Text16.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
b = b + Val(Text17.Text) \ 256
b = b + (Val(Text17.Text) Mod 256) \ 128
b = b + (Val(Text17.Text) Mod 256 Mod 128) \ 64
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64) \ 32
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32) \ 16
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16) \ 8
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8) \ 4
b = b + (Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4) \ 2
b = b + Val(Text17.Text) Mod 256 Mod 128 Mod 64 Mod 32 Mod 16 Mod 8 Mod 4 Mod 2
Text22.Text = n + w + m + b
Text23.Text = (2 ^ Text22.Text) - 2
'If b <> 0 Then
ord = Text21.Text \ (2 ^ (8 - (b)))
Text32.Text = ord
'End If
Text31.Text = ((2 ^ (8 - Val(Text22.Text))) - 2) + Text21.Text
Text28.Text = Text18.Text
Text29.Text = Text19.Text
Text30.Text = Text20.Text
Text33.Text = Text28.Text
Text34.Text = Text29.Text
Text35.Text = Text30.Text
Text36.Text = Text31.Text + 1
Else:
msg = MsgBox("the number is error", vbCritical, "wrong")
Text9.Text = " "

```



```
Text5.Text = " "  
Text6.Text = " "  
Text7.Text = " "  
Text8.Text = " "  
Text18.Text = " "  
Text19.Text = " "  
Text20.Text = " "  
Text21.Text = " "  
Text22.Text = " "  
Text23.Text = " "  
Text24.Text = " "  
Text25.Text = " "  
Text26.Text = " "  
Text27.Text = " "  
Text28.Text = " "  
Text29.Text = " "  
Text30.Text = " "  
Text31.Text = " "  
Text32.Text = " "  
Text33.Text = " "  
Text34.Text = " "  
Text35.Text = " "  
Text36.Text = " "  
End If  
Text39.Text = Text18.Text & "." & Text19.Text & "." & Text20.Text & "." & Text21.Text  
Text40.Text = Text5.Text & "." & Text6.Text & "." & Text7.Text & "." & Text8.Text  
Text41.Text = Text24.Text & "." & Text25.Text & "." & Text26.Text & "." & Text27.Text  
Text42.Text = Text28.Text & "." & Text29.Text & "." & Text30.Text & "." & Text31.Text  
Text43.Text = Text33.Text & "." & Text34.Text & "." & Text35.Text & "." & Text36.Text  
Text51.Text = Text9.Text  
Text53.Text = Text22.Text  
Text56.Text = Text23.Text  
Text58.Text = Text32.Text  
Data1.Refresh  
End Sub
```

CHAPTER V

5.CONCLUSIONS AND FUTURE WORK

5.1. Conclusions:

Based on the thesis which I presented on statistic bases of detecting DOS attacks by using IPS and the specific time, we classified the interfering IPS to the system into negative and positive and this was conducted through specific detection protocol. We also set up the Algorithm of intervention time to the IP in a specific time which is 4 seconds. Through this algorithm we reduced the possibilities of exposing networks to piracy attacks, in addition, through this process and in order to deprive some people of reaching any informational services, we could through this process prevent the people who use the websites of companies to abuse the regulations. We also stopped some people who are members in some websites to misuse the rules and conditions. We were able to reach information on negative or positive IPS as follows:

- Classes
- Subnet mask default
- Subnet address of this subnet or wire
- The number of subnet
- The maximum number of subnet
- Ordinal number
- Range form first host to lost host
- Broadcast address of this subnet

And identify the Number and time of interfering the IPS (hour/minute/second rate) and the exceptional statistic IP base which is required to be researched promptly in the database and diagram to discover the statistic of the attacks in form of diagram. in addition, this will be applied in the practical field so that the agents can get the information and prevent from any simple illegal penetration .

5.2. Future Work:

Through the work which I have conducted and the statistic base of negative and positive IPs and through my research on DOS, I recommend to set up DDOS statistic base and change the time of IP entry and make some improvements on the program through linking it with WIN and adding special codes for entering the system or to link between DOS and DDOS with each other via statistic base which can involve both of them and this will prevent the existence of (dual attacks) in the form of (one attack) based on previous concepts from the presented project.

REFERENCES

- [1] A Case Study: Using Architectural Features To Improve Sophisticated Denial-Of-Service Attack Detections Ran Tao¹, Li Yang², Lu Peng¹, Bin Li³, Alma Cemerlic² ¹Department Of Electrical And Computer Engineering, Louisiana State University.
- [2] ADoS/DDoS Attack Detection System Using Chi-Square Statistic Approach*Fang-Yieleu*Department Of Computer Science, Tunghai University R.K.C. Chang .
- [3] CCNA Cisco Certified Network Associate Study Guide Sixth Edition Todd Lammle Wiley Publishing, Inc 2007.
- [4] Deciphering Detection Techniques: Part III Denial Of Service Detection By Dr. Fengmin Gong, Chief Scientist, Mcafee Network Security Technologies Group January 2003.
- [5] Denial of Service Attack [Neeharika Buddha Graduate Student, University Of Kansas]2009.
- [6] Detection Of Resource-Drained Attacks On SIP-Based Wireless Voip Networks Jin Tang, Yong Hao, Yu Cheng And Chi Zhou Department Of Electrical And Computer Engineering Illinois Institute Of Technology, Chicago, IL, USA 60616.
- [7] Detecting Service Violations And DoS Attacks Ahsan Habib, Mohamed M. Hefeeda, And Bharat K. Bhargava CERIAS And Department Of Computer Sciences Purdue University, West Lafayette, IN 47907 Fhabib, Mhefeeda, Bbg@Cs.Purdue.Edu
- [8] Didem Çakırtaş DETECTING DENIAL OF SERVICE ATTACKS IN NETWORK TRAFFIC WITH MAXIMUM ENTROPY AND HYPOTHESIS TESTING TECHNIQUES Master of Science Istanbul Technical University, 2004
- [9] Denial of service attacks, CERT. http://cert.org/tech_tips/denial_of_service.html .
- [10] Distributed Denial Of Service Attacks/ Princeton University Electrical Engineering Department/September 23, 2002/Prepared For: Prof. Ruby Lee ELE 572

- [11] DoS/DDoS Detection Scheme Using Statistical Method Based On The Destination Port Number Shunsuke Oshima Graduate School Of Science And Technology Kumamoto University
- [12] [Http://Ar.Wikipedia.Org/Wiki/DoS](http://Ar.Wikipedia.Org/Wiki/DoS).
- [13] [Http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack](http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack) Denial-Of-Service Attack - Wikipedia, The Free Encyclopedia .
- [14] Karig, David And Ruby Lee. Remote Denial Of Service Attacks And Countermeasures, Princeton University Department Of Electrical Engineering Technical Report CE-L2001-002, October 2001.
- [15] Kelly, C., D. Spears, C. Karlsson, P. Polyakov, An Ensemble Of Anomaly Classifiers For Identifying Cyber Attacks, [Http://Www.Cs.Uwyo.Edu/~Dspears/Papers/Ensemble](http://Www.Cs.Uwyo.Edu/~Dspears/Papers/Ensemble) . Pdf, 2005.
- [16] Kendall, K., A Database Of Computer Attacks For The Evaluation Of Intrusion Detection Systems, M.S. Thesis, Massachusetts Institute Of Technology, 1999.
- [17] Lightweight Detection Of DoS Attacks Sirikarn Pukkawanna*, Vasaka Visoottiviseth*, Panita Pongpaibool† *Department Of Computer Science, Mahidol University, Rama 6 Rd., Bangkok 10400, THAILAND
- [18]- On The Detection Of Signaling DoS Attacks On 3G Wireless Networks Patrick P. C. Lee, Tian Bu, And Thomas Woo.
- [19] Patrikakis C., M. Masikos, O. Zouraraki, “Distributed Denial Of Service Attacks”, Internet Protocol Journal, Vol. 7 Issue 4, Pp.13-35, 2004.
- [20] “Ping Of Death Attack”, [Http://En.Wikipedia.Org/Wiki/Ping_Of_Death](http://En.Wikipedia.Org/Wiki/Ping_Of_Death) .
- [21] Teardrop Attack, [Http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack#Teardrop_Attack](http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack#Teardrop_Attack).
- [22] "Unintentional Denial Of Service" [Http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack#Denial-Of-Service_Level_II](http://En.Wikipedia.Org/Wiki/Denial-Of-Service_Attack#Denial-Of-Service_Level_II)
- [23] U.S. Department Of Energy, “G-48: TCP SYN Flooding And IP Spoofing Attacks”, Computer Incident Advisory Capability (CIAC) Information Bulletin, September 20, 1996.
- [24] Using Frequency Domain Analysis Fazirulhisyam Hashim The University Of Sydney .

APPENDIX

PERSONAL INFORMATION

Surname, Name: Tariq .Abed. Mohamad

Nationality: IRAQ (IR)

Date and Place of Birth: 17 /05 /1983 , IRAQ / KIRKUK

Marital Status: Married

Phone: +90 539 24 87 390

Fax :+90 534 720 85 58

Email: tariq.atar@gmail.com

EDUCATION

Degree	Institution	Year Of Graduation
MS	Cankaya Univ . Mathematics Computer Seince	2012
Bs	Kirkuk Univ.Computer Seince	2007
High School	Al.Hakema -Kirkuk	2003

WORK EXPERIENCE

YEAR	PLACE	Enrollment
2003	Salemana center for internet and communication	Networking
2005	Turkish company in the Iraq	Supervisor
2007	Anbal Al Arabya Bank	Chief programmer
2009	North Oil Company -Kirkuk	Programmer

Foreign languages : Arabic , English Turkish , Kurdish .