JYOTHISHMATI INSTITUTE OF TECHNOLOGY AND SCIENCE
NUSTULAPUR, KARIMNAGAR, AFFLIATED TO JNTU.

INFORMATION SECUTRITY
LAB MANUAL

Program- 1 :

Aim: examine how sniffing software works

1. Go to the site ethereal.com
2. Download the tool
3. In that environment click on option capture ⇨ interfaces ⇨ capture, this opens a Window wait for few seconds, it will give the details of all packets captured. (Or) right click on open in a new window. It gives all the data related to the packet.
4. Click edit/apply design window. This is a filter.
5. Go to statistics, click on destination. It will give all the destination paths.
6. Statistics ⇨ IP addresses.
7. Like this explore all the options.

Program-2:

Aim: examine how PGP works

1. Go to site GnuPGP.org this will run on windows as well as Linux.
2. Get the tool, download it.
3. In that go to Documentation, in that click on Guides, in that many are there but we used Brendan's page, click on that
4. List of commands are given
5. Execute it.

Program-3:

Aim: write a program to implement format string vulnerabilities

Format String Program :

```
# include <stdio.h>

# include <conio.h>

void main()

{

      int input =2;

   clrscr ();
       printf("ABCdef \n\n%n",&input);

     printf("value of var :%d",input);

  getch();

}
```

output vs output :

Expected output :

ABCdef

Value of var : 2

Real output :

ABCdef

Value of var : 8

Program-4:

Aim: examine how NMAP software works

1. Go to the site Insecure.org
2. Download the tool for NMAP
3. This should be executed using internet .But if all the systems are having Internet, then the result might be unpredictable.So only one system is taken and for Demonstrating this.

Using NMAP for Port monitoring:

What is NMAP?

Nmap: - "Network Mapper"

It was developed by Fyodor

- An open source tool for network exploration and security auditing.
- N-map uses raw IP packets in novel ways for information gathering

N-MAP features

- **Ping Sweeping**:- Identifying computers on a network.
- **Port Scanning**:- Enumerating the open Ports on one or more target computers.
- **OS Detection** :- Remotely determining the operating system and some hardware characteristics Of network devices.

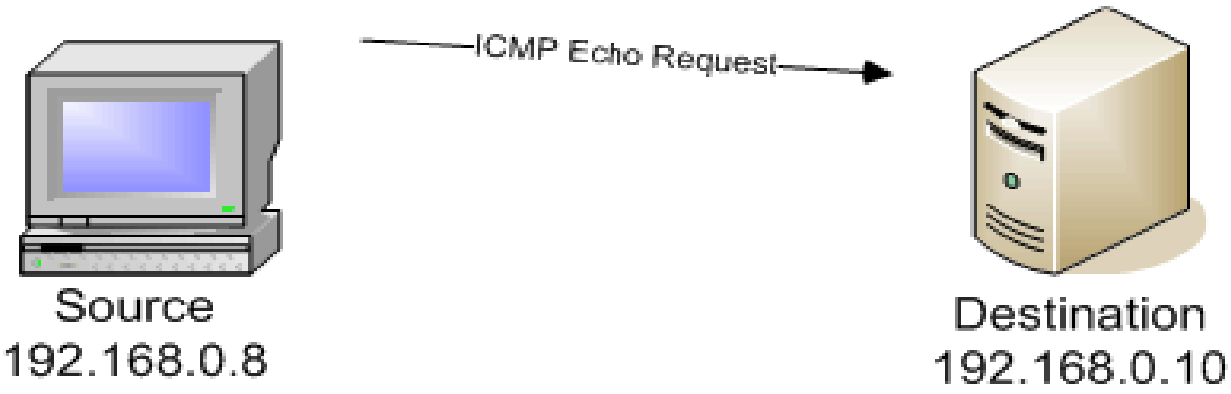
Ping Sweeping

- Ping Sweeping allows the hackers to automatically map out the entire target network and pinpoint all alive systems within a particular range of IP addresses.

Operations On PING Sweeping:

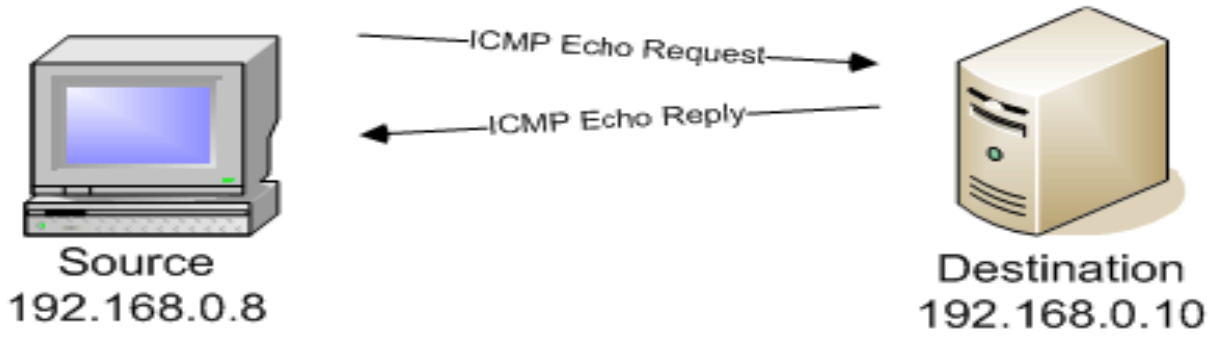
When Host is not Alive

If the station isn't available on the network or a packet filter is preventing ICMP packets from passing, there will be no response to the echo frame.



When Host is Alive

A response from an active host will return an ICMP echo reply, unless the IP address is not available on the network or ICMP is filtered



Nmap command line prompt:

- **C:\>nmap –sP 192.168.0.10**

Ex: c:\nmap> nmap -sP www.sreechaitanya.org

Output

Host 8.95.5646.static.theplanet.com (70.86.149.8) appears to be up.

 # Nmap run completed at Sat Nov 24 10:43:34 2007 -- 1 IP address (1 host up) scanned in 1.719 seconds.

When to use the Ping Sweeping

- When building an inventory of available stations on a network.
- When more detailed information is not needed
- If needed a different scan type may be a better choice.

Port Scanning

- A port scan is like ringing the doorbell to see whether someone's at home.
- A port scanner is a piece of software designed to search a network host for open ports.
- This is often used by administrators to check the security of their networks and by hackers to compromise it.

Port Scan – Port Numbers
The Port Numbers are divided into Three Ranges:

- Well Known Ports (0 - 1023)
- Registered Ports (1024 - 49151)
- Dynamic and/or Private Ports (49152 - 65535)

Port Scanning Basic Techniques:

- The simplest port scan tries each of the ports from 0 to 65535 on the victim to see which ones are open.

Scans available through nmap

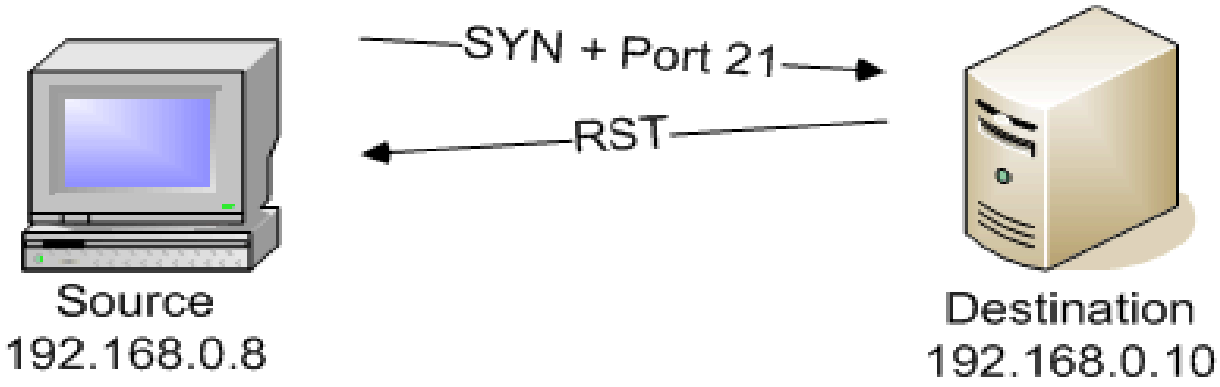
Scan Type	Switch	Description
TCP connect() scan	**-sT**	**Opens a connection to every potentially interesting port on the target machine.**
TCP SYN scan	**-sS**	**This is a "half-open" scan.**
TCP FIN	**-sF**	**This scan attempts to pass through packet filters by sending a TCP FIN packet.**
Xmas Tree	**-sX**	**Sends a packet with FIN, URG and push flags set**
Null	**-sN**	**Sends a packet without any flags turned on.**
Scan Type	**Switch**	**Description**
ACK scan	**-sA**	**An ACK packet with random acknowledgment and sequence numbers is sent.**
UDP scan	**-sU**	**This sends 0 byte UDP packets to each port on the target machine(s).**
List scan	**-sL**	**Simply lists targets to scan.**

TCP connect() scan (-sT)

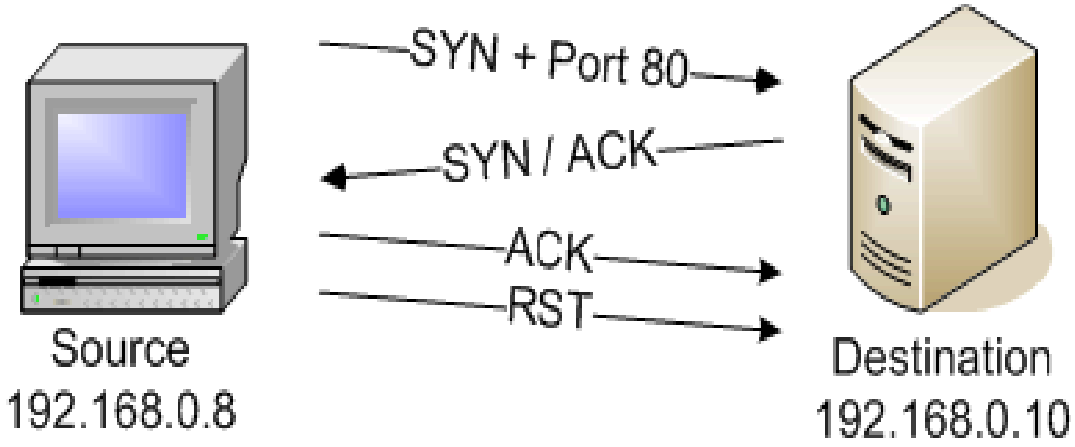
- This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect () will succeed, otherwise the port isn't reachable.

Operations On TCP Connect() Scan :

When port is closed



When port is open



Nmap command line prompt for TCP Connect ( ):

Ex: c:\nmap> **nmap -sT www.hackingmobilephones.com**

Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1689 filtered ports
PORT STATE SERVICE
21/tcp open ftp
25/tcp closed SMTP

80/tcp open http
587/tcp closed submission
3306/tcp open mysql
6666/tcp closed irc-serv
6667/tcp closed irc
6668/tcp closed irc
Nmap run completed at Sat Nov 24 10:49:30 2007 -- 1 IP address (1 host up) scanned in 130.266 seconds

Pros and Cons

Pros:

- Fast
- Easy to implement
- Accurate

Cons:

- Easily detected
- Traceable

When to use the TCP connect () Scan

If privileged access isn't available and determination of open TCP ports is absolutely necessary, however, this scan may be the only method available

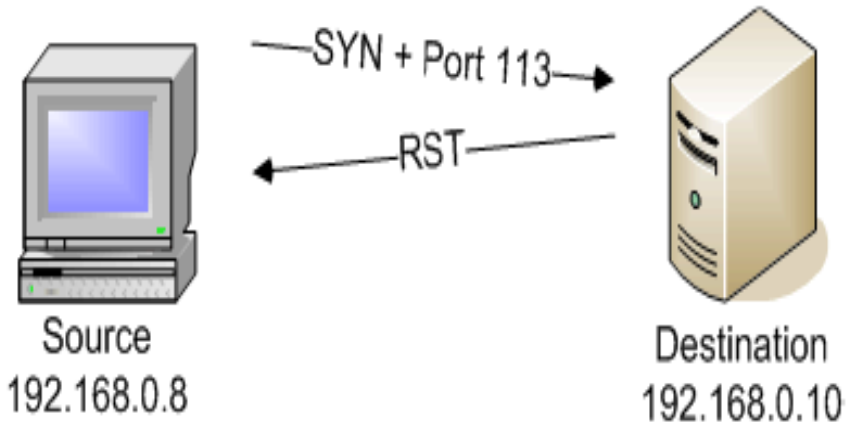
TCP SYN Scan (-sS)

- This technique is often referred to as "half-open" scanning, because we don't open a full TCP connection.
- we send a SYN packet, as if you are going to open a real connection and wait for a response.
- A SYN/ACK indicates the port is listening.
- A RST is indicative of a non- listener.
- If a SYN|ACK is received, we immediately send a RST to tear down the connection

Operations On TCP SYN Scan

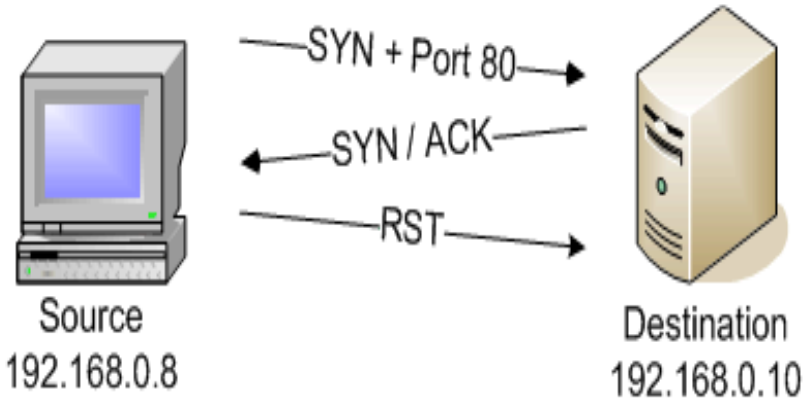
When port is Closed

- closed port responses to the TCP SYN frame with a RST frame from the destination station



When port is open

- If nmap receives an acknowledgment to a SYN request, then the port is open. Nmap then sends an RST to reset the session, and the handshake is never completed.



Nmap command line prompt for TCP SYN Scan:

Ex: c:\nmap> nmap -sS www.hackingmobilephones.com
Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1673 filtered ports
PORT STATE SERVICE
7/tcp closed echo
13/tcp closed daytime
21/tcp open ftp
25/tcp open smtp
53/tcp closed domain
80/tcp open http
110/tcp closed pop3
113/tcp closed auth
123/tcp closed ntp
143/tcp closed imap
366/tcp closed odmr
433/tcp closed nnsp
443/tcp closed https
610/tcp closed npmp-local
626/tcp closed unknown
631/tcp closed ipp
1433/tcp closed ms-sql-s
1434/tcp closed ms-sql-m
3306/tcp open mysql
5900/tcp closed vnc
6000/tcp closed X11
6001/tcp closed X11:1
6002/tcp closed X11:2
7938/tcp closed lgtomapper
Nmap run completed at Sat Nov 24 11:21:58 2007 -- 1 IP address (1 host up) scanned in 242.562 seconds

Pros and Cons
Pros:

- Fast
- Accurate
- Fairly easy to implement
- Harder to trace than the TCP connect port-scan method

Cons:

- Not totally stealthy
- Easily blocked

When to use the TCP SYN Scan

- The SYN scan is a common scan when looking for open ports on a remote device.
- Its simple SYN methodology works on all operating systems.
- As only half-opens the TCP connections, it's considered a very 'clean' scan type

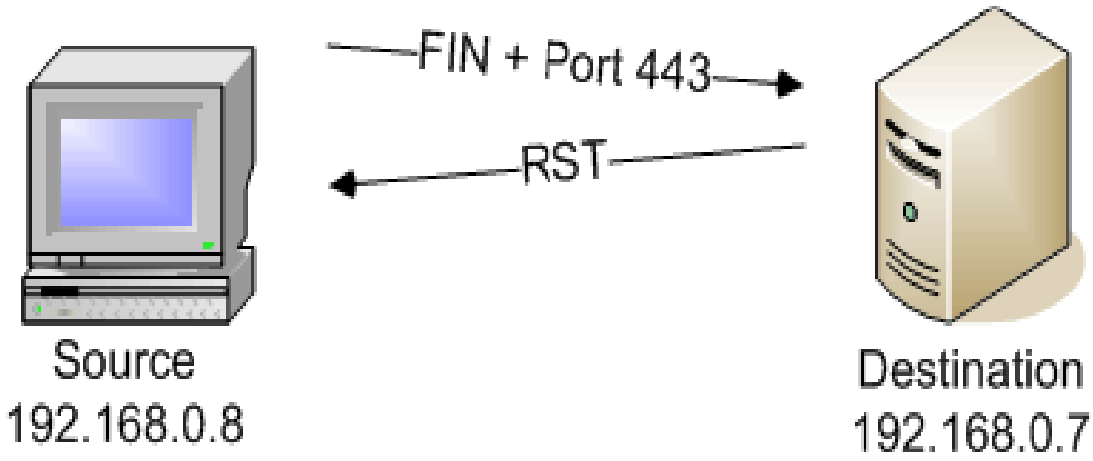
FIN Scan (–sF):

- Some firewalls and packet filters watch for SYNs to restricted ports, and programs like synlogger and Courtney are available to detect these scans. FIN packets, on the other hand, may be able to pass through unmolested.

Operations on TCP FIN Scan

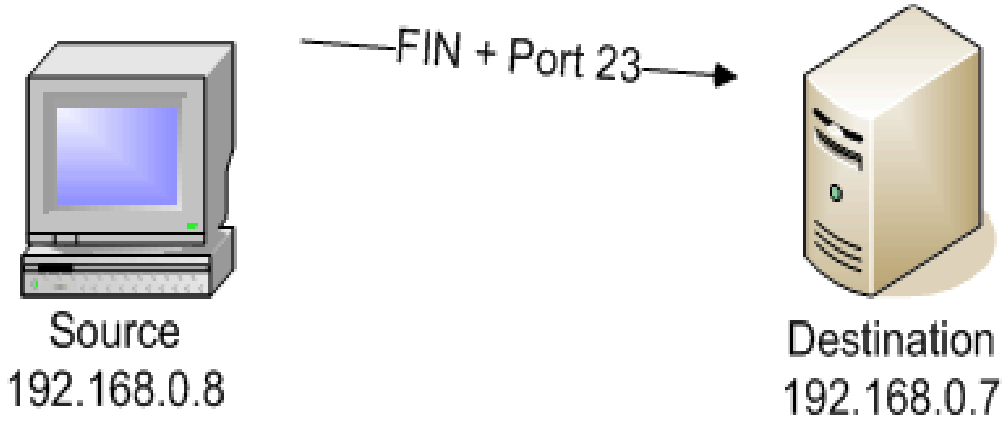
When Port is closed

- In this FIN scan, TCP port 443 is closed so the remote station sends a RST frame response to the FIN packet:



When Port is Open

- If a port is open on a remote device, no response is received to the FIN scan:



Nmap command line prompt for TCP FIN
C:\nmap> **nmap -sF www.hackingmobilephones.com**

OUTPUT:
All 1697 scanned ports on corp2.net4india.com (202.71.129.91) are open/filtered

Nmap run completed at Sat Nov 24 11:27:13 2007 -- 1 IP address (1 host up) scanned in 161.110 seconds

Pros and Cons

Pros:
- Fairly fast
- Easy to implement
- Stealthy to a certain extent

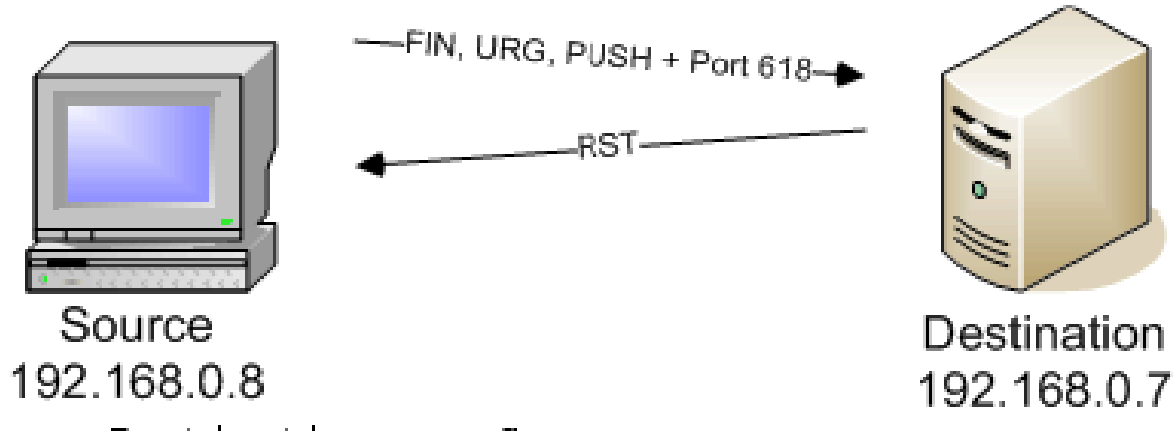
Cons:
- Inaccurate with certain operating systems

The Xmas Tree Scan (-sX) and Null Scan(-sN):

- **XMAS scans** where all flags in the TCP packet are set.
- **NULL scans** where none of the bits are set.
- However, different operating systems respond differently to these scans, and it becomes important to identify the OS and even its version and patch level

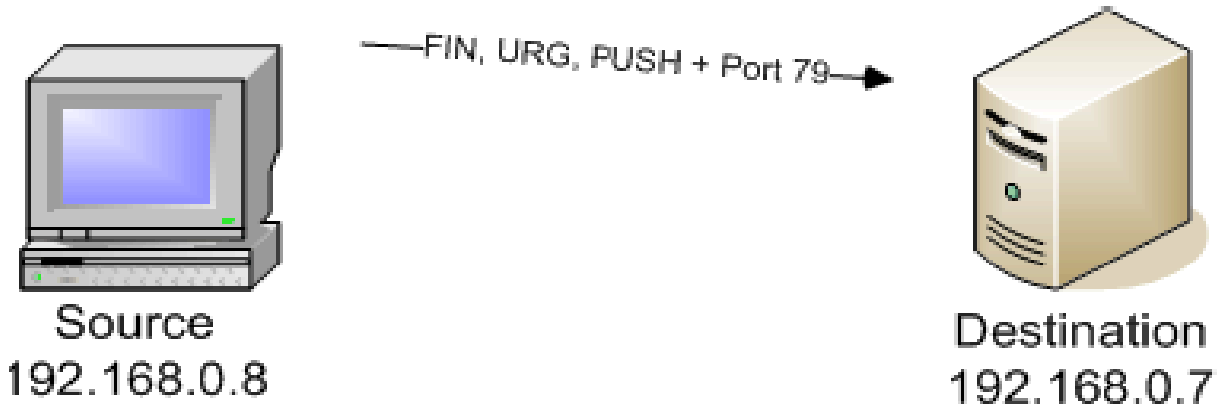
OPERATIONS ON TCP XMAS Scan

When Port is Close
A closed port responds to a Xmas tree scan with a RST:



When Port is Open

Similar to the FIN scan, an open port on a remote station is conspicuous by its silence:



Nmap command line prompt for Xmas Tree Scan

Ex: c:\nmap> **nmap -sX www.hackingmobilephones.com**

OUTPUT

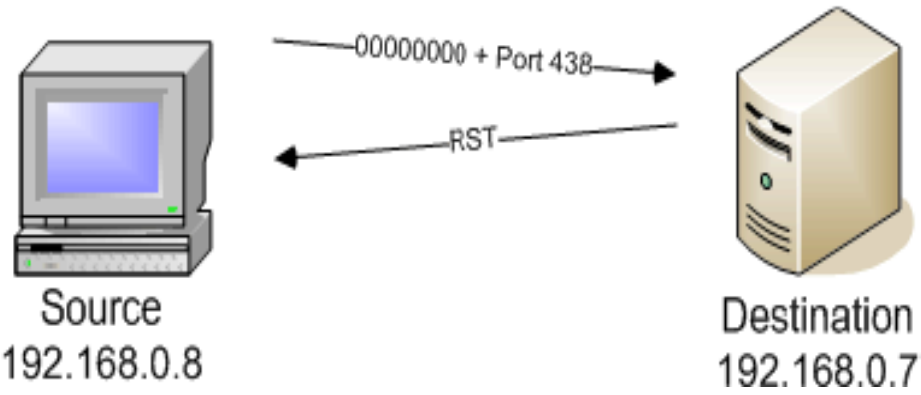
All 1697 scanned ports on corp2.net4india.com (202.71.129.91) are open|filtered.

Nmap run completed at Sat Nov 24 11:37:10 2007 -- 1 IP address (1 host up) scanned in 193.062 seconds

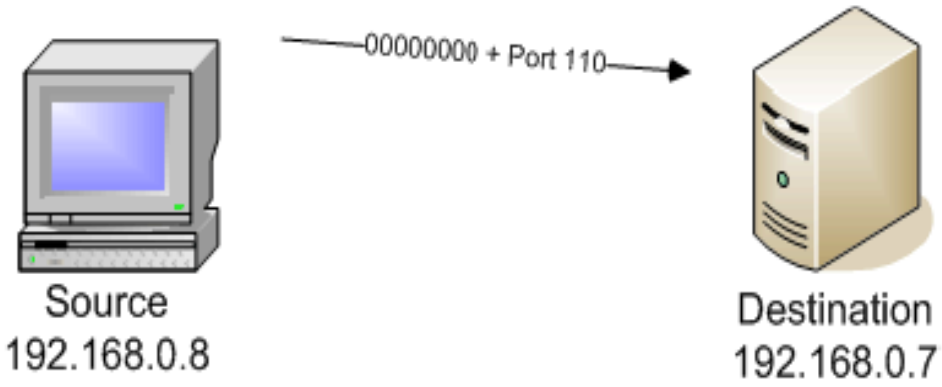
Operations on TCP NULL Scan

When Port is closed

If the port is closed, a RST frame should be returned:



When Port is Open
As expected, the response of a null scan to an open port results in no response



Nmap command line prompt for Null Scan

c:\nmap> nmap -sN www.hackingmobilephones.com

OUTPUT
All 1697 scanned ports on corp2.net4india.com (202.71.129.91) are open filtered

Nmap run completed at Sat Nov 24 11:38:02 2007 -- 1 IP address (1 host up) scanned in 193.062 seconds

Pros and Cons
Pros:

- Fairly fast
- Easy to implement
- Stealthy to a certain extent

Cons:

- Works only with UNIX and some other operating systems

When to use FIN, Xmas Tree, and Null Scan

- Although TCP SYN scans are relatively subtle, the FIN, Xmas tree, and null scans are even more invisible on the network.
- They don't show up in application log files.
- They take little network bandwidth.

- They provide extensive port information on non-Windows based systems.

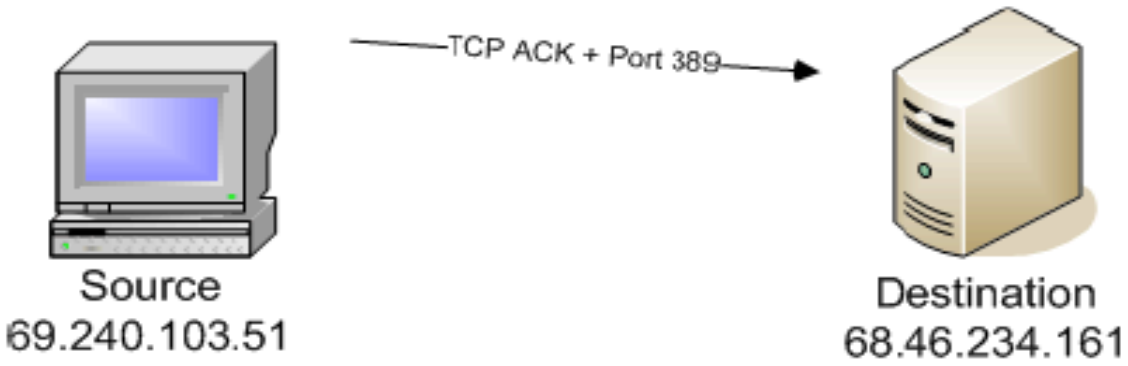
ACK Scan (-SA)

- Nmap's unique ACK scan will never locate an open port.
- The ACK scan only provides a "filtered" or "unfiltered" disposition because it never connects to an application to confirm an "open" state.
- At face value this appears to be rather limiting, but in reality the ACK scan can characterize the ability of a packet to traverse firewalls or packet filtered links

OPERATIONS ON TCP ACK Scan

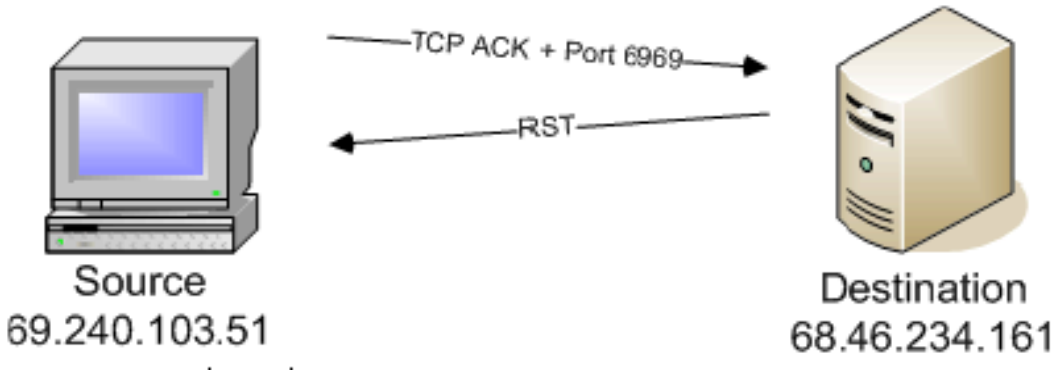
When Port is Filtered

An ACK scan operates by sending a TCP ACK frame to a remote port. If there are no responses or an ICMP destination unreachable message is returned, then the port is considered to be "filtered:“



When Port is Unfiltered

If the remote port returns an RST packet, the connection between nmap and the remote device is categorized as unfiltered:



Nmap command line prompt for ACK Scan
c:\nmap> **nmap -sA www.hackingmobilephones.com**

OUTPUT
All 1697 scanned ports on 202.71.129.91 are filtered

Nmap run completed at Sat Nov 24 11:50:59 2007 -- 1 IP address (1 host up) scanned in 232.656 seconds

Pros and Cons

Pros:

- Since the ACK scan doesn't open any application sessions, the conversation between nmap and the remote device is relatively simple.
- This scan of a single port is unobtrusive and almost invisible when combined with the other network traffic.

Cons:

- The ACK scan's simplicity is also its largest disadvantage. Because it never tries to connect to a remote device, it can never definitively identify an open port.

When to use the ACK Scan

- Although the ACK scan doesn't identify open ports, it does a masterful job of identifying ports that are filtered through a firewall.

- This list of filtered and unfiltered port numbers is useful as reconnaissance for a more detailed scan that focuses on specific port numbers.

UDP Scan (-sU)

- In order to find UDP ports, the attacker generally sends empty UDP datagrams.

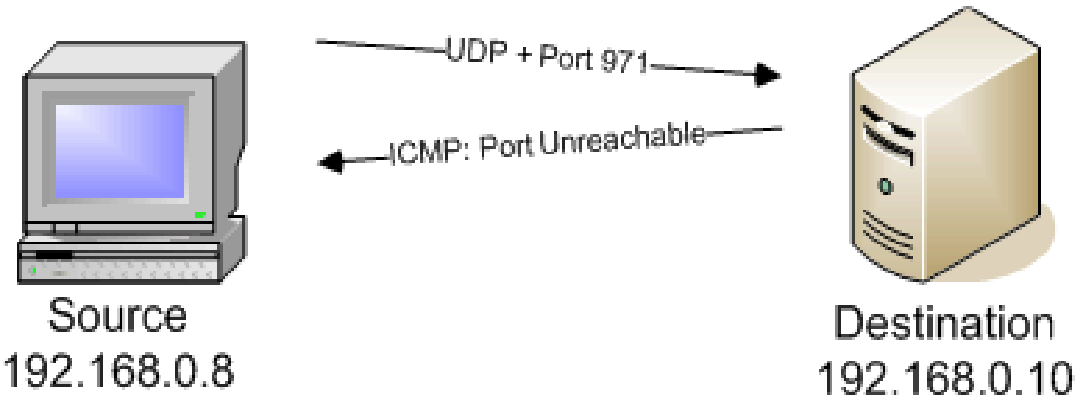
- If the port is listening, the service should send back an error message or ignore the incoming datagram.

- If the port is closed, then most operating systems send back an "ICMP Port Unreachable" message.

Operations On TCP UDP Scan

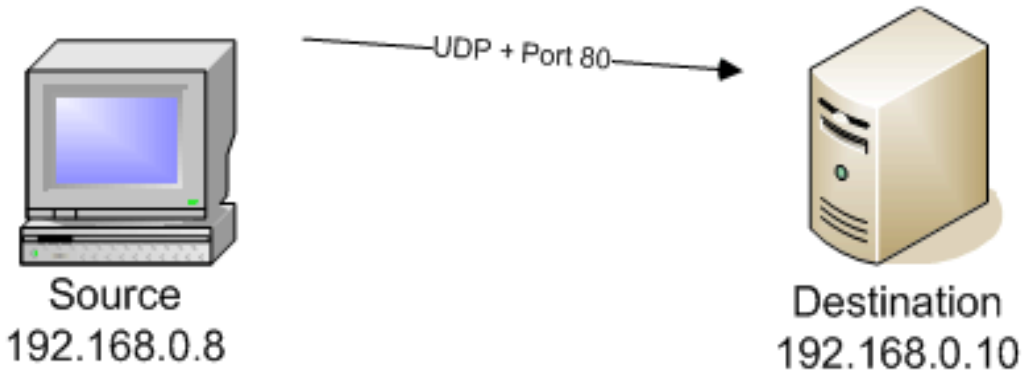
When Port is closed

A station that responds with an ICMP port unreachable is clearly advertising a closed port:



When Port is Open

A station that doesn't respond to the UDP scan is considered to be open / filtered:



Pros and Cons

Pros:

- Since there's no overhead of a TCP handshake, the UDP scan is inherently less "chatty" once it finds an open port.
- operates very efficiently on Windows-based devices.

Cons:

- The UDP scan requires privileged access.

When to use the UDP Scan

- Because of the huge amount of TCP traffic on most networks, the usefulness of the UDP scan is often incorrectly discounted.
- There are numerous examples of open UDP ports caused by spy ware applications, Trojan horses, and other malicious software.
- The UDP scan will locate these open ports and provide the security manager with valuable information that can be used to identify and contain these infestations.

List Scan (-sl)

- The list scan isn't really a scan, but it does provide nmap with some troubleshooting and testing capabilities.

- The list scan simply lists the IP addresses that would normally be actively scanned.

List Scan Operation

- The list scan doesn't ping the host names, and it doesn't send a TCP ACK to the default port number.

- The list scan output shows this lack of activity by identifying the IP address as "not scanned:"

Nmap command line prompt for List Scan

c:\nmap> **nmap -sl 202.71.129.0-10**

OUTPUT

Host 202.71.129.0 not scanned
Host 202.71.129.1 not scanned
Host 202.71.129.2 not scanned
Host 202.71.129.3 not scanned
Host 202.71.129.4 not scanned
Host 202.71.129.5 not scanned
Host 202.71.129.6 not scanned
Host 202.71.129.7 not scanned
Host 202.71.129.8 not scanned
Host 202.71.129.9 not scanned
Host 202.71.129.10 not scanned
Nmap run completed at Sat Nov 24 11:26:25 2007 -- 11 IP addresses (0 hosts up) scanned in 6.687 seconds

Pros and Cons

Pros:

- Good method to sanity-check a complex nmap scan prior to starting a large batch process or a large group of IP addresses.
- If any of the IP addresses are defined incorrectly on the command line or the option file, the list scan will identify the errors.
- These problems can be identified and repaired prior to running any "live" scans.

Cons:

- The list scan isn't really an active scan. It doesn't show availability, it doesn't find any ports, and it doesn't directly connect with an end device.

Some more useful options are shown in the following table

Option	Explanation
-P0	**Tells nmap not to ping hosts before scanning. (This is used to scan hosts that sit behind packet filters that don't allow ICMP traffic.)**
-f	**Causes nmap to fragment its scanning packets, making it more difficult to block the scan with packet filters.**
-v	**Puts nmap into verbose mode, causing it to display much more information about what it's doing.**
-oN <logfile>	**Writes output into a human readable logfile.**
-oM <logfile>	**Writes output into a machine-parsable logfile.**
--resume <logfile&;	**Resumes an incomplete scan from a logfile.**
-iL <logfile>	**Causes nmap to read input from a logfile instead of really scanning a host.**
-g <portnumber>	**Allows you to define the port nmap uses as its source port.**
-p <port range>	**Allows you to define the range of ports nmap will scan.**

Some more Examples

Ex:range for sp
C:\nmap> **nmap -oN ping3.txt -sP 202.71.129.0-5**

OUTPUT
Host 202.71.129.1 appears to be up.
Nmap run completed at Sat Nov 24 11:29:23 2007 -- 11 IP addresses (1 host up) scanned in 1.281 seconds

Ex: range for port numbers
C:\nmap> **nmap -oN tcp3.txt -sT -p 1-1023 www.hackingmobilephones.com**

OUTPUT
Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1019 filtered ports
PORT STATE SERVICE
21/tcp open ftp
25/tcp closed smtp
80/tcp open http
587/tcp closed submission
Nmap run completed at Sat Nov 24 11:14:56 2007 -- 1 IP address (1 host up) scanned in 95.750 seconds

Operating System Detection

- Allows the user to use TCP/IP fingerprinting to determine the operating system of the remote host.

Nmap command line prompt for OS Detection

C:\nmap> nmap -O www.hackingmobilephones.com
Ex: c:\nmap> nmap -oN op.txt -O www.hackingmobilephones.com
Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1673 filtered ports
PORT STATE SERVICE
7/tcp closed echo
13/tcp closed daytime
20/tcp closed ftp-data
21/tcp open ftp
25/tcp open smtp
53/tcp closed domain
80/tcp open http
110/tcp closed pop3
113/tcp closed auth
123/tcp closed ntp
199/tcp closed smux
366/tcp closed odmr
433/tcp closed nnsp
443/tcp closed https
626/tcp closed unknown
631/tcp closed ipp
1433/tcp closed ms-sql-s
1434/tcp closed ms-sql-m
5800/tcp closed vnc-http
5900/tcp closed vnc
6000/tcp closed X11
6001/tcp closed X11:1
6002/tcp closed X11:2
7938/tcp closed lgtomapper
Device type: general purpose
Running: OpenBSD 3.X
OS details: OpenBSD 3.6 x86 with pf "scrub in all"
Uptime: 28.341 days (since Sat Oct 27 02:20:14 2007)
OS detection performed. Please report any incorrect results at http://insecure.org/nmap/submit/ .
Nmap run completed at Sat Nov 24 10:31:00 2007 -- 1 IP address (1 host up) scanned in 264.500 seconds

Conclusion

Is nmap Good or Evil?

The bad guys are already using nmap for reconnaissance, because a single scan can tell you a lot about the open doors and windows in a computer's house. What the bad guys do once they have this information is why they are called the "bad guys."

Program 5
Aim: write a program to implement virus program

```
//*************************************************************//
            //virus program-created
//*************************************************************//

#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<stdio.h>
#include<process.h>
#include<graphics.h>
#include<fstream.h>

void fool();
void main()
{
clrscr();
for(int i=0;i<=100;i++)
   {

          textcolor(YELLOW+BLINK);
          gotoxy(35,12);
          cprintf("VIRUS LOADING");
          gotoxy(39,15);
          textcolor(GREEN);
          cout<<i<<"%";
          delay(75);
          clrscr();
          }
delay(100);
clrscr();
flushall();
gotoxy(20,12);
cout<<" 'AISHWARYA' VIRUS CREATED NOW BY SANDEEP";
gotoxy(20,14);
cout<<"SAY GOOD BYE TO YOUR PC IN ";
for(int j=10;j>=0;j--)
{
gotoxy(48,14);
cout<<j<<" SECONDS";
delay(1000);
}
clrscr();
cout<<"\n1.HARD-DISK CORRUPTION: ";
delay(4000);
cout<<"completed";
cout<<"\n\n2.MOTHER BOARD CORRUPTION: ";
delay(4000);
cout<<"completed";
cout<<"\n\n3.INSTALLING CYBERBOB.DLL -->WINDOWS/COMMAND :";
delay(4000);
cout<<"completed";
cout<<"\n\nPROCRAETORIAN.SYS SUCCESSFULLY PLANTED";
delay(3000);
```

```
cout<<"\n\n\tVIRUS.EXE";
delay(2000);
cout<<"\n\n*************************";
cout<<"\n\nBuddy it's a simply joke ";
cout<<"\n\n*************************";
delay(4000);
cout<<"\n\n***********************************";
cout<<"\n\nFor Real Virus ";
cout<<"\n\nContact Me: Sandeep Udaipur ";
cout<<"\n\nMo: 010101010101 ";
cout<<"\n\nEmail: sandeep@yahoo.co.in ";
cout<<"\n\n***********************************";
delay(10000);
}

void fool()
{
        clrscr();
        int g=DETECT,h;
        initgraph(&g,&h,"c:\tc\bgi");
        cleardevice();
        delay(1000);
        setcolor(2);
        settextstyle(1,0,1);
        delay(1000);
        setbkcolor(BLUE);
        getch();
        delay(4000);
        closegraph();
        exit(0);
}
```

OUTPUT:

1.HARD-DISK CORRUPTION: completed

2.MOTHER BOARD CORRUPTION: completed

3.INSTALLING CYBERBOB.DLL -->WINDOWS/COMMAND :completed

PROCRAETORIAN.SYS SUCCESSFULLY PLANTED

VIRUS.EXE

****

Buddy it's a simply joke

****

Program 6
Aim: write a program to implement cryptography program

```
/*cryptography program
    Example
  "6.6 As before, suppose p = 467, alpha = 4,
  a = 101 and beta = 449. Suppose the message
  x = 286 is signed with the (bogus) signature
  y = 83, and Bob wants to convince Alice that
  the signature is invalid."
  -Douglas R. Stinson-
  See "Cryptogrphy: Theory and Practice" by
  Douglas R. Stinson pages 222-223.
*/

#include <stdio.h>
#include<conio.h>
long exp_mod(long x, long b, long n)
/* returns x ^ b mod n */
{
  long a = 1l, s = x;

  while (b != 0) {
            if (b & 1l) a = (a * s) % n;
            b >>= 1;
            if (b != 0) s = (s * s) % n;
  }
  if (a < 0) a += n;
  return a;
}

long Extended_Euclidean(long b, long n)
{
  long b0 = b, n0 = n, t = 1, t0 = 0, temp, q, r;

  q = n0 / b0;
  r = n0 - q * b0;
  while (r > 0) {
            temp = t0 - q * t;
            if (temp >= 0) temp = temp % n;
            else temp = n - (- temp % n);
            t0 = t;
            t = temp;
            n0 = b0;
            b0 = r;
            q = n0 / b0;
            r = n0 - q * b0;
  }
  if (b0 != 1) return 0;
  else return t % n;
}

int main(void)
{
  long a = 101, alpha = 4, beta = 449, e1 = 45;
  long e2 = 237, f1 = 125, f2 = 9, i, j, p = 467;
```

```
  long q, x = 286, y = 83, c, d, C, D, r, s, t;
clrscr();
  q = (p - 1) >> 1;
  printf("a = %ld\n", a);
  printf("alpha = %ld\n", alpha);
  printf("beta = %ld\n", beta);
  printf("e1 = %ld\n", e1);
  printf("e2 = %ld\n", e2);
  printf("f1 = %ld\n", f1);
  printf("f2 = %ld\n", f2);
  printf("p = %ld\n", p);
  printf("q = %ld\n", q);
  printf("x = %ld\n", x);
  printf("y = %ld\n", y);
  i = Extended_Euclidean(a, q);
  c = (exp_mod(y, e1, p) * exp_mod(beta, e2, p)) % p;
  d = exp_mod(c, i, p);
  printf("Alice's challenge c = %ld\n", c);
  printf("Bob's response d = %ld\n", d);
  if (d != (exp_mod(x, e1, p) * exp_mod(alpha, e2, p)) % p)
    printf("d != x ^ e1 * alpha ^ e2 mod p\n");
  else
    printf("d == x ^ e1 * alpha ^ e2 mod p\n");
  C = (exp_mod(y, f1, p) * exp_mod(beta, f2, p)) % p;
  D = exp_mod(C, i, p);
  printf("Alice's challenge C = %ld\n", C);
  printf("Bob's response D = %ld\n", D);
  if (D != (exp_mod(x, f1, p) * exp_mod(alpha, f2, p)) % p)
    printf("D != x ^ f1 * alpha ^ f2 mod p\n");
  else
    printf("D == x ^ f1 * alpha ^ f2 mod p\n");
  i = q - e2;
  if (i < 0) i += q;
  j = q - f2;
  if (j < 0) j += q;
  r = (d * exp_mod(alpha, i, p)) % p;
  s = exp_mod(r, f1, p);
  r = (D * exp_mod(alpha, j, p)) % p;
  t = exp_mod(r, e1, p);
  if (s == t)
    printf("Alice concludes y is a forgery\n");
  else
    printf("Alice does not conclude y is a forgery\n");
  return 0;
}
```

OUTPUT:

```
a = 101
alpha = 4
beta = 449
e1 = 45
e2 = 237
f1 = 125
```

f2 = 9
p = 467
q = 233
x = 286
y = 83
Alice's challenge c = 305
Bob's response d = 109
d != x ^ e1 * alpha ^ e2 mod p
Alice's challenge C = 270
Bob's response D = 68
D != x ^ f1 * alpha ^ f2 mod p
Alice concludes y is a forgery

Program 7
Aim: write a program to implement n queens

```
/* N QUEENS */
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<graphics.h>
class queen
{       int n,x[100];
        public:
                queen();
                void nqueen(int,int);
                int place(int,int);
};
queen::queen()
{
        int k=1,n;
        textcolor(GREEN);
        cprintf("Enter the number of queens:");
        cin>>n;
        nqueen(k,n);
}
void queen::nqueen(int k,int n)
{
        for(int i=1;i<=n;i++)
        {
                if(place(k,i))
                {
                        x[k]=i;
                        clrscr();
                        if(k==n)
                        {
                                for(int j=1;j<=n;j++)
                                {textcolor(x[j]);
                                gotoxy(1,j);
                                cprintf("queen %d : %d",j,x[j]);}
                                getch();

                                break;
                        }
                        else
                        nqueen(k+1,n);
                }
        }
        return;
}
int queen::place(int k,int i)
{
        int j;
        for(j=1;j<=k;j++)
        {
                if((x[j]==i)||(abs(x[j]-i)==abs(j-k)))
                return 0;
        }
```

```
        return 1;
}
void main()
{
        clrscr();
        queen q;
}
```

OUTPUT:

Enter the number of Queens : 4

Queen 1 : 3
Queen 2 : 1
Queen 3 : 4
Queen 4 : 2

Enter the number of Queens : 8

Queen 1 : 1 1 2
Queen 2 : 5 7 8
Queen 3 : 8 4 6
Queen 4 : 6 6 1
Queen 5 : 3 8 3
Queen 6 : 7 2 5
Queen 7 : 2 5 7
Queen 8 : 4 3 4

queen 1 : 2
queen 2 : 8
queen 3 : 6
queen 4 : 1
queen 5 : 3
queen 6 : 5
queen 7 : 7
queen 8 : 4

Program 8
Aim: write a program to implement encryption and decryption

```
/*  ENCRYPTION  and DECRYPTION*/


/*This Program Will Encrypt Or Decrypt Ant Text Document*/
/* This Program Will Encrypt Any Type Of Text Document */
#include<fstream.h>
#include<conio.h>
#include<stdio.h>
int main()
{
char name[30],target[30],ch,mod;
//Declare Variables //
int num[100],i,option;
clrscr();
cout<<"\nEnter Your Option ";
cout<<"\n1. To Encrypt The File ";
cout<<"\n2. To Decrypt The File ";
cout<<"\nOption : ";
cin>>option;
if(option==1)
{
cout<<"\nEnter The Path Of A File Which Is To Be Encrypted : ";
gets(name);
ifstream fin(name,ios::binary);
//Open The Input File In A Binary Mode//
if(!fin)
{
//Show Error Occur If File Does Not Exist//
cout<<"\nError In Openinig Of A File : ";
//Or Any Error Occurs //
return 1;
}
cout<<"\nEnter The New Encrypted File Name : ";
gets(target);
ofstream fout(target,ios::binary);
//Open The OutPut File In A Binary Mode//
if(!fout)
{
cout<<"\nError In Opening Of Target File : ";
//Show Error If Any Error Occrs In Opening Of A File//
return 1;
}
for(i=0;i<9;i++)
{
//Multiple for Loops For Storing The Numbers//
num[i]=i;
//An Array //
}
for(i=14;i<31;i++)
//Loops Will Store 100 Numbers//
{
num[i-5]=i;
```

```
//which Will Encrypt The Contents Of A File//
}
for(i=33;i<=68;i++)
//To Avoid Theerror Ocuur Caused By The//
{
//Enter Key,Tab Key & Space Key//
num[i-7]=i;
//TheseVariations In Loops Is Made//
}
for(i=97;i<=122;i++)
{
num[i-35]=i;
}
while(fin)
{
//Open The Input File//
fin.get(ch);
if(ch==EOF)break;
//Exit To Loop When End Of File//
if((ch>=97) && (ch<=122))
{
//Encrypt The Small Letters//
i=97;
mod=num[ch-i];
fout<<mod;
}
if((ch>=65) && (ch<=90))
{
i=39;
//Encrypt The Capital Letters//
mod=num[ch-i];
//And Store In An Output File//
fout<<mod;
}
if((ch>=48) && (ch<=57))
{
i=4;
//Encrypt The Numbers//
mod=num[ch+i];
fout<<mod;
}
if((ch==10)||(ch==13))
{
mod=ch;
//For Enter Key//
fout<<mod;
}
if(ch==32)
fout<<ch;
//For Space Key//
if(ch==9)
fout<<ch;
//For Tab Key//
if((ch>=33)&&(ch<=47))
{
//For Special Symbols//
```

```
mod=ch+64;
fout<<mod;
}
if((ch>=58)&&(ch<=64))
{
//For Special Symbols//
mod=ch+54;
fout<<mod;
}
if((ch>=91)&&(ch<=96))
{
mod=ch+28;
//For Special Symbols//
fout<<mod;
}
if((ch>=123)&&(ch<=126))
{
mod=ch-40;
//For Special Symbols//
fout<<mod;
}
}
fin.close();
//Close The Input File//
fout.close();
//Close The Output file//
cout<<"\nYour File Is Encrypted Now..... ";
getch();
return 0;
}
/*=============================================================================
=======================================*/
/* This Program Will Decrypt The Document */
if(option==2)
{
char name[30],target[30],ch,mod;
//Declare Variables//
int num[100],i,flag;
cout<<"\nEnter The Path Of A File Name Which Is To Be Decrypted : ";
gets(name);
ifstream fin(name,ios::binary);
if(!fin)
//Open The Encryped File In A Binary Mode//
{
cout<<"\nError In Opening Of A File : ";
return 1;
//Show Error If File Does Not Exist//
}
//Or Any Occurs In Opening Of A File//
cout<<"\nEnter The New Decrypted File Name : ";
gets(target);
ofstream fout;
fout.open(target,ios::binary);
//Opens The Output File In An Binary Mode//
if(!fout)
{
```

```
//Show Error if Any Error Occurs In Opening Of A File//
cout<<"\nError In Opening Of A Target File : ";
return 1;
}
for(i=0;i<9;i++)
{
num[i]=i;
//In An Array//
}
for(i=14;i<31;i++)
{
num[i-5]=i;
//Loops Will Store 100 Numbers//
}
for(i=33;i<=68;i++)
//Which  encrypts The Document Also Decrypt It//
{
num[i-7]=i;
}
while(fin)
{
//Opens The Encryped File//
fin.get(ch);
flag=0;
//Turn Off Flag//
if(ch==EOF)break;
for(i=26;i<52;i++)
{
if(ch==num[i])
//Loop For Match The Small Letters Letters//
{
mod=i+39;
//If Match Then Put Appropriate Letter//
fout<<mod;
//In A OutPut File//
flag=1;
break ;
//Turn On Flag And Exit The Loop//
}
}
if (flag==1) continue ;
//If Flag Is On Then Continue Outer Loop//
for(i=0;i<26;i++)
{
//Loop For Match The Capital Letters//
if(ch==num[i])
{
//If Match Then Put Appropriate Letter//
mod=i+97;
//In A OutPut File//
fout<<mod;
flag=1;break;
//Turn On Flag And Exit From This Loop//
}
}
if (flag==1) continue ;
```

```
//If Flag Is On Then Continue Outer Loop//
for(i=52;i<62;i++)
{
//Loop For Numerical Numbers//
if(ch==num[i])
{
mod=i-4;
fout<<mod;
flag=1; break ;
}
}
if (flag==1) continue ;
if((ch==10)||(ch==13))
{
mod=ch;
//Condition For Enter Key//
fout<<mod;
}
if(ch==32)
fout<<ch;
//condition For Space Key//
if(ch==9)
fout<<ch;
//Condition For Tab Key//
if((ch>=97)&&(ch<=111))
{
mod=ch-64;
//For Special Symbols//
fout<<mod;
}
if((ch>=112)&&(ch<=118))
{
mod=ch-54;
//For Special Symbols//
fout<<mod;
}
if((ch>=119)&&(ch<=124))
{
mod=ch-28;
//For Special Symbols//
fout<<mod;
}
if((ch>=83)&&(ch<=86))
{
//For Special Symbols//
mod=ch+40;
fout<<mod;
}
}
fin.close();
//Close The Encrypted File//
fout.close();
//Close Your Original Decrypted File//
cout<<"\nThe File Is Being Decrypted..... ";
getch();
return 0;
```

```
}
return 0;
}
```

OUTPUT:

```
Enter Your Option
1. To Encrypt The File
2. To Decrypt The File
Option : 1

Enter The Path Of A File Which Is To Be Encrypted : c:\a.txt

Enter The New Encrypted File Name : c:\enc.txt

Your File Is Encrypted Now.....
Enter Your Option
1. To Encrypt The File
2. To Decrypt The File
Option : 1

Enter The Path Of A File Which Is To Be Decrypted : c:\enc.txt

Enter The New Decrypted File Name : c:\dec.txt

The File Is Being Decrypted .....
```