Netcat

The TCP/IP Swiss Army Knife

“Learned Skills in Practice”

Lab Objectives

In this lab, you will perform the following six exercises:

- Exercise 1 - Use Netcat for port scanning
- Exercise 2 - Use Netcat for Banner Grabbing
- Exercise 3 - Use Netcat to transfer files between hosts
- Exercise 4 – Use Netcat to configure a remote backdoor
- Exercise 5 – Use Netcat for chatting between two users
- Exercise 6 – Use Netcat to perform a remote backups

Lab Overview

In this lab you will get the opportunity to try most of the neat features of Netcat that has been demonstrated by the instructor.

This lab can be performed in two ways:

1. It can be done locally on your own computer using the Windows VM guest operating system in conjunction with your BackTrack VM.
2. The second method is to team with the person sitting beside you. The results and procedure will be the same, choose the method that is more conducive to your learning style.

At this point if you do not have your VMware Backtrack VM up and running, do turn it on.

Text that is grayed out indicates instruction and explanations. If you're already an experience Netcat user you could skip over and go directly to what has to be done below. However, be warned that you must read ahead and see what has to be done, if you do not follow the instructions closely, it will simply not work.

The graphics are only examples. YOU MUST adapt the example to your own IP address or the specific filename that you wish to use. When you transfer a file, the file must exist or else it will not automagically create the file for you.

Estimated Completion Time

Exercise 1 - 10 minutes
Exercise 2 - 10 minutes
Exercise 3 - 10 minutes
Exercise 4 - 10 minutes
Exercise 5 - 10 minutes
Exercise 6 - 10 minutes

Total Time – 60 minutes

Exercise 1 Procedures

Using Netcat for port scanning

First let's do a quick refresher on the command syntax of Netcat.

The most basic form of command is:

 # nc [options] host port(s)

Options are described below
Host can be either an IP address or valid hostname
Ports can be a single port or a range of ports such as 20-53 or individual ports separated by spaces. A port argument is always required for outbound connections, it can be numeric or a name listed in /etc/services. However, if you use the **–n** switch only numeric arguments will be valid for the port range.

NOTE: At the end of the lab you have a list of all of the switches we have mentioned and their meaning. If you're stuck you can consult this list.

LET'S GET STARTED

Netcat has the ability to do either UDP or TCP port scanning. Of course there are better tools out there to do this but sometimes some of these tools do require the installation of libraries in order to work properly. It is NOT always possible to install libraries on remote hosts that have been compromised. So a small tool that can do remote port scanning would be nice to have and Netcat can fill this role very well and a lot of other ones.

A typical command to perform port scanning would be:

 # nc –v –w 3 –z 192.168.1.69 20-150

The first portion of the command line that says: **nc –v –w 3** which simply tells Netcat to give us more verbose feedback and to timeout after 3 seconds if no connections could be established.

The **–z** switch prevent Netcat from sending any data to a TCP connection and it will only send very limited data to a UDP connection.

The target in this case is: **192.168.1.69**

Last but not least, we have the port range that will be used: **20-150**.

TASK 1

You will perform a port scan using the TCP protocol. You will port scan your partner computer or your VM machine if you're working on your own. While doing this scan you wish to have more verbose output than the default Netcat setting, you would also like to setup a timeout value of 5 seconds, and you will scan the port range 20 to 100. Please write below what would be the specific command you used to do this.

Answer:

_____

_____

Also write down below what ports numbers were detected as Open as you will need these results a bit further in the lab.

Ports detected:

_____

_____

TASK 2

In this task you will repeat what you have done in TASK 1 but you will use the UDP protocol instead of the TCP protocol. Write down the syntax used below.

Answer:

_____

_____

TASK 3

As you have seen in your previous tasks, scanning could be very noisy on a network. You will most likely be detected by any anomaly or intrusion detection devices. In order to do this smarter you will slow down your scan and use an interval of 5 second between probe and you will randomize your port numbers as well. Write down the syntax below.

Answer:

_____

_____

_____

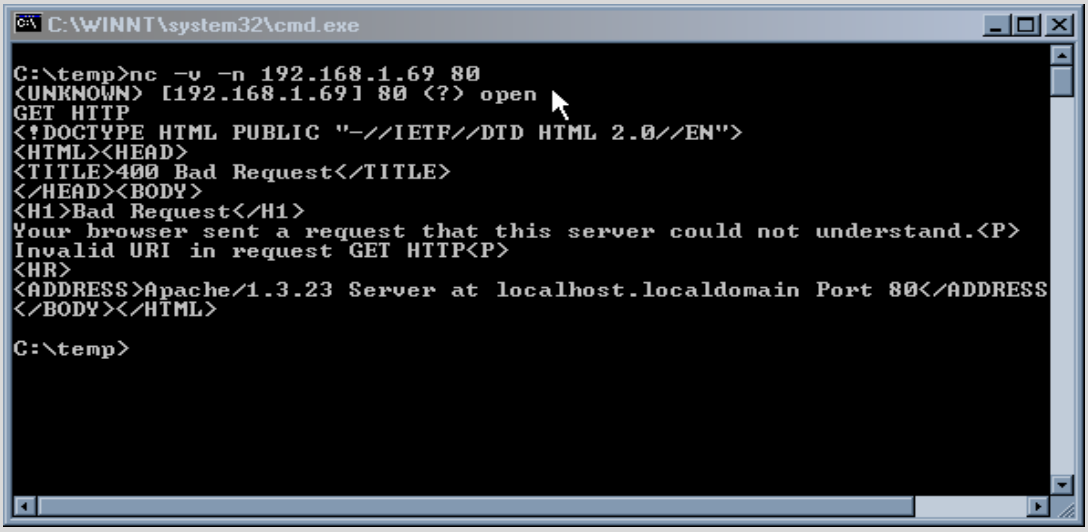
_____

Exercise 2 Procedures

Using Netcat for banner grabbing

Another neat usage of Netcat is for banner grabbing. As you have seen in the TCP port scan that you have performed above, there were some ports that were reported as open. Let's do a bit more digging to see what could the application running behind those ports.

An example of such command is demonstrated below:



Once connected above you simply issued the command: **GET HTTP** which will generate and error as it is an invalid HTTP command and the web server identifies itself Apache 1.3.23 (Interesting).

TASK 1

Using the ports that were identified in your TCP scan above, perform banner grabbing to see what could be the applications running on those ports. Do this for all of the TCP ports that were detected. Write your results below.

Results:

_____

_____

_____

_____

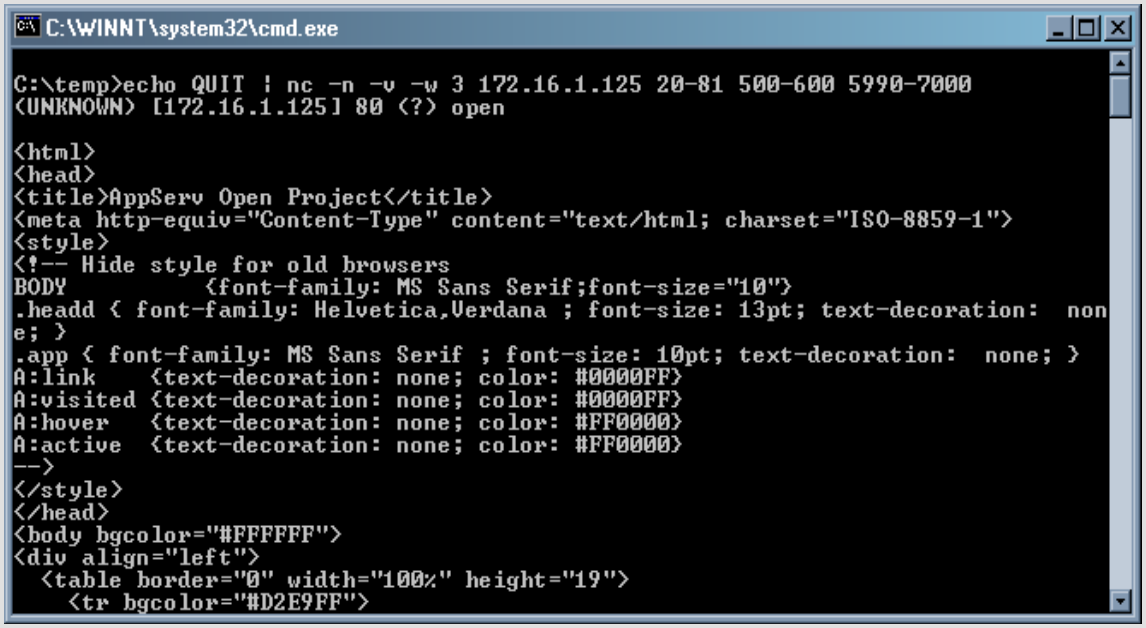
_____

COMBINING BOTH PORT SCAN AND BANNER IDENTIFICATION

Now we will combine both Port Scan and banner identification.

As you have seen above, it could become quite a task to manually verify every port that was found manually one by one. So let's get a bit more sophisticated and do this in one easy command.

An example of such scanning would be:



The command above will inform you about a target's various well-known TCP servers, including r-services, X, IRC, and maybe a few you didn't expect. Sending in QUIT and using the timeout will almost guarantee that you see some kind of greeting or error from each service, which usually indicates what it is and what version. As you can see above the web server was identified and the default page returned.

TASK 2

Perform a banner and port identification against your partner machine or your VM machine if working alone

Did you get any results? Was Netcat able to identify any of the services?

_____

_____

_____

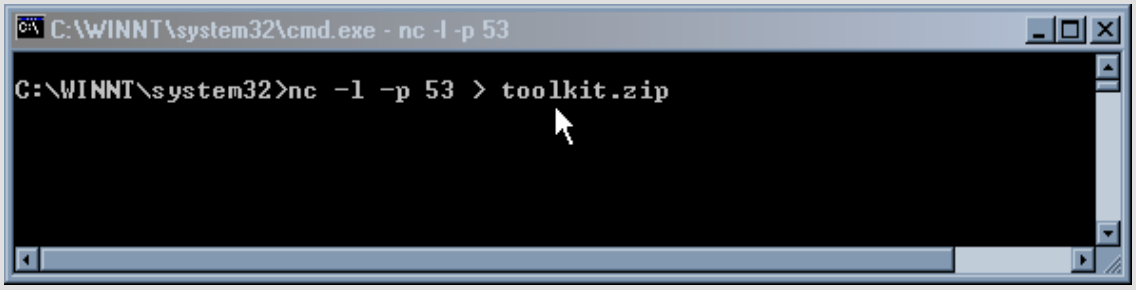
Exercise 3 Procedures

Using Netcat to transfer files

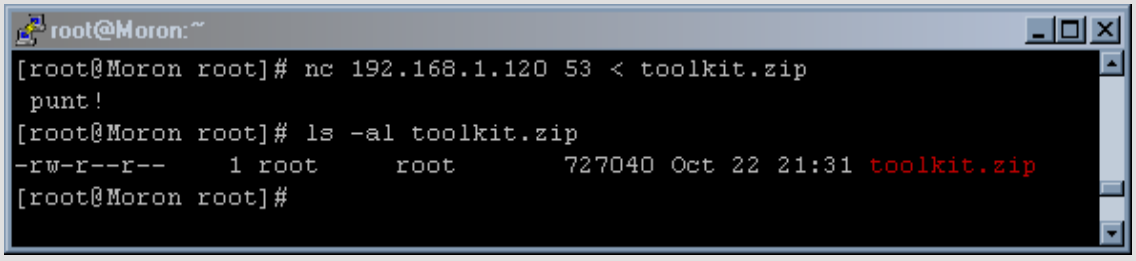
Netcat can be used as well to transfer files between two hosts. It has flexible command syntax and can bind to any ports that it is allowed to bind. It can be a very effective tool to transfer the files that you need in order to escalate your privileges on a remote host. By choosing an appropriate port you could avoid firewall restrictions as well. It seems that port 53 TCP is one that is often open by default on lots of the firewalls.

Let's suppose we have the following scenario. You have a Windows host that you successfully compromised and you wish to transfer additional tools to it in order to escalate your privileges.

The file transfer has to be done in two steps. First you must setup a listener on your compromised Windows host as demonstrated below:



The second step consists of connecting to our listener and transferring the file as demonstrated below:



TASK 1

You will now choose a file that you wish to transfer from your BackTrack VM to your Windows host. Using the examples above and your course note transfer the file across and ensure it was transferred correctly.

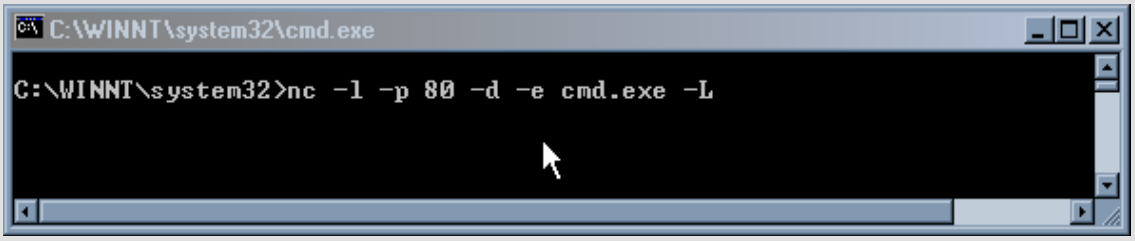
Exercise 4 Procedures

Setting up a backdoor

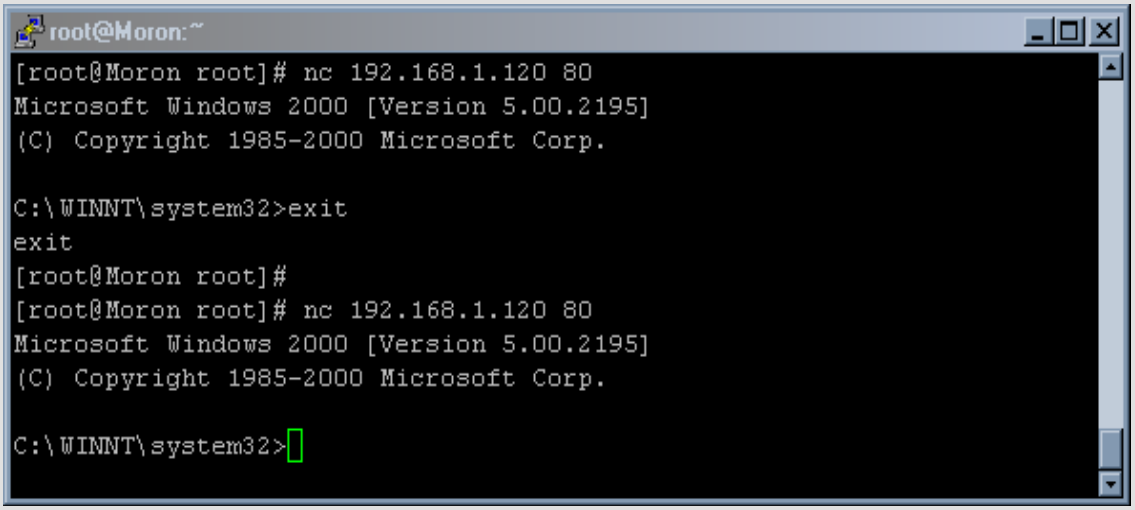
Netcat has the ability of providing a remote shell as well. This can be useful if you wish to easily come back to a host after the host has been compromised. No complicated setup or registry entries to change.

Let say that we have the same scenario as we did above. You have compromised a Windows host and you wish to establish a backdoor that you will be able to connect to from you BackTrack attack machine.

First you will setup a listener on the compromised windows host:



Now we will connect to the backdoor as demonstrated below:



TASK 1

You will now setup a listener on your windows system that will wait for connection to a port of your choice. Be aware that port 80 might already be used; pick a random port above 1024 for better results. Once the listener is up and running, you will connect to the listener from your BackTrack VM.

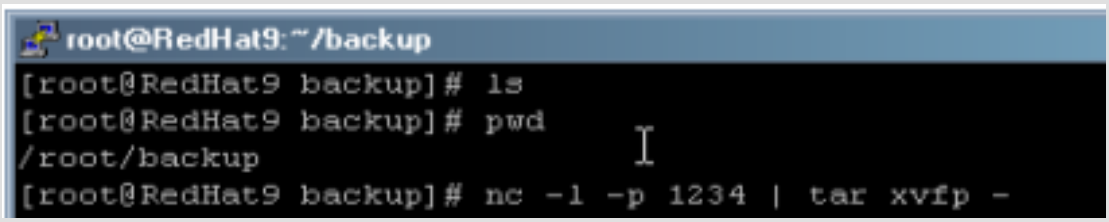
Exercise 5 Procedures

Creating a backup on a remote machine

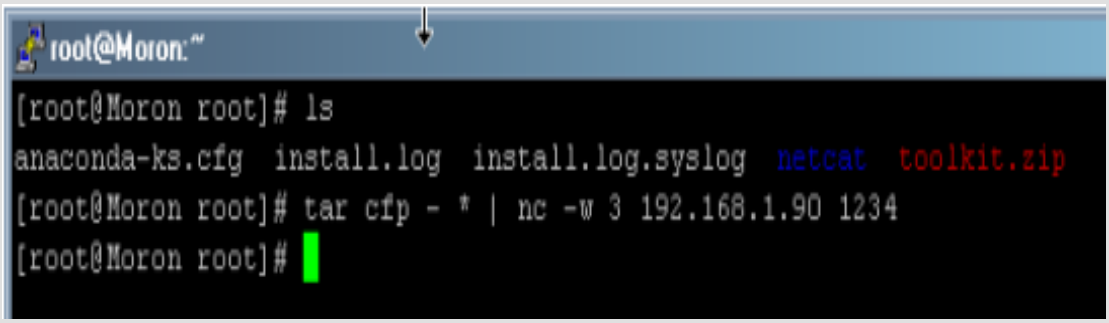
THIS EXERCISE MUST BE DONE IN PAIR WITH YOUR PARTNER AS YOU NEED TWO LINUX MACHINES.

Often time you have a requirement to move files from one machine to another. Netcat comes very handy for such a task. It could be for backup purpose or simply to escalate your privilege on a compromise host, Netcat will come to the rescue.

First you must setup a listener on the host that will receive the files. After issuing the **nc –l –p 134 | tar xvfp –** command, it seems like no activity is taking place. As soon as the transfer is initiated on the remote side you will see a list of files as they are sent across the Netcat pipe.



On the remote side, you issue the following command to send the files across:



TASK 1

In pair with your partner, attempt to send a TAR backup across a Netcat link.

EXTENSIVE LIST OF NETCAT SWITCHES

As it was mentioned at the beginning of the lab, Netcat can be used for multiple purposes and not all of them have been practice within this lab. Covering everything you can do with Netcat would require a full day of exploration.

Below you have a summary of switches mentioned within the Readme file that is shipped with Netcat.

SWITCH	DESCRIPTION
-v	The -v switch controls the verbosity level of messages sent to standard error (Usually it is your screen). You will probably want to run Netcat most of the time with -v turned on, so you can see info about the connections it is trying to make. If -v is not specified at all, Netcat silently does its work unless some error happens, whereupon it describes the error and exits with a nonzero status. Refused network connections are generally NOT considered to be errors, unless you only asked for a single TCP port and it was refused.
-w	You will probably also want to give a smallish -w argument, which limits the time spent trying to make a connection. A good trick is to alias "nc" to "nc -v -w 3" The timeout is easily changed by a subsequent -w argument which overrides the earlier one. Specifying -v more than once makes diagnostic output MORE verbose. Note that -w also sets the network inactivity timeout. This does not have any effect until standard input closes, but then if nothing further arrives from the network in the next seconds, Netcat tries to read the net once more for good measure, and then closes and exits. There are a lot of network services now that accept a small amount of input and return a large amount of output, such as Gopher and Web servers, which is the main reason Netcat was written to "block" on the network staying open rather than standard input. Handling the timeout this way gives uniform behavior with network servers that *don't* close by themselves until told to.
-u	UDP connections are opened instead of TCP when -u is specified. These aren't really "connections" per se since UDP is a connectionless protocol,

	although Netcat does internally use the "connected UDP socket" mechanism that most kernels support. Although Netcat claims that an outgoing UDP connection is "open" immediately, no data is sent until something is read from standard input. Only thereafter is it possible to determine whether there really is a UDP server on the other end, and often you just can't tell. Most UDP protocols use timeouts and retry to do their thing and in many cases won't bother answering at all, so you should specify a timeout and hope for the best. You will get more out of UDP connections if standard input is fed from a source of data that looks like various kinds of server requests.
-o	To obtain a hex dump file of the data sent either way, use "-o logfile". The dump lines begin with "<" or ">" to respectively indicate "from the net" or "to the net", and contain the total count per direction, and hex and ascii representations of the traffic. Capturing a hex dump naturally slows Netcat down a bit, so don't use it where speed is critical.
-e	If netcat is compiled with -DGAPING_SECURITY_HOLE, the –e argument specifies a program to exec after making or receiving a successful connection. In the listening mode, this works similarly to "inetd" but only for a single instance. Use with GREAT CARE. This piece of the code is normally not enabled; if you know what you're doing, have fun. This hack also works in UDP mode. Note that you can only supply -e with the name of the program, but no arguments. If you want to launch something with an argument list, write a two-line wrapper script or just use inetd like always. Also read about the –d options below as it is commonly used in conjunction with the –e switch.
-t	If netcat is compiled with -DTELNET, the -t argument enables it to respond to telnet option negotiation [always in the negative, i.e. DONT or WONT]. This allows it to connect to a telnetd and get past the initial negotiation far enough to get a login prompt from the server. Since this feature has the potential to modify the data stream, it is not enabled by default. You have to understand why you might need this and turn on the #define yourself.
-i	The -i switch specifies an "interval time" which slows this down considerably. Standard input is still read in large batches, but Netcat then tries to find where line breaks exist and sends one line every interval time.

	Note that if standard input is a terminal, data is already read line by line, so unless you make the -i interval rather long, what you type will go out at a fairly normal rate. -i is really designed for use when you want to "measure out" what is read from files or pipes.
-z	The -z switch prevents sending any data to a TCP connection and very limited probe data to a UDP connection, and is thus useful as a fast scanning mode just to see what ports the target is listening on. Example: nc -v -w 2 -z target 20-30 The above command will try connecting to every port between 20 and 30 [inclusive] at the target, and will likely inform you about an FTP server, telnet server, and mailer along the way. For each range of ports specified, scanning is normally done downward within that range, in the example above it would starts with port 30.
-i	To limit scanning speed if desired, -i will insert a delay (specified as a number of seconds) between each port probe.
-r	If the -r switch is used, scanning hops randomly around within that range and reports open ports as it finds them. If you want them listed in order regardless, pipe standard error through "sort". In addition, if random mode is in effect, the local source ports are also randomized. This prevents Netcat from exhibiting any kind of regular pattern in its scanning. You can exert fairly fine control over your scan by judicious use of -r and selected port ranges to cover. If you use –r for a single connection, the source port will have a random value above 8192, rather than the next one the kernel would have assigned you. Note that selecting a specific local port with -p overrides any local-port randomization.
-g -G	Many people are interested in testing network connectivity using IP source routing, even if it's only to make sure their own firewalls are blocking source-routed packets.

	On systems that support it, the -g switch can be used multiple times [up to 8] to construct a loose-source-routed path for your connection, and the -G argument positions the "hop pointer" within the list. If your network allows source-routed traffic in and out, you can test connectivity to your own services via remote points in the internet. Netcat's handling of "-g" is modeled after "traceroute".
-d	The –d switch allows to deteach Netcat from the command window where it was started. This is most useful when you wish to setup a backdoor that will spawn a remote shell window when connecting to a port.
-l	This switch is used to setup a listener using Netcat. This option is used in conjunction with the –p option which tells Netcat which port it will listen on for incoming connections.
-L	This option is to setup a permanent listener. If you use the –l option, Netcat will terminate after a connection has been established. Using the –L option will allow you to setup a listening that is permanent and that allows you to connect at will. The –L switch is also used in conjunction with the –p switch to specify which port it will listen on.
-n	This switch will turn off name resolution and greatly speed up your scans. If the –n switch is used, you cannot use hostnames but only valid IP addresses.
-p	This switch allows you to specify which port Netcat will be listening on. This switch is used in conjunction with the –l and –L switch that we have mentioned above.
-s	This switch is used to spoof the source address.

END OF LAB