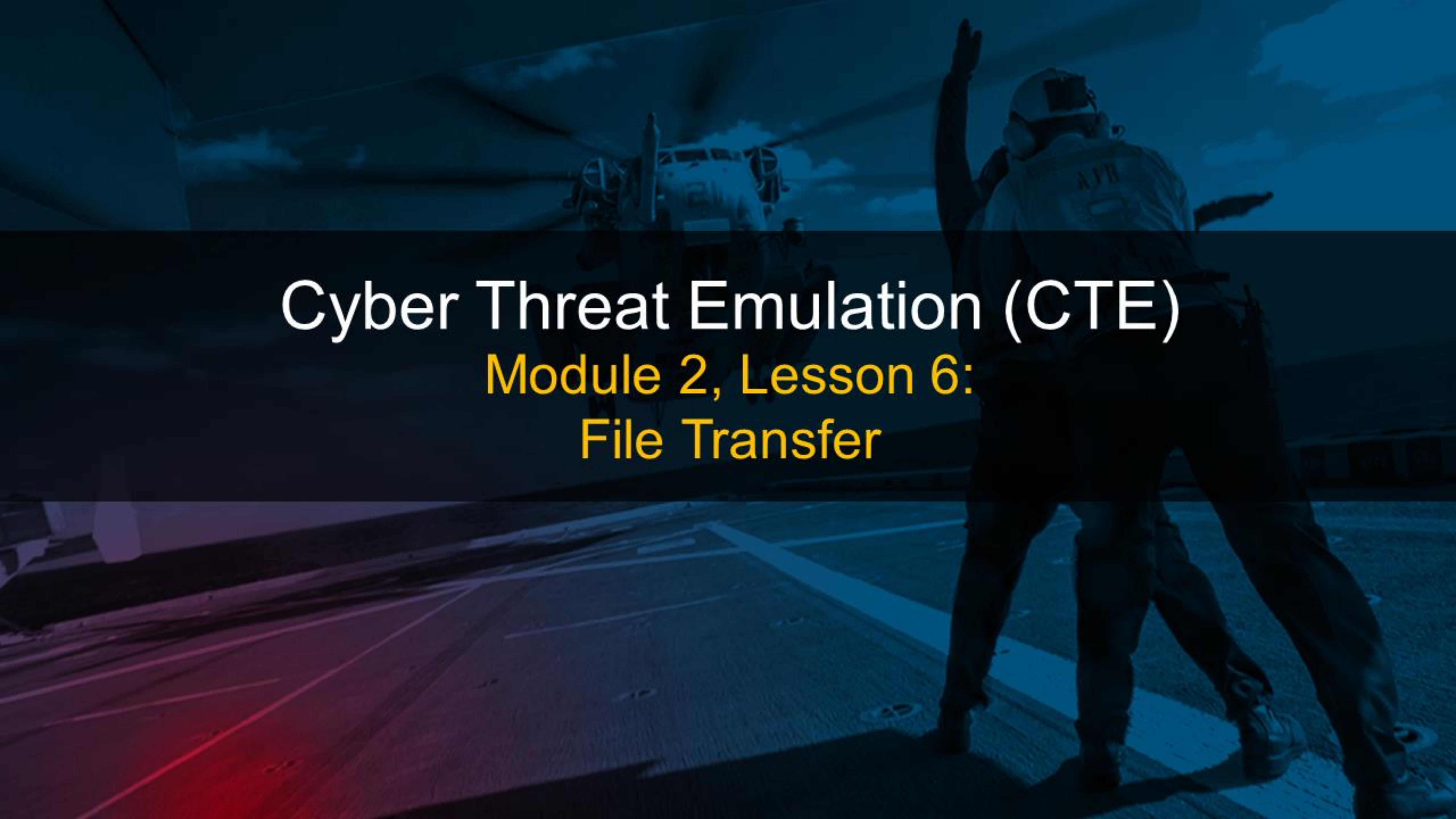


Cyber Threat Emulation (CTE)

Module 2, Lesson 6:
File Transfer

Course Objectives

After completing this course, students will be able to:

- Summarize the CTE squad's responsibilities, objectives, and deliverables from each CPT stage
- Analyze threat information
- Develop a Threat Emulation Plan (TEP)
- Generate mitigative and preemptive recommendations for local defenders
- Develop mission reporting
- Conduct participative operations
- Conduct reconnaissance
- Analyze network logs for offensive and defensive measures

Course Objectives (Continued)

Students will also be able to:

- Analyze network traffic and tunneling protocols for offensive and defensive measures
- Plan non-participative operations using commonly used tools, techniques and procedures (TTPs)

Module 2: Threat Emulation (Objectives)

- Conduct reconnaissance
- Generate mission reports from non-participative operations
- Plan a non-participative operation using social engineering
- Plan a non-participative operation using Metasploit
- Analyze network logs for offensive and defensive measures
- Analyze network traffic and tunneling protocols for offensive and defensive measures
- Plan a non-participative operation using Python
- Develop fuzzing scripts
- Develop buffer overflow exploits

Module 2 – Lesson 6: File Transfer (Objectives)

- Describe standard methods of transferring files
- Conduct file transfers with netcat
- Conduct uncommon methods of file transfers

Transferring Files

What are some methods used to transfer files?

- Raw methods (socat, netcat and others)

Common Name	Acronym	Typical Ports
Secure Copy Protocol/ Secure Shell	SCP/SSH	TCP 22
File Transfer Protocol	FTP	TCP 20, 21
Trivial File Transfer Protocol	TFTP	TCP 69
Hypertext Transfer Protocol/ Hypertext Transfer Protocol Secure	HTTP/HTTPS	HTTP: TCP 80 HTTPS: TCP 443
Server Message Block/ Common Internet File System	SMB/CIFS	SMB: TCP 445
Network File System	NFS	TCP/UDP 2049, 111

Transferring Files

Secure copy

- `scp [[user@]src_host:]src_file [[user@]dst_host:]dst_file`

Pulling file
(from another box, saving locally)

```
scp
<user@src_host:src_file>
<dst_file>
```

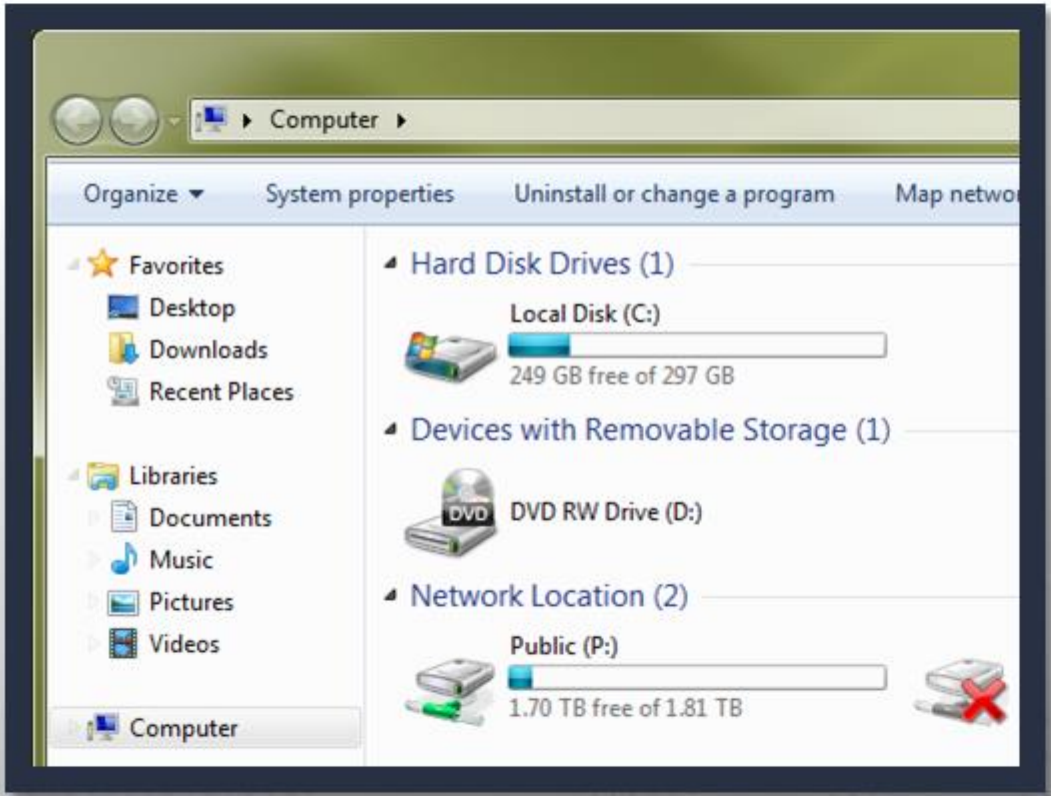
Pushing file
(from your box to a destination)

```
scp  <src_file>
<user@dst_host:dst_file>
```

Transferring Files

Windows SMB

```
net use <drive_letter>:  <sharename>
/user:[domain]\<username>
```


Computer

Organize ▾ System properties Uninstall or change a program Map networ

Favorites
Desktop
Downloads
Recent Places

Libraries
Documents
Music
Pictures
Videos

Computer

Hard Disk Drives (1)

Local Disk (C:)
249 GB free of 297 GB

Devices with Removable Storage (1)

DVD RW Drive (D:)

Network Location (2)

Public (P:)
1.70 TB free of 1.81 TB

Netcat

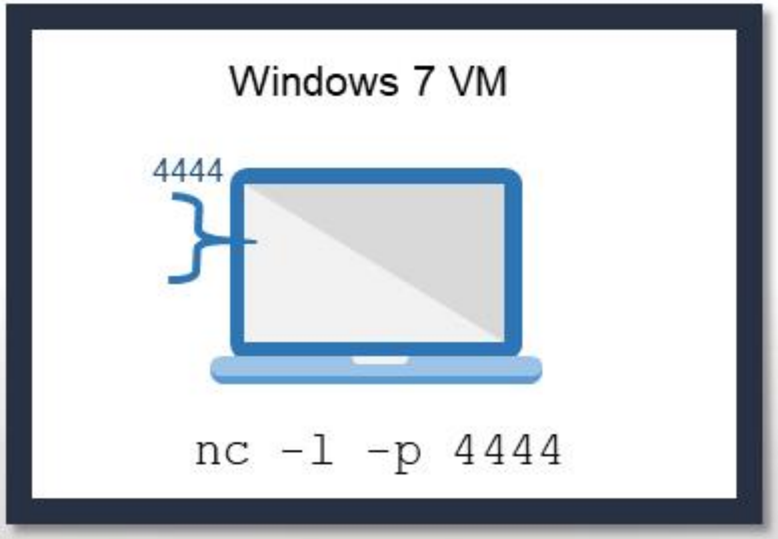
- Networking "Swiss Army knife"
- Can either initiate a TCP/UDP connection or bind to a port and listen for incoming connections
- Can be used for file transfers, banner grabbing, and port scanning
- Syntax varies depending on OS and Netcat version
- Netcat is not identical to ncat

Common Options	Use
-e <prog>	Inbound execute program, often removed
-l	Listen for inbound connections
-p <port>	Local port number
-u	UDP mode
-v	Verbose mode
-h	Help

Basic Netcat Usage

Open a listening port on your Windows 7 VM

- Don’t forget to check your syntax

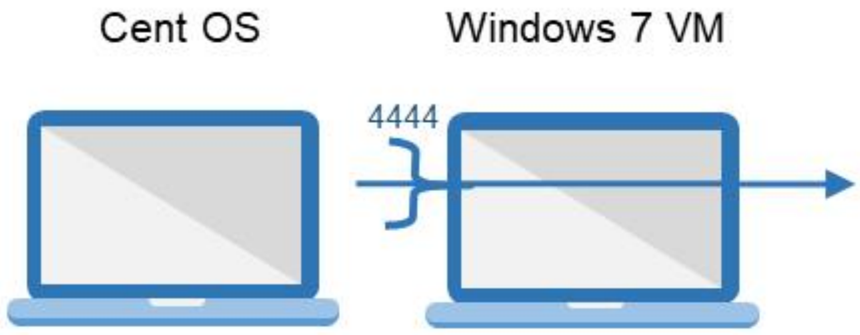


```
nc -l -p 4444
```

How can we check to see if this port is listening?

Basic Netcat Usage

Connect to Windows 7 VM from CentOS

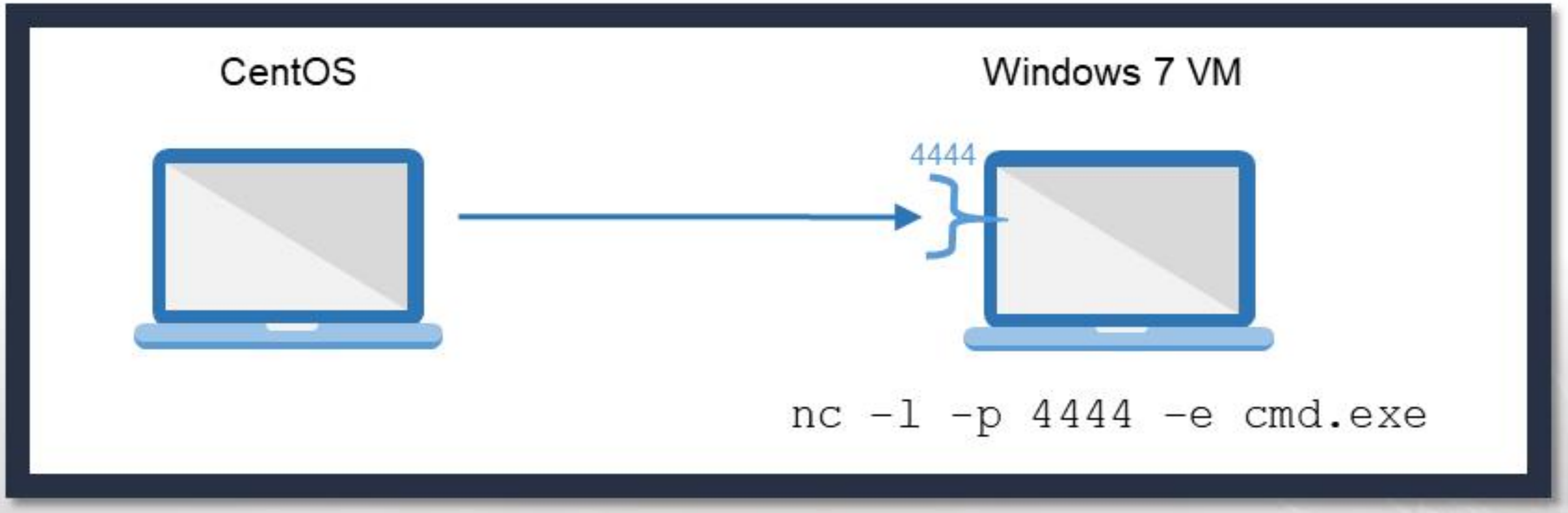


```
nc 192.168.1.15 4444
```

Note: Use Ctrl+C to break out of the connection.

Using Netcat to Get a Remote Shell

Use the `-e` option to execute a program after connection.



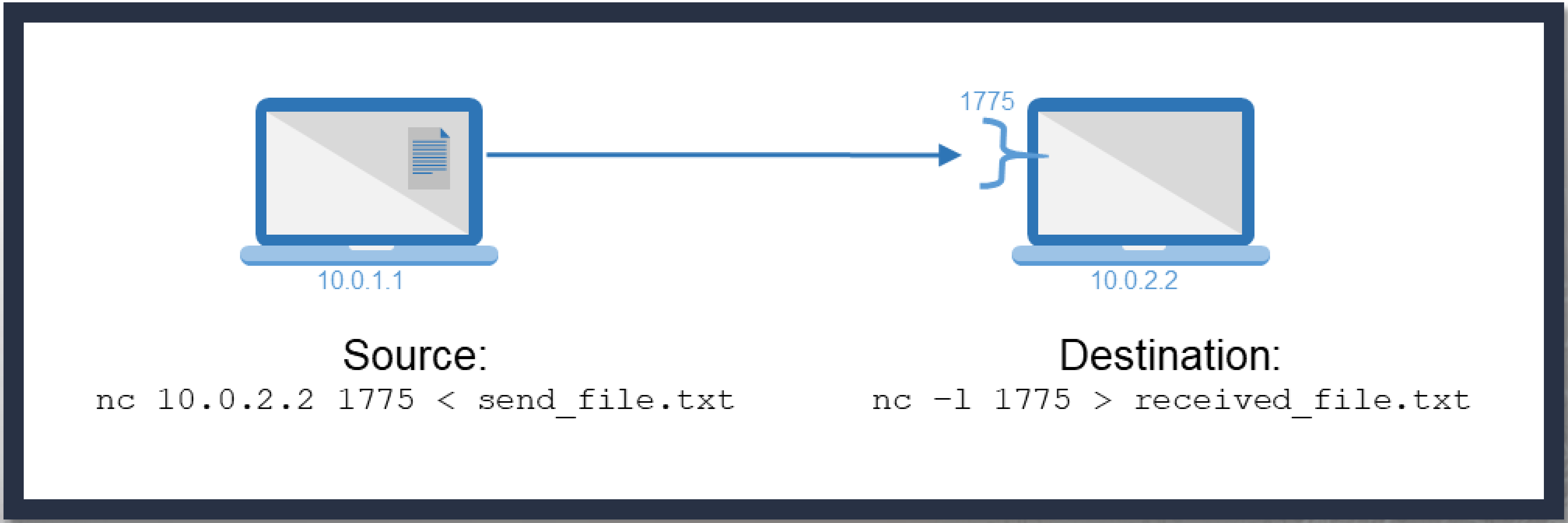
```
nc -l -p 4444 -e cmd.exe
```

Ensure the nc version you are using has the –e option.

Transferring Files With Netcat

Forward transfer

- Receiver sets up listener; sender calls forward
- Destination: `nc -l <dst_port> > <filename>`
- Source: `nc 10.0.2.2 <dst_port> < <filename>`



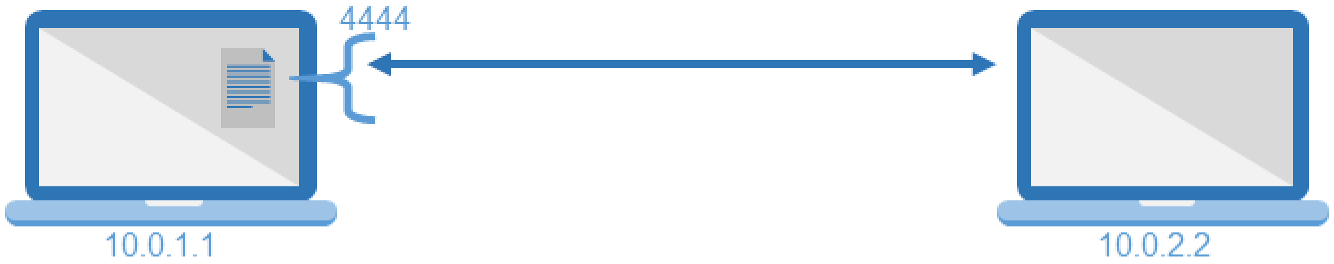
Source:
`nc 10.0.2.2 1775 < send_file.txt`

Destination:
`nc -l 1775 > received_file.txt`

Transferring Files With Netcat

Reverse transfer

- Sender sets up listener; receiver calls back
- Source: `nc -l <src_port> < <filename>`
- Destination: `nc <src_ip> <src_port> > <filename>`



Source:

```
nc -l 4444 < send_file.txt
```

Destination:

```
nc 10.0.1.1 4444 > recv_file.txt
```

Socat

- Socat accepts two bidirectional byte streams and transfers data between them.
- Typical Examples:

TCP4:<host>:<port>

- Opens TCP over IPv4 connection

TCP6-LISTEN:<port>,fork

- Opens a TCP listener on port, IPv6 only
- `fork` option – multiple simultaneous uses

UDP:<host>:<port> -open UDP connection

- Autoselect network protocol based on `<host>`

Transferring Files via Terminal

Sometimes all you have is a console window

- For example, telnet; shell from exploitation

Paste can copy text, but what about binaries?

- Need to encode as text, then paste and decode

Solutions

- uuencode/uudecode—common on UNIX
- Interpreters on target—Perl, Python, Bash, GCC
  - For example, perl has uudecode built in

Packers

Executable packers are applications that compress and obfuscate an executable

- Smaller-sized executable
- Different file hash

A common packer used by malware is UPX

- Most antivirus software detects the presence of UPX packing and flags it as possible malware

The following example is provided for the upx.exe program to create a UPX-compressed executable

- `upx.exe -o <OutFile> -<0-9> <Input File>`

Exercise: File Transfers

Objectives

After completing this exercise, students will be able to:

- Describe standard methods of transferring files
- Conduct file transfers with netcat
- Conduct uncommon methods of file transfers

Duration

This exercise will take approximately **2.5** hours to complete.

Exercise: File Transfers

Note:

Server	IP Address
Kali	10.10.1.60
Ubuntu	10.10.1.70
Windows 10	10.10.1.20
Windows 7	10.10.1.30

Debrief

General Questions

- How did you feel about this section?
- Were there any areas in particular where you had difficulty?
- Do you understand how this relates to the work you will be doing?

Summary

- One of an attacker's primary goals upon gaining entry to a targeted network is to establish a foothold and further the scope of the attack
- Tunneling and redirection permit an attacker to form channels of communication that would otherwise be impossible given existing infrastructure and setup
- Network defenders must understand these tools, tactics and procedures to gain a tactical advantage against attackers and prevent loss of critical data

End of Module 2, Lesson 6