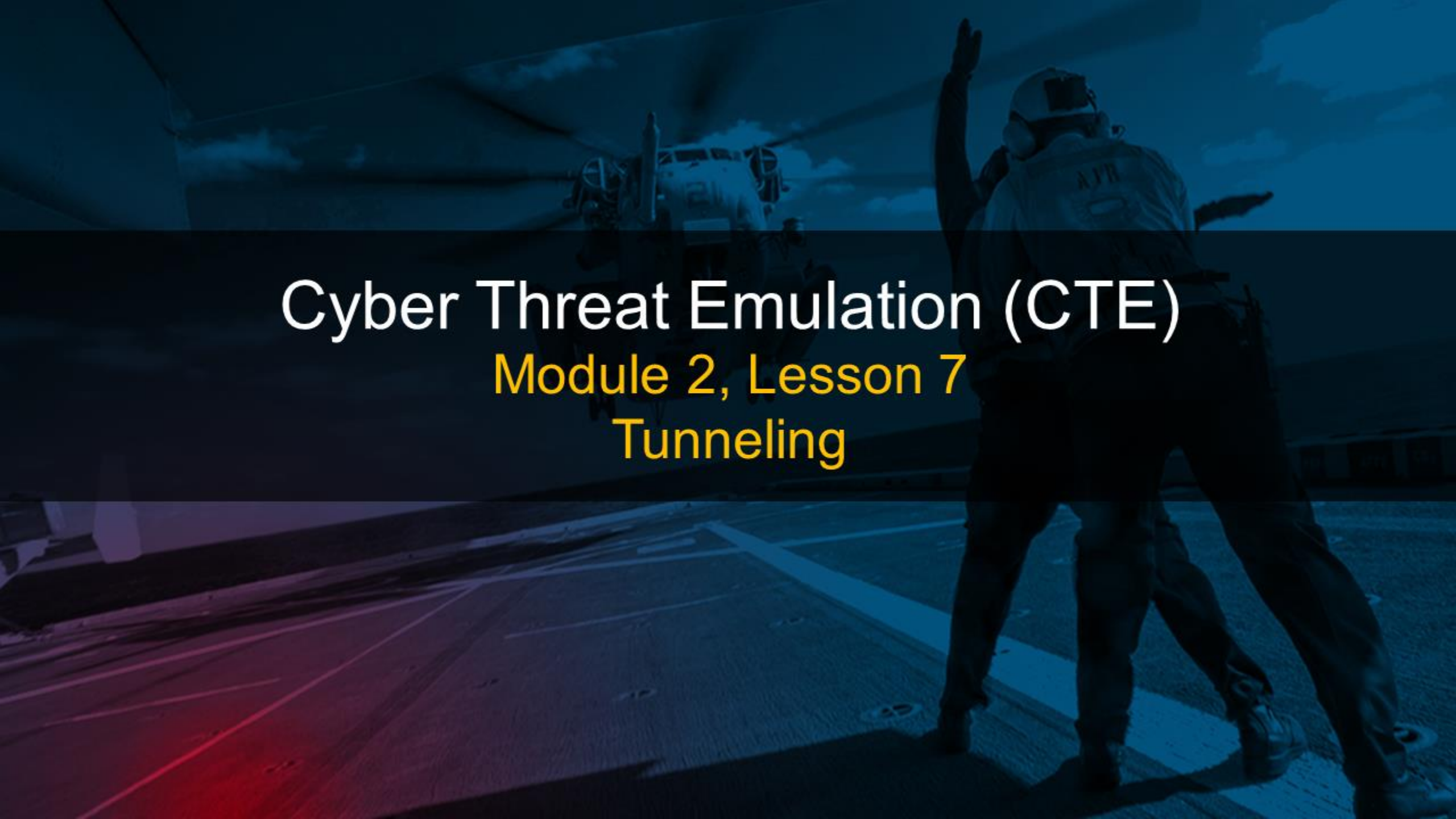


Cyber Threat Emulation (CTE)

Module 2, Lesson 7

Tunneling

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 7: Tunneling (Objectives)

- Describe the principles and methods of tunneling network traffic
- Describe the different uses of SSH
- Describe the differences between forward and reverse tunnels when using SSH
- Explain how to redirect traffic using SSH forward and reverse tunnels
- Use SSH to redirect and tunnel network traffic through multiple hosts
- Analyze network tunneling diagrams
- Describe SSH reverse tunnels and their purpose
- Recognize the difference between tunneling and redirecting network traffic

- Implement reverse SSH tunnels
- Analyze traffic to locate covert channels

Network Engineering Blues

Sometimes traffic does not play nicely with pipes it needs to go through

Between privately addressed networks

Intervening links will not route

Protocol is unsupported

Your ISP does not route IPv6 yet

Someone wants to block your traffic

Someone wants to spy on your traffic

Tunneling

Tunneling is the solution

- Put traffic you want to send inside a protocol that can get to your desired destination

Example: VPN

- Bob cannot directly access his company's internal network from home
- Bob uses a VPN client on his laptop to connect to his company's VPN concentrator, creating a tunnel
- Using the tunnel, Bob's VPN client encapsulates and encrypts all traffic destined for the company network and sends it to the concentrator

IPv6 to IPv4 Tunneling

IPv6 migration: <25% of world ISPs have adopted IPv6

- Makes it tough to be an early adopter
- Multiple solutions proposed—ISATAP, Teredo
- 6to4 Tunneling

Solution is simple

- Put IPv6 packet in an IPv4 packet
- Methodology is standardized, IPv4 next protocol 41
- Packet routes over IPv4 to the other endpoint
- IPv4 framing is stripped at the other end, and IPv6 packet is processed

Finer Points of Tunneling

- Tunneling can be at any layer of the network stack:
  - Lower levels usually integrated into OS
  - Higher levels typically into application software
- Tunneling can put lower layers into other ones.
  - Ethernet over IP … why not? (Why?)
- As long as you have the software on both ends to process it, you can tunnel any protocol over another

Secure Shell (SSH)

SSH is used for encrypted terminal access across a network

- SSH server (sshd) listens on a bound port 22
- SSH client initiates a TCP session to the server

SSH has multiple channels/tunnels

- Tunnels can be set to listen on a preconfigured port
- Tunnels forward packets to the SSH peer
- The receiving end sends packets to a preconfigured destination

Forward vs. Reverse

Each channel opens only one listener

Forward Tunnel

- SSH client opens the tunnel listener
- SSH server redirects received data
- Call forward: Initiate a connection to the remote machine

Reverse Tunnel

- SSH server opens the tunnel listener
- SSH client redirects received data
- Call back: Expect something else to establish a connection

Tunnels in Detail

Tunnel setup:

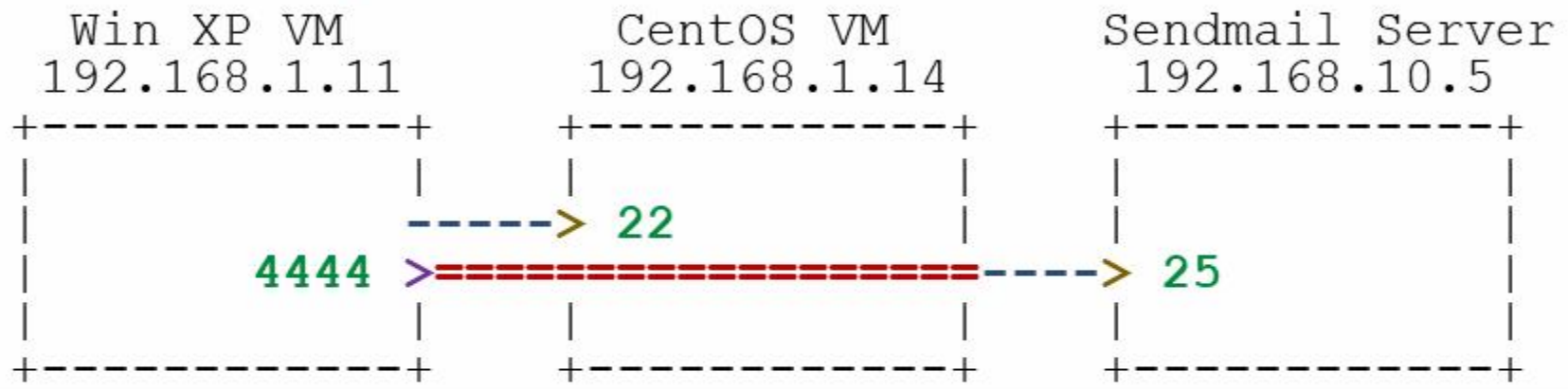
- Issue command on client:
  - `ssh <userA>@<server1> -L<lis_port>:<dst_ip>:<dst_port>`
- Client connects to server1 with userA credentials
- Client/server negotiates a channel for the tunnel
- Client creates a listening socket on `<lis_port>`
- Server redirects traffic traveling through tunnel to `<dst_ip:dst_port>`

Tunneling in Detail

Connection:

- Connect to the tunnel listener using client software (e.g., ssh, telnet, web browser, netcat)
- Client negotiates the TCP handshake with the tunnel listener
- Packet from the client is passed through the tunnel
- SSH peer negotiates the TCP handshake with the intended target
- Data are forwarded to the intended destination
- All subsequent packets flow through tunnels and are redirected

Tunnel Diagrams



First line shows SSH connection:

- Single dash (----) represents a TCP connection
- > denotes that the host is listening on a public interface

Second line represents a TCP connection to the third host via the tunnel:

- > denotes that the host is listening on a local loopback interface
- ==== represents the SSH tunnel

Operational Concept

Conceptually, split machines into three types:

1 Ops Machines

- Machines under your direct physical control
- Can reconfigure, add software, and more at will

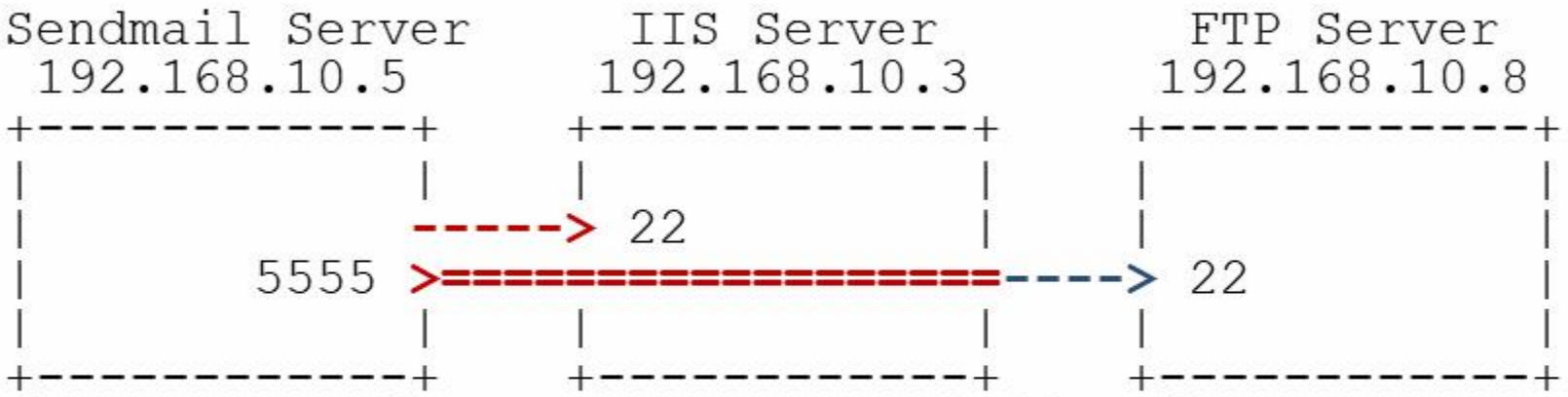
2 Redirectors

- Machines to which you have access but not control
- Standalone tools can be uploaded

3 Target

- Machine you are trying to access

SSH Into Remote Machine



Connect to redirector, setup tunnel with redirector:

- `ssh administrator@192.168.10.3 -L5555:192.168.10.8:22`

Connect to target host through tunnel:

- `ssh root@127.0.0.1 -p 5555`
- Network destination changes to local listener
- Username/password remain the same for the target host

Multiple Operations Boxes

By default, forward tunnels listen on localhost (127.0.0.1)

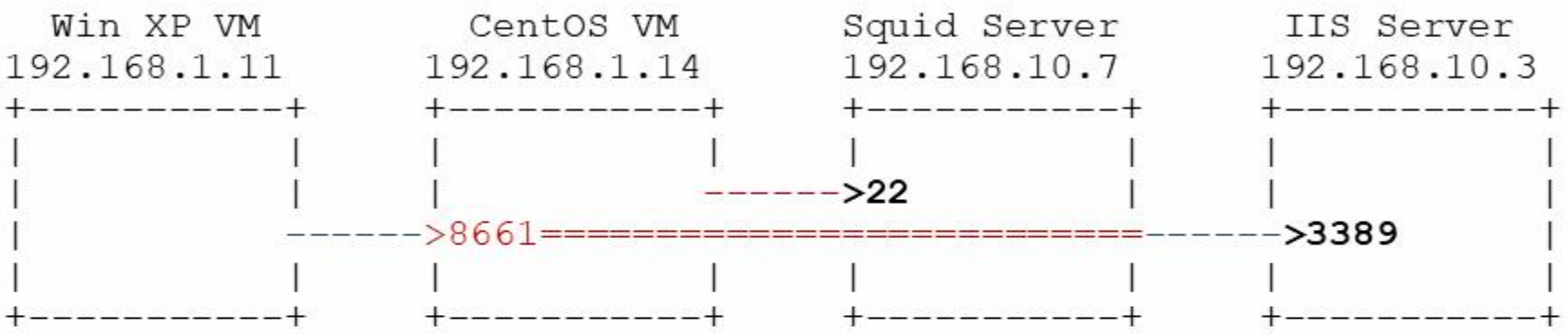
May want multiple ops boxes to access a tunnel:

- Usually set up tunnels w/Linux (better SSH tools)
- Client may be Windows based (RDP, SMB)

Can configure using ssh syntax:

- -gL14560:192.168.100.4:22
- -L0.0.0.0:14560:192.168.100.4:22

Multiple Operations Boxes



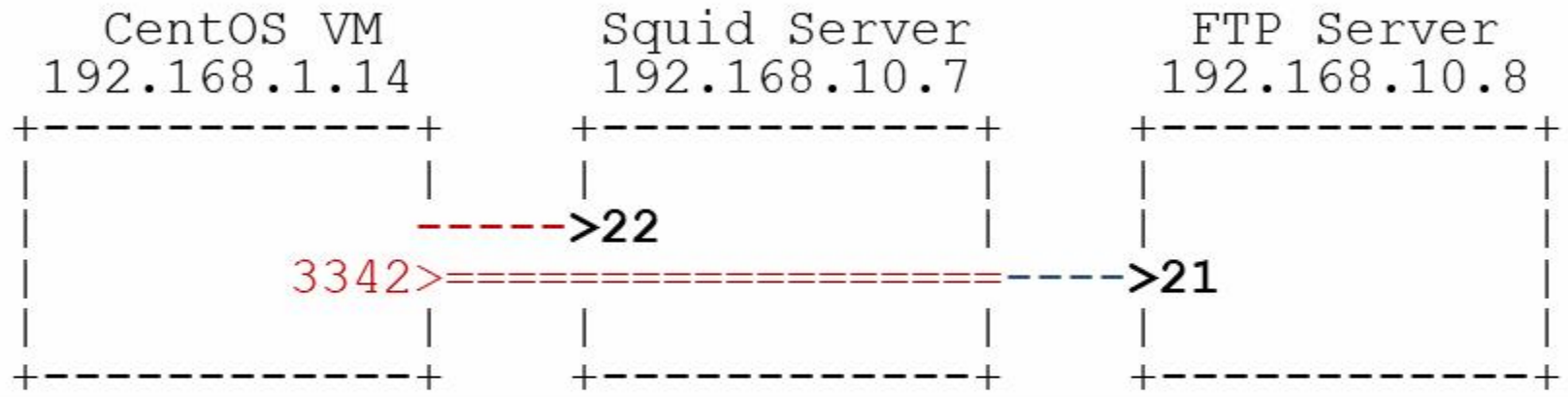
Set up tunnel on CentOS VM:

- ssh root@192.168.10.7 -gL8661:192.168.10.3:3389

Connect to target from Windows XP VM:

- mstsc /v:192.168.1.14:8661
- Log in using credentials for 192.168.10.3 (IIS Server)

FTP Into Remote Machine



Set up tunnel:

- `ssh root@192.168.10.7 -L3342:192.168.10.8:21`

Connect to target via tunnel:

- `ftp 127.0.0.1 3342`

Get an FTP connect but cannot get data back. Why not?

Why Multi-Hop?

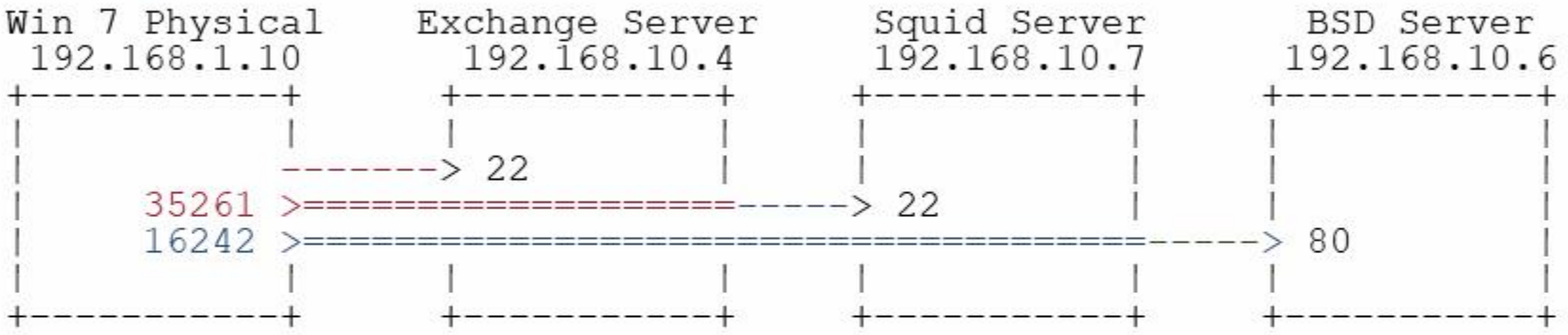
Multiple redirectors:

- Hide your original location better

Multiple targets:

- Ultimate target may be buried within network
- Multiple hops may be required to circumvent filtering and security devices

Using Two Hops



Connection to first redirector and first tunnel:

- `ssh administrator@192.168.10.4 -L35261:192.168.10.7:22`

Connection to second redirector and second tunnel via tunnel:

- `ssh root@127.0.0.1 -p 35261 -L16242:192.168.10.6:80`

Connect to target via tunnels:

- Point web browser at http://127.0.0.1:16242

Public/Private

Often, one set of addresses is used for public access, while another is used for private

- Machines can have multiple network interfaces
- Network address translation (NAT)

Remember who is connecting!

- Addressing is done on a hop-by-hop basis
- If using public addressing to get beyond firewall/NAT, you need private addressing to redirect to hosts in the network

Reverse Tunnels

Why?

Port Forwarding	Exploitation	Evade Filtering Devices
• May want a remote server to access a service	• Many exploits work by calling back to a machine that you control	• Some scenarios allow outbound connections only

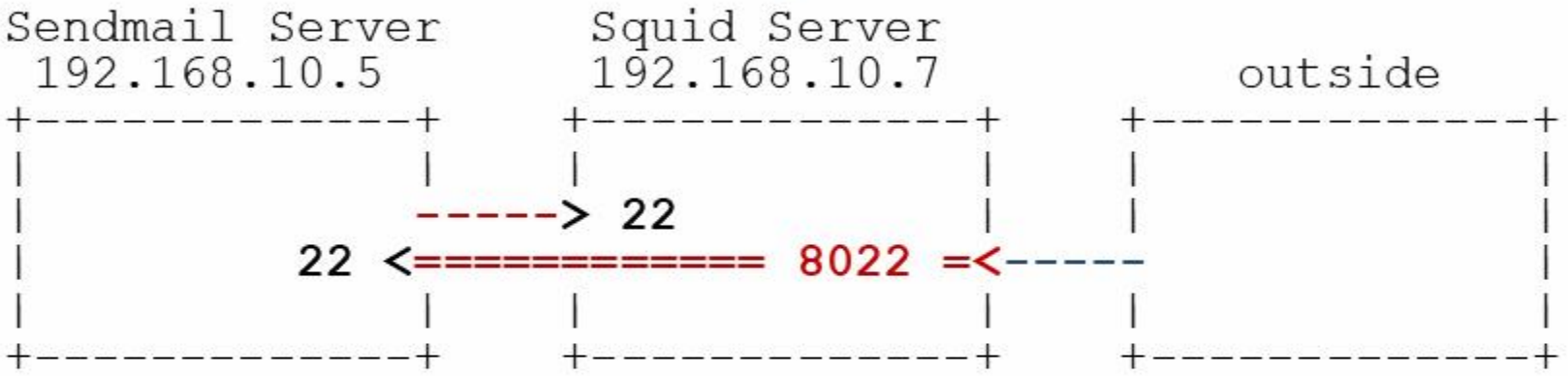
Common thread: The remote end initiates the TCP connection

Reverse Tunnel Syntax

`-R[<l_address>:]<l_port>:<dst_ip>:<dst_port>`

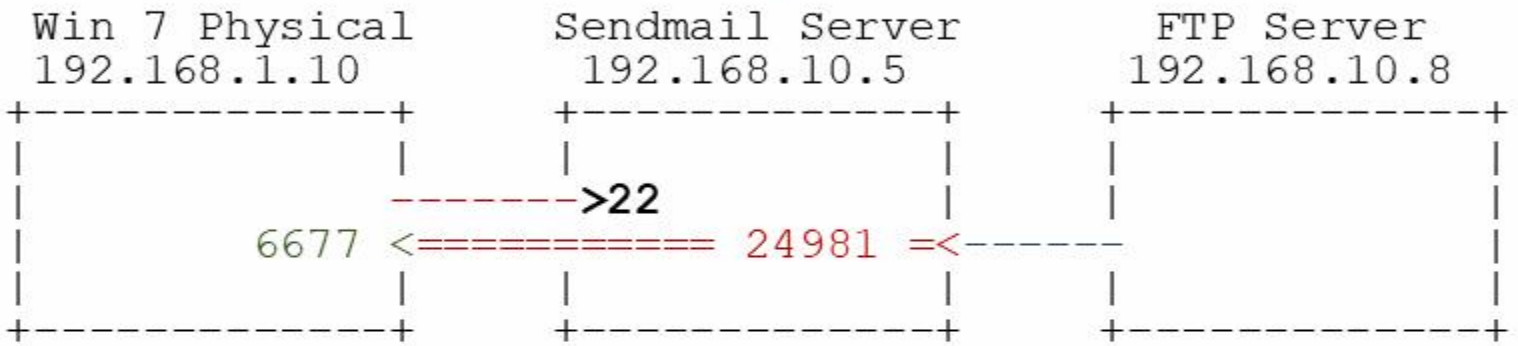
- SSH server opens a socket listener on <l_port> on <l_address>
  - Default address for -R is 0.0.0.0
- Client/server negotiates channel for tunnel
- When some remote machine connects to listener, packets are forwarded to SSH client through tunnel
- SSH client opens connection to <dst_ip> on <dst_port> and forwards packets

Basic Port Forwarding



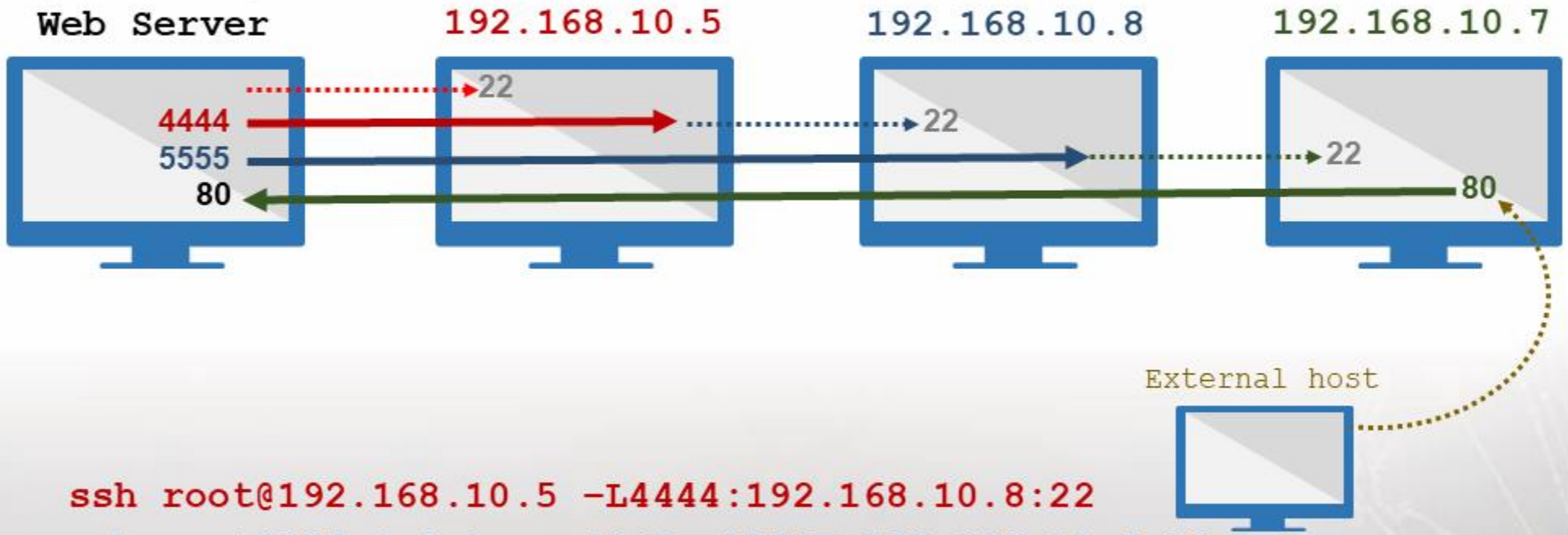
- SSH is already running on your machine
- Set up tunnel:
  - `ssh root@192.168.10.7 -R8022:127.0.0.1:22`
- Connect from outside the network:
  - `ssh root@192.168.10.7 -p 8022`

Reverse Tunnel Diagram



- Set up tunnel on Win 7 physical:
  - `ssh root@192.168.10.5 -R24981:127.0.0.1:6677`
- Set up netcat listener on Win 7 physical:
  - `nc -L -p 6677`
- Connect from remote host on FTP server:
  - `nc 192.168.10.5 24981`

Multiple Hops: Reverse Tunnel




```
ssh root@192.168.10.5 -L4444:192.168.10.8:22
ssh root@127.0.0.1 -p 4444 -L5555:192.168.10.7:22
ssh root@127.0.0.1 -p 5555 -R80:127.0.0.1:80
From Web Browser, go to URL:  http://192.168.10.7
```

Additional Tunnels

- Suppose we want to add a tunnel after we have already set up our infrastructure
  - Closing and reopening = bad OPSEC
- Native ssh command has built-in SSH prompt
  - Entering ~C in an open SSH window gives you a new prompt that allows you to set up tunnels:
    - `[root@localhost ~]#` **<~> + <C>**
    - `ssh>` **-L4444:127.0.0.1:8080**
    - `Forwarding port`
    - … **<CTRL> + <C>**
    - `[root@localhost ~]#`

Exercise: Tunneling and Data Exfiltration

Objectives

After completing this exercise, students will be able to:

- Describe the principles and methods of tunneling network traffic
- Describe the different uses of SSH
- Describe the differences between forward and reverse tunnels when using SSH
- Explain how to redirect traffic using SSH forward and reverse tunnels
- Use SSH to redirect and tunnel network traffic through multiple hosts
- Analyze network tunneling diagrams
- Describe SSH reverse tunnels and their purpose
- Recognize the difference between tunneling and redirecting network traffic
- Implement reverse SSH tunnels
- Analyze traffic to locate covert channels

Duration

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

Exercise: Tunneling and Data Exfiltration

Note:

Server	IP Address
Windows 10	10.10.1.20
CentOS 7	10.10.1.40
Kali	10.10.1.60
Ubuntu	10.10.1.70

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

- From both a defensive and offensive perspective, SSH tunnels provide users with the assurance that their end-to-end communications are secure in a potentially hostile environment
- Attackers use secure tunneling to hide, obscure and redirect their traffic, subverting the security of existing infrastructure
- Network defenders must be familiar with the use and application of secure tunnels to counter network attacks and to detect and trace the origin of intrusions

End of Module 2, Lesson 7