

# CS165 – Computer Security

Network Security

December 1, 2025

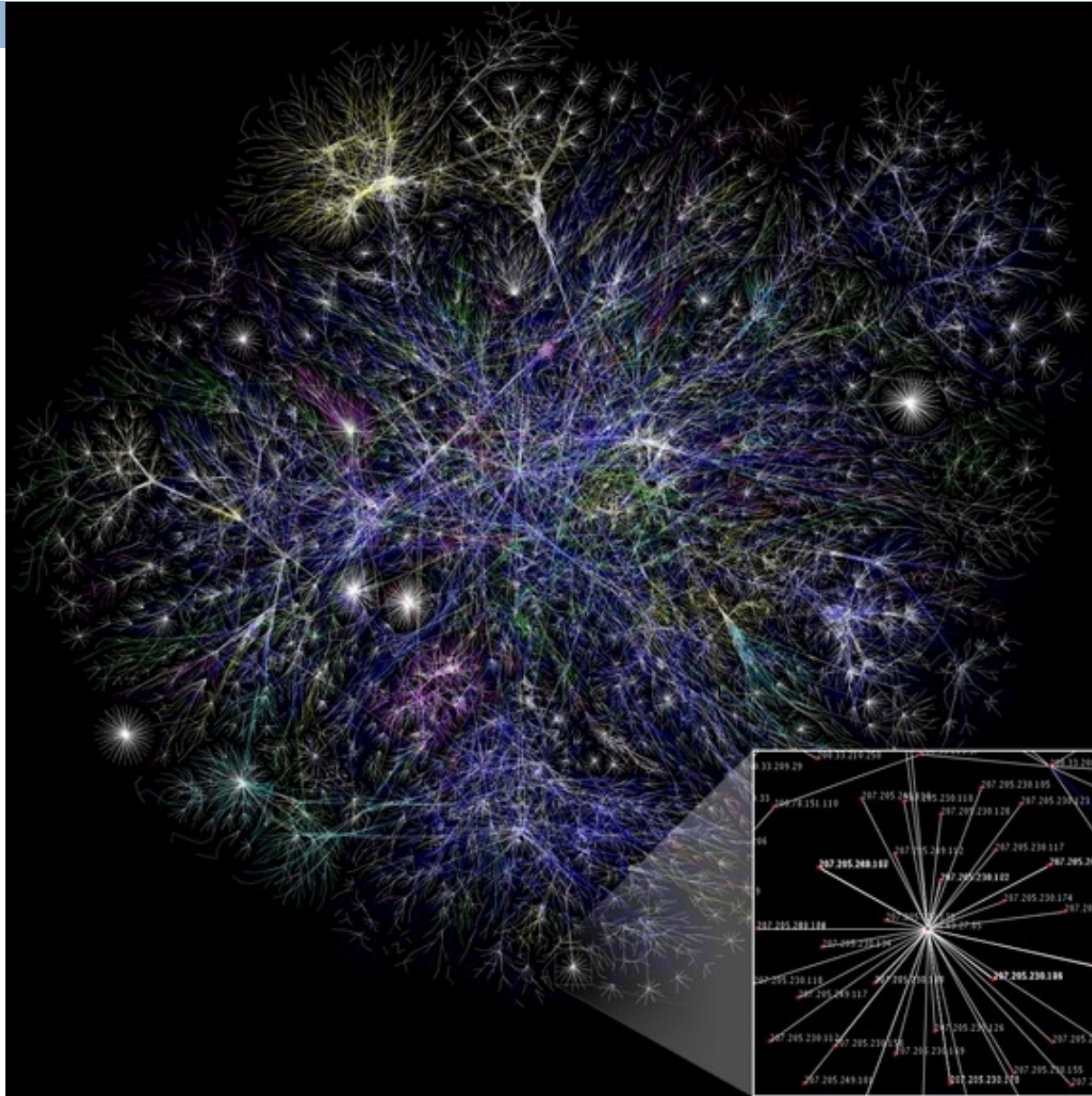
# History of Network Security

2

- Initially built for communication between research institutions
  - ▣ ARPANET (TCP/IP)
  - ▣ First packet sent from UCLA to SRI
- Internet designed without security in mind
  - ▣ Including key protocols such as TCP/IP
  - ▣ Getting it to work is already an amazing job
- Hard to retrofit security into existing protocols
  - ▣ Have to remain backward-compatible
    - E.g., TCP/IP used by every machine now
  - ▣ Solutions often are patches or require an additional layer of indirection

# How the Internet looks

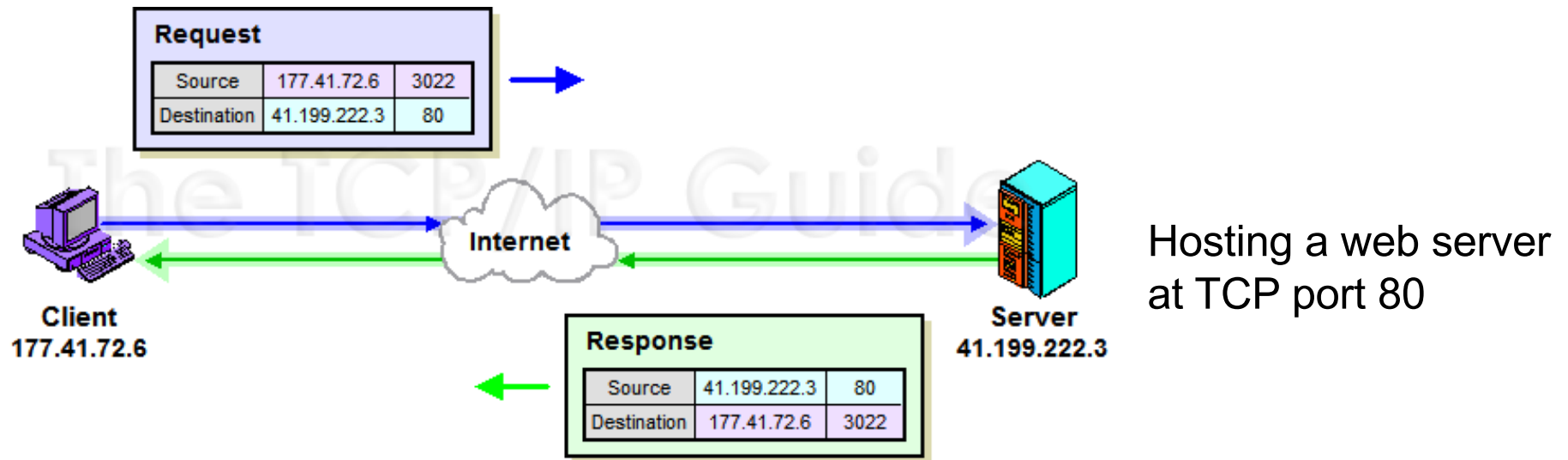
3



# Quick Overview of TCP/IP

4

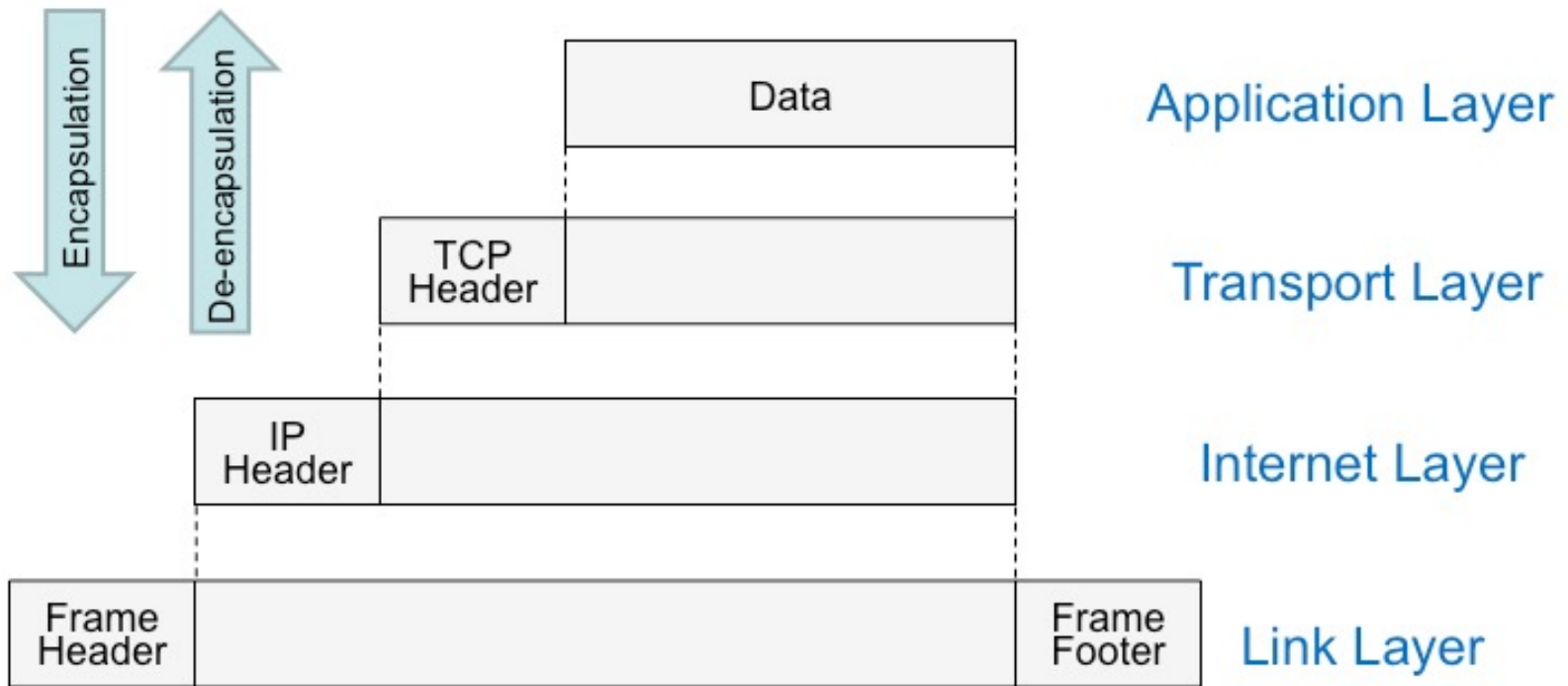
## □ Example:



- Network traffic is broken down into “packets” containing information at 4 main layers

# TCP/IP Network Layers

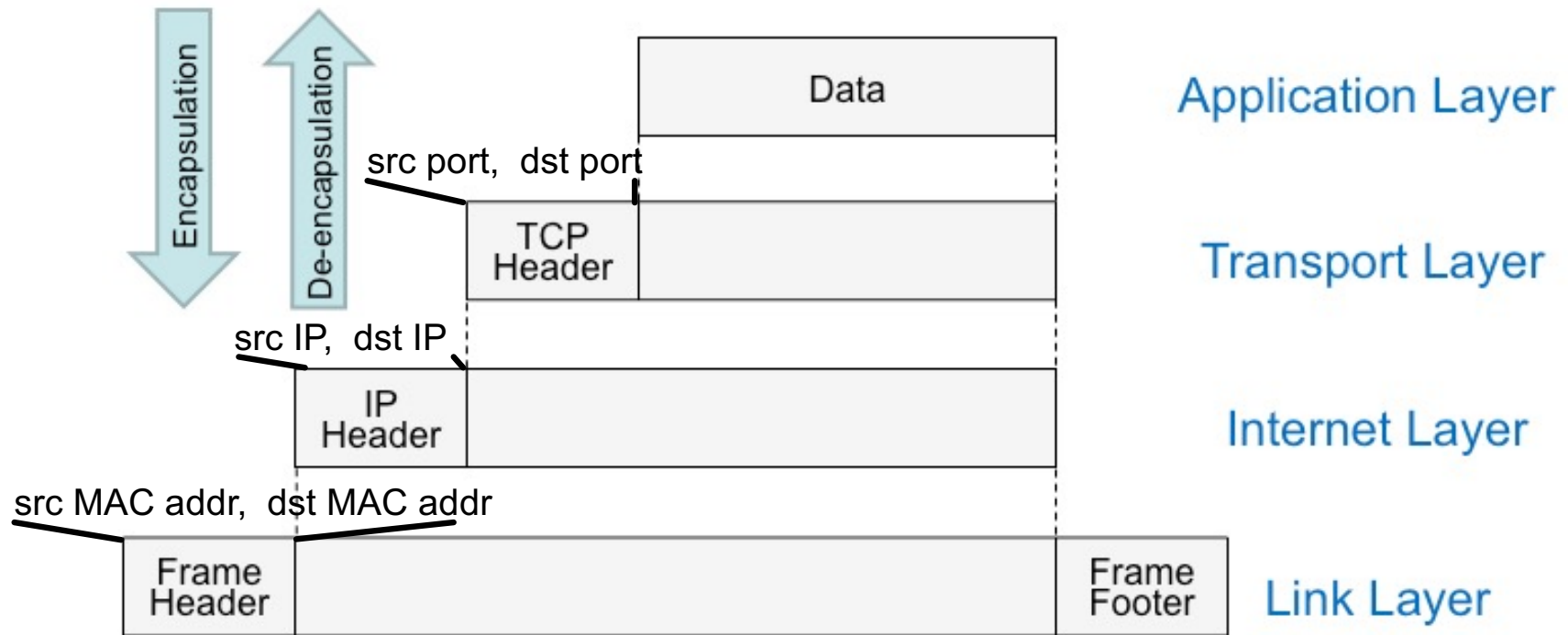
5



Headers at higher layers become data at lower layers

# TCP/IP Network Layers

6



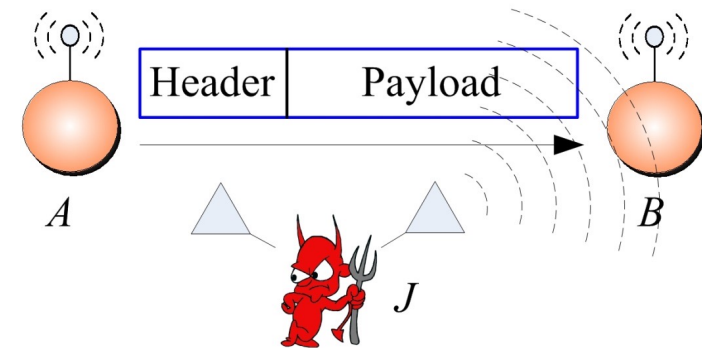
Headers at higher layers become data at lower layers

# Common Threat Models in Networks

8

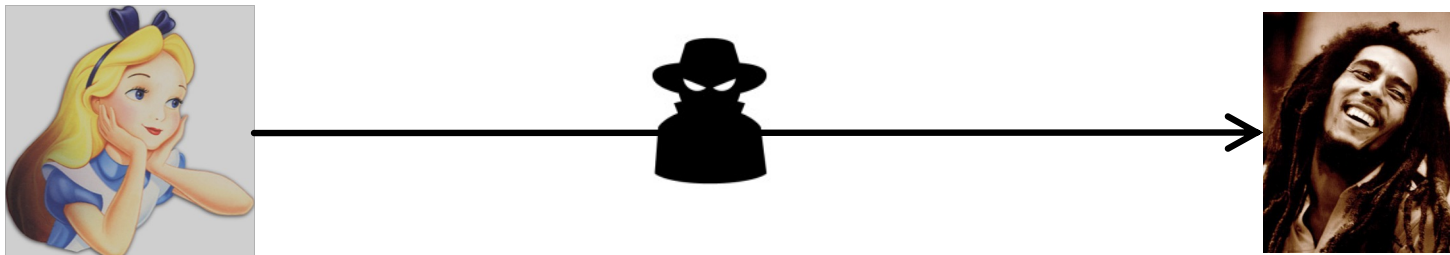
## □ Passive Eavesdropper

- Read

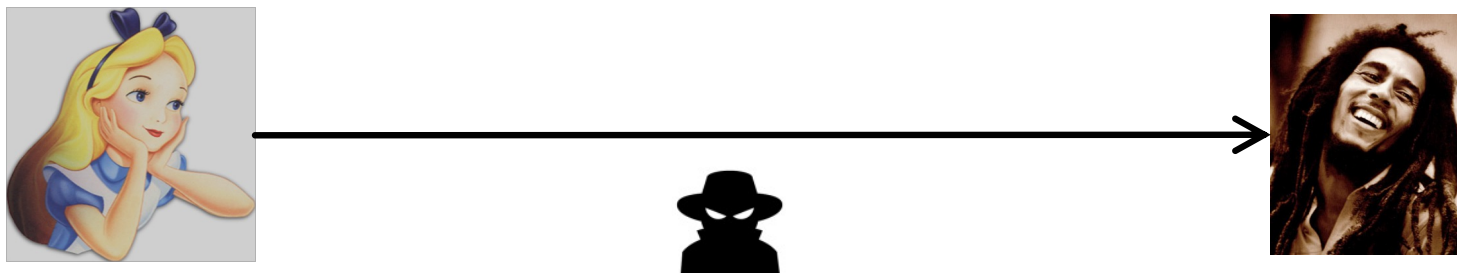


## □ Man-in-the-middle (MITM)

- On the communication path (compromised router)
- Arbitrary Read/Write capability (modify, drop, etc.)



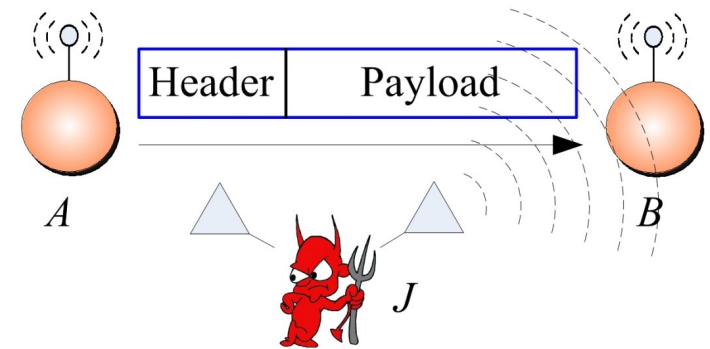
## □ Off-Path attacker (no read capability)





# Threat Model 1: Passive Eavesdropper

## ▣ Read



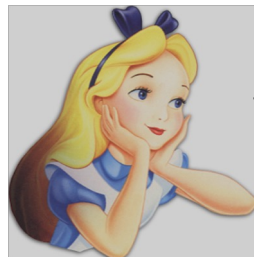
Let's hang out tomorrow night?





# Threat Model 2: Man-in-the-middle

- Read/Write (drop, modify, inject)



Let's hang out tomorrow night?



Bye



# Threat Model 3: Off-Path

## □ Inject only



Let's hang out tomorrow night?

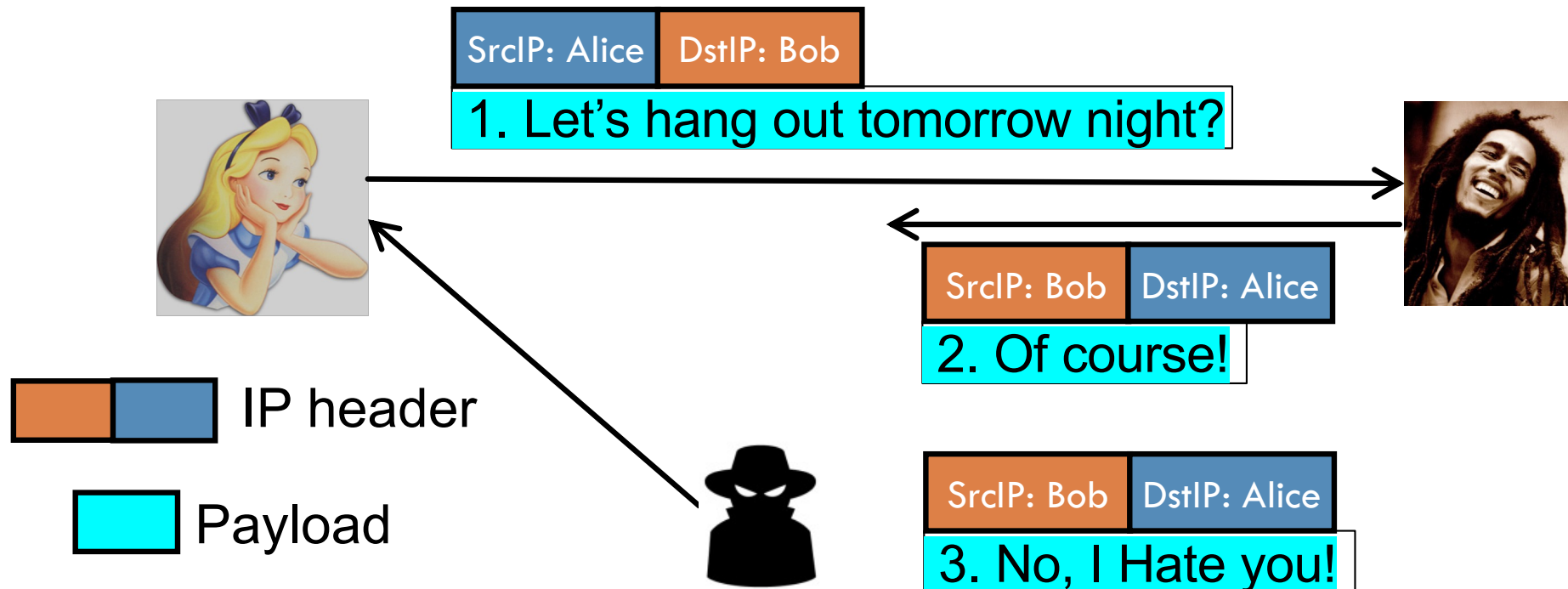


I hate you



# Problem with IP Address (Off-Path Attack)

- Source address in a packet can be filled arbitrarily by a host (think of USPS mail)
  - ▣ Lack of authentication of packet sources
  - ▣ Many vulnerabilities arise because of this



# Back to TCP

- Reliable, *full-duplex, connection-oriented, stream* delivery
  - ▣ Interface presented to the application does not require data in individual packets
  - ▣ Data is guaranteed to arrive and in the correct order without duplicates
    - Or the connection will be dropped
  - ▣ Imposes significant overheads

# Applications of TCP



- Most things!
  - ▣ HTTP, FTP, SMTP...
- Saves applications a lot of work, so used unless there's a good reason not to
  - ▣ QUIC and HTTP 3.0 build on UDP for max performance

# TCP implementation

- Connections are established using a *three-way handshake*
  - ▣ SYN: Client request to setup connection
  - ▣ SYN-ACK: Server responds to client
  - ▣ ACK: Client acknowledges server response
- Data is divided up into packets by the operating system
- Packets are numbered, and received packets are acknowledged
- Connections are explicitly closed
  - ▣ Or may abnormally terminate

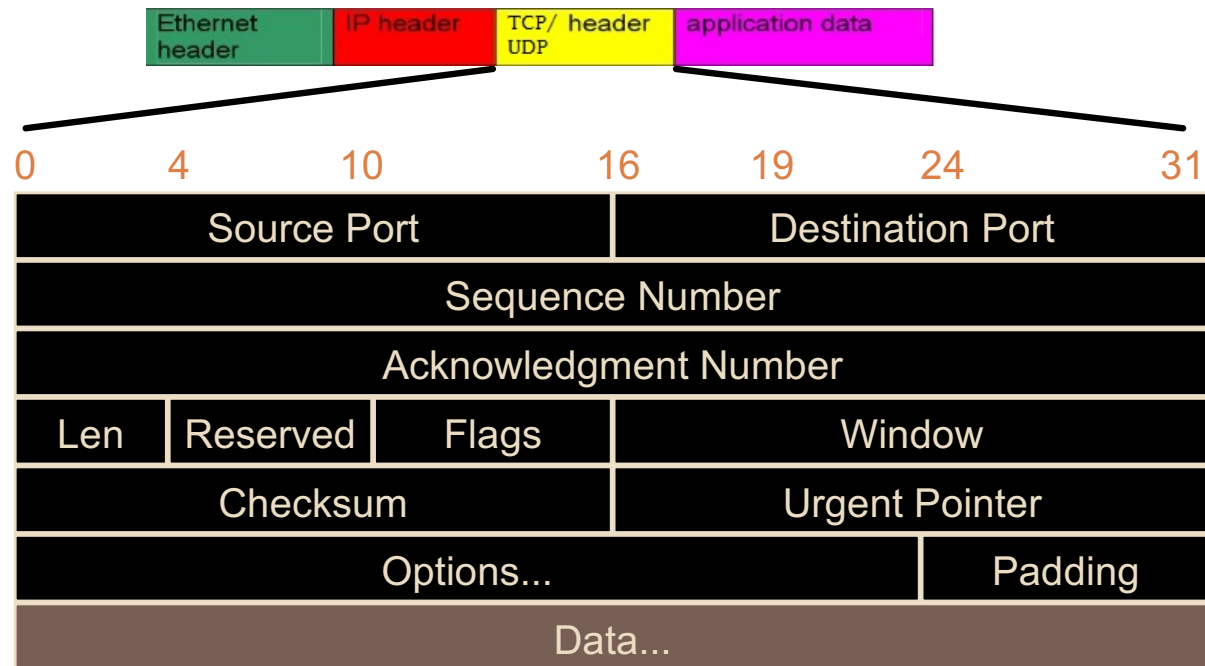
# TCP Packets



- ❑ Source + destination ports
- ❑ Sequence number
- ❑ Acknowledgement number
- ❑ Checksum
- ❑ Various options



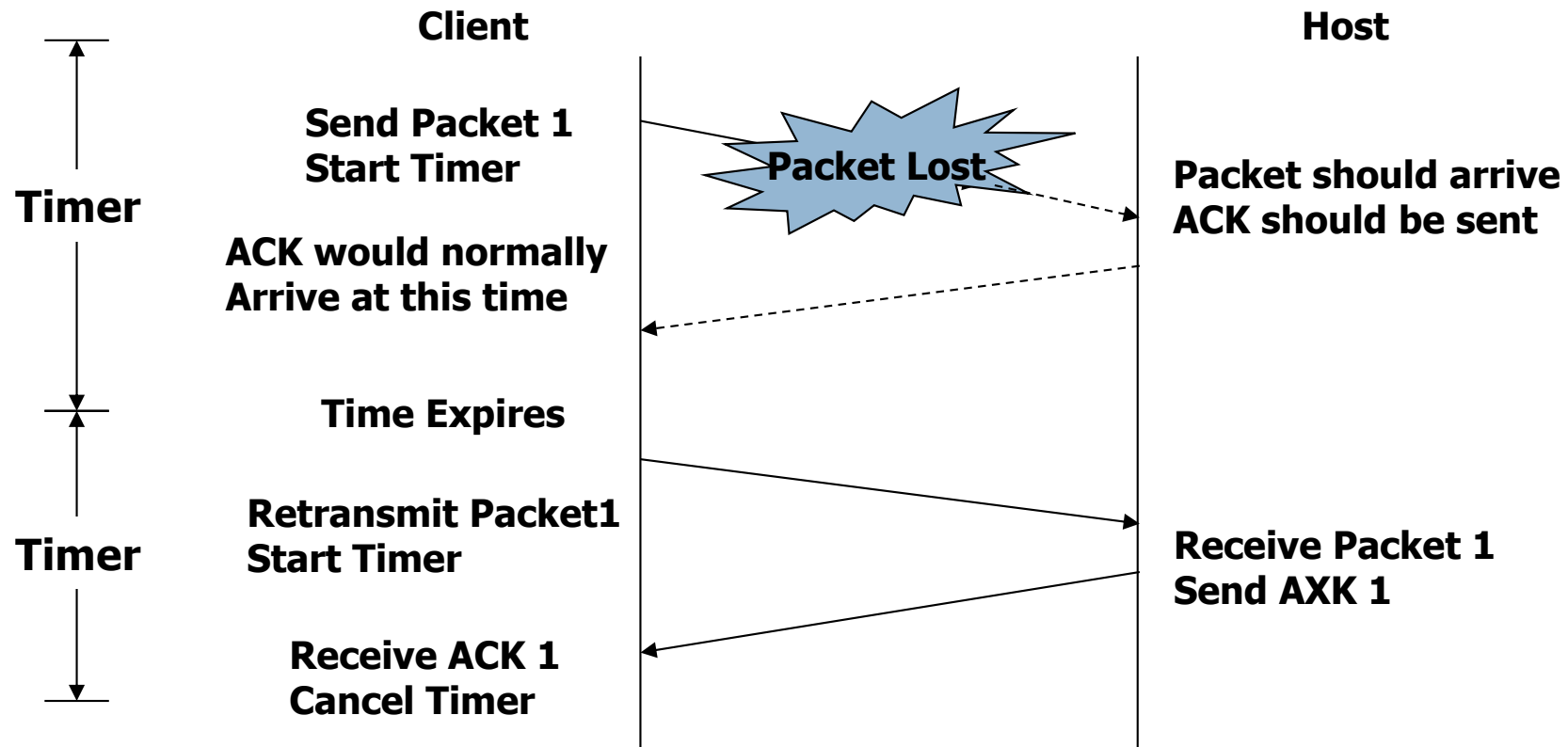
# TCP Segment



Field	Purpose
Source Port	Identifies originating application
Destination Port	Identifies destination application
Sequence Number	Sequence number of first octet in the segment
Acknowledgment #	Sequence number of the next expected octet (if ACK flag set)
Len	Length of TCP header in 4 octet units
Flags	TCP flags: SYN, FIN, RST, PSH, ACK, URG
Window	Number of octets from ACK that sender will accept
Checksum	Checksum of IP pseudo-header + TCP header + data
Urgent Pointer	Pointer to end of "urgent data"
Options	Special TCP options such as MSS and Window Scale

**You just need to know port numbers, seq and ack are added**

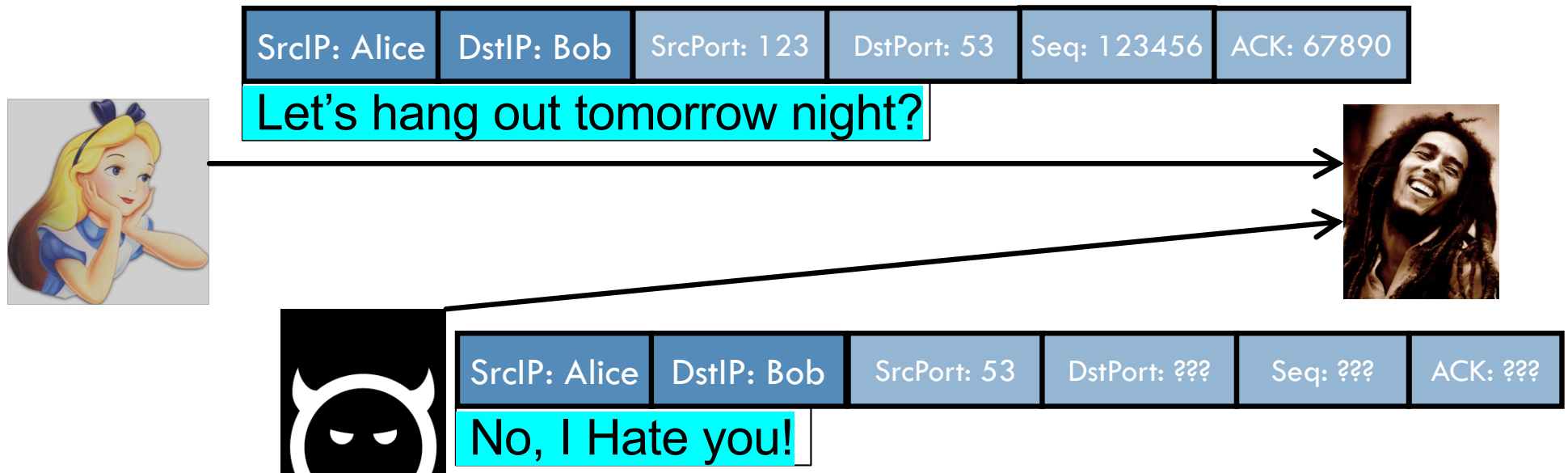
# TCP : Data Transfer






# Off-Path Attack Against TCP

19

- Need to guess the **port number, sequence number, and acknowledgement number!**



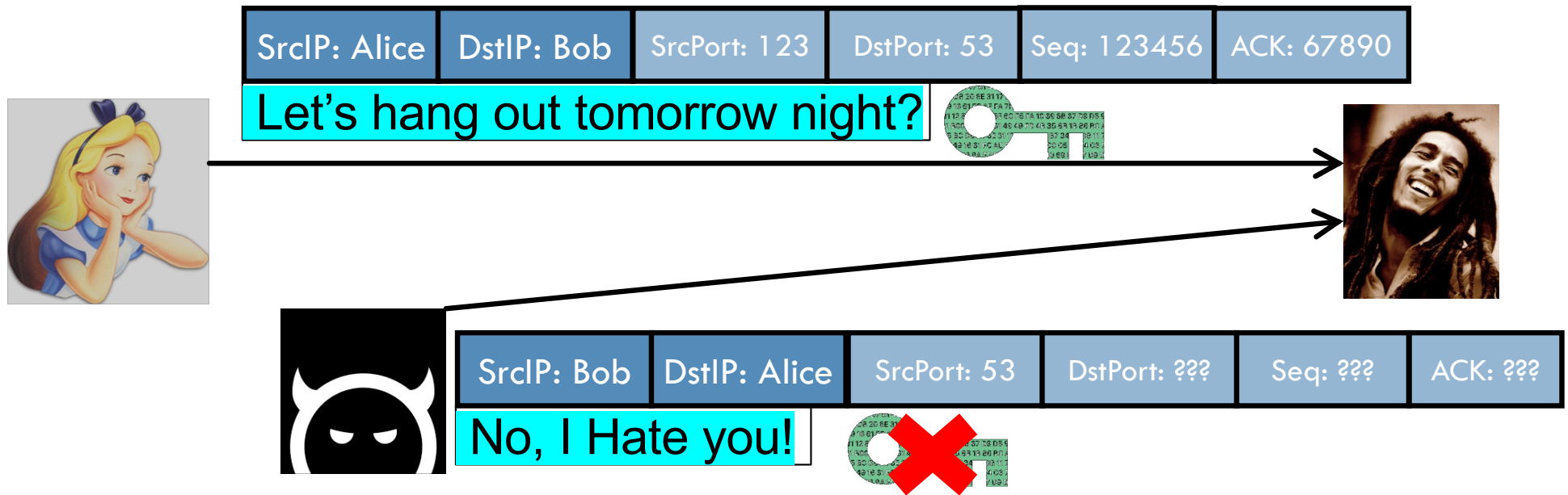
 IP header  
 TCP header  
 Payload

What about eavesdropping and MITM?

# Defenses - encryption

20

- Without the secret key, an attacker cannot read, write, or inject data

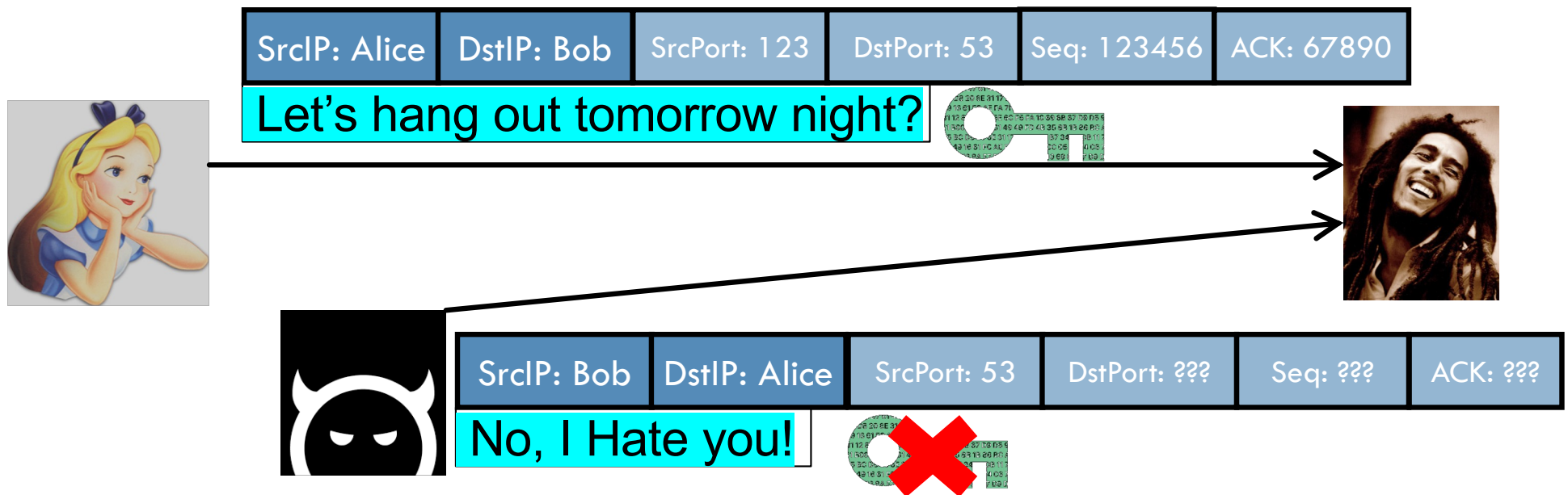


- IP header
- TCP header
- Payload

# Defenses - encryption

21

- Without the secret key, an attacker cannot read, write, or inject data – **replay? Only w/i session**



- IP header
- TCP header
- Payload

# Current state of affairs

22

- **Traffic almost never encrypted**
  - ▣ IP, TCP/UDP, DNS, telnet
- **Traffic encrypted often**
  - ▣ Web traffic (HTTPS, QUIC) > 95% encrypted
  - ▣ Finally, almost all
- **Traffic always encrypted**
  - ▣ SSH, Email traffic (SMTP)
- **Encryption sometimes can be broken too!**

# Finding a way into the network --

## Scanning

24

Host 192.168.2.1 appears to be up.

MAC Address: 00:04:E2:34:B6:CE (SMC Networks)

Host 192.168.2.79 appears to be up.

MAC Address: 00:11:11:5B:7A:CD (Intel)

Host 192.168.2.82 appears to be up.

MAC Address: 00:10:5A:0D:F6:D7 (3com)

Host 192.168.2.198 appears to be up.

MAC Address: 00:10:DC:55:89:27 (Micro-star International)

Host 192.168.2.199 appears to be up.

MAC Address: 00:C0:4F:36:33:91 (Dell Computer)

Host 192.168.2.200 appears to be up.

MAC Address: 00:0C:41:22:CC:01 (The Linksys Group)

Host 192.168.2.251 appears to be up.

MAC Address: 00:0F:66:75:3D:75 (Cisco-Linksys)



# Does That Matter?

26

- If they identify a service that has a known vulnerability (e.g., buffer overflow), they can launch the corresponding exploit

```
$ nmap -Pn www.cs.ucr.edu
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-11-17 20:03 UTC
```

```
Nmap scan report for www.cs.ucr.edu  
(169.235.30.15)
```

```
Host is up (0.00033s latency).
```

```
rDNS record for 169.235.30.15: thoth.cs.ucr.edu
```

```
Not shown: 996 closed ports
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
80/tcp    open  http
```

```
111/tcp   open  rpcbind
```

```
5666/tcp  open  nrpe
```

# Firewalls



27

- Basic problem – many network applications and protocols have security problems that are fixed over time (i.e., may not be fixed yet)
  - ▣ Difficult for users to keep up with patches to keep hosts secure
  - ▣ Solution
    - Administrators limit access to end hosts by using a firewall
    - Firewall is kept up-to-date by administrators
- **Access control** over network communications with hosts and their services

# Firewalls



28

- A firewall is like a castle with a drawbridge
  - ▣ Only one point of access into the network
    - Need to ensure **complete mediation**
  - ▣ This can be good or bad
- Can be hardware or software
  - ▣ E.g., Some routers come with firewall functionality
  - ▣ ipfw, iptables, pf on Unix systems, Windows XP and Mac OS X have built in firewalls

# Firewalls



29

- Used to filter packets based on a combination of features
  - ▣ These are called **packet filtering firewalls**
    - There are other types too, but they will not be discussed
  - ▣ E.g., Drop packets with destination port of 23 (Telnet)
  - ▣ Can use any combination of IP/UDP/TCP header information
- But why don't we just turn Telnet off?

# Firewalls



30

- Used to filter packets based on a combination of features
  - ▣ These are called **packet filtering firewalls**
    - There are other types too, but they will not be discussed
- Consist of a rule base
  - ▣ Rules are **checked in sequence**
  - ▣ **First one to match** is determines the response
  - ▣ One rule may **match many packets due to wildcards**
    - E.g., Block all telnet (by port number 23) to your network (1.1.1.\*) – all hosts

# Firewall Rules



32

- Specifies what traffic is (not) allowed
  - Maps attributes to address and ports
  - Example: HTTP should be allowed to any external host, but inbound only to web-server

Source		Destination		Protocol	Flags	Actions
Address	Port	Address	Port			
*	*	1.1.1.1	80	TCP	SYN	Accept
1.1.1.*	*	*	80	TCP	SYN	Accept
*	*	*	80	TCP		Accept
*	*	*	*	TCP		Deny

# Firewall Rules



33

- Specifies what traffic is (not) allowed
  - Maps attributes to address and ports
  - Example: HTTP should be allowed to any external host, but inbound only to web-server

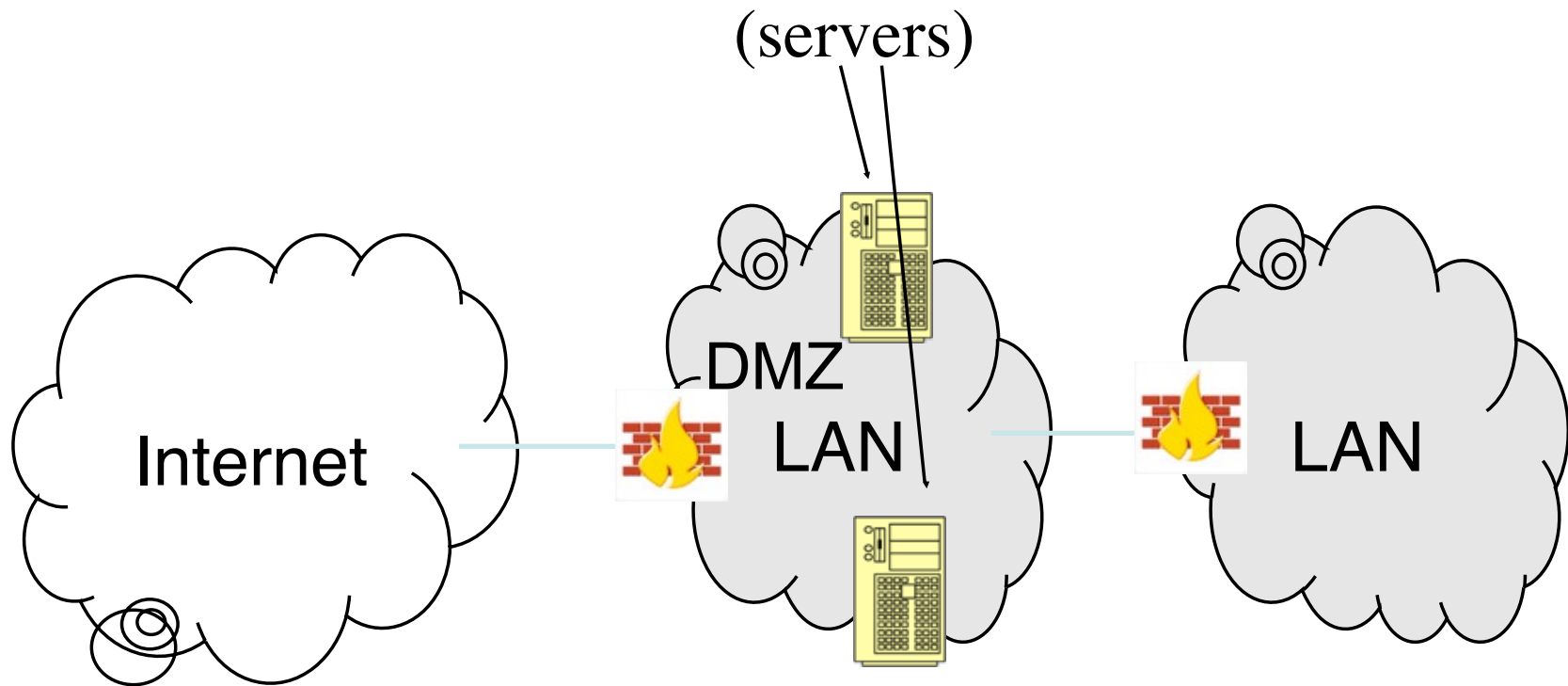
Source		Destination		Protocol	Flags	Actions
Address	Port	Address	Port			
*	*	1.1.1.1	80	TCP	SYN	Accept
1.1.1.*	*	*	80	TCP	SYN	Accept
*	*	*	80	TCP	SYN	Deny
*	*	*	*	TCP		Deny



# De-Militarized Zone (DMZ)



34



- Zone between LAN and Internet (*public facing*)

# Stateful, Proxy, and Transparent



36

- Single packet may not contain sufficient data to make an access control decision
- **Stateful**: allows historical context consideration
  - ▣ Firewall collects data over time
  - ▣ e.g., TCP packet is part of established session
- Firewalls may affect network traffic
  - ▣ **Proxy**: Application firewall
    - Receives, interprets, and reinitiates communication
  - ▣ **Transparent**: Network firewall
    - Invisible to communication, like a router
- Transparent good for speed, and proxies good for complex state

# Real Firewall: Linux iptables



37

- The **iptables** firewall looks in the firewall **table** to seek if a **rule** in the current **chain** **matches** a packet – executes the rule's **target** if it does.
- **Table**: all the firewall rules, grouped in chains
- **Chain**: one list of rules
  - ▣ A rule can initiate a new chain (like a function call)
- **Rule**: description of packets with a target
- **Match**: when all a rule's fields match the packet
- **Target**: operation to execute on a packet given a match



# iptables Simple Test

39

- Use **loopback** to test the rules locally on your machine using ICMP
  - ▣ IP address 127.0.0.1
- ICMP protocol
  - ▣ Submit “**ping**” requests to 127.0.0.1 (next slide)
- TCP protocol (use “**netcat**” – **nc**)
  - ▣ submit requests to 127.0.0.1 at specific port
  - ▣ `nc -l -p 3750` (server)
    - Listen at port 3750
  - ▣ `nc -p 3000 localhost 3750`
    - Send from port 3000 to localhost at port 3750

# iptables Simple Test



40

- Run "ping" on a Linux machine – sending to "loopback"
  - ▣ `ping -c 1 127.0.0.1`
- Add iptables rule to block this ping (must be "root")
  - ▣ `iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP`
    - `-A INPUT` – Add to the INPUT chain
    - `-s 127.0.0.1` – Source address (loopback)
    - `-p icmp` – for the ICMP protocol (used by "ping")
    - `-j DROP` – target is to drop the packet
- Run ping again
  - ▣ Should be blocked



# iptables Simple Test

41

- Run "ping" on a Linux machine – sending to "localhost"
  - ▣ `ping -c 1 127.0.0.1`
- Add iptables rule to block this ping
  - ▣ `iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP`
- Run ping again
- Delete the rule (one of)
  - ▣ `iptables -D INPUT 1`
  - ▣ `iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP`
  - ▣ `iptables -F INPUT`

# iptables Targets



42

- Define what to do with the packet on match
- ACCEPT/DROP
- QUEUE for user-space application
- LOG any packet that matches
- REJECT drops and returns error packet
- RETURN enables packet to return to previous chain
- <user-specified> passes packet to that chain



# iptables Examples

43

```
iptables -A INPUT -s 200.200.200.2 -j ACCEPT
```

- Accept packets from source 200.200.200.2

```
iptables -A INPUT -s 200.200.200.1 -j DROP
```

- Drop packets from source 200.200.200.1

```
iptables -A INPUT -s 200.200.200.1 -p tcp -j DROP
```

- Drop TCP packets (only) from source 200.200.200.1

```
iptables -A INPUT -s 200.200.200.1 -p tcp --dport telnet -j DROP
```

- Drop packet from that source destined for the telnet port

```
iptables -A INPUT -p tcp --destination-port telnet -i ppp0 -j DROP
```

- Drop TCP packet destined for the telnet port using network interface ppp0



# Conclusions

44

- Network communication is fundamentally useful
  - ▣ But, threatened by **eavesdroppers** and **MITM attacks**
- **TCP** is the common communication protocol
  - ▣ Possible to **forge messages** unless encryption is used
- Networks of hosts have **many network services**
  - ▣ Early worms exploited the easy access to services
- Modern networks employ **firewalls** to access control over packet send/rcv to hosts, services (ports), protocols, etc.
  - ▣ **Linux iptables** is one example

# Questions

45

