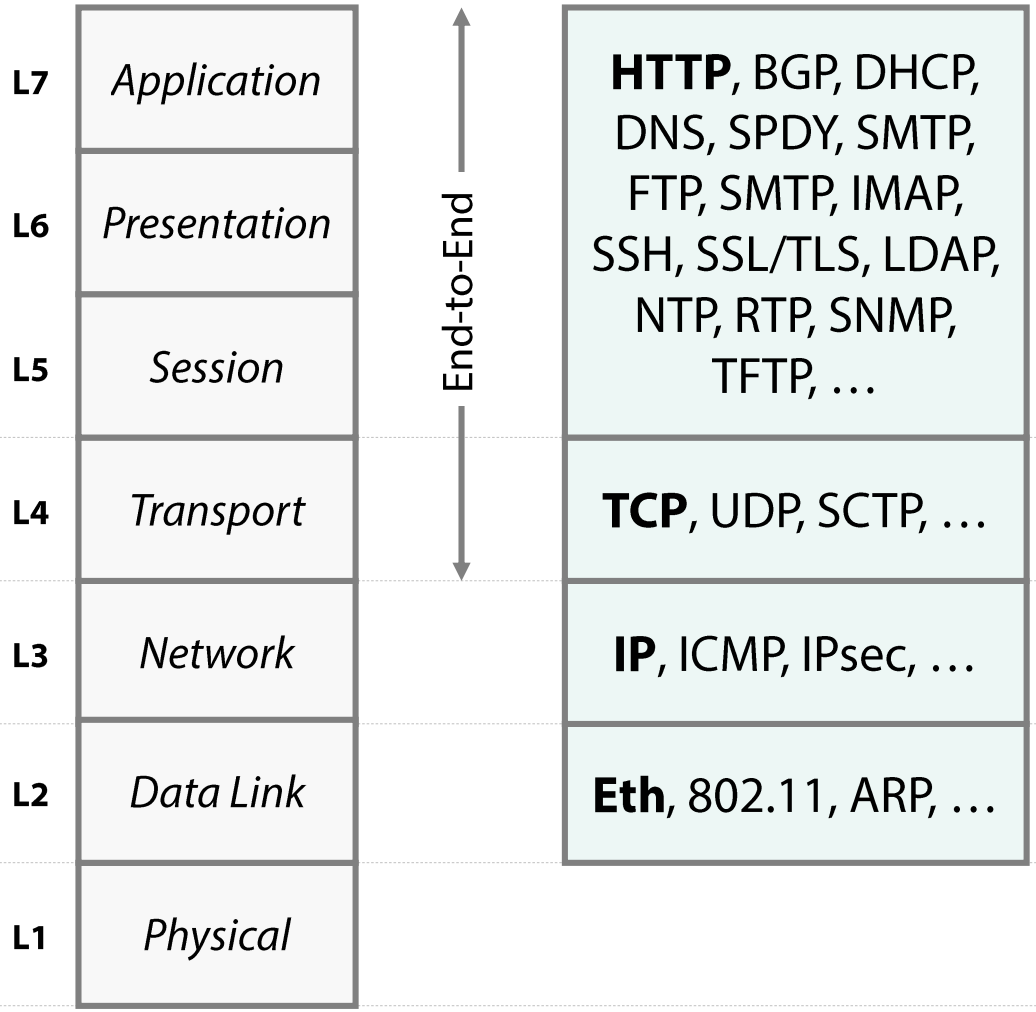
CSE508 Network Security

2/1/2016 **Lower Layers**

Michalis Polychronakis

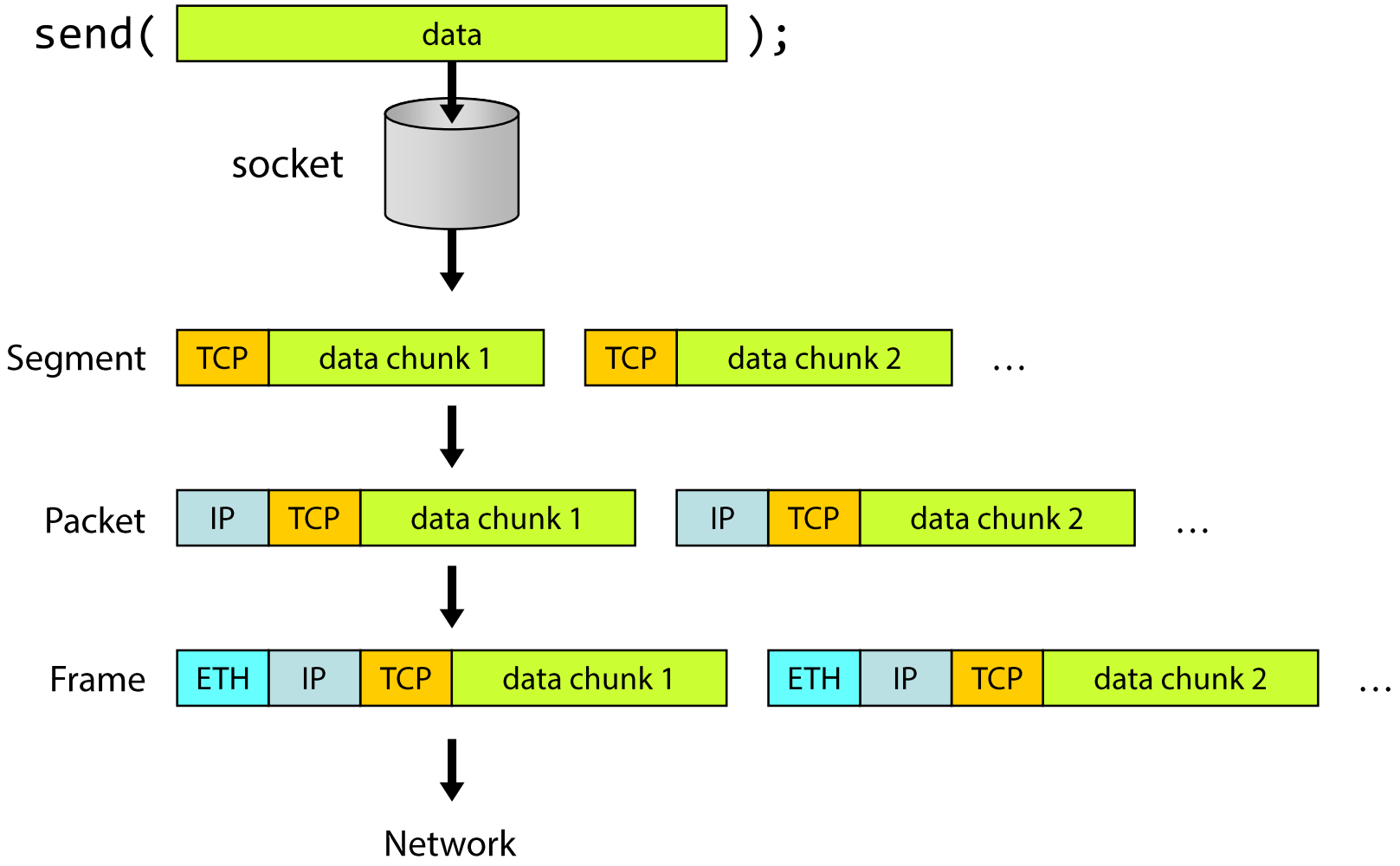
Stony Brook University

Basic Internet Protocols (OSI Model vs. Reality)



	Deliver to:	**Based on:**
L4	Dst. application	Port
L3	Dst. machine	IP
L2	Next hop	MAC
L1	Wire/air/pigeon	NIC

Streams vs. Packets



Active vs. Passive Attacks

Passive: the attacker eavesdrops but does not modify the message stream in any way

Traffic snooping, wiretapping, passive reconnaissance, listening for unsolicited/broadcast traffic, traffic analysis, …

Active: the attacker may transmit messages, replay old messages, modify messages in transit, or drop selected messages from the wire

Spoofing, session replay, data injection/manipulation (man-in-the-middle), DoS, malicious requests/responses, …

Physical Layer Attacks

Network eavesdropping

- NIC in promiscuous mode captures all traffic

Wiretapping (wire, optical fiber)

- Not needed for WiFi networks! ➔ WPA

Wirecutting

Jamming

Electronic emanations/side channels

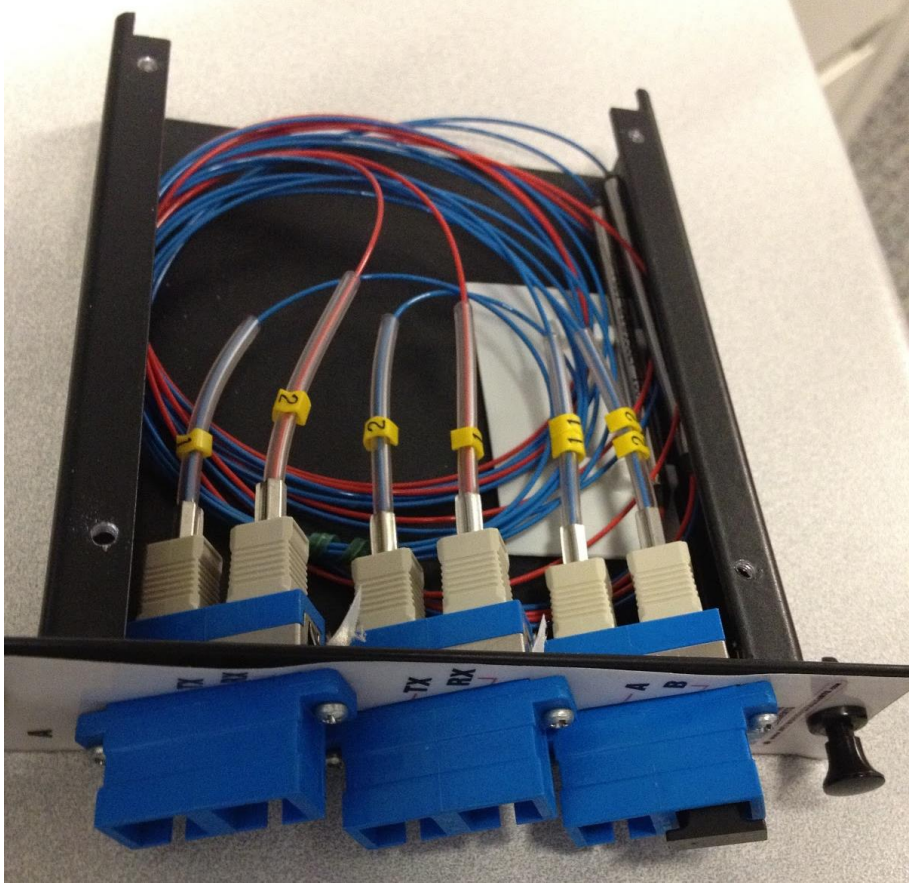
Tracking

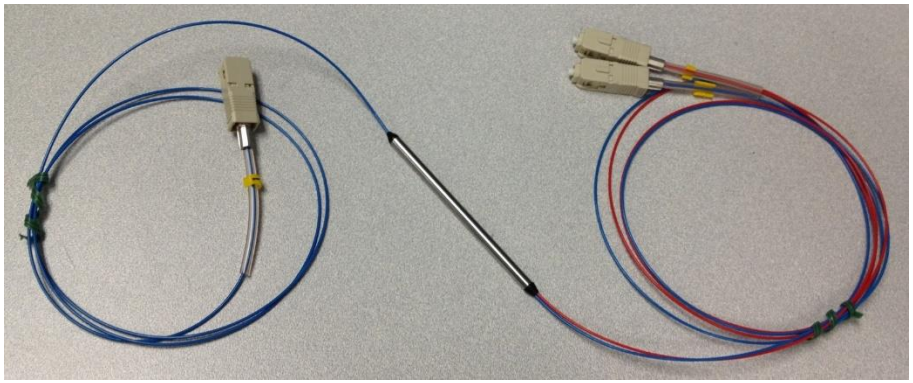
- Device fingerprinting
- Location tracking (cellular, WiFi)
- Many techniques of varying precision: trilateration/triangulation, nearest sensor, received signal strength, …

Network Taps

Up to 100Mbit/s can be completely passive

Most high-end switches can also mirror traffic

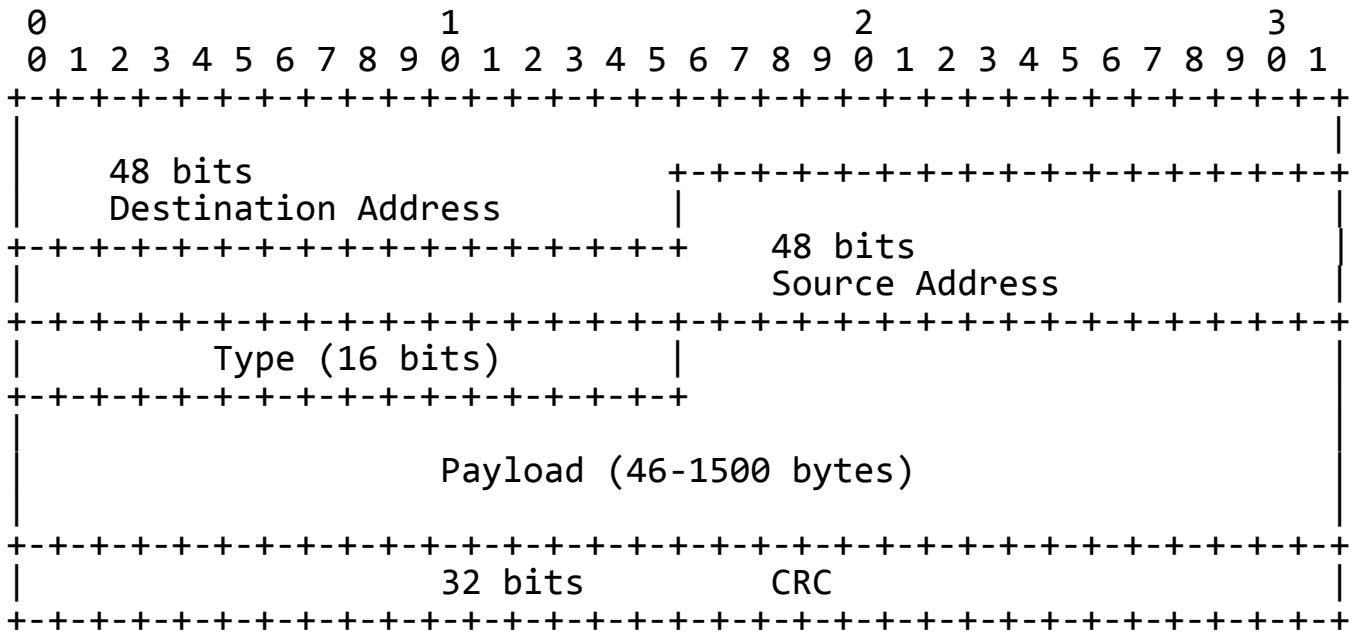




Ethernet

Most commonly used data link layer protocol for LANs

Communication based on *frames*



Link Layer Attacks

Eavesdropping

- WiFi: shared medium ➔ *trivial*
- Hub: broadcasts packets to all ports ➔ *trivial*
- Switch: learns which device is connected to which port and forwards packets only to the appropriate port ➔ *still possible!*
  ARP cache poisoning, CAM table exhaustion

Spoofing

- Impersonate another machine and receive its traffic
- Bypass address-based authentication
- Change MAC address to get 30' more of free WiFi
- Hide the device's vendor (first three bytes of MAC address)

DoS: flooding, deauth (WiFi), … (future lecture)

Rogue access point

Address Resolution Protocol (ARP)

Allows mapping of IP addresses to physical addresses

A new machine joins a LAN; How can it find the MAC addresses of a neighbor machine?

ARP request (broadcast): *Who has IP 192.168.0.1?*

ARP reply by 192.168.0.1: *Hey, here I am, this is my MAC address*

Each host maintains a local ARP cache

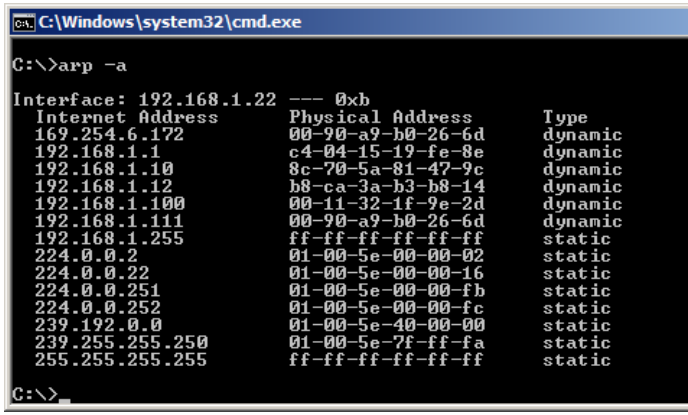
Send request only if local table lookup fails

ARP announcements (*gratuitous ARP*)

Voluntarily announce address updates

(NIC change, load balancing/failover, …)

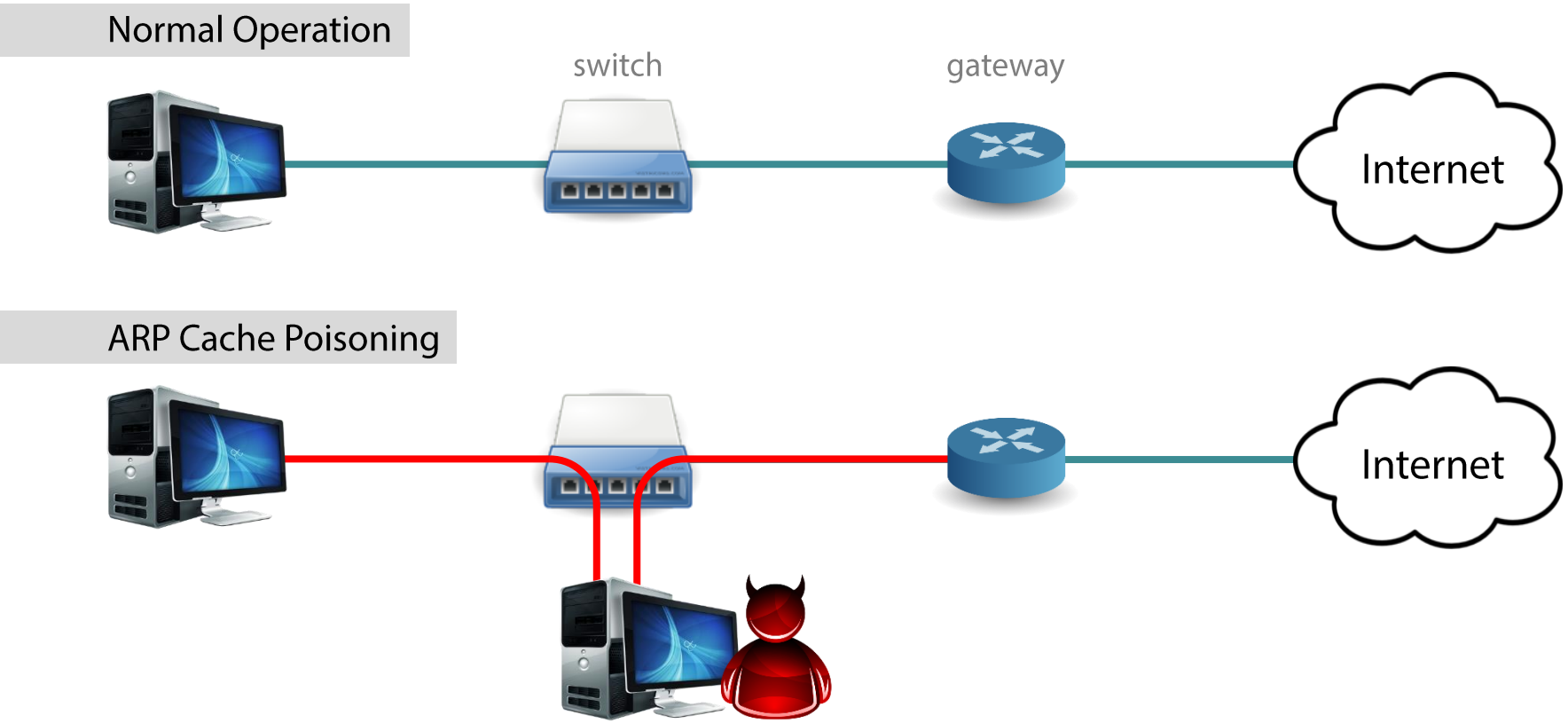
Can be abused…



ARP Cache Poisoning

ARP replies can be **spoofed:** IP to MAC mapping is not authenticated!

Enables traffic sniffing/manipulation through MitM



ARP Cache Poisoning

Attack steps

- ARP reply to victim, mapping gateway's IP to attacker's MAC
- ARP reply to gateway, mapping victim's IP to attacker's MAC
- Just forward packets back and forth

Tools: `arpspoof` (`sslstrip`), `ettercap`, `nemesis`, …

Various Defenses

- Static ARP entries: ignore ARP reply packets
- OS configuration: ignore unsolicited replies, …
- ARPwatch and other detection tools
- Managed switches

CAM Table Exhaustion

Switches use Content Addressable Memory (CAM) to keep MAC address to port mappings

- Finite resource!

Flooding a switch with a large number of randomly generated MAC addresses can fill up the CAM table

- Failsafe operation: send all packets to all ports
- Essentially the switch turns into a hub ➔ eavesdropping!
- Noisy attack, can be easily detected

Tool: `macof` (part of `dsniff`)

Rogue Access Points

No authentication of the AP to the client

Set up fake access point with an existing SSID or just an enticing name

- Starbucks-FREE-WiFi
- “Auto-connect”/”Ask to join network” mobile phone features greatly facilitate this kind of attacks
- Pineapple, Power Pwn, …

Wireless backdoor

- Ship an iPhone/special purpose device to an office and use 4G connection for C&C
- Hide a tiny AP in a wall plug etc.

Detection

- NetStumbler: show all WiFi networks
- RF monitoring systems
- Wireless IDS/IPS

Internet Protocol (IP)

Routing: deliver packets from a source to a destination based on the destination IP address

- Through several hops (routers) – see `traceroute`
- Connectionless, best effort: no ordering or delivery guarantees
- Source IP address is not authenticated ➔ can be easily spoofed!

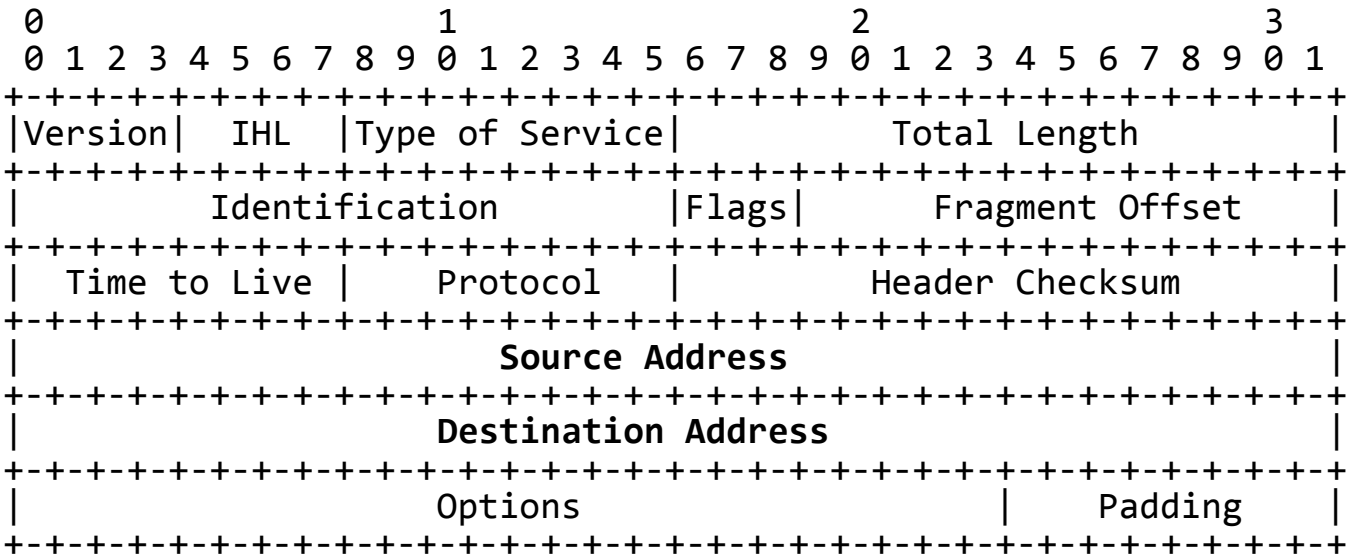
IPv6: most recent version, uses 128-bit addresses

- IPv4 space has been exhausted
- IPv6 deployment is slow

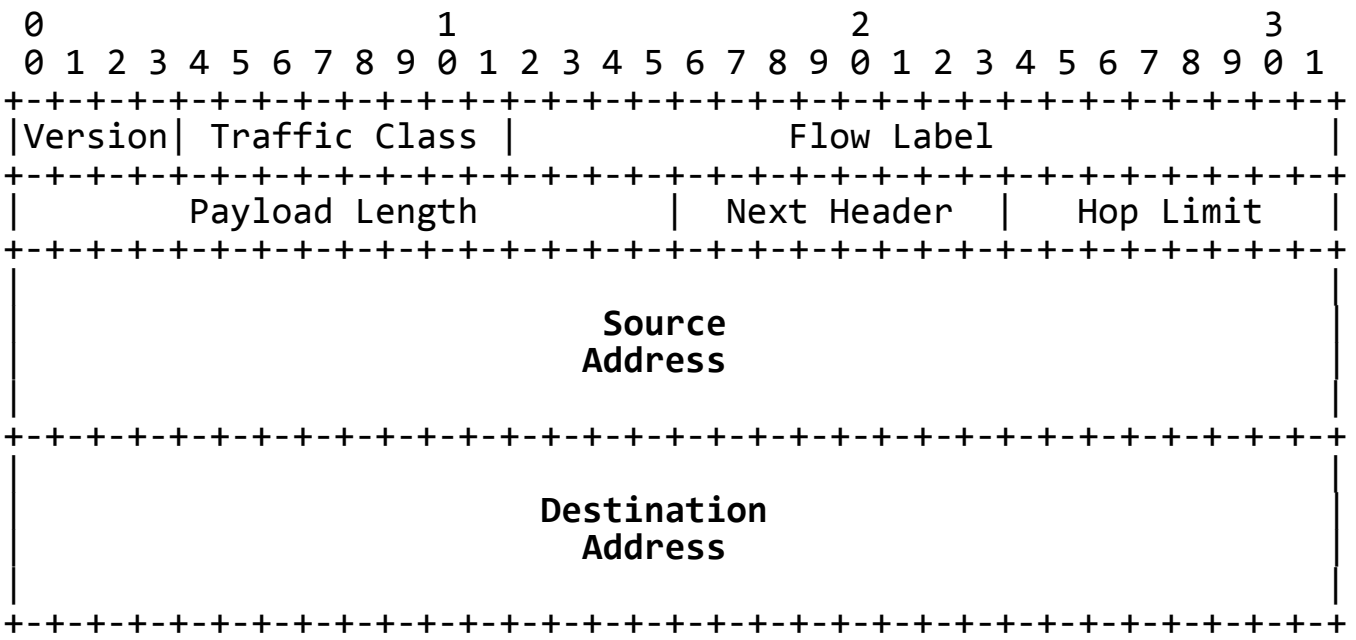
If a packet is too large for the next hop, it can be *fragmented* into smaller ones

- Maximum transmission unit (MTU)

IPv4


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding

IPv6


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address

Network Layer Attacks

ICMP (Internet Control Message Protocol): Used to exchange error messages about IP datagram delivery

- Smurf Attack (DoS with spoofed broadcast Echo request)
- Reconnaissance
- Exfiltration using ICMP Tunneling
- ICMP redirect MitM
- Organizations typically block incoming/outgoing ICMP traffic

IP spoofing: conceal the real IP address of the sender

- Mostly used in DDoS attacks
- Ingress and egress filtering limit its applicability

IP fragmentation: confuse packet filters and intrusion detection systems

- Split important information across two or more packets

Transmission Control Protocol (TCP)

Provides *reliable* virtual circuits to user processes

- Connection-oriented, reliable transmission
- Packets are shuffled around, retransmitted, and reassembled to match the original data ***stream***

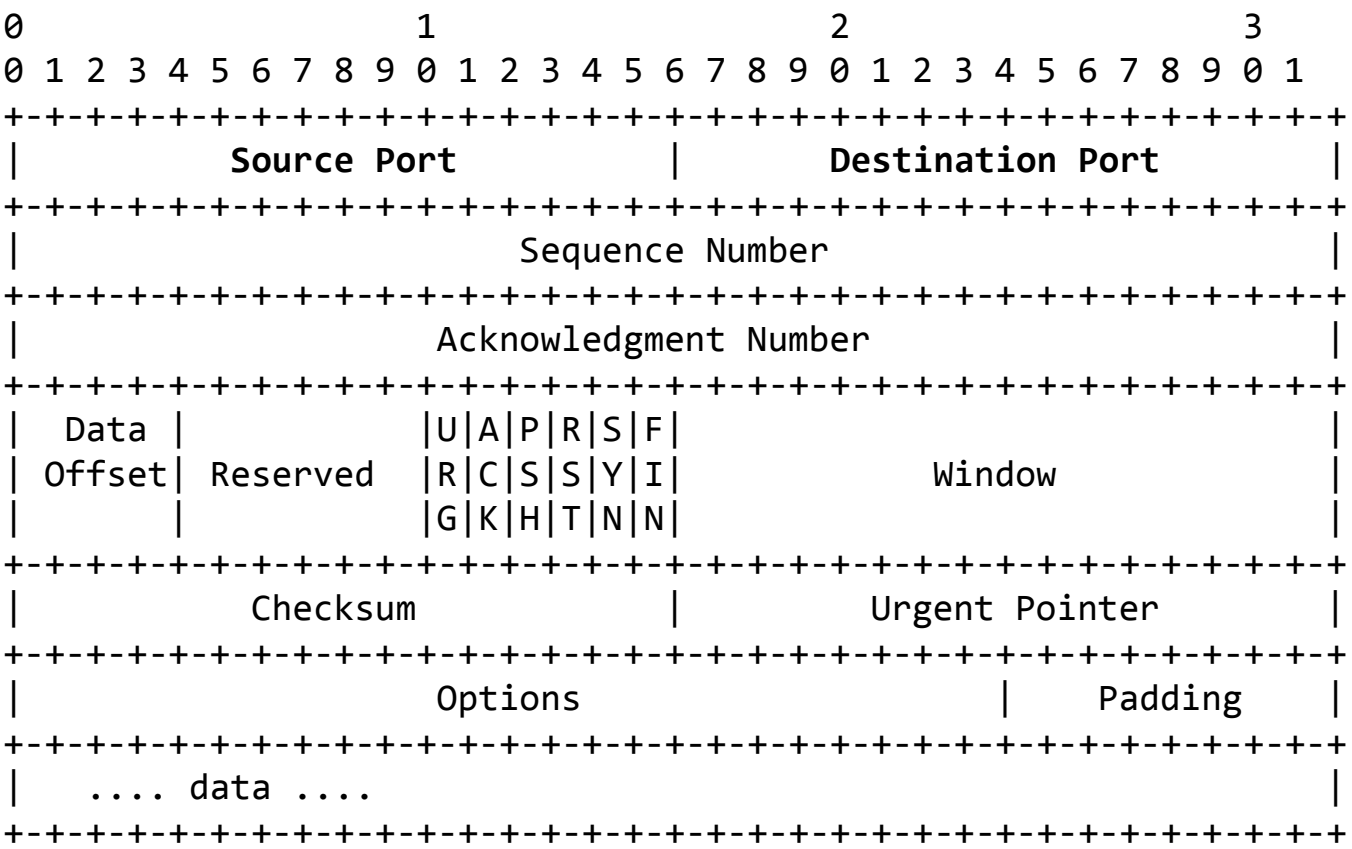
Sender: breaks data stream into packets

- Attaches a sequence number on each packet

Receiver: reassembles the original stream

- Acknowledges receipt of received packets - lost ones are resent

TCP


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
U R G
A C K
P S H
R S T
S Y N
F I N
Window
Checksum
Urgent Pointer
Options
Padding
.... data ....

TCP Handshake

Sequence/acknowledgement numbers

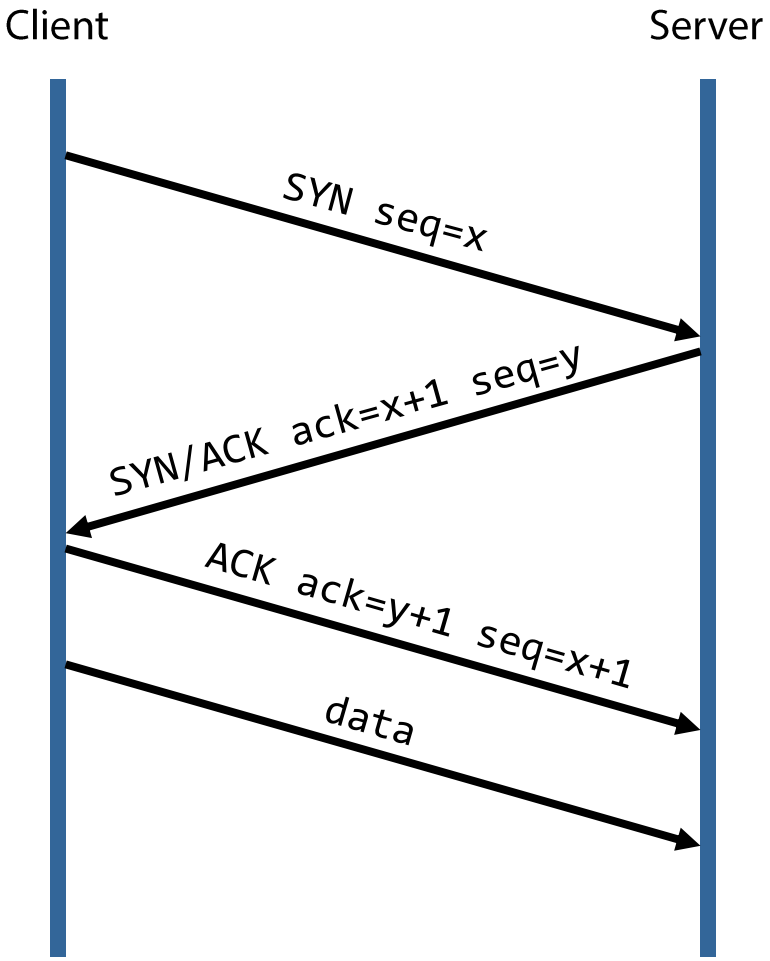
- Retransmissions, duplicate filtering, flow control

***Seq*: the position of the segment's data in the stream**

- *The payload of this segment contains data starting from X*

***Ack*: the position of the next expected byte**

- *All bytes up to X received correctly, next expected byte is X+1*



TCP Issues

Sequence Number Attacks

- TCP connection hijacking/spoofing
- DoS (connection reset)

Port scanning (future lecture)

OS Fingerprinting

- Intricacies of TCP/IP stack implementations

DoS: (future lecture)

- Resource exhaustion
- Blind RST injection

Content injection/manipulation (MitM, MotS)

TCP Sequence Number Prediction

Goal: spoof a trusted host

- Initially described by Robert Morris in 1985

Construct a valid TCP packet sequence without ever receiving any responses from the server

- Exploits predictability in Initial Sequence Number generation

TCP sessions are established with a three-way handshake.

- Client → Server: SYN(ISN_C)
- Server → Client: SYN(**ISN_S**), ACK(ISN_C)
- Client → Server: ACK(ISN_S)

If the ISNs generated by a host are predictable, the other end-point need not see the SYN response to successfully establish a TCP session

Impersonating a Trusted Host

Old TCP stacks would increment the sequence number by a constant amount once per second

- Highly predictable with a single observation at a known time

Attacker impersonates trusted host, predicts **ISN_S**

- Attacker → Server: SYN(ISN_A), SRC = Trusted
- Server → Trusted: SYN(ISN_S), ACK(ISN_A)
- Attacker → Server: ACK(**ISN_S**), SRC = Trusted
- Attacker → Server: ACK(**ISN_S**), SRC = Trusted, ***attack data***

Execute commands based on trusted hosts

- `rsh`, `rcp`, other "r" commands… (hopefully not used these days)

Solution: randomized ISN generation

Man-on-the-Side Attack

Packet capture + packet injection

- Sniff for requests, and forge responses

Requires a privileged position between the victim and the destination server

- Attackers **can** *observe* transmitted packets and *inject* new ones
- Attackers **cannot** *modify* or *drop* transmitted packets

But a less privileged position than what is required for a man-in-the-middle attack!

- Also much easier: no need to keep per-connection state and relay traffic

Example: unprotected (non-encrypted) WiFi network

- MotS: any client that joins the network can mount it
- MitM: need to compromise the access point

Man-on-the-Side Attack

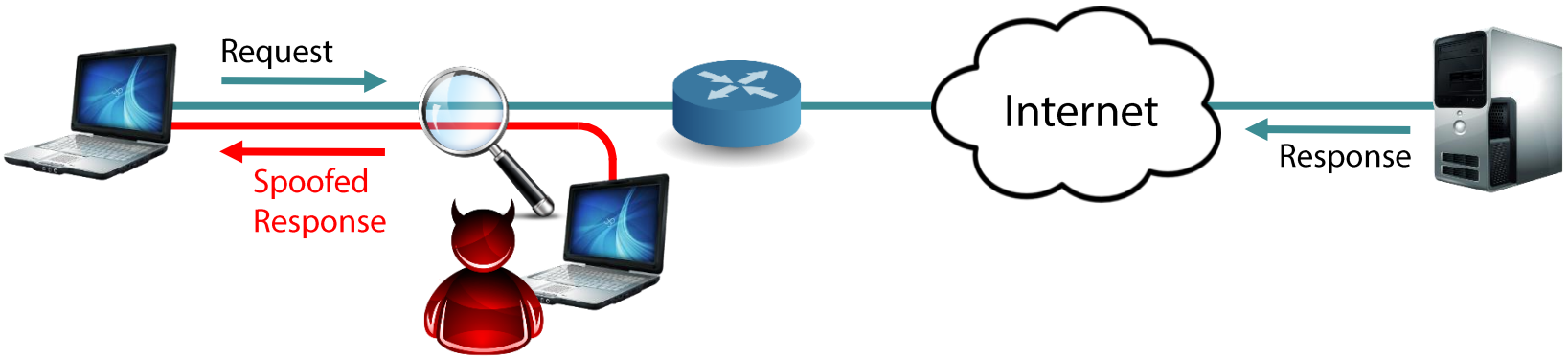
Race condition: attacker's forged response should arrive before the actual server's response

- Most OSes will accept the first packet they see as valid

No need to guess TCP seq/ack numbers!

- The rest of the original stream can follow after the injected packet

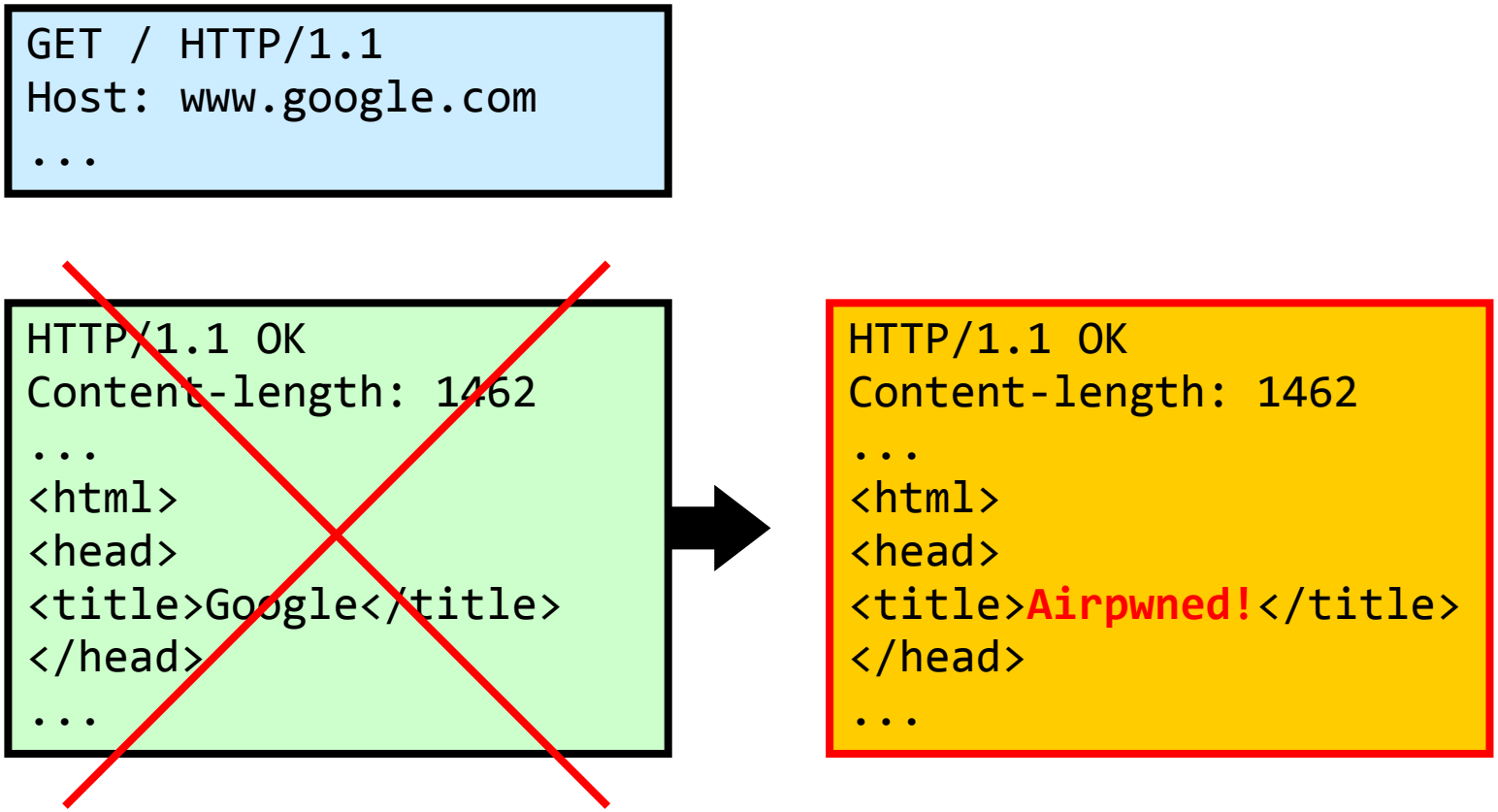
Powerful: redirect to malicious server, manipulate content, inject exploits, …



Airpwn

Listens to wireless packets and acts on interesting HTTP requests based on predefined rules

Beating server's response is easy: the server is several hops away (10s-100s ms) while the attacker is local



A Close Look at the NSA's Most Powerful Internet Attack Tool

BY NICHOLAS WEAVER 03.13.14 | 12:47 PM | PERMALINK

MOST RECENT WIRED POST


New WH
Rules o
Surveill
Short, F
Group S


Is It Eth
Create I
Three D
Sources
Absolut


Lack of
the Onl
Behind
Brutal D


Robot C
Rescue
Its Clas
Drivers


Animat
Lego M

Passive Network Monitoring

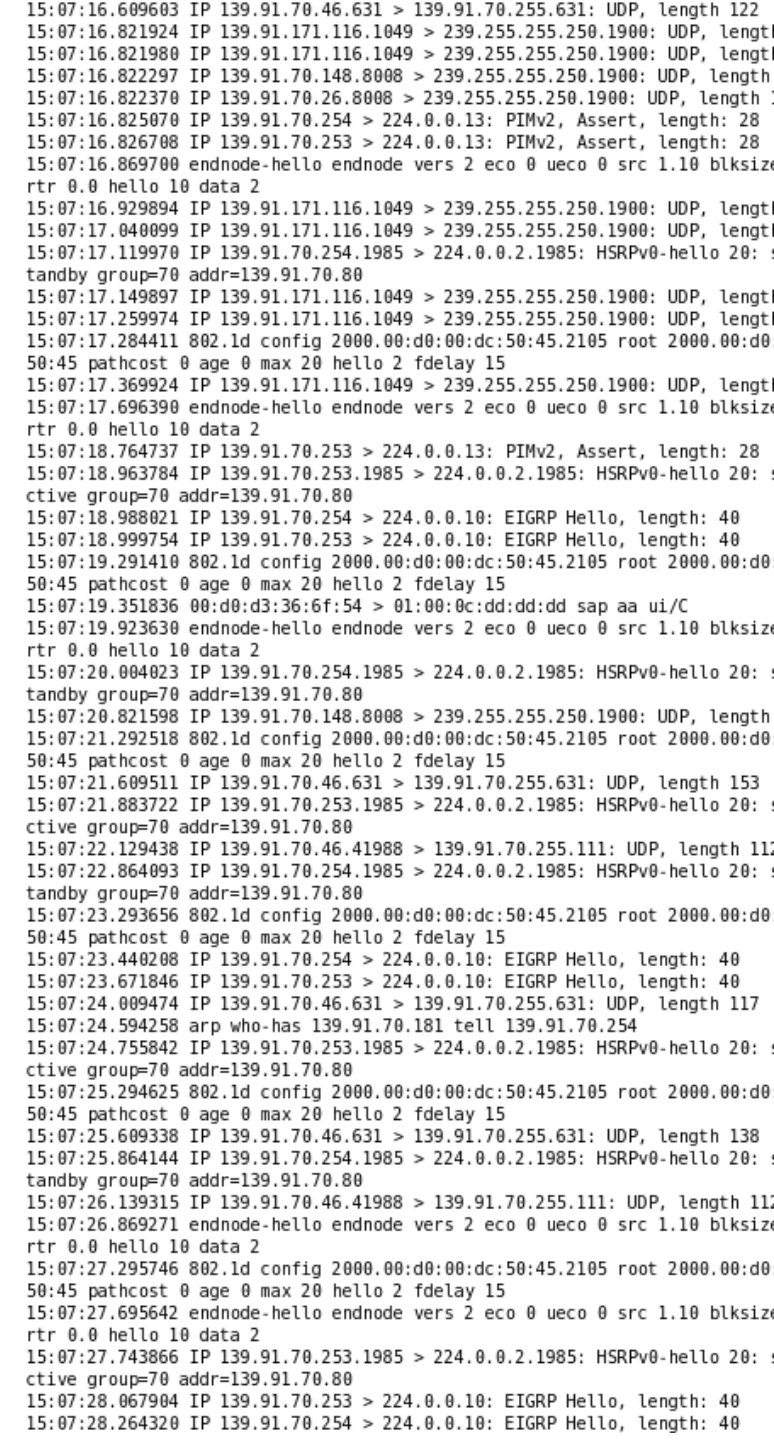
Packet capture
- Headers or full payloads
- Network taps
- Router/switch span/mirror ports

Netflow export
- Connection-level traffic summaries
- Built-in capabilities in most routers

Non-intrusive: invisible on the network

Basis for a multitude of defenses
- IDS/IPS
- Anomaly detection
- Network forensics

Sophisticated attackers might erase all evidence on infected hosts
- Captured network-level data might be all that is left



Packet Capture Tools

Libpcap/Winpcap: user-level packet capture

- Standard interface used by most passive monitoring applications

PF_RING: High-speed packet capture

- Zero-copy, multicore-aware

tcpdump: just indispensable

Wireshark: tcpdump on steroids, with powerful GUI

dsniff: password sniffing and traffic analysis

ngrep: name says it all

Kismet: 802.11sniffer

many more…

Packet Parsing/Manipulation/Generation

Decode captured packets (L2 – L7)

Generate and inject new packets

Tools

- Libnet: one of the oldest
- Scapy: powerful python-based framework
- Nemesis: packet crafting and injection utility
- Libdnet: low-level networking routines
- dpkt: packet creation/parsing for the basic TCP/IP protocols

many more…