CSE508 Network Security

2024-02-01

Lower Layers (Part 1)

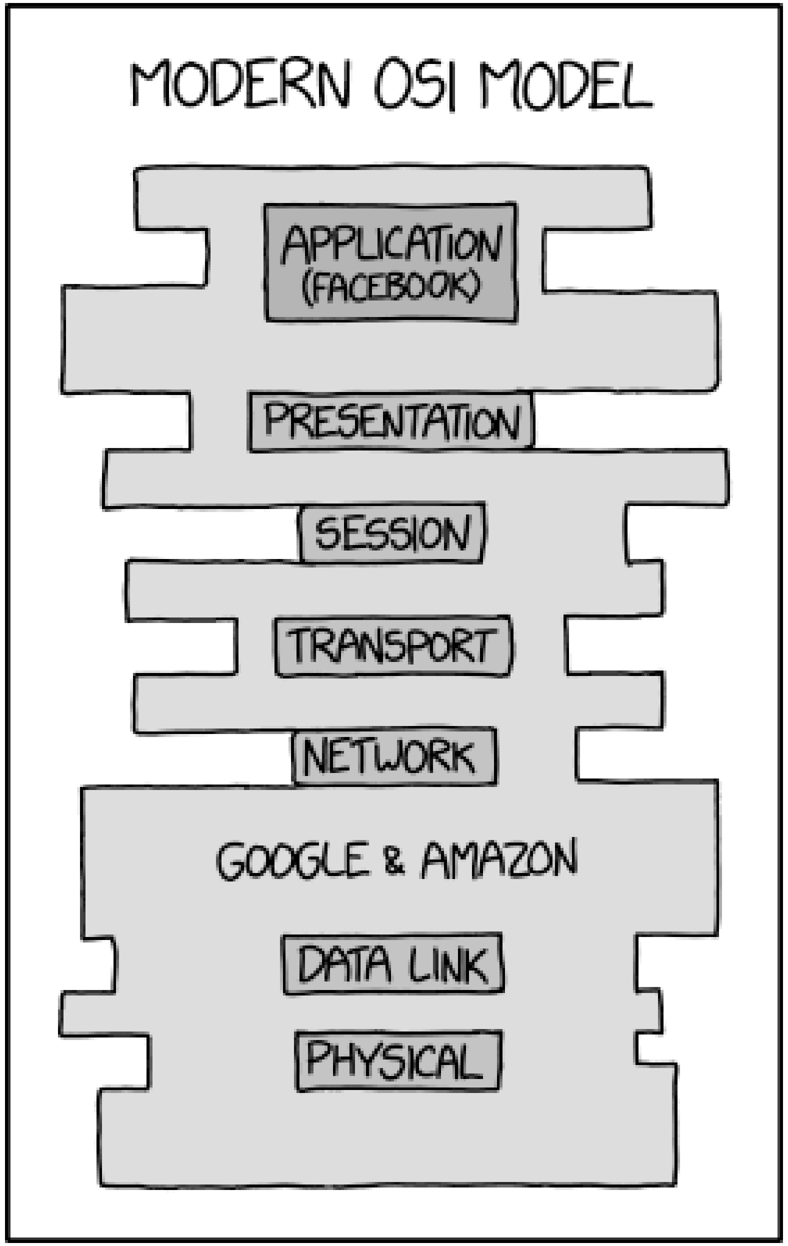
Michalis Polychronakis

Stony Brook University

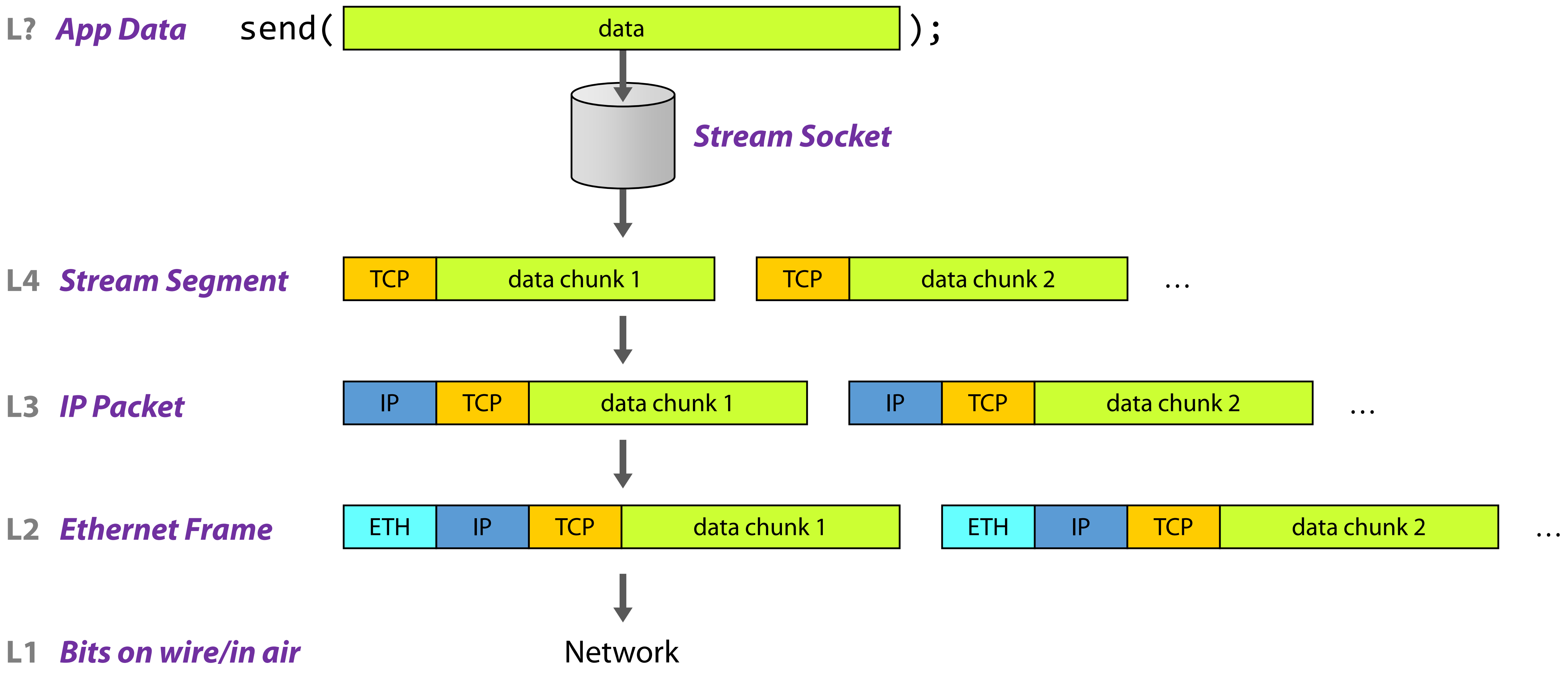
Basic Internet Protocols (OSI Model vs. Reality)

Layer	OSI Model	Reality	Deliver to:	Based on:
L7	*Application*	**HTTP**, BGP, DHCP, DNS, SPDY, SMTP, FTP, SMTP, IMAP, SSH, SSL/TLS, LDAP, NTP, RTP, SNMP, TFTP, …		
L6	*Presentation*			
L5	*Session*			
L4	*Transport*	**TCP**, UDP, SCTP, …	Dst. application	Port number
L3	*Network*	**IP**, ICMP, IPsec, …	Dst. host	IP address
L2	*Data Link*	**Eth**, 802.11, ARP, …	Next hop	MAC address
L1	*Physical*	Fiber, WiFi, …	Wire/air	Network interface (NIC)

End-to-End


MODERN OSI MODEL
APPLICATION
(FACEBOOK)
PRESENTATION
SESSION
TRANSPORT
NETWORK
GOOGLE & AMAZON
DATA LINK
PHYSICAL

Streams vs. Packets



Active vs. Passive Network Attacks

Passive: the adversary eavesdrops but does not modify the network traffic in any way

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

Active: the adversary may transmit new messages, replay old messages, and modify or drop messages in transit

Spoofing, data injection (man-on-the-side), data interception (man-in-the-middle), session replay, DoS, scanning, malicious requests/responses, …

Types of Active Network Attacks

Spoofing (this/future lectures)

Scanning (reconnaissance lecture)

DoS (DoS attacks lecture)

Unsolicited network traffic created from scratch

…

Man-on-the-side (next lecture)

Man-in-the-middle (this/next lecture)

Session replay (web security lecture)

Carefully crafted network traffic based on prior legitimate communication

…

L1

Physical Layer (L1) Attacks

Sniffing

Interception

Wire cutting, jamming, …

Electronic emanations/side channels

Tracking

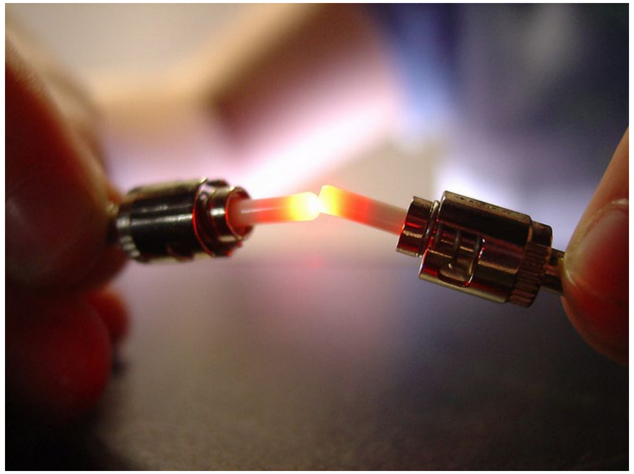
- Device fingerprinting
- Location tracking (cellular, WiFi, Bluetooth, …)
- Many techniques of varying precision: trilateration/triangulation, nearest sensor, received signal strength, …

GLOBAL

The Creepy, Long-Standing Practice of Undersea Cable Tapping

The newest NSA leaks reveal that governments are probing "the Internet's backbone." How does that work?

OLGA KHAZAN JULY 16, 2013



Barta IV/Flickr

In the early 1970's, the U.S. government learned that an undersea cable ran parallel to the Kuril Islands off the eastern coast of Russia, providing a vital communications link between two major Soviet naval bases. The problem? The Soviet Navy had completely blocked foreign ships from entering the region.

Not to be deterred, the National Security Agency launched Operation Ivy Bells, deploying fast-attack submarines and combat divers to drop waterproof recording pods on the lines. Every few weeks, the divers would return to gather the tapes and deliver them to the NSA, which would then binge-listen to their juicy disclosures.

L2

Ethernet

Most commonly used data link layer protocol for LANs

Communication based on *frames*

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|    48 bits                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Destination Address          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    48 bits                    |
|                                      Source Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type (16 bits)         |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                     Payload (46-1500 bytes)                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     32 bits         CRC                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Switch vs. Hub

Hub: Layer 1

A signal introduced at the input of any port appears at the output of every other port

Hubs are now obsolete

Replaced by network switches except old installations or specialized applications

Switch: Layer 2/3

Learn which device is connected to which port and forward packets only to the appropriate port

L3: routing capabilities across VLANs and other advanced routing/management features

Link Layer (L2) Attacks

Traffic sniffing (this lecture)

Traffic injection or interception (this and future lectures)

- Man on the Side (MotS)
- Man in the Middle (MitM)

Spoofing

- Impersonate another machine to 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)

Denial of Service (DoS) (future lecture)

- Flooding, WiFi deauth, …

Network Sniffing

A network interface in *promiscuous* mode can capture all or subset of the traffic that reaches it

- Even if it is not destined for that machine

WiFi: shared medium ➔ *trivial*

Hub: broadcasts packets to all ports ➔ *trivial* *(but hubs are now rare)*

Switch: learns device-to-port mappings and forwards packets *only* to the appropriate port ➔ *still possible!*

- *CAM table exhaustion, ARP cache poisoning*

Wiretapping (wire, fiber)

- Physically "tap" the cable

Network Taps

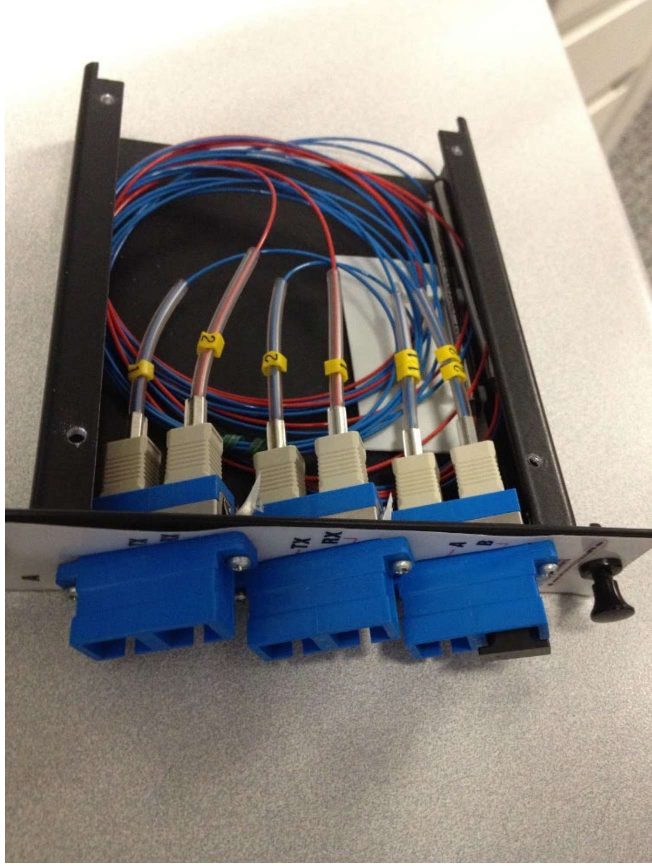
For up to 100Mbit/s can be completely passive

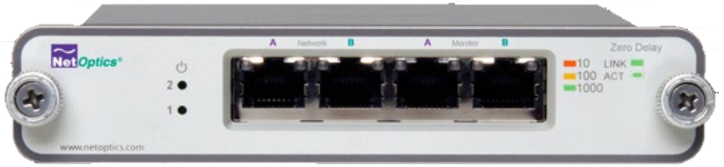
- Gigabit and above needs power for demodulating the signal

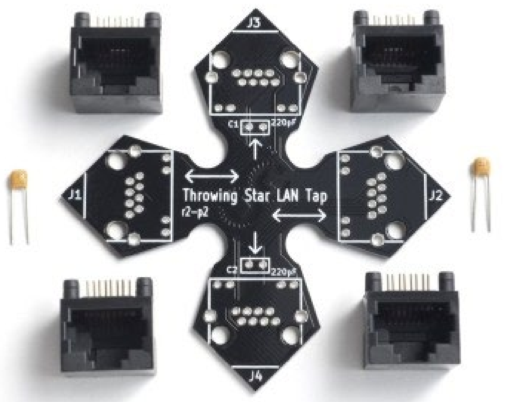
Fiber optical network taps are also completely passive

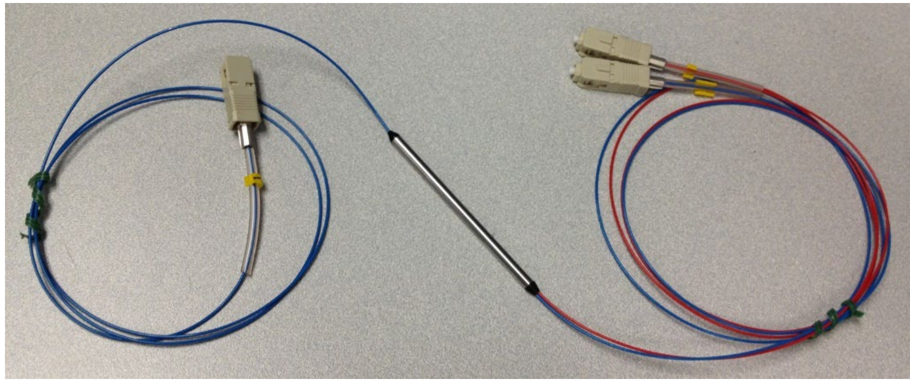
Most high-end switches/routers can mirror traffic

- Span ports





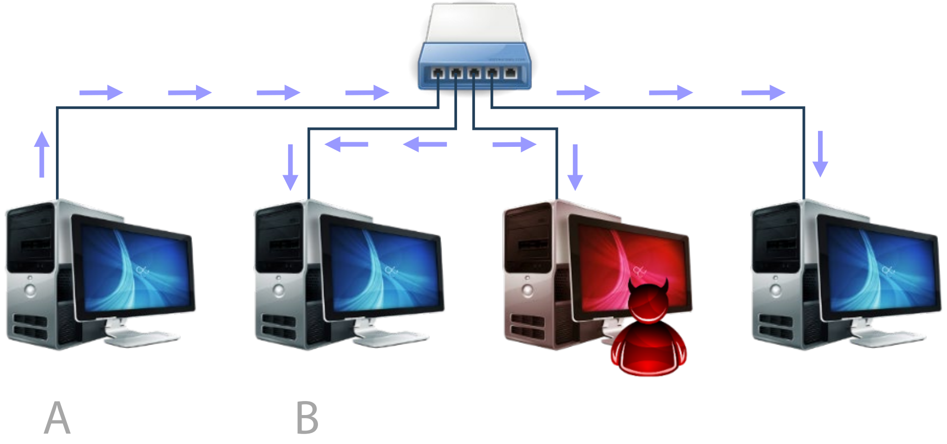




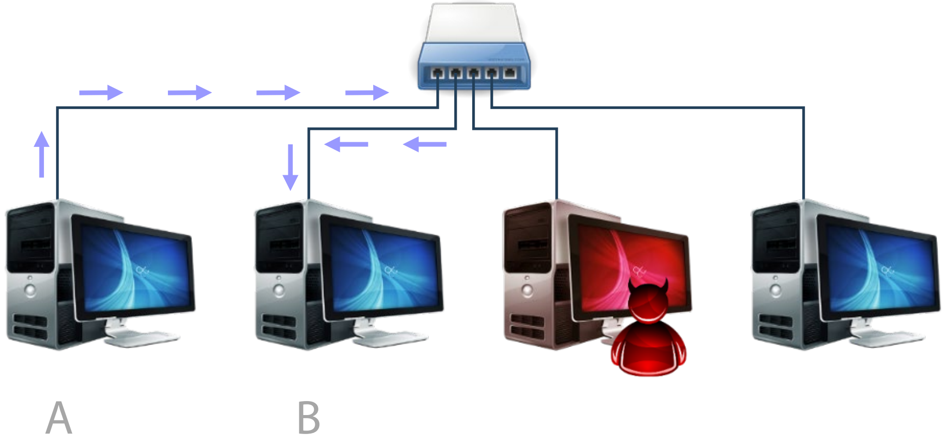
Sniffing without a Tap: Hub vs. Switch

Host A communicates with host B

Hub: packets are broadcast to all hosts, including the attacker



Switch: packets are forwarded only to the destination's port



CAM Table Exhaustion

Network switches use Content Addressable Memory (CAM) to keep a table of mappings between MAC address to physical switch port

- 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`)

Address Resolution Protocol (ARP)

Enables the mapping of IP addresses to physical addresses

A new machine joins a LAN: how can it find the MAC addresses of a neighbor machine (with a known IP address)?

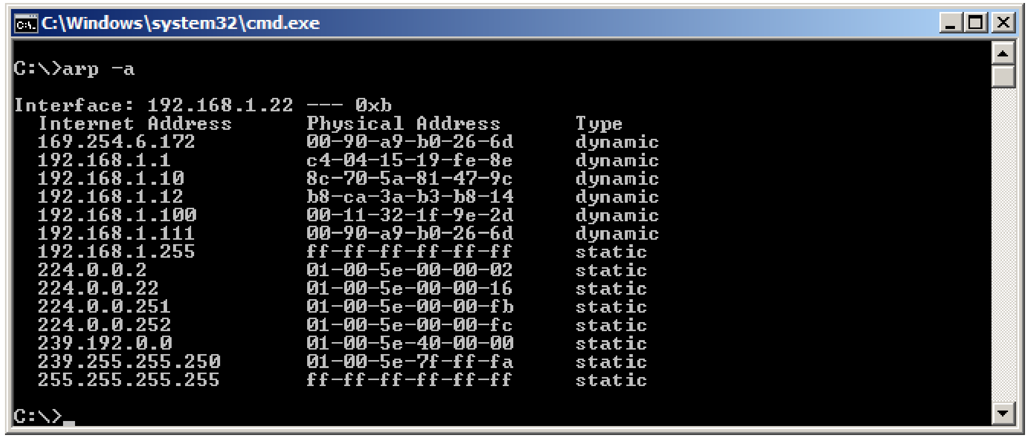
- **ARP request (broadcast):** *Who has IP 192.168.0.1?*
- **ARP reply by 192.168.0.1:** *Here I am, this is my MAC address*

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…***



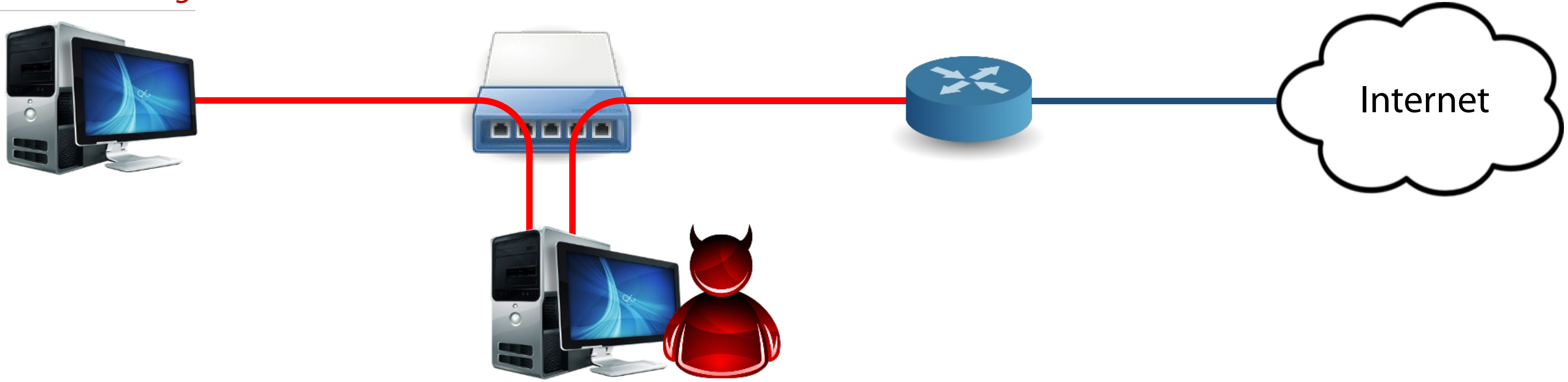
ARP Cache Poisoning

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

Enables traffic interception and manipulation (**man-in-the-middle attack**)

Normal Operation

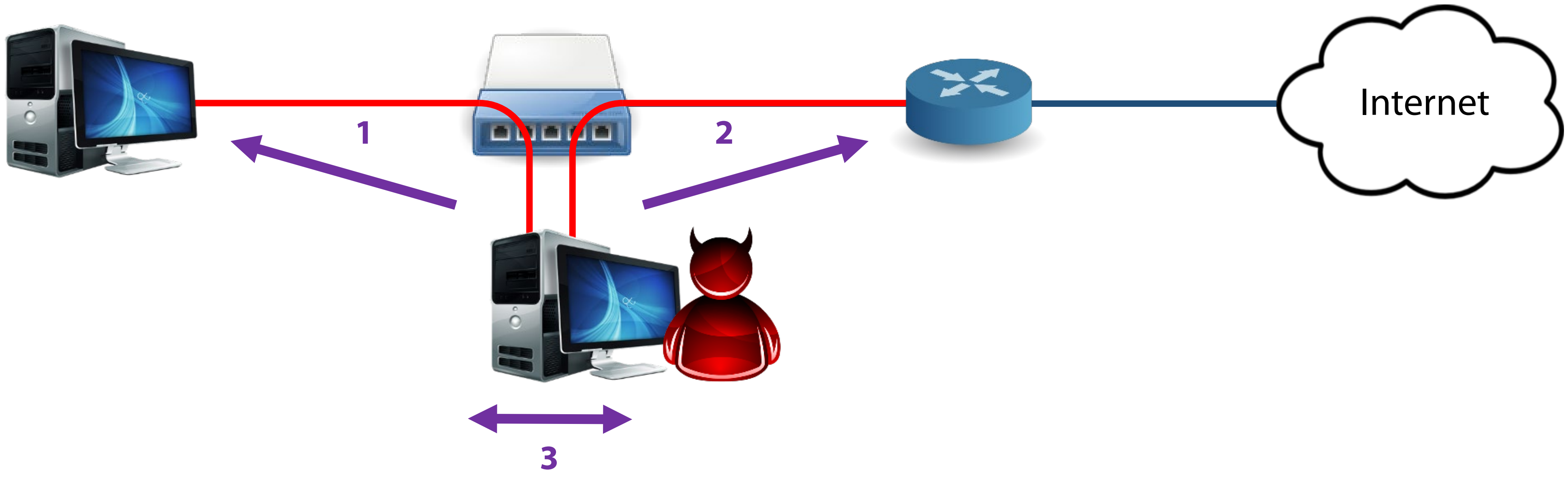
ARP Cache Poisoning



ARP Cache Poisoning

Attack steps

1. ARP reply to victim, mapping gateway's IP to attacker's MAC
2. ARP reply to gateway, mapping victim's IP to attacker's MAC
3. Just forward packets back and forth



ARP Cache Poisoning

Tools

- `arpspoof` (`dsniff`)
- `Ettercap -> Bettercap`
- `nemesis`
- …

Various Defenses

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

What about encrypted WiFi?

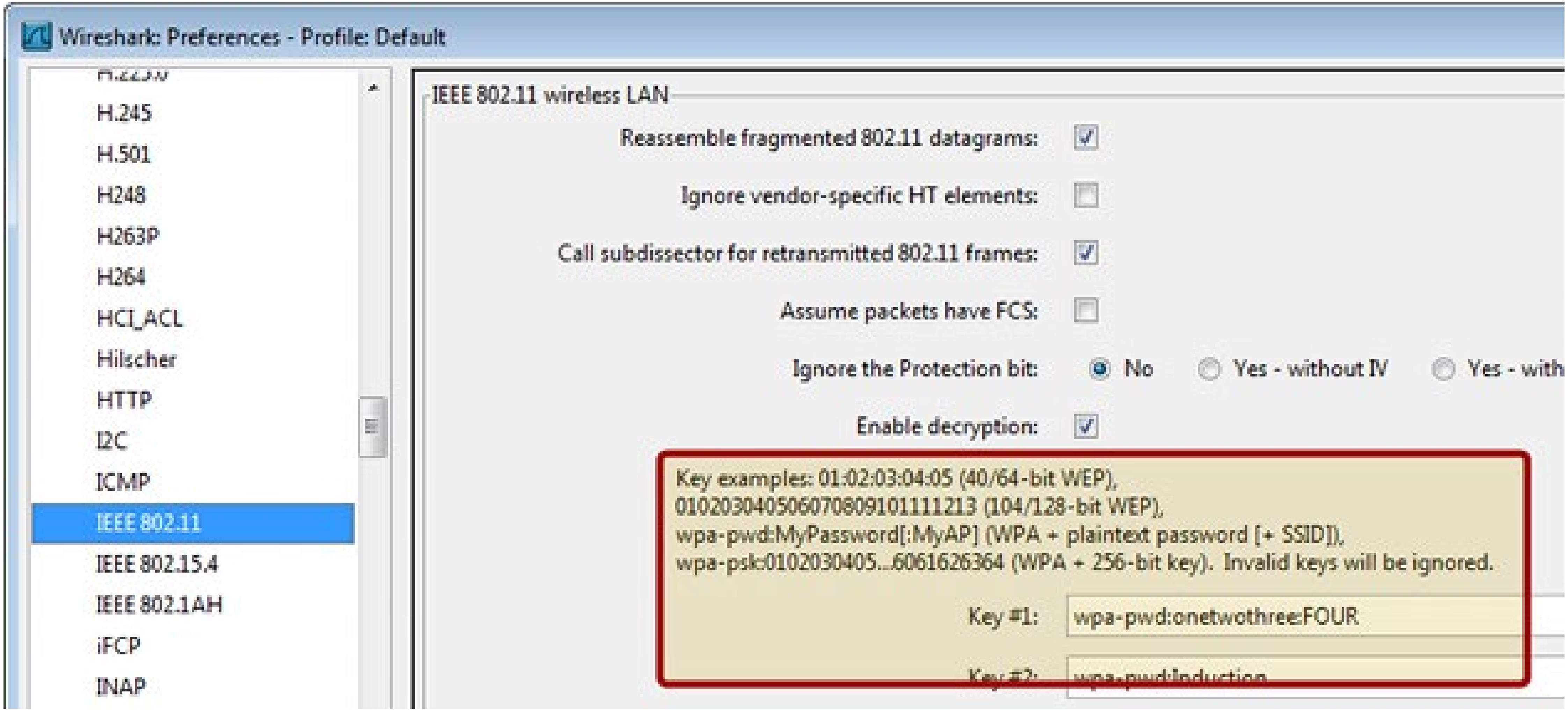
WEP: same pre-shared RC4 key for all clients (now rarely used)

- From within: can freely sniff and decrypt all packets using the same key
- From outside: broken, can trivially crack the key (even brute force takes a short time)

WPA-PSK: a different "pairwise transient key" derived from the pre-shared key is generated for every client

- From within: can sniff and decrypt a client's packets *if* the association process is witnessed (4-way PSK handshake)
- Not a problem! force a re-association by sending a *deauth* packet to the victim
- Injection is then trivial through a connected and authenticated client on the AP
- WPA-Enterprise (802.1X) doesn't suffer from this problem (but is less commonly used)
- From outside: crack the WiFi password (mainly using wordlists)

Native support in Wireshark



Rogue Access Points

No authentication of the AP to the client

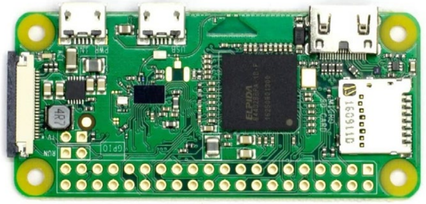
- Set up fake AP with an existing SSID or an enticing name (e.g., `Starbucks-FREE-WiFi`)
- "Auto-connect"/"Ask to join network" mobile phone features greatly facilitate this kind of attacks

Wireless backdoor

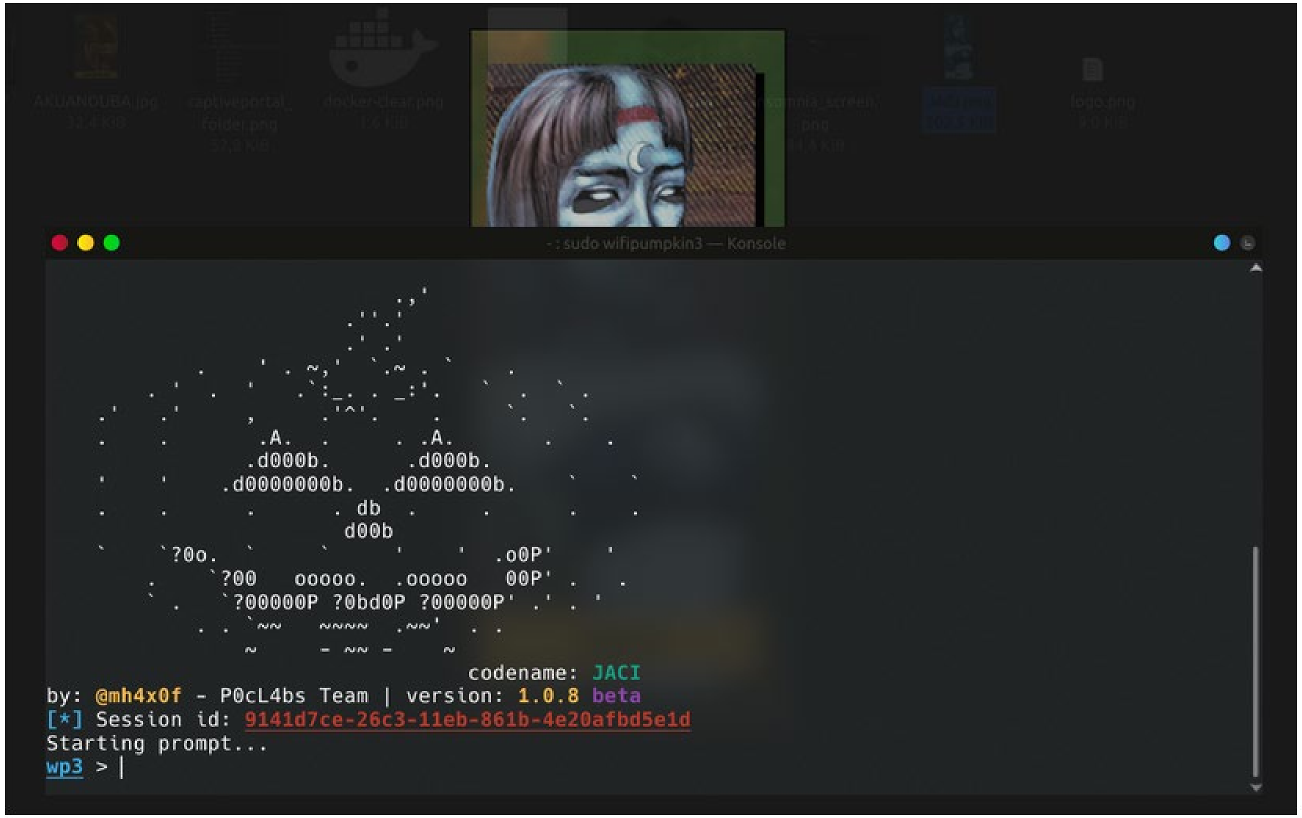
- Ship a WiFi+Cellular-enabled device (e.g., phone) to the target and use its cellular connection for C&C
- Hide a tiny access point in a wall plug, keyboard, …

Detection

- NetStumbler (shows all WiFi networks), RF monitoring systems, Wireless IDS/IPS



WiFi-Pumpkin https://github.com/P0cL4bs/wifipumpkin3



Overview

wifipumpkin3 is powerful framework for rogue access point attack, written in Python, that allow and offer to security researchers, red teamers and reverse engineers to mount a wireless network to conduct a man-in-the-middle attack.

Main Features

- Rogue access point attack
- Man-in-the-middle attack
- Module for deauthentication attack
- Module for extra-captiveflask templates
- Rogue **Dns Server**
- Captive portal attack (captiveflask)
- **Intercept**, inspect, modify and replay **web traffic**
- **WiFi** networks scanning
- **DNS** monitoring service
- Credentials harvesting
- Transparent Proxies
- LLMNR, NBT-NS and MDNS poisoner (Responder3)
- RestFulAPI
- and **more!**

Passive Network Monitoring

Main benefit: *non-intrusive (invisible on the network)*

Basis for a multitude of defenses

- Intrusion detection (IDS) and prevention (IPS) systems
- Network visibility
- Network forensics

Sophisticated attackers may erase all evidence from infected hosts ➔ captured network-level data may be all that is left

Two main types: *individual packets* and *flow summaries*

Packet Capture

Capture all network traffic: maximum visibility

- Using network taps, router/switch mirror ports

Drawback: immense amount of data

- Example: fully utilized 1Gbit/s link for 24 hours: ~~10.8TB~~
- Ethernet links are bidirectional: $2 \times 10.8\text{TB} = 21.6\text{TB}$

Compromise: capture only packet headers

- The increasing use of encrypted communication makes (most) packet payloads useless anyway
- Counter example: ESNI field in TLS *(future lecture)*

Compromise: capture only certain traffic

- Example: DNS requests/responses *(future lecture)*

```
15:07:16.609603 IP 139.91.70.46.631 > 139.91.70.255.631: UDP, length 122
15:07:16.821924 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 325
15:07:16.821980 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 325
15:07:16.822297 IP 139.91.70.148.8008 > 239.255.255.250.1900: UDP, length 101
15:07:16.822370 IP 139.91.70.26.8008 > 239.255.255.250.1900: UDP, length 101
15:07:16.825070 IP 139.91.70.254 > 224.0.0.13: PIMv2, Assert, length: 28
15:07:16.826708 IP 139.91.70.253 > 224.0.0.13: PIMv2, Assert, length: 28
15:07:16.869700 endnode-hello endnode vers 2 eco 0 ueco 0 src 1.10 blksize 149
rtr 0.0 hello 10 data 2
15:07:16.929894 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 325
15:07:17.040099 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 361
15:07:17.119970 IP 139.91.70.254.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
tandby group=70 addr=139.91.70.80
15:07:17.149897 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 361
15:07:17.259974 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 429
15:07:17.284411 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:17.369924 IP 139.91.171.116.1049 > 239.255.255.250.1900: UDP, length 429
15:07:17.696390 endnode-hello endnode vers 2 eco 0 ueco 0 src 1.10 blksize 149
rtr 0.0 hello 10 data 2
15:07:18.764737 IP 139.91.70.253 > 224.0.0.13: PIMv2, Assert, length: 28
15:07:18.963784 IP 139.91.70.253.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
ctive group=70 addr=139.91.70.80
15:07:18.988021 IP 139.91.70.254 > 224.0.0.10: EIGRP Hello, length: 40
15:07:18.999754 IP 139.91.70.253 > 224.0.0.10: EIGRP Hello, length: 40
15:07:19.291410 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:19.351836 00:d0:d3:36:6f:54 > 01:00:0c:dd:dd:dd sap aa ui/C
15:07:19.923630 endnode-hello endnode vers 2 eco 0 ueco 0 src 1.10 blksize 149
rtr 0.0 hello 10 data 2
15:07:20.004023 IP 139.91.70.254.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
tandby group=70 addr=139.91.70.80
15:07:20.821598 IP 139.91.70.148.8008 > 239.255.255.250.1900: UDP, length 101
15:07:21.292518 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:21.609511 IP 139.91.70.46.631 > 139.91.70.255.631: UDP, length 153
15:07:21.883722 IP 139.91.70.253.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
ctive group=70 addr=139.91.70.80
15:07:22.129438 IP 139.91.70.46.41988 > 139.91.70.255.111: UDP, length 112
15:07:22.864093 IP 139.91.70.254.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
tandby group=70 addr=139.91.70.80
15:07:23.293656 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:23.440208 IP 139.91.70.254 > 224.0.0.10: EIGRP Hello, length: 40
15:07:23.671846 IP 139.91.70.253 > 224.0.0.10: EIGRP Hello, length: 40
15:07:24.009474 IP 139.91.70.46.631 > 139.91.70.255.631: UDP, length 117
15:07:24.594258 arp who-has 139.91.70.181 tell 139.91.70.254
15:07:24.755842 IP 139.91.70.253.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
ctive group=70 addr=139.91.70.80
15:07:25.294625 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:25.609338 IP 139.91.70.46.631 > 139.91.70.255.631: UDP, length 138
15:07:25.864144 IP 139.91.70.254.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
tandby group=70 addr=139.91.70.80
15:07:26.139315 IP 139.91.70.46.41988 > 139.91.70.255.111: UDP, length 112
15:07:26.869271 endnode-hello endnode vers 2 eco 0 ueco 0 src 1.10 blksize 149
rtr 0.0 hello 10 data 2
15:07:27.295746 802.1d config 2000.00:d0:00:dc:50:45.2105 root 2000.00:d0:00:d
50:45 pathcost 0 age 0 max 20 hello 2 fdelay 15
15:07:27.695642 endnode-hello endnode vers 2 eco 0 ueco 0 src 1.10 blksize 149
rtr 0.0 hello 10 data 2
15:07:27.743866 IP 139.91.70.253.1985 > 224.0.0.2.1985: HSRPv0-hello 20: state
ctive group=70 addr=139.91.70.80
15:07:28.067904 IP 139.91.70.253 > 224.0.0.10: EIGRP Hello, length: 40
15:07:28.264320 IP 139.91.70.254 > 224.0.0.10: EIGRP Hello, length: 40
```

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…

NetFlow

Connection-level traffic summaries and statistics

- Introduced on Cisco routers around 1996

Built-in capability in high-end routers

Also possible using passive network probes

- Capture traffic → export flow summaries → discard packets

Open alternative: IPFIX

- Internet Protocol Flow Information Export
- Universal IP flow information standard by IETF

Typical NetFlow Setup

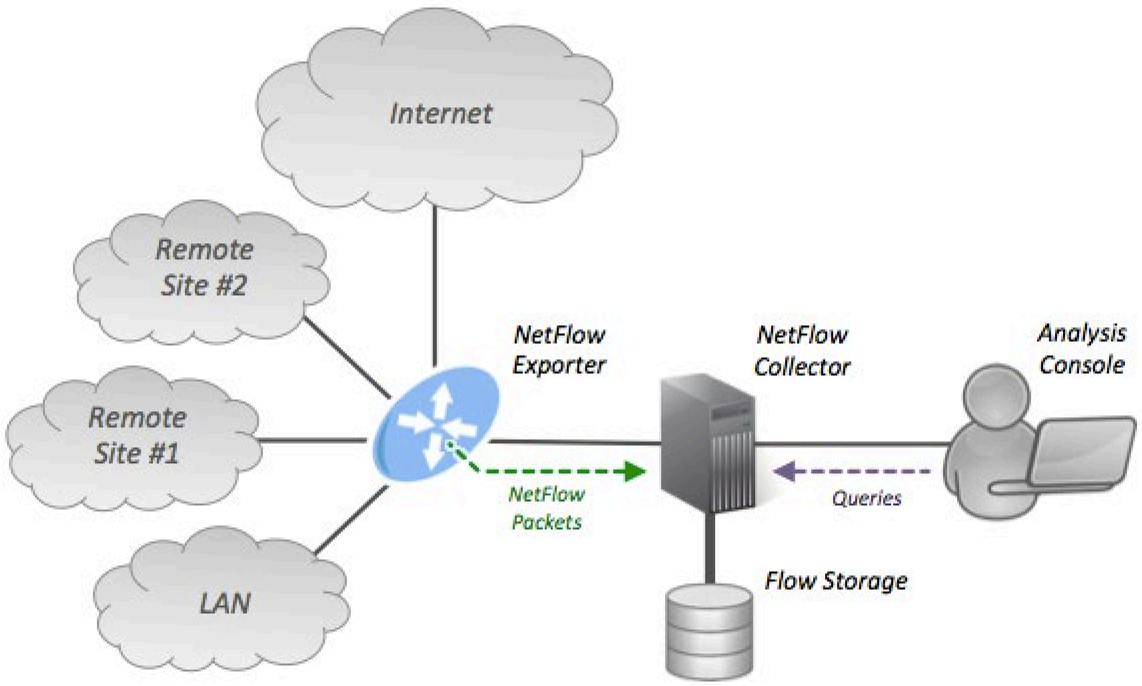
Flow exporter

Router or passive probe

Aggregates packets into flows and exports flow records to flow collectors

Flow collector

Reception, storage and pre-processing of flow data received from flow exporters



Analysis application

Network performance: capacity planning, bandwidth utilization optimization, …

Network visibility: Intrusion detection, traffic profiling, root cause diagnosis, forensics, situational awareness, …

https://en.wikipedia.org/wiki/NetFlow

Flow Definition (NetFlow v5)

Unidirectional sequence of packets that all share seven values

- Ingress interface
- Source IP address
- Destination IP address
- IP protocol number
- Source port for UDP/TCP, 0 for other protocols
- Destination port for UDP/TCP, type and code for ICMP, or 0 for other protocols
- IP Type of Service

The above define a unique *key* for the flow

Advanced NetFlow or IPFIX implementations support user-defined keys

NetFlow v9 Export Packet Example

Packet header followed by *template FlowSets* and/or *data FlowSets*

- Can be intermingled within export packets

Template FlowSet

- Provides a description of the fields that will be present in future data FlowSets

Data FlowSet

- A collection of *data records*
- Each data record provides information about a particular IP flow