Lower Layers (Part 1)

Michalis Polychronakis

Stony Brook University
# Basic Internet Protocols (OSI Model vs. Reality)

<table>
<thead>
<tr>
<th>Layer (L)</th>
<th>Function</th>
<th>Protocols</th>
<th>Deliver to</th>
<th>Based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>L7</td>
<td>Application</td>
<td>HTTP, BGP, DHCP, DNS, SPDY, SMTP, FTP, SMTP, IMAP, SSH, SSL/TLS, LDAP, NTP, RTP, SNMP, TFTP, …</td>
<td>Dst. application</td>
<td>Port number</td>
</tr>
<tr>
<td>L6</td>
<td>Presentation</td>
<td>TCP, UDP, SCTP, …</td>
<td>Dst. application</td>
<td>Port number</td>
</tr>
<tr>
<td>L5</td>
<td>Session</td>
<td>IP, ICMP, IPsec, …</td>
<td>Dst. host</td>
<td>IP address</td>
</tr>
<tr>
<td>L4</td>
<td>Transport</td>
<td>Eth, 802.11, ARP, …</td>
<td>Next hop</td>
<td>MAC address</td>
</tr>
<tr>
<td>L3</td>
<td>Network</td>
<td>Fiber, WiFi, …</td>
<td>Wire/air</td>
<td>Network interface (NIC)</td>
</tr>
<tr>
<td>L2</td>
<td>Data Link</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>Physical</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Streams vs. Packets

L5  App Data  
    send(data);

L4  Stream Segment
    TCP  data chunk 1  TCP  data chunk 2  ...

L3  IP Packet
    IP  TCP  data chunk 1  IP  TCP  data chunk 2  ...

L2  Ethernet Frame
    ETH  IP  TCP  data chunk 1  ETH  IP  TCP  data chunk 2  ...

L1  Bits on wire/in air
    Network
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)

... 

Man-on-the-side (next lecture)
Man-in-the-middle (this/next lecture)
Session replay (web security lecture)

Unsolicited network traffic created from scratch

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, …
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*
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 \Rightarrow 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

Key examples: 01:02:03:04:05 (40/64-bit WEP),
010203040506070809101111213 (104/128-bit WEP),
wpa-pwd:MyPassword[:MyAP] (WPA + plaintext password [+ SSID]),
wpa-psk:0102030405...6061626364 (WPA + 256-bit key). Invalid keys will be ignored.

Key #1: wpa-pwd:onetwothree:FOUR
Key #2: wpa-pwd:induction
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

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!

Overview

wifipumpkin3 is a 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.
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 × 10.8TB = 21.6TB

Compromise: capture only packet headers
The increasing use of encrypted communication makes (most) packet payloads useless anyway
Counter example: ESI field in TLS (future lecture)

Compromise: capture only certain traffic
Example: DNS requests/responses (future lecture)
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:* 802.11 sniffer

*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, …

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