Intro to PCAP

Reid Gilman

Creative Commons

This Is Not A Networking Class

It is about **problem solving**

Goals

1. Teach problem solving strategies using network analysis examples
2. Demystify the fundamentals
3. Know your tools

Syllabus

Day 1

1. Why PCAP?
2. Collection Techniques
3. PCAP Storage
4. Berkeley Packet Filter
5. Connectivity Problems
6. **Lunch**
7. HTTP
8. Chopshop

Day 2

1. Unknown Protocols
2. DNS
3. **Lunch**
4. Final Exercise

WHY PCAP?

(a.k.a. Who Cares?)

What Is PCAP?

- PCAP == **P**acket **Cap**ture
- Complete record of network activity
  - Layers 2 – 7
- Most common format is `libpcap`
  - Open-source
  - Available on *nix and Windows
  - C library, bindings in many languages
  - Others proprietary formats not covered

Pop Quiz!

Who Remembers The OSI Model?

7	• Application
6	• Presentation
5	• Session
4	• Transport
3	• Network
2	• Data Link
1	• Physical

Use Cases

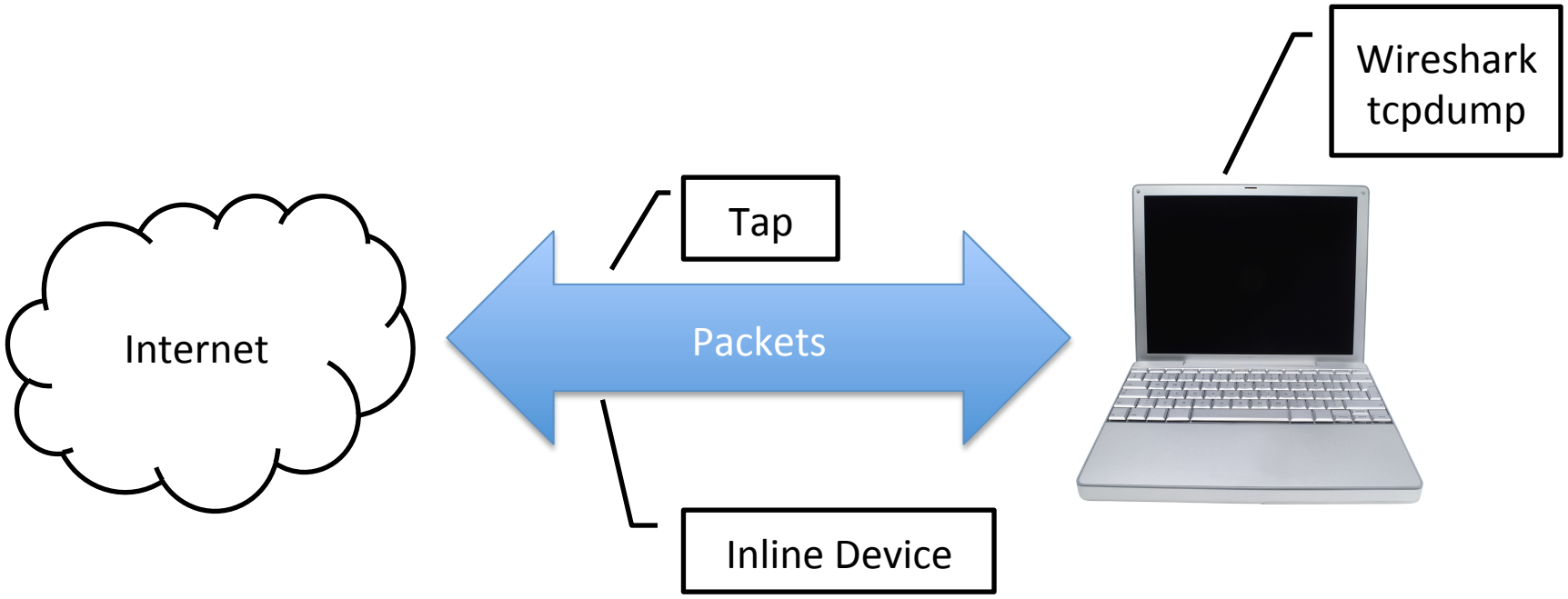
- Identify rogue DHCP servers
- Search for evidence of malware activity
  - "malicious traffic"
- Follow insider threat's footsteps
- Audit bandwidth usage
- Passive DNS resolution
- Monitor intrusions
- Test research hypothesis

Who Uses PCAP?

- **Researchers**: access to raw data
- **Administrators**: debug network problems
- **Analysts**: characterize malware activity
- **Incident Responders**: follow malware

You!

Collecting PCAP



Fair Warning

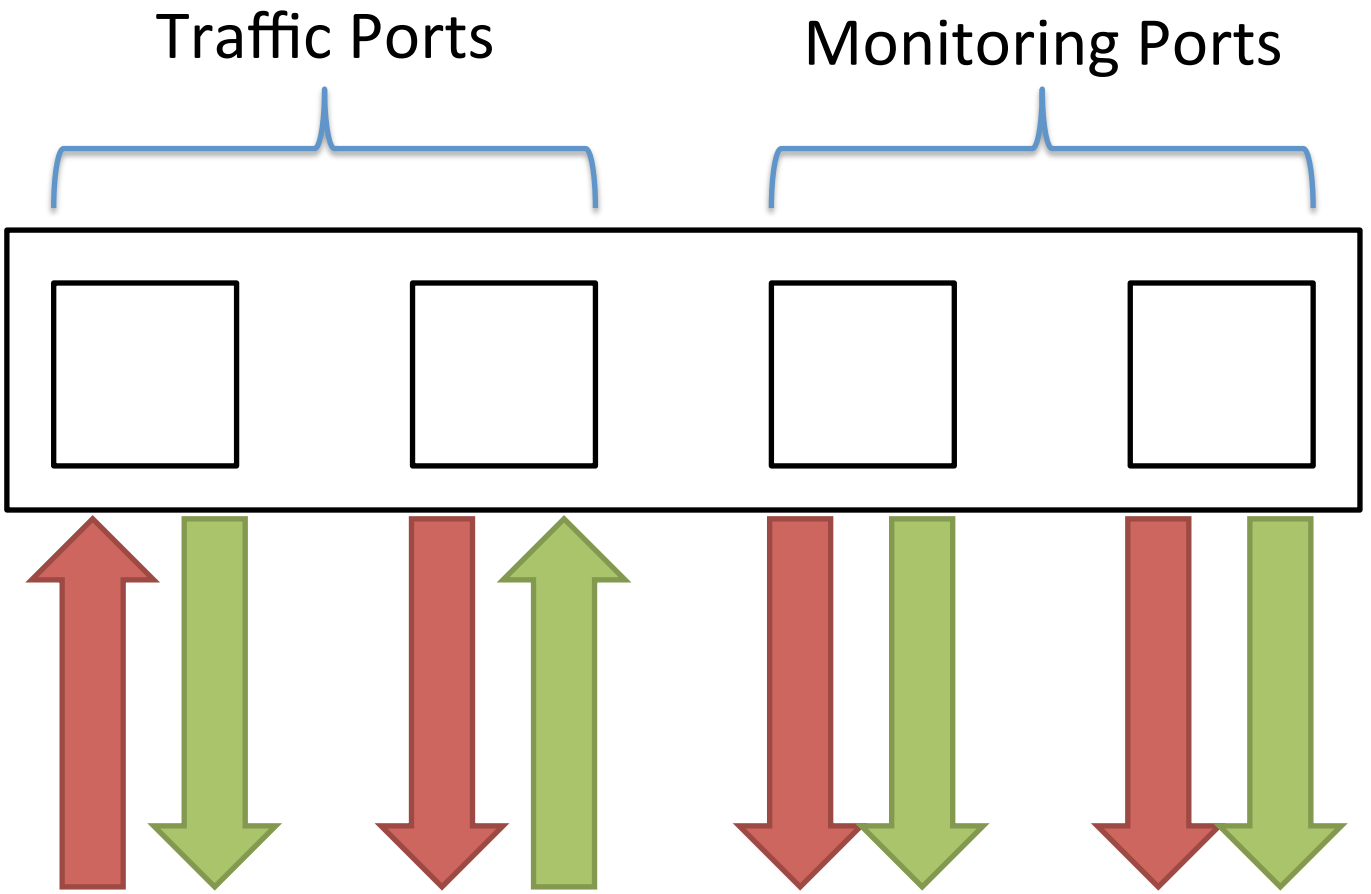
- Any plaintext protocol will be visible
- Careful what you log in to
- You'll be surprised what uses plaintext

Exercise

Use Wireshark and tcpdump to capture traffic while you ping `google.com`. What is in the ICMP Echo Request payload?

- Both tools installed in your VM
- `"ping google.com"`
- You will need to read the tcpdump man page
- `"man tcpdump"`

Aggregating Taps



Inline Devices

Naïve PCAP Storage

$$1\text{gbps} \times 3600 \times 24 = 86400 \text{ gigabits}$$

$$86400 \div 8 = 10800 \text{ gigabytes}$$

- Double that for full-duplex
- Storage can get expensive quickly

Packets Per Second

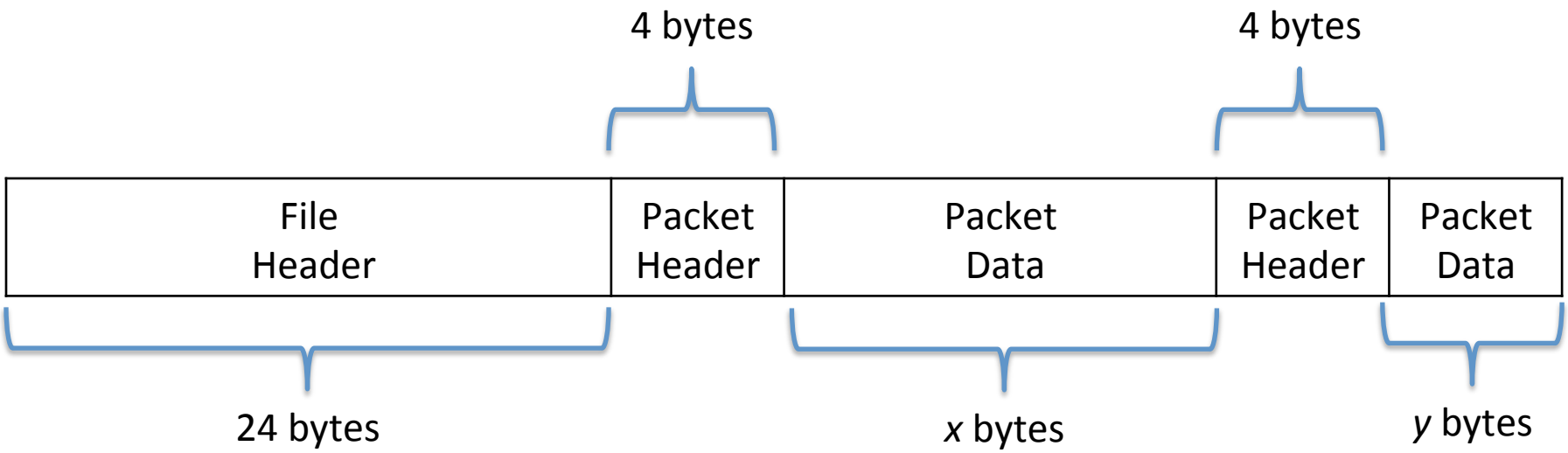
$$1\text{gbps} \div (64\,\text{bytes} \times 8\,\text{bits}) =$$

$$10^9\,\text{bits/s} \div 512\,\text{bits} = 1953125\text{pps}$$

Let H_i represent the overhead of storing one packet

$$N\,\text{pps} \times H_{\text{PCAP}} = NH\,\text{Bps}$$

libpcap Format



Exercise

How much overhead does libpcap incur storing packets for one hour on a saturated simplex 1gbps link with an average packet size of 1500 bytes?

libpcap overhead

Avg. Packet Size (bytes)	Packets Per Second	Overhead (MB/s)	Overhead (GB / day)
64	1,953,125	7.45	628.64
1500	83,333	0.32	26.82
7981	15,662	0.06	5.04
9000	13,888	0.05	4.47

Retention Policies

What to keep and for how long?

Data	Example Retention Period
Full PCAP	Weeks - Months
Flow Records	Indefinitely
DNS	Indefinitely
First N Bytes	Months - Years

BERKELEY PACKET FILTER

Surprisingly Powerful

Berkeley Packet Filter

- a.k.a. BPF
- "`man pcap-filter`" on Unix systems
- Conceptually similar to Wireshark filters
- Filter on layer 2+
- Richest in layers 2 – 4
- Very fast

Filtering Techniques

- BPF is limited, but fast
  - Compiles to an optimized form
  - Almost certainly faster than filters you write
- If you can use BPF, do it

Demo: Counting TCP Packets

You know a particular backdoor sends exactly one message per TCP packet.

How can you use tcpdump and command line tools to get a rough count of how many messages have been sent?

BPF Logic

- Combine BPF primitives with logical operators
  - NOT, AND, OR
- Easy to filter host and TCP/UDP port
- Advanced filters for TCP, UDP, ICMP, etc.
- Access to raw packet bytes

What Does This Do?

```
host 8.8.4.4 and udp port 53
```

Only traffic to or from this IP

Only traffic to or from this UDP port

How About This?

```
dst host 74.125.228.36 and
icmp[icmptype] = icmp-echo
```

How About This?

```
ip dst 74.125.228.36 and
```

Only traffic to this IP

```
icmp[icmptype] = icmp-echo
```

Filter on ICMP type

One More

```
ip[2:2] >= 86 and ip[8:1] <= 4
    and tcp[13:1] & 4 == 4
```

One More

```
ip[2:2] >= 86 and ip[8:1] <= 4
```

IP Length >= 86

IP TTL <= 4

```
and tcp[13:1] & 4 == 4
```

TCP RST

Exercise

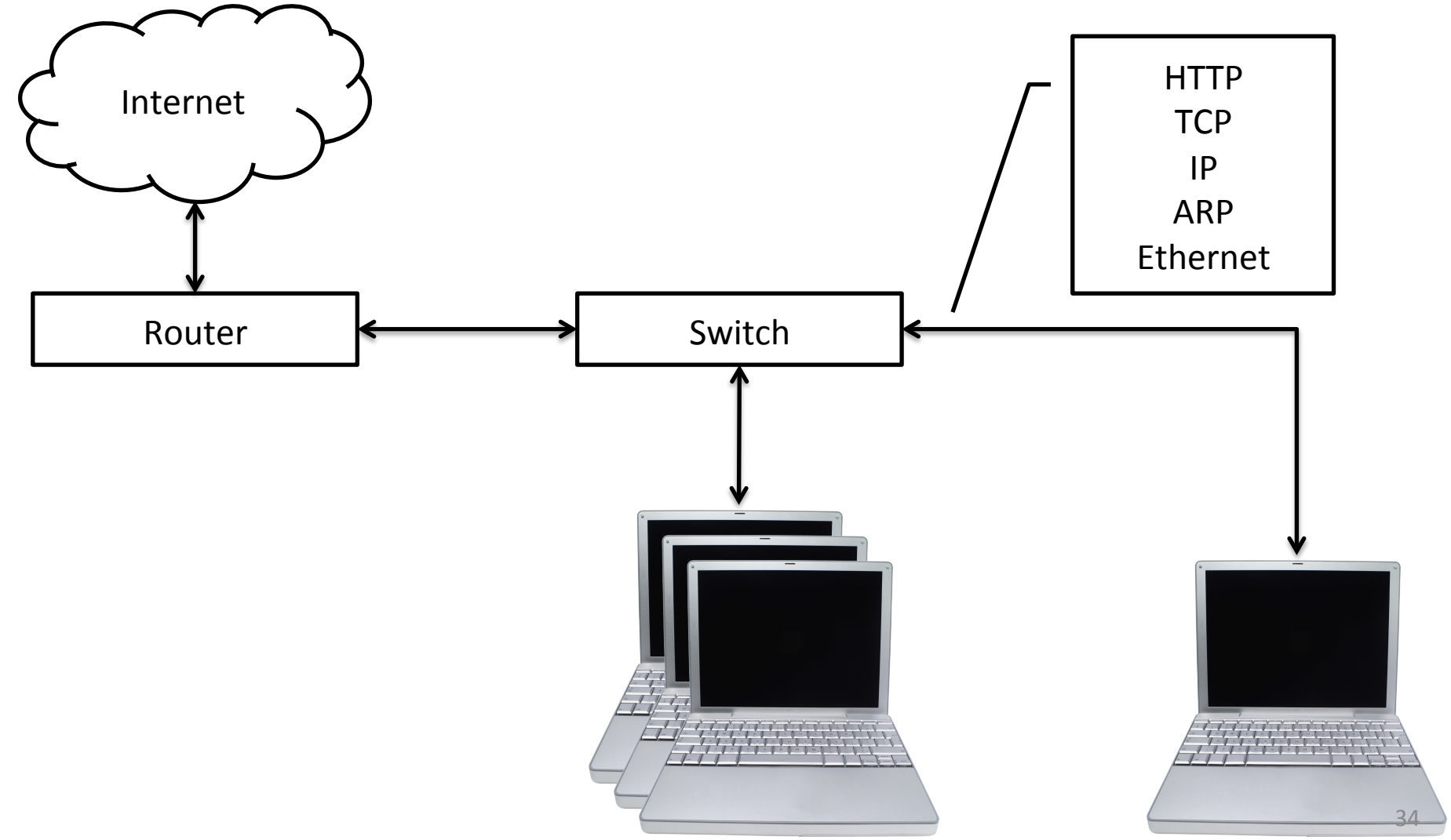
You need to be notified immediately if anyone sets up a successful TCP handshake to 172.16.191.1 on TCP port 80 or if they send it more than 200 bytes on UDP port 53. Look at alert.pcap.

Write a script using tcpdump that will send you an email when either condition triggers.

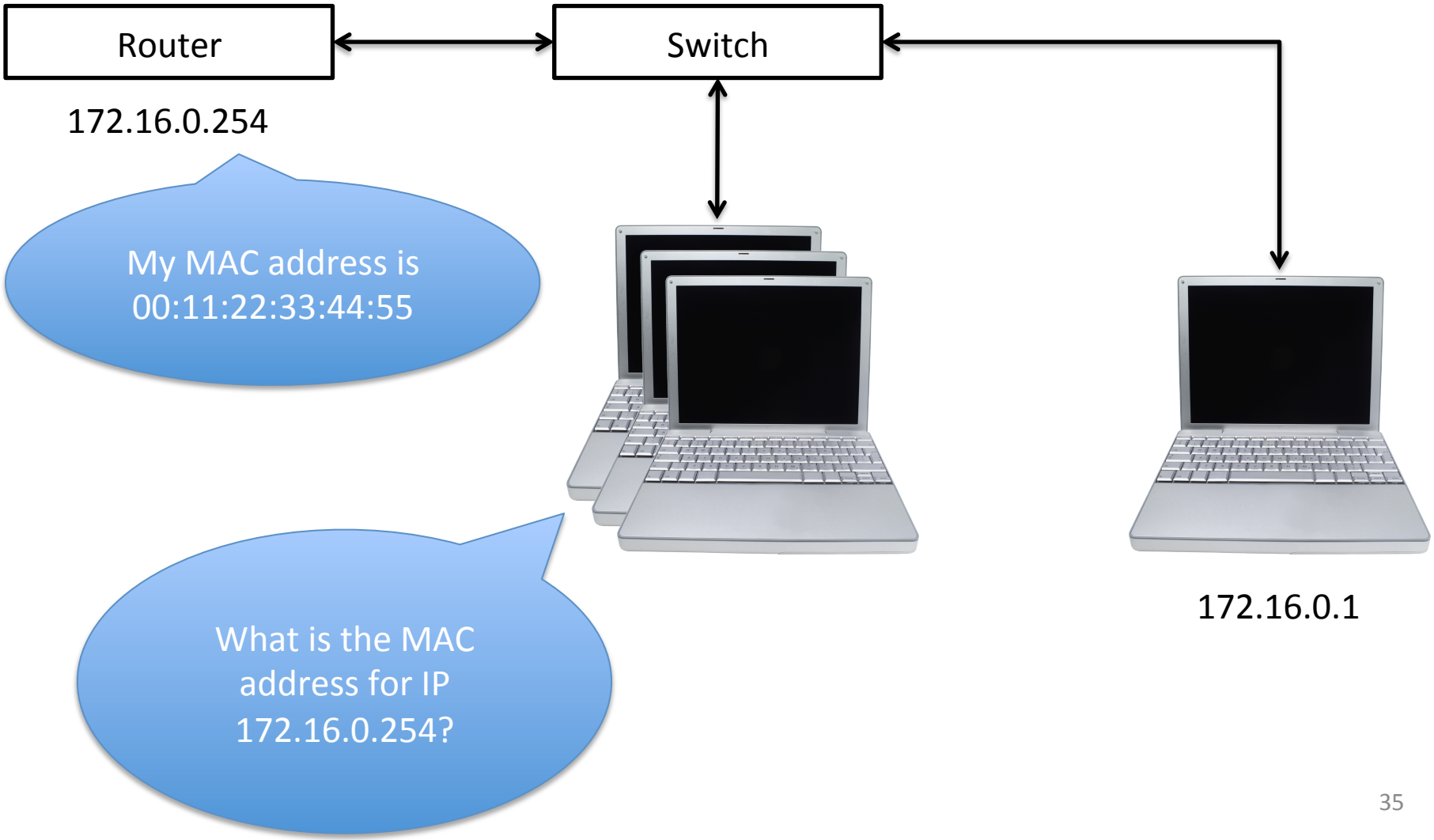
ADDRESS RESOLUTION PROTOCOL

Is It Plugged In?

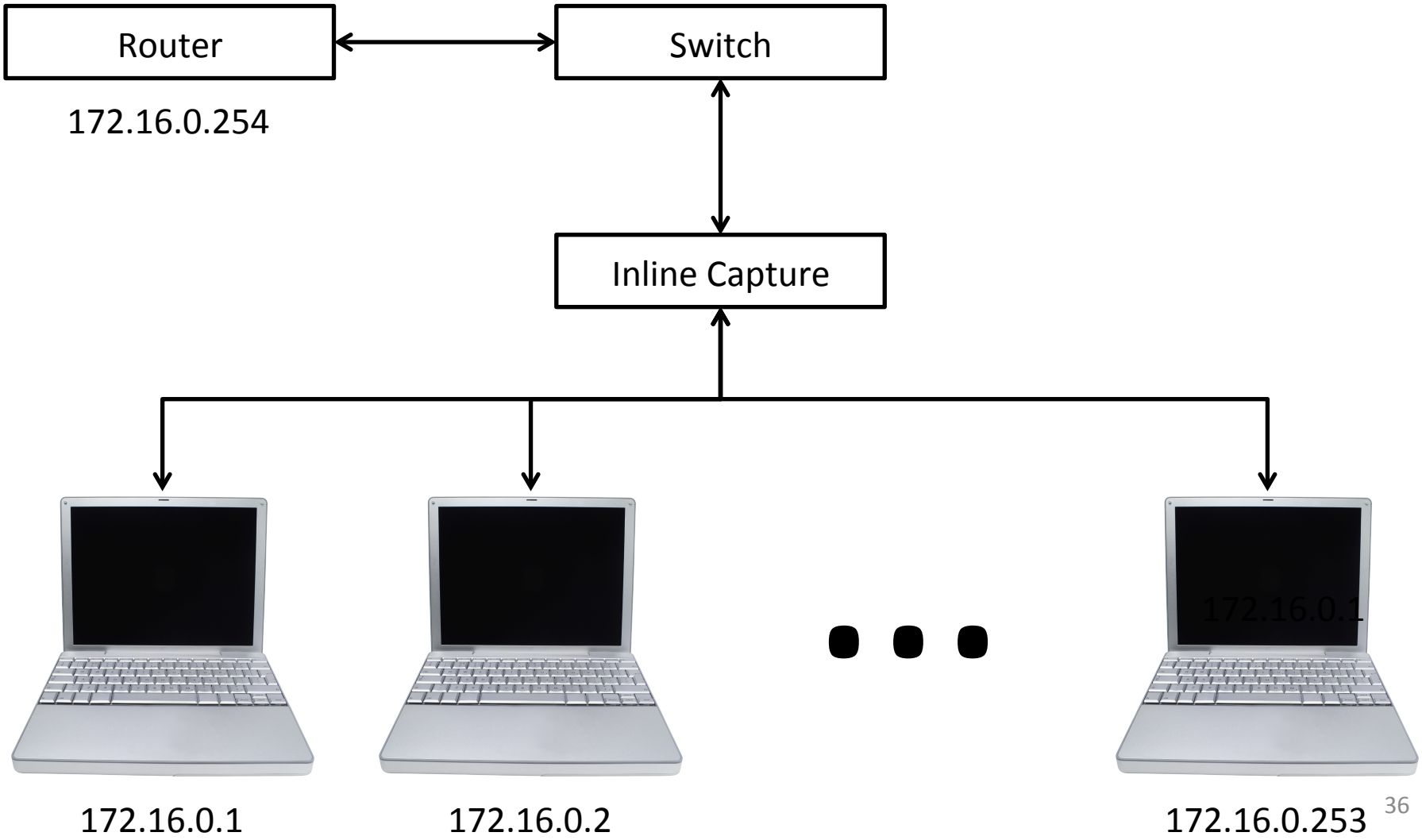
Network Connectivity



ARP


Router
Switch
172.16.0.254
My MAC address is 00:11:22:33:44:55
What is the MAC address for IP 172.16.0.254?
172.16.0.1

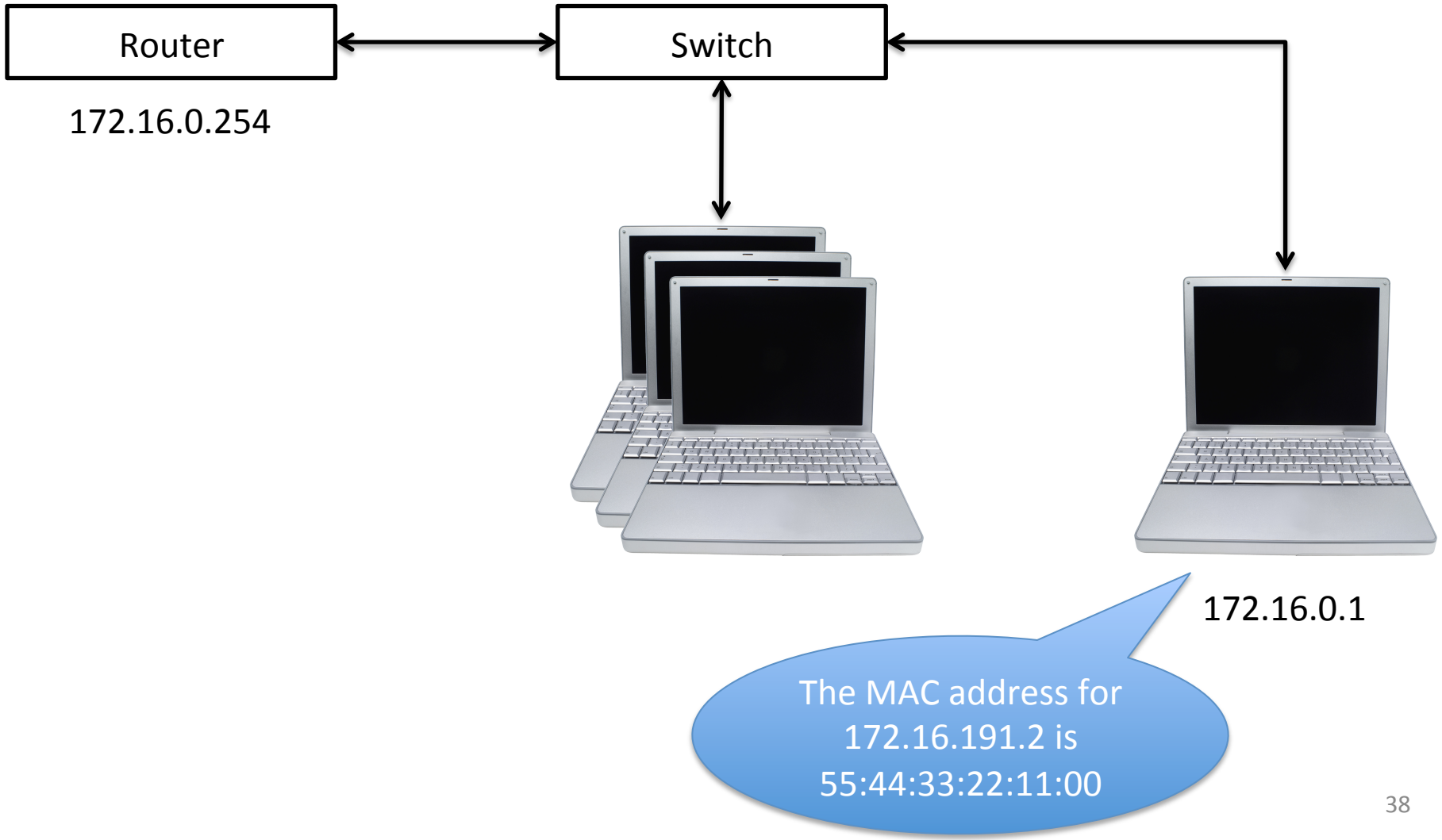
Exercise


Router
Switch
172.16.0.254
Inline Capture
172.16.0.1
172.16.0.1
172.16.0.2
172.16.0.253

Exercise

- Open `arp[0–9]{3}.pcap`
- `arp`*N*`.pcap` shows traffic from 172.16.0.*N*
- Identify:
  - Default router IP address
  - Default router MAC address
  - IP and MAC address mappings

ARP Poisoning



ARP Poisoning

- Intercept all local traffic
- Low processor requirements
- Existing tools:
  - arpspoof + fragroute
  - sslstrip
  - Ettercap
  - Cain & Abel

HTTP

Hypertext Transfer Protocol

- Line-based protocol
- Intuitive fundamentals
- Many corner-cases

- Ubiquitous
- Many uses

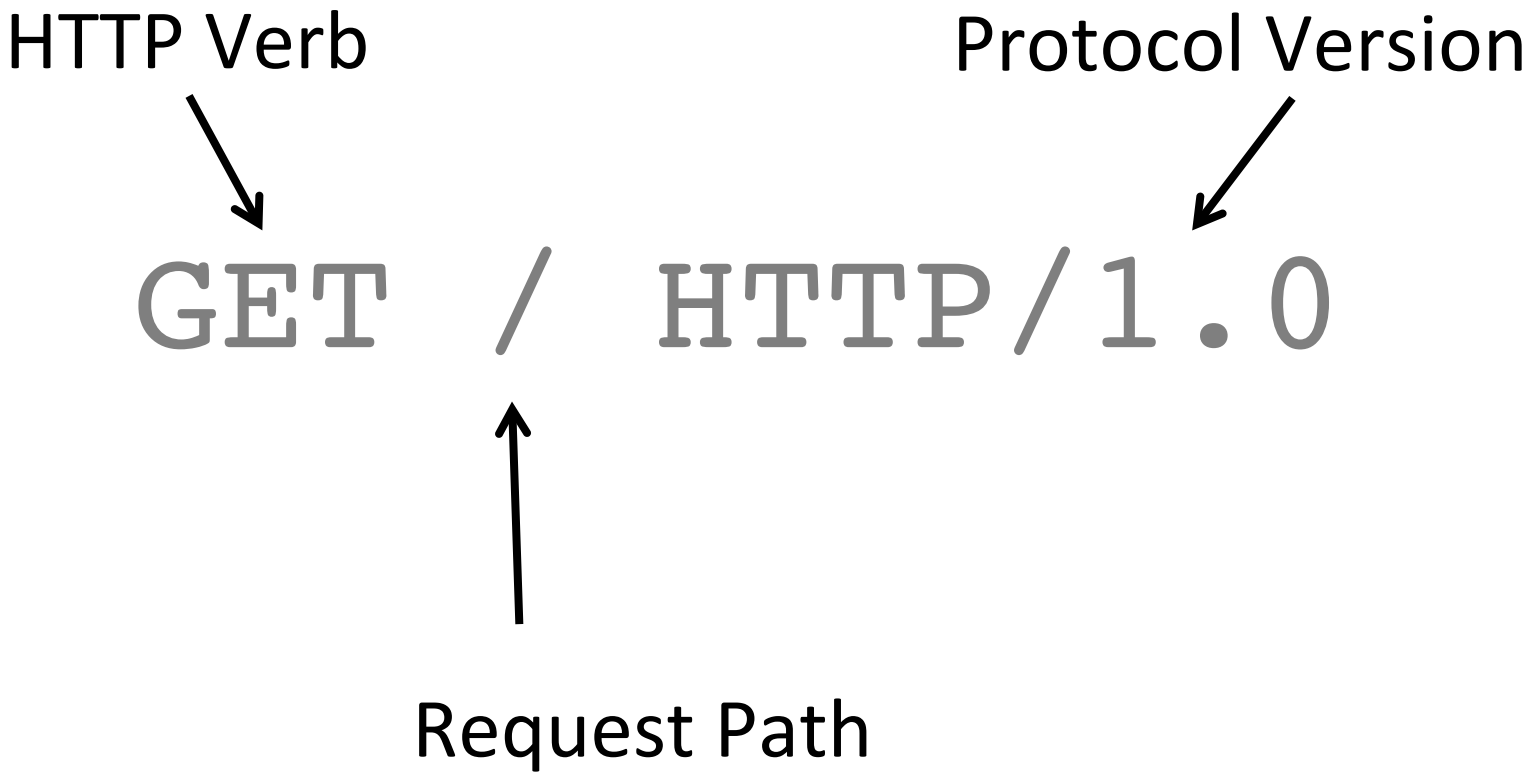
Line-Based

- Headers are separated by line breaks
  - "\r\n"
  - Carriage-Return, Line-Feed
- Easy to read
- Works with existing line-based tools
  - grep, sed, awk, tr, etc.

HTTP

Header 1
Header 2
Header 3
Header *n*
Body

Headers



Headers

Example

`GET / HTTP/1.0`
`Host: www.google.com`
`User-Agent: wget`
`Connection: close`
`BodyDataBodyDataBodyDataBodyDataBodyData`

Example

```
19db 5a85 e52c af9c 4745 5420 6874 7470  ..Z..,..GET.http
3a2f 2f78 6b63 642e 636f 6d2f 2048 5454  ://xkcd.com/.HTT
502f 312e 310d 0a55 7365 722d 4167 656e  P/1.1..User-Agen
743a 2057 6765 742f 312e 3134 2028 6461  t:.Wget/1.14.(da
7277 696e 3131 2e34 2e32 290d 0a41 6363  rwin11.4.2)..Acc
6570 743a 202a 2f2a 0d0a 486f 7374 3a20  ept:.*/*..Host:.
786b 6364 2e63 6f6d 0d0a 436f 6e6e 6563  xkcd.com..Connec
7469 6f6e 3a20 436c 6f73 650d 0a50 726f  tion:.Close..Pro
7879 2d43 6f6e 6e65 6374 696f 6e3a 204b  xy-Connection:.K
6565 702d 416c 6976 650d 0a0d 0a         eep-Alive....
```

Example

```
19db 5a85 e52c af9c 4745 5420 6874 7470  ..Z..,..GET.http
3a2f 2f78 6b63 642e 636f 6d2f 2048 5454  ://xkcd.com/.HTT
502f 312e 310d 0a55 7365 722d 4167 656e  P/1.1..User-Agen
743a 2057 6765 742f 312e 3134 2028 6461  t:.Wget/1.14.(da
7277 696e 3131 2e34 2e32 290d 0a41 6363  rwin11.4.2)..Acc
6570 743a 202a 2f2a 0d0a 486f 7374 3a20  ept:.*/*..Host:.
786b 6364 2e63 6f6d 0d0a 436f 6e6e 6563  xkcd.com..Connec
7469 6f6e 3a20 436c 6f73 650d 0a50 726f  tion:.Close..Pro
7879 2d43 6f6e 6e65 6374 696f 6e3a 204b  xy-Connection:.K
6565 702d 416c 6976 650d 0a0d 0a         eep-Alive....
```

What's happening here?

Example

```
GET / HTTP/1.0\r\n
Host: www.google.com\r\n
User-Agent: wget\r\n
Connection: close\r\n
\r\n
Body Data
```

← CRLF splits headers

← Blank line with CRLF ends headers

Everybody Try This

```
$ echo -e "GET / HTTP/1.0\r\n
> Host: www.google.com\r\n
> \r\n" |
> nc www.google.com 80
```

What Did That Do?

This Is Just Text

- How would you find a particular header?
  - It's value?
- Can you search for strings in the body?
- What is the response code?

Exercise: Find All The Titles

We need to extract all of the web page titles from a PCAP. Look in `http.pcap` for data. List every title exactly once.

Use tcpdump for this exercise.

Exercise: All Websites

Find all of the websites that the host 172.16.191.140 visited in `websites.pcap`. Do not list websites that other hosts visited. Don't forget about servers that may host multiple websites!

Limitations

- tcpdump and grep fall apart on large PCAPs
- tcpdump output not really parseable
- Could use libpcap or pynids
  - Lots of boilerplate code to get going
  - Not ideal for rapid prototyping

CHOPSHOP

Not Just For Cars

Chopshop

- http://www.github.com/MITRECND/chopshop
- MITRE-developed packet framework
  - Based on libnids
  - TCP reassembly
  - Handles boilerplate code
  - Python
  - Great for rapid prototyping

Chopshop

- Framework provides a standard API
- Framework does not analyze packets
- Modules provide all the brains
- Invoke with a list of PCAP files and modules

payloads

- Module to dump packet contents
- Useful for human-readable protocols
  - HTTP, SMTP, IMAP, etc.
- Few command line flags
- Can XOR data
- Can hexdump data
- Good first step in analysis

Invoking Chopshop

```
$ chopshop -f http.pcap "payloads "
```

- Run payloads module on `http.pcap`

```
$ find pcaps -type f |
> chopshop "payloads "
```

- Run payloads on all files in pcaps directory

Simple Obfuscation

- Many simple techniques are frustrating
  - Compression
  - Packing
  - Encoding
- Obfuscation is not encryption
  - No key required to “break” it
  - Still aggravating

XOR

- E**x**clusive **Or**
- Basis for many ciphers
  - RC4, AES
- Fast in hardware
- Trivial in most programming languages
  - Typically a built-in operator
- Key management is easy

XOR Truth Table

Operand *a*	Operand *b*	Result
1	0	1
1	1	0
0	0	0
0	1	1

For $a, b \ni \{0,1\}$, $a \oplus b$ is true iff a $\neq$ b

In Other Words

One or the other,

but not both.

Examples

Operand *a*	Operand *b*	Result
0111	1101	1010
1100	1100	0000
1100	1111	0011
1111 0000	1100 1100	0011 1100
1011 0100	1010 1010	0001 1110
0x10	0x0A	0x1A
0x10	0x32	0x22

XORcise

```
$ chopshop -f xor.pcap "payloads "
```

- Add "`-o 0x`*`NN`*" to XOR contents
- Can you guess the XOR key?

File Carving From HTTP

- Demo: Wireshark
- Exercise: chopshop “http_extractor”

KNOWN UNKNOWNS

Why Can’t Everyone Use HTTP?

Unknown Protocols

- You will encounter something unfamiliar
  - Frequently without client or server to test
- Malware often uses custom protocols
- So do many proprietary programs

Can You Identify It?

- Well-known port?
- Unusual port/protocol pairing?
  - This will break Wireshark
- Constant or repeating values?
- Repeating structure or pattern?
  - Check beginning of packets
  - TLV is very common
  - http://en.wikipedia.org/wiki/Type-length-value

You Can't?

- Can you acquire a client or server?
  - May need to reverse it
- Ask around
- Consider obfuscation and encryption

- You may need to consider alternatives
  - Some things will always be a mystery

Endianness

Hexadecimal	Little Endian	Big Endian
10	1000	0010
432	3204	0432
5555	5555	5555
5432	3254	5432

- “network order” is big endian
- x86 is little endian

Warmup Exercise

1. Use tcpdump, chopshop, or Wireshark to identify the "unknown" flows in:
   1. `unknown-1.pcap`
   2. `unknown-2.pcap`
   3. `unknown-3.pcap`
2. Can you identify the traffic?
3. Can you decode it? How?

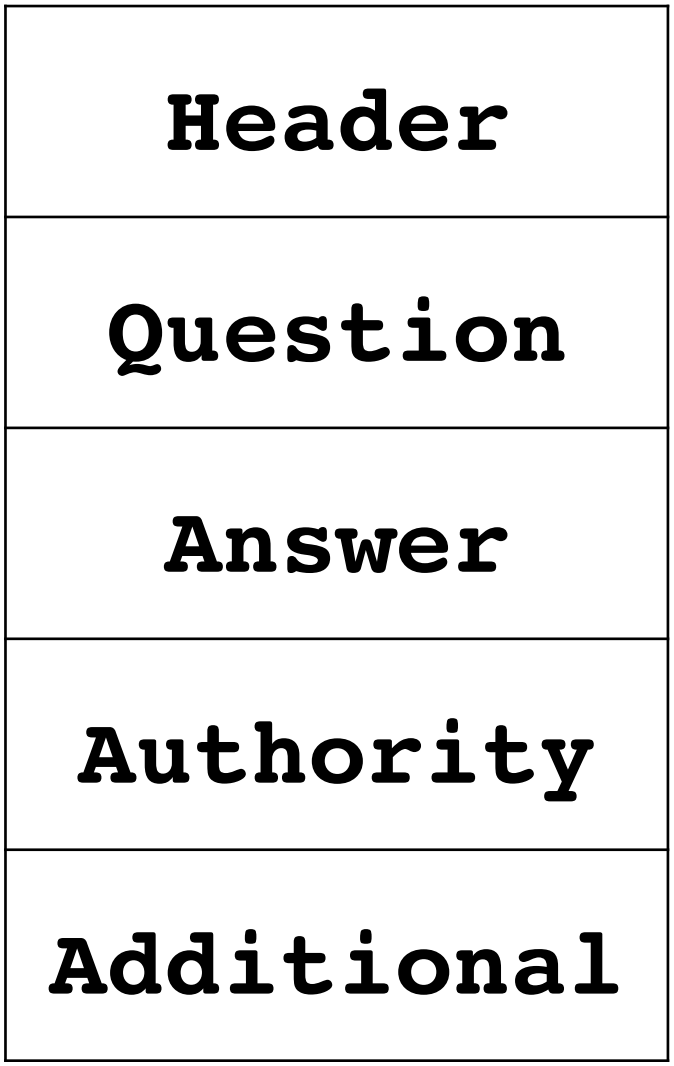
DNS

Domain Name System

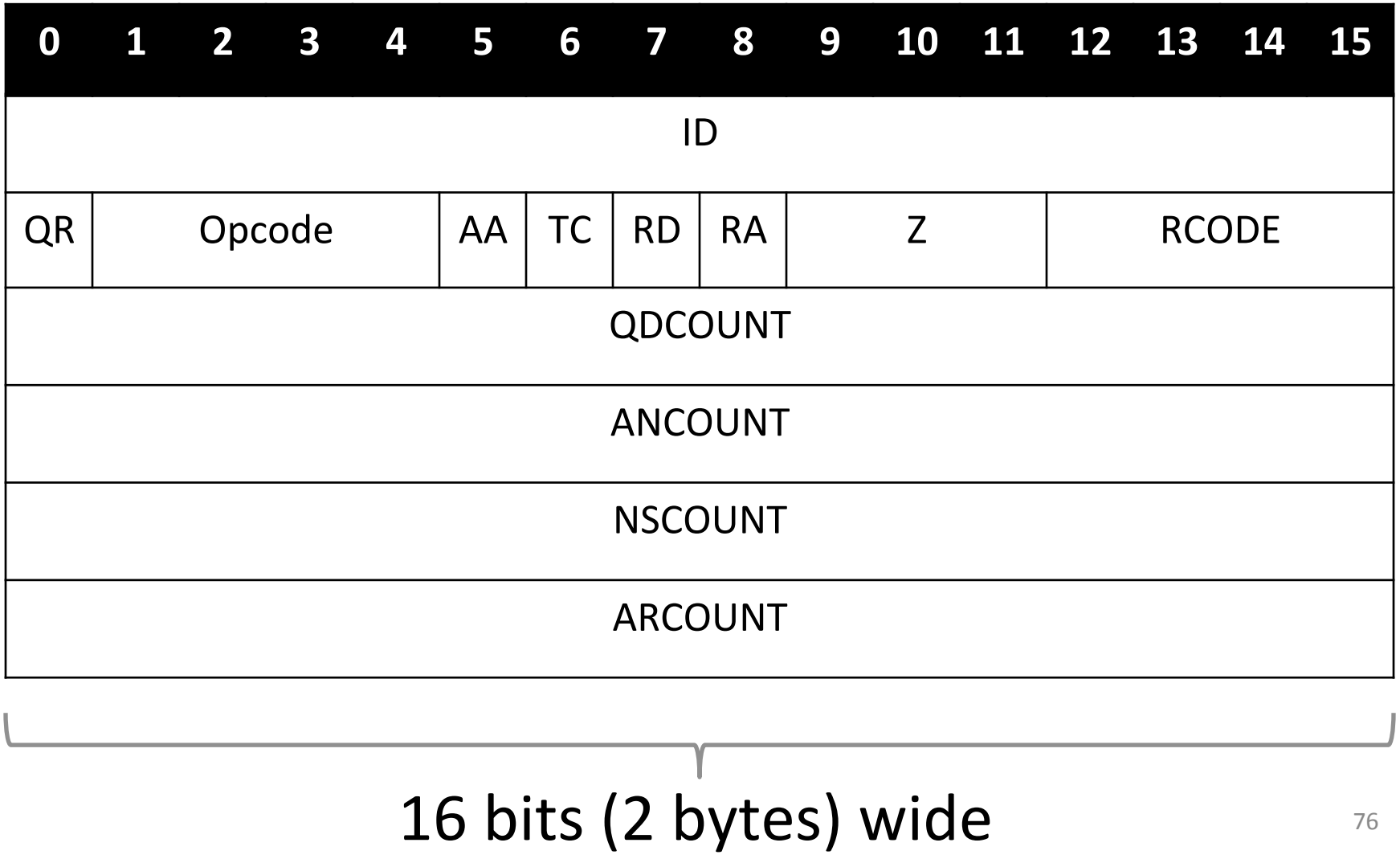
Domain Name System

- Resolve names to IP addresses
  - e.g. www.google.com -> 74.125.228.3
- Most applications use DNS
- DNS servers configured in operating system
  - DHCP
  - /etc/resolv.conf
  - Windows NIC Configuration

Message Format



Header


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
ID
QR
Opcode
AA
TC
RD
RA
Z
RCODE
QDCOUNT
ANCOUNT
NSCOUNT
ARCOUNT
16 bits (2 bytes) wide

Reading Hexdumps

```
$tcpdump -i eth0 –XX -nn udp port 53

0x0000:  406c 8f4d 5c05 5057 a808 e000 0800 4500
0x0010:  0046 3783 0000 3c11 f4f4 8153 1472 8153
0x0020:  3b17 0035 dda1 0032 60e9 b150 8180 0001
0x0030:  0001 0000 0000 0478 6b63 6403 636f 6d00
0x0040:  0001 0001 c00c 0001 0001 0000 020b 0004
0x0050:  6b06 6a52
```

Exercise: DNS and tcpdump

Can you use a combination of tcpdump and grep to discover what IP address a name resolves to?

google.com

Look at dns-tcpdump.pcap

Solution

```
$tcpdump -r dns-tcpdump.pcap “udp
port 53” | grep -A 1 google.com
```

dns_extractor

- Bundled chopshop module
- Examines UDP packets for DNS
- Prints or stores requests and responses

dns_extractor

```
$chopshop -f dns.pcap “dns_extractor -p”
```

– Print every DNS record

```
$chopshop -f dns.pcap -J out “dns_extractor -J”
```

– Write DNS records to file “out” in structured format

```
$find pcaps -type f |
> chopshop “dns_extractor –p”
```

– Run chopshop on every file in the “pcaps” directory

```
$find /example{01..03} -type f -iname > \*.pcap
| sort | chopshop “dns_extractor –p”
```

– Examine PCAPs in three directories in sorted order

Exercise

List all DNS names and their resolutions found in dns.pcap. Only list each (name, resolution) pair once. You may use chopshop or tcpdump. If you can, do it with both.

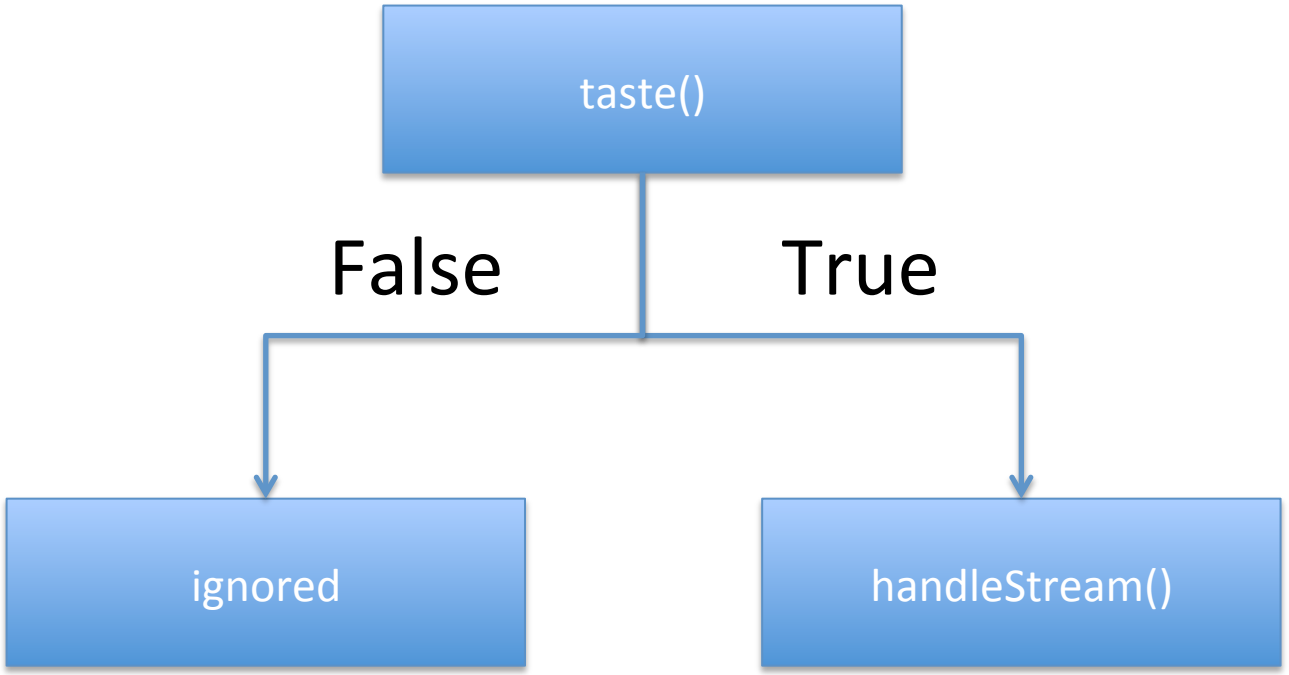
- Example "file-reader.py" might help
- Consider chopshop's JSON-to-file option

Is Everything As It Seems?

- How many (name, resolution) pairs are there?
- How many DNS responses?
- How many `udp src port 53`?

CHOPSHOP MODULES

Module API Basics



Demo

- Open `my_first_module.py`

Mystery Port 53 Traffic

- How to understand this traffic?
- Is it encrypted or obfuscated?
  - With what algorithm?
  - Is there a key? Can you acquire it?

XOR in Python

```
s = "\x00\x01\x01\x01\x04\x05"
key = 0x01
out = ""
for char in s:
  out += chr(ord(char) ^ key)
```

XOR Operator

Convert character to int

XORcise

Decrypt and characterize the mystery traffic

Tips:

1. Remember the principles we just discussed
2. Iterate quickly
3. Worry about the process, not results

Outcomes

1. You captured PCAP
2. You worked on solving realistic, challenging PCAP analysis problems
3. You studied how and when to use different tools and how they might lie to you
4. You analyzed an unknown protocol

Hopefully…

1. You understand how PCAP can help you accomplish your goals
2. You can use a fundamental understanding of network protocols to go above and beyond existing tools
3. You had fun!