Intro to PCAP

Reid Gilman

Creative Commons

2

This Is Not A Networking Class

It is about **problem solving**

3

Goals

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

4

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

5

WHY PCAP?

(a.k.a. Who Cares?)

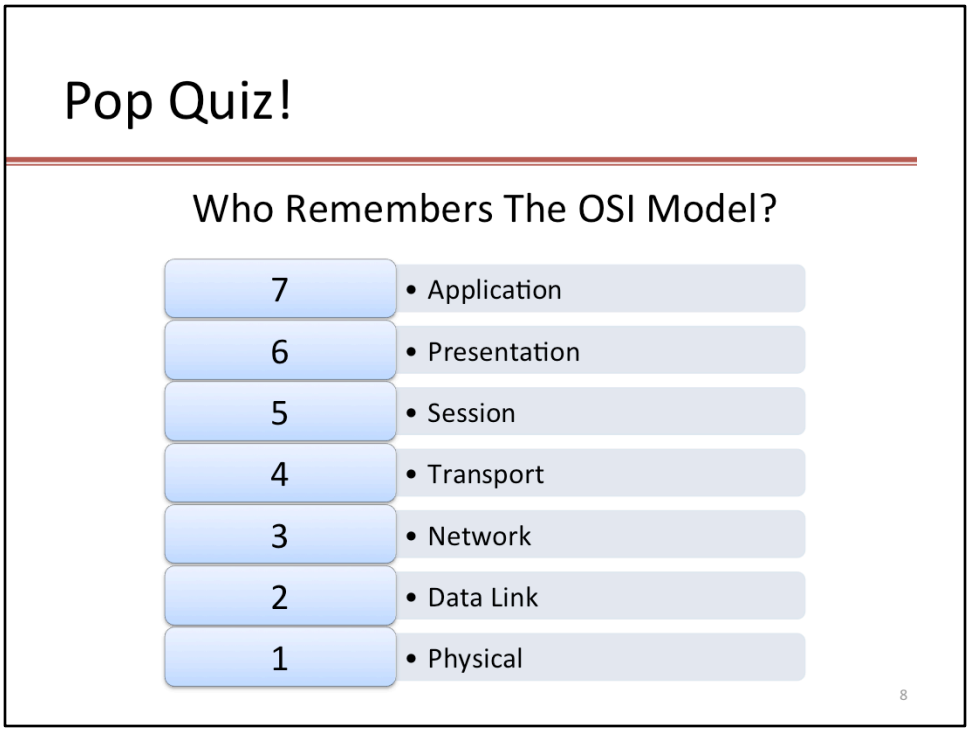
6

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

7

libpcap: http://www.tcpdump.org/



Some discussion questions to make sure that students are all at a reasonable level:

1. What are some examples of protocols at each layer?
    1. FDDI, token ring, Ethernet, 802.11 (actual sending of bits over wires or radio)
    2. Ethernet (spans two layers) (frames, MAC, physical addresses)
    3. IPv4 and IPv6 (this is the level routers operate at; adds routing and logical addressing and fragmentation)
    4. TCP, UDP, IPSec (reliable delivery, flow control, congestion management)
    5. 5-7 are sort of fuzzy and the differences aren't germane to this class
2. What is the difference between TCP and UDP?
3. What is tunneling? What restrictions exist with regards to tunneling at each layer? Can you tunnel Ethernet over HTTP (sure)? Can you tunnel IP over Ethernet over HTTP over FTP? Sure, but why?
4. How does TCP work?
    1. What is a three-way handshake?
    2. What is a sequence number?
    3. Acknowledgement number?
    4. Does TCP provide reliable delivery?
5. How does UDP work?
    1. Does it provide reliable delivery?
    2. When might you want to use UDP over TCP?

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

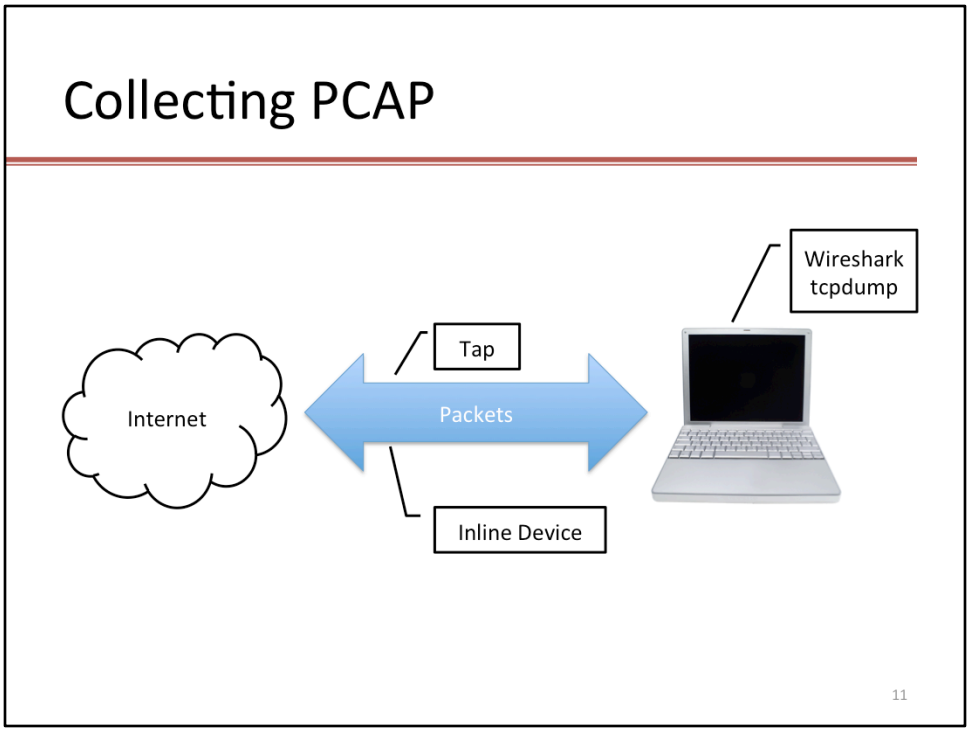
9

Who Uses PCAP?

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

You!

10



Do you collect the same information capturing at different locations on the network? Why or why not?

- Generally no. The contents of a TCP stream might not change but physical (layer 2) traffic will change at each Ethernet hop. NAT can also give a radically different perspective upstream vs. behind the NAT. There is substantial information loss.

Fair Warning

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

12

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"`

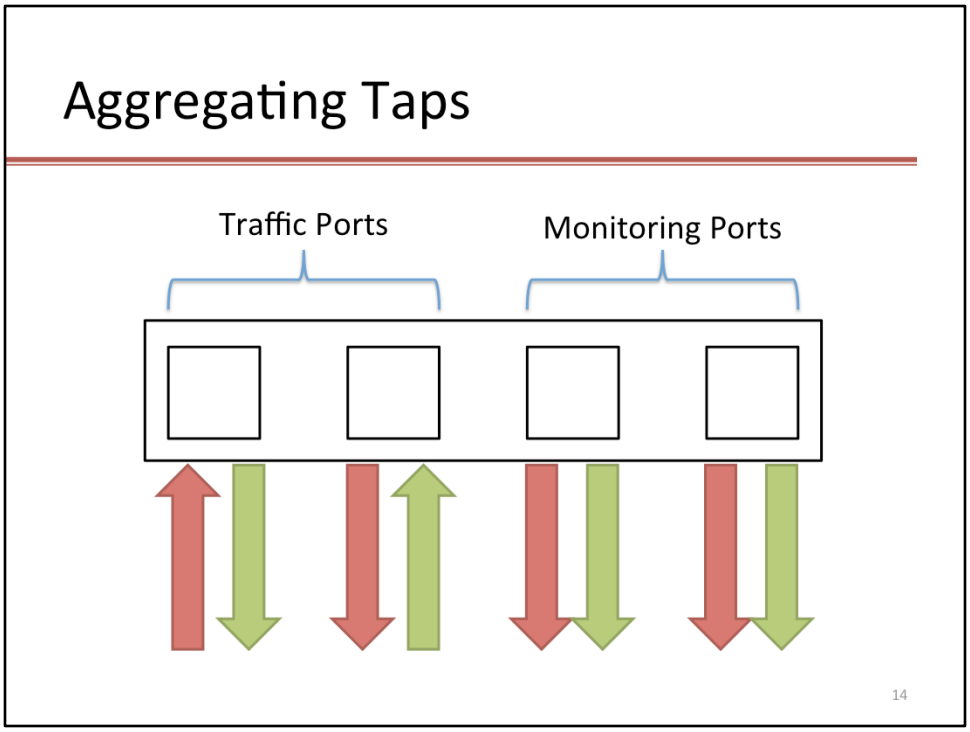
13

Do it with tcpdump as a class but don't show -X, then let them go

- Talk about snaplen
- Talk about –n and when you might want to use it

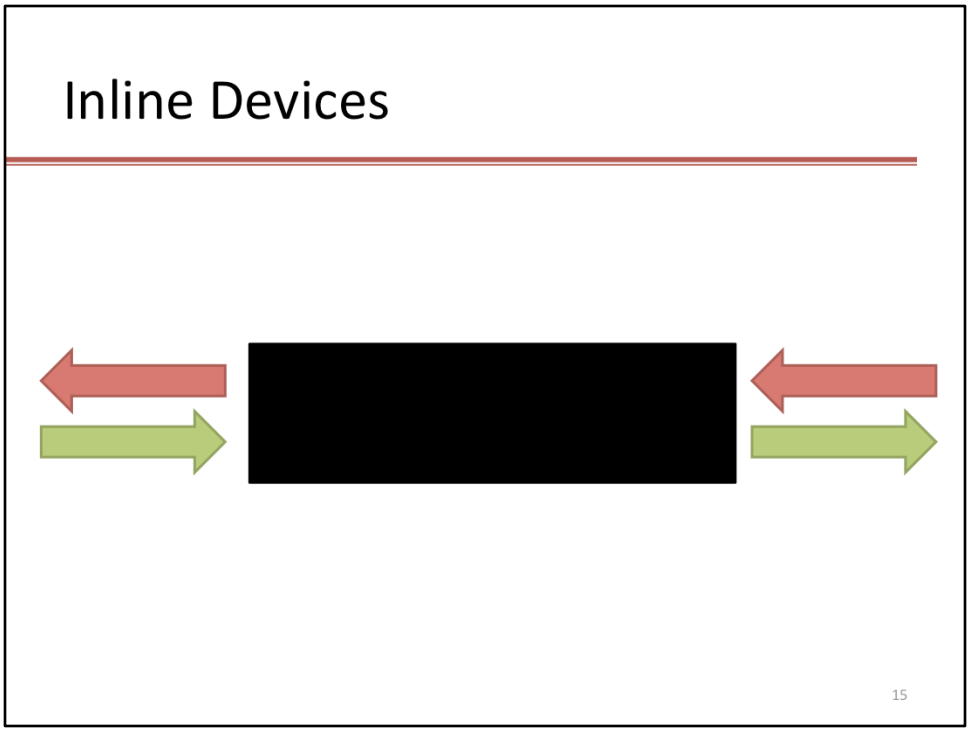
13:01:55.993329 IP 173.194.43.9 > 172.16.191.136: ICMP echo reply, id 27608, seq 357, length 64

```
0x0000:  000c 295c 00be 0050 56f4 b5eb 0800 4500  ..)\...PV.....E.
0x0010:  0054 f70a 0000 8001 ff39 adc2 2b09 ac10  .T.......9..+...
0x0020:  bf88 0000 ac83 6bd8 0165 435b fc50 0000  ......k..eC[.P..
0x0030:  0000 d9bf 0e00 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ..........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
```



"Bump in the wire"
Collection device can't change packets
Cheap compared to inline devices

Where might taps be useful? Why? What are their downsides?



Much more expensive than a tap
Typically programmable and able to change specific packets in specific ways
If it fails it takes the link with it
Bad software can also take the link down

Where would an inline device be useful?
Why would you use it in place of a tap? What are the risks?

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

16

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

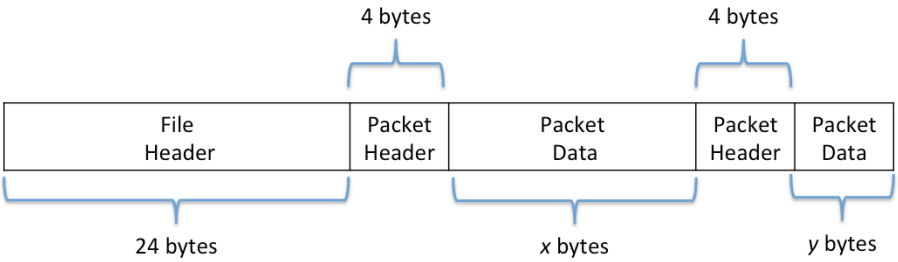
$$N\ \text{pps} \times H_{\text{PCAP}} = NH\ \text{Bps}$$

17

1gbps composed of average packet size 64 bytes

1gbps / (64 * 8 bits) = 10^9 bits/s / 512 bits = 1,953,125 packets per second
1gbps / (1500 * 8 bits) = 10^9 / 12000 = 83,333 pps
1gbps / (9000 * 8 bits) = 10^9 / 72000 = 13888.9 pps

libpcap Format



18

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

20

1gbps / (64 * 8 bits) = 10^9 bits/s / 512 bits = 1,953,125 packets per second
1gbps / (1500 * 8 bits) = 10^9 / 12000 = 83,333 pps
1gbps / (9000 * 8 bits) = 10^9 / 72000 = 13888.9 pps

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

21

Since keeping full PCAP indefinitely is unrealistic, what can you retain indefinitely? How do you get maximum value from very little information?

- examples: DNS, DHCP, first N bytes of each packet, flow data, eliminate data you can never decrypt

BERKELEY PACKET FILTER

Surprisingly Powerful

22

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.

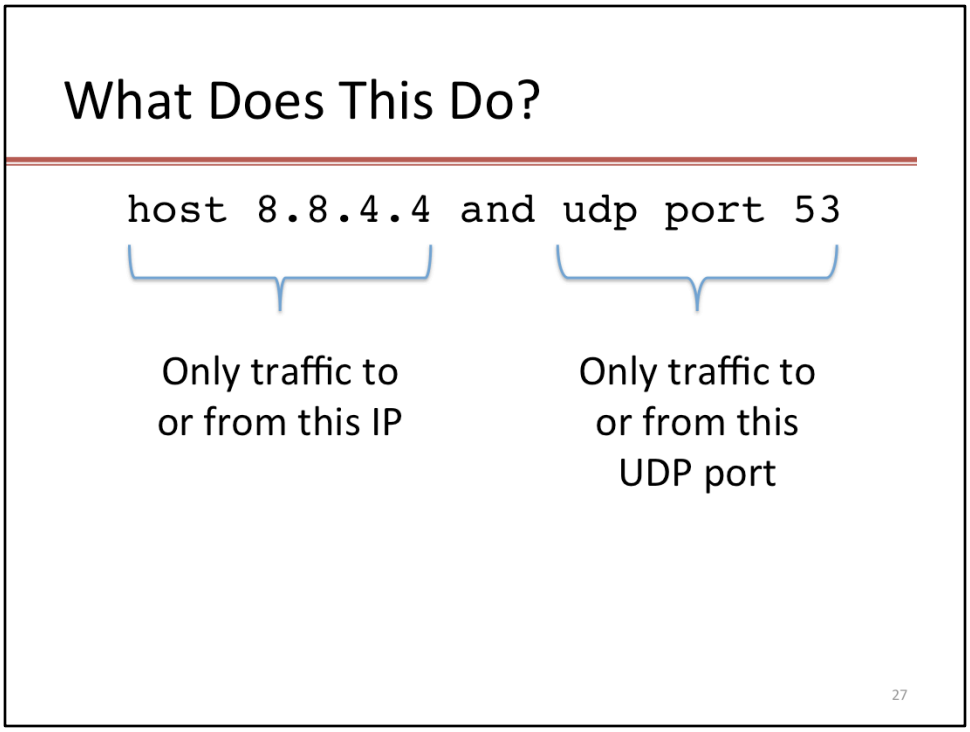
How can you use tcpdump and command line tools to get a rough count of how many messages have been sent?

25

tcpdump –r counting-tcp.pcap -s 0 “host 172.16.0.2 and tcp” 2>/dev/null | wc -l

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

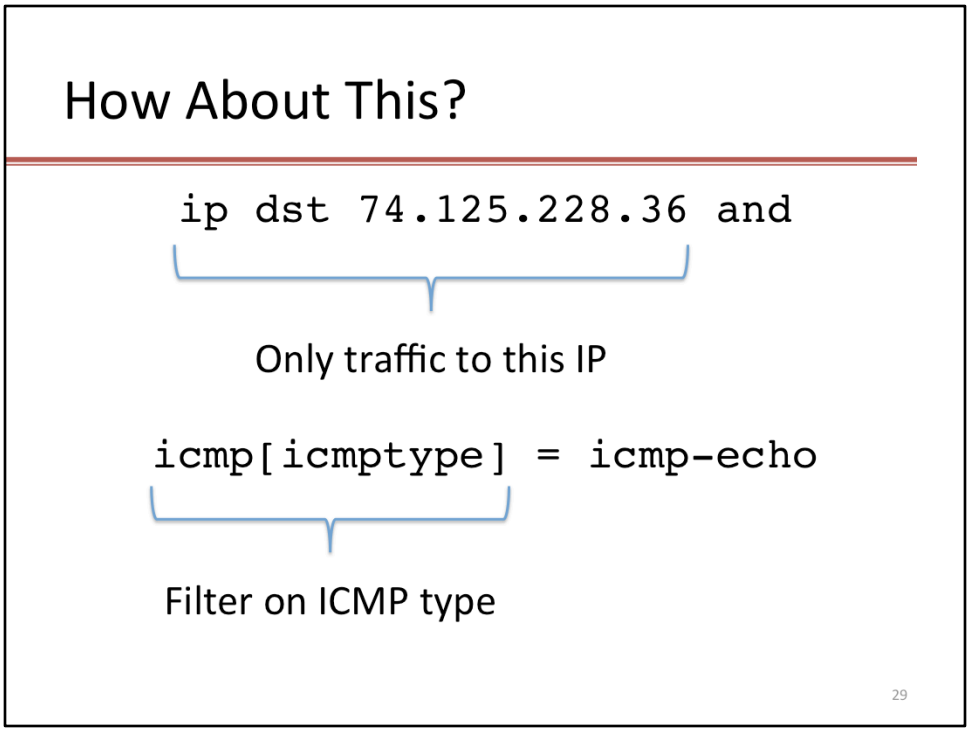


Shows DNS requests and responses to or from 4.4.2.2. Unless you're Google, this will show you DNS traffic to and from your computers being answered by Google public DNS.

How About This?

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

28

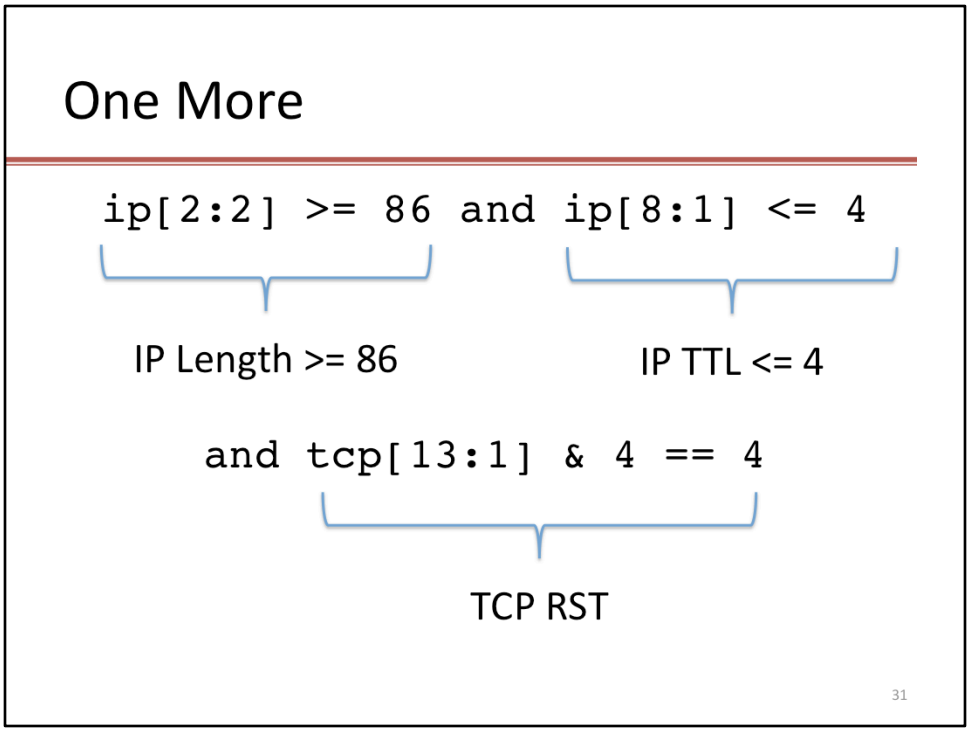


ICMP Echo Request (ping) packets with IP destination 74.125.228.36 (resolves to google.com as of 29 December 2012). Will you see Echo Reply packets with this filter?

One More

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

30



Why does logical & with 4 work for TCP RST?
This could be written much more easily: “greater 100 and ip[8:1] <= 4 and tcp[tcpflags] & tcp-reset == 1”

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.

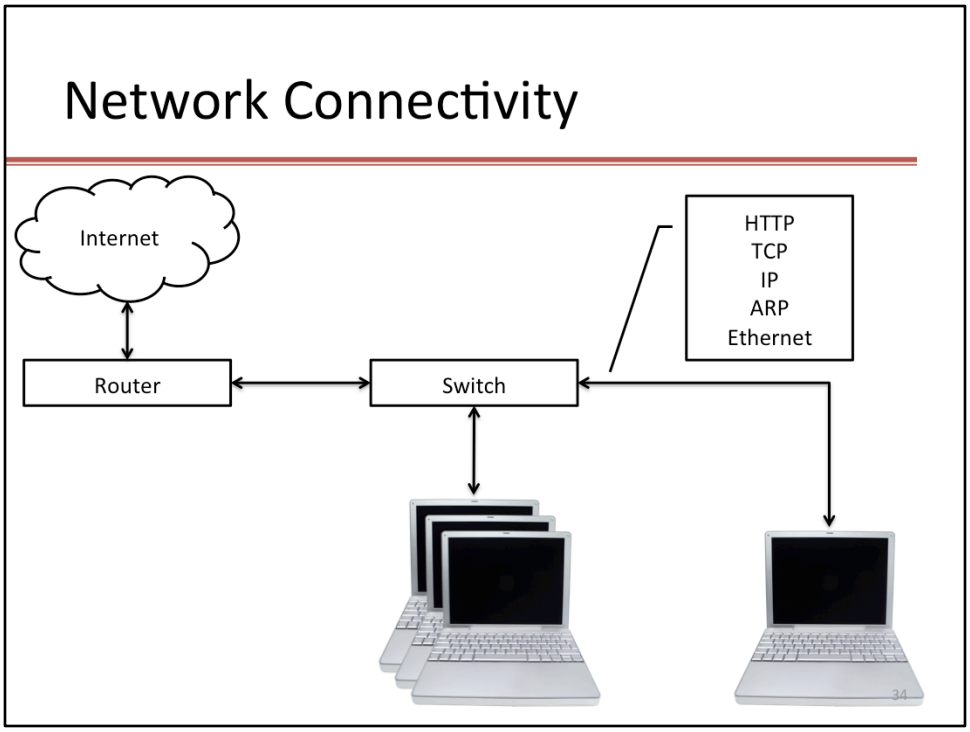
32

tcpdump -n -r alert.pcap -XX 'host 172.16.191.1 and ((tcp port 80 and tcp[tcpflags] & 0x12 = 0x12) or (udp port 53 and greater 200))'

ADDRESS RESOLUTION PROTOCOL

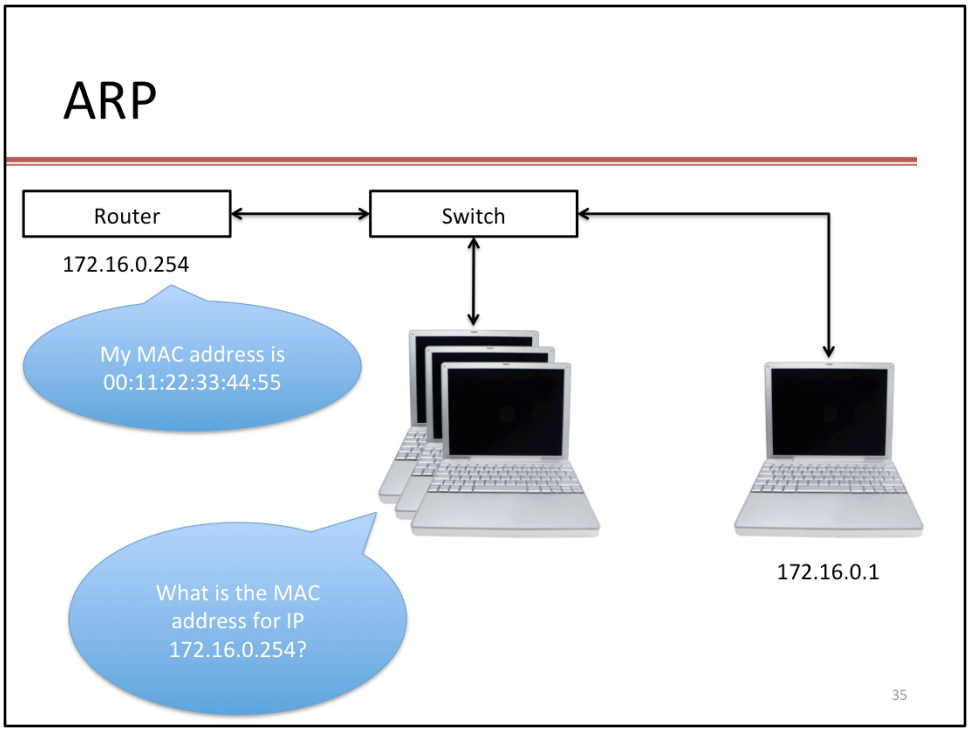
Is It Plugged In?

33



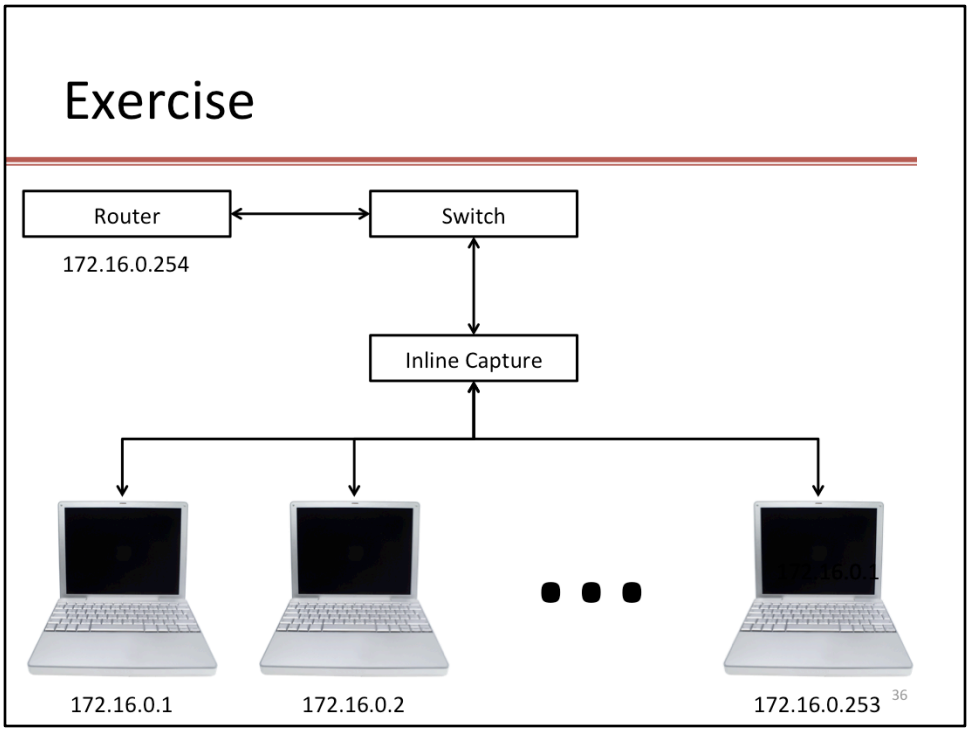
Where could problems occur in here at each layer of the OSI model?

- layer 1: signaling problems
- layer 2: ARP poisoning
- layer 3: DHCP problems
- layer 4: dropped packets, congestion
- layer 5: not too much ;)



Where could problems occur in here at each layer of the OSI model?

- layer 1: signaling problems
- layer 2: ARP poisoning
- layer 3: DHCP problems
- layer 4: dropped packets, congestion
- layer 5: not too much ;)



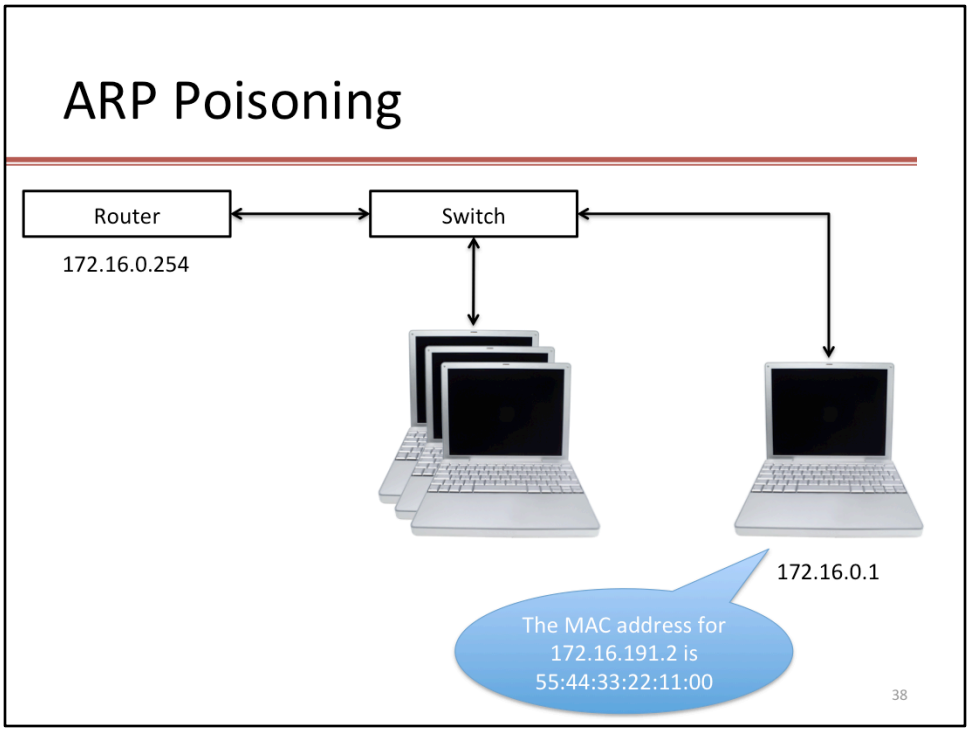
Where could problems occur in here at each layer of the OSI model?

- layer 1: signaling problems
- layer 2: ARP poisoning
- layer 3: DHCP problems
- layer 4: dropped packets, congestion
- layer 5: not too much ;)

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

37



Where could problems occur in here at each layer of the OSI model?

- layer 1: signaling problems
- layer 2: ARP poisoning
- layer 3: DHCP problems
- layer 4: dropped packets, congestion
- layer 5: not too much ;)

ARP Poisoning

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

39

HTTP

Hypertext Transfer Protocol

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

- Ubiquitous
- Many uses

41

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.

42

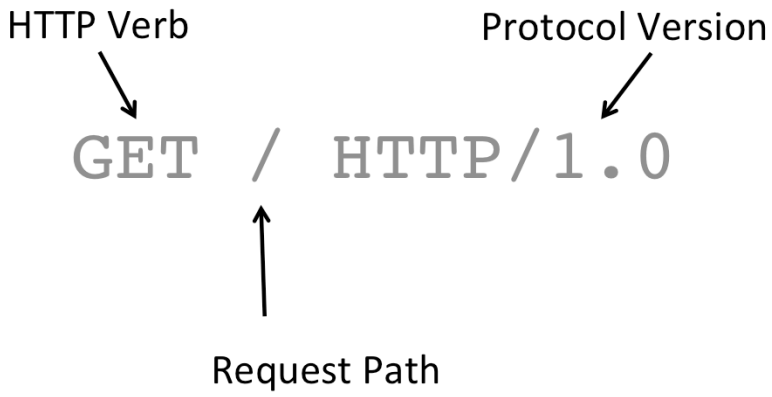
Arp133.pcap
Arp136.pcap
Arp138.pcap

HTTP

Header 1
Header 2
Header 3
Header *n*
Body

43

Headers



44

Headers

45

Example

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

46

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?

48

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

49

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?

50

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?

51

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.

52

html.pcap

tcpdump -r html.pcap -nn -w - |strings|grep -E '<title>.*</title>'|sed -r 's/<[/]*title>//g'|sort -u

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!

53

Looking for unique hosts, not URI paths

tcpdump -r websites .pcap -n tcp port 80 -w - |strings|grep 'Host: '|sort -u

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

55

Chopshop

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

56

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

58

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

59

Simple Obfuscation

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

60

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

61

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

62

In Other Words

One or the other,

but not both.

63

Due to: http://en.wikipedia.org/wiki/Exclusive_or

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

64

XORcise

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

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

65

Need to download PCAP – not included in VM

File Carving From HTTP

- Demo: Wireshark
- Exercise: chopshop “http_extractor”

66

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

69

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

70

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

71

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?

72

Unknown-1.pcap is actually naked SSL (SSL outside the context of e.g., HTTP) on port 22. Wireshark will assume that it's SSH. You can tell that it's SSL by looking at the first six bytes sent from client to server or by observing the SSL certificate.

Unknown-2.pcap is a very simple message transmitted in plaintext with a little endian DWORD header that encodes the length of the message that follows.

Unknown-3.pcap is actually the JackCR dfir challenge

DNS

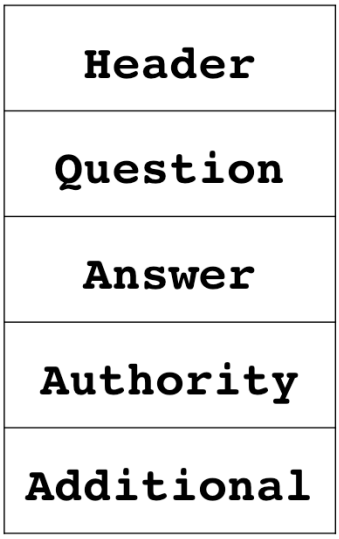
Domain Name System

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

74

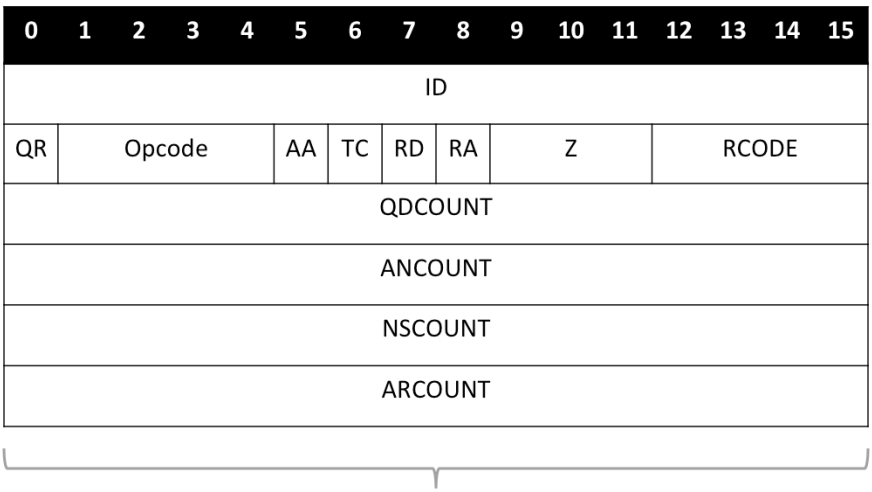
Message Format



75

See RFC 1035

Header



See RFC 1035

- Header format shared for requests and responses
- ID is a random 16 bit int
- QR is set to 0 for queries and 1 for responses

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
```

77

This is a DNS response for xkcd.com

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

78

Dns-tcpdump.pcap

Solution

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

79

Can check that you have the correct response by checking the DNS ID printed by tcpdump

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

81

There was an error on the slides; "-r" should be "-f"

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

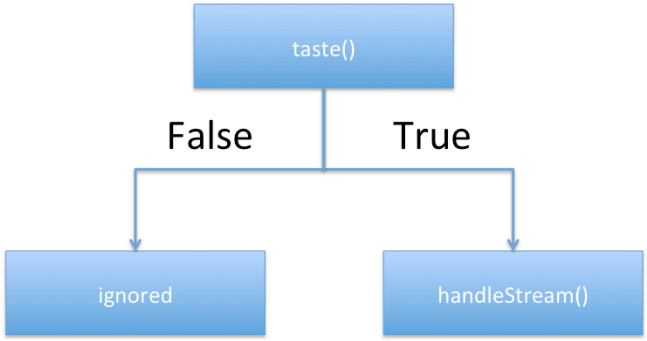
82

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



85

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

88

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

89

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

90

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!