Packet Analysis

v 4.0.0

Why Packets?

- Allows complete reconstruction of network events
- Contains all the raw data
- Allows humans to analyze and tell the machine what to do

Size of Full PCAP Storage

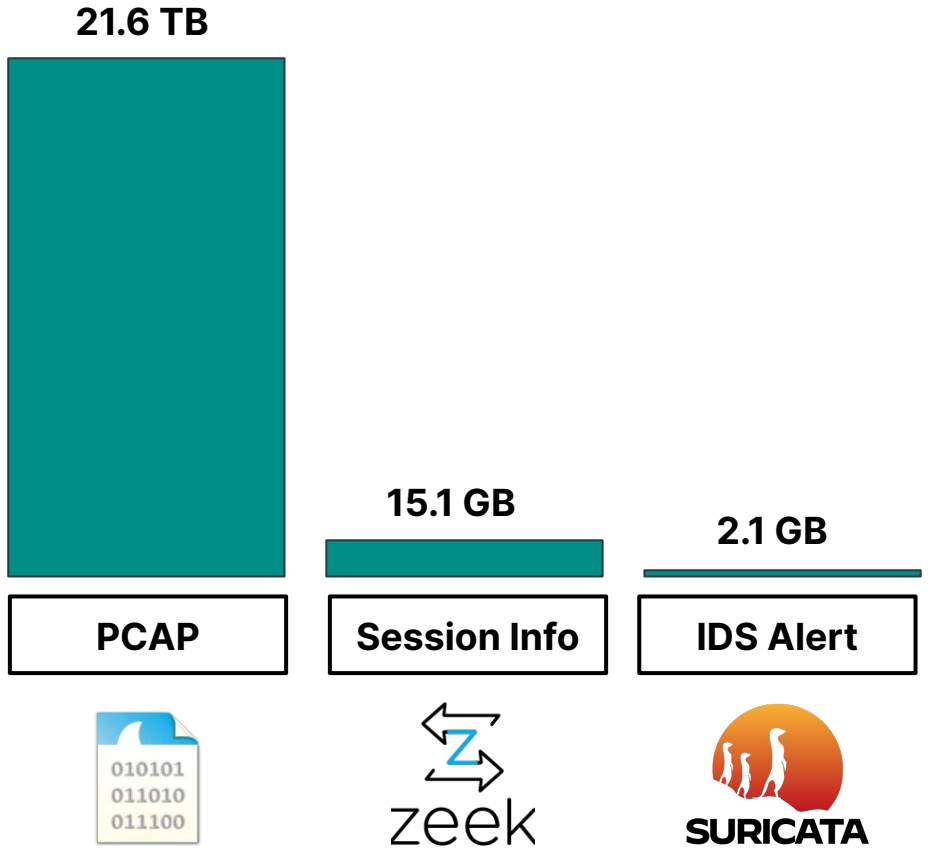
How much storage would be needed to monitor the following network?

- 1 Gbps link speed
- Fully saturated
- 24 Hours of traffic

Size of Full PCAP Storage

How much storage would be needed to monitor the following network?

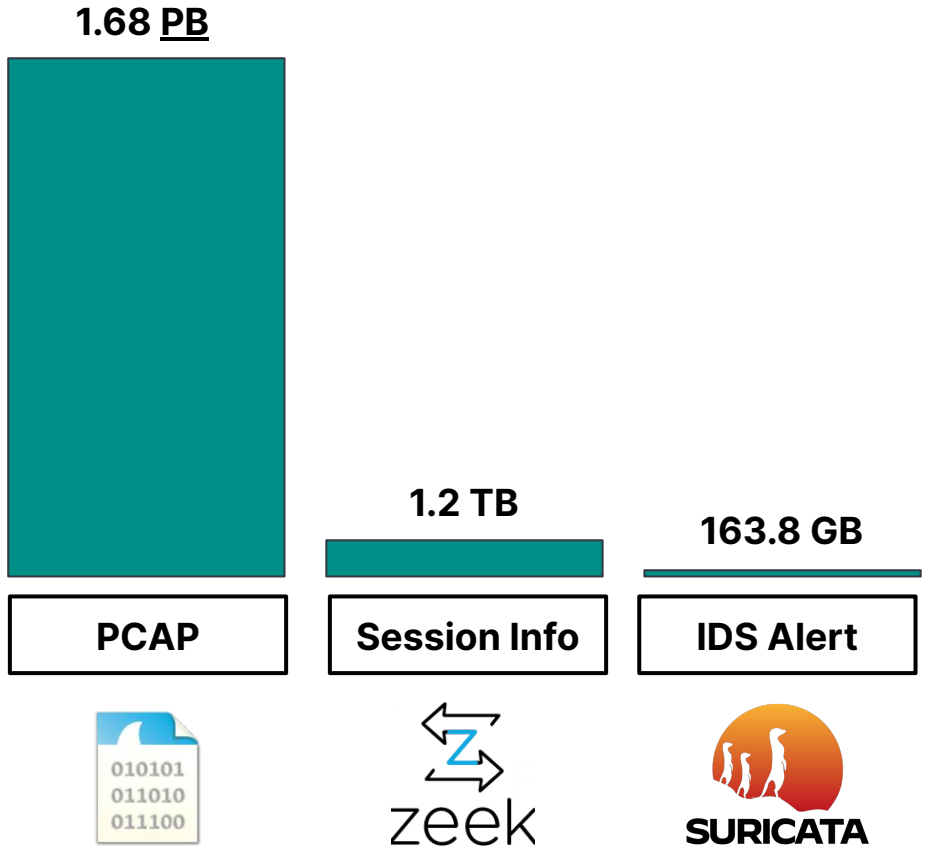
- 1 Gbps link speed
- Fully saturated
- 24 Hours of traffic



Size of Full PCAP Storage - Average Dwell Time

The average **dwell time** of an attacker is **78 days**.

Based on that information, how much storage is required for adequate retention?



elastic

PCAP Math

$$\text{Storage} = 2 \times \frac{B_{Avg} \times 86{,}400}{8}$$

- 1 day @ saturated 1 Gbps = 21,600 GB file storage
- B_{Avg} is the average link speed

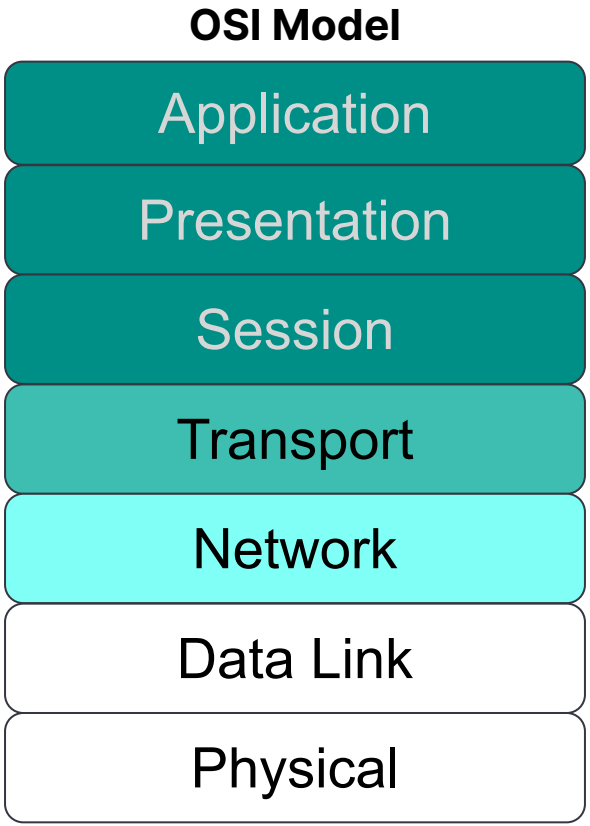
Packet Analysis

1. TCP/IP Model
2. Protocol Basics
3. Wireshark
4. TCPdump/BPF
5. Stenographer/Docket

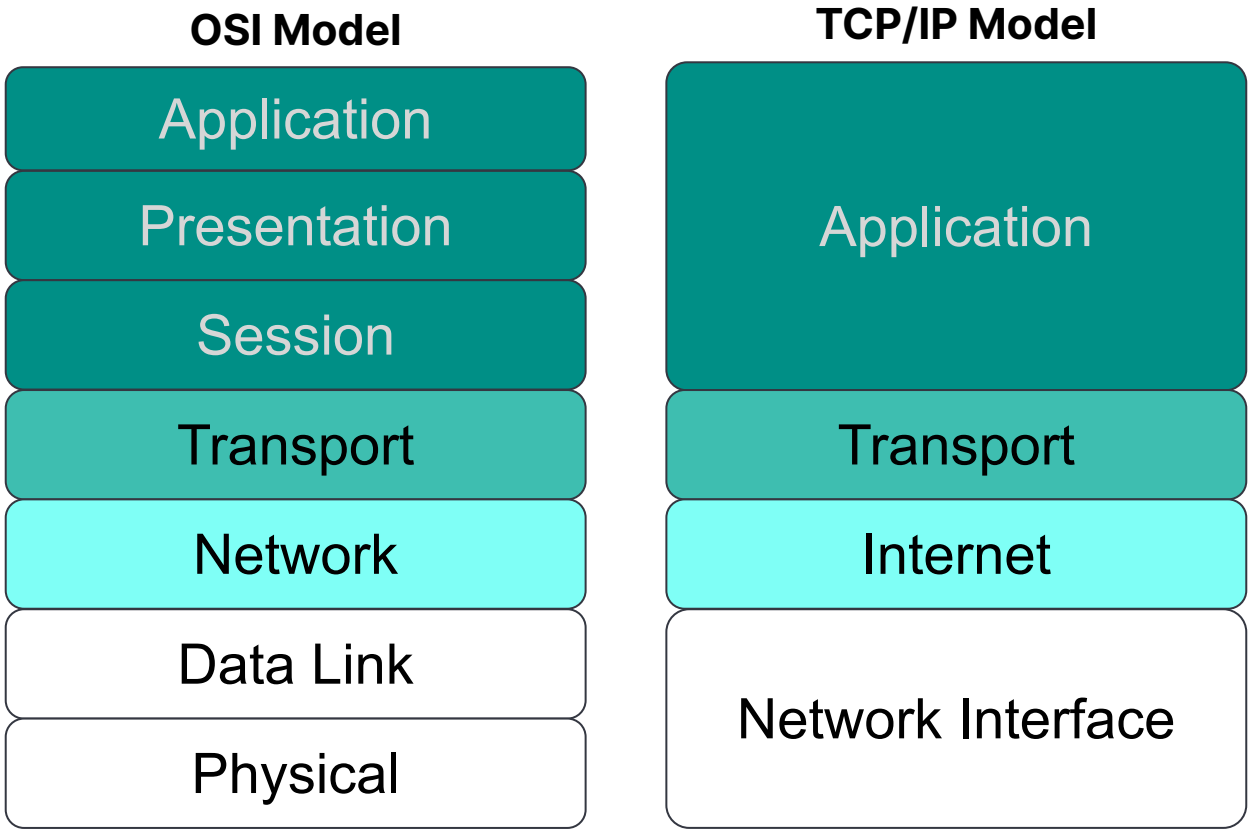
Packet Analysis

1. TCP/IP Model
2. Protocol Basics
3. Wireshark
4. TCPdump/BPF
5. Stenographer/Docket

OSI Model Conceptual Layer



TCP/IP Conceptual Layer

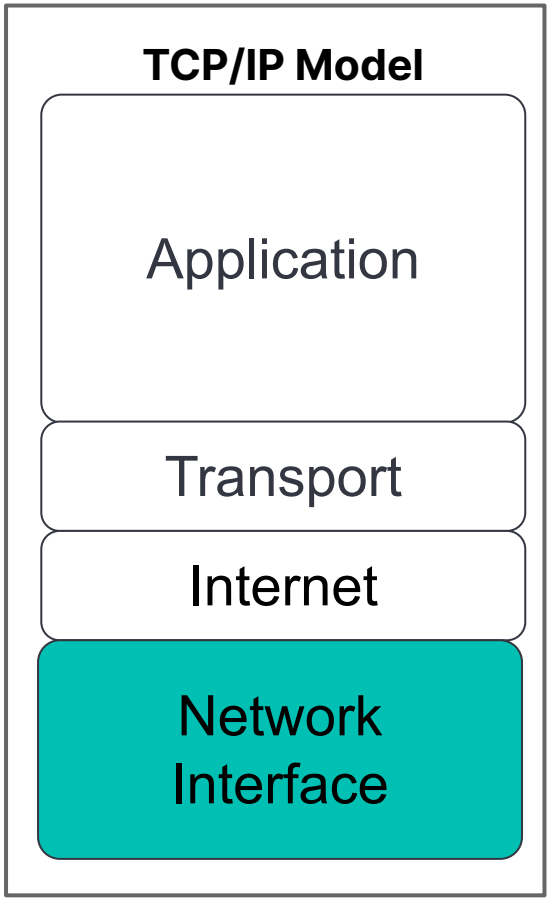


Network Interface

- Manages the electrical signal that represents data transfer
- Sends data from one machine to another based physical (MAC) address

Protocols: Ethernet (Physical), Ethernet (Logical), Bluetooth, USB, Wifi, MAC, LLDP

Devices: CAT5 Cable, Hubs, Repeaters, Switches

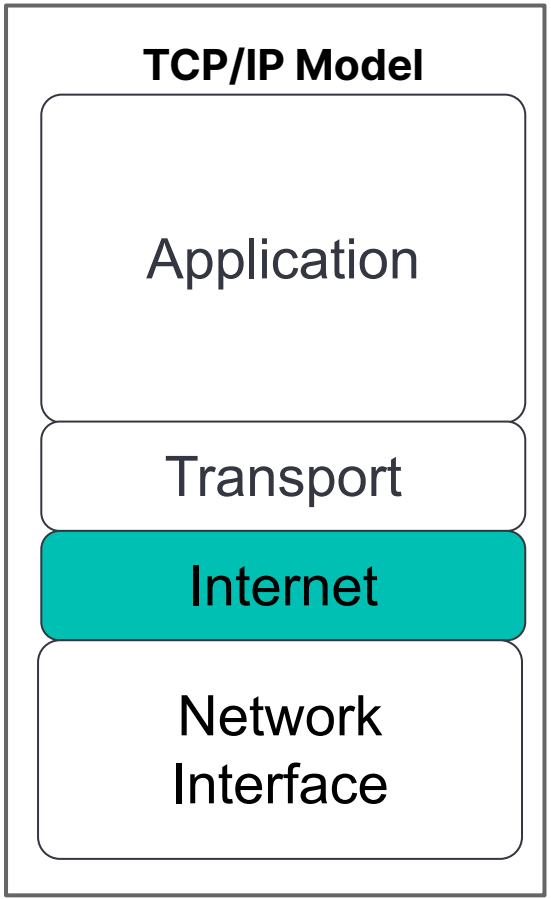


Internet Layer

- Routes packets to the next-hop host and passes link layer info
- Manages data routing paths and protocols

Protocols: IPv4, IPv6, ICMP, RIP, OSPF

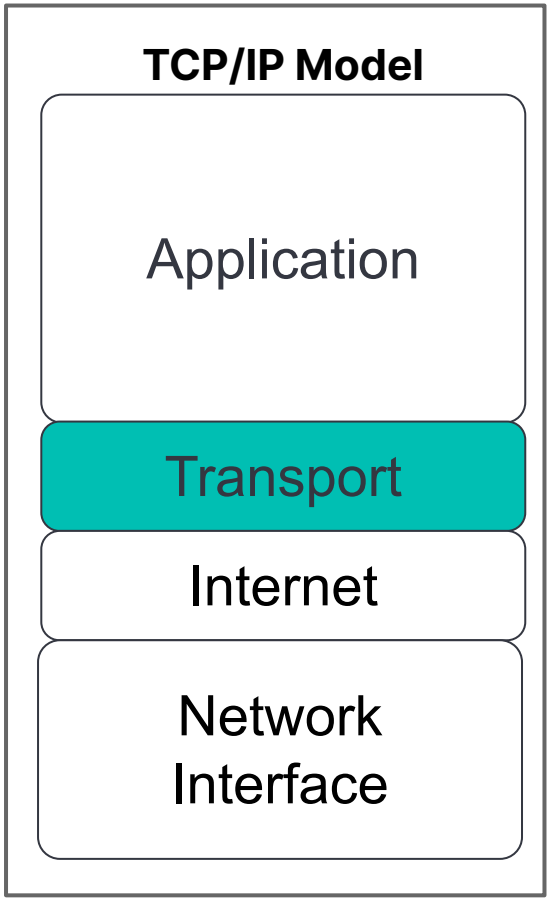
Devices: Routers, Gateways



Transport Layer (4)

- Operates over port numbers
- Connection-oriented
- Flow control

Protocols: TCP, UDP, iSCSI

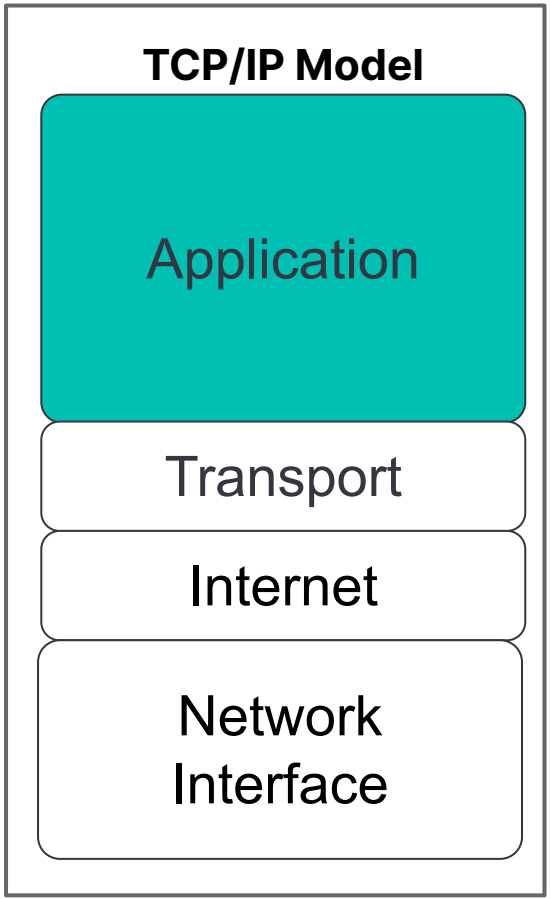


Application

- Data intended for final destination software
- Users are most likely to interact with this layer

Protocols: HTTP, IMAP, DNS, DHCP, SSL/TLS, SMB

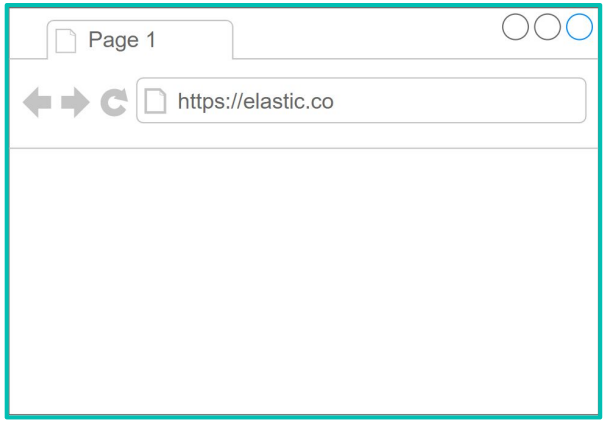
Devices: Web Application Firewall, Layer 7 Switch



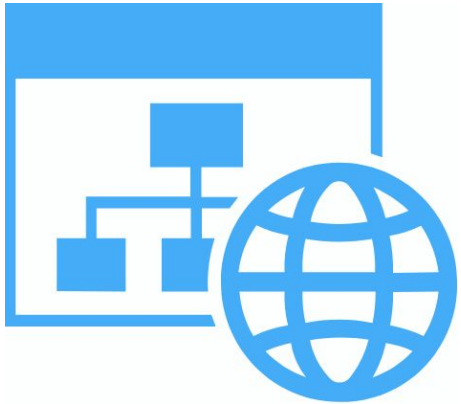
Journey of a Packet

- Primary goal is to get data from one application to another
- These two pieces of software know how to communicate with each other

Browser Software

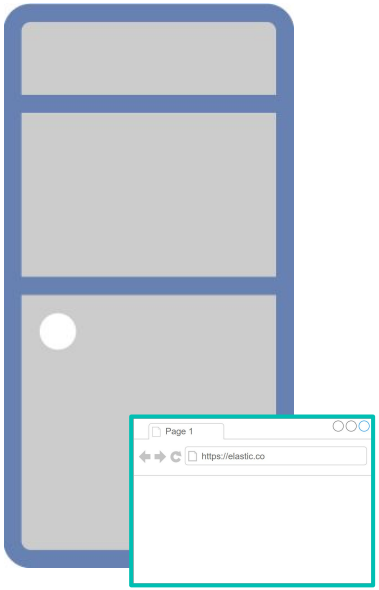


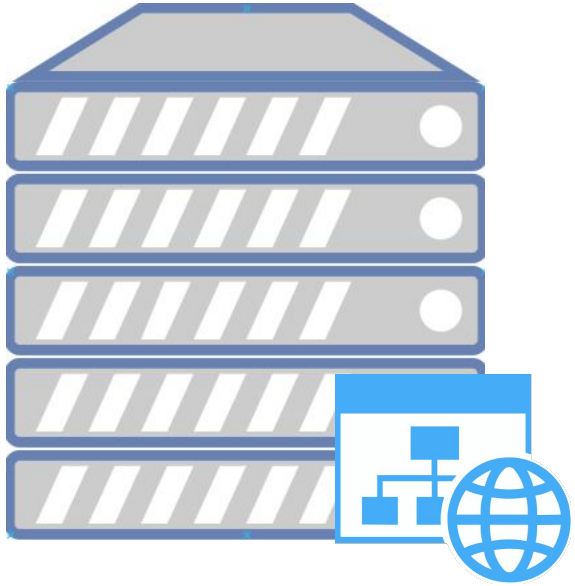
Web Server Software



Journey of a Packet

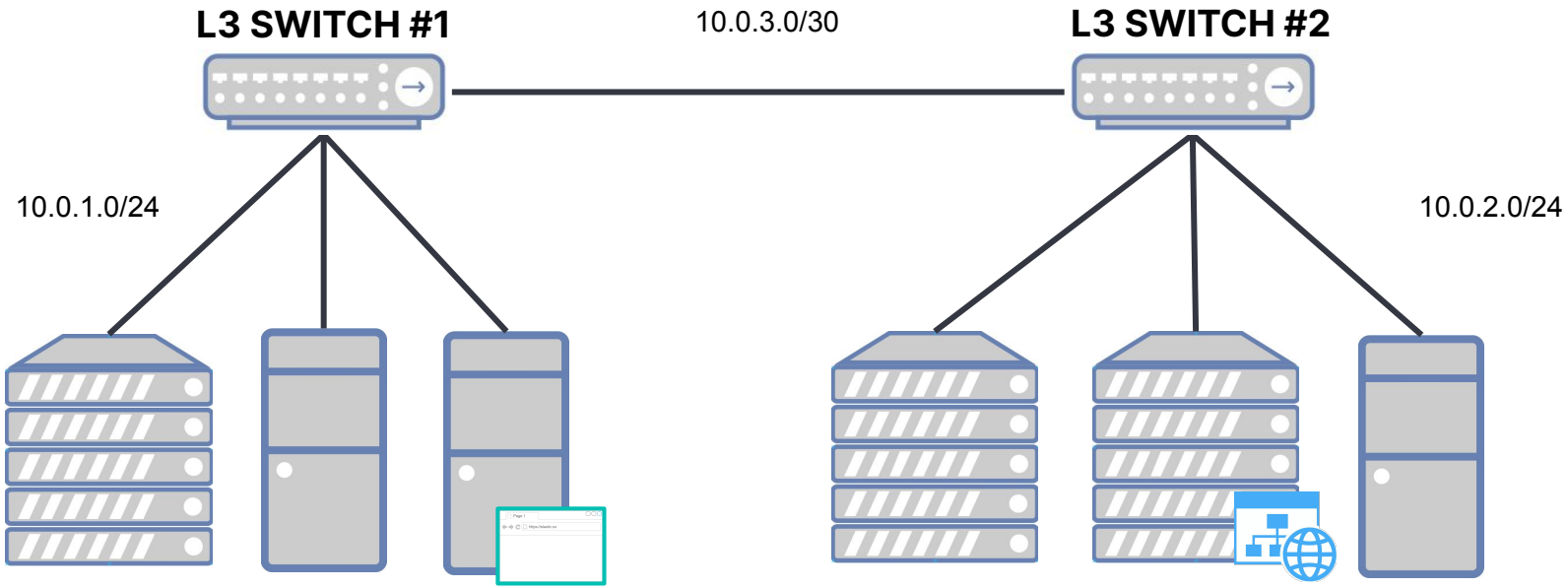
Unfortunately the software is usually not on the same machine





Journey of a Packet

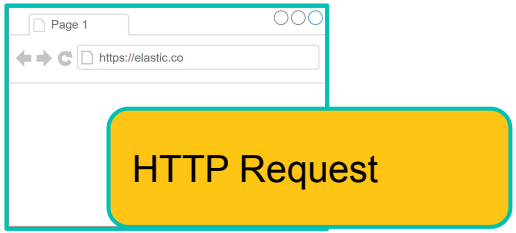
The systems also aren't directly connected to one another



elastic

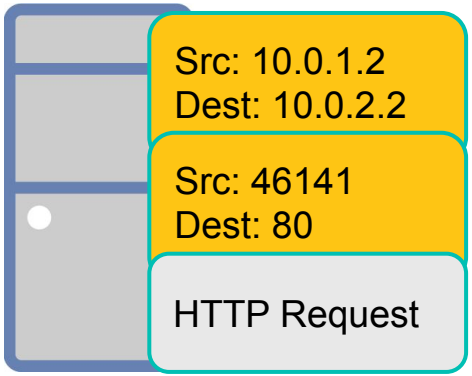
Journey of a Packet

1



The application (browser) creates a payload intended for the web server

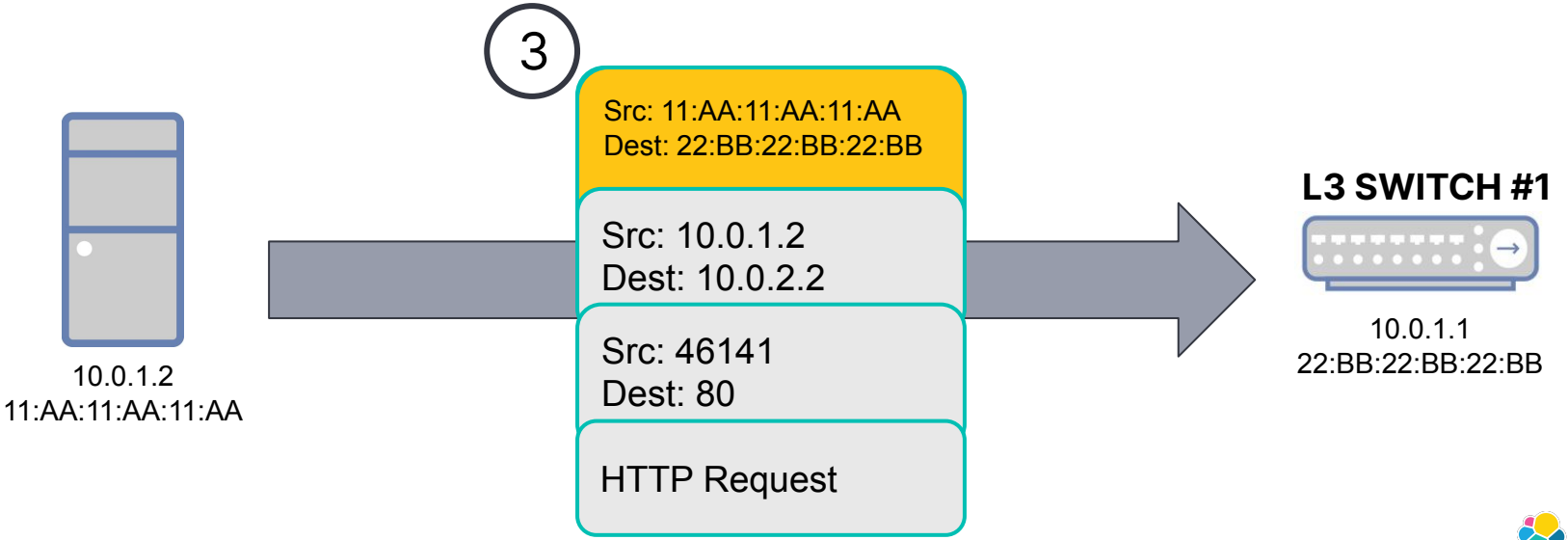
2



The host then adds port and IP information indicating the final destination and where to send data back

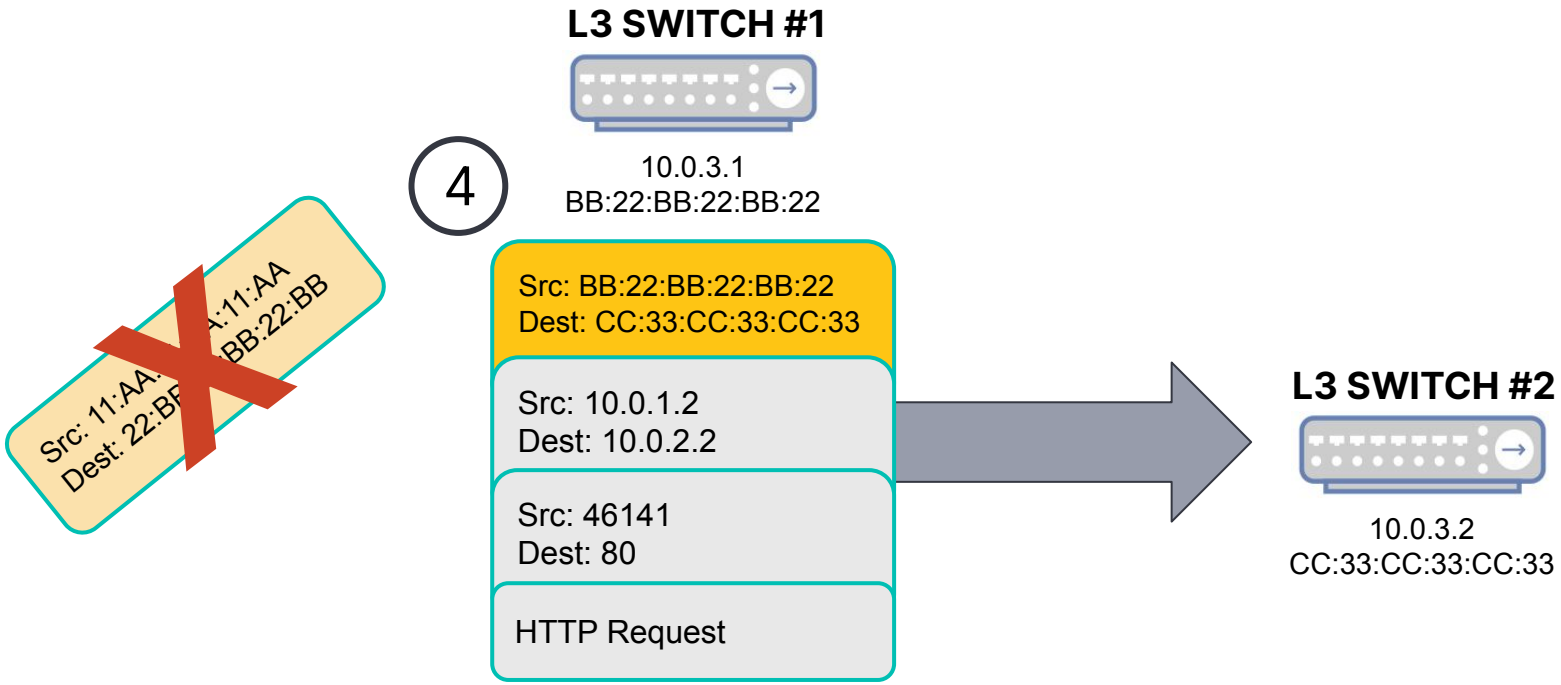
Journey of a Packet

- The packet can't skip other devices and magically appear on the web server so we use the network interface layer to help it hop through devices
- Because the destination host exists outside the client's subnet, the MAC address of the default gateway is used for the destination MAC



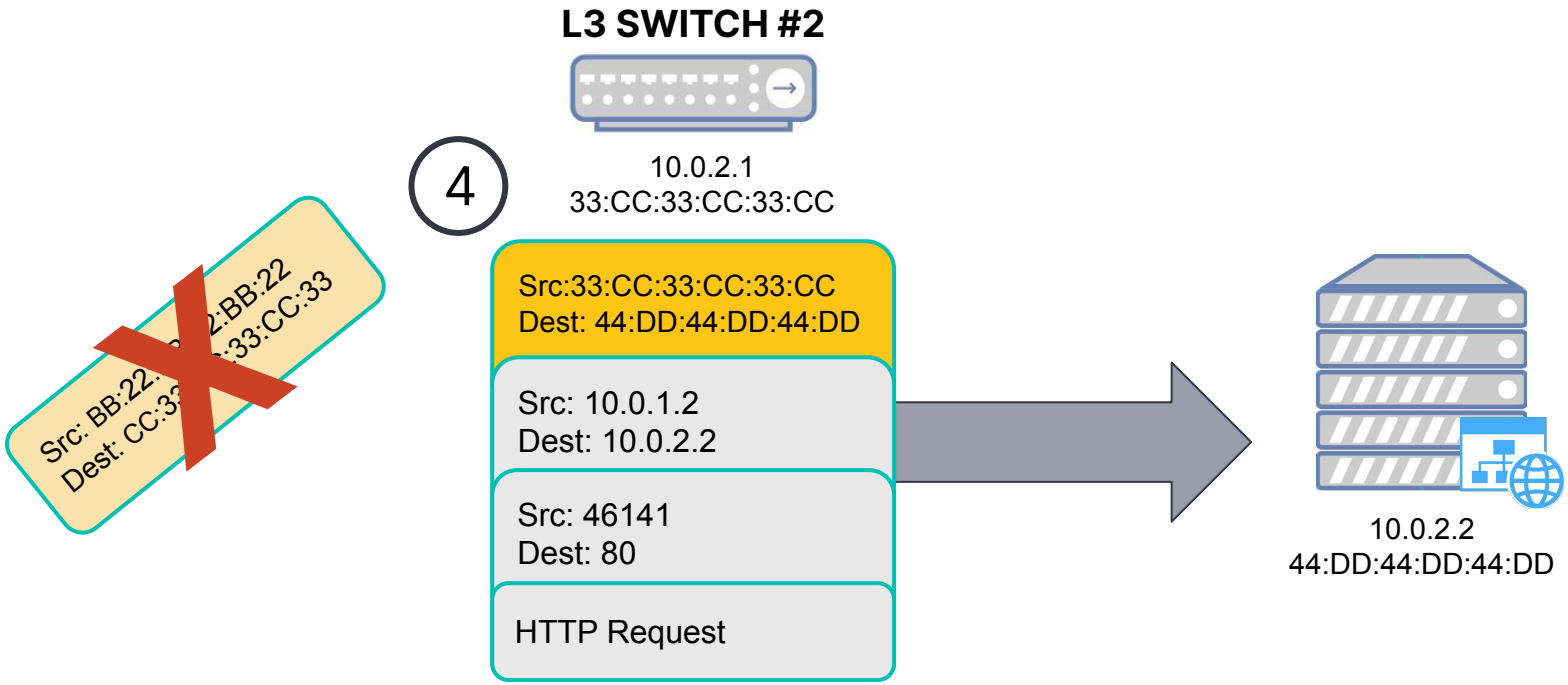
Journey of a Packet

- The first L3 switch checks its route table for the destination 10.0.2.2
- Then the switch replaces the Ethernet Header for the next hop L3 switch



Journey of a Packet

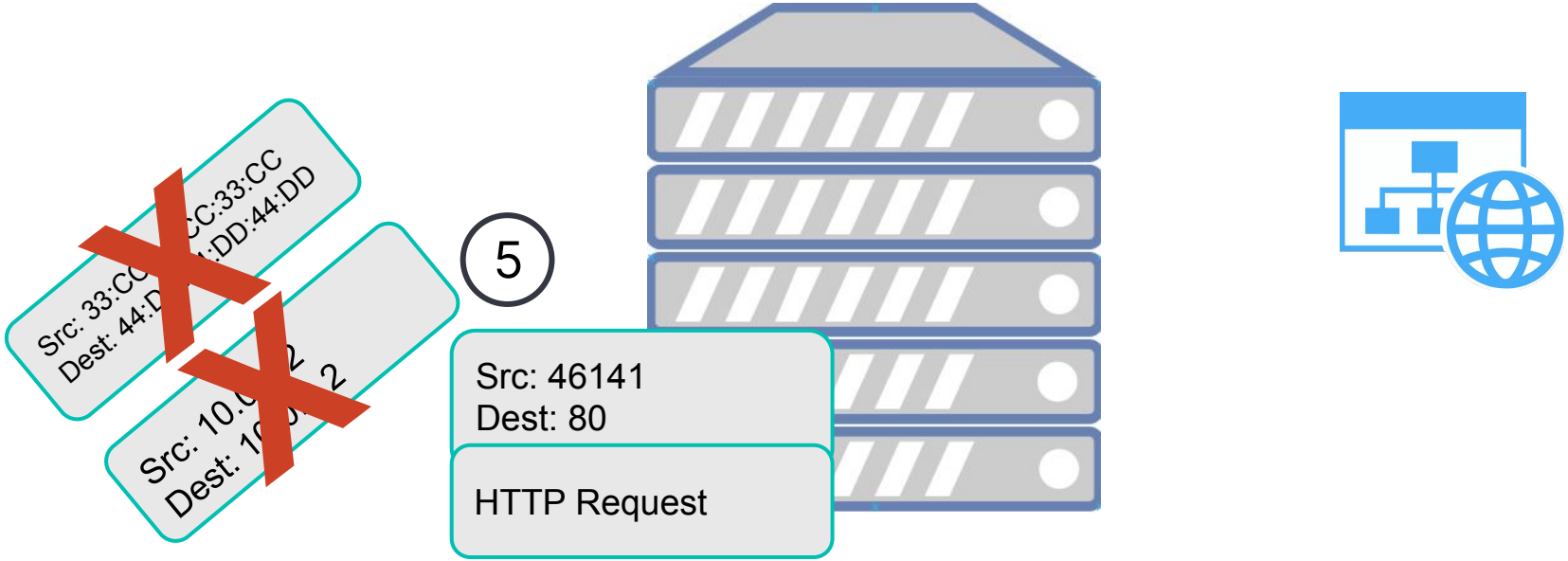
- The switch replaces the Ethernet Header with the correct info for the next hop
- The destination is the MAC of the destination web server



elastic

Journey of a Packet

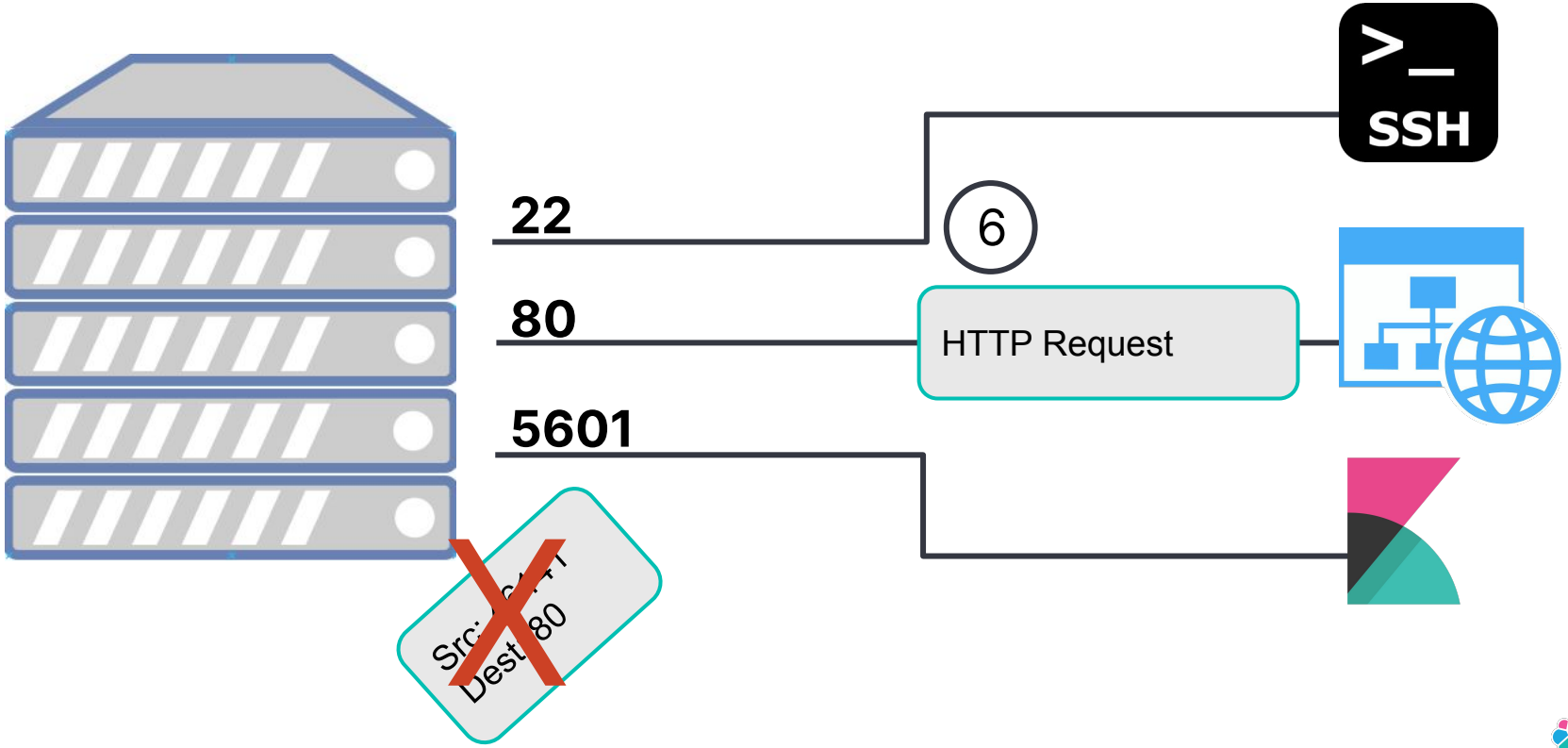
- The packet has now reached the host and can strip the previous layers
- A response packet would follow this same process in reverse



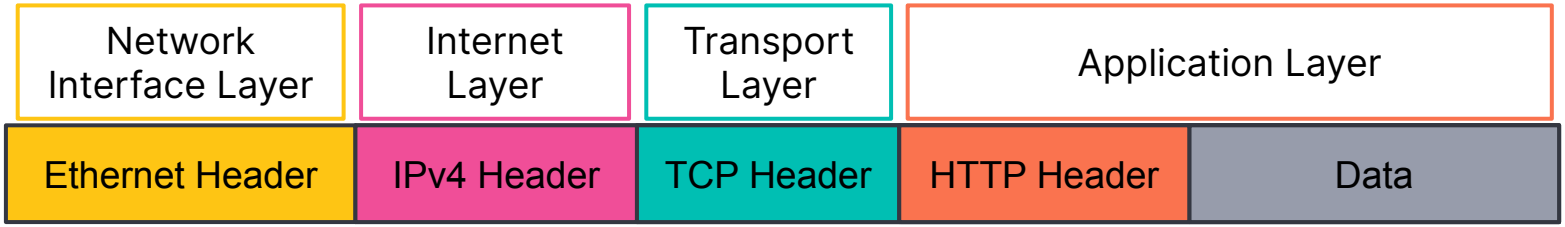
elastic

Journey of a Packet

- The port number within the transport layer directs packets to the correct application



Packet Encapsulation - Layer Perspective



CTF: TCP/IP

Packet Analysis

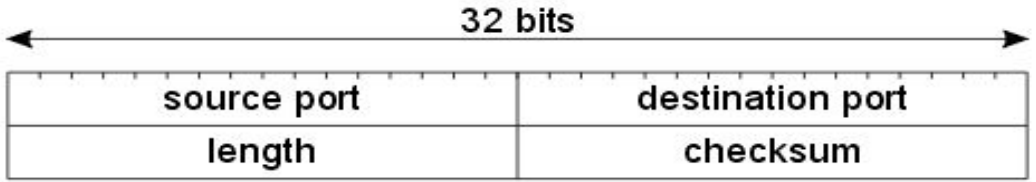
1. TCP/IP Model
2. **Protocol Basics**
3. Wireshark
4. TCPdump/BPF
5. Stenographer/Docket

Header Diagrams Explained

Raw Packet (Binary Data)

0100110110101000000000000011011111011101011010000100110110101000

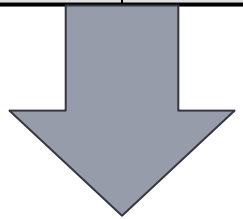
Protocol Header Diagram (Decoder Key)



Header Diagrams Explained

Apply Protocol Header to Raw Data

Source Port	Destination Port
0100 1101 1010 1000	0000 0000 0011 0111
Length	Checksum
1101 1101 0110 1000	1100 1101 1010 1000



Convert from Binary

Source Port	Destination Port
19880	53
Length	Checksum
56680	52648

Ethernet II Frame

	0	1	2	3	4	5	6	7
0	MAC Destination						MAC -->	
8	--- Source				VLAN Tag (optional)			
16	Type/Length				**DATA (Payload)**			
--	Frame Check Sequence							

0×0800	Internet Protocol version 4 (IPv4)
0×0806	Address Resolution Protocol (ARP)
0×86DD	Internet Protocol version 6 (IPv6)

MAC (Hardware) Address

Network **I**nterface **C**ard

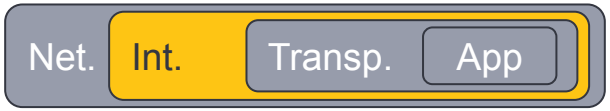
00 : 14 : 22 : 01 : 23 : 45

Organizationally
Unique
Identifier

NIC Specific

OUI Examples
FC:FC:48 Apple
B4:E9:B0 Cisco
00:50:56 VMware

IPv4 Header



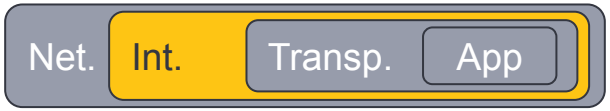
	0	1	2	3	4	5	6	7
0	Ver | IHL	TOS	Total Length		IP Identification		X/D/M | Offset	
8	TTL	Protocol	Checksum		Source Address			
16	Destination Address				Options (optional)			

Time To Live

- Value that decrements at each router hop.
- Keeps packets from looping forever
- Can be used to fingerprint OS

1	ICMP
6	TCP
8	EGP
17	UDP

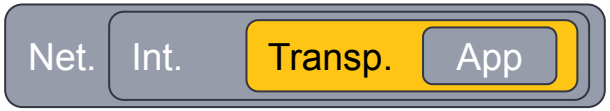
ICMP(v4) Header



	0	1	2	3
0	Type	Code	Checksum	
4	Additional Information (dependent of type/code)			

Type		Code	
0	Echo Reply	0	No Code
3	Destination Unreachable	0	Net Unreachable
		1	Host Unreachable
		2	Protocol Unreachable
		3	Port Unreachable
		...	...
8	Echo	0	No Code
11	Time Exceeded	0	Time to Live exceeded in Transit
		1	Fragment Reassembly Time Exceeded

TCP vs UDP



TCP

	0	1	2	3	4	5	6	7
0	Source Port		Destination Port		Sequence Number			
8	Acknowledgement Number				HL | R	Flags	Window Size	
16	Checksum		Urgent Pointer		Options (up to 40 bytes)			

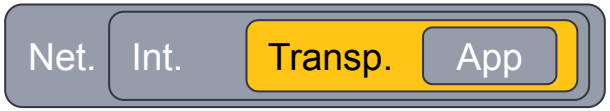
- Connection based
- Flow Control
- File transmission, Email

UDP

	0	1	2	3	4	5	6	7
0	Source Port		Destination Port		Length		Checksum	

- Connection-less
- Fire and forget
- Streaming A/V, Gaming, Small Request/Response

TCP Header



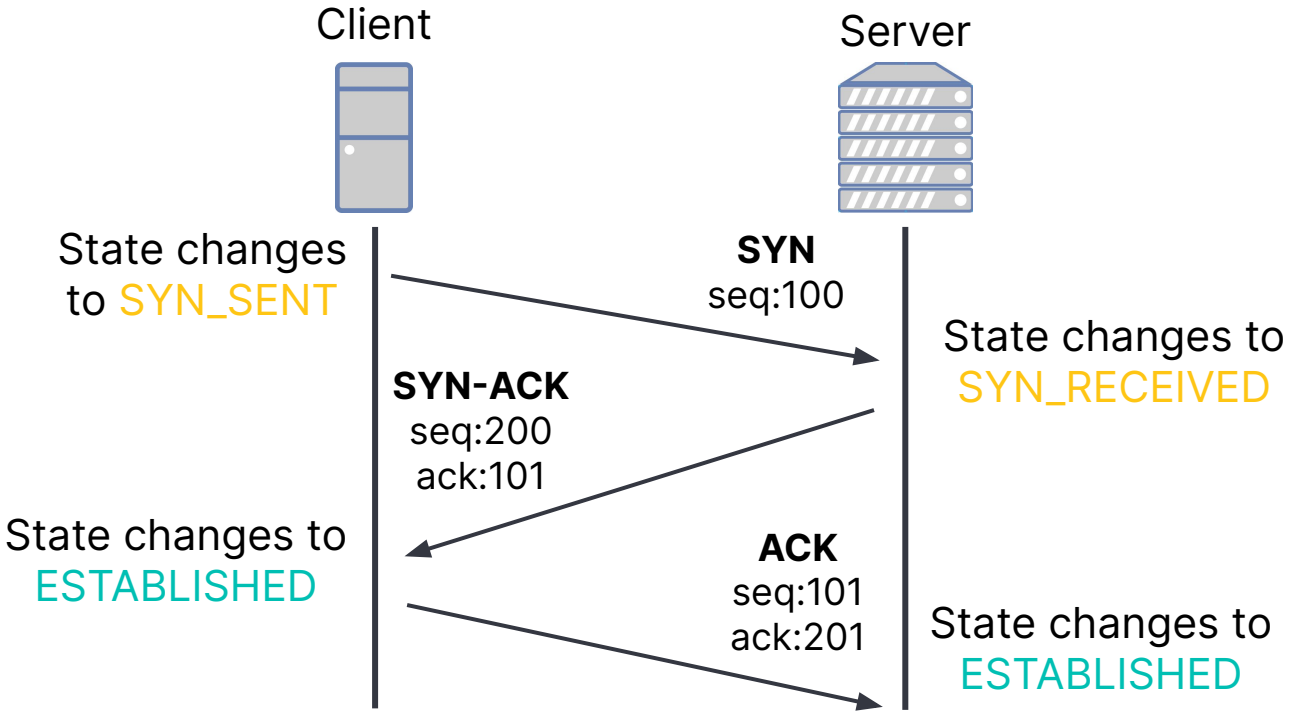
	0	1	2	3	4	5	6	7
0	Source Port		Destination Port		Sequence Number			
8	Acknowledgement Number				HL | R	Flags	Window Size	
16	Checksum		Urgent Pointer		Options (up to 40 bytes)			

CWR	ECE	URG	ACK	PUSH	RST	SYN	FIN
0	0	0	1	0	0	1	0
0x1				0x2			

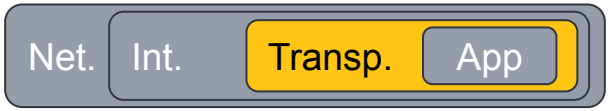
0x12

elastic

TCP 3-Way Handshake



UDP Header

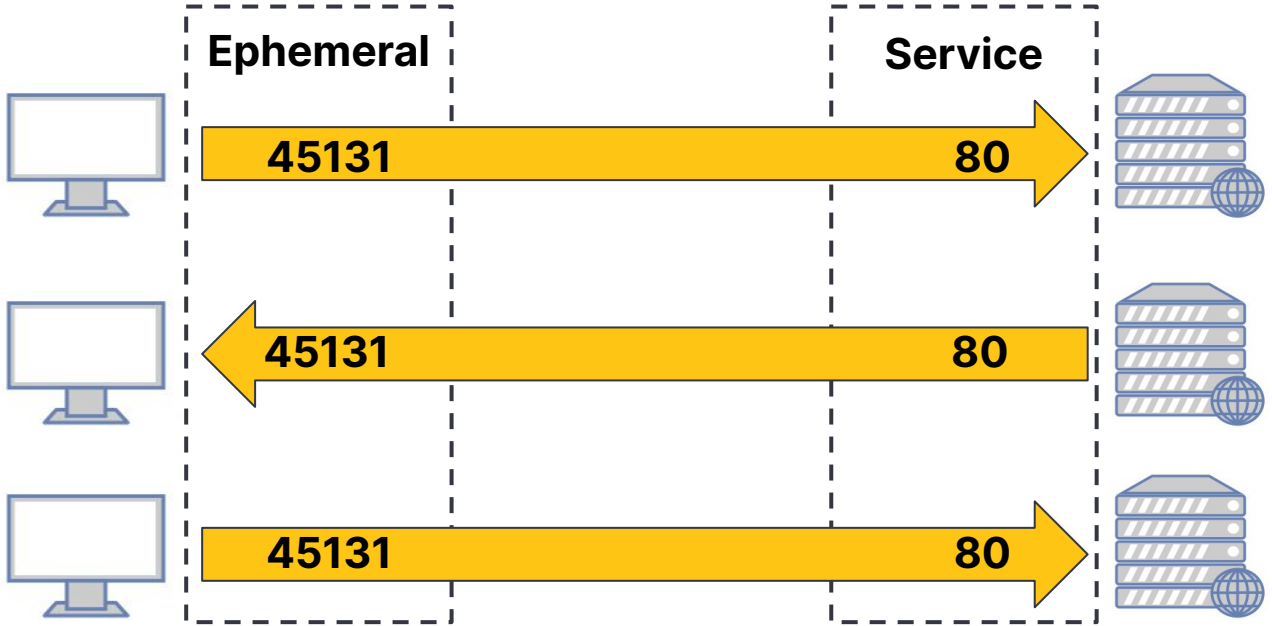


	0	1	2	3
0	Source Port		Destination Port	
4	Length		Checksum	

elastic

Ephemeral Ports

- Ephemeral ports are used by clients as source ports as they reach out to server ports
- Higher numbers are set aside to be used as ephemeral ports



Ephemeral Ports

- Short-lived transport protocol port
- Ephemeral Port ←→ Service Port

Standard?	Ports
IANA Suggestion	49152 - 65535
“Modern” Windows (Vista, 7, Server 2008 and up)	49152 - 65535
Windows Server 2008	1025 - 60000
“Older” Windows (XP and older)	1025 - 5000
Most Linux kernels	32768 - 61000

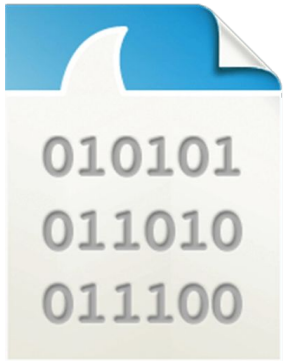
Packet Analysis

1. TCP/IP Model
2. Protocol Basics
3. **Wireshark**
4. TCPdump/BPF
5. Stenographer/Docket

Wireshark

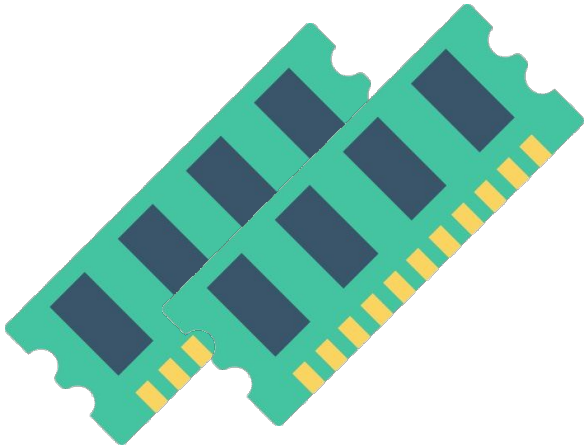
- Great GUI tool to deep dive into packets
- Limited usability on larger pcap files
- Parses over 2,000 network protocols
- Contains 180,000+ fields

Wireshark Resource Cost



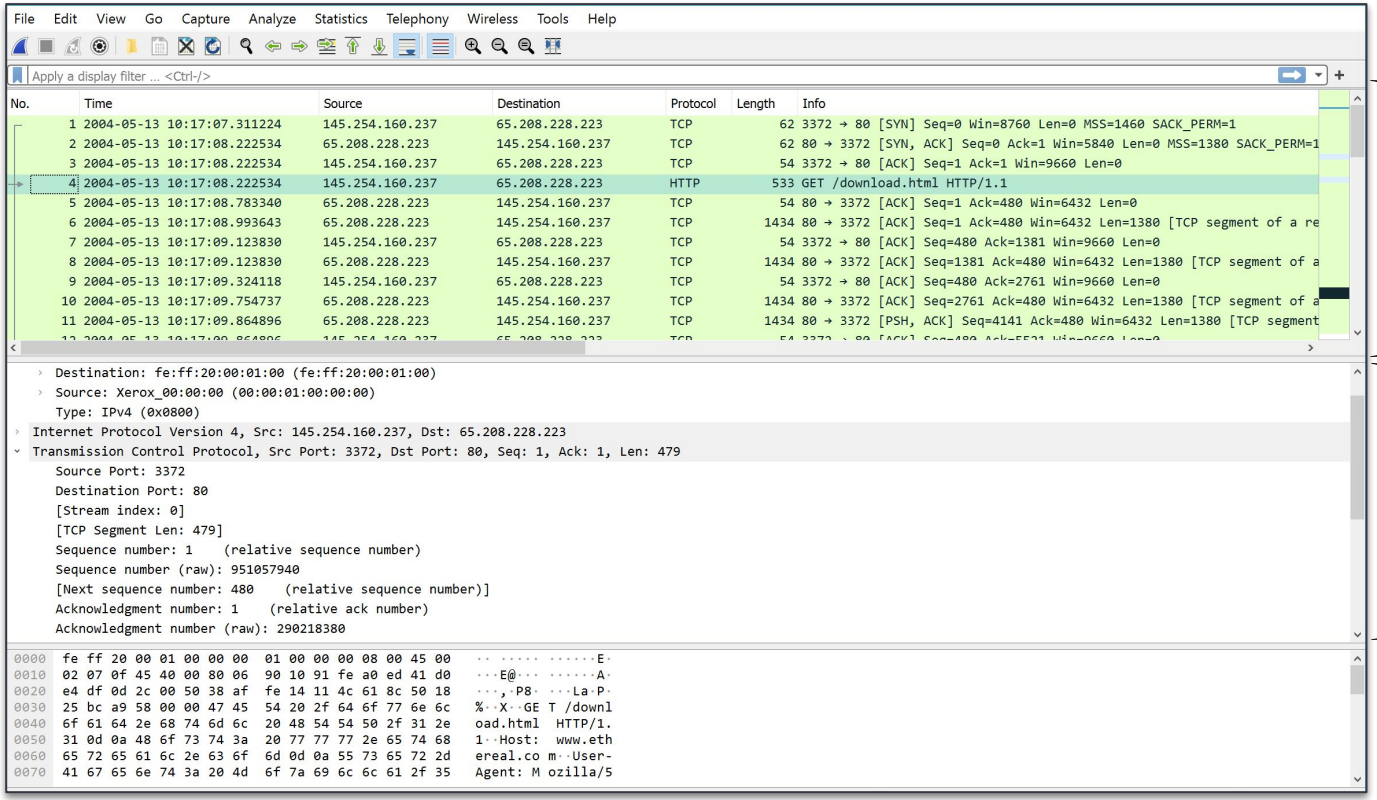
1 GB PCAP File

x4



4GB RAM Utilization

Wireshark Basic Interface



Packet List

Packet Details

Packet Bytes

elastic

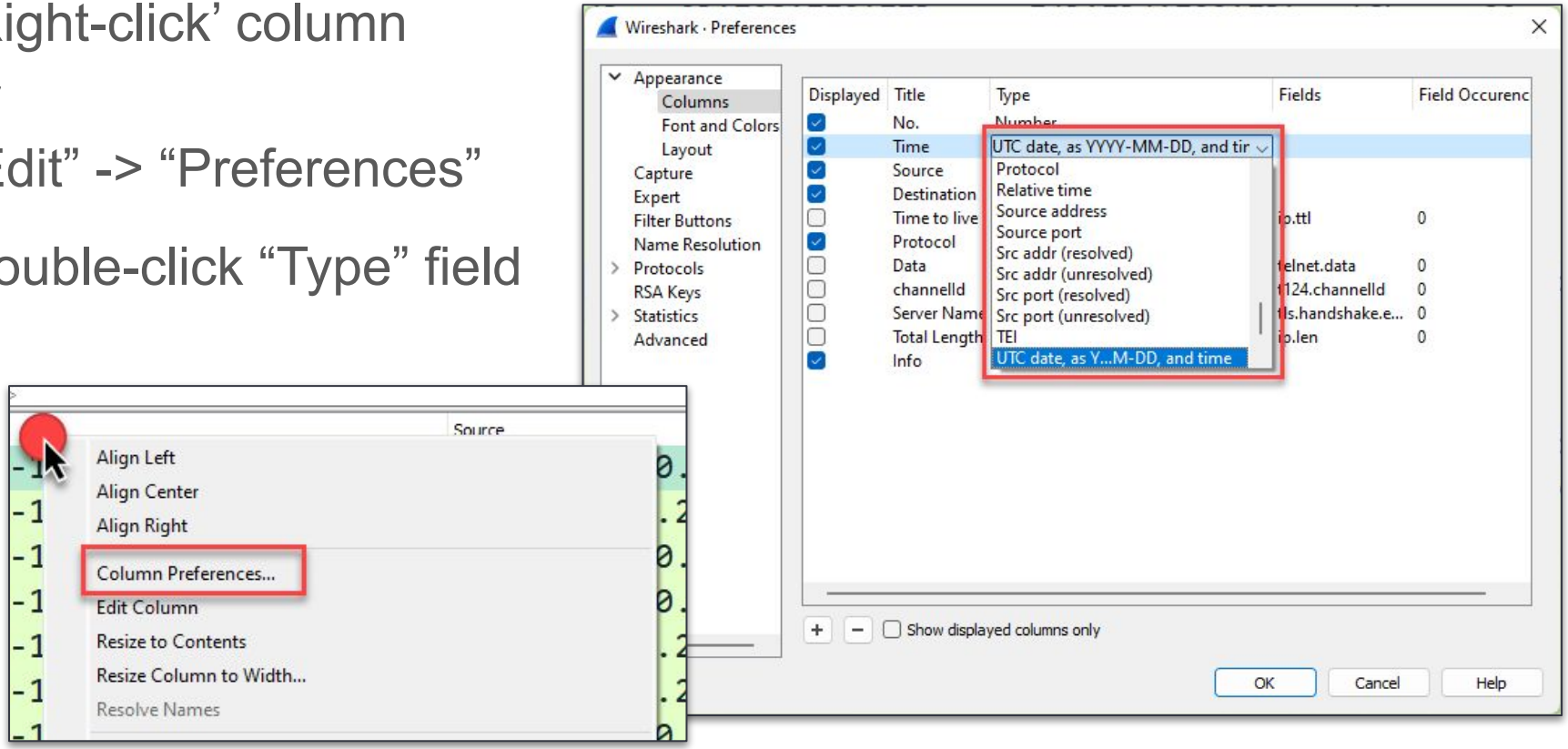
Packet List Frame

- Displays an overview list of all the packets

No.	Time	Source	Destination	Protocol	Info
1	2004-05-13 10:17:07.311224	145.254.160.237	65.208.228.223	TCP	3372 → 80 [SYN] Seq=0 Win=8760 Len=0 MSS=1460 SACK_PERM=1
2	2004-05-13 10:17:08.222534	65.208.228.223	145.254.160.237	TCP	80 → 3372 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
3	2004-05-13 10:17:08.222534	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=1 Ack=1 Win=9660 Len=0
4	2004-05-13 10:17:08.222534	145.254.160.237	65.208.228.223	HTTP	GET /download.html HTTP/1.1
5	2004-05-13 10:17:08.783340	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=1 Ack=480 Win=6432 Len=0
6	2004-05-13 10:17:08.993643	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=1 Ack=480 Win=6432 Len=1380 [TCP segment of a re
7	2004-05-13 10:17:09.123830	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660 Len=0
8	2004-05-13 10:17:09.123830	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=1381 Ack=480 Win=6432 Len=1380 [TCP segment of a
9	2004-05-13 10:17:09.324118	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660 Len=0
10	2004-05-13 10:17:09.754737	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=2761 Ack=480 Win=6432 Len=1380 [TCP segment of a
11	2004-05-13 10:17:09.864896	65.208.228.223	145.254.160.237	TCP	80 → 3372 [PSH, ACK] Seq=4141 Ack=480 Win=6432 Len=1380 [TCP segment
12	2004-05-13 10:17:09.864896	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=5521 Win=9660 Len=0
13	2004-05-13 10:17:09.864896	145.254.160.237	145.253.2.203	DNS	Standard query 0x0023 A pagead2.googlesyndication.com
14	2004-05-13 10:17:09.945011	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=5521 Ack=480 Win=6432 Len=1380 [TCP segment of a
15	2004-05-13 10:17:10.125270	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=6901 Win=9660 Len=0
16	2004-05-13 10:17:10.205385	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=6901 Ack=480 Win=6432 Len=1380 [TCP segment of a
17	2004-05-13 10:17:10.225414	145.253.2.203	145.254.160.237	DNS	Standard query response 0x0023 A pagead2.googlesyndication.com CNAME
18	2004-05-13 10:17:10.295515	145.254.160.237	216.239.59.99	HTTP	GET /pagead/ads?client=ca-pub-2309191948673629&random=10844434302858
19	2004-05-13 10:17:10.325558	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	2004-05-13 10:17:10.686076	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=8281 Ack=480 Win=6432 Len=1380 [TCP segment of a
21	2004-05-13 10:17:10.806249	65.208.228.223	145.254.160.237	TCP	80 → 3372 [PSH, ACK] Seq=9661 Ack=480 Win=6432 Len=1380 [TCP segment
22	2004-05-13 10:17:10.806249	145.254.160.237	65.208.228.223	TCP	3372 → 80 [ACK] Seq=480 Ack=11041 Win=9660 Len=0
23	2004-05-13 10:17:10.946451	65.208.228.223	145.254.160.237	TCP	80 → 3372 [ACK] Seq=11041 Ack=480 Win=6432 Len=1380 [TCP segment of
24	2004-05-13 10:17:10.956465	216.239.59.99	145.254.160.237	TCP	80 → 3371 [ACK] Seq=1 Ack=722 Win=31460 Len=0

Time Preferences

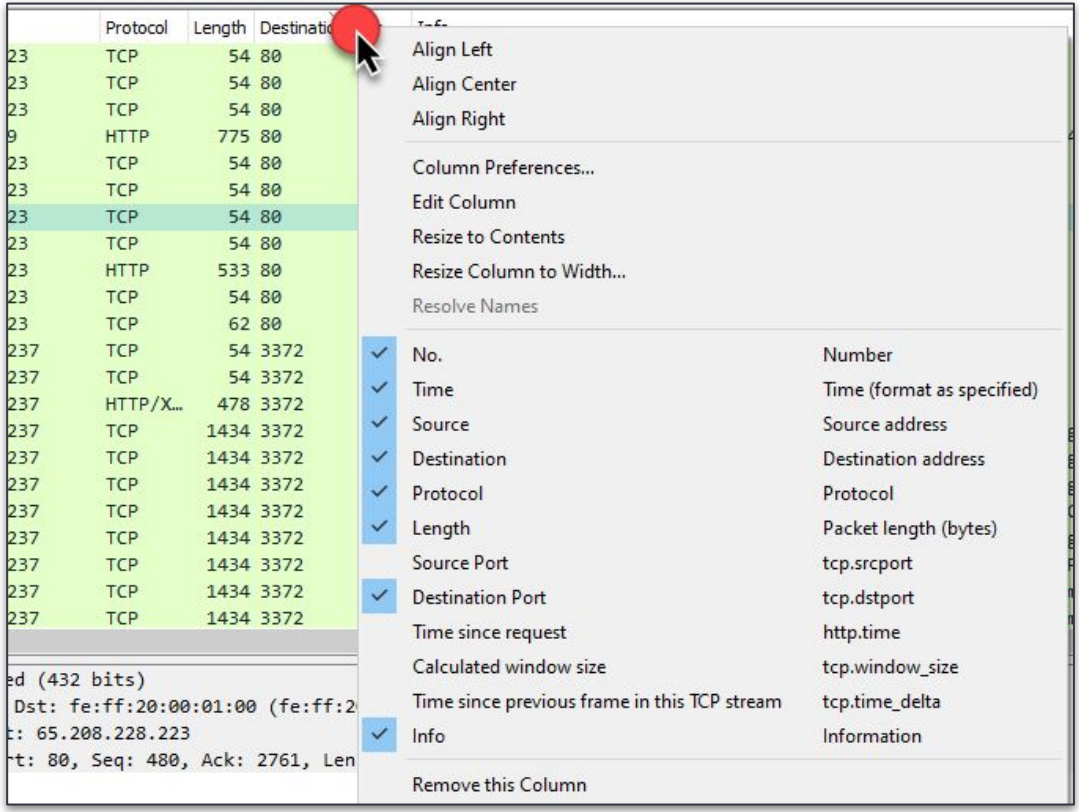
- ‘Right-click’ column
  or
  “Edit” -> “Preferences”
- Double-click “Type” field



elastic

Changing available columns

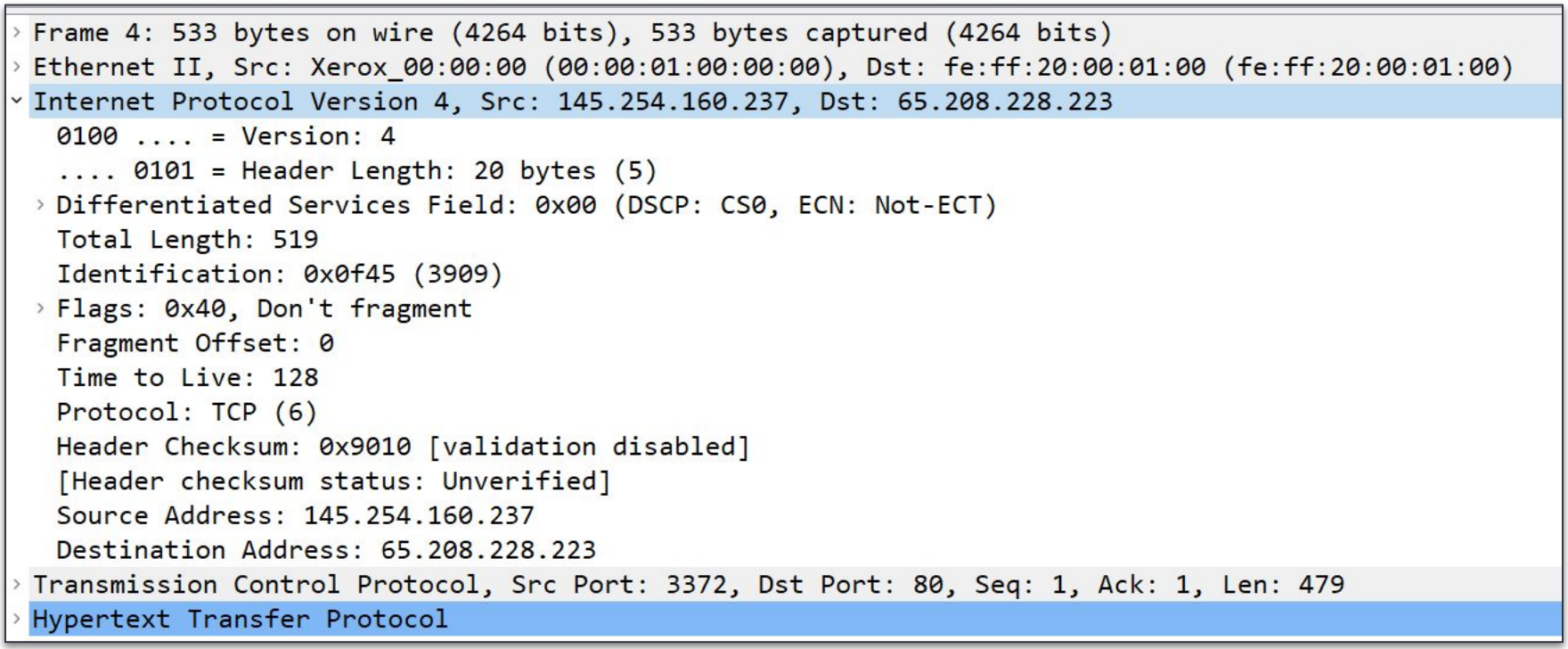
- Quickly add or remove columns from display



elastic

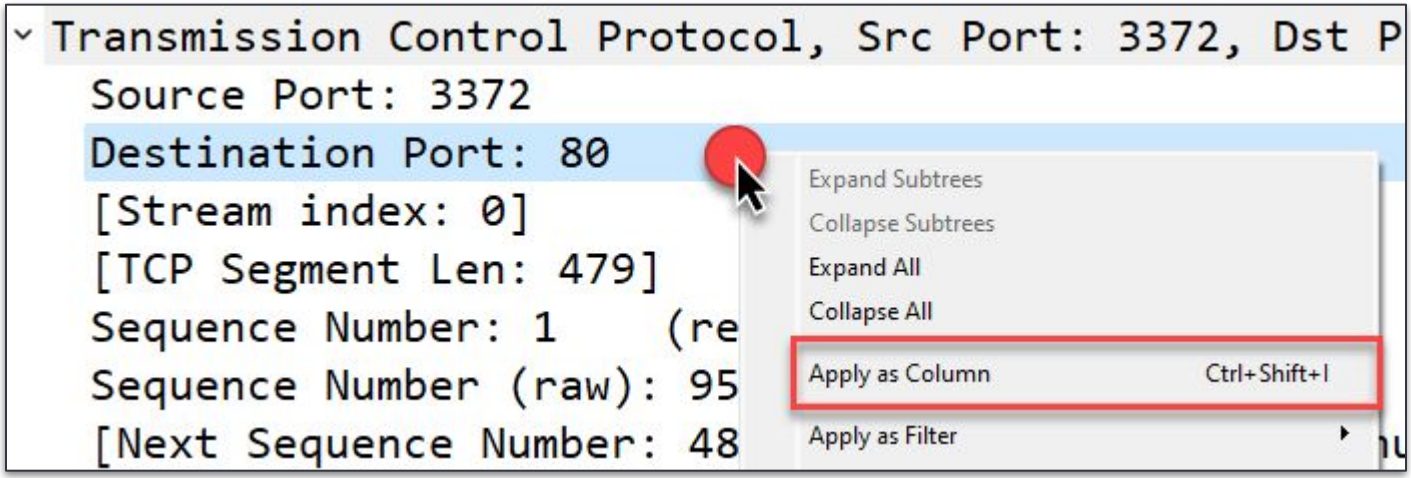
Packet Details Frame

- Displays full details grouped by layers



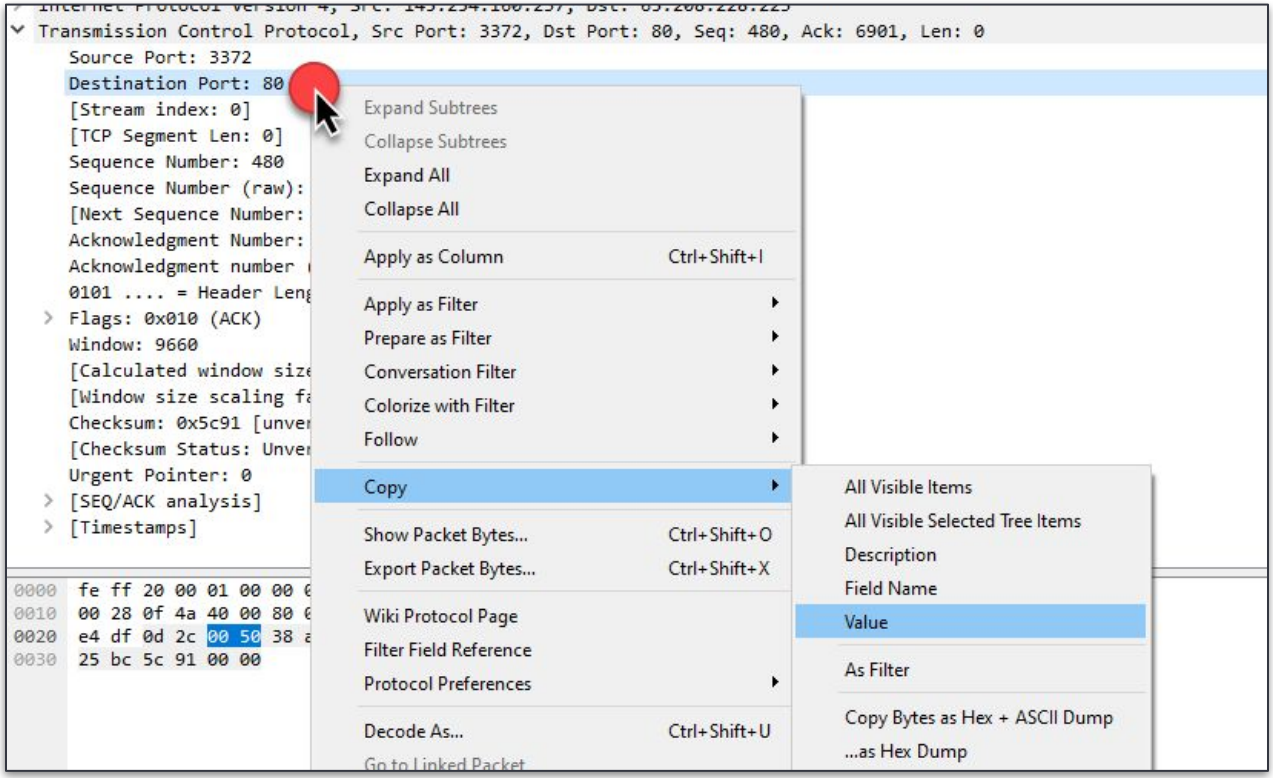
Apply as Column

- ‘Right-click’ any field
- Adds the column as a quick-view to the packet list

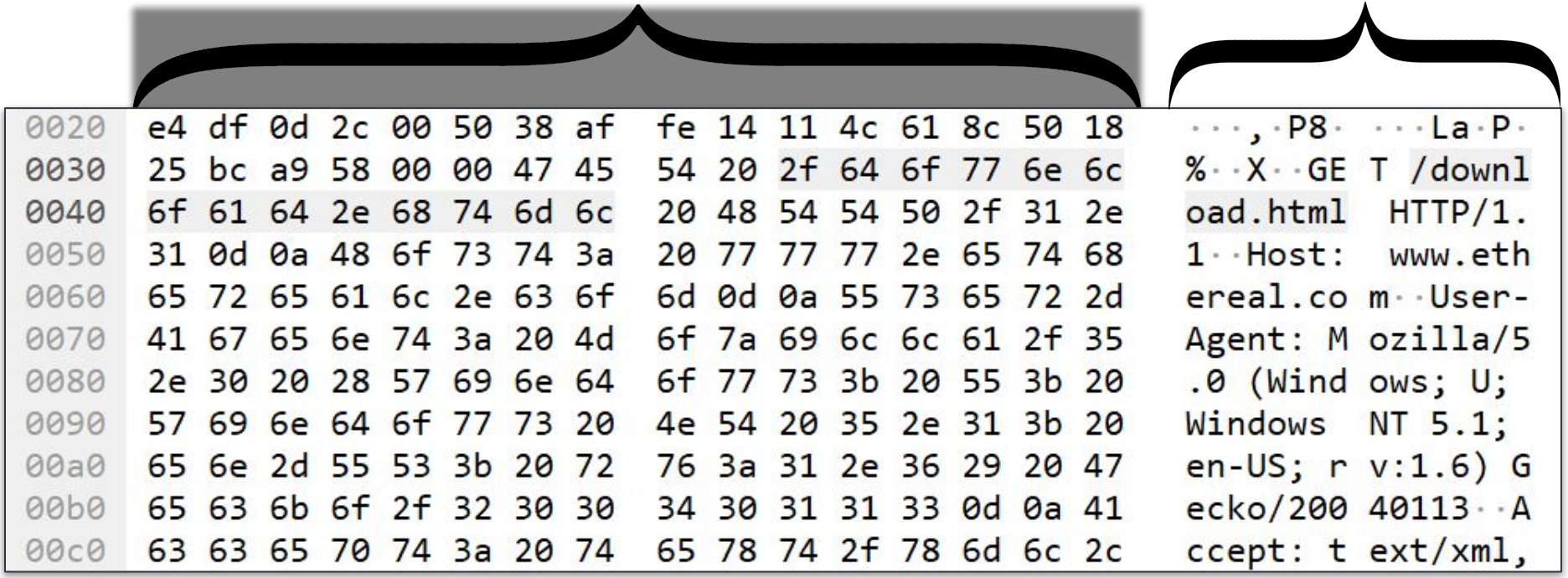


Copying Values

- Copies field values into clipboard for documenting



Packet Bytes Frame



elastic

CTF: Packet Encapsulation

Wireshark Syntax

Basic Fields (Too many to cover here!)

- ip.addr
- ip.src_host
- ip.dst_host
- tcp.dstport
- tcp.srcport
- http.request.method

Operators

==	eq
contains	
<	
>	
!	not
&&	and
||	or

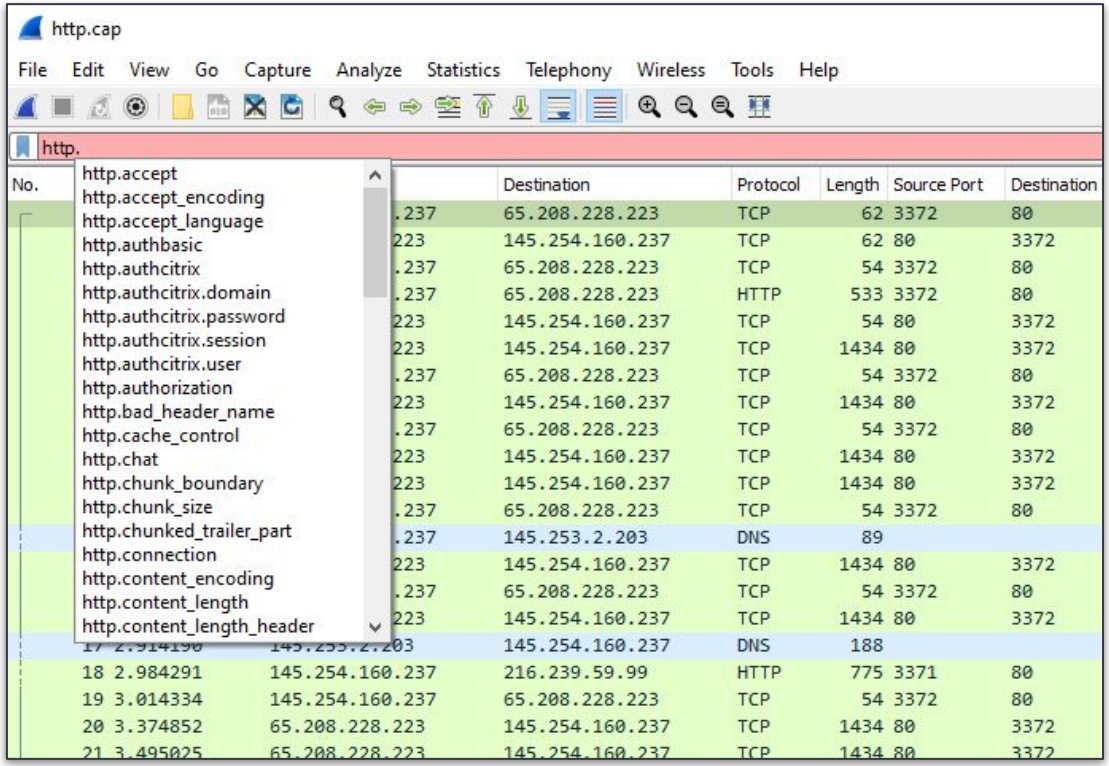
https://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

Wireshark examples

```
# HTTP packets (Wireshark does port-association for protocols)
http

# DNS packets using TCP for transport (DNS typically uses UDP)
dns and tcp

# Packets that have a source or destination IP of 1.2.3.4 and not TCP port 53
ip.addr == 1.2.3.4 and not tcp.port == 53

# Packets coming from external to internal IPs
not ip.src_host == 10.0.0.0/8 and ip.dst_host == 10.0.0.0/8
```

Auto-fill Syntax

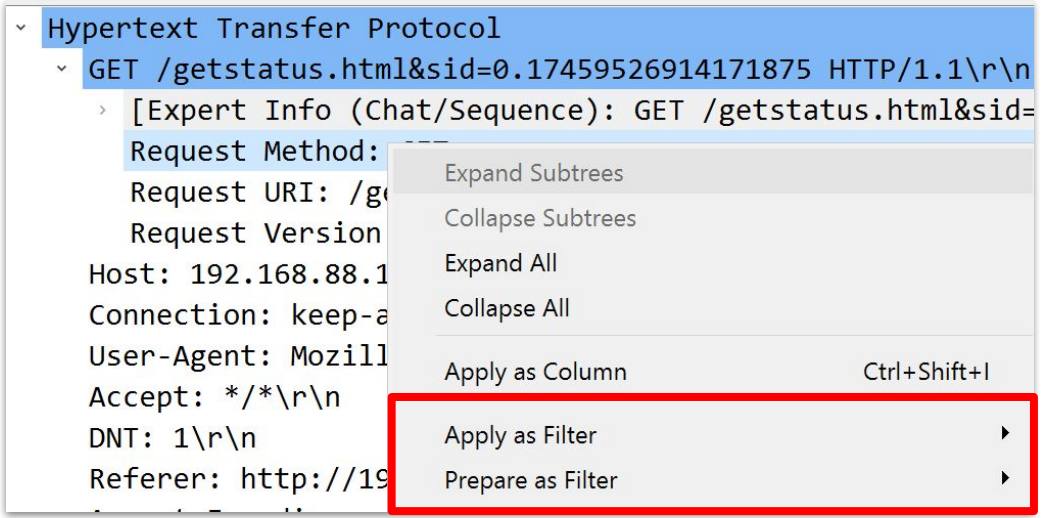
- Fields are “nested” and will auto-populate with available options



elastic

Wireshark: Right Click → Apply/Prepare As Filter

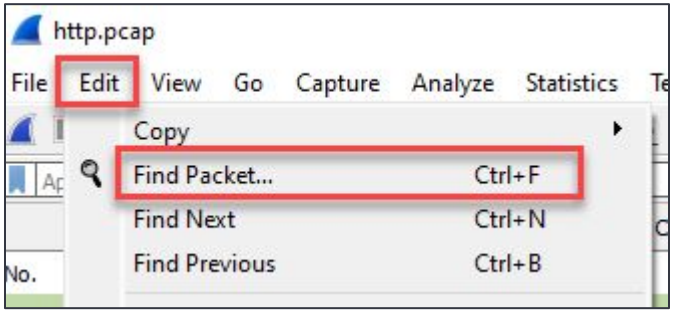
- **Apply as Filter** - Create and apply the a query in search bar
- **Prepare as Filter** - Create query in search bar but do not apply it

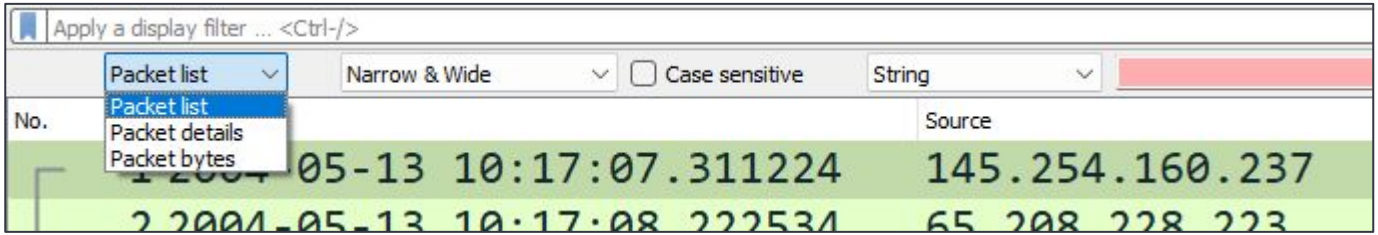


elastic

Find

- "Standard" search
  - Search all three panes
  - String or Regex options available

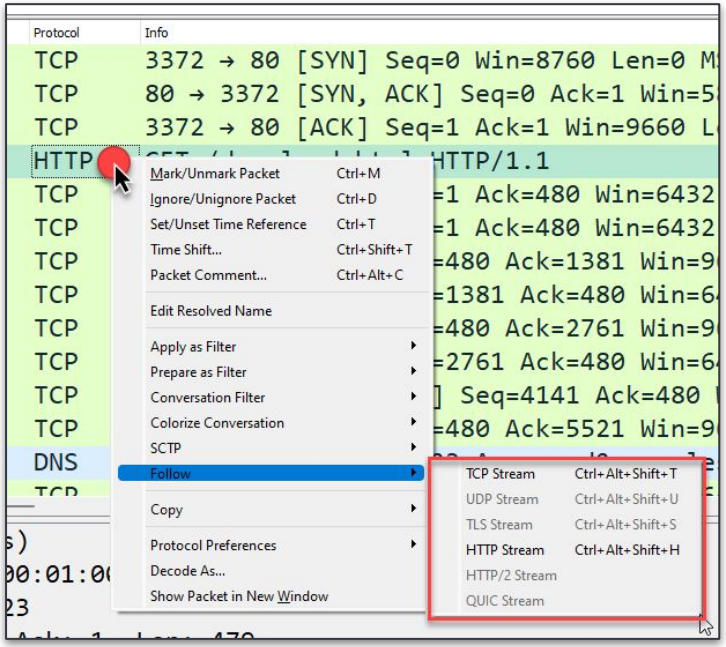




elastic

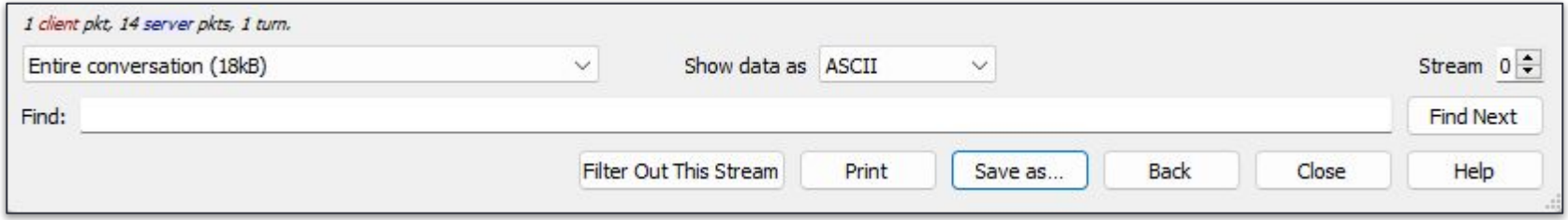
Wireshark: Right Click → Follow (TCP/UDP/HTTP)

- Follows a conversation across multiple packets
- Displays the application payload

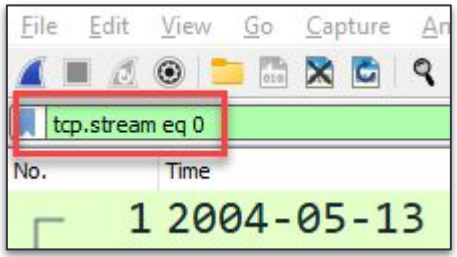


elastic

TCP Stream - Continued



- Extra Options!
  - Display format
  - Find
  - Moving through streams
- Automatically applies a filter



elastic

Misc

- Statistics Menu
  - High-level metadata about protocols and endpoints
- Live Capture
  - Live capture from any interface (ethernet, Wi-Fi, Bluetooth, or USB)
- File Extraction
  - For certain protocols (http, smb, tftp, etc.)
- Protocol Decryption
  - Can decrypt protocols if you load in the appropriate key file
- ‘tshark’
  - CLI companion utility (same filter syntax - much lower memory requirements)

CTF: More PCAP!

Packet Analysis

1. TCP/IP Model
2. Protocol Basics
3. Wireshark
4. **TCPdump/BPF**
5. Stenographer/Docket

TCPdump

- Created in 1988 by researchers at Lawrence Berkeley Labs
- Literally dumps TCP packets
- Can **capture** from interface or **read** from file
- Can write **summary data** to screen or full PCAP to file

TCPdump Options Flags

Option flags are mainly used to change the output of TCPdump

```
#Basic TCPdump Syntax
tcpdump <OPTIONS> -r <FILE>
```

-r	read file
-c X	limit the results to the first X number of packets that match the filter
-v(vv)	verbose, adds more information to the output of tcpdump (has three levels of verbosity v, vv, and vvv)
-n	no name resolution on IP addresses
-nn	no name resolution on IPs or port numbers

TCPdump - options examples

#Read the file traffic.pcap

```
tcpdump -r traffic.pcap
04:17:07.311224 IP dialin-145-254-160-237.pools.arcor-ip.net.3372 > 65.208.228.223.http: Flags [S], seq 951057939, win
8760, options [mss 1460,nop,nop,sackOK], length 0
```

#Read the file traffic.pcap but limit it to the first 10 packets. Do not resolve IPs or ports.

```
tcpdump -nn -c 10 -r traffic.pcap
04:17:07.311224 IP 145.254.160.237.3372 > 65.208.228.223.80: Flags [S], seq 951057939, win 8760, options [mss
1460,nop,nop,sackOK], length 0
```

#Read the file traffic.pcap and limit it to the first 10 packets at the highest level of verbosity.

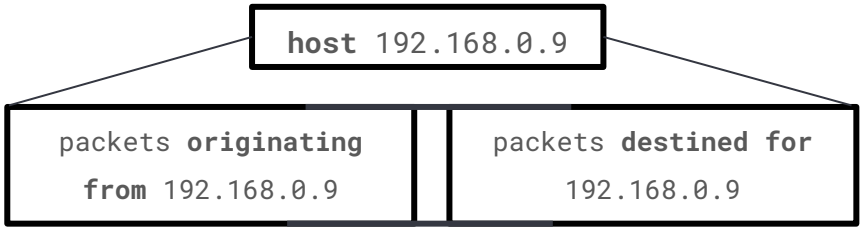
```
tcpdump -c 10 -vvv -r traffic.pcap
04:17:07.311224 IP (tos 0x0, ttl 128, id 3905, offset 0, flags [DF], proto TCP (6), length 48)
    dialin-145-254-160-237.pools.arcor-ip.net.3372 > 65.208.228.223.http: Flags [S], cksum 0xc30c (correct), seq
951057939, win 8760, options [mss 1460,nop,nop,sackOK], length 0
```

TCPdump filtering - BPF

- Simple query language to filter on network traffic
- Provides raw interface to data link layers
- Extremely efficient when run in kernel space - Fast!
- Can be used along TCPdump Option Flags

BPF Syntax

- Basic Tokens
  - Host IPv4 Address ‘**host** 192.168.0.9’
  - Net CIDR Address ‘**net** 192.168.0.0/16’
  - Port TCP/UDP Port ‘**port** 53’



- Protocol Tokens
  - icmp ‘**icmp**’
  - tcp ‘**tcp**’
  - udp ‘**udp**’

BPF Syntax

- Direction Operators
  - src — Originating host/net/port — ‘**src** host 10.0.0.32’
  - dst — Receiving host/net/port — ‘**dst** port 53

- Logical Tokens
  - or — || — ‘dst port 80 **or** dst port 443’
  - and — && — ‘host 10.0.0.3 **and** src port 53’
  - not — ! — **‘not** net 192.168.0.0/16’

TCPdump + BPF: Basic Query Function

#Basic TCPdump/BPF Syntax
tcpdump <OPTIONS> -r <FILE> '<BPF FILTER>'

#All TCP packets destined for 8.8.8.8 to port 53
tcpdump -r file.pcap 'tcp and dst host 8.8.8.8 and dst port 53'

#All non-ICMP packets heading from 10.0.0.7 to 10.0.0.3
tcpdump -r traffic.pcap 'src host 10.0.0.7 and dst host 10.0.0.3 and not icmp'

#All packets originating from outside the internal network and do not resolve IP/ports
tcpdump -nn -r traffic.pcap 'not src net 192.168.1.0/24'

#Query functions:
host 8.8.8.8
net 1.0.0.0/8
port 80
icmp
tcp
udp

#Direction operators:
src
dst

#Query operators:
or ||
and &&
not !

CTF: TCPdump & Berkeley Packet Filters

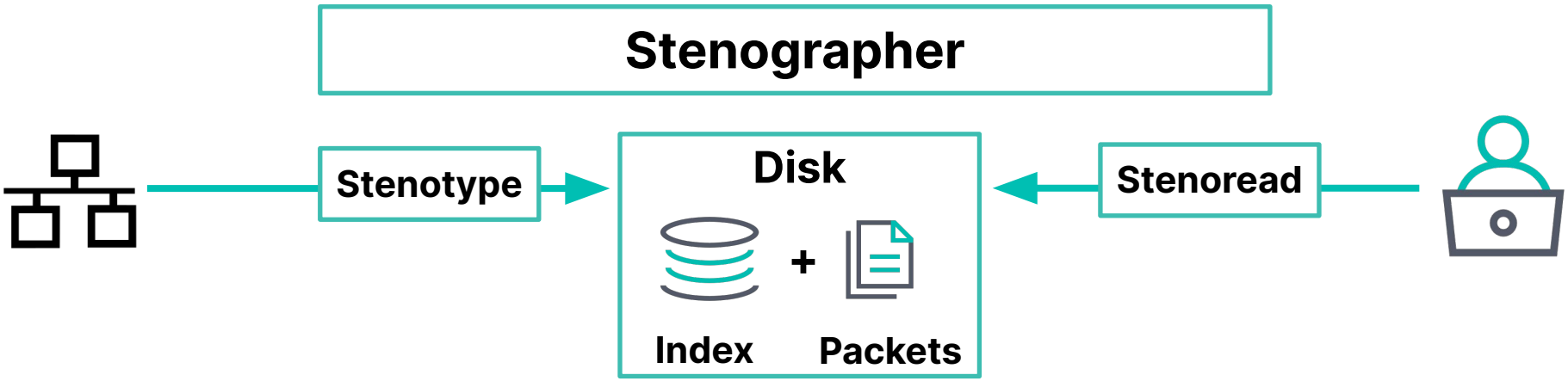
Packet Analysis

1. TCP/IP Model
2. Protocol Basics
3. Wireshark
4. TCPdump/BPF
5. **Stenographer/Docket**

Google Stenographer

- 80/20 project by Graeme Connell
- High speed packet capture and index
- Highly Reliable
- Automatically Manages Disk space
- Written in Go

Process Architecture - Simplified



Stenoread

Stenoread is the command used to query Stenographer for packets via the Application Programming Interface (API)

- Uses subset of BPF, **CANNOT specify direction (src/dst)**
- After retrieving packets, stenoread passes them over to tcpdump to display to user
- Supports tcpdump options

Stenoread - Basic Query Function

```
#Basic Syntax
stenoread `<STENO OPTIONS>'
```

```
#Return all packets involving host 8.8.8.8
stenoread ‘host 8.8.8.8’


#Return all TCP packets involving hosts in 10.0.0.0/8
stenoread ‘tcp and net 10.0.0.0/8’


#Return all UDP packets involving host 1.2.3.4 and
5.6.7.8
stenoread ‘udp and host 1.2.3.4 and host 5.6.7.8’
```

```
#Basic query functions:
host 8.8.8.8
net 1.0.0.0/8
port 80
icmp
tcp
udp
```

```
#Basic query operators:
or   ||
and  &&
```

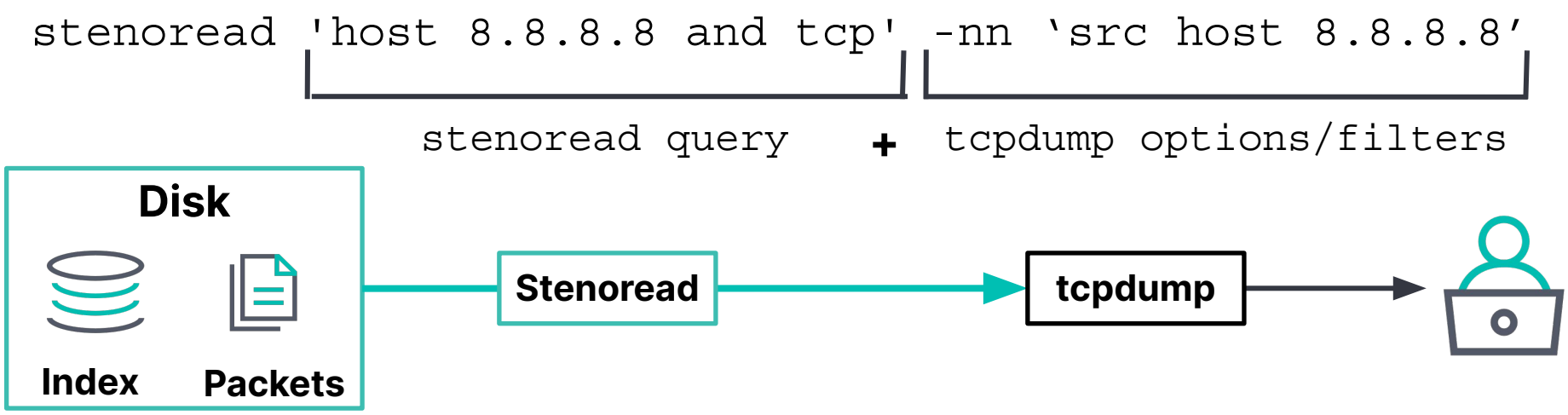
Stenoread - Time Function

```
#Stenographer-specific time additions:
before 2012-11-03T11:05:00Z       # Packets before a specific time (UTC)
after 2012-11-03T11:05:00-07:00   # Packets after a specific time (with TZ)
before 45m ago                    # Packets before a relative time
after 3h ago                      # Packets after a relative time
```

```
#Return all packets involving host 8.8.8.8 from the last 2 hours
stenoread 'host 8.8.8.8 and after 2h ago'

#Return all packets from a specific 6 hours
stenoread 'before 2020-11-03T11:05:00Z and after 2020-11-03T05:05:00Z'

#Return all UDP packets involving host 1.2.3.4 and 5.6.7.8
stenoread 'udp and host 1.2.3.4 and host 5.6.7.8'
```

Stenographer + TCPdump Options

- The results from stenoread are given to tcpdump to display to the user
- Able to use filters and display flags
- Allows to specify direction



Stenographer + TCPdump Examples

 # All packets originating from 1.1.1.1 without DNS resolution (-n).

stenoread 'host 1.1.1.1' -n 'src host 1.1.1.1'

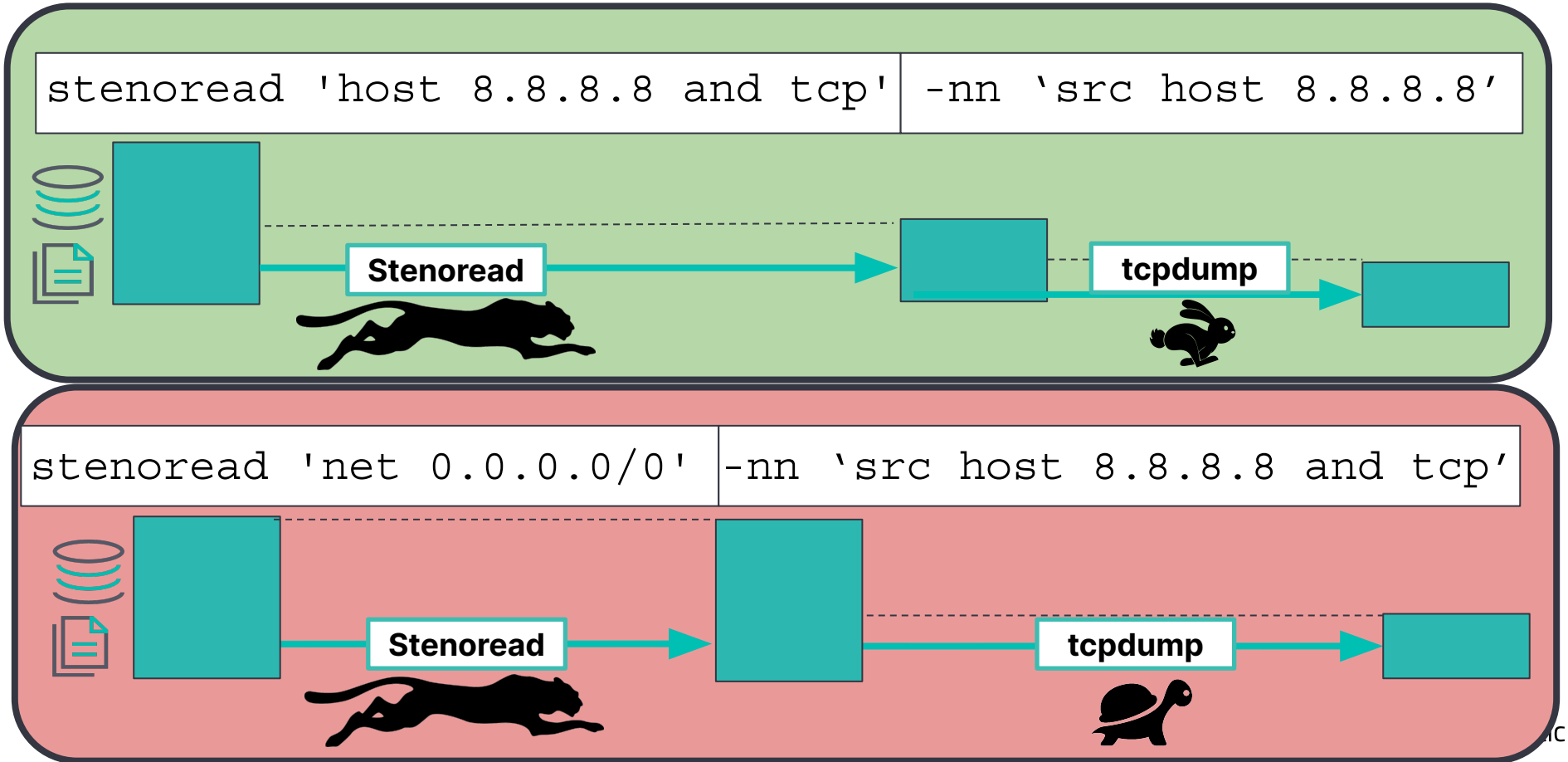
 # UDP packets from 10.0.0.7 to 2.2.2.2 without DNS or port resolution (-nn):

stenoread 'udp and host 10.0.0.7 and host 2.2.2.2' -nn 'src host 10.0.0.7 and dst host 2.2.2.2'

 # Write all packets involving 1.1.1.1 from the last three hours to a pcap.

stenoread 'host 1.1.1.1 and after 3h ago' -w /tmp/out.pcap

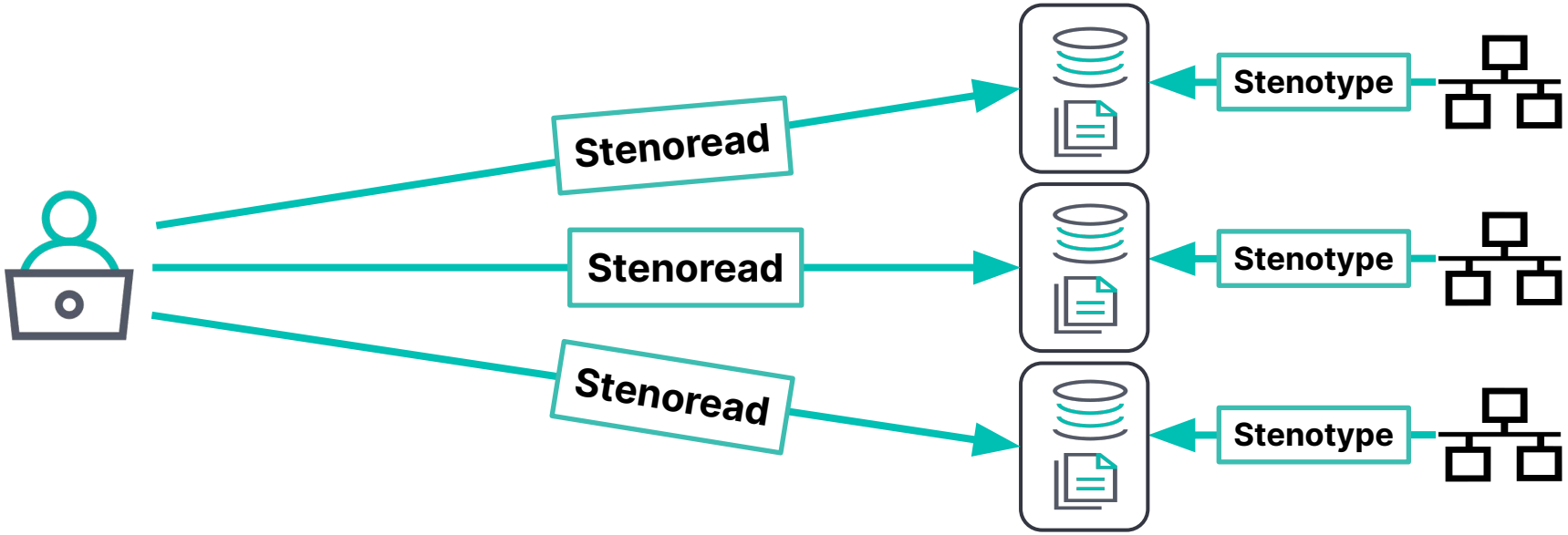
Stenographer + TCPdump Efficiency



CTF: Stenographer

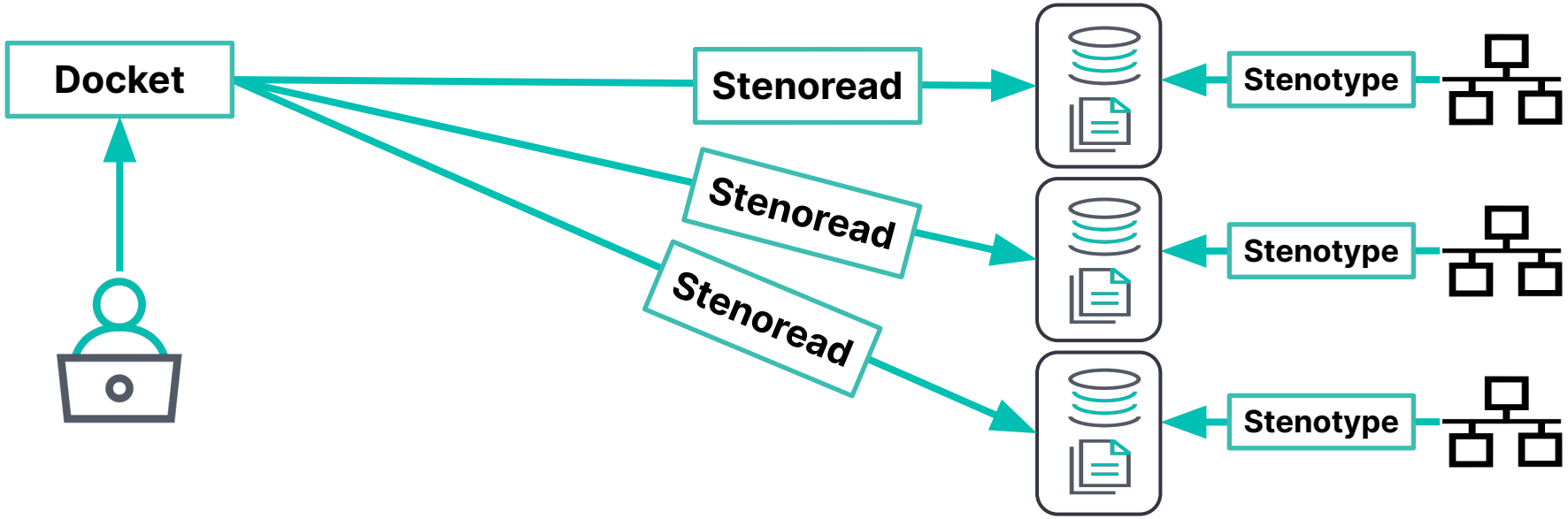
Stenoread Across Multiple Sensors

- Time Intensive - Logging on to each individual machine
- Not scalable - Returning multiple PCAPS over possibly 100's of machines



Docket

- Sends stenoread API calls to multiple machines
- Returns single PCAP to user
- Reduces Analyst Overhead



elastic

Docket -> Stenoread

```
curl http://docket/api/uri/host/1.2.3.4/udp/port/53/

stenoread 'host 1.2.3.4 and udp and port 53'
```

Docket Web Front

Stenoread Time Options
- Tab for absolute and relative time

Stenoread Host Options
- Still cannot specify src/dst
- Tab for CIDR range

Stenoread Port/Protocol Options
- Still cannot specify src/dst

elastic

CTF: Docket

Quiz - Part 1

1. Stenographer is written in what language?

2. Which command line tool allows you to query Stenographer's indexed packets?

3. True or False: When querying Stenographer for packets, you can use tcpdump filters.

Quiz - Part 2

4. Which of the following is an additional query option that is available in Stenographer but not available in tcpdump?

a before 2012-11-03T11:05:00Z
b host 1.1.1.1
c tcp[tcpflags] & (tcp-syn|tcp-fin)
d net 2.2.2.2/24

5. Which of the following is NOT true about PCAP?

a allows the analyst to reconstruct what actually happened
b PCAP scales well... the more the better!
c contains all the raw data

Quiz - Part 3

6. True or False: Stenographers indexed packets on disk are not in tcpdump format because they have an additional 4 byte header.

7. Analyzing PCAP is a memory intensive task. What is the general memory usage multiplier?

8. What does BPF stand for?

9. What is Docket?

10. True or False: BPF is more efficient than text filtering tools, like grep?