EECS 388: Lab 6

- Project 3 Introduction
- Python Socket Tutorial
- Wireshark Primer

Current Assignments

- Lab Assignment 3 **Thursday, Oct. 12 at 6 p.m.**
- Project 3: Networking **due Thursday, Oct. 26 at 6 p.m.**
    - Coverage:
        - Network traces
        - Password cracking
        - Identity management
        - DNS resolver

Reminder: Midterm is Friday, Oct 20 7-8:30 p.m.

Web Project Recap

SQL Injection

1.0 - No defense

- Basic exploitation of data vs. code

1.1 - Simple escaping

- New ways to escape characters?

1.2 - Hashing

- The password isn’t sanitized, but it’s hashed
- How can we control the hash?

XSS

2.0 - No defense

- Basic exploitation of data vs. code

2.1 - Remove “script”

- Use other tags or trick Regex?

2.2 - Remove several tag

- Use other tags or trick Regex?

2.3 - Remove some punctuation

- Combine techniques (sanitization, tricking regex?)

CSRF

3.0 - No defense

- HTML form with *username* and *password* inputs
- *ajax* POST requests to the correct URL

3.1 - Token validation

- Combine XSS and CSRF!
- Bypass SOP by using an *<iframe>*

Networking Project Intro

Introduce the story

You are hired by the U.S. Department of Cyber Espionage (USDCE) 😈 , and your first job is to conduct an investigation of a cyber attack.

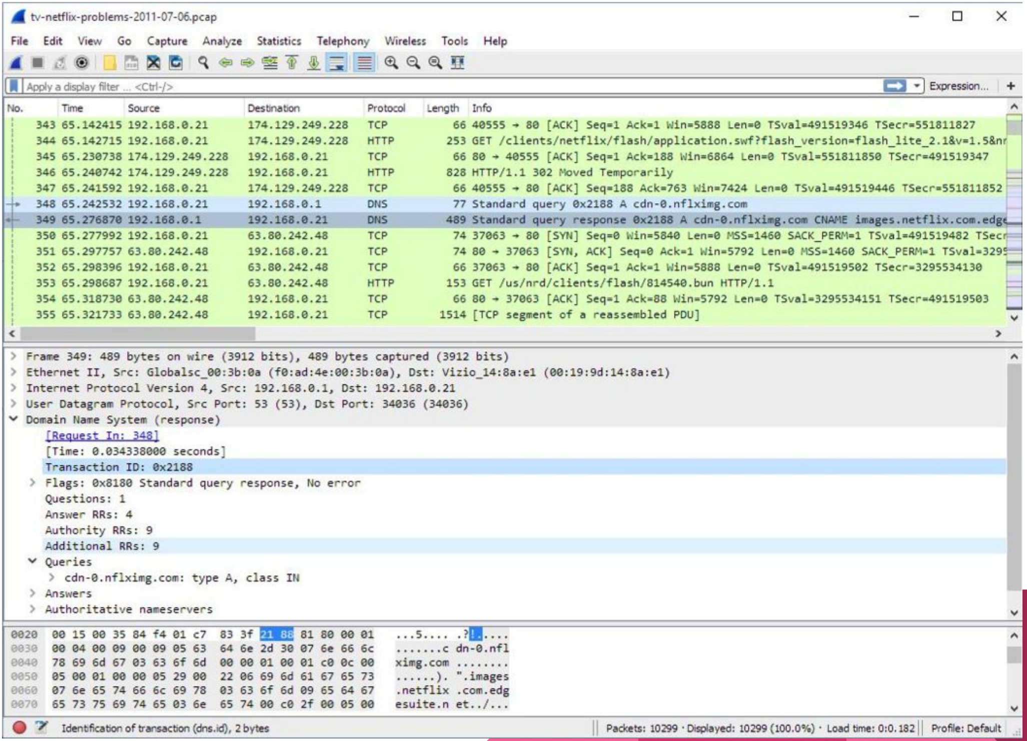
- There are five checkpoints
- Some tools you may find helpful along the way:
    - Wireshark
    - Python `ssl`
    - Python sockets
    - John the Ripper
    - …

Wireshark

Apply a filter based on protocol (HTTP, TLS, etc.)

Packet list

Protocol breakdown



tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

No.	Time	Source	Destination	Protocol	Length	Info
343	65.142415	192.168.0.21	174.129.249.228	TCP	66	40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344	65.142715	192.168.0.21	174.129.249.228	HTTP	253	GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345	65.230738	174.129.249.228	192.168.0.21	TCP	66	80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346	65.240742	174.129.249.228	192.168.0.21	HTTP	828	HTTP/1.1 302 Moved Temporarily
347	65.241592	192.168.0.21	174.129.249.228	TCP	66	40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348	65.242532	192.168.0.21	192.168.0.1	DNS	77	Standard query 0x2188 A cdn-0.nflximg.com
349	65.276870	192.168.0.1	192.168.0.21	DNS	489	Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350	65.277992	192.168.0.21	63.80.242.48	TCP	74	37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351	65.297757	63.80.242.48	192.168.0.21	TCP	74	80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352	65.298396	192.168.0.21	63.80.242.48	TCP	66	37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353	65.298687	192.168.0.21	63.80.242.48	HTTP	153	GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354	65.318730	63.80.242.48	192.168.0.21	TCP	66	80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355	65.321733	63.80.242.48	192.168.0.21	TCP	1514	[TCP segment of a reassembled PDU]

```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

Identification of transaction (dns.id), 2 bytes

Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182

Profile: Default

John the Ripper

- Password Cracker!
- Helpful in trying to brute-force decrypt password-protected files or crack password hashes
- Use a *wordlist* to “guess”
  - Consists of passwords discovered in breaches of other systems
  - Common wordlists can be found online
  - John the Ripper has a good built-in wordlist
    - Or you can specify a custom wordlist using the `--wordlist` argument
      - `john --wordlist=<wordlist> <target hash>`

Crack your own file

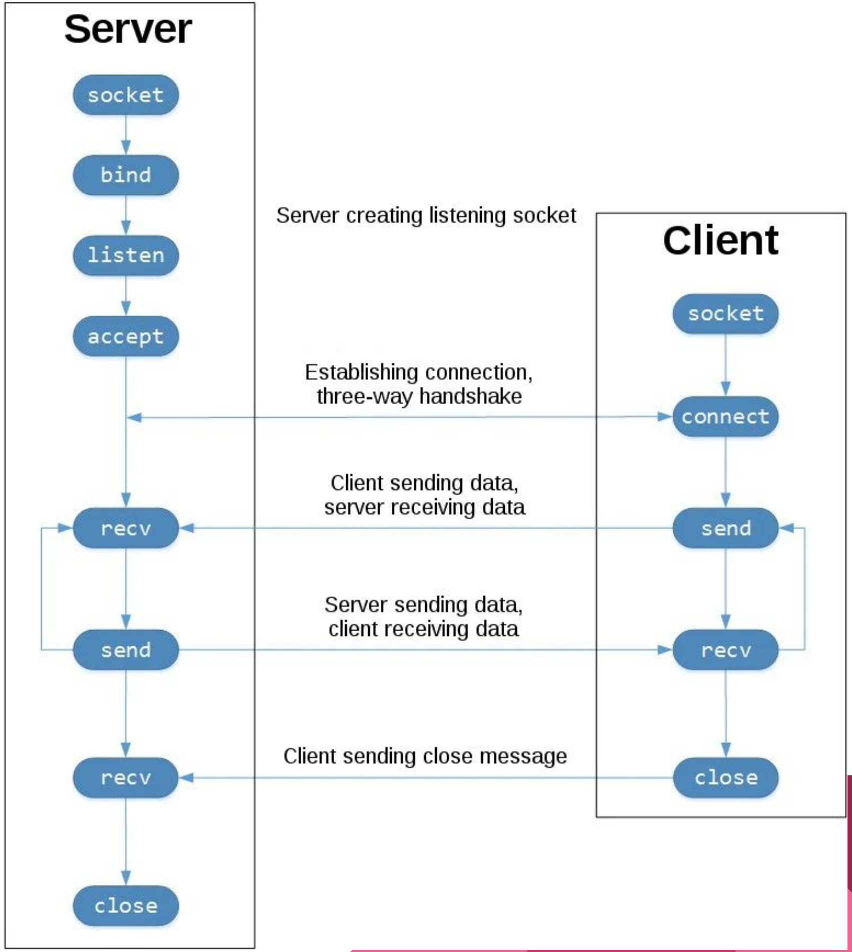
- Add a weak password to your PDF file
  - Implementation varies depending on platform
    - Mac: https://tinyurl.com/4zk654kp
    - Windows: https://tinyurl.com/2p8vnc9v
    - Linux: https://tinyurl.com/y9sz5whv
- Generate PDF hash file
  - `pdf2john.pl pdf_protected.pdf > pdf.hash`
- Crack it!
  - `john pdf.hash`

Python Networking

- Some common Python modules:
    - `ssl`, `socket`
- **Socket**: endpoints of the communication channel between client and server (typically provide bare TCP and UDP)
- Use **ssl** to wrap sockets to provide TLS connections

Python Socket

- Create a socket object using `socket.socket()`
- Default protocol used is TCP
- TCP Socket Flow


Server
socket
bind
listen
accept
recv
send
recv
close
Server creating listening socket
Establishing connection, three-way handshake
Client sending data, server receiving data
Server sending data, client receiving data
Client sending close message
Client
socket
connect
send
recv
close

Python Socket Example

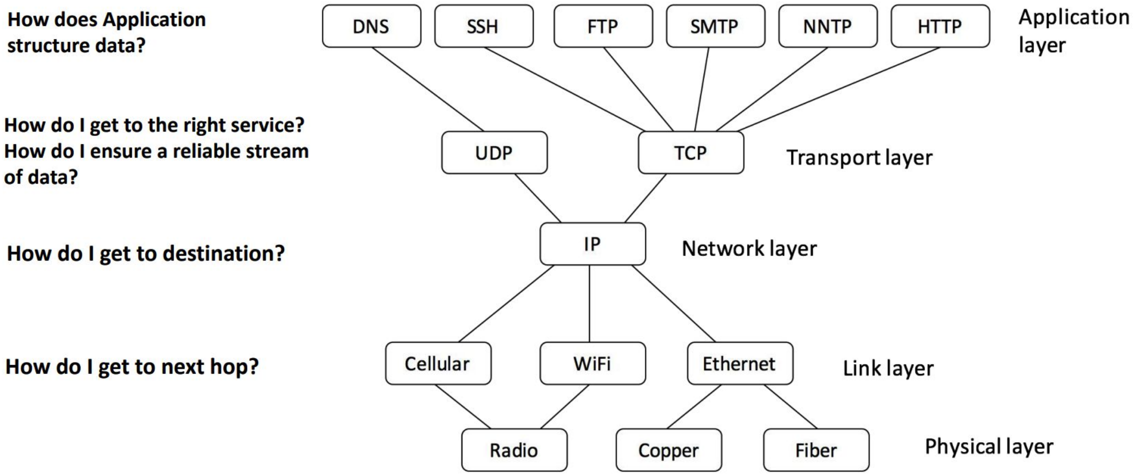
```
import socket

HOST = "127.0.0.1"  # server’s hostname or IP address
PORT = 65432  # port used by server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:  # create a socket object
     sock.connect((HOST, PORT))  # connect to the server
     sock.sendall(b"Hello, world!")  # send message
     data = s.recv(1024)  # read server’s reply - maximum data received at once is 1024 bytes

print(data)  # print server’s reply
```

- `AF_INET` is the Internet address family for IPv4
- `SOCK_STREAM` is the socket type for TCP
- `s.connect` expects a pair `(host, port)`
- `s.recv` returns a bytes object

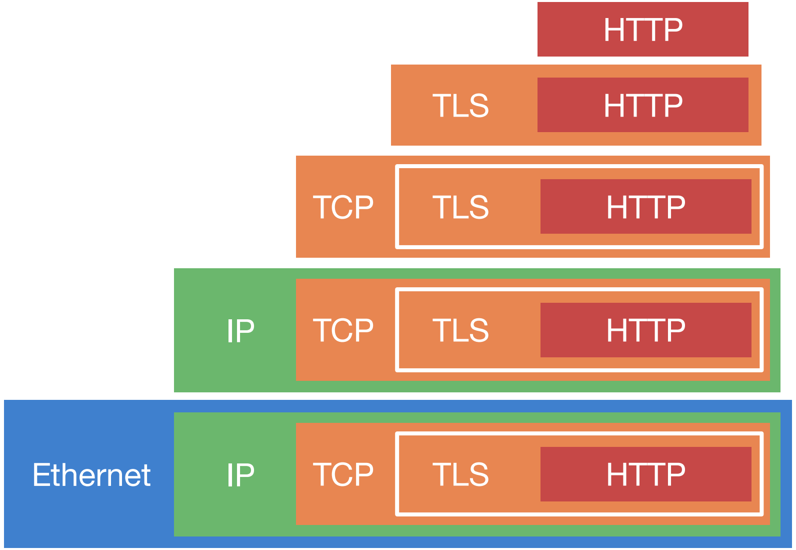
Wireshark Walkthrough

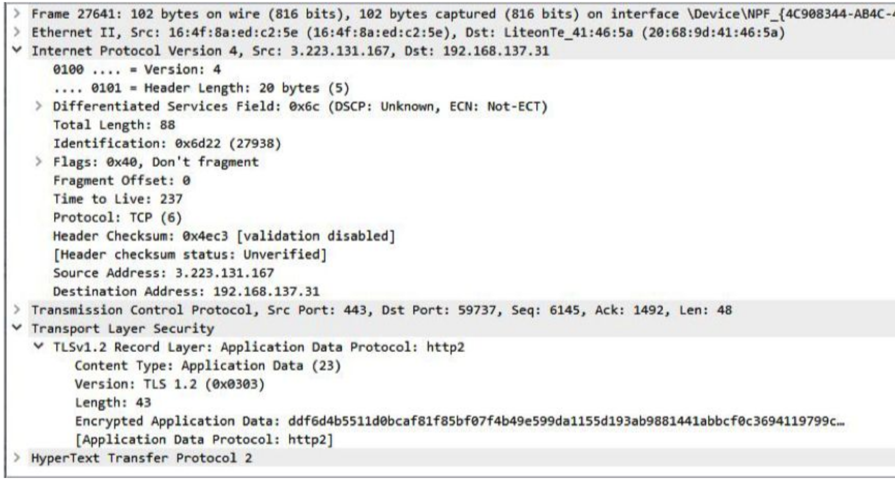
Review: Computer Networking in a Nutshell

- How do we send data over a network?
- Layers separate protocols according to the task they have to do
  - Layers don't depend on each other (in theory)


How does Application structure data?
DNS
SSH
FTP
SMTP
NNTP
HTTP
Application layer
How do I get to the right service? How do I ensure a reliable stream of data?
UDP
TCP
Transport layer
How do I get to destination?
IP
Network layer
How do I get to next hop?
Cellular
WiFi
Ethernet
Link layer
Radio
Copper
Fiber
Physical layer

Review: Network Encapsulation





```
> Frame 27641: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface \Device\NPF_{4C908344-AB4C-
> Ethernet II, Src: 16:4f:8a:ed:c2:5e (16:4f:8a:ed:c2:5e), Dst: LiteonTe_41:46:5a (20:68:9d:41:46:5a)
v Internet Protocol Version 4, Src: 3.223.131.167, Dst: 192.168.137.31
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x6c (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 88
    Identification: 0x6d22 (27938)
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 237
    Protocol: TCP (6)
    Header Checksum: 0x4ec3 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 3.223.131.167
    Destination Address: 192.168.137.31
> Transmission Control Protocol, Src Port: 443, Dst Port: 59737, Seq: 6145, Ack: 1492, Len: 48
v Transport Layer Security
  v TLSv1.2 Record Layer: Application Data Protocol: http2
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 43
      Encrypted Application Data: ddf6d4b5511d0bcaf81f85bf07f4b49e599da1155d193ab9881441abbcf0c3694119799c…
      [Application Data Protocol: http2]
> HyperText Transfer Protocol 2
```

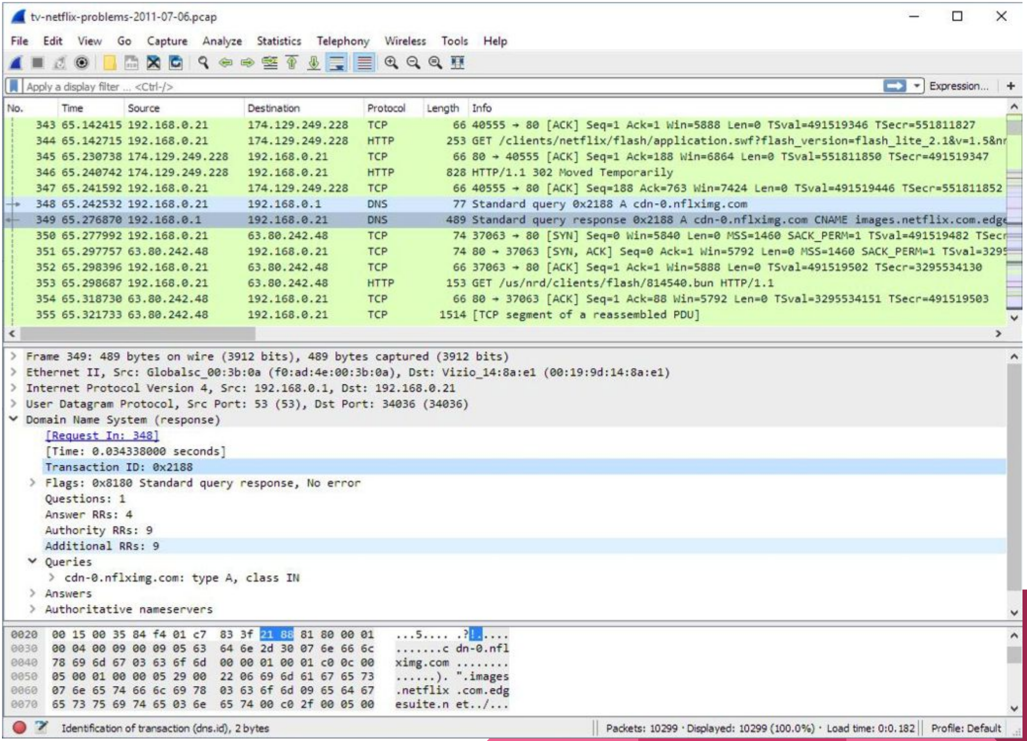
Wireshark

Apply a filter based on protocol (HTTP, TLS, etc.)

Packet list

Parsed packet layers

Raw packet bytes


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.   Time      Source          Destination     Protocol Length Info
  343 65.142415 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
  344 65.142715 192.168.0.21    174.129.249.228 HTTP      253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
  345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
  346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
  347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
  348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
  349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
  350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
  351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
  352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
  353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
  354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
  355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]
```

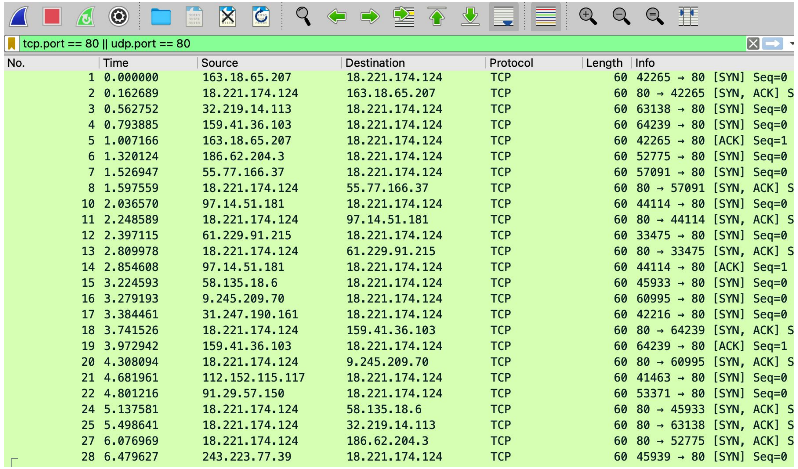
```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

Identification of transaction (dns.id), 2 bytes | Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182 | Profile: Default

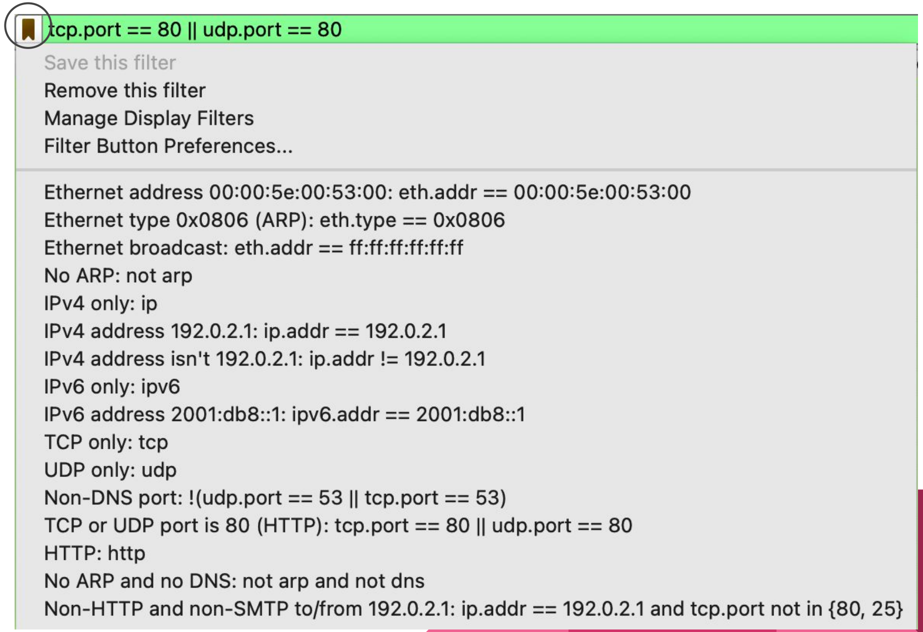
Wireshark Demo: Apply filter

Select all packets of TCP or UDP protocol with port 80:



tcp.port == 80 || udp.port == 80

```
No.     Time      Source           Destination      Protocol   Length Info
      1 0.000000  163.18.65.207    18.221.174.124   TCP            60 42265 → 80 [SYN] Seq=0
      2 0.162689  18.221.174.124   163.18.65.207    TCP            60 80 → 42265 [SYN, ACK] S
      3 0.562752  32.219.14.113    18.221.174.124   TCP            60 63138 → 80 [SYN] Seq=0
      4 0.793885  159.41.36.103    18.221.174.124   TCP            60 64239 → 80 [SYN] Seq=0
      5 1.007166  163.18.65.207    18.221.174.124   TCP            60 42265 → 80 [ACK] Seq=1
      6 1.320124  186.62.204.3     18.221.174.124   TCP            60 52775 → 80 [SYN] Seq=0
      7 1.526947  55.77.166.37     18.221.174.124   TCP            60 57091 → 80 [SYN] Seq=0
      8 1.597559  18.221.174.124   55.77.166.37     TCP            60 80 → 57091 [SYN, ACK] S
     10 2.036570  97.14.51.181     18.221.174.124   TCP            60 44114 → 80 [SYN] Seq=0
     11 2.248589  18.221.174.124   97.14.51.181     TCP            60 80 → 44114 [SYN, ACK] S
     12 2.397115  61.229.91.215    18.221.174.124   TCP            60 33475 → 80 [SYN] Seq=0
     13 2.809978  18.221.174.124   61.229.91.215    TCP            60 80 → 33475 [SYN, ACK] S
     14 2.854608  97.14.51.181     18.221.174.124   TCP            60 44114 → 80 [ACK] Seq=1
     15 3.224593  58.135.18.6      18.221.174.124   TCP            60 45933 → 80 [SYN] Seq=0
     16 3.279193  9.245.209.70     18.221.174.124   TCP            60 60995 → 80 [SYN] Seq=0
     17 3.384461  31.247.190.161   18.221.174.124   TCP            60 42216 → 80 [SYN] Seq=0
     18 3.741526  18.221.174.124   159.41.36.103    TCP            60 80 → 64239 [SYN, ACK] S
     19 3.972942  159.41.36.103    18.221.174.124   TCP            60 64239 → 80 [ACK] Seq=1
     20 4.308094  18.221.174.124   9.245.209.70     TCP            60 80 → 60995 [SYN, ACK] S
     21 4.681961  112.152.115.117  18.221.174.124   TCP            60 41463 → 80 [SYN] Seq=0
     22 4.801216  91.29.57.150     18.221.174.124   TCP            60 53371 → 80 [SYN] Seq=0
     24 5.137581  18.221.174.124   58.135.18.6      TCP            60 80 → 45933 [SYN, ACK] S
     25 5.498641  18.221.174.124   32.219.14.113    TCP            60 80 → 63138 [SYN, ACK] S
     27 6.076969  18.221.174.124   186.62.204.3     TCP            60 80 → 52775 [SYN, ACK] S
     28 6.479627  243.223.77.39    18.221.174.124   TCP            60 45939 → 80 [SYN] Seq=0
```

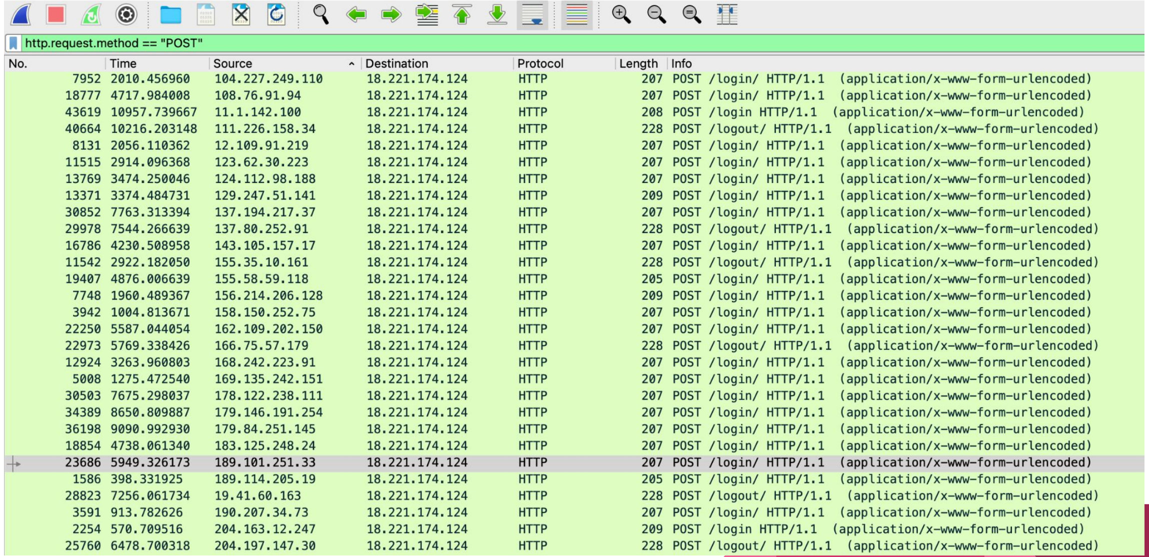


tcp.port == 80 || udp.port == 80

Ethernet address 00:00:5e:00:53:00: eth.addr == 00:00:5e:00:53:00
Ethernet type 0x0806 (ARP): eth.type == 0x0806
Ethernet broadcast: eth.addr == ff:ff:ff:ff:ff:ff
No ARP: not arp
IPv4 only: ip
IPv4 address 192.0.2.1: ip.addr == 192.0.2.1
IPv4 address isn't 192.0.2.1: ip.addr != 192.0.2.1
IPv6 only: ipv6
IPv6 address 2001:db8::1: ipv6.addr == 2001:db8::1
TCP only: tcp
UDP only: udp
Non-DNS port: !(udp.port == 53 || tcp.port == 53)
TCP or UDP port is 80 (HTTP): tcp.port == 80 || udp.port == 80
HTTP: http
No ARP and no DNS: not arp and not dns
Non-HTTP and non-SMTP to/from 192.0.2.1: ip.addr == 192.0.2.1 and tcp.port not in {80, 25}

Wireshark Demo: Apply filter

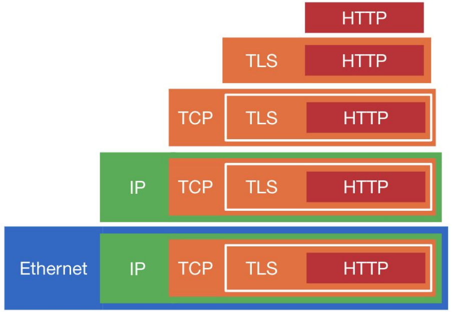
Select all packets of HTTP with POST method:

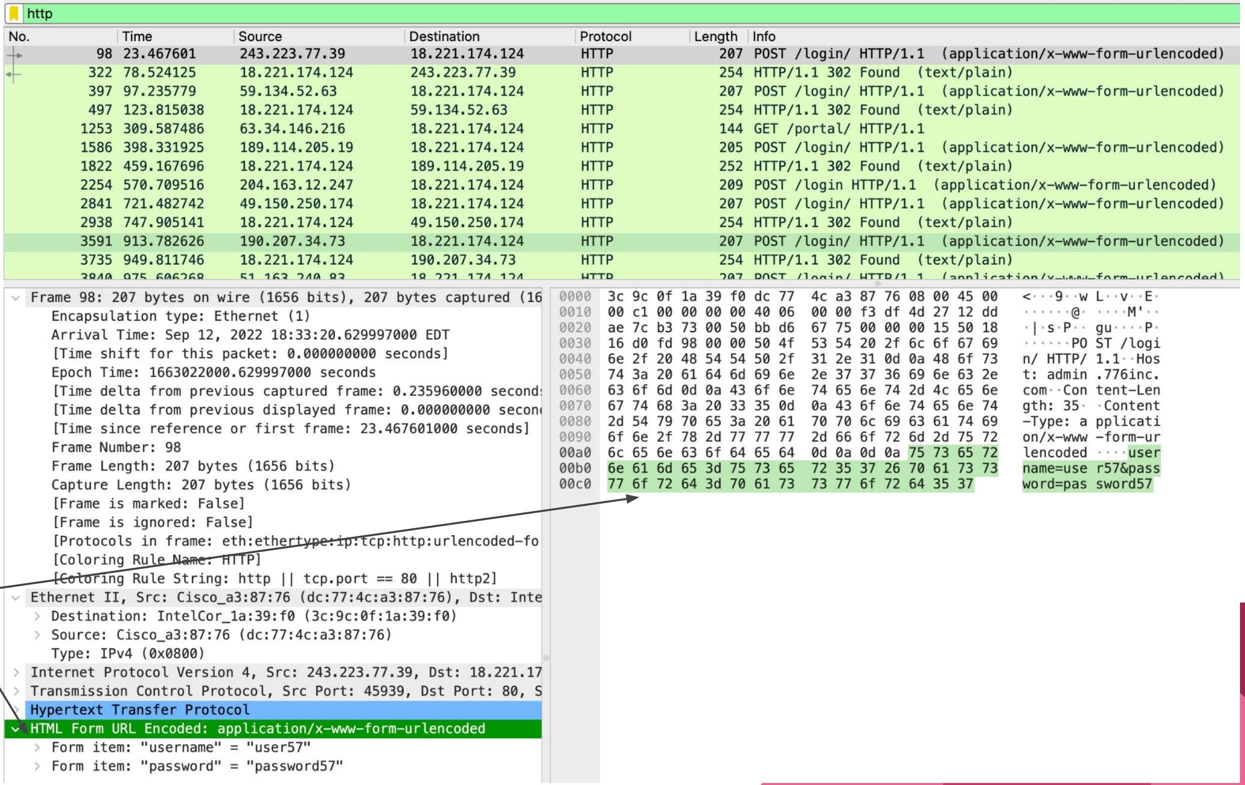


http.request.method == "POST"

No.	Time	Source	Destination	Protocol	Length	Info
7952	2010.456960	104.227.249.110	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
18777	4717.984008	108.76.91.94	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
43619	10957.739667	11.1.142.100	18.221.174.124	HTTP	208	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
40664	10216.203148	111.226.158.34	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)
8131	2056.110362	12.109.91.219	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
11515	2914.096368	123.62.30.223	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
13769	3474.250046	124.112.98.188	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
13371	3374.484731	129.247.51.141	18.221.174.124	HTTP	209	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
30852	7763.313394	137.194.217.37	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
29978	7544.266639	137.80.252.91	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)
16786	4230.508958	143.105.157.17	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
11542	2922.182050	155.35.10.161	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)
19407	4876.006639	155.58.59.118	18.221.174.124	HTTP	205	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
7748	1960.489367	156.214.206.128	18.221.174.124	HTTP	209	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
3942	1004.813671	158.150.252.75	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
22250	5587.044054	162.109.202.150	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
22973	5769.338426	166.75.57.179	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)
12924	3263.960803	168.242.223.91	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
5008	1275.472540	169.135.242.151	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
30503	7675.298037	178.122.238.111	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
34389	8650.809887	179.146.191.254	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
36198	9090.992930	179.84.251.145	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
18854	4738.061340	183.125.248.24	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
23686	5949.326173	189.101.251.33	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
1586	398.331925	189.114.205.19	18.221.174.124	HTTP	205	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
28823	7256.061734	19.41.60.163	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)
3591	913.782626	190.207.34.73	18.221.174.124	HTTP	207	POST /login/ HTTP/1.1 (application/x-www-form-urlencoded)
2254	570.709516	204.163.12.247	18.221.174.124	HTTP	209	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
25760	6478.700318	204.197.147.30	18.221.174.124	HTTP	228	POST /logout/ HTTP/1.1 (application/x-www-form-urlencoded)

Wireshark Demo: Dissect Packet





```
http
```

```
No.     Time          Source            Destination       Protocol   Length  Info
   98   23.467601     243.223.77.39     18.221.174.124    HTTP          207  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
  322   78.524125     18.221.174.124    243.223.77.39     HTTP          254  HTTP/1.1 302 Found  (text/plain)
  397   97.235779     59.134.52.63      18.221.174.124    HTTP          207  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
  497   123.815038    18.221.174.124    59.134.52.63      HTTP          254  HTTP/1.1 302 Found  (text/plain)
 1253   309.587486    63.34.146.216     18.221.174.124    HTTP          144  GET /portal/ HTTP/1.1
 1586   398.331925    189.114.205.19    18.221.174.124    HTTP          205  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
 1822   459.167696    18.221.174.124    189.114.205.19    HTTP          252  HTTP/1.1 302 Found  (text/plain)
 2254   570.709516    204.163.12.247    18.221.174.124    HTTP          209  POST /login HTTP/1.1  (application/x-www-form-urlencoded)
 2841   721.482742    49.150.250.174    18.221.174.124    HTTP          207  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
 2938   747.905141    18.221.174.124    49.150.250.174    HTTP          254  HTTP/1.1 302 Found  (text/plain)
 3591   913.782626    190.207.34.73     18.221.174.124    HTTP          207  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
 3735   949.811746    18.221.174.124    190.207.34.73     HTTP          254  HTTP/1.1 302 Found  (text/plain)
 3840   975.606268    51.163.240.83     18.221.174.124    HTTP          207  POST /login/ HTTP/1.1  (application/x-www-form-urlencoded)
```

```
v Frame 98: 207 bytes on wire (1656 bits), 207 bytes captured (16
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 12, 2022 18:33:20.629997000 EDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1663022000.629997000 seconds
    [Time delta from previous captured frame: 0.235960000 second
    [Time delta from previous displayed frame: 0.000000000 secon
    [Time since reference or first frame: 23.467601000 seconds]
    Frame Number: 98
    Frame Length: 207 bytes (1656 bits)
    Capture Length: 207 bytes (1656 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http:urlencoded-fo
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
v Ethernet II, Src: Cisco_a3:87:76 (dc:77:4c:a3:87:76), Dst: Inte
  > Destination: IntelCor_1a:39:f0 (3c:9c:0f:1a:39:f0)
  > Source: Cisco_a3:87:76 (dc:77:4c:a3:87:76)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 243.223.77.39, Dst: 18.221.17
> Transmission Control Protocol, Src Port: 45939, Dst Port: 80, S
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "user57"
  > Form item: "password" = "password57"
```

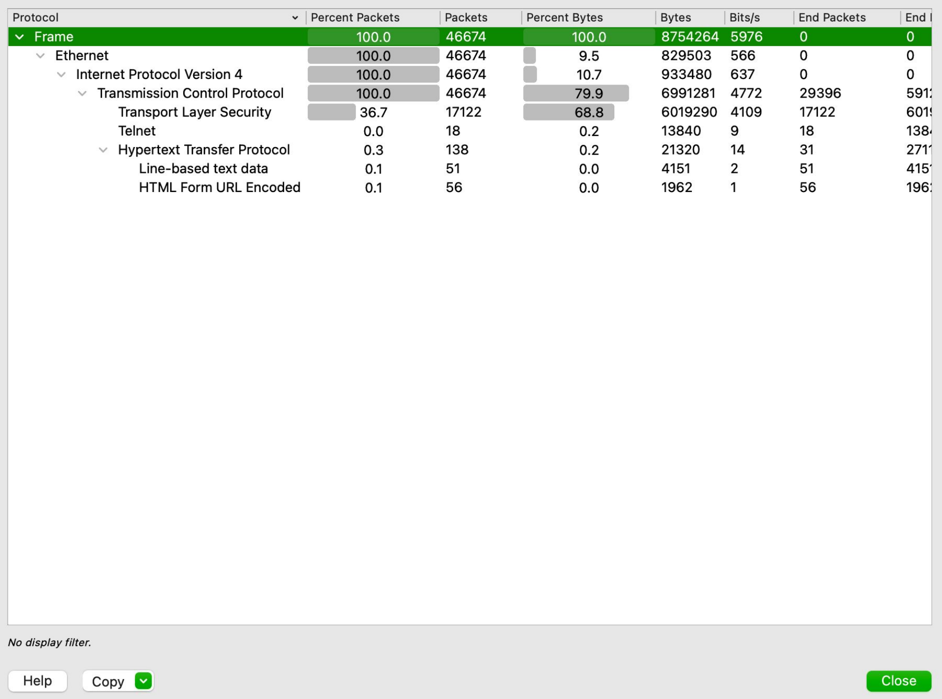
```
0000  3c 9c 0f 1a 39 f0 dc 77  4c a3 87 76 08 00 45 00   <···9··w L··v··E·
0010  00 c1 00 00 00 00 40 06  00 00 f3 df 4d 27 12 dd   ······@· ····M'··
0020  ae 7c b3 73 00 50 bb d6  67 75 00 00 00 15 50 18   ·|·s·P·· gu····P·
0030  16 d0 fd 98 00 00 50 4f  53 54 20 2f 6c 6f 67 69   ······PO ST /logi
0040  6e 2f 20 48 54 54 50 2f  31 2e 31 0d 0a 48 6f 73   n/ HTTP/ 1.1··Hos
0050  74 3a 20 61 64 6d 69 6e  2e 37 37 36 69 6e 63 2e   t: admin .776inc.
0060  63 6f 6d 0d 0a 43 6f 6e  74 65 6e 74 2d 4c 65 6e   com··Con tent-Len
0070  67 74 68 3a 20 33 35 0d  0a 43 6f 6e 74 65 6e 74   gth: 35· ·Content
0080  2d 54 79 70 65 3a 20 61  70 70 6c 69 63 61 74 69   -Type: a pplicati
0090  6f 6e 2f 78 2d 77 77 77  2d 66 6f 72 6d 2d 75 72   on/x-www -form-ur
00a0  6c 65 6e 63 6f 64 65 64  0d 0a 0d 0a 75 73 65 72   lencoded ····user
00b0  6e 61 6d 65 3d 75 73 65  72 35 37 26 70 61 73 73   name=use r57&pass
00c0  77 6f 72 64 3d 70 61 73  73 77 6f 72 64 35 37      word=pas sword57
```

Application layer

Corresponding raw bytes

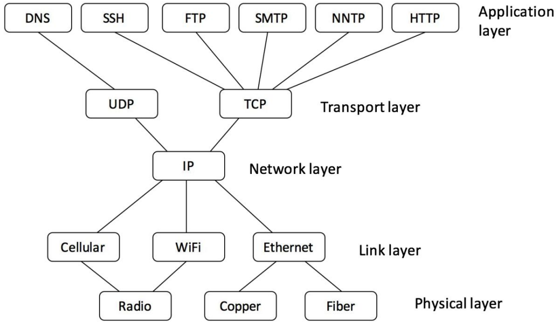
Wireshark Demo: Protocol Hierarchy

Statistics – Protocol Hierarchy:



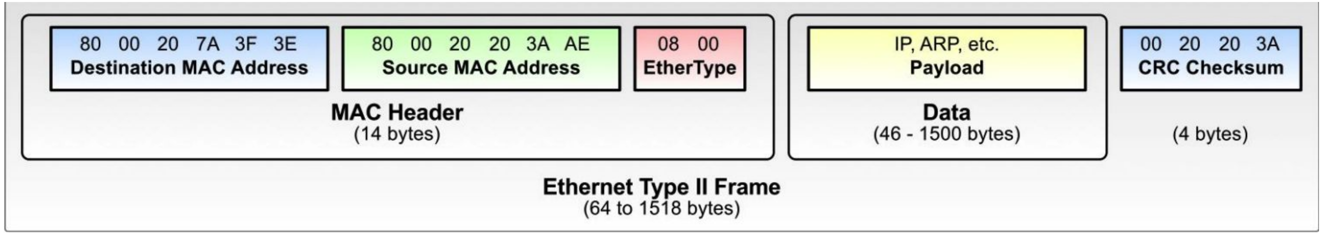
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End
Frame	100.0	46674	100.0	8754264	5976	0	0
Ethernet	100.0	46674	9.5	829503	566	0	0
Internet Protocol Version 4	100.0	46674	10.7	933480	637	0	0
Transmission Control Protocol	100.0	46674	79.9	6991281	4772	29396	5912
Transport Layer Security	36.7	17122	68.8	6019290	4109	17122	6019
Telnet	0.0	18	0.2	13840	9	18	1384
Hypertext Transfer Protocol	0.3	138	0.2	21320	14	31	2711
Line-based text data	0.1	51	0.0	4151	2	51	4151
HTML Form URL Encoded	0.1	56	0.0	1962	1	56	1962

No display filter.



Review: Link Layer/Ethernet

- Ethernet is the most common protocol
- Provides connectivity between hosts and routers
- You can find MAC addresses here
    - It is unique identifier assigned to a network interface controller
    - Assigned at the hardware/physical level (e.g. the WiFi card of your laptop)



80 00 20 7A 3F 3E
Destination MAC Address

80 00 20 20 3A AE
Source MAC Address

08 00
EtherType

MAC Header
(14 bytes)

IP, ARP, etc.
Payload

Data
(46 - 1500 bytes)

00 20 20 3A
CRC Checksum

(4 bytes)

Ethernet Type II Frame
(64 to 1518 bytes)

Review: IP (Internet Protocol)

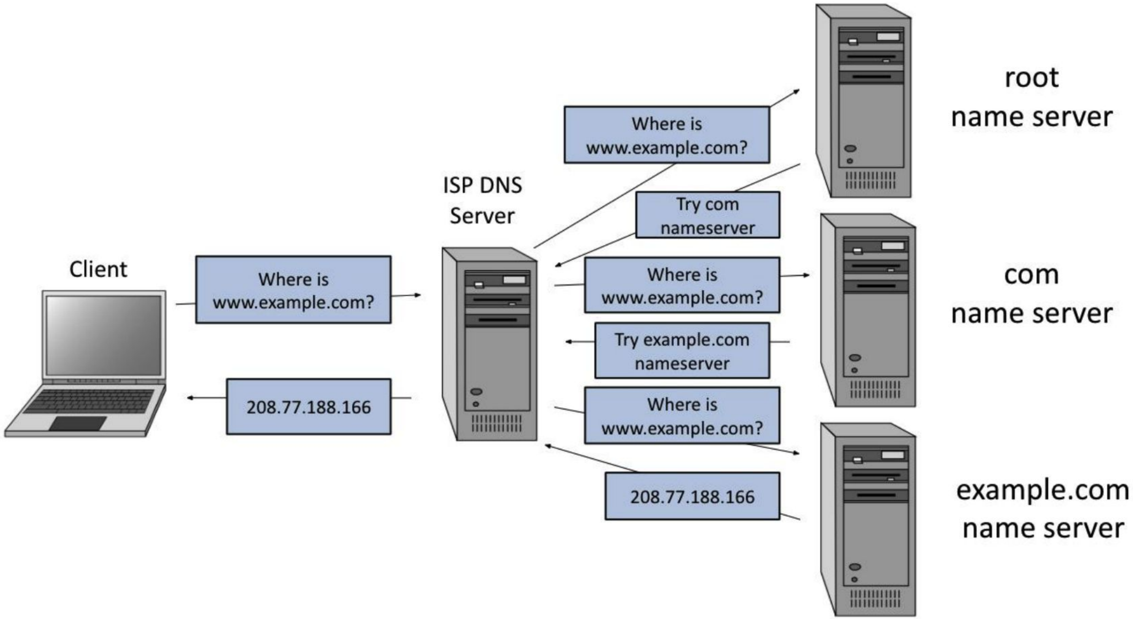
- Responsible for delivering packets from source hosts to destination host
- Every host has a unique identifier known as an IP address
- IP by itself is unreliable
  - Packets may be dropped, reordered, duplicated, or corrupted
  - No acknowledgements provided
- You can arbitrarily change the source IP; routers do not verify source IP
  - This can lead to Denial of Service attacks
- Each packet is sent independent of other packets
- You may encounter both IPv4 packets (32-bit addresses) or IPv6 packets (128-bit addresses)

Review: Transport Layer

- TCP and UDP are two of the protocols within the Transport layer
- TCP is connection-oriented and reliable
    - It is useful when you require all data to be transmitted, in a consistent order
    - E.g. HTML, pictures, etc.
- UDP is packet-oriented, not reliable or ordered
    - UDP is useful when you want to keep up with something in “real-time”, or when you have simple, short query/answer requests
    - E.g. Video streaming: you don’t care if a couple frames are lost, just that the video stays streaming

Review: Domain Name System

- Distributed database for resolving domain names to IP addresses
- Hierarchical organization
- Uses UDP for **speed**
  (minimize latency)


Client
Where is www.example.com?
208.77.188.166
ISP DNS Server
Where is www.example.com?
Try com nameserver
Where is www.example.com?
Try example.com nameserver
Where is www.example.com?
208.77.188.166
root name server
com name server
example.com name server

No Labs Next Two Weeks!