

# Digital Forensics & Packet Analysis

UBNetDef, Fall 2021  
Week 14

# About Me

- Education
  - Bachelor of Science, Business Administration
  - Master of Science, MIS
- Security Experience
  - Consultant/Senior Consultant, Cyber Risk services, Deloitte
  - Lead Cybersecurity Consultant and vCISO, Loptr LLC
- Professional Affiliations
  - ISC^2; Certified Information Systems Security Professional (CISSP)
  - Buffalo Electronic Crimes Task Force
- Publications:
  - Vulnerability Assessment (ISACA, 2017)
- Hats worn:
  - Virtual CISO
  - Project Manager
  - Security Analyst
  - Security Monitoring Analyst
  - Security Architect
  - Software Developer



Dominic Sellitto, CISSP



Clinical Assistant Professor  
Program Director, MS Business Analytics

## Skills

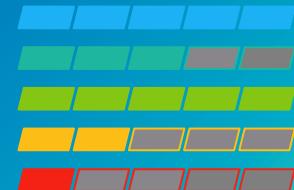
Strategy

Tech

Risk

Dev

Sports



# Agenda

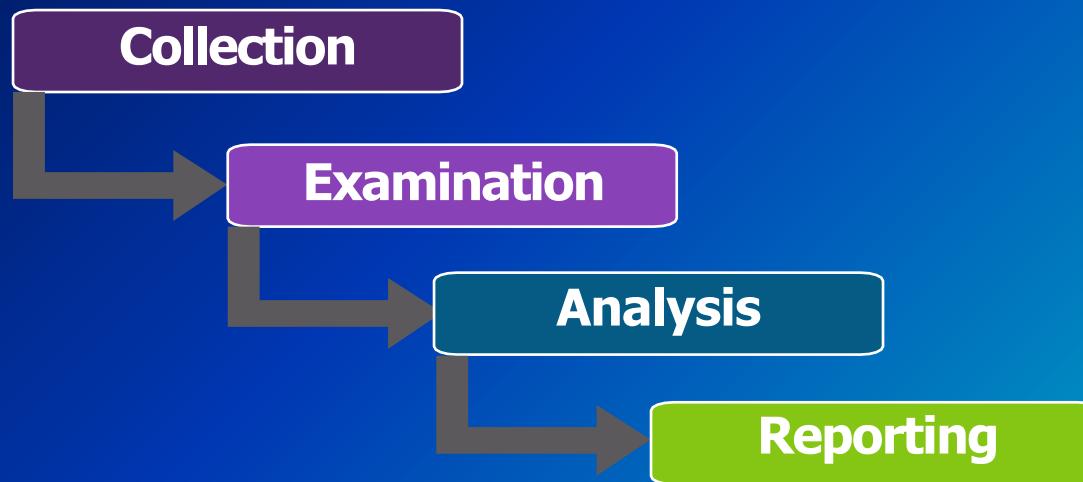
1. Digital Forensics Overview
2. Subdomains within Digital Forensics
3. Network Forensics Overview
4. Wireshark Exercise 1
5. Wireshark Exercise 2
6. Homework

# What is Digital Forensics

- Digital Forensics is “the application of science to the identification, collection, examination, and analysis of data while preserving the integrity of the information and maintaining a strict chain of custody for the data.”
  - ◊ NIST SP-800-86, Guide to Integrating Forensic Techniques Into Incident Response (Pg. 15)
- Digital Forensics may also be referred to as:
  - ◊ Computer and Network Forensics
  - ◊ Data Forensics

# Phases of the Forensics Process

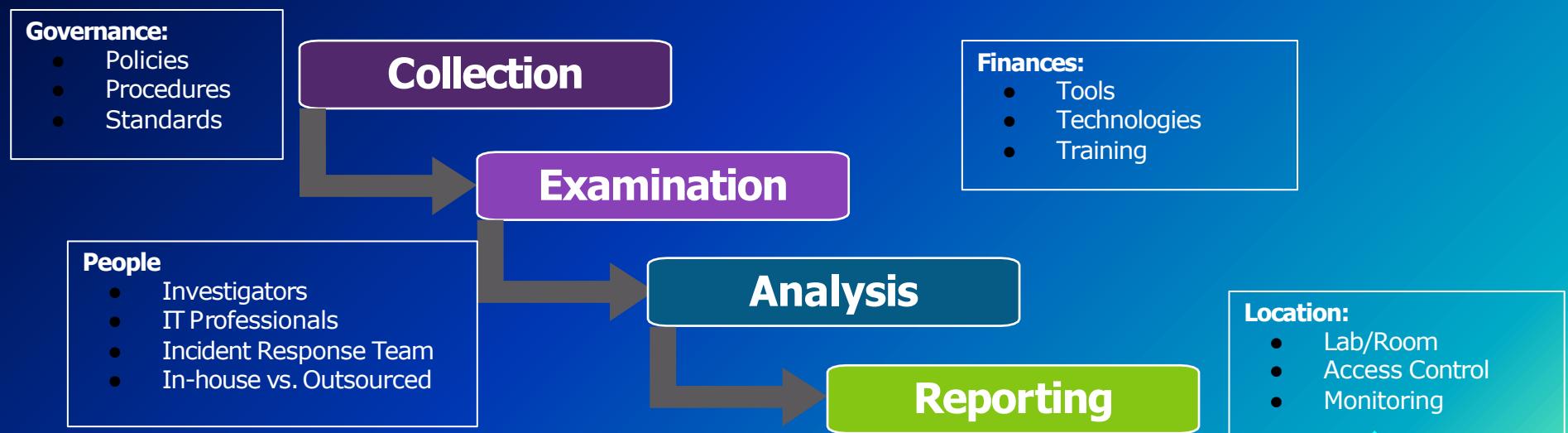
- **NIST 800-86: Guide to Integrating Forensic Techniques into Incident Response** describes the 4 phases of the forensics process as follows:



Source: NIST 800-86: Guide to Integrating Forensic Techniques into Incident Response

# Enabling Factors

- In order to repeatedly execute the process, you need some things...



Source: NIST 800-86: Guide to Integrating Forensic Techniques into Incident Response

# Forensic Areas of Practice

- You might just think of forensics as examining hard drives, but it's much more than that:



Media Forensics



Malware Analysis



Memory Forensics



Network Forensics



Mobile Forensics



Cloud Forensics



Email Forensics



Digital Media Manipulation



IoT Forensics



Automobile Forensics

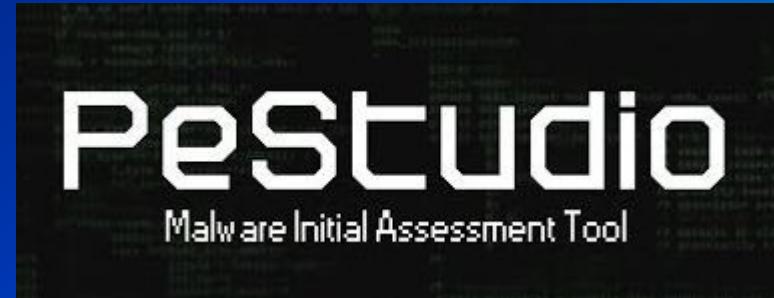
# Digital Media Manipulation

- ◇ Which of these is fake?



# Malware Analysis...

- What's that program **really** doing?



# Email Forensics...

Job Opportunity Inbox ☆

Elizabeth Stout 9:44 PM to me ...

This message was not sent to Spam based on your organization's settings.

Good day,

I'm a student here. I have an uncle who is moving to the school area and needs someone to care for his 2 dogs. He needs someone to walk and bath them three times a week. He is offering \$300 weekly, including \$50 for gas. If you're interested, email him ([jeremystout62@aol.com](mailto:jeremystout62@aol.com)).

Since this is a private job, please contact him with your personal email and provide a phone number for faster communication.

All the best.

# Network Forensics

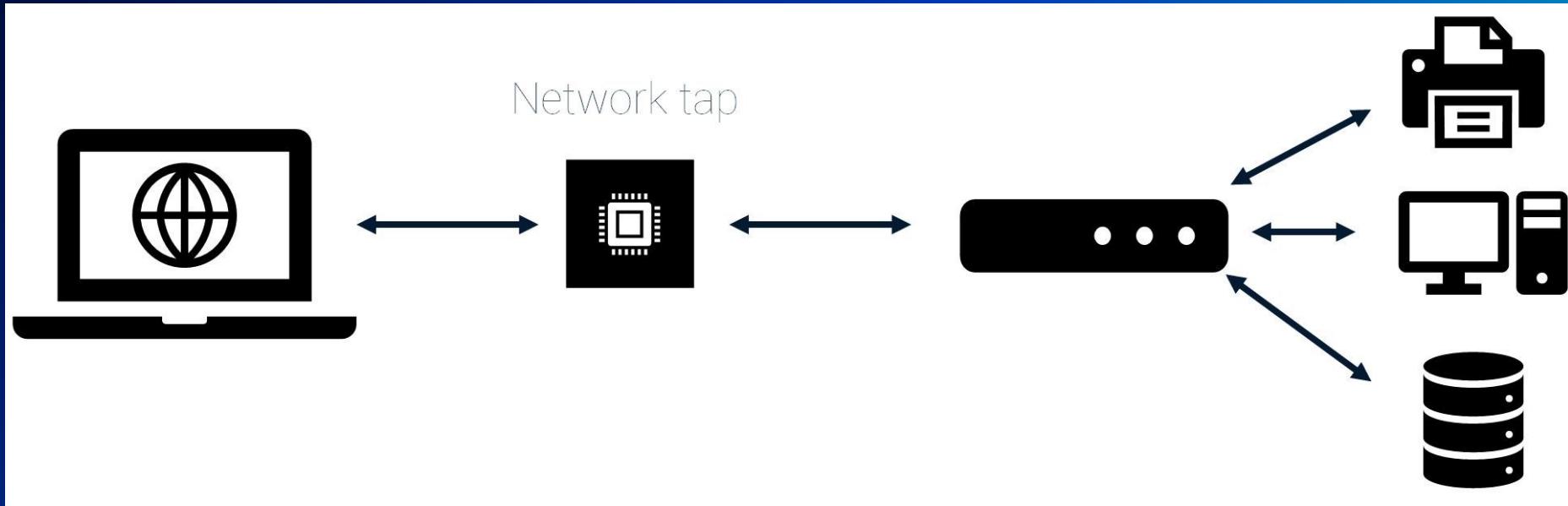
- **Packets** contain all of the information being sent across a network, including the source and destination machine, protocol being used, and the actual data being sent
- **Network logs** are records of network events - they tell you that something happened over the network (like source, destination, protocol) but do not contain the actual data that was sent

# Network Forensics- Capturing Packets

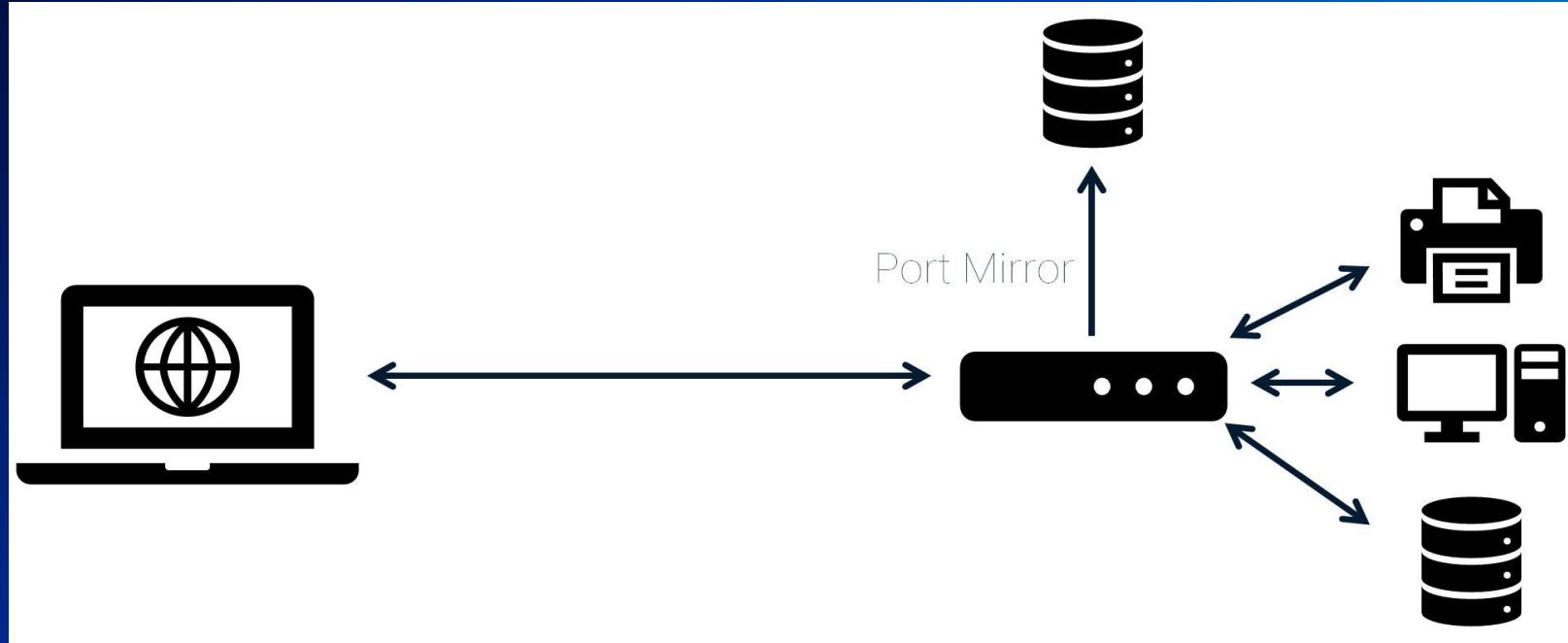
Packets can be captured using a variety of methods:

- hexagon Network tap
  - hexagon A device placed between two networked devices that captures traffic flowing between
- hexagon Port Mirroring
  - hexagon Sends “copies” of packets flowing through a network switch to a specified location (e.g., packet capture server)
- hexagon Wireless Sniffing
  - hexagon Listens over a wireless network for traffic and captures packets

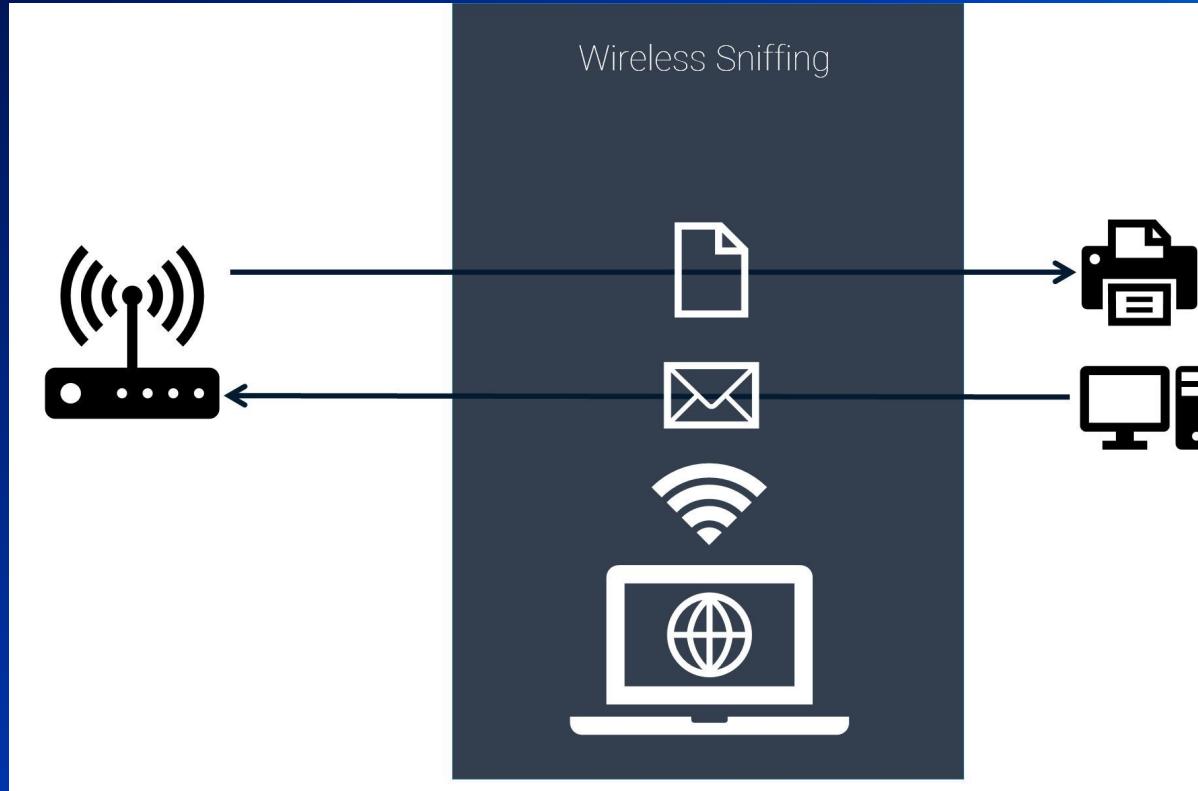
# Network Forensics- Capturing Packets



# Network Forensics- Capturing Packets



# Network Forensics- Capturing Packets



# Packets: Examination and Analysis

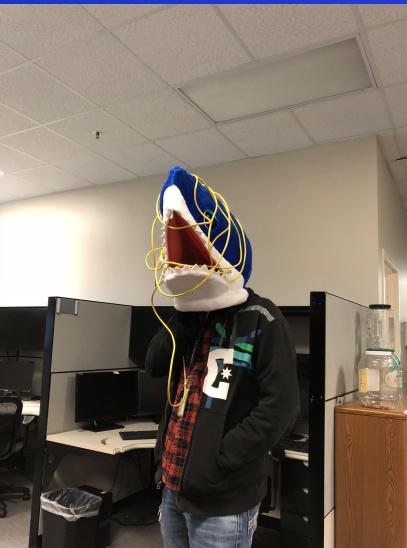
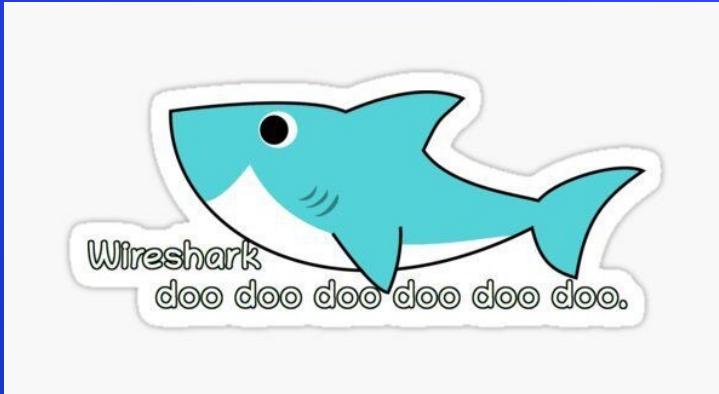
- You can use a packet analyzer, like **Wireshark**, to dive into packets to identify what data was transmitted over a network

The screenshot shows the Wireshark interface with three panels: 'Packets', 'Details', and 'Hexadecimal'.  
The 'Packets' panel displays a list of network frames. Frame 9 is selected, showing details for a TCP SYN packet from 10.1.1.101 to 209.225.11.237.  
The 'Details' panel provides a detailed breakdown of the selected frame, including its bytes on wire and captured, source and destination MAC addresses, and protocol information.  
The 'Hexadecimal' panel shows the raw byte sequence of the selected frame.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.101	10.1.1.1	TCP	62	3177 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
2	0.000651	10.1.1.1	10.1.1.101	TCP	62	80 → 3177 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
3	0.000697	10.1.1.101	10.1.1.1	TCP	54	3177 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	0.013669	10.1.1.101	10.1.1.1	HTTP	530	GET / HTTP/1.1
5	0.014730	10.1.1.1	10.1.1.101	TCP	60	80 → 3177 [ACK] Seq=1 Ack=477 Win=6432 Len=0
6	0.032289	10.1.1.1	10.1.1.101	HTTP	489	HTTP/1.1 200 OK (text/html)
7	0.032346	10.1.1.1	10.1.1.101	TCP	60	80 → 3177 [FIN, ACK] Seq=436 Ack=477 Win=6432 Len=0
8	0.032407	10.1.1.101	10.1.1.1	TCP	54	3177 → 80 [ACK] Seq=477 Ack=437 Win=65100 Len=0
9	0.121783	10.1.1.101	209.225.11.237	TCP	62	3179 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
10	0.136302	10.1.1.101	10.1.1.1	TCP	54	3177 → 80 [FIN, ACK] Seq=477 Ack=437 Win=65100 Len=0

► Frame 9: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)  
► Ethernet II, Src: SmcNetwo\_22:5a:03 (00:04:e2:22:5a:03), Dst: D-Link\_6f:d7:c1 (00:05:5d:6f:d7:c1)  
► Internet Protocol Version 4, Src: 10.1.1.101, Dst: 209.225.11.237  
► Transmission Control Protocol, Src Port: 3179, Dst Port: 80, Seq: 0, Len: 0

Hex	Dec	Printable
0000	00 05 5d 6f d7 c1 00 04	..lo... "Z ..E..
0010	00 30 b3 0a 40 00 80 06	·0 ..@ ..^ ..e ..
0020	0b ed 0c 6b 00 50 34 9d	..k-P4 \ ..p ..
0030	00 00 fb d6 00 00 02 04	.....

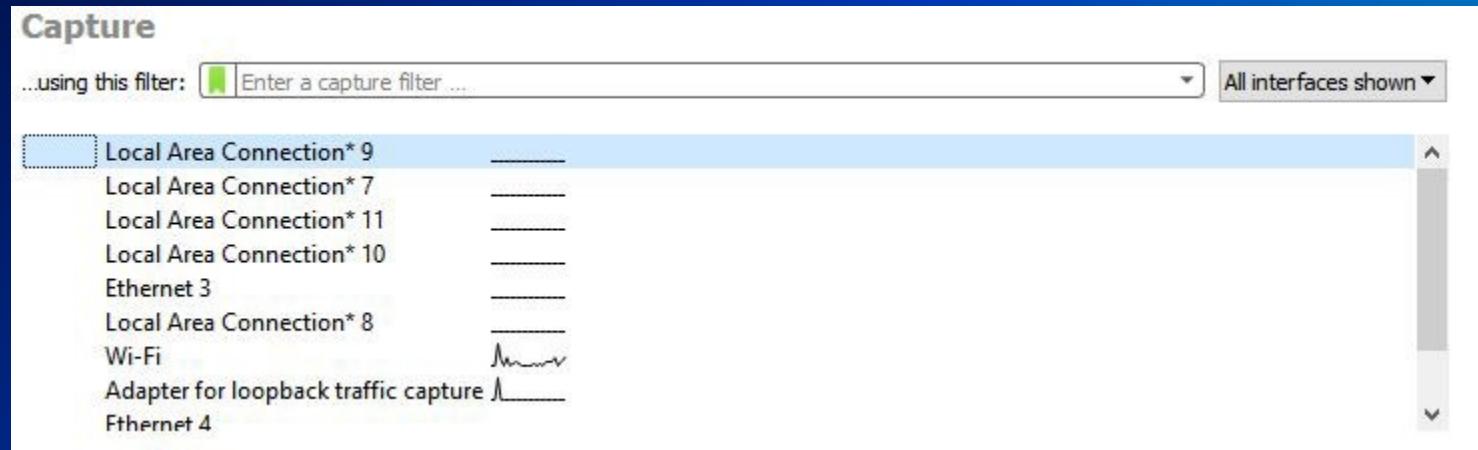


# Break Time!



# Demo: Introduction to Wireshark

# Capturing packets from your network adapter



# Stopping your packet capture and examining results...

The screenshot shows a Wireshark capture window titled "Wi-Fi". The main pane displays a list of network packets, with the 164th packet selected. The packet details pane at the bottom shows the User Datagram Protocol (Src Port: 55886, Dst Port: 53) and the Domain Name System (query). The bytes pane shows the raw hex and ASCII data for the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
161	19:20:15.575108			DNS	107	Standard query response 0x1c6e AAAA gsas.nyu.edu CNAME
162	19:20:15.575697			DNS	71	Standard query 0x93fa AAAA twitter.com
163	19:20:15.580920			DNS	156	Standard query response 0x7a74 AAAA cas.nyu.edu CNAME
164	19:20:15.581425			DNS	72	Standard query 0xdffe AAAA home.nyu.edu
165	19:20:15.586816			DNS	157	Standard query response 0x1c6e AAAA gsas.nyu.edu CNAME
166	19:20:15.603485			TCP	74	55981 → 80 [ACK] Seq=516 Ack=4990 Win=64252 Len=0
167	19:20:15.608104			DNS	72	Standard query response 0xdffe AAAA home.nyu.edu
168	19:20:15.608802			DNS	73	Standard query 0x2596 AAAA medli.nyu.edu
169	19:20:15.614226			DNS	137	Standard query response 0x7784 A www.instagram.com CNAME
170	19:20:15.618414			DNS	77	Standard query 0x15a2 AAAA www.instagram.com
171	19:20:15.620176			DNS	73	Standard query response 0x2596 AAAA medli.nyu.edu
172	19:20:15.620767			DNS	75	Standard query 0x0110 AAAA nursing.nyu.edu
173	19:20:15.621333			DNS	143	Standard query response 0x93fa AAAA twitter.com SOA
174	19:20:15.621761			DNS	67	Standard query 0x6baf A nyu.edu
175	19:20:15.621862			DNS	67	Standard query 0xd581 AAAA nyu.edu
176	19:20:15.635305			DNS	75	Standard query response 0x0110 AAAA nursing.nyu.edu
177	19:20:15.635875			DNS	73	Standard query 0xf669 A nyuad.nyu.edu
178	19:20:15.635977			DNS	73	Standard query 0x65c7 AAAA nyuad.nyu.edu
179	19:20:15.640669			DNS	149	Standard query response 0x15a2 AAAA www.instagram.com
180	19:20:15.641173			DNS	80	Standard query 0x9cac AAAA publichealth.nyu.edu
181	19:20:15.653003			DNS	67	Standard query response 0xd581 AAAA nyu.edu
182	19:20:15.675812			DNS	80	Standard query response 0x9cac AAAA publichealth.nyu.edu

User Datagram Protocol, Src Port: 55886, Dst Port: 53  
Domain Name System (query)

```
0000 b8 27 eb 4e a4 de c8 58 c0 25 ac 4c 08 00 45 00 ..N..X %.L..E.
0010 00 3a 29 18 00 00 80 11 00 00 a8 00 d7 c0 a8 ::).... .
0020 00 fa da 4e 00 35 00 26 83 59 df fe 01 00 00 01 ..N 5 & Y....
0030 00 00 00 00 00 00 04 68 6f 6d 65 03 6e 79 75 03 .....h ome.nyu.
0040 65 64 75 00 00 1c 00 01 edu.....
```

wireshark\_Wi-FNZvF20.pcapng

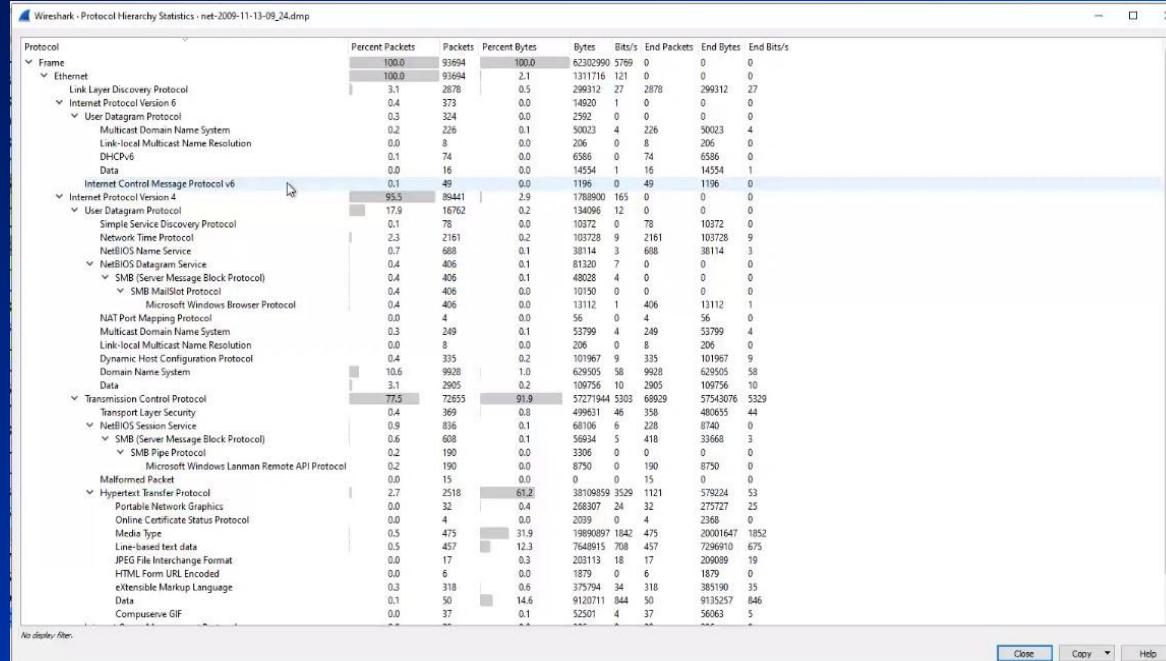
Packets: 247 · Displayed: 247 (100.0%) · Dropped: 0 (0.0%)

# Overview of the packet list

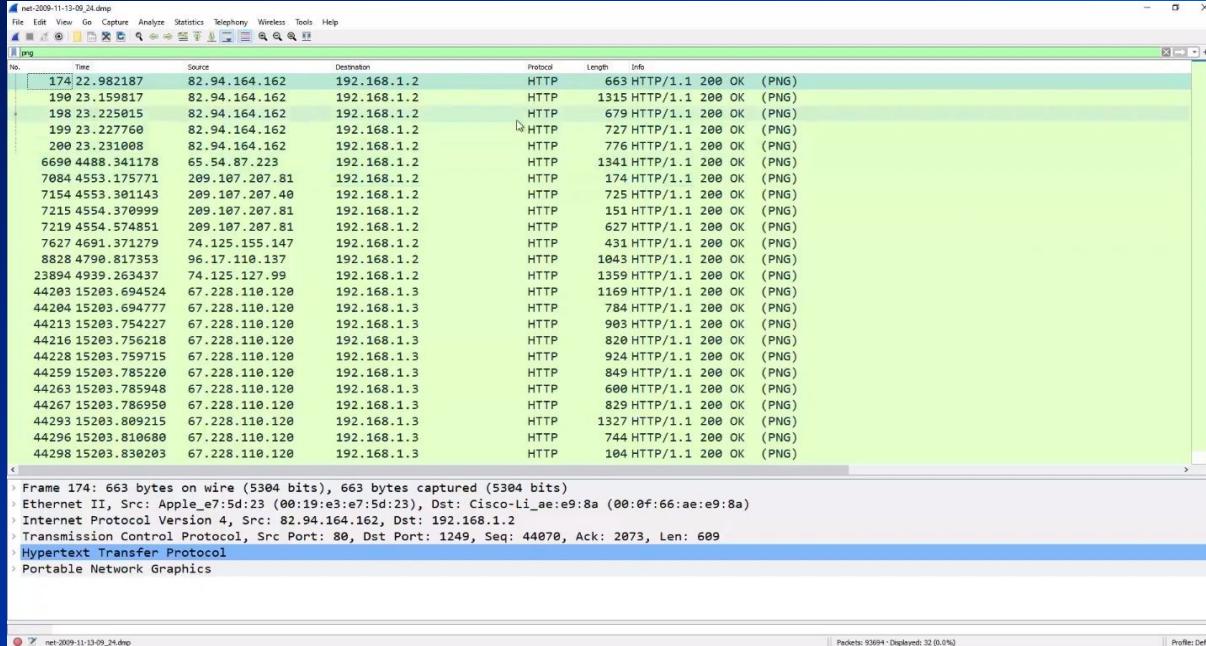
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	SMCNetwo_81:db:10	LLDP_Multicast	LLDP	118	MA:00:22:2d:81:db:10 LA/1 120 SysN=SMCGS8P-Smart SysD=SMCGS8P-Smart - SMCGS8P-Smart
2	14.497478	192.168.1.1	224.0.0.1	UDP	75	626 → 626 Len=33
3	19.384386	192.168.1.2	4.2.2.2	DNS	79	Standard query 0x2c8c A wiki.github.com
4	19.309878	192.168.1.2	4.2.2.2	DNS	82	Standard query 0xf06f A addons.mozilla.org
5	19.314981	4.2.2.2	192.168.1.2	DNS	127	Standard query response 0xf06f A addons.mozilla.org CNAME amo.glb.mozilla.org
6	19.319872	192.168.1.2	4.2.2.2	DNS	85	Standard query 0x3b5a A en-us.www.mozilla.com
7	19.324718	4.2.2.2	192.168.1.2	DNS	97	Standard query response 0x3b5a A en-us.www.mozilla.com A 63.245.209.10
8	19.331614	192.168.1.2	63.245.209.91	TCP	66	1245 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
9	19.332110	192.168.1.2	63.245.209.10	TCP	66	1246 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
10	19.338460	63.245.209.10	192.168.1.2	TCP	58	89 → 1246 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1380
11	19.339201	63.245.209.91	192.168.1.2	TCP	58	443 → 1245 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380
12	19.340109	192.168.1.2	63.245.209.10	TCP	64	1246 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
13	19.341109	192.168.1.2	63.245.209.91	TCP	64	1245 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
14	19.341856	192.168.1.2	63.245.209.10	HTTP	793	GET /favicon.ico HTTP/1.1
15	19.342354	192.168.1.2	63.245.209.91	TLSv1	227	Client Hello
16	19.348455	63.245.209.10	192.168.1.2	HTTP	265	HTTP/1.1 304 Not Modified
17	19.351441	63.245.209.91	192.168.1.2	TCP	54	443 → 1245 [ACK] Seq=1 Ack=170 Win=6432 Len=0
18	19.351697	63.245.209.91	192.168.1.2	TLSv1	1434	Server Hello
19	19.351707	63.245.209.91	192.168.1.2	TCP	1434	443 → 1245 [ACK] Seq=1381 Ack=170 Win=6432 Len=1380 [TCP segment of a reassembly]
20	19.353853	192.168.1.2	63.245.209.91	TCP	64	1245 → 443 [ACK] Seq=170 Ack=2761 Win=65535 Len=0
21	19.359448	63.245.209.91	192.168.1.2	TCP	1434	443 → 1245 [ACK] Seq=2761 Ack=170 Win=6432 Len=1380 [TCP segment of a reassembly]
22	19.359456	63.245.209.91	192.168.1.2	TLSv1	365	Certificate, Server Hello Done
23	19.361347	192.168.1.2	63.245.209.91	TCP	64	1245 → 443 [ACK] Seq=170 Ack=4452 Win=65535 Len=0
24	19.401571	192.168.1.2	4.2.2.2	DNS	81	Standard query 0x3fa5 A oscp.verisign.com



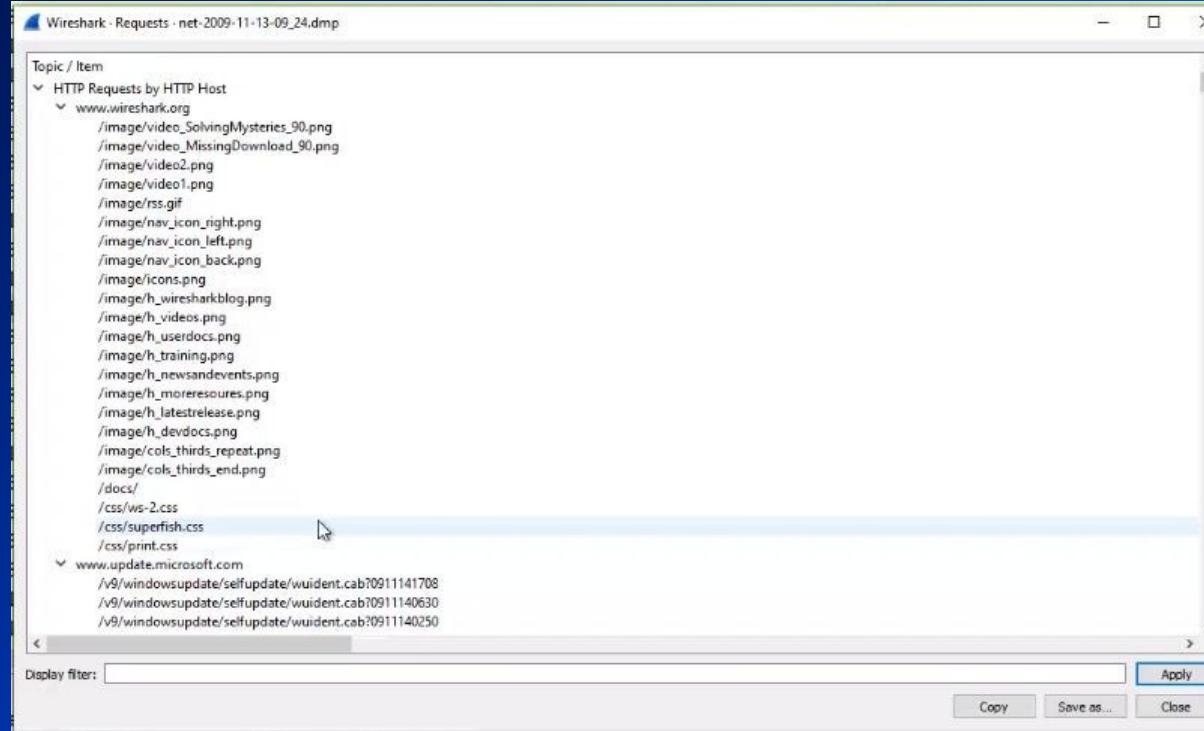
# Protocol hierarchy statistics



# Display filters



# HTTP request statistics



# String searches

The screenshot shows a Wireshark capture window titled "net-2009-11-13-09\_24.dmp". A search bar at the top right contains the string "49281-49360". The results pane displays a list of network frames, with frame 39687 highlighted in yellow. This frame is a TCP GET request from 192.168.1.2 to 74.125.15.18, port 80, containing the URL "/safebrowsing/rd/goog-phish-shavar\_s\_49281-49360.49281-49360.. HTTP/1.1". The details pane shows the full HTTP header and the body of the request. The bottom status bar indicates 77 bytes captured and 5912 bits on wire.

No.	Time	Source	Destination	Protocol	Length	Info
39675	9526.488144	4.2.2.2	192.168.1.2	DNS	140	Standard query response 0x5011 A safebrowsing-cache.google.com CNAME safebrowsing-cache.google.com
39676	9526.491298	192.168.1.2	74.125.15.18	TCP	66	1305 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
39677	9526.496893	74.125.15.18	192.168.1.2	TCP	62	80 → 1305 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
39678	9526.498302	192.168.1.2	74.125.15.18	TCP	64	1305 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
39679	9526.499298	192.168.1.2	74.125.15.18	HTTP	739	GET /safebrowsing/rd/goog-phish-shavar_s_49241-49280.49241-49280.. HTTP/1.1
39680	9526.504693	74.125.15.18	192.168.1.2	TCP	54	80 → 1305 [ACK] Seq=1 Ack=682 Win=6810 Len=0
39681	9526.505662	74.125.15.18	192.168.1.2	TCP	290	80 → 1305 [PSH, ACK] Seq=1 Ack=682 Win=6810 Len=236 [TCP segment of a reassembly]
39682	9526.505885	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=237 Ack=682 Win=6810 Len=1380 [TCP segment of a reassembly]
39683	9526.505895	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=1617 Ack=682 Win=6810 Len=1380 [TCP segment of a reassembly]
39684	9526.508038	192.168.1.2	74.125.15.18	TCP	64	1305 → 80 [ACK] Seq=682 Ack=2997 Win=65535 Len=0
39685	9526.513381	74.125.15.18	192.168.1.2	HTTP	1177	HTTP/1.1 200 OK (application/vnd.google.safebrowsing-chunk)
39686	9526.664438	192.168.1.2	74.125.15.18	TCP	64	1305 → 80 [ACK] Seq=682 Ack=4120 Win=64412 Len=0
39687	9527.092928	192.168.1.2	74.125.15.18	HTTP	739	GET /safebrowsing/rd/goog-phish-shavar_s_49281-49360.49281-49360.. HTTP/1.1
39688	9527.100046	74.125.15.18	192.168.1.2	TCP	288	80 → 1305 [PSH, ACK] Seq=4120 Ack=1363 Win=8172 Len=234 [TCP segment of a reassembly]
39689	9527.100057	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=4354 Ack=1363 Win=8172 Len=1380 [TCP segment of a reassembly]
39690	9527.100066	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=5734 Ack=1363 Win=8172 Len=1380 [TCP segment of a reassembly]
39691	9527.100261	74.125.15.18	192.168.1.2	TCP	1390	80 → 1305 [PSH, ACK] Seq=7114 Ack=1363 Win=8172 Len=1336 [TCP segment of a reassembly]
39692	9527.102660	192.168.1.2	74.125.15.18	TCP	64	1305 → 80 [ACK] Seq=1363 Ack=7114 Win=65535 Len=0
39693	9527.108015	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=8450 Ack=1363 Win=8172 Len=1380 [TCP segment of a reassembly]
39694	9527.108025	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=9830 Ack=1363 Win=8172 Len=1380 [TCP segment of a reassembly]
39695	9527.108033	74.125.15.18	192.168.1.2	TCP	1434	80 → 1305 [ACK] Seq=11210 Ack=1363 Win=8172 Len=1380 [TCP segment of a reassembly]
39696	9527.108051	74.125.15.18	192.168.1.2	HTTP	197	HTTP/1.1 200 OK (application/vnd.google.safebrowsing-chunk)
39697	9527.110909	192.168.1.2	74.125.15.18	TCP	64	1305 → 80 [ACK] Seq=1363 Ack=11210 Win=65535 Len=0
				TCP	64	1305 → 80 [ACK] Seq=1363 Ack=11779 Win=65535 Len=0

Frame 39687: 739 bytes on wire (5912 bits), 739 bytes captured (5912 bits)  
Ethernet II, Src: Cisco-Li\_ae:e9:8a (00:0f:66:ae:e9:8a), Dst: Apple\_e7:5d:23 (00:19:e3:e7:5d:23)  
Internet Protocol Version 4, Src: 192.168.1.2, Dst: 74.125.15.18  
Transmission Control Protocol, Src Port: 1305, Dst Port: 80, Seq: 682, Ack: 4120, Len: 681  
Hypertext Transfer Protocol  
GET /safebrowsing/rd/goog-phish-shavar\_s\_49281-49360.49281-49360.. HTTP/1.1\r\nHost: safebrowsing-cache.google.com\r\nUser-Agent: Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5\r\n

# Time settings

net-2009-11-13-09\_24.dmp

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter: <Ctrl>/>

No.	Time	Source	Destination	Protocol	Length	Info
1	17:25:02.901824	SMCNetwo_81:db:10	LLDP_Multicast	LLDP	118	MA/00:22:2d:81:db:10 LA/1 120 SysN=SMCGSBP-Smart SysD=SMCGSBP-Smart
2	17:25:17.399302	192.168.1.1	224.0.0.1	UDP	75	626 → 626 Len=33
3	17:25:22.206210	192.168.1.2	4.2.2.2	DNS	79	Standard query 0x2c8c A wiki.github.com
4	17:25:22.211702	192.168.1.2	4.2.2.2	DNS	82	Standard query 0xf06f A addons.mozilla.org
5	17:25:22.216805	4.2.2.2	192.168.1.2	DNS	127	Standard query response 0xf06f A addons.mozilla.org CNAME amo.glib.mozilla.org
6	17:25:22.221696	192.168.1.2	4.2.2.2	DNS	85	Standard query 0x3b5a A en-us.www.mozilla.com
7	17:25:22.226542	4.2.2.2	192.168.1.2	DNS	97	Standard query response 0x3b5a A en-us.www.mozilla.com A 63.245.209.10
8	17:25:22.233438	192.168.1.2	63.245.209.91	TCP	66	1245 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
9	17:25:22.233934	192.168.1.2	63.245.209.10	TCP	66	1246 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
10	17:25:22.240284	63.245.209.10	192.168.1.2	TCP	58	80 → 1246 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1380
11	17:25:22.241075	63.245.209.91	192.168.1.2	TCP	58	1246 → 80 [ACK] Seq=1 Ack=1 Win=8190 Len=0 MSS=1380

Frame 3: 79 bytes on wire (632 bits), 79 bytes captured (632 bits)  
Encapsulation type: Ethernet (1)  
Arrival Time: Nov 13, 2009 12:25:22.206210000 Eastern Standard Time  
[Time shift for this packet: 0.000000000 seconds]  
Epoch Time: 1258133122.206210000 seconds  
[Time delta from previous captured frame: 4.806908000 seconds]  
[Time delta from previous displayed frame: 4.806908000 seconds]  
[Time since reference or first frame: 19.304386000 seconds]  
Frame Number: 3  
Frame Length: 79 bytes (632 bits)  
Capture Length: 79 bytes (632 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:ethertype:ip:udp:dns]  
[Coloring Rule Name: UDP]  
[Coloring Rule String: udp]  
Ethernet II, Src: Cisco-L1\_ae:e9:8a (00:0f:66:ae:e9:8a), Dst: Apple\_e7:5d:23 (00:19:e3:e7:5d:23)  
Internet Protocol Version 4, Src: 192.168.1.2, Dst: 4.2.2.2  
User Datagram Protocol, Src Port: 56242, Dst Port: 53  
Domain Name System (query)

Ready to load or capture

Packets: 93694 - Displayed: 93694 (100.0%)

Profile: Default



NetDef

# Packet details

net-2009-11-13-09\_24.dmp

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

(Apply a display filter - <ctrl>f)

No.	Time	Source	Destination	Protocol	Length	Info
1	17:25:02.901824	SMCNetwo_81:db:10	LLDP_Multicast	LLDP	118	MA/00:22:2d:81:db:10 LA/1 120 SysN=SMCGS8P-Smart SysD=SMCGS8P-Smart -
2	17:25:17.399302	192.168.1.1	224.0.0.1	UDP	75	626 → 626 Len=33
3	17:25:22.206210	192.168.1.2	4.2.2.2	DNS	79	Standard query 0x2c8c A wiki.github.com
4	17:25:22.211702	192.168.1.2	4.2.2.2	DNS	82	Standard query 0xf06f A addons.mozilla.org
5	17:25:22.216805	4.2.2.2	192.168.1.2	DNS	127	Standard query response 0xf06f A addons.mozilla.org CNAME amo.glib.mozilla.org
6	17:25:22.221696	192.168.1.2	4.2.2.2	DNS	85	Standard query 0x3b5a A en-us.www.mozilla.com
7	17:25:22.226542	4.2.2.2	192.168.1.2	DNS	97	Standard query response 0x3b5a A en-us.www.mozilla.com A 63.245.209.10
8	17:25:22.233438	192.168.1.2	63.245.209.91	TCP	66	1245 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
9	17:25:22.233934	192.168.1.2	63.245.209.10	TCP	66	1246 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
10	17:25:22.240284	63.245.209.10	192.168.1.2	TCP	58	80 → 1246 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1380
11	17:25:22.240305	63.245.209.10	192.168.1.2	TCP	60	1246 → 80 [ACK] Seq=1 Ack=1 Win=8190 Len=0 MSS=1380

> User Datagram Protocol, Src Port: 56242, Dst Port: 53  
  ` Domain Name System (query)  
    Transaction ID: 0x2c8c  
    Flags: 0x0100 Standard query  
    Questions: 1  
    Answer RRs: 0  
    Authority RRs: 0  
    Additional RRs: 0  
    ` Queries  
      ` wiki.github.com: type A, class IN  
      [Response In: 27]

0000 00 19 e3 e7 5d 23 00 0f 66 ae e9 8a 08 00 45 00 ... ]#.. f .. E  
0010 00 3d 00 5a 00 00 80 11 68 a8 c0 a8 01 02 04 02 .. =Z... h.....  
0020 02 02 db b2 00 35 00 29 91 8b 2c 8c 01 00 00 01 ..... 5 ) .....,  
0030 00 00 00 00 00 04 77 69 6b 69 06 67 69 74 68 ..... w iki.gith  
0040 75 62 03 63 6f 6d 00 00 01 00 01 c1 7f 0b f7 ub.com .....

Text item (hex), 21 bytes

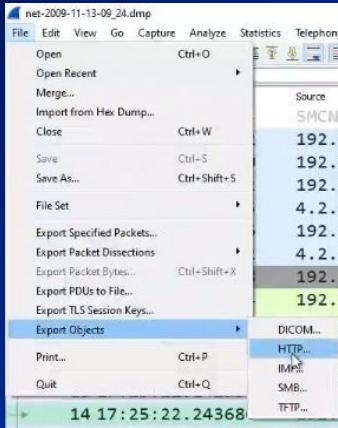
Packets: 93694 - Displayed: 93694 (100.0%)

Profile: Default



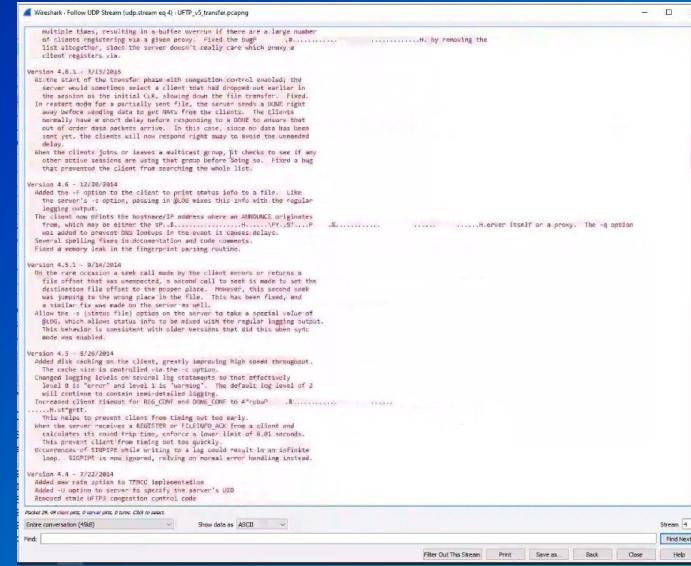
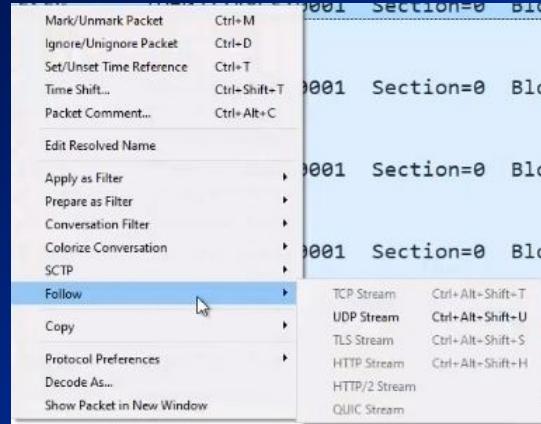
NetDef

# Export objects



Packet	Hostname	Content Type	Size	Filename
31	ocsp.verisign.com	application/ocsp-request	115 bytes	\
45	evsecure-ocsp.verisign.com	application/ocsp-request	115 bytes	\
82	www.python.org	text/html	12kB	2.6.4
105	www.python.org	application/javascript	6902 bytes	iotbs2-core.js
110	www.python.org	text/css	35 bytes	screen-switcher-default.css
113	www.python.org	text/css	1625 bytes	netscape4.css
116	www.python.org	text/css	505 bytes	print.css
117	www.python.org	application/javascript	745 bytes	iotbs2-key-directors-load.js
124	www.python.org	application/javascript	3418 bytes	iotbs2-directors.js
153	www.python.org	text/css	29kB	styles.css
166	www.python.org	image/gif	48 bytes	trans.gif
167	www.python.org	image/gif	2549 bytes	python-logo.gif
174	www.python.org	image/png	247 bytes	header-bg2.png
190	www.python.org	image/png	7796 bytes	donate.png
198	www.python.org	image/png	262 bytes	button-on-bg.png
199	www.python.org	image/png	310 bytes	nav-off-bg.png
200	www.python.org	image/png	359 bytes	nav-on-bg.png
214	www.python.org	text/css	16kB	largestyles.css
218	www.python.org	image/gif	49 bytes	blank.gif
219	www.python.org	image/gif	54 bytes	bullet.gif
222	www.python.org	text/css	16kB	defaultfonts.css
449		text/html	218 bytes	
460	javadvl-esd.sun.com	application/xml	3136 bytes	map-1.6.0.xml
490	content.microsoft.com	text/html	43 bytes	BRI MEM BDRB DEV DACS CIBR

# Following packet streams



# Guided Exercise: Capturing and Analyzing Packets

# Steps:

- Open Wireshark
- Find the active network interface and double-click to start the capture
- Open a web browser and navigate to <http://www.buffalony.gov/>  
Close your browser and stop the packet capture
- Open the protocol hierarchy statistics window and note the protocols
- Filter the packet list to display HTTP traffic
- Run a string search for buffalony.gov
- Export HTTP objects and note the files that were transferred (hint:  
you may have to manually change the filename to a valid image file)

# Summary

- Packets are powerful, but require infrastructure considerations to capture, store, and analyze
- Encrypted traffic can hinder the effectiveness of packet analysis activities
- Tools, like Wireshark, are instrumental in analyzing network traffic and identifying patterns of activity and data transmitted across a network

# Homework

- ◊ You have been provided with a PCAP (packet capture) file. Your job is to review the network traffic to identify the “flags” hidden throughout. There are 5 flags. For each flag you capture, provide the following information:
  - ◊ The flag itself
  - ◊ A brief description of the type of network traffic examined to identify the flag (e.g., protocol, source, destination) (1-2 sentences)
  - ◊ A screenshot of the captured flag (from Wireshark, or exported files from Wireshark)
- ◊ Hints!
  - ◊ Remember the exercises today-- use the Wireshark functions you learned to find the flags!

# Homework

- Flag 1: Haveibeenpwned?
  - ◊ What is the email address of the user involved in this network activity?
- Flag 2: What's the password?
  - ◊ The user logs into a server (or servers) with a specific password, what is it?
- Flag 3: Switching things up...
  - ◊ The user tries to change the above password, but can't. Why not?
- Flag 4: Hidden in plain sight...
  - ◊ The user has accessed three files over FTP, but one in particular looks suspicious. What is the file?
- Flag 5: Higher-level thinking...
  - ◊ What are all of the protocols used throughout this PCAP? Of the identified protocols, document the security significance of at least one of them.

# Additional Resources

- ◇ Autopsy (Digital Forensics Platform and Graphical Interface)
  - ◇ <https://www.autopsy.com/>
- ◇ Wireshark Wiki - Sample Captures
  - ◇ <https://wiki.wireshark.org/SampleCaptures>
- ◇ Wireshark User's Guide (Advanced Topics = Chapter 7)
  - ◇ [https://www.wireshark.org/docs/wsug\\_html/](https://www.wireshark.org/docs/wsug_html/)
- ◇ FTK Imager (Data Preview and Imaging Tool)
  - ◇ [https://accessdata.com/products-services/forensic-toolkit-ftk/ftkimager#:~:text=FTK%20Imager%20is%20a,%C2%AE%20\(FTK\)%20is%20warranted.](https://accessdata.com/products-services/forensic-toolkit-ftk/ftkimager#:~:text=FTK%20Imager%20is%20a,%C2%AE%20(FTK)%20is%20warranted.)

If you have any questions,  
please DM us on mattermost  
or come to Office Hours!