Rami Matouk

Traffic analysis with Wireshark

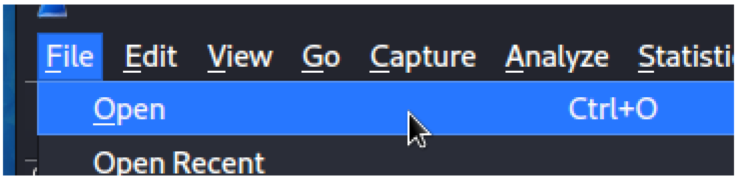
Scenario 1: Data Extraction

We have been provided with pcap file. There is an image that needs to be used as evidence of improper network usage. Our task is to extract image from network traffic.

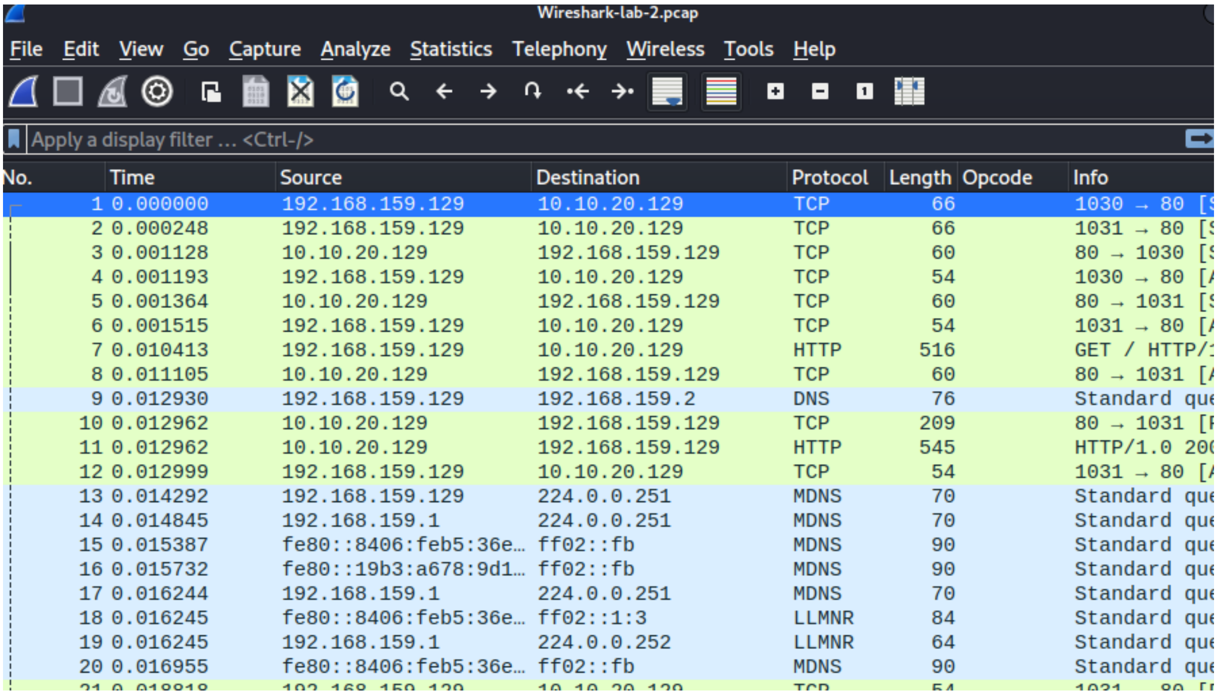
Preparation

Task 1: Open pre-captured file

First let’s open pcap file. Start Wireshark and then select on toolbar File -> Open

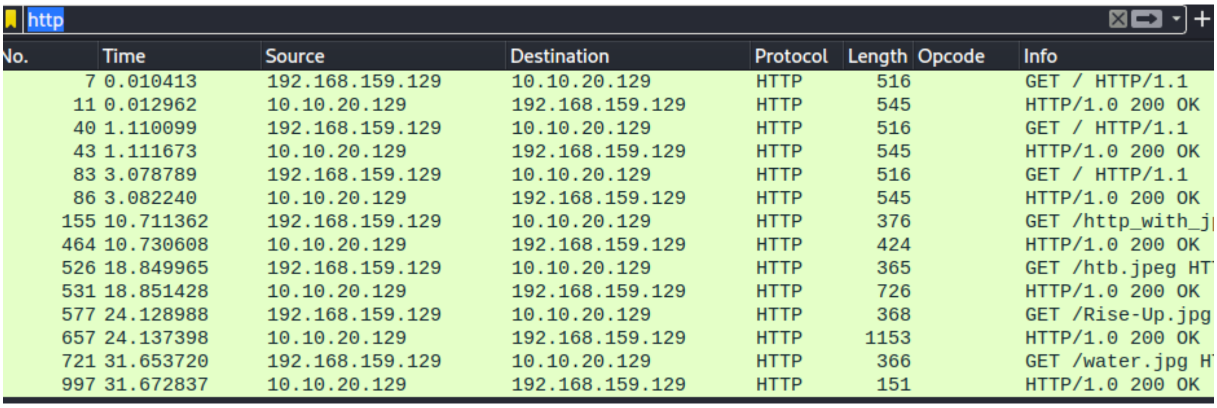


Task 2: Filter results



No.	Time	Source	Destination	Protocol	Length	Opcode	Info
1	0.000000	192.168.159.129	10.10.20.129	TCP	66		1030 → 80 [S
2	0.000248	192.168.159.129	10.10.20.129	TCP	66		1031 → 80 [S
3	0.001128	10.10.20.129	192.168.159.129	TCP	60		80 → 1030 [S
4	0.001193	192.168.159.129	10.10.20.129	TCP	54		1030 → 80 [A
5	0.001364	10.10.20.129	192.168.159.129	TCP	60		80 → 1031 [S
6	0.001515	192.168.159.129	10.10.20.129	TCP	54		1031 → 80 [A
7	0.010413	192.168.159.129	10.10.20.129	HTTP	516		GET / HTTP/1
8	0.011105	10.10.20.129	192.168.159.129	TCP	60		80 → 1031 [A
9	0.012930	192.168.159.129	192.168.159.2	DNS	76		Standard que
10	0.012962	10.10.20.129	192.168.159.129	TCP	209		80 → 1031 [P
11	0.012962	10.10.20.129	192.168.159.129	HTTP	545		HTTP/1.0 200
12	0.012999	192.168.159.129	10.10.20.129	TCP	54		1031 → 80 [A
13	0.014292	192.168.159.129	224.0.0.251	MDNS	70		Standard que
14	0.014845	192.168.159.1	224.0.0.251	MDNS	70		Standard que
15	0.015387	fe80::8406:feb5:36e…	ff02::fb	MDNS	90		Standard que
16	0.015732	fe80::19b3:a678:9d1…	ff02::fb	MDNS	90		Standard que
17	0.016244	192.168.159.1	224.0.0.251	MDNS	70		Standard que
18	0.016245	fe80::8406:feb5:36e…	ff02::1:3	LLMNR	84		Standard que
19	0.016245	192.168.159.1	224.0.0.252	LLMNR	64		Standard que
20	0.016955	fe80::8406:feb5:36e…	ff02::fb	MDNS	90		Standard que
21	0.018818	192.168.159.129	10.10.20.129	TCP	54		1031 → 80 [

As we can see, there is a lot of traffic. In total, there are around 1000 packets, and less than 20 of them are HTTP packets. Our goal is to extract data from HTTP packets. Let’s filter out noise.

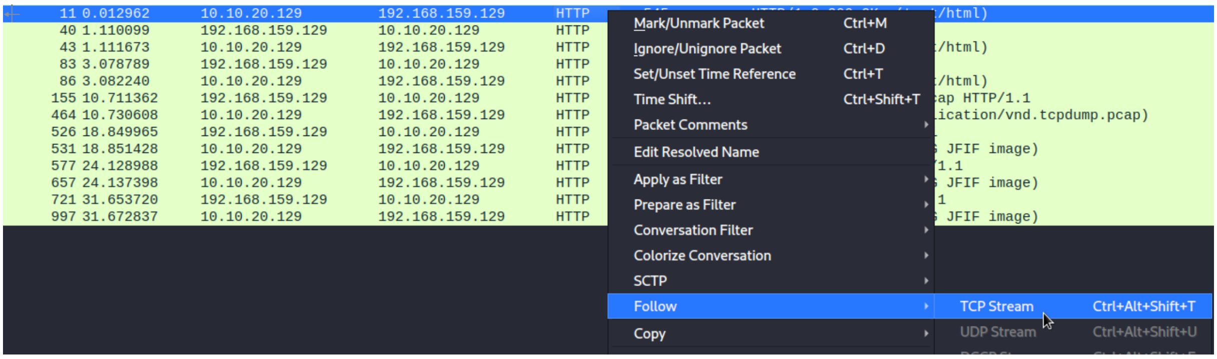


No.	Time	Source	Destination	Protocol	Length	Opcode	Info
7	0.010413	192.168.159.129	10.10.20.129	HTTP	516		GET / HTTP/1.1
11	0.012962	10.10.20.129	192.168.159.129	HTTP	545		HTTP/1.0 200 OK
40	1.110099	192.168.159.129	10.10.20.129	HTTP	516		GET / HTTP/1.1
43	1.111673	10.10.20.129	192.168.159.129	HTTP	545		HTTP/1.0 200 OK
83	3.078789	192.168.159.129	10.10.20.129	HTTP	516		GET / HTTP/1.1
86	3.082240	10.10.20.129	192.168.159.129	HTTP	545		HTTP/1.0 200 OK
155	10.711362	192.168.159.129	10.10.20.129	HTTP	376		GET /http_with_j
464	10.730608	10.10.20.129	192.168.159.129	HTTP	424		HTTP/1.0 200 OK
526	18.849965	192.168.159.129	10.10.20.129	HTTP	365		GET /htb.jpeg HT
531	18.851428	10.10.20.129	192.168.159.129	HTTP	726		HTTP/1.0 200 OK
577	24.128988	192.168.159.129	10.10.20.129	HTTP	368		GET /Rise-Up.jpg
657	24.137398	10.10.20.129	192.168.159.129	HTTP	1153		HTTP/1.0 200 OK
721	31.653720	192.168.159.129	10.10.20.129	HTTP	366		GET /water.jpg H
997	31.672837	10.10.20.129	192.168.159.129	HTTP	151		HTTP/1.0 200 OK

We can see GET requests and OK response from webserver. We are interested in extracting data contained in packets with OK Response.

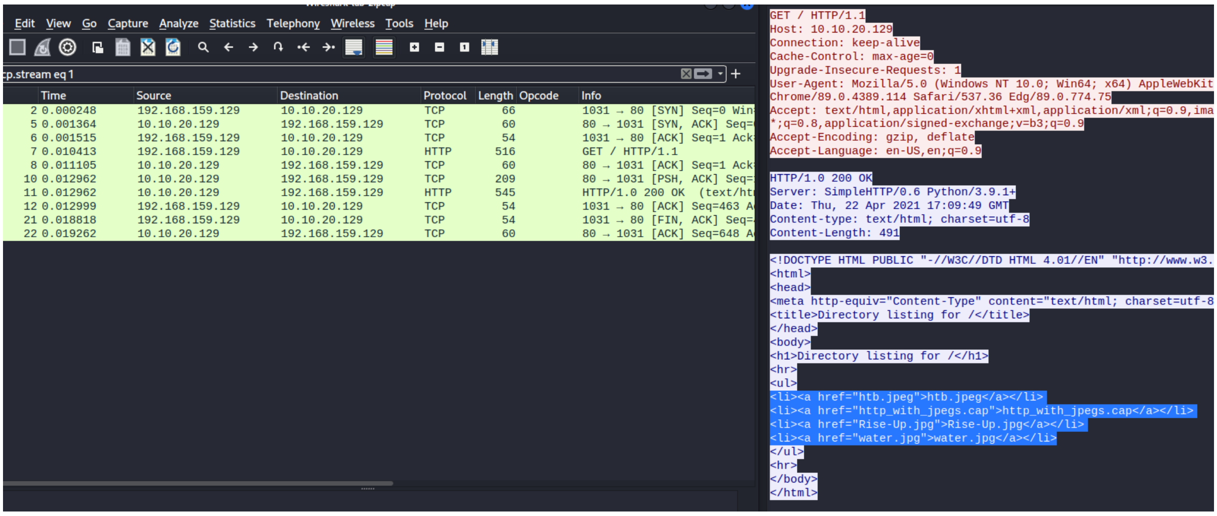
Task 3: Follow the stream and extract the items found

Let’s select one of OK response packets and follow their TCP stream



```
 11 0.012962   10.10.20.129     192.168.159.129  HTTP   /html)
 40 1.110099   192.168.159.129  10.10.20.129     HTTP
 43 1.111673   10.10.20.129     192.168.159.129  HTTP   /html)
 83 3.078789   192.168.159.129  10.10.20.129     HTTP
 86 3.082240   10.10.20.129     192.168.159.129  HTTP   /html)
155 10.711362  192.168.159.129  10.10.20.129     HTTP   ap HTTP/1.1
464 10.730608  10.10.20.129     192.168.159.129  HTTP   ication/vnd.tcpdump.pcap)
526 18.849965  192.168.159.129  10.10.20.129     HTTP
531 18.851428  10.10.20.129     192.168.159.129  HTTP    JFIF image)
577 24.128988  192.168.159.129  10.10.20.129     HTTP   /1.1
657 24.137398  10.10.20.129     192.168.159.129  HTTP    JFIF image)
721 31.653720  192.168.159.129  10.10.20.129     HTTP   1
997 31.672837  10.10.20.129     192.168.159.129  HTTP    JFIF image)
```

Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift… Ctrl+Shift+T
Packet Comments
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U



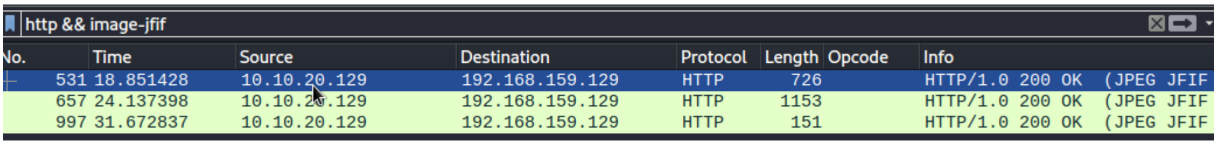
Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Filter: cp.stream eq 1

```
    Time       Source           Destination      Protocol Length Opcode Info
  2 0.000248   192.168.159.129  10.10.20.129     TCP        66          1031 → 80 [SYN] Seq=0 Win
  5 0.001364   10.10.20.129     192.168.159.129  TCP        60          80 → 1031 [SYN, ACK] Seq=
  6 0.001515   192.168.159.129  10.10.20.129     TCP        54          1031 → 80 [ACK] Seq=1 Ack
  7 0.010413   192.168.159.129  10.10.20.129     HTTP      516          GET / HTTP/1.1
  8 0.011105   10.10.20.129     192.168.159.129  TCP        60          80 → 1031 [ACK] Seq=1 Ack
 10 0.012962   10.10.20.129     192.168.159.129  TCP       209          80 → 1031 [PSH, ACK] Seq=
 11 0.012962   10.10.20.129     192.168.159.129  HTTP      545          HTTP/1.0 200 OK  (text/ht
 12 0.012999   192.168.159.129  10.10.20.129     TCP        54          1031 → 80 [ACK] Seq=463 A
 21 0.018818   192.168.159.129  10.10.20.129     TCP        54          1031 → 80 [FIN, ACK] Seq=
 22 0.019262   10.10.20.129     192.168.159.129  TCP        60          80 → 1031 [ACK] Seq=648 A
```

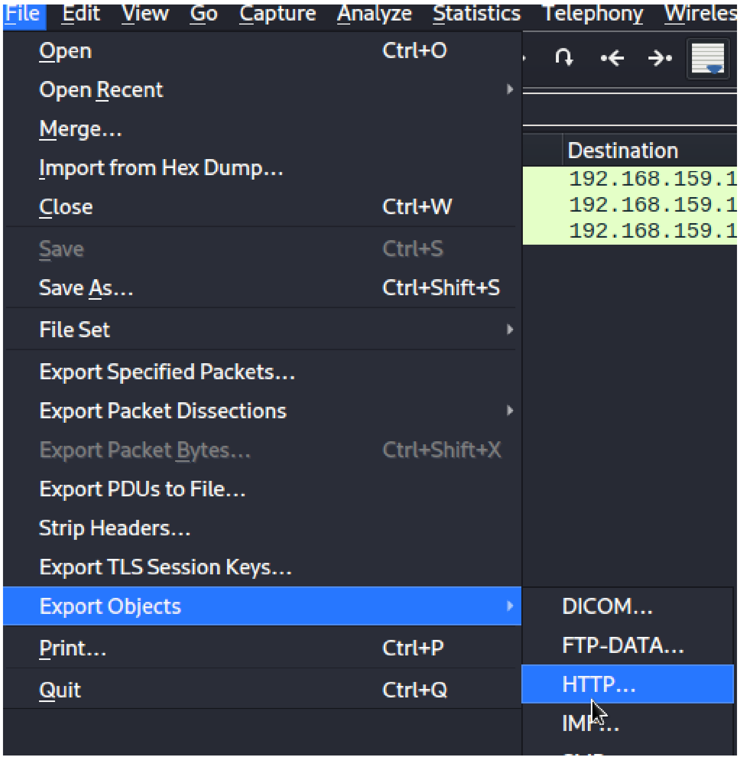
```
GET / HTTP/1.1
Host: 10.10.20.129
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
Chrome/89.0.4389.114 Safari/537.36 Edg/89.0.774.75
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima
*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.9.1+
Date: Thu, 22 Apr 2021 17:09:49 GMT
Content-type: text/html; charset=utf-8
Content-Length: 491

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="htb.jpeg">htb.jpeg</a></li>
<li><a href="http_with_jpegs.cap">http_with_jpegs.cap</a></li>
<li><a href="Rise-Up.jpg">Rise-Up.jpg</a></li>
<li><a href="water.jpg">water.jpg</a></li>
</ul>
<hr>
</body>
</html>
```

Not only we see TCP handshake and data transfer but there were in total 3 JPG files that must be used for further investigation. If we filter to **http && image-jfif** we can see that those 3 files were transferred. Let’s extract them!

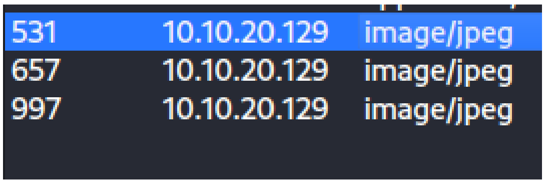


Filter: http && image-jfif

```
No.  Time       Source        Destination      Protocol Length Opcode Info
531 18.851428   10.10.20.129  192.168.159.129  HTTP       726         HTTP/1.0 200 OK  (JPEG JFIF
657 24.137398   10.10.20.129  192.168.159.129  HTTP      1153         HTTP/1.0 200 OK  (JPEG JFIF
997 31.672837   10.10.20.129  192.168.159.129  HTTP       151         HTTP/1.0 200 OK  (JPEG JFIF
```



Click File -> Export Objects -> HTTP. And extract these 3 files. This is all for this scenario.



```
531   10.10.20.129   image/jpeg
657   10.10.20.129   image/jpeg
997   10.10.20.129   image/jpeg
```

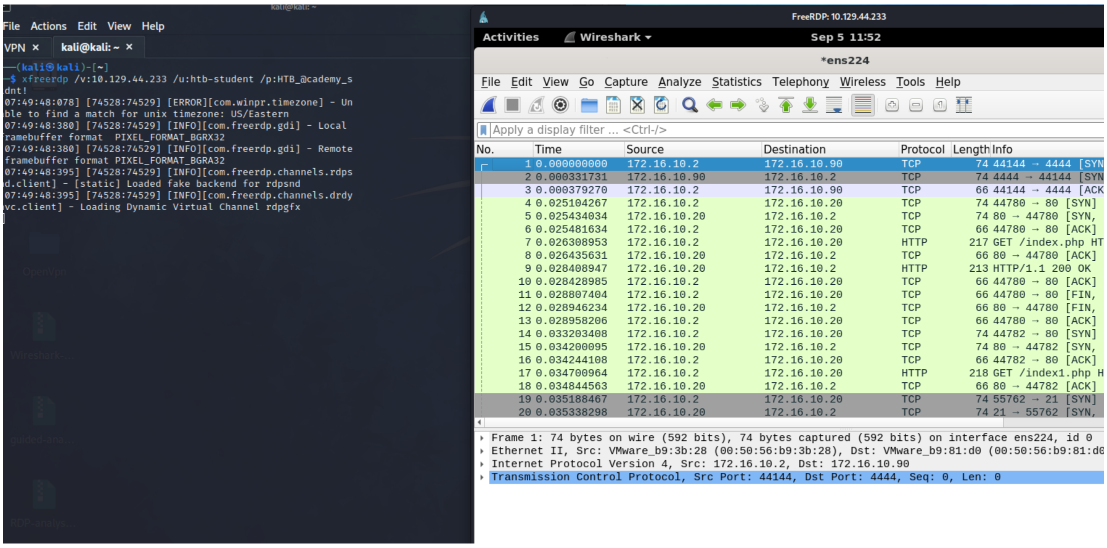
Scenario 2: Live Capture

We are requested that we now capture traffic to determine if anything else is going on from the user's host 10.129.43.4 (my own VM). We will need to start a capture, categorize and filter the data, and extract anything significant to the investigation.

Preparation

Let's connect to host 10.129.43.4 via xfreerdp. We will use this host to generate traffic to capture it in real time. We will even try to extract downloaded images. We are provided with credentials.

- IP == 10.129.43.4
- Username == htb-student
- Password == HTB_@cademy_stdnt!



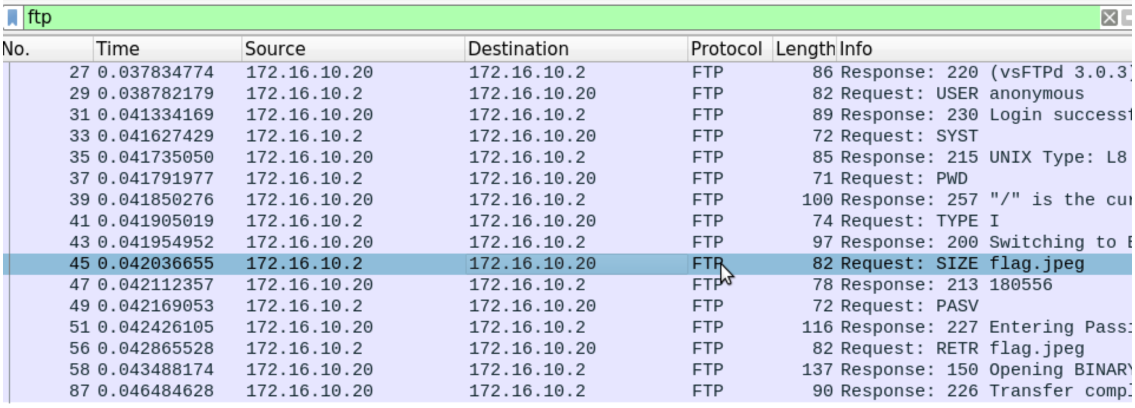
kali@kali: ~
File Actions Edit View Help
VPN ×
kali@kali: ~ ×
FreeRDP: 10.129.44.233
Activities
Wireshark ▾
Sep 5 11:52
*ens224
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
┌──(kali㉿kali)-[~]
└─$ xfreerdp /v:10.129.44.233 /u:htb-student /p:HTB_@cademy_s
tdnt!
[07:49:48:078] [74528:74529] [ERROR][com.winpr.timezone] - Un
able to find a match for unix timezone: US/Eastern
[07:49:48:380] [74528:74529] [INFO][com.freerdp.gdi] - Local
framebuffer format  PIXEL_FORMAT_BGRX32
[07:49:48:380] [74528:74529] [INFO][com.freerdp.gdi] - Remote
 framebuffer format PIXEL_FORMAT_BGRA32
[07:49:48:395] [74528:74529] [INFO][com.freerdp.channels.rdps
nd.client] - [static] Loaded fake backend for rdpsnd
[07:49:48:395] [74528:74529] [INFO][com.freerdp.channels.drdy
nvc.client] - Loading Dynamic Virtual Channel rdpgfx
```

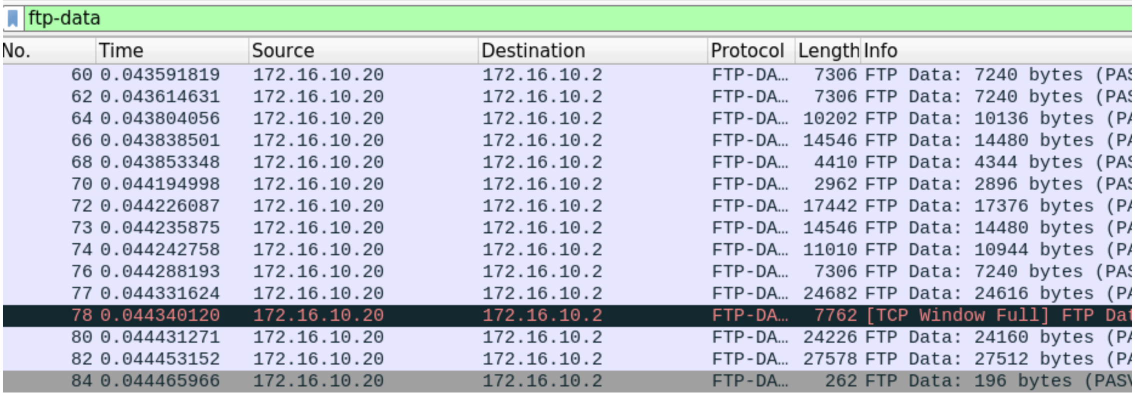
```
No.   Time         Source         Destination    Protocol Length Info
    1 0.000000000  172.16.10.2    172.16.10.90   TCP        74 44144 → 4444 [SYN
    2 0.000331731  172.16.10.90   172.16.10.2    TCP        74 4444 → 44144 [SYN
    3 0.000379270  172.16.10.2    172.16.10.90   TCP        66 44144 → 4444 [ACK
    4 0.025104267  172.16.10.2    172.16.10.20   TCP        74 44780 → 80 [SYN]
    5 0.025434034  172.16.10.20   172.16.10.2    TCP        74 80 → 44780 [SYN,
    6 0.025481634  172.16.10.2    172.16.10.20   TCP        66 44780 → 80 [ACK]
    7 0.026308953  172.16.10.2    172.16.10.20   HTTP      217 GET /index.php HT
    8 0.026435631  172.16.10.20   172.16.10.2    TCP        66 80 → 44780 [ACK]
    9 0.028408947  172.16.10.20   172.16.10.2    HTTP      213 HTTP/1.1 200 OK
   10 0.028428985  172.16.10.2    172.16.10.20   TCP        66 44780 → 80 [ACK]
   11 0.028807404  172.16.10.2    172.16.10.20   TCP        66 44780 → 80 [FIN,
   12 0.028946234  172.16.10.20   172.16.10.2    TCP        66 80 → 44780 [FIN,
   13 0.028958206  172.16.10.2    172.16.10.20   TCP        66 44780 → 80 [ACK]
   14 0.033203408  172.16.10.2    172.16.10.20   TCP        74 44782 → 80 [SYN]
   15 0.034200095  172.16.10.20   172.16.10.2    TCP        74 80 → 44782 [SYN,
   16 0.034244108  172.16.10.2    172.16.10.20   TCP        66 44782 → 80 [ACK]
   17 0.034700964  172.16.10.2    172.16.10.20   HTTP      218 GET /index1.php H
   18 0.034844563  172.16.10.20   172.16.10.2    TCP        66 80 → 44782 [ACK]
   19 0.035188467  172.16.10.2    172.16.10.20   TCP        74 55762 → 21 [SYN]
   20 0.035338298  172.16.10.20   172.16.10.2    TCP        74 21 → 55762 [SYN,

▸ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens224, id 0
▸ Ethernet II, Src: VMware_b9:3b:28 (00:50:56:b9:3b:28), Dst: VMware_b9:81:d0 (00:50:56:b9:81:d0
▸ Internet Protocol Version 4, Src: 172.16.10.2, Dst: 172.16.10.90
▸ Transmission Control Protocol, Src Port: 44144, Dst Port: 4444, Seq: 0, Len: 0
```

We connected and captured some interesting traffic on this host. Let's analyse it.

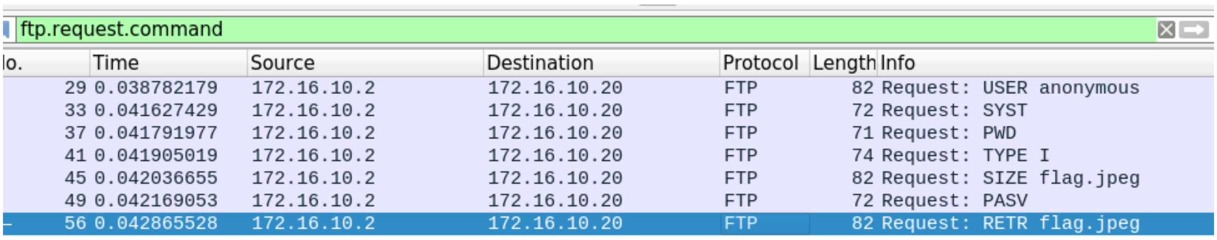
FTP Analysis: going deeper into captured packets we can see FTP communication. Let's dive in to them and try to find something interesting.



```
ftp
No.   Time         Source         Destination    Protocol Length Info
   27 0.037834774  172.16.10.20   172.16.10.2    FTP        86 Response: 220 (vsFTPd 3.0.3)
   29 0.038782179  172.16.10.2    172.16.10.20   FTP        82 Request: USER anonymous
   31 0.041334169  172.16.10.20   172.16.10.2    FTP        89 Response: 230 Login successf
   33 0.041627429  172.16.10.2    172.16.10.20   FTP        72 Request: SYST
   35 0.041735050  172.16.10.20   172.16.10.2    FTP        85 Response: 215 UNIX Type: L8
   37 0.041791977  172.16.10.2    172.16.10.20   FTP        71 Request: PWD
   39 0.041850276  172.16.10.20   172.16.10.2    FTP       100 Response: 257 "/" is the cur
   41 0.041905019  172.16.10.2    172.16.10.20   FTP        74 Request: TYPE I
   43 0.041954952  172.16.10.20   172.16.10.2    FTP        97 Response: 200 Switching to B
   45 0.042036655  172.16.10.2    172.16.10.20   FTP        82 Request: SIZE flag.jpeg
   47 0.042112357  172.16.10.20   172.16.10.2    FTP        78 Response: 213 180556
   49 0.042169053  172.16.10.2    172.16.10.20   FTP        72 Request: PASV
   51 0.042426105  172.16.10.20   172.16.10.2    FTP       116 Response: 227 Entering Passi
   56 0.042865528  172.16.10.2    172.16.10.20   FTP        82 Request: RETR flag.jpeg
   58 0.043488174  172.16.10.20   172.16.10.2    FTP       137 Response: 150 Opening BINARY
   87 0.046484628  172.16.10.20   172.16.10.2    FTP        90 Response: 226 Transfer compl
```



```
ftp-data
No.   Time         Source         Destination    Protocol Length Info
   60 0.043591819  172.16.10.20   172.16.10.2    FTP-DA…   7306 FTP Data: 7240 bytes (PAS
   62 0.043614631  172.16.10.20   172.16.10.2    FTP-DA…   7306 FTP Data: 7240 bytes (PAS
   64 0.043804056  172.16.10.20   172.16.10.2    FTP-DA…  10202 FTP Data: 10136 bytes (PA
   66 0.043838501  172.16.10.20   172.16.10.2    FTP-DA…  14546 FTP Data: 14480 bytes (PA
   68 0.043853348  172.16.10.20   172.16.10.2    FTP-DA…   4410 FTP Data: 4344 bytes (PAS
   70 0.044194998  172.16.10.20   172.16.10.2    FTP-DA…   2962 FTP Data: 2896 bytes (PAS
   72 0.044226087  172.16.10.20   172.16.10.2    FTP-DA…  17442 FTP Data: 17376 bytes (PA
   73 0.044235875  172.16.10.20   172.16.10.2    FTP-DA…  14546 FTP Data: 14480 bytes (PA
   74 0.044242758  172.16.10.20   172.16.10.2    FTP-DA…  11010 FTP Data: 10944 bytes (PA
   76 0.044288193  172.16.10.20   172.16.10.2    FTP-DA…   7306 FTP Data: 7240 bytes (PAS
   77 0.044331624  172.16.10.20   172.16.10.2    FTP-DA…  24682 FTP Data: 24616 bytes (PA
   78 0.044340120  172.16.10.20   172.16.10.2    FTP-DA…   7762 [TCP Window Full] FTP Dat
   80 0.044431271  172.16.10.20   172.16.10.2    FTP-DA…  24226 FTP Data: 24160 bytes (PA
   82 0.044453152  172.16.10.20   172.16.10.2    FTP-DA…  27578 FTP Data: 27512 bytes (PA
   84 0.044465966  172.16.10.20   172.16.10.2    FTP-DA…    262 FTP Data: 196 bytes (PASV
```

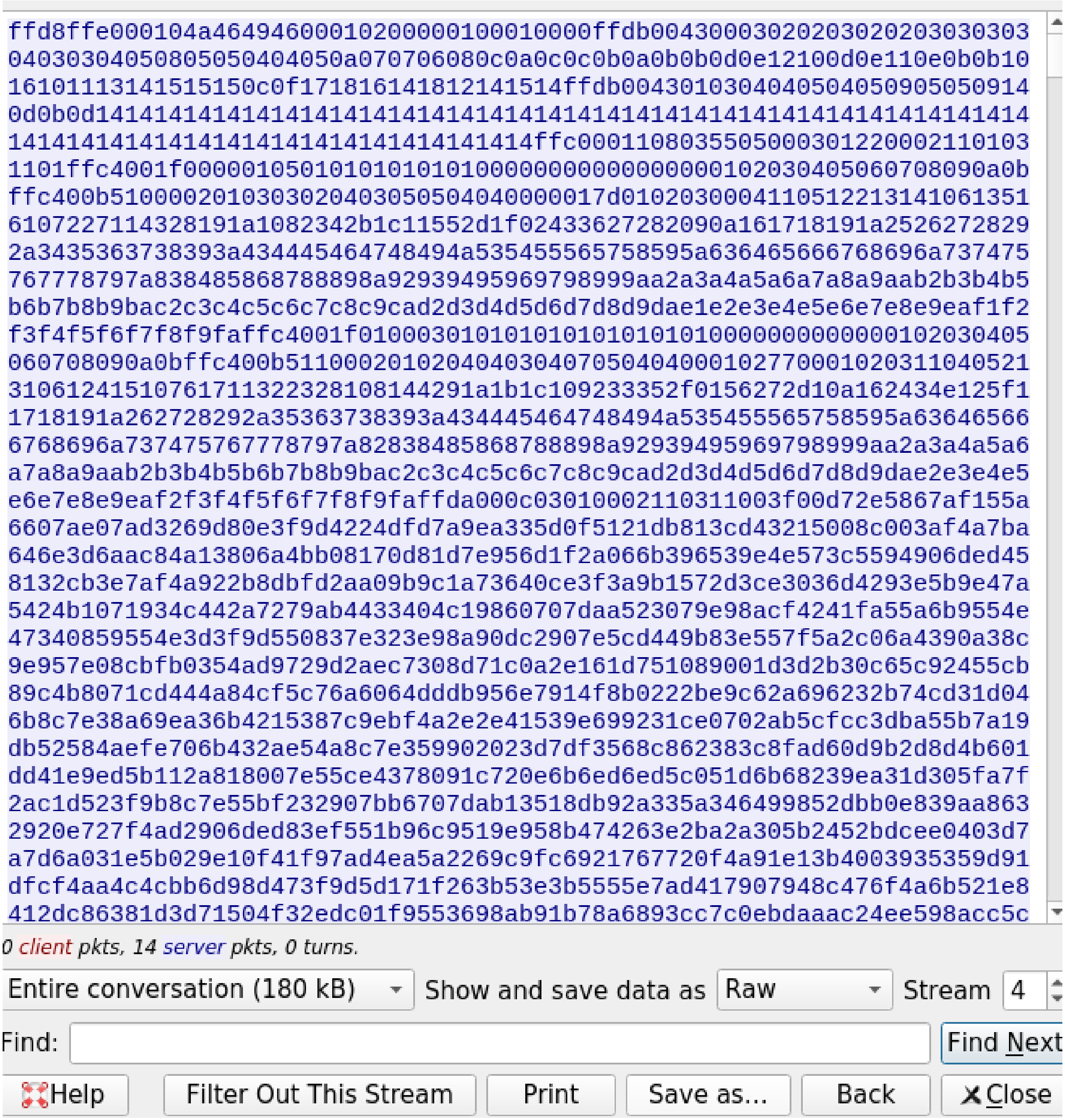


ftp.request.command

```
lo.     Time          Source        Destination   Protocol Length Info
     29 0.038782179   172.16.10.2   172.16.10.20  FTP          82 Request: USER anonymous
     33 0.041627429   172.16.10.2   172.16.10.20  FTP          72 Request: SYST
     37 0.041791977   172.16.10.2   172.16.10.20  FTP          71 Request: PWD
     41 0.041905019   172.16.10.2   172.16.10.20  FTP          74 Request: TYPE I
     45 0.042036655   172.16.10.2   172.16.10.20  FTP          82 Request: SIZE flag.jpeg
     49 0.042169053   172.16.10.2   172.16.10.20  FTP          72 Request: PASV
-    56 0.042865528   172.16.10.2   172.16.10.20  FTP          82 Request: RETR flag.jpeg
```

We found that anonymous user performed action of requesting some kind of images. Let's extract these jpeg files. We can easily determine that 172.16.10.20 is server and 172.16.10.2 is a client.

I followed TCP stream of ftp-data packet correlated with it.



```
ffd8ffe000104a46494600010200000100010000ffdb004300030202030202030303
04030304050805050404050a070706080c0a0c0c0b0a0b0b0d0e12100d0e110e0b0b10
16101113141515150c0f171816141812141514ffdb00430103040405040509050509 14
```

```
ffd8ffe000104a46494600010200000100010000ffdb004300030202030202030303
04030304050805050404050a070706080c0a0c0c0b0a0b0b0d0e12100d0e110e0b0b10
161011131415151 50c0f171816141812141514ffdb0043010304040504050905050914
```

0 client pkts, 14 server pkts, 0 turns.

Entire conversation (180 kB)

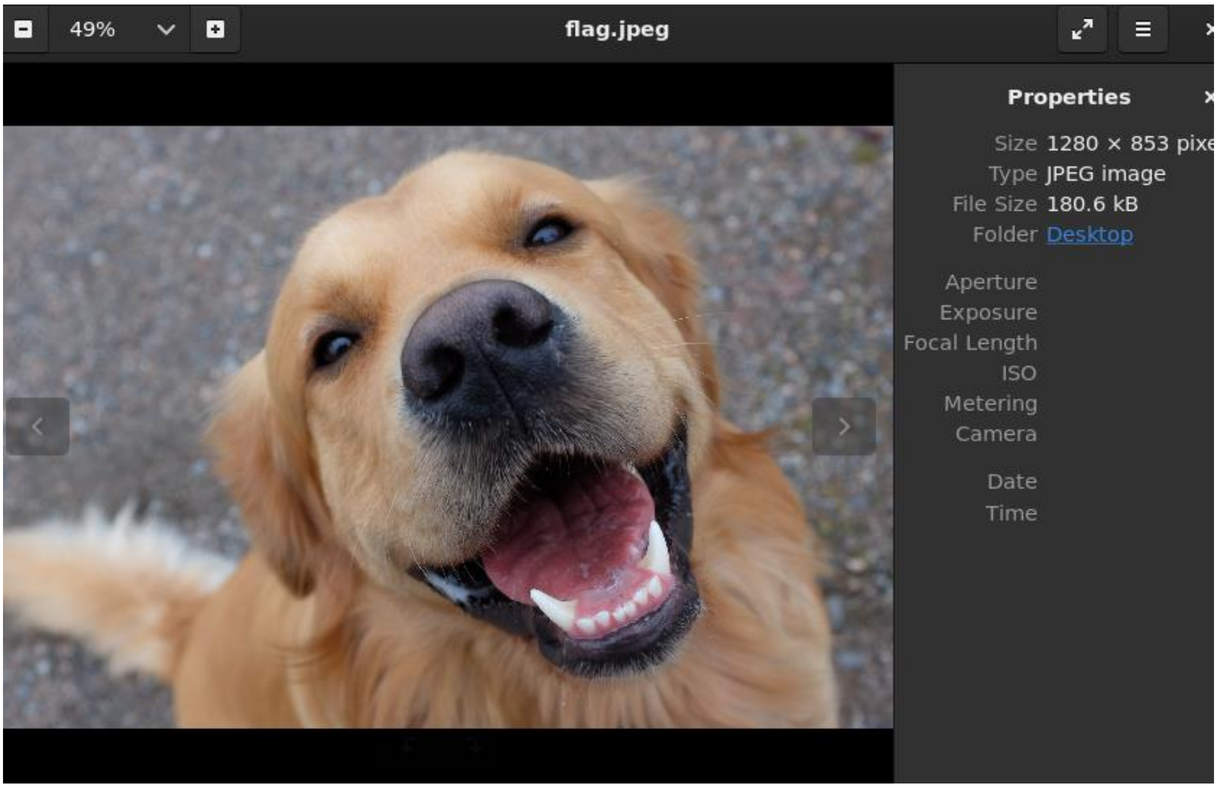
Show and save data as Raw

Stream 4

Find:

Now, let's change "Show and save data as" to "Raw" and save content as original file name. Now let's open this file to check what it was

Success! We extracted raw data from packet, reassembled and extracted cute dog image from live capture.



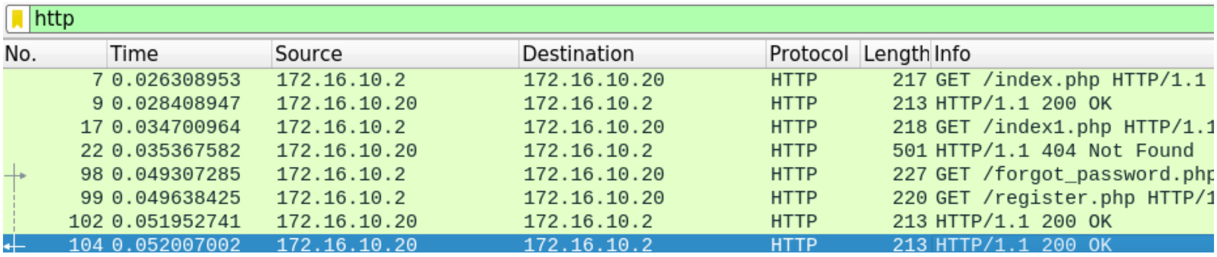
Size 1280 × 853 pixe
Type JPEG image
File Size 180.6 kB
Folder Desktop

Aperture
Exposure
Focal Length
ISO
Metering
Camera

Date
Time

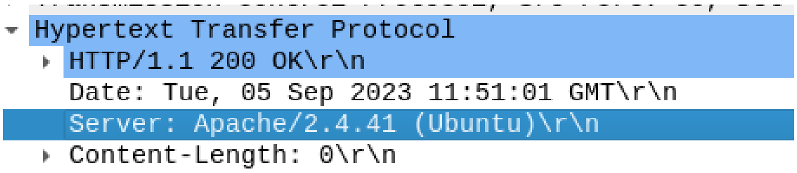
HTTP Analysis

Our job is not over. We saw some HTTP packets as well. Let’s explore.



```
http
No.    Time         Source         Destination    Protocol Length Info
     7 0.026308953  172.16.10.2    172.16.10.20   HTTP     217 GET /index.php HTTP/1.1
     9 0.028408947  172.16.10.20   172.16.10.2    HTTP     213 HTTP/1.1 200 OK
    17 0.034700964  172.16.10.2    172.16.10.20   HTTP     218 GET /index1.php HTTP/1.1
    22 0.035367582  172.16.10.20   172.16.10.2    HTTP     501 HTTP/1.1 404 Not Found
    98 0.049307285  172.16.10.2    172.16.10.20   HTTP     227 GET /forgot_password.php
    99 0.049638425  172.16.10.2    172.16.10.20   HTTP     220 GET /register.php HTTP/1
   102 0.051952741  172.16.10.20   172.16.10.2    HTTP     213 HTTP/1.1 200 OK
   104 0.052007002  172.16.10.20   172.16.10.2    HTTP     213 HTTP/1.1 200 OK
```

We can determine that 172.16.10.20 is webserver ran by Apache server



```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Tue, 05 Sep 2023 11:51:01 GMT\r\n
    Server: Apache/2.4.41 (Ubuntu)\r\n
  ▸ Content-Length: 0\r\n
```

Most frequent method request was GET request. I found nothing interesting. User didn’t requested any files to extract (lab specified it as file.jpg).

This concludes this lab exercises. Thanks for reading and I hope you found the information here useful.

Source: https://academy.hackthebox.com/module/81/section/789