



Live Network Traffic Analysis

rootsh3ll Bank is a multinational bank that serves over 100 million customers and has over 13,000 branches worldwide.

One of your colleagues has installed a Kali-based dropbox on rootsh3ll Bank's wired network. The dropbox has an embedded wireless NIC capable of monitor mode and packet injection.

With remote access to the dropbox, your job is to silently sniff on the wired network and look for any interesting data. Make sure not to transmit any packet over the network, or you might get banned from rootsh3ll Bank's wired network.

For wireless, to prevent detection, you'd have to decrypt the live WPA2-PSK data in monitor mode, without actually connecting to the target access point.

Objective:

1. Analyze live wireless traffic
2. Decrypt WiFi traffic without authenticating with the router.
3. Grok user behaviour from packet data analysis.
4. Perform sniffing over wired LAN
5. Extract assets from the captured data stream

TABLE OF CONTENTS

0. OBJECTIVE
1. FLAG 1 - IDENTIFY TARGET USER DESIRED INFORMATION
2. FLAG 2 - IDENTIFY WHAT REFERED BOB TO THE INFORMATION
3. FLAG 3 - WHAT GOOGLE QUERY WAS MADE FOR REACHING THE RESULT
4. FLAG 4 - IDENTIFY TARGET DOMAIN NAME USED FOR FILE UPLOAD
5. FLAG 5 - FIND THE TARGET DIRECTORY USED FOR FILE UPLOAD
6. FLAG 6 - EXTRACT THE UPLOADED FILE FROM SAVED PCAP FILE

Helpful tips

A wireless access point named: **Coherer** is active in your vicinity. Use Wireshark to decrypt the live wireless traffic from this access point with the password: **Induction**

Use Wireshark's filter `wlan.ssid == "Coherer"` to filter the target SSID's traffic and save it to a pcap file for further processing.

Flag 1

Bob (192.168.0.50) tried to look for a fundraiser campaign over WiFi. Which year's fundraiser campaign was that?

Target is surfing over WiFi. Let's put our Wireless card into monitor mode, enable Decryption Keys for target's SSID with the key:SSID combination provided in the Mission Statement.

But first, see if we have any wireless interface available.

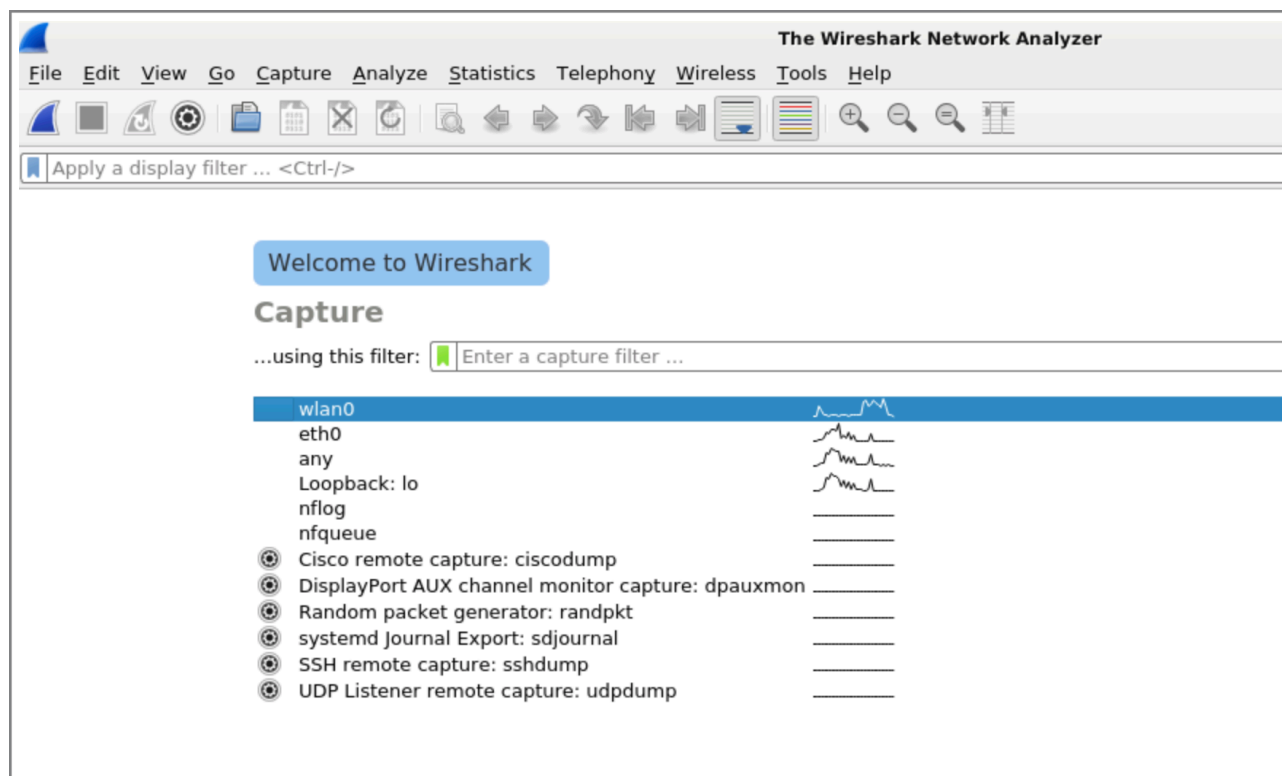
```
ifconfig wlan0
```

```
wlan0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 02:00:00:00:00:00 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Put the wlan0 card into monitor mode and set the interface up for Wireshark to detect and start sniffing.

```
iwconfig wlan0 mode monitor
ifconfig wlan0 up
```

Open Wireshark and choose wlan0 as target Interface.



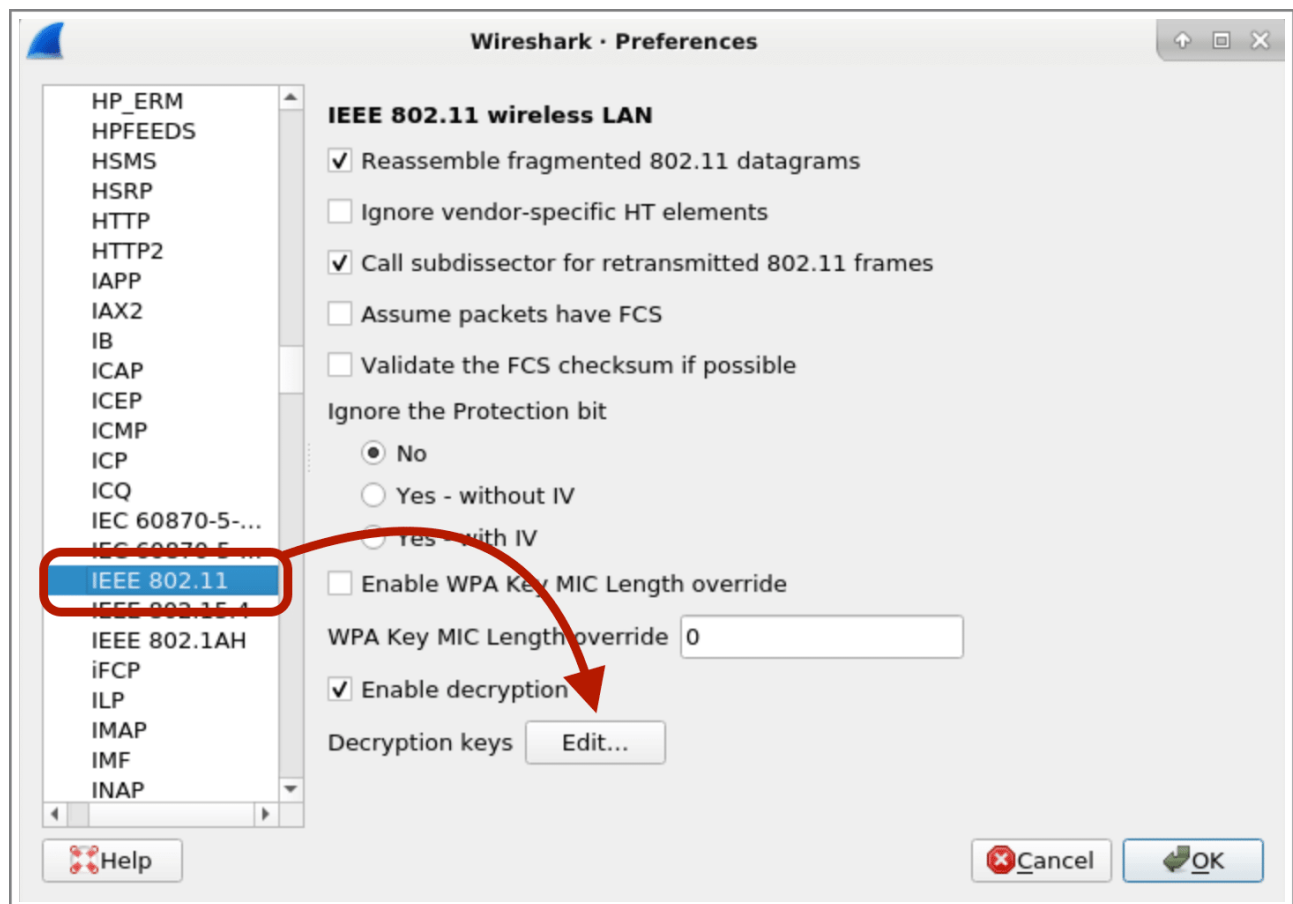
Use Wireshark display filter to filter data from our target MAC address.

```
wlan.addr == 00:0C:41:82:B2:55
```

Now, stop capture and go to **Edit > Preferences**

On the left panel, double click and expand "Protocols" and start typing IEEE to jump to "IEEE 802.11"

Click on "Edit" button placed adjacent to "Decryption Keys".



Click the '+' button on lower left of the window and choose wpa-pwd from the dropdown menu.

Under the Key section Enter **Key:SSID** pair like:

Induction:Coherer

Note that Key and SSID are case-sensitive and very important part of WPA2 hashing algorithm.

Hit OK, OK and Start capturing again. You shall now see decrypted data from Coherer inn Wireshark window.

Filter the HTTP traffic from the target traffic

```
wlan.addr == 00:0C:41:82:B2:55 && http
```

Destination	Protocol	Length	Info
66.230.200.100	HTTP	697	GET /wiki/Landshark HTTP/1.1
192.168.0.50	HTTP	264	[TCP Previous segment not captured] Continuation
66.230.200.228	HTTP	628	GET /fundraising/2006/meter.png HTTP/1.1
192.168.0.50	HTTP	508	[TCP Previous segment not captured] Continuation
209.188.21.206	HTTP	705	GET /75/75d jaws2.phtml HTTP/1.1
192.168.0.50	HTTP	649	[TCP Previous segment not captured] Continuation
209.188.21.206	HTTP	581	GET /style.css HTTP/1.1
192.168.0.50	HTTP	867	HTTP/1.1 200 OK (text/css)

The packet header is self explanatory. Fundraiser year is 2006.

Flag 2

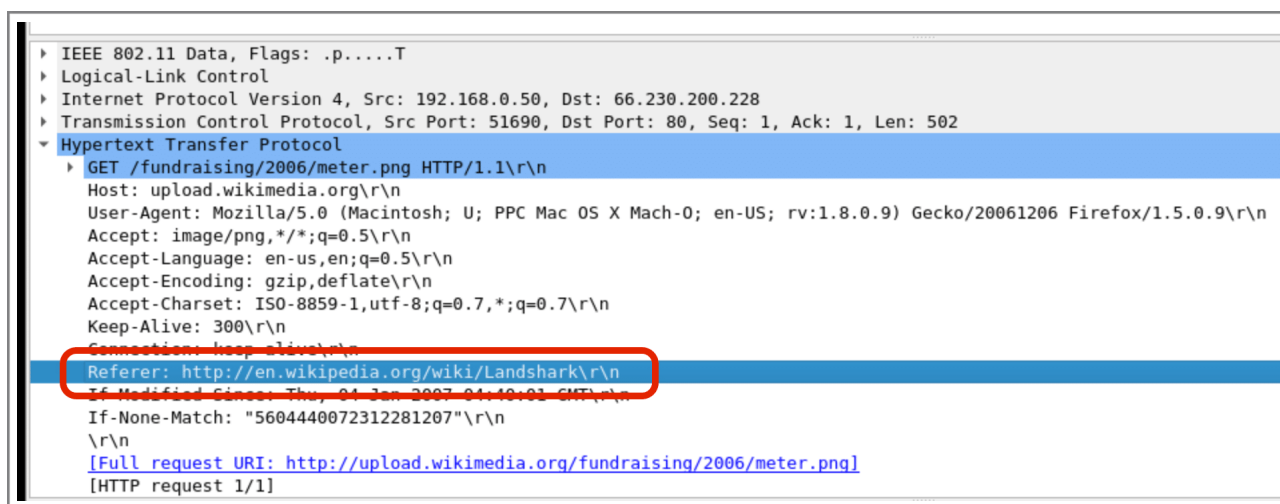
Which parent URL directed Bob to the final image?

Every web request is send with a certain number of headers. Most of the time you'd see a Referer header which is added when a redirect is applied on a link.

Like When searching on Google, when you click on your desired post, Google is added as the Referer to that post's URL.

To find the Referer in the Wireshark we need to look for HTTP GET request's header.

Select and Expand "Hypertext Transfer Protocol" in the bottom section of Wireshark for the same packet as previously selected.



You can copy the text by right clicking on it > **Copy** > **...as printable text**

Flag 3

What Google query did Bob make to reach that parent URL?

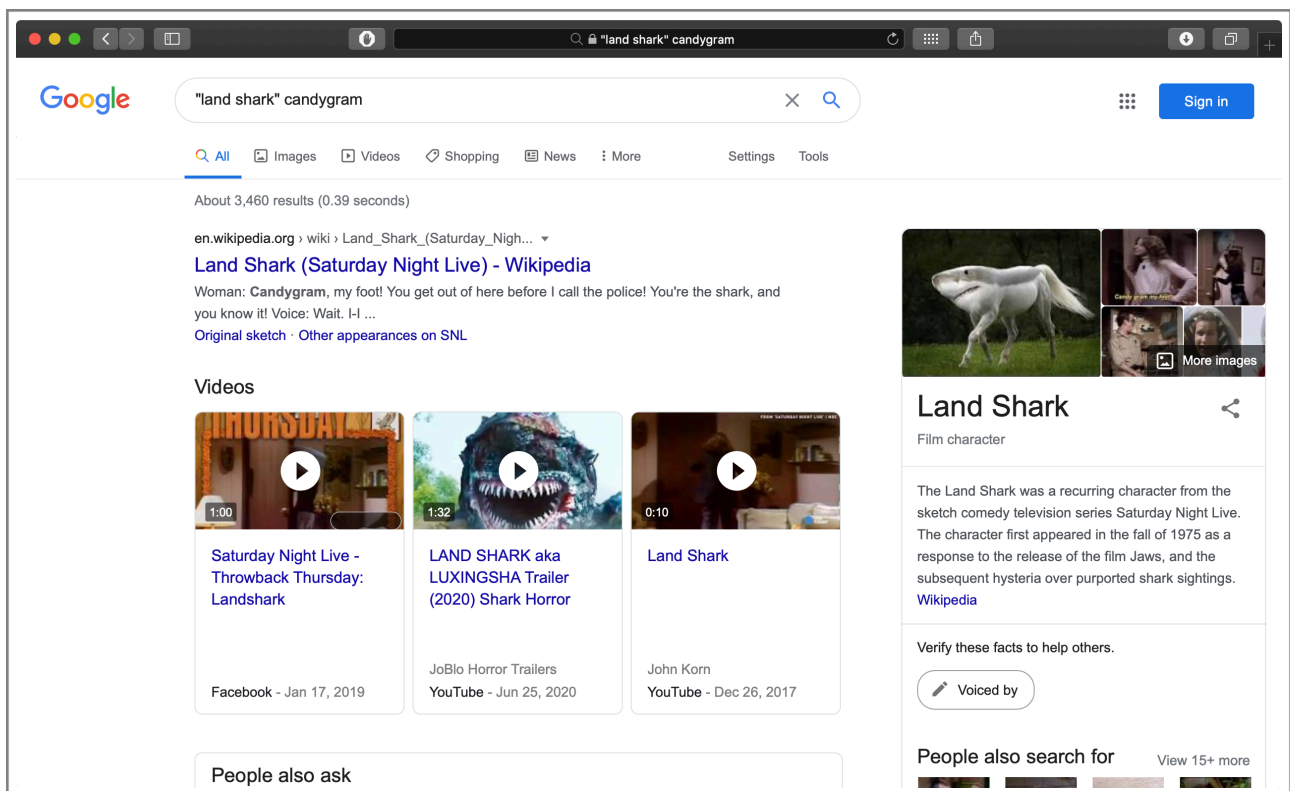
Since we know the user was redirected from **/wiki/Landshark** URL. We'd select the previous packet that shows Info as **GET /wiki/Landshark** and look for its parent Referer URL.

That'll reveal us which query/url helped Bob to reach the fundraiser campaign. Something like this:

```
Referer: http://www.google.com/search?
q=%22land+shark%22+candygram&start=0&ie=utf-8&oe=utf-8&client=firefox-
a&rls=org.mozilla:en-US:official
```

The query isn't clear enough. Let's copy it onto our clipboard and run in our local browser to see the resulting google query. Make sure to remove "Referer: " from the copied text.

The Google query show this result:



Flag 4

What domain name was used by client (10.1.3.219) to transfer a file over HTTP?

Upon running *ifconfig*, we notice that 10.1.3.x range is associated with ethernet interface.

```
ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 31000
    inet 10.1.3.37 netmask 255.255.255.0 broadcast 10.1.3.255
    ether 02:42:0a:01:03:25 txqueuelen 0 (Ethernet)
    RX packets 87321 bytes 7087205 (6.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 75242 bytes 117631133 (112.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Put eth0 in promiscuous mode and listen through Wireshark for incoming traffic.

```
ifconfig eth0 promisc
```

Run *ifconfig eth0* and see if **PROMISC** flag is set on eth0.

```
ifconfig eth0 | grep -i promisc
```

```
eth0: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 31000
```

Now, go to Wireshark > **Capture > Options** and select eth0 for sniffing

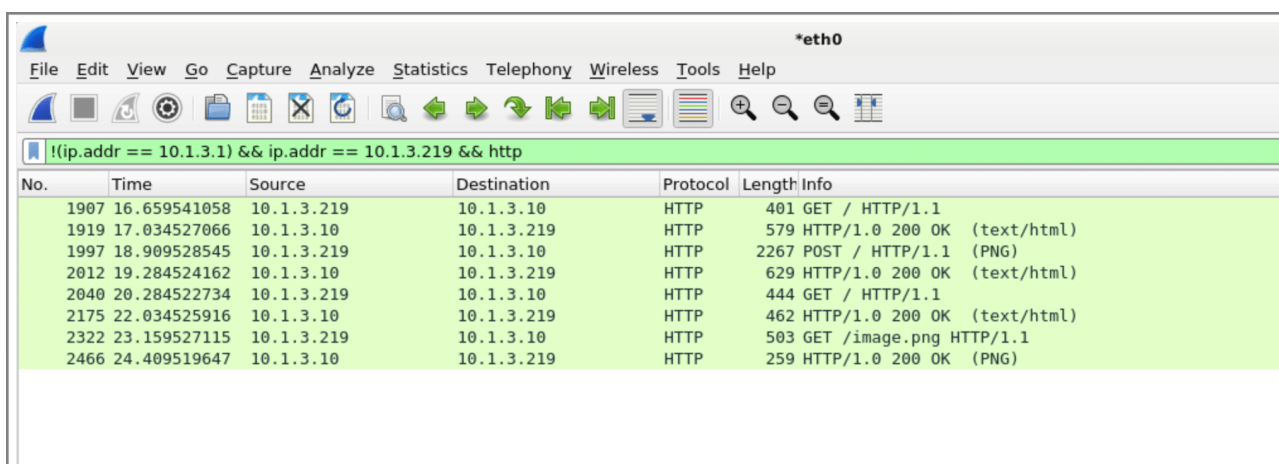
Replace (if any) existing filter with **!(ip.addr == 10.1.3.1)** to ignore traffic from the router itself.

And since our target IP is 10.1.3.219 and target protocol is http, add a condition to the existing filter. So that it means,

1. Do not show traffic from IP.address == 10.1.3.1, AND
2. Show traffic if IP.address == 10.1.3.219, AND
3. Protocol is http

It's very simple. Just expand it to **!(ip.addr == 10.1.3.1) && ip.addr == 10.1.3.219 and http**

Hit Enter and your output would look something like this;

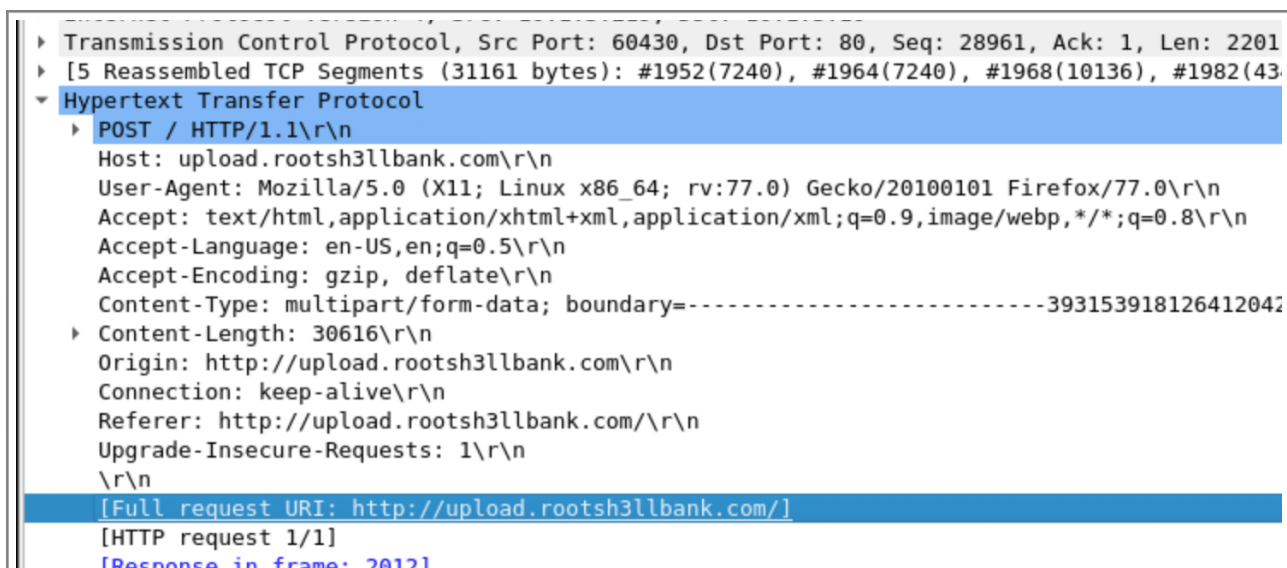


No.	Time	Source	Destination	Protocol	Length	Info
1907	16.659541058	10.1.3.219	10.1.3.10	HTTP	401	GET / HTTP/1.1
1919	17.034527066	10.1.3.10	10.1.3.219	HTTP	579	HTTP/1.0 200 OK (text/html)
1997	18.909528545	10.1.3.219	10.1.3.10	HTTP	2267	POST / HTTP/1.1 (PNG)
2012	19.284524162	10.1.3.10	10.1.3.219	HTTP	629	HTTP/1.0 200 OK (text/html)
2040	20.284522734	10.1.3.219	10.1.3.10	HTTP	444	GET / HTTP/1.1
2175	22.034525916	10.1.3.10	10.1.3.219	HTTP	462	HTTP/1.0 200 OK (text/html)
2322	23.159527115	10.1.3.219	10.1.3.10	HTTP	503	GET /image.png HTTP/1.1
2466	24.409519647	10.1.3.10	10.1.3.219	HTTP	259	HTTP/1.0 200 OK (PNG)

For uploading a file or any kind of data, certain HTTP Verbs are used. POST and PUT are the most widely used HTTP Verbs.

In the output, we notice a POST request for PNG filetype. Let's explore that.

In the Hypertext Transfer Protocol section we see the Full request URI to be <http://upload.rootsh3llbank.com>



Transmission Control Protocol, Src Port: 60430, Dst Port: 80, Seq: 28961, Ack: 1, Len: 2201
[5 Reassembled TCP Segments (31161 bytes): #1952(7240), #1964(7240), #1968(10136), #1982(43...
Hypertext Transfer Protocol
POST / HTTP/1.1\r\n
Host: upload.rootsh3llbank.com\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Content-Type: multipart/form-data; boundary=-----393153918126412042
Content-Length: 30616\r\n
Origin: http://upload.rootsh3llbank.com\r\n
Connection: keep-alive\r\n
Referer: http://upload.rootsh3llbank.com/\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
[Full request URI: http://upload.rootsh3llbank.com/]
[HTTP request 1/1]
[Response in frame 2012]

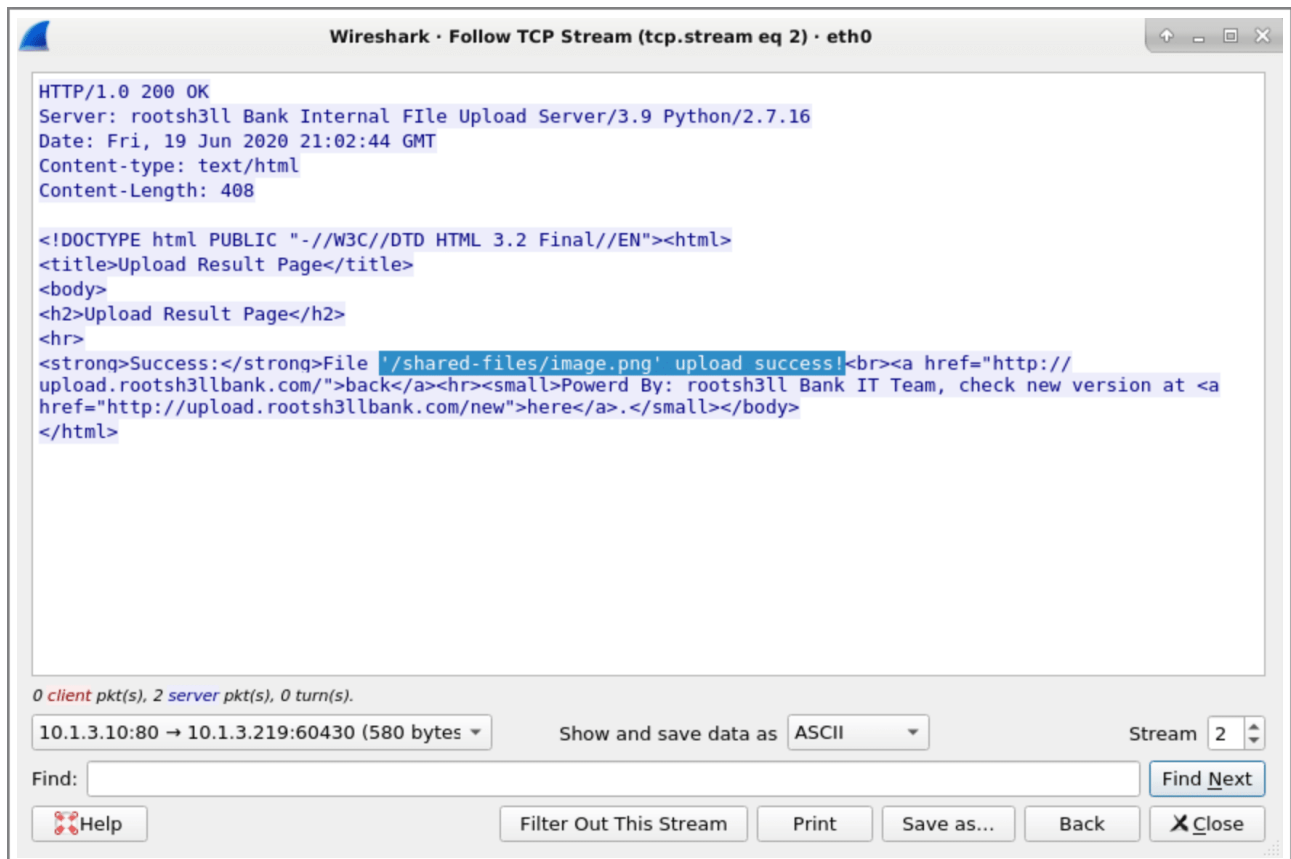
Flag 5

Identify in which directory image.png was uploaded

Server's sometimes leaks the data of the target directory. Let's follow the target traffic's TCP stream and look for the plaintext response from the server on successful response.

Right click on the target packet > follow > TCP Stream

In the popup widow, on the bottom left, chose the option with lowest size. To filter response from the whole stream which includes PNG file data,



The HTTP response from the server does show the target directory for the successful file upload. Enter the value in the Verify Flag section to mark the flag as complete.

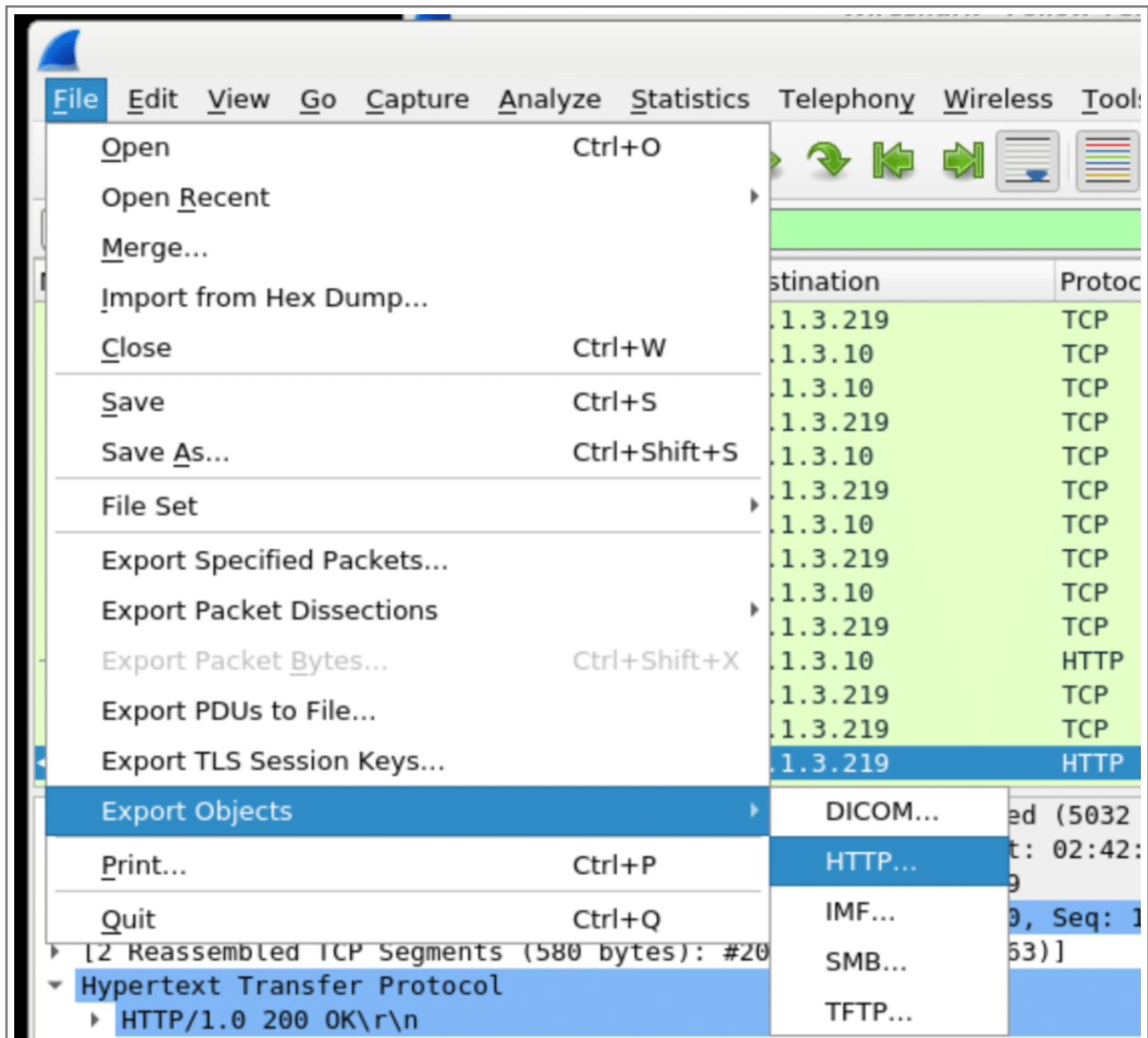
Flag 6

What do you see in the extracted `image.png` file?

Wireshark is very helpful when it comes to extracting assets from the pcap file.

It's simple.

Go to **File > Export Objects > HTTP**



Then select the target PNG file, click Save. Close Wireshark and open the PNG file and Enter the file based on what you saw in the downloaded PNG file.

Happy hacking! :)