Faculty: Faculty of Computing & Informatics

Subject Code: PSP0201

Subject Name: MINI IT PROJECT

Section: TL7L

Assignment Name: Week 2 Tutorial Progress

Trimester: Trimester 3 2021/2022(T2120)

Lecturer: Mr Wong Ya Ping

STUDENT NAME	ID NUMBER
AZRYL SHAMIN BIN AZRIZAL	1211103145
TAN EASON	1211101844
JERRELL SU MING JIE	1211103690

Day 1: Web Exploitation – A Christmas Crisis

Tools used: Kali Linux, Firefox

Solution/walkthrough:

Question 1

Inspecting the website. We know that the website title on header tag is "**Christmas Console**"

```
<title>Christmas Console</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Question 2

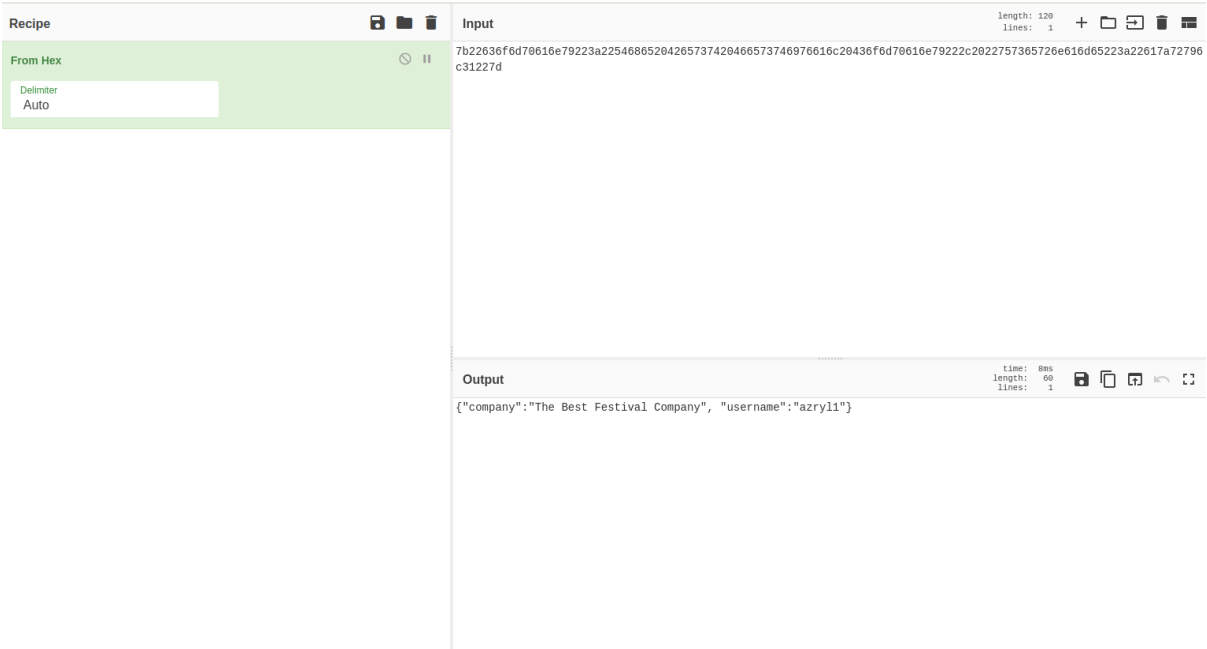
Value	
7b22636f6d70616e79223a225468652042657374204665737469766c20436f6d70616e79222c2022757365726e616d65223a2274696d6f746879227d	1

Question 3

We then know that the cookie gives a **hexadecimal** code and went to CyberChef to decode the code

Question 4

After decoding from CyberChef, I got a data in JSON file with two fields



Question 5

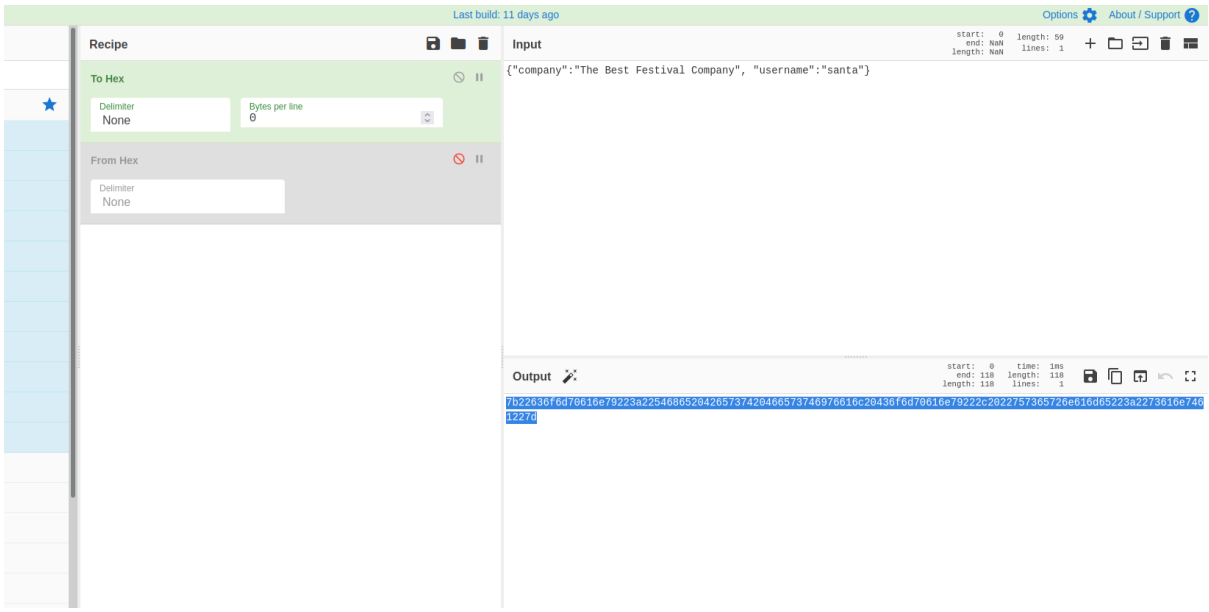
The value inside company seen is “**The best Festival company**”

Question 6

Username field also found besides the “company” field

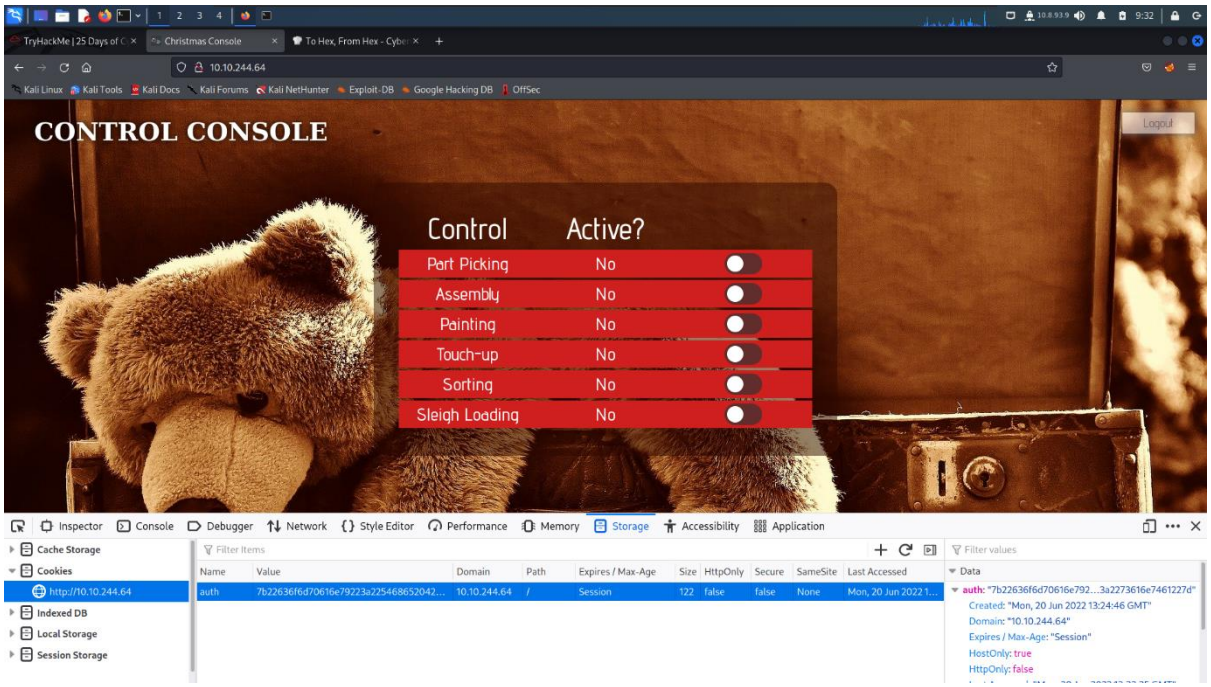
Question 7

To have access to Santa’s login we then replace the username with Santa’s and convert it back to hex

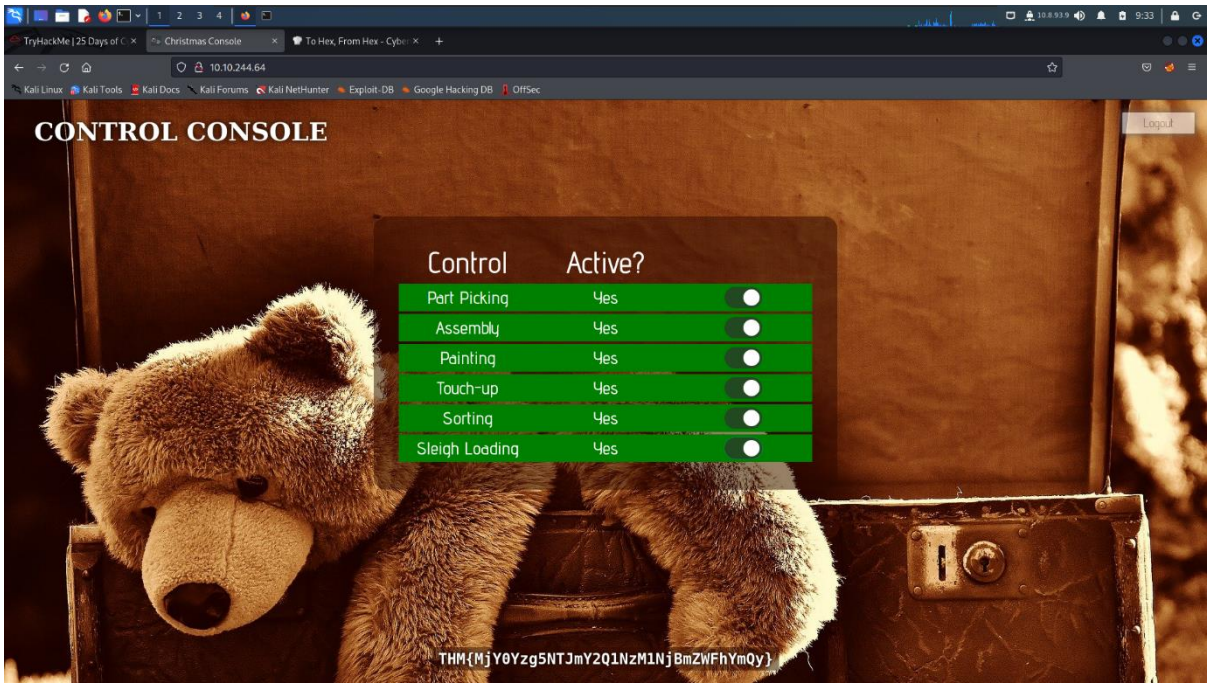


Question 8

After replacing the cookie value to Santa's, I am now in control of the control console.



The flag shows up once we toggled every controls to on.



Thought Process/Methodology:

When accessed the target IP, we were shown a control console page which used an inherently stateless protocol. We know that it must not have any separate server and try on registering and logging in the console. We were then chosen to view the site cookie from the storage tab using browser's inspect tool. Knowing that it was in hexadecimal form, we then headed to CyberChef in order to decode it. We saw the data is in JSON format and got "username" so we replace ours with "Santa". We then encode it back to hexadecimal form and copy pasting it onto the cookie value. This gave us access to Santa's account and we are able to turn on all the controls. The flag appeared marked that we are done for the day.

Day 2: Web Exploitation – The Elf Strikes Back!

Tools used: Kali Linux

Solution/Walkthrough:

Question 1

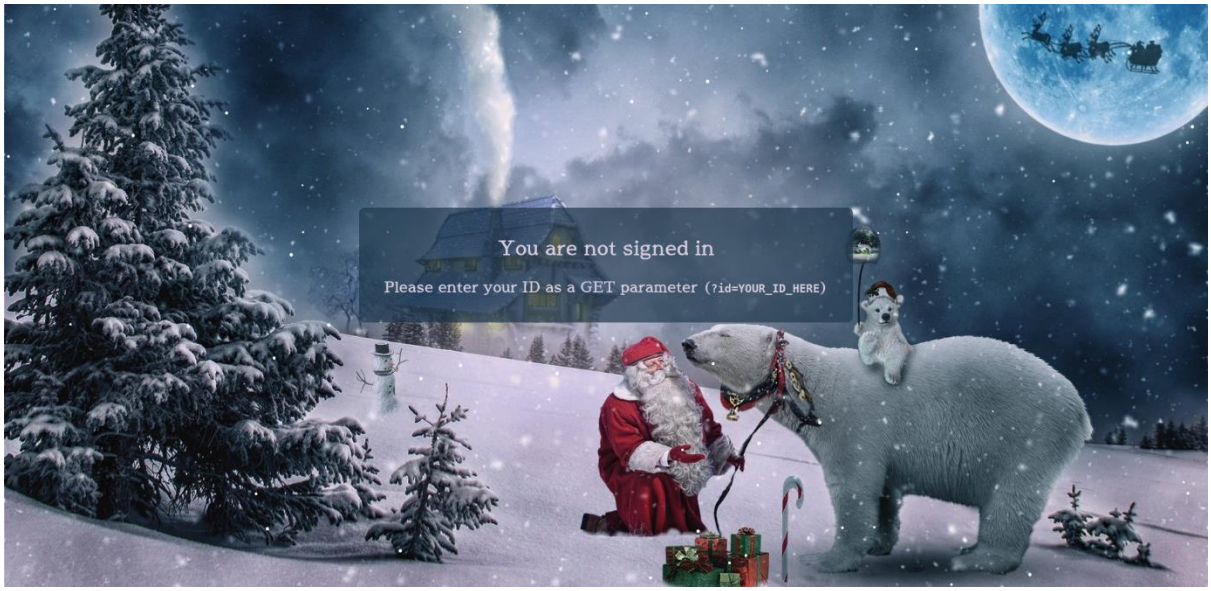
Before we are accessing the system, we are given a sticky note at the bottom of dossier with this message:

For Elf McEager:
You have been assigned an ID number for your audit of the system: `ODIzODI5MTNiYmYw` . Use this to gain access to the upload section of the site.
Good luck!

*You note down the ID number and **navigate to the displayed IP address (MACHINE_IP) in your browser.***

By this message, we know that we need to use the string of text "ODIzODI5MTNiYmYw" to access the system

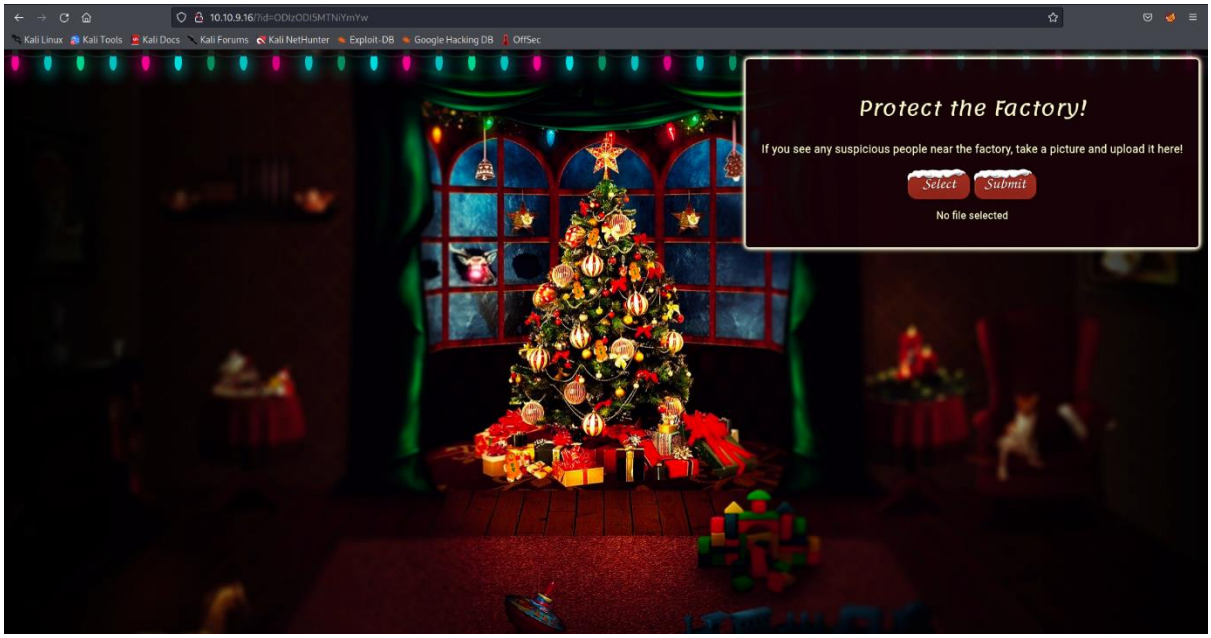
Then once in the machine, they asked us to use ID as get parameter



So we put in the ID in URL as GET Parameter.

Question 2

Once we enter the machine, we can see a box on the top right telling us to drop a file contains details about suspicious person we saw.



To test this, we try uploading all kind of files to see what it accepts and what it doesn't

After countless of try, we know that it accepts only image files.

Question 3

By guessing the common URL Path, we found out that the files were kept in /uploads/ (reference on question 5)

Question 4

In order to capture the flags, we are using netcat so we are familiarizing ourselves with the netcat commands.

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ netcat -h
[v1.10-47]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous
!!]
        -e filename             program to exec after connect [dangerou
s!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports sc
anned
        -k                      set keepalive option on socket
        -l                      listen mode, for inbound connects
        -n                      numeric-only IP addresses, no DNS
        -o file                 hex dump of traffic
        -p port                 local port number
        -r                      randomize local and remote ports
        -q secs                 quit after EOF on stdin and delay of se
cs
        -s addr                 local source address
        -T tos                  set Type Of Service
        -t                      answer TELNET negotiation
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net read
s
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').

┌──(kali㉿kali)-[~]
```

From the netcat parameter options, we know that:

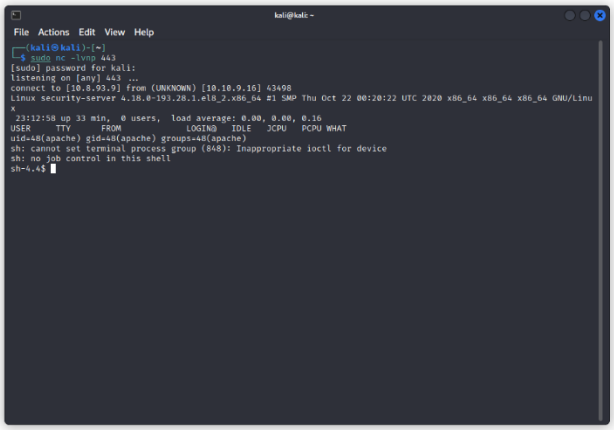
Parameter	Explanation
-v	Have nc give more verbose output.
-l	Used to specify that nc should listen for an incoming connection rather than initiate a connection to a remote host.
-n	Do not do any DNS or service lookups on any specified addresses, hostnames or ports.
-p	Specifies the source port nc should use, subject to privilege restrictions and availability.

Question 5

To capture the flag, we are going to bypass the upload by uploading a double-barrelled extension.

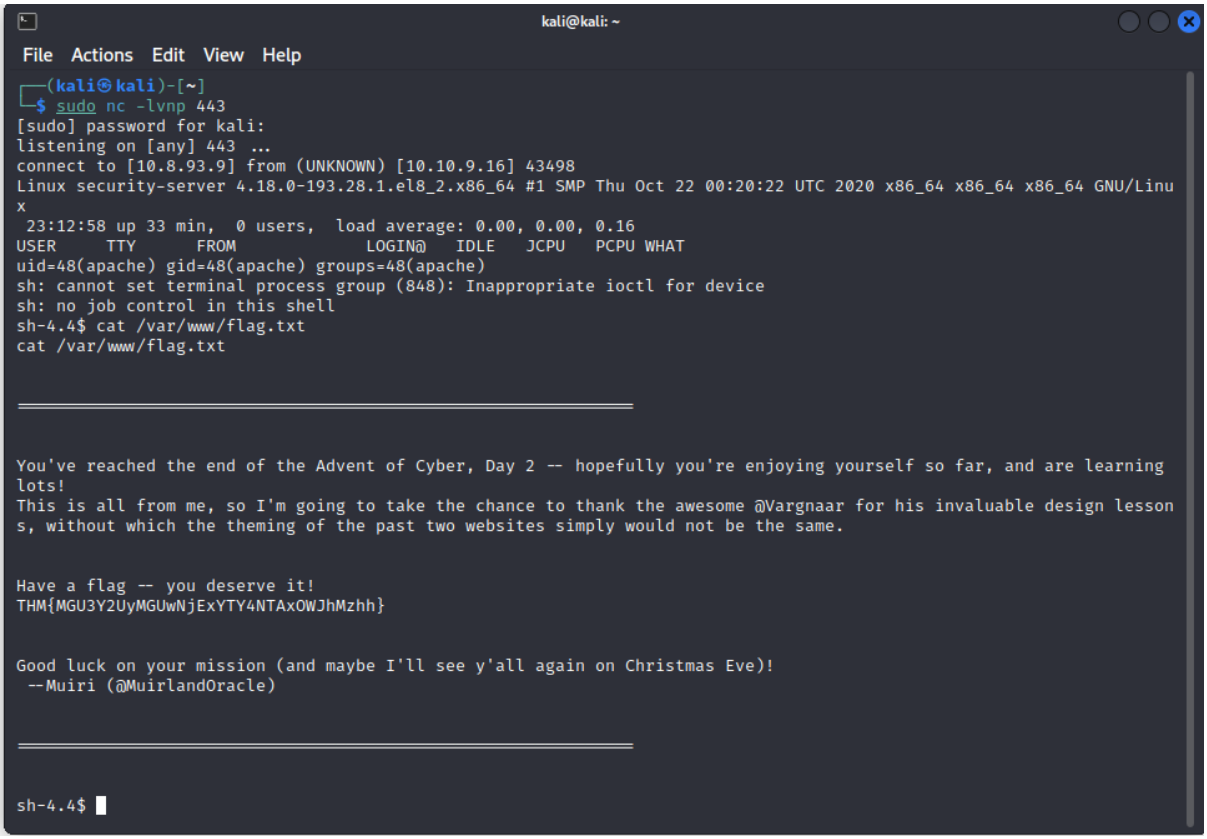
This method will confuse the system and allowing our malicious file to be uploaded

Earlier (in question 3), we found out that all the files are kept in a URL Path /upload/



After activated the file, our reverse shell connected back allowing us to have control to the system.

Knowing where the flag in, all we need to do now is search it using cat.



Thought Process/Methodology:

Once we enter the machine, we can see a box on the top right telling us to drop a file contains details about suspicious person we saw. To test this, we try uploading all kind of files to see what it accepts and what it doesn't. After countless of try, we know that it accepts only image files. Knowing this allow us to upload malicious scripts using double-barrelled extension. After uploading out reverse shell and setting up netcat, we trying to find the files location by using trial and error on common URL path, found out that the files are kept in /uploads/. We then proceed on activating the file we uploaded earlier. Once the netcat is in, we used it to find the flag from the path we already know. Then, the hack is successful.

Day 3: Web Exploitation – Christmas Chaos

Tools used: Kali Linux, Firefox, Burp Suite

Solution/Walkthrough:

Question 1

On the dossier, we got a text about 'Default Credentials'

Default Credentials

You've probably purchased (or downloaded a service/program) that provides you with a set of credentials at the start and requires you to change the password after it's set up (usually these credentials that are provided at the start are the same for every device/every copy of the software). The trouble with this is that if it's not changed, an attacker can look up (or even guess) the credentials.

What's even worse is that these devices are often exposed to the internet, potentially allowing anyone to access and control it. In 2018 it was reported that a botnet (a number of internet-connected devices controlled by an attacker to typically perform DDoS attacks) called Mirai took advantage of Internet of Things (IoT) devices by remotely logging, configuring the device to perform malicious attacks at the control of the attackers; the Mirai botnet infected over 600,000 IoT devices mostly by scanning the internet and using default credentials to gain access.

In fact, companies such as Starbucks and the US Department of Defense have been victim to leaving services running with default credentials, and bug hunters have been rewarded for reporting these very simple issues responsibly (Starbucks paid $250 for the reported issue):

- https://hackerone.com/reports/195163 - Starbucks, bug bounty for default credentials.
- https://hackerone.com/reports/804548 - US Dept Of Defense, admin access via default credentials.

In 2017, it was reported that 15% of all IoT devices still use default passwords.

SecLists is a collection of common lists including usernames, passwords, URLs and much more. A password list known as "rockyou.txt" is commonly used in security challenges, and should definitely be a part of your security toolkit.

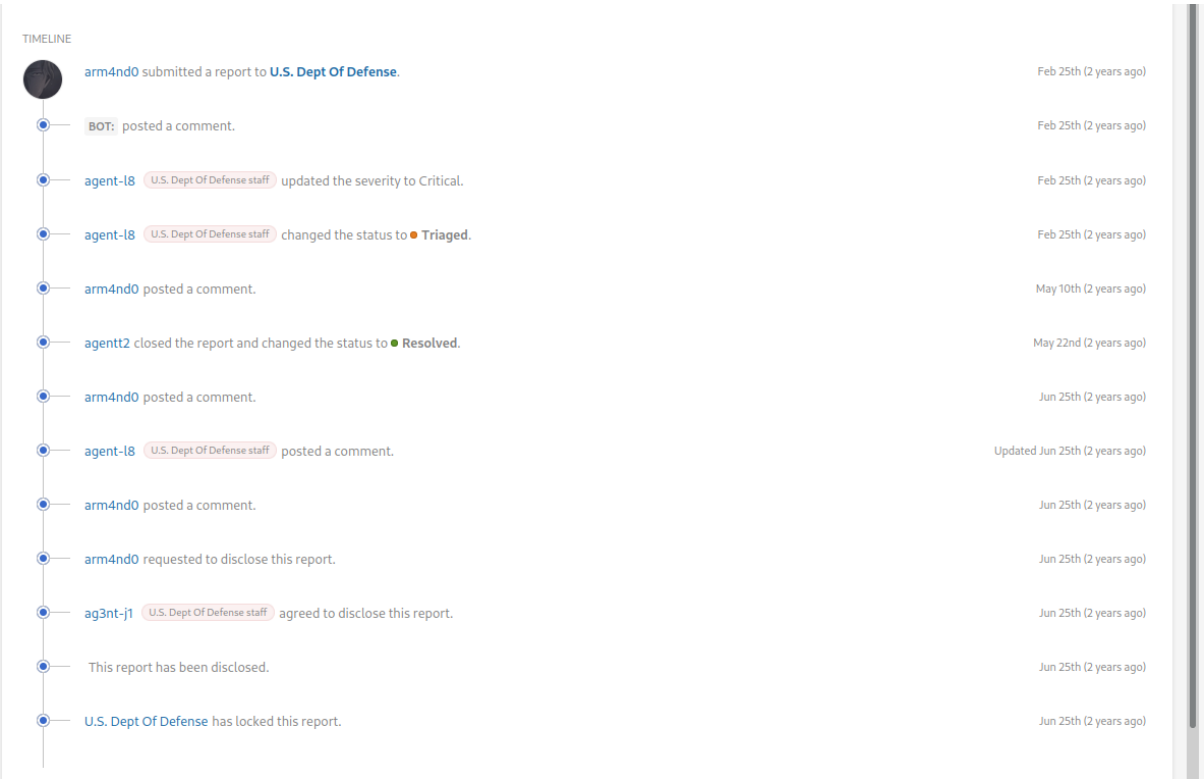
After reading it, we know that the name of the botnet mentioned in the text that was reported in 2018 is **Mirai**

Question 2

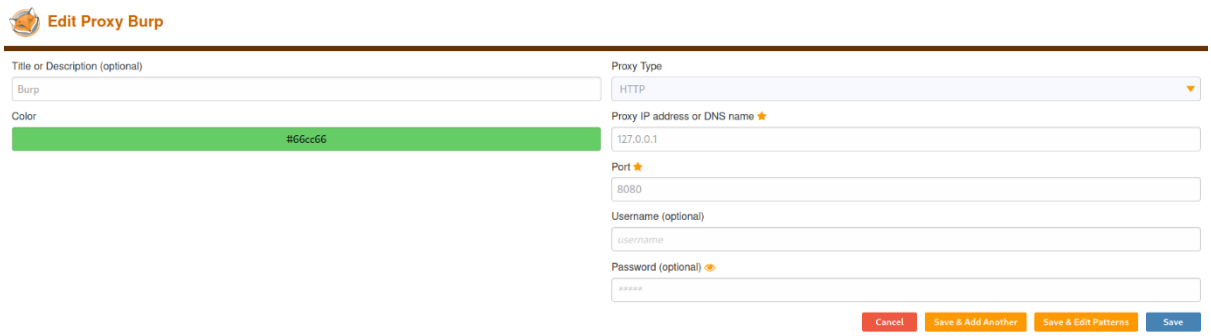
On the third paragraph, it is mentioned that Starbucks paid 250 USD for reporting default credentials

Question 3

Based on the report Hackerone ID:804548 timeline, the agent assigned from the Dept of Defense that disclosed the report on Jun 25th is ag3nt-j1



Question 4

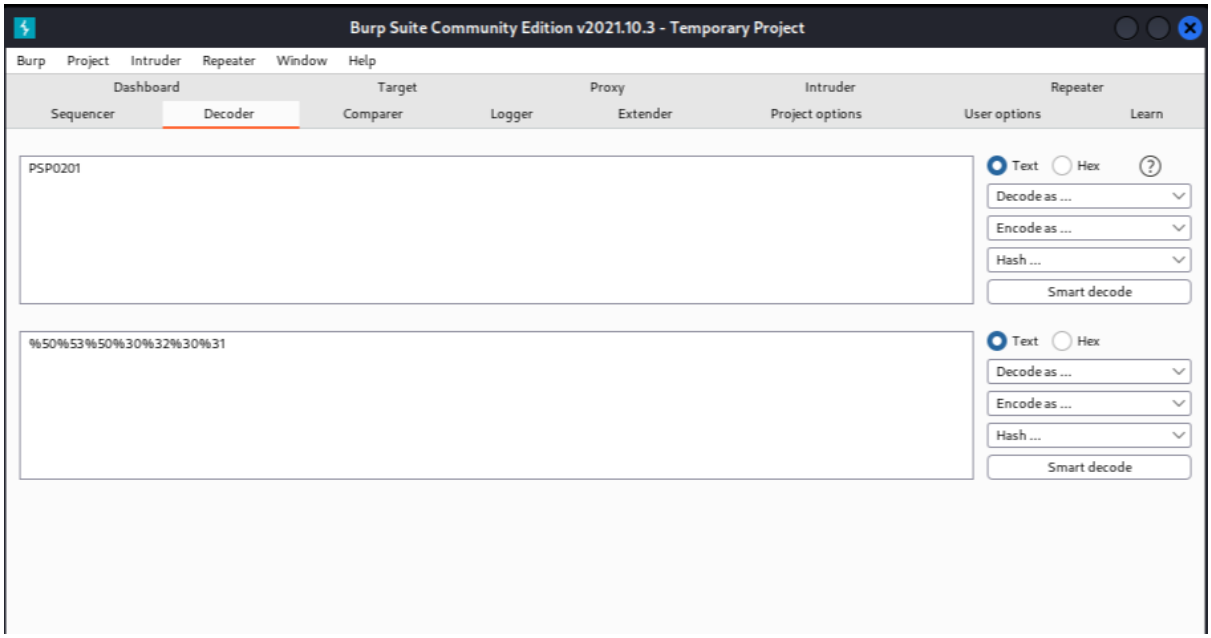


The port number for Burp is **8080**

Question 5

The proxy type of Burp is **HTTP**

Question 6



Playing with the encoder, we got the URL encoding for "PSP0201" in hex.

Question 7

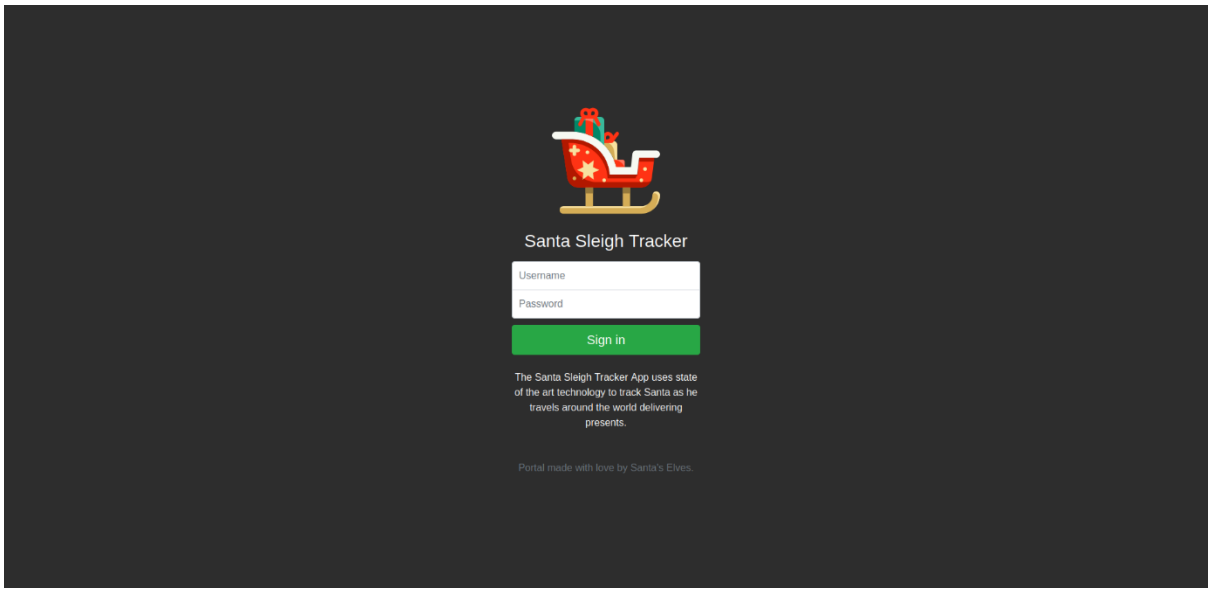
To look up the types of attack and what they do, we went on reading the Burp documentation.

- **Cluster bomb** - This uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested. I.e., if there are two payload positions, the attack will place the first payload from payload set 2 into position 2, and iterate through all the payloads in payload set 1 in position 1; it will then place the second payload from payload set 2 into position 2, and iterate through all the payloads in payload set 1 in position 1. This attack type is useful where an attack requires different and unrelated or unknown input to be inserted in multiple places within the request (e.g. when guessing credentials, a username in one parameter, and a password in another parameter). The total number of requests generated in the attack is the product of the number of payloads in all defined payload sets - this may be extremely large.

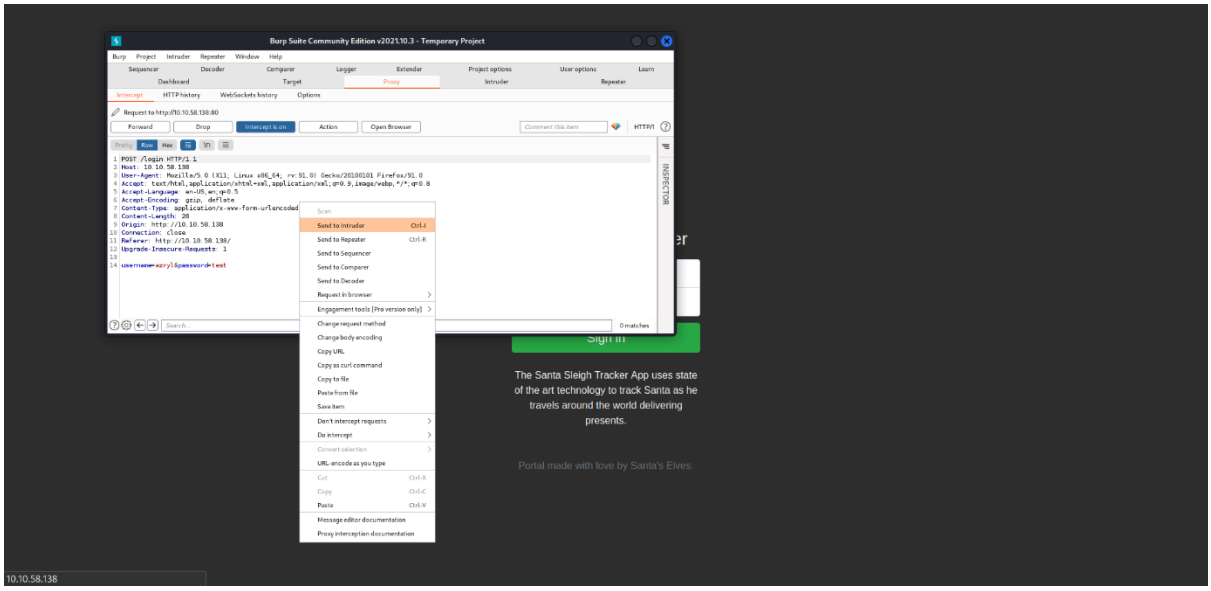
After reading all the types, we found out that the attack that matches the description given is **Cluster bomb**.

Question 8

When we first accessing the machine, we saw a login page.



In order to gain the access, We will be brute forcing the login.

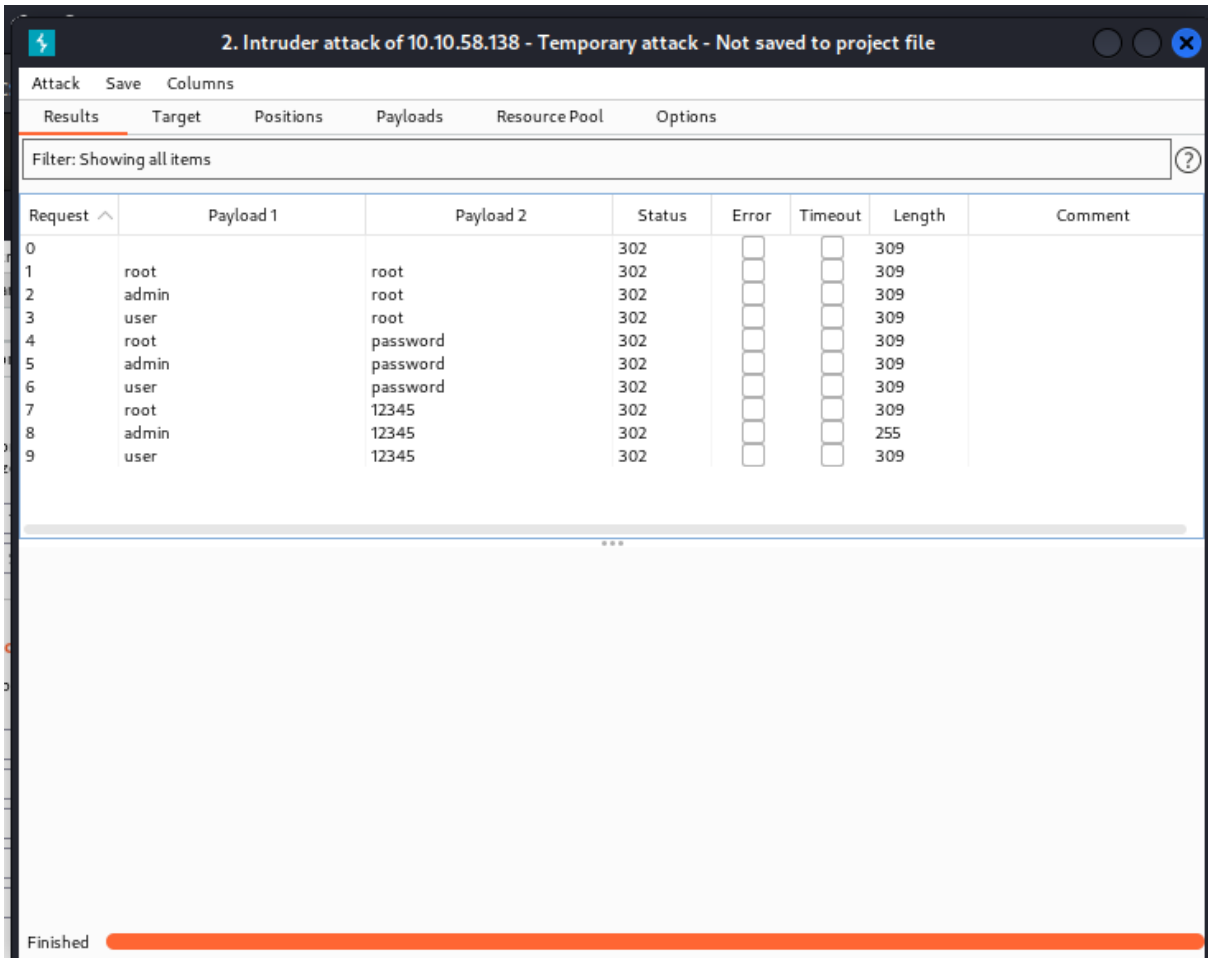


We were using Burp for this. By intercepting the request, we able to manipulate the request using the tools in Burp. We were using intruder as it able us to loop through and submit a login request using the list of default credentials.

The list of default credentials we were looping is as below:

Username	Password
root	root
admin	password
user	12345

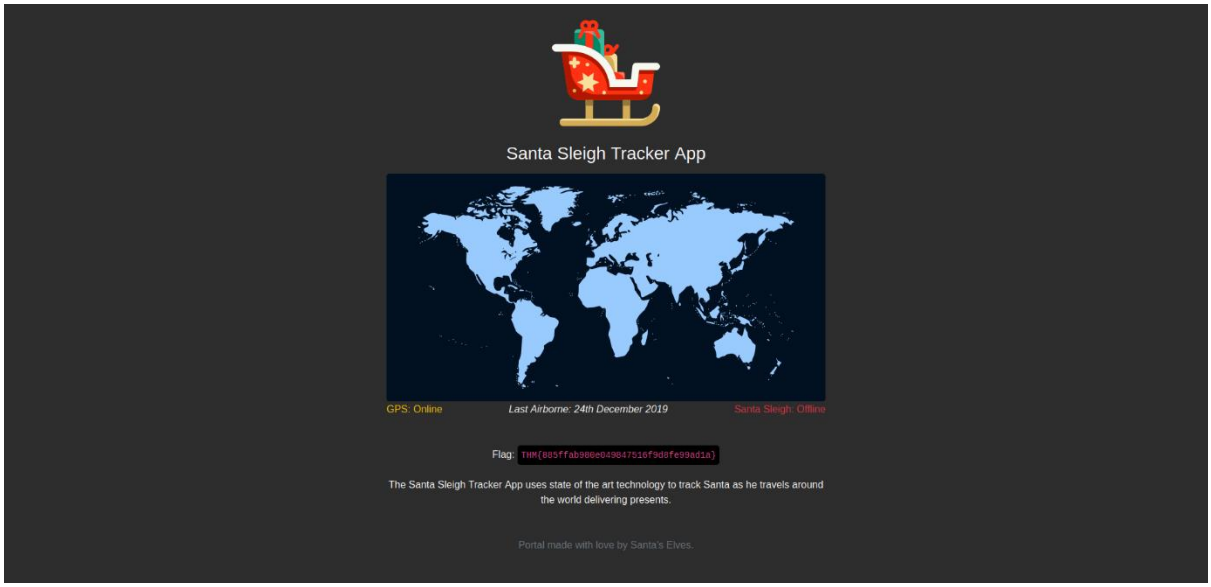
We loop each of the username to each of the password using the Burpsuite.



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
0			302			309	
1	root	root	302			309	
2	admin	root	302			309	
3	user	root	302			309	
4	root	password	302			309	
5	admin	password	302			309	
6	user	password	302			309	
7	root	12345	302			309	
8	admin	12345	302			255	
9	user	12345	302			309	

The request 8 (username = admin, password= 12345) shows a different length indicates the correct combination.

After login, we successfully get the flag.



Thought Process/Methodology:

When we first accessing the machine, we saw a login page. We then decided on brute forcing the login panel. We used Burpsuite to accomplish this. First, we intercept the traffic of login to get the POST request of the login to send our brute force to. Then, we setting up all the parameters of the login and insert the payloads with common usernames and passwords. Once we hit attack, all the usernames' payloads will be paired with each of the passwords payloads as we are using cluster bomb attack. We saw a different length on one of the requests indicates that the login succeed. Then, we are logging in with the right login details and the flag has been captured.

Day 4: Web Exploitation – Santa's watching

Tools used: Kali Linux, Firefox

Solution/Walkthrough:

Question 1

The correct wfuzz command for the info given will be

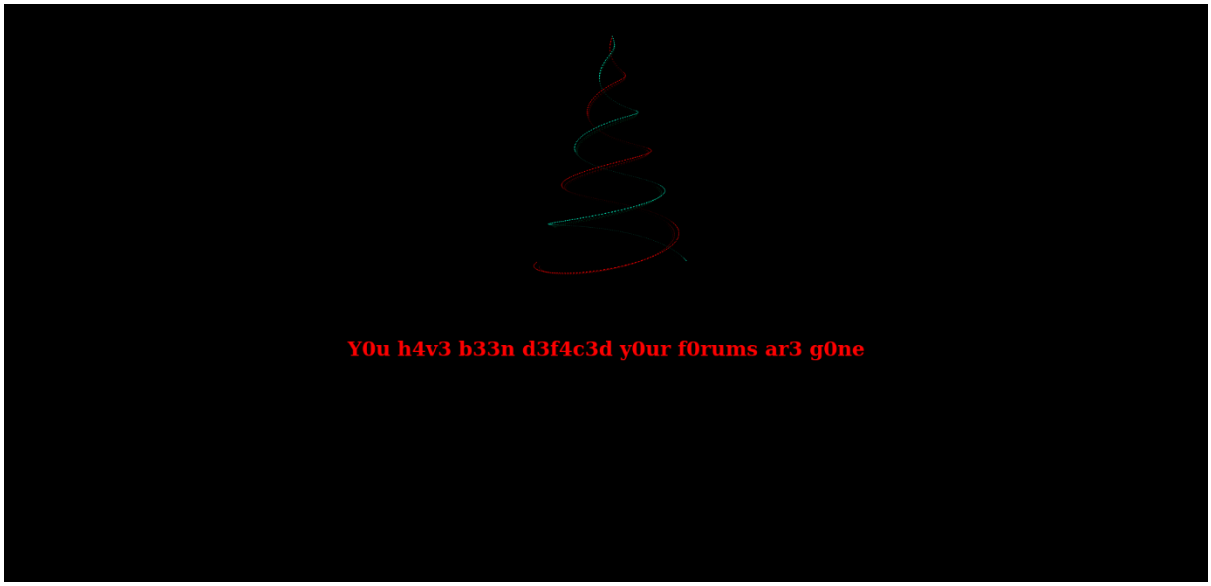
Given the URL "http://shibes.xyz/api.php", what would the entire wfuzz command look like to query the "breed" parameter using the wordlist "big.txt" (assume that "big.txt" is in your current directory)

Note: For legal reasons, do *not* actually run this command as the site in question has not consented to being fuzzed!

wfuzz -c -z file,big.txt http://shibes.xyz/api.php?breed=FUZZ | Correct Answer | Hint

Question 2

The moment we access the machine, it shows a page without any login panel or hotspot to click on.



We figured that there must be a directory which isn't shown, so we are using gobuster to enumerate through a list of paths.

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ gobuster dir  -u 10.10.106.233/ -w big.txt -x php,txt,html
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.106.233/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php,txt,html
[+] Timeout:                 10s
===============================================================
2022/06/24 01:57:24 Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd.html       (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/.htaccess.php        (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/.htaccess.txt        (Status: 403) [Size: 278]
/.htpasswd.php        (Status: 403) [Size: 278]
/.htaccess.html       (Status: 403) [Size: 278]
/.htpasswd.txt        (Status: 403) [Size: 278]
/LICENSE              (Status: 200) [Size: 1086]
/api                  (Status: 301) [Size: 312] [──> http://10.10.106.233/api
```

Once the gobuster done brute forcing the directory, we found there's a directory called 'api'

Index of /api

Name	Last modified	Size	Description
Parent Directory		-	
site-log.php	2020-11-22 06:38	110	

Apache/2.4.29 (Ubuntu) Server at 10.10.106.233 Port 80

Inside there's file named "**site-log.php**"

Question 3

We will then FUZZ the date parameter on the file

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ wfuzz -c -z file,wordlist -u http://10.10.106.233/api/site-log.php?date=FUZZ
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Ope
correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.106.233/api/site-log.php?date=FUZZ
Total requests: 63

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000037:   200        0 L      0 W        0 Ch        "20201206"
000000036:   200        0 L      0 W        0 Ch        "20201205"
000000003:   200        0 L      0 W        0 Ch        "20201102"
000000032:   200        0 L      0 W        0 Ch        "20201201"
000000030:   200        0 L      0 W        0 Ch        "20201129"
000000035:   200        0 L      0 W        0 Ch        "20201204"
000000029:   200        0 L      0 W        0 Ch        "20201128"
000000033:   200        0 L      0 W        0 Ch        "20201202"
000000034:   200        0 L      0 W        0 Ch        "20201203"
000000015:   200        0 L      0 W        0 Ch        "20201114"
000000001:   200        0 L      0 W        0 Ch        "20201100"
000000038:   200        0 L      0 W        0 Ch        "20201207"
000000025:   200        0 L      0 W        0 Ch        "20201124"
000000023:   200        0 L      0 W        0 Ch        "20201122"
000000022:   200        0 L      0 W        0 Ch        "20201121"
000000021:   200        0 L      0 W        0 Ch        "20201120"
000000020:   200        0 L      0 W        0 Ch        "20201119"
000000026:   200        0 L      1 W        13 Ch       "20201125"
000000017:   200        0 L      0 W        0 Ch        "20201116"
```

The payload “20201125” shows that there are characters inside. This means that it able to access the thing inside the php file and omit the result.

We open the /site-log.php with the parameter found and got the flag

```
10.10.106.233/api/site-log.php?date=20201125
Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  OffSec
```

THM{D4t3_AP1}

Question 4

We accessing the advance help file of the wfuzz for the question.

```
        -v                          : Verbose information.
        -f filename,printer         : Store results in the output file using the specified printer (raw printer if omitte
d).
        -o printer                  : Show results using the specified printer.
        --interact                  : (beta) If selected,all key presses are captured. This allows you to interact with t
he program.
        --dry-run                   : Print the results of applying the requests without actually making any HTTP request
.
        --prev                      : Print the previous HTTP requests (only when using payloads generating fuzzresults)
        --efield <expr>             : Show the specified language expression together with the current payload. Repeat fo
```

We found out that '-f' parameter store results in filename, printer,

Thought Process/Methodology:

The moment we access the machine, it shows a page without any login panel or hotspot to click on. We figured that there must be a directory which isn't shown, so we are using gobuster to enumerate through a list of paths. Once the gobuster done brute forcing the directory, we found there's a directory called 'api'. We then access the /api directory to find out that it contains a file named "site-log.php". To access it we need to have a correct log date so we FUZZ the date parameter to see the content of the log. Once the fuzzing is done, we got the correct date and put it as a parameter on the URL. Inside contain a flag for today hacking mission and we are done.

Day 5: Web Exploitation – Someone stole Santa's gift list!

Tools used: Kali Linux, Firefox, Burpsuite

Solution/Walkthrough:

Question 1

By default SQL Server listens on TCP port number **1433**, but for named instances the TCP port is dynamically configured. There are several options available to get the listening port for a SQL Server named instance.

Question 2

Without brute forcing, we looked at the hint given and guess the secret login panel.

Without using directory brute forcing, what's Santa's secret login panel?

/santapanel

Question 3

Santa reads some documentation that he wrote when setting up the application, it reads:

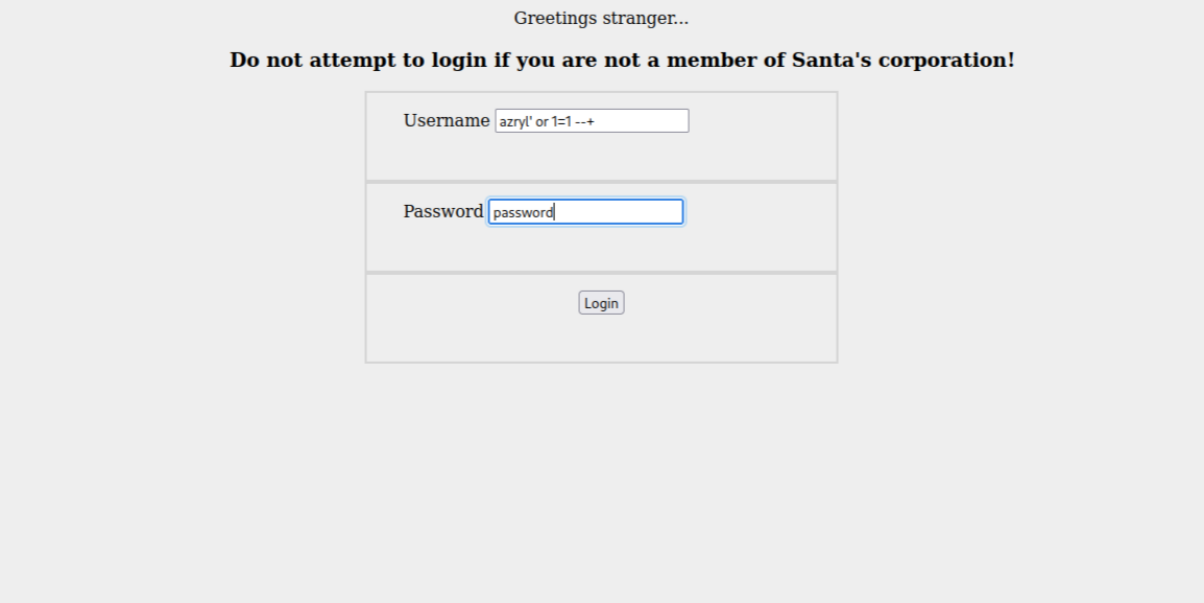
Santa's TODO: Look at alternative database systems that are better than sqlite. Also, don't forget that you installed a Web Application Firewall (WAF) after last year's attack. In case you've forgotten the command, you can tell SQLMap to try and bypass the WAF by using `--tamper=space2comment`

The Santa's TODO tells us that he is currently be using **SQLite** as the database.

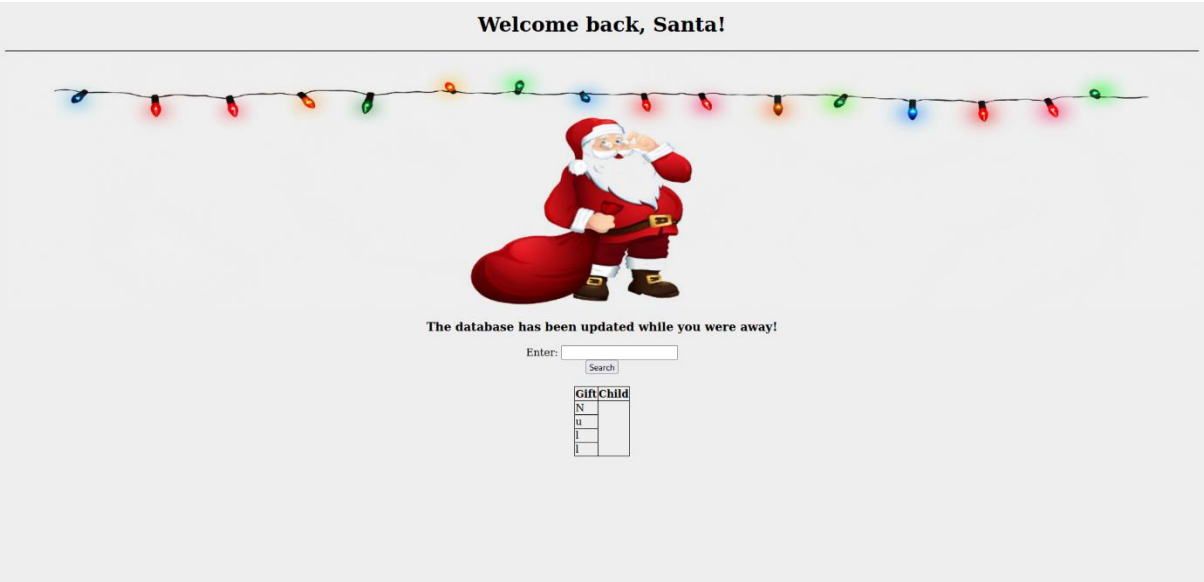
Question 4

Once we access the Santa's secret login panel, we saw a login page.

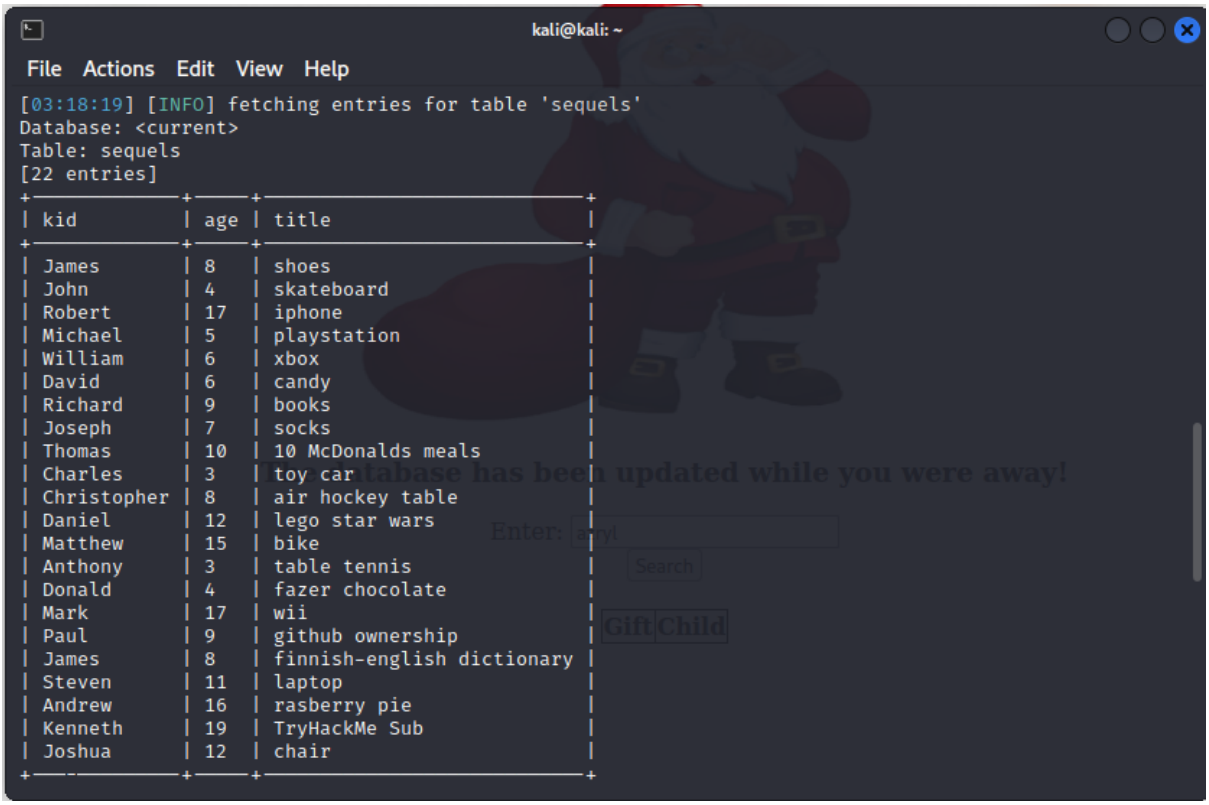
To gain the access, we are using SQL injection (SQLi) on the login panel.



This allow us to check the database.



Now all we need to do is dumping all the database. To do this, we used sqlmap.



```
kali@kali: ~
File  Actions  Edit  View  Help
[03:18:19] [INFO] fetching entries for table 'sequels'
Database: <current>
Table: sequels
[22 entries]
+-------------+-----+---------------------------+
| kid         | age | title                     |
+-------------+-----+---------------------------+
| James       | 8   | shoes                     |
| John        | 4   | skateboard                |
| Robert      | 17  | iphone                    |
| Michael     | 5   | playstation               |
| William     | 6   | xbox                      |
| David       | 6   | candy                     |
| Richard     | 9   | books                     |
| Joseph      | 7   | socks                     |
| Thomas      | 10  | 10 McDonalds meals        |
| Charles     | 3   | toy car                   |
| Christopher | 8   | air hockey table          |
| Daniel      | 12  | lego star wars            |
| Matthew     | 15  | bike                      |
| Anthony     | 3   | table tennis              |
| Donald      | 4   | fazer chocolate           |
| Mark        | 17  | wii                       |
| Paul        | 9   | github ownership          |
| James       | 8   | finnish-english dictionary |
| Steven      | 11  | laptop                    |
| Andrew      | 16  | rasberry pie              |
| Kenneth     | 19  | TryHackMe Sub             |
| Joshua      | 12  | chair                     |
+-------------+-----+---------------------------+
```

The data shows us that there are **22** entries all together in Santa's database.

Question 5

James' age is **8** as seen in the database.

Question 6

Also from the database, we found out that Paul is asking for **github ownerships.**

Question 7

As we scroll down, we can also find the flag for today.

```
Database: <current>
Table: hidden_table
[1 entry]
+-----------------------------------------+
| flag                                    |
+-----------------------------------------+
| thmfox{All_I_Want_for_Christmas_Is_You} |
+-----------------------------------------+
```

Question 8

The next table to the flag is the user details including the password.

```
[03:18:20] [INFO] fetching entries for table 'users'
Database: <current>
Table: users
[1 entry]
+------------------+----------+
| password         | username |
+------------------+----------+
| EhCNSWzzFP6sc7gB | admin    |
+------------------+----------+
```

Thought Process/Methodology:

Once we access the Santa's secret login panel, we saw a login page. To gain the access, we are using SQL injection (SQLi) on the login panel. This allows us to check the database. But then not all the data is there so we need to dump all contents in the database in order view all the gifts list. To do this, we used sqlmap. The data shows us that there are **22** entries all together in Santa's database. As we scroll down, we can also find the flag for today. Hacking done!