PSP0201

Week 2

Writeup

Group Name: Woohoo

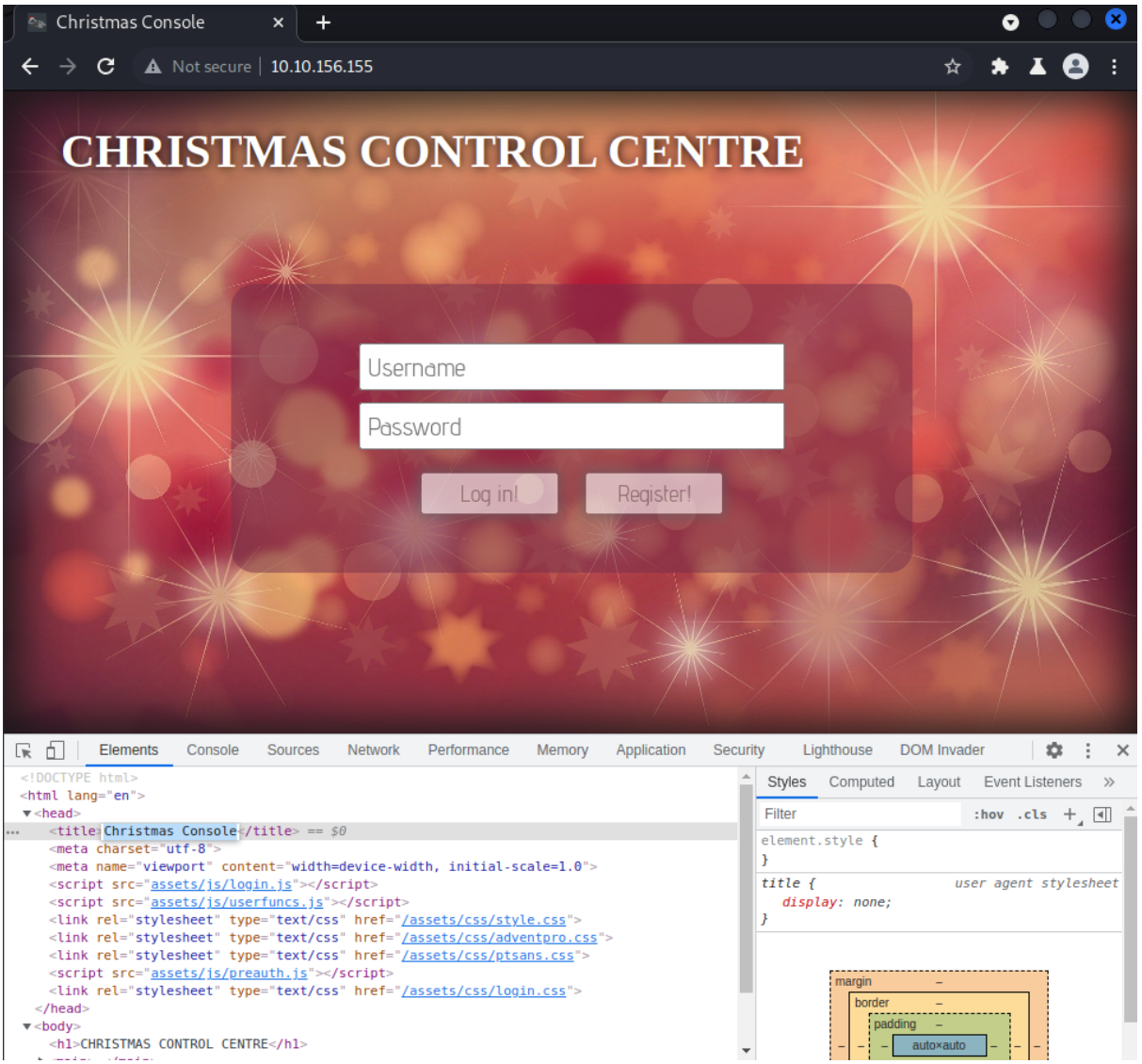
Members

ID	Name	Role
1211100312	CHAN HAO YANG	Leader
1211101506	LEONG JIA YI	Member
1211101961	CHAI DI SHENG	Member
1211101726	TAI JIN PEI	Member

Day 1: [Web Exploitation] Christmas Chaos

Tools used: Kali Linux, Firefox, Burp Suite Community Edition

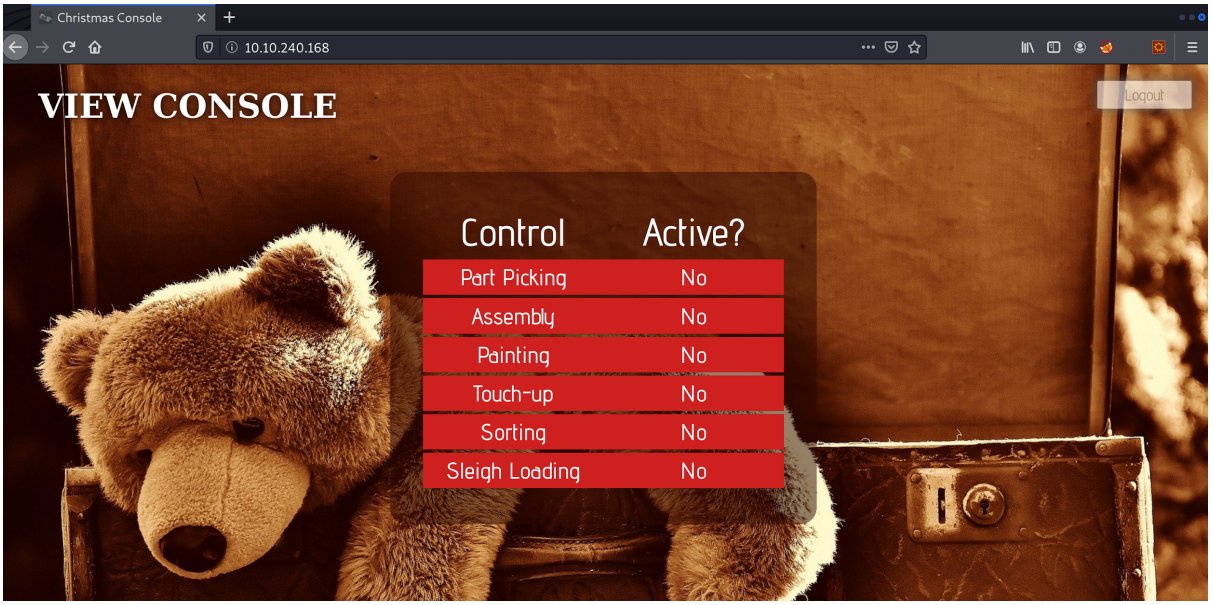
Solution/walkthrough:



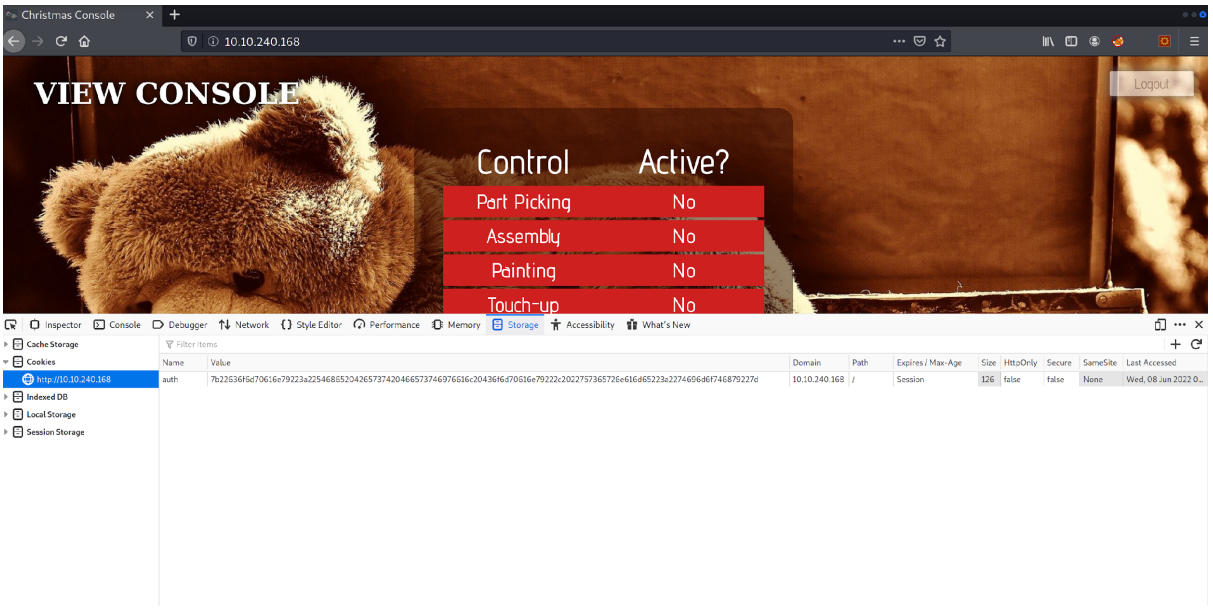
Inspect the website

Q1: Inspect the website. What is the title of the website?

Christmas Console



Register a new user and then login.



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

Q2: What is the name of the cookie used for authentication?

auth

Value
7b22636f6d70616e79223a2254686520426573742046657374976616c20436f6d70616e79222c2022757365726e616d65223a2274696d6f746879227d

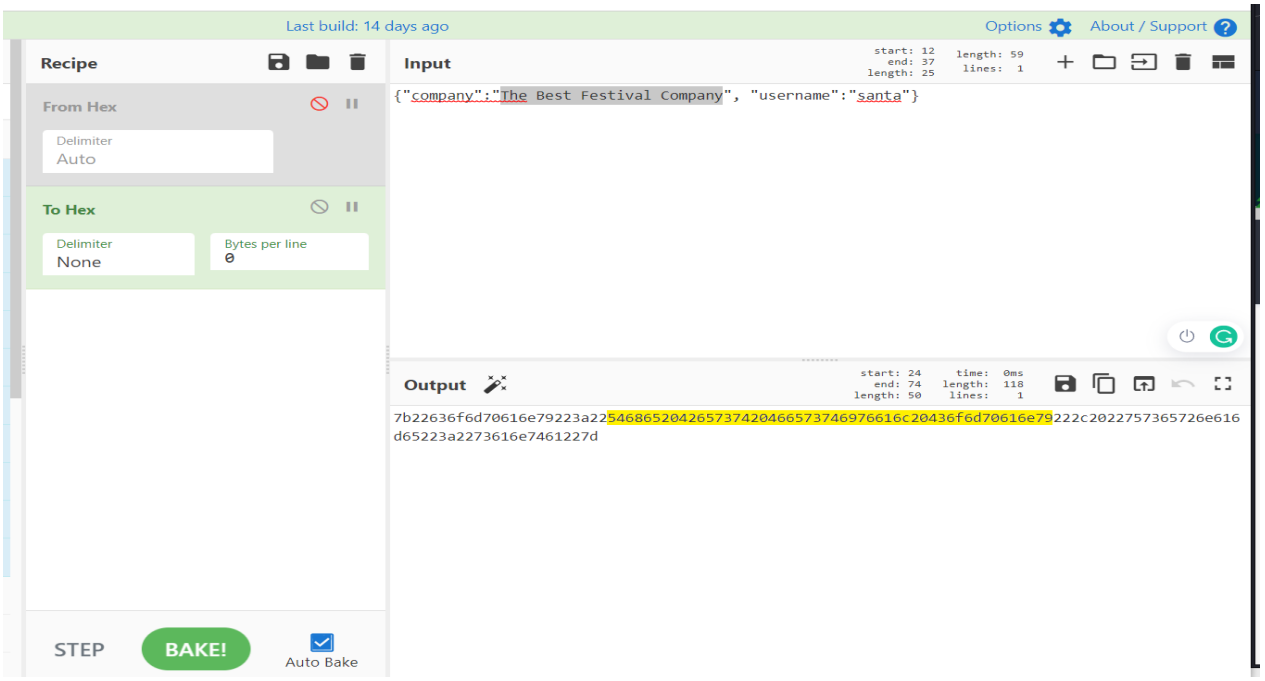
Q3: In what format is the value of this cookie encoded?

Hexadecimal

Use Cyberchef, convert the cookie value to string.

Q4: Having decoded the cookie, what format is the data stored in?

JSON

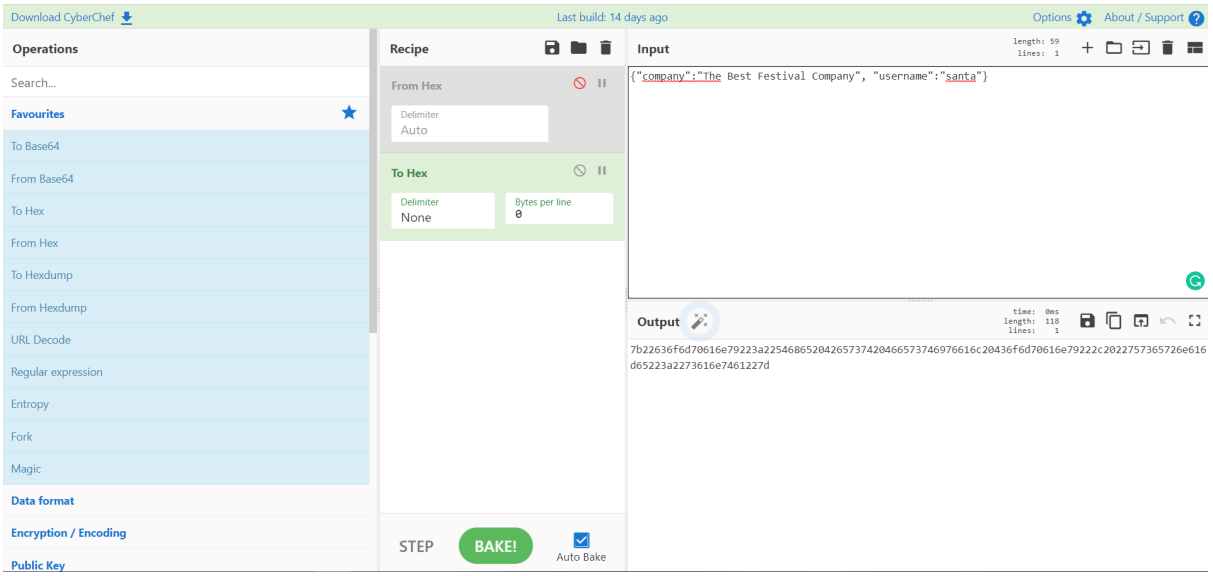


Q5: What is the value for the company field in the cookie?

546865204265737420466573746976616c20436f6d70616e79

Q6: What is the other field found in the cookie?

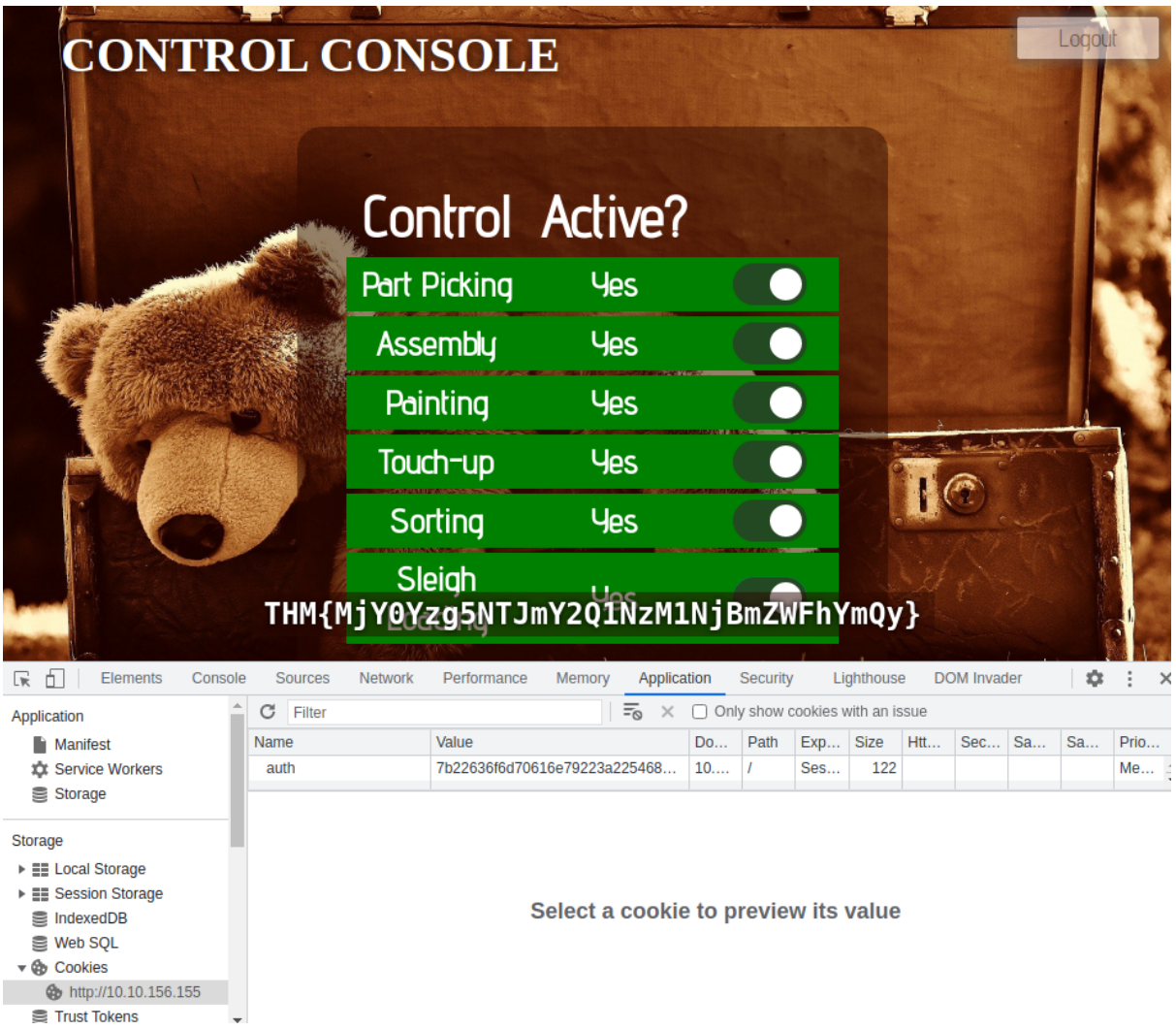
username



Change the username to ‘santa’, convert the JSON statement to hex.

Q7: What is the value of Santa's cookie?

7b22636f6d70616e79223a2254686520426573742046657374697616c20436f6d70616e79222c2022757365726e616d65223a2273616e7461227d



Replace the Value with the converted Santa's cookie, then active all the lines.

Then there's the flag!

Q8: What is the flag you're given when the line is fully active?

THM{MjY0Yzg5NTJmY2Q1NzM1NjBmZWFhYmQy}

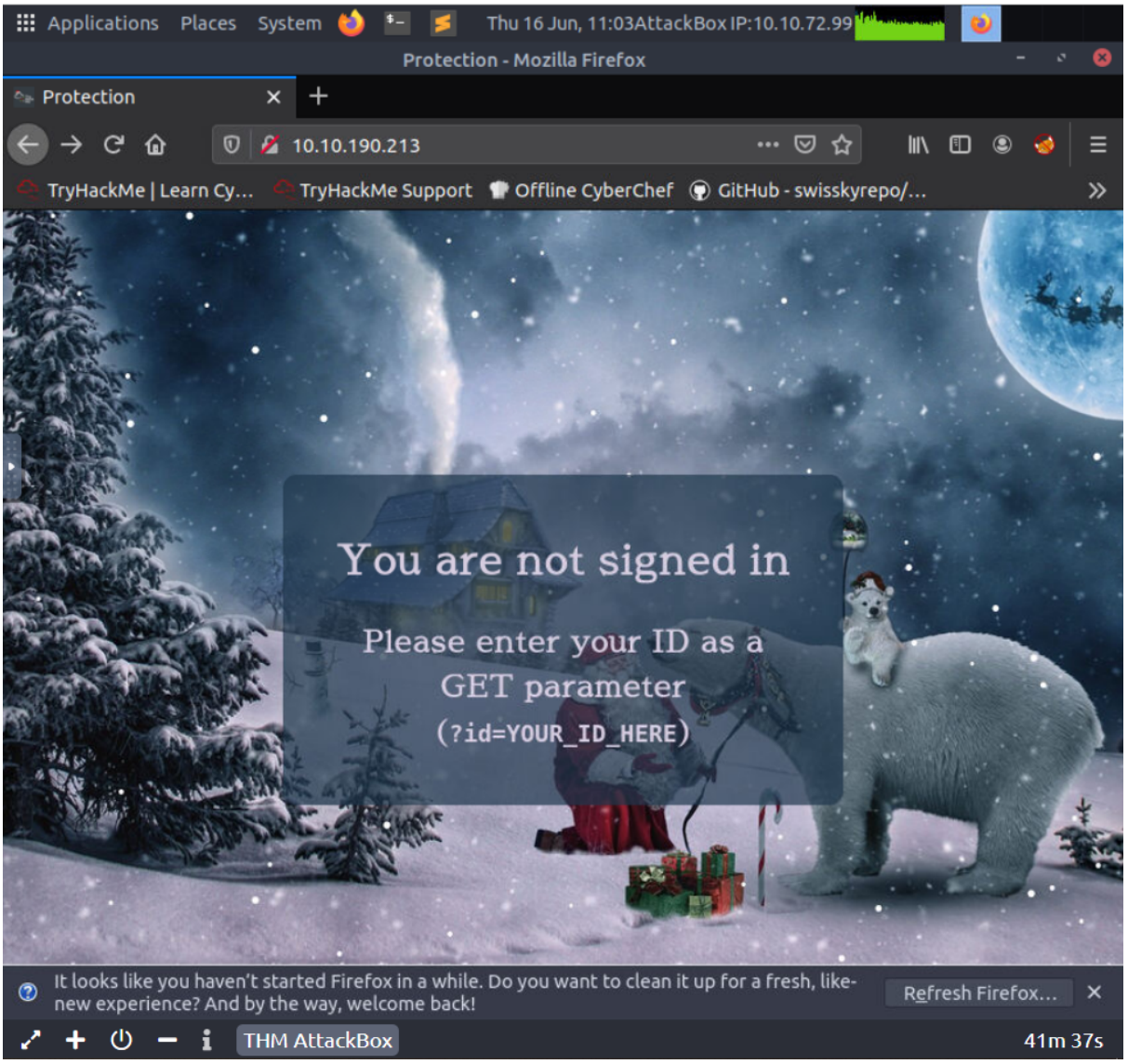
Thought Process/Methodology:

First, 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 inspect the browser 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 change 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.

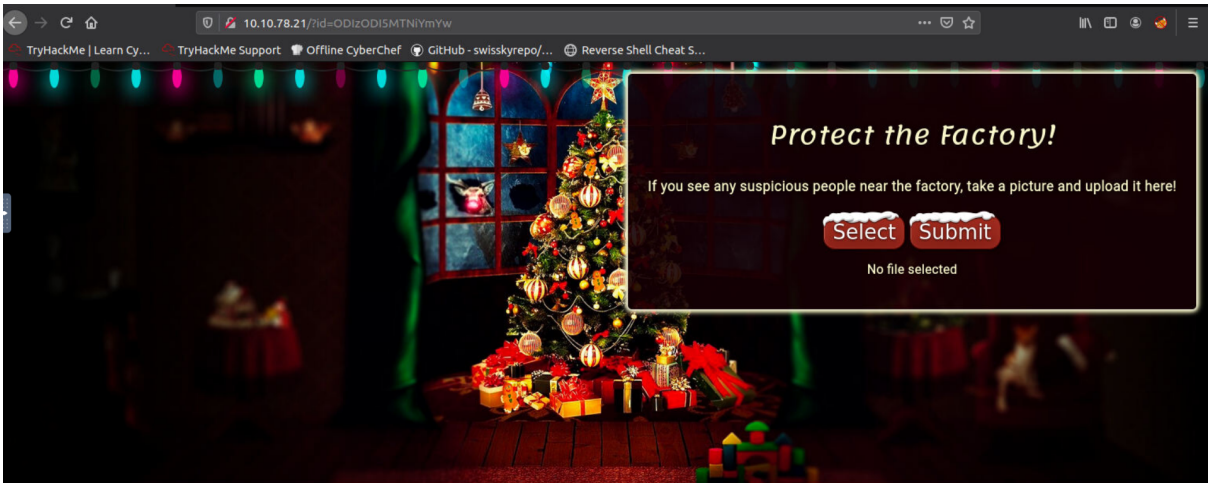
Day 2: [Web Exploitation] The Elf Strikes Back!

Tools used: Kali Linux, Firefox, Burp Suite Community Edition, Sublime Text, Terminal

Solution/walkthrough:



Start machine



add “/?id=ODIzODI5MTNiYmYw” into the link “http://10.10.78.21/”

? : used to specify that a GET parameter is forthcoming.

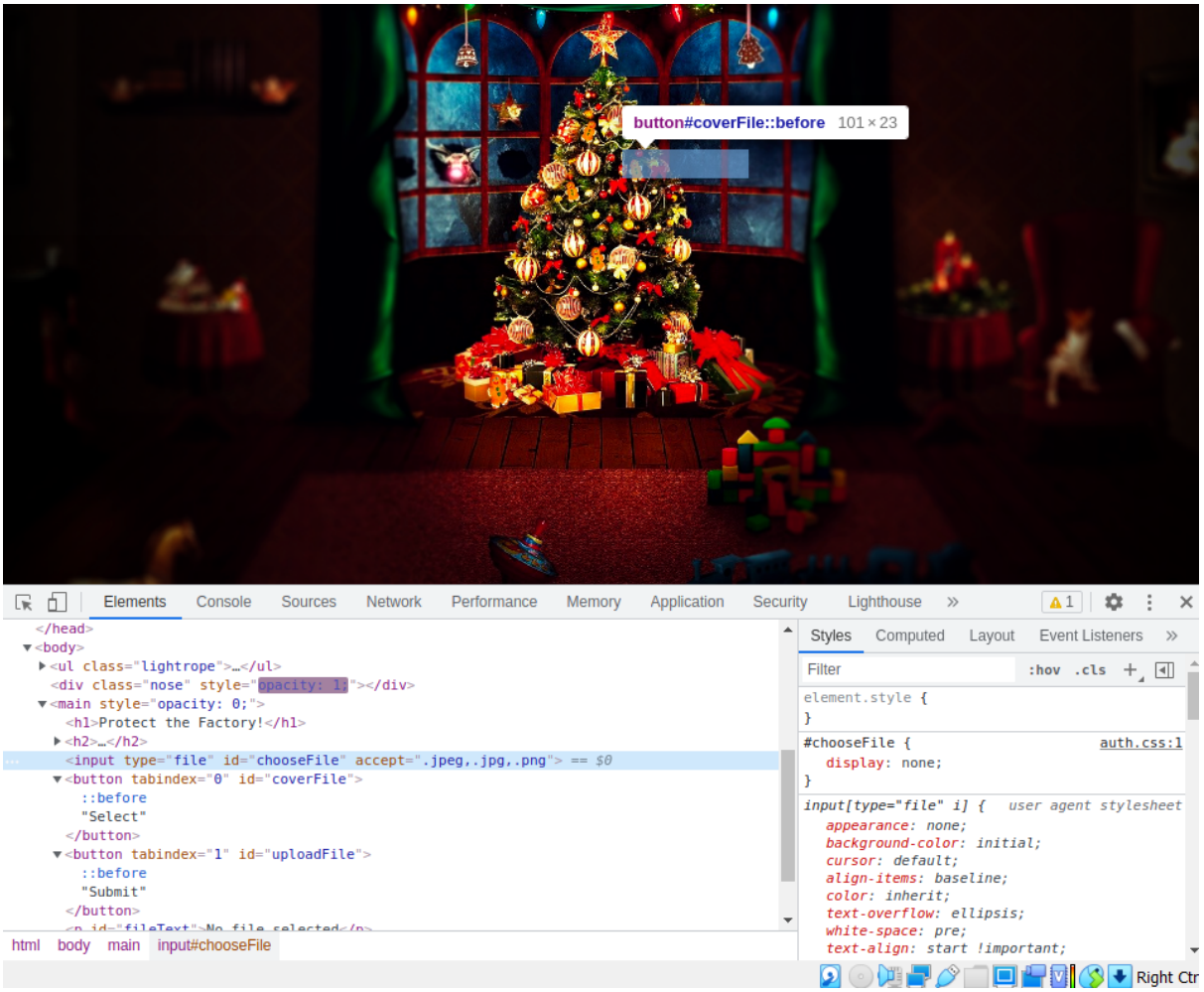
Parameter name (id) : used to identify the parameter to the server.

Equals sign (=) : indicating that the value will come next.

value : ODIzODI5MTNiYmYw

Q1: What string of text needs adding to the URL to get access to the upload page?

?id=ODIzODI5MTNiYmYw



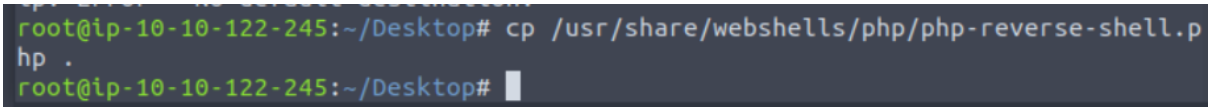
Inspect the webdiste and from the elements, we can know the file accepted is image(jpeg, jpg, png).

1.Find a file upload point.

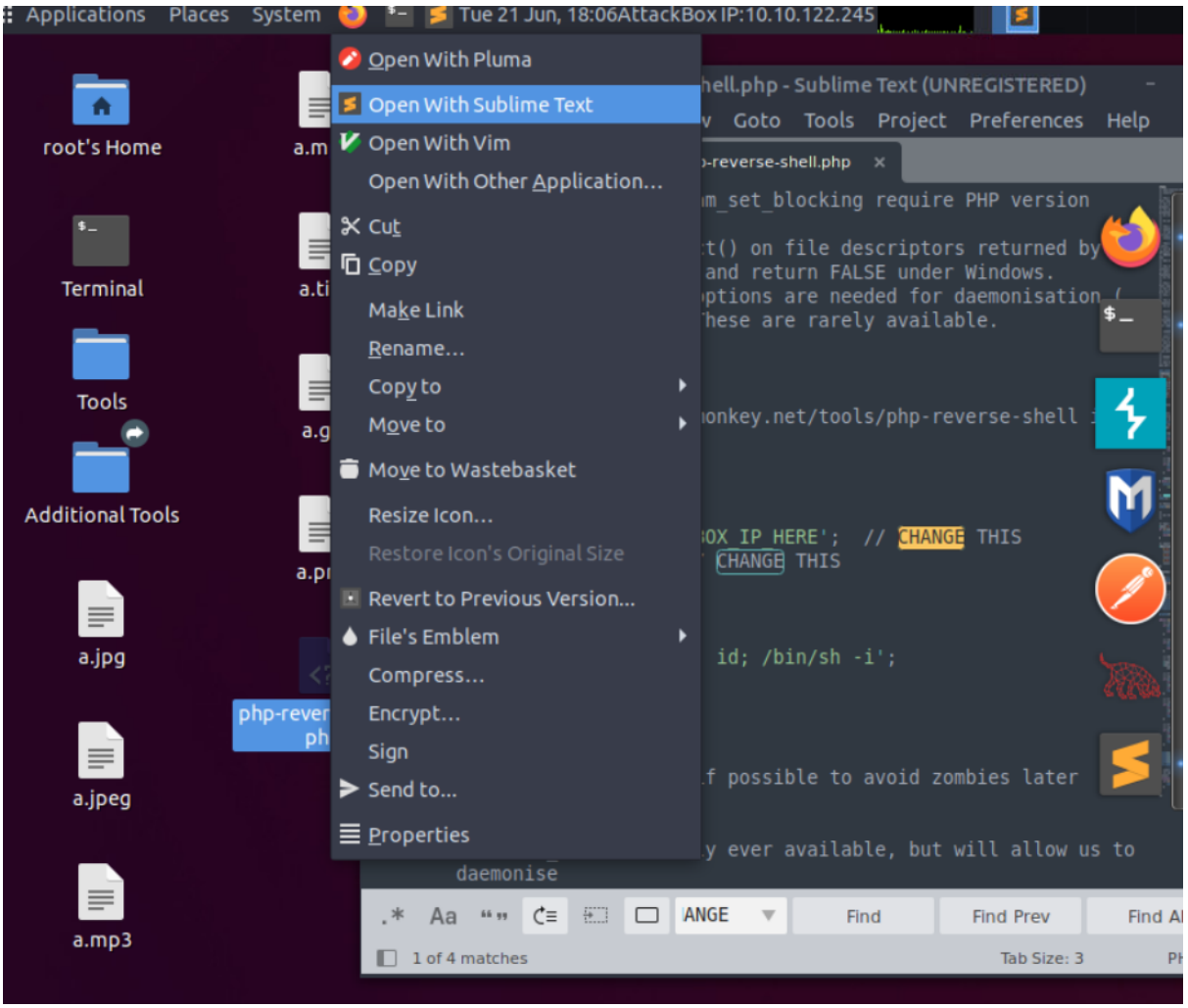
2.Try uploading some innocent files -- what does it accept? (Images, text files, PDFs, etc)

Q2: What type of file is accepted by the site?

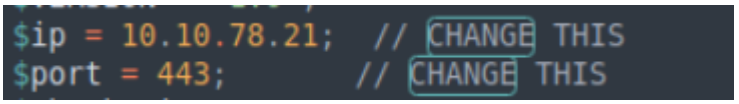
image



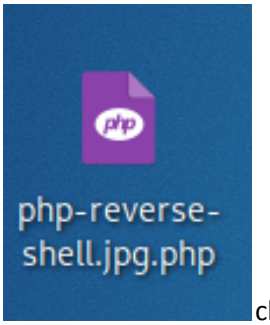
copy and paste ”cp /usr/share/webshells/php/php-reverse-shell.php .” into terminal to create the php reverse shell file.



open the “php-reverse-shell.php” with text editor and search “CHANGE” in the text.



change the ip and the port, then save it.



change the file name to “php-reverse-shell.jpg.php”

Back to the website, press ”Select”, show all files type and Submit “php-reverse-shell.jpg.php .”.

We have bypass the filter and uploaded the reverse shell successfully.

change the link to: http://10.10.78.21**/uploads**/?id=ODIzODI5MTNiYmYw and open the php reverse shell file.

3.Find the directory containing your uploads.

4.Try to bypass any filters and upload a reverse shell.

Q3: In which directory are the uploaded files stored?

/uploads/

```
└─$ nc -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 [dangerous!!]
        -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 scanned
        -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 secs
        -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 reads
        -C                      Send CRLF as line-ending
        -z                      zero-I/O mode [used for scanning]
```

nc : netcat

-h : help

Q4: Read up on netcat's parameter explanations. Match the parameter with the explanation below.

l : listen mode, for inbound connects

v : verbose [use twice to be more verbose]

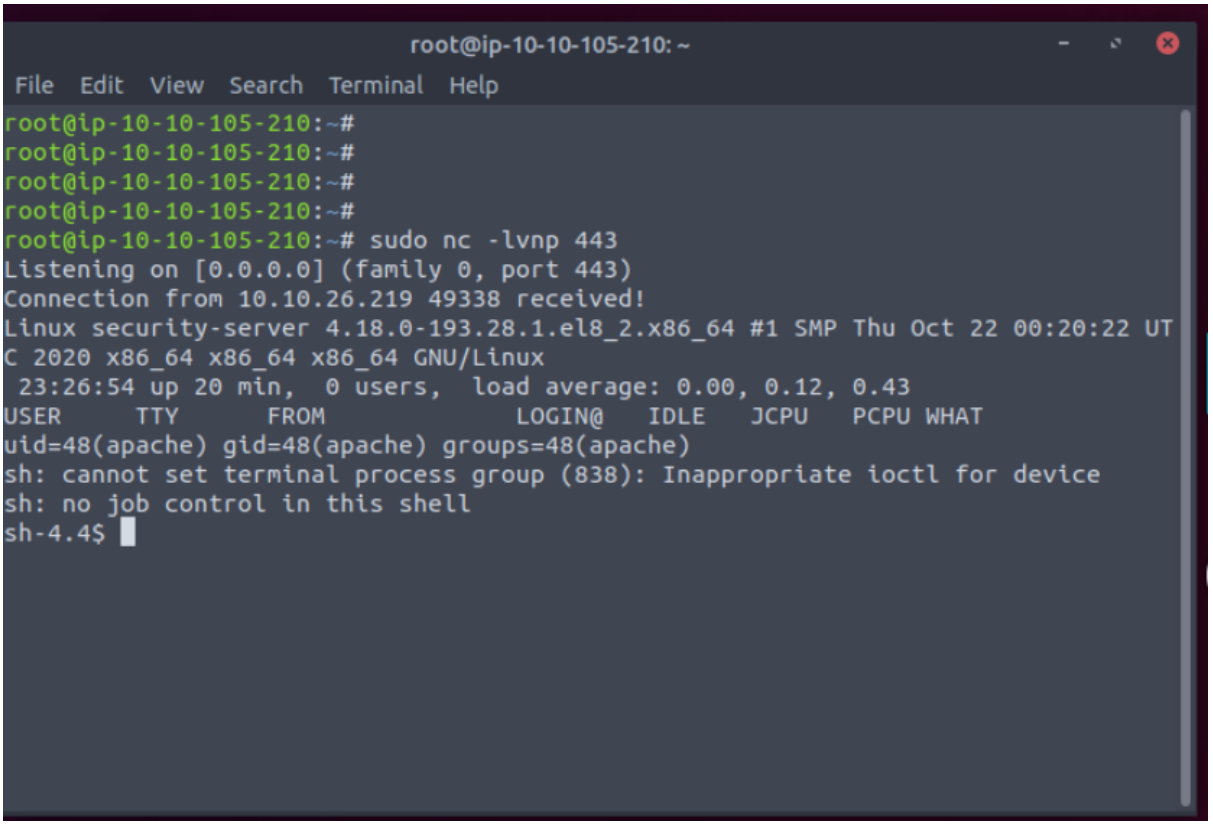
n : numeric-only IP addresses, no DNS

p(port) : local port number

```
root@ip-10-10-122-245:~/Desktop# sudo nc -lvnp 443
Listening on [0.0.0.0] (family 0, port 443)
```

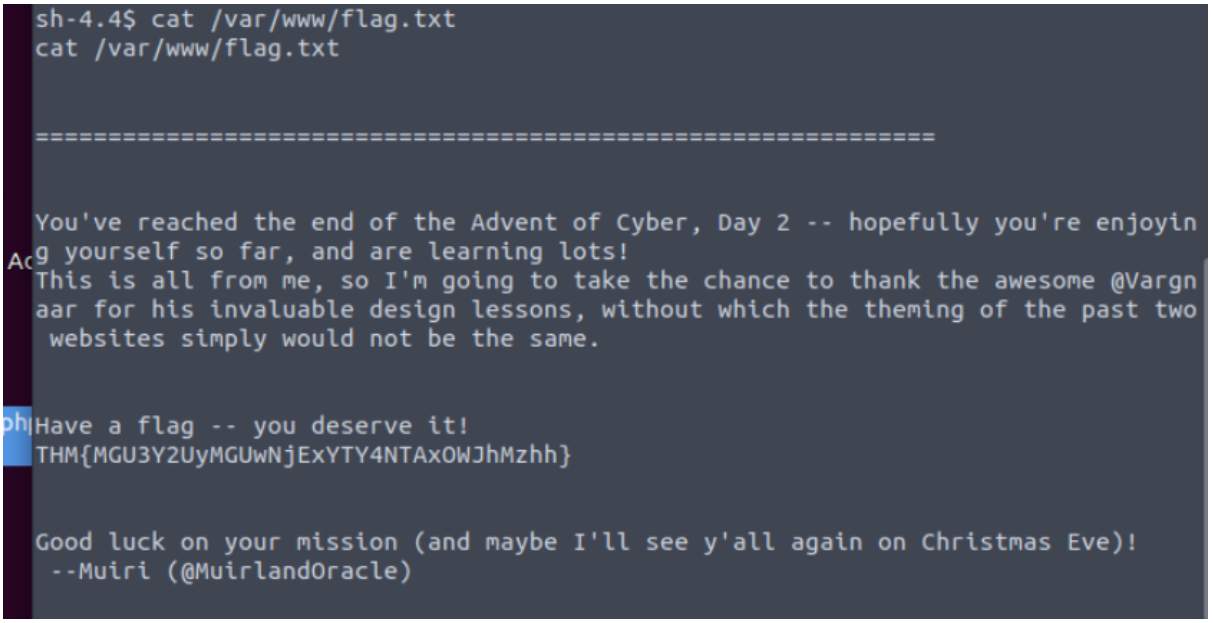
create a listener for the uploaded reverse shell by using the command: sudo nc -lvnp 443

5.Start a netcat listener to receive the shell



6.Navigate to the shell in the browser and receive a connection

add in: cat /var/www/flag.txt



Here's the flag!

Q5: What is the flag in /var/www/flag.txt?

THM{MGU3Y2UyMGUwNjExYTY4NTAxOWJhMzhh}

Thought Process/Methodology:

First, having accessed the target machine, we were shown a not signed in page. We added "/?id=ODIzODI5MTNiYmYw" into the link "http://10.10.78.21/" and we were shown a christmas tree page with upload point. Second, we saved the php reverse shell file and edited it using text editor. We changed the ip and the port in the file and saved it. After that, we change the file name to .jpg.php as the file only accepted .jpg .jpeg and .png. Third, we submit the php reverse shell file to bypass the filter. Forth, we opened the php reverse file in /uploads/. Then, we opened terminal and started a listener for the uploaded reverse shell by using the command: sudo nc -lvnp 443. After navigating to the shell in the browser and received a connection, we cat the flag.txt and then it showed the flag.

Day 3: [Web Exploitation] Christmas Chaos

Tools used: Kali Linux, Firefox, Burp Suite Community Edition

Solution/walkthrough:

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 **I**nternet **o**f **T**hings (**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.

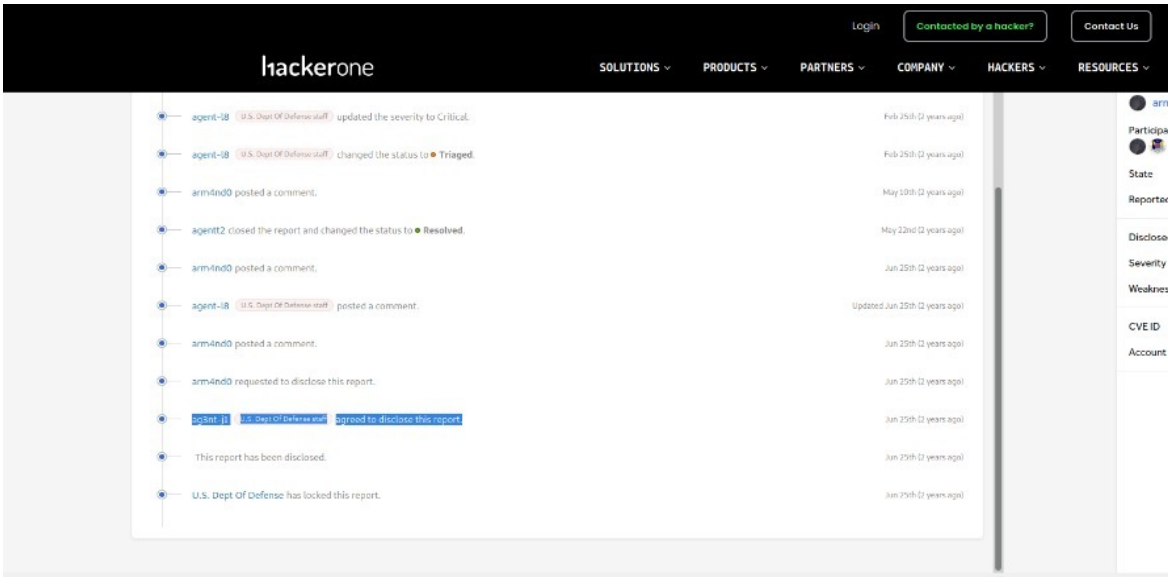
In fact, companies such as Starbucks and the US Department of Defense have been victim to leaving services running with default credentials, and bug hunters have been rewarded for reporting these very simple issues responsibly (Starbucks paid $250 for the reported issue):

Q1: What is the name of the botnet mentioned in the text that was reported in 2018?

Mirai

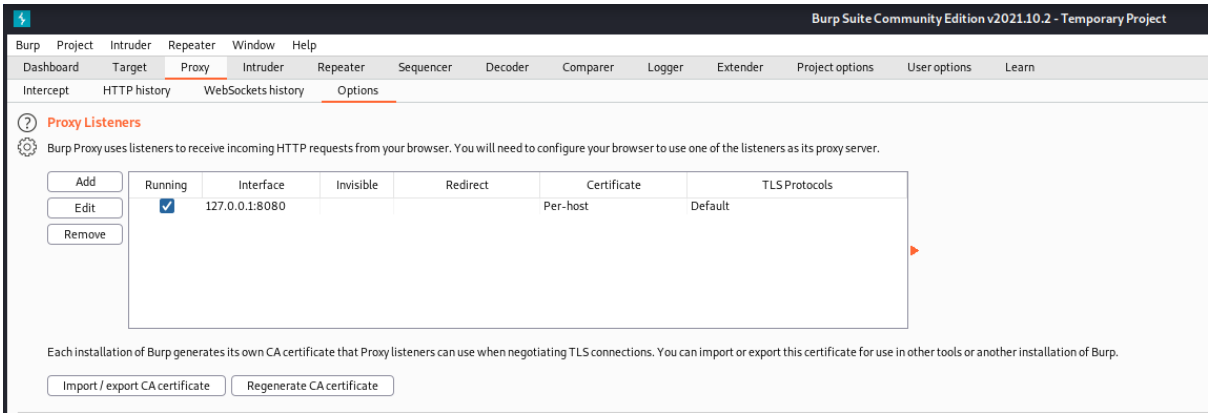
Q2: How much did Starbucks pay in USD for reporting default credentials according to the text?

$250



Q3: Read the report from Hackerone ID:804548 - who was the agent assigned from the Dept of Defense that disclosed the report on Jun 25th?

ag3nt-j1

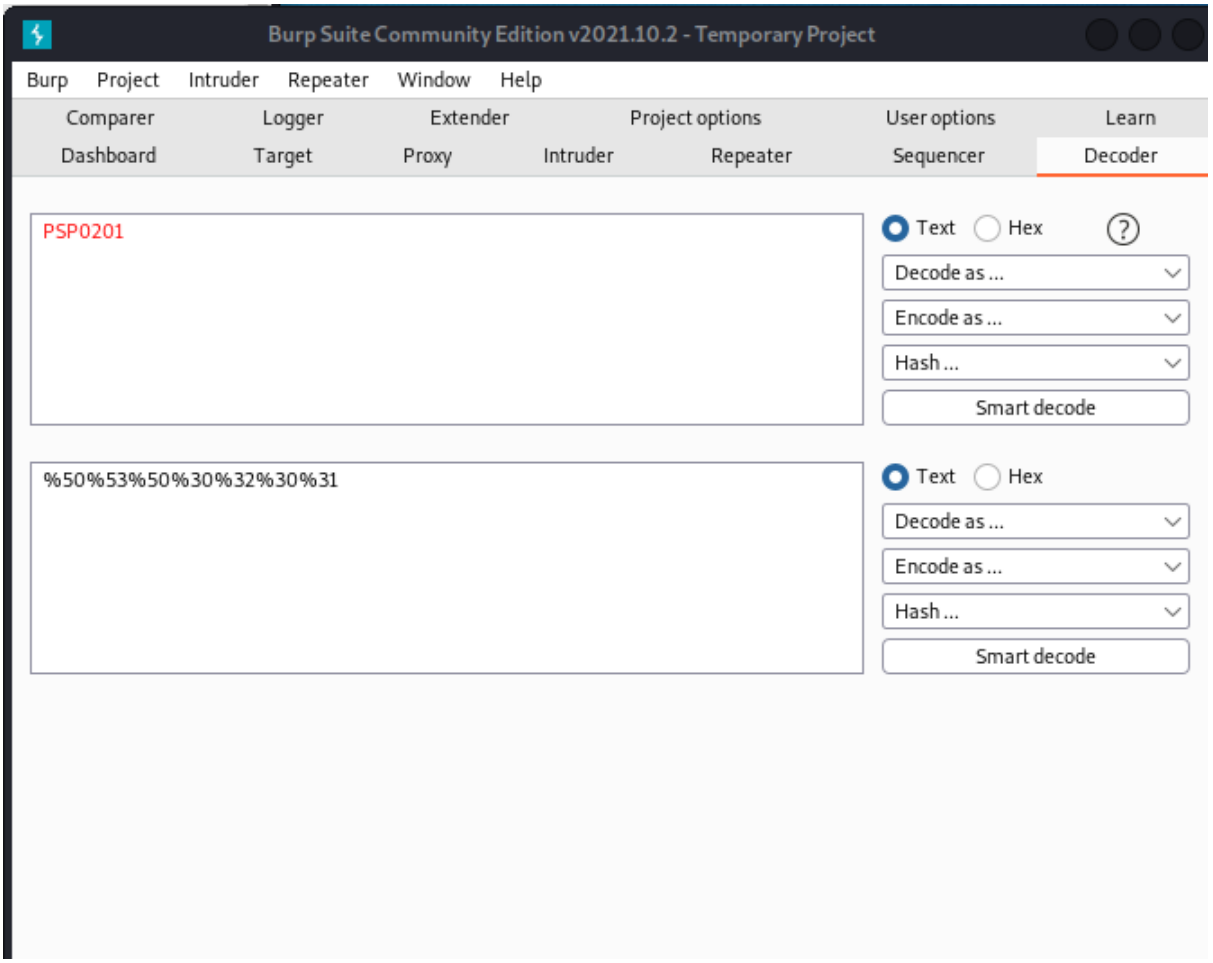


Q4: Examine the options on FoxyProxy on Burp. What is the port number for Burp?

8080

Q5: Examine the options on FoxyProxy on Burp. What is the proxy type?

HTTP



Q6: Experiment with decoder on Burp. What is the URL encoding for "PSP0201"?

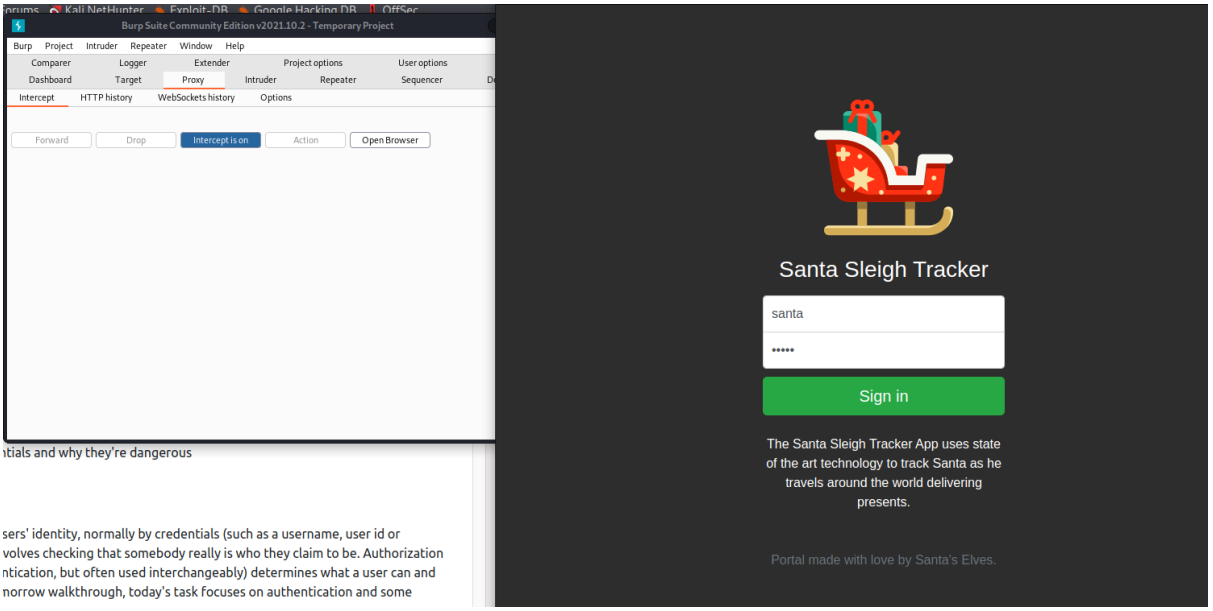
%50%53%50%30%32%30%31

- **Sniper** – This uses a single set of payloads. It targets each payload position in turn, and places each payload into that position in turn. Positions that are not targeted for a given request are not affected – the position markers are removed and any enclosed text that appears between them in the template remains unchanged. This attack type is useful for fuzzing a number of request parameters individually for common vulnerabilities. The total number of requests generated in the attack is the product of the number of positions and the number of payloads in the payload set.
- **Battering ram** – This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once. This attack type is useful where an attack requires the same input to be inserted in multiple places within the request (e.g. a username within a Cookie and a body parameter). The total number of requests generated in the attack is the number of payloads in the payload set.
- **Pitchfork** – 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 all payload sets simultaneously, and places one payload into each defined position. In other words, the first request will place the first payload from payload set 1 into position 1 and the first payload from payload set 2 into position 2; the second request will place the second payload from payload set 1 into position 1 and the second payload from payload set 2 into position 2, etc. This attack type is useful where an attack requires different but related input to be inserted in multiple places within the request (e.g. a username in one parameter, and a known ID number corresponding to that username in another parameter). The total number of requests generated in the attack is the number of payloads in the smallest payload set.
- **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.

Q7: Look at the list of attack type options on intruder. Which of the following options matches the one in the description?

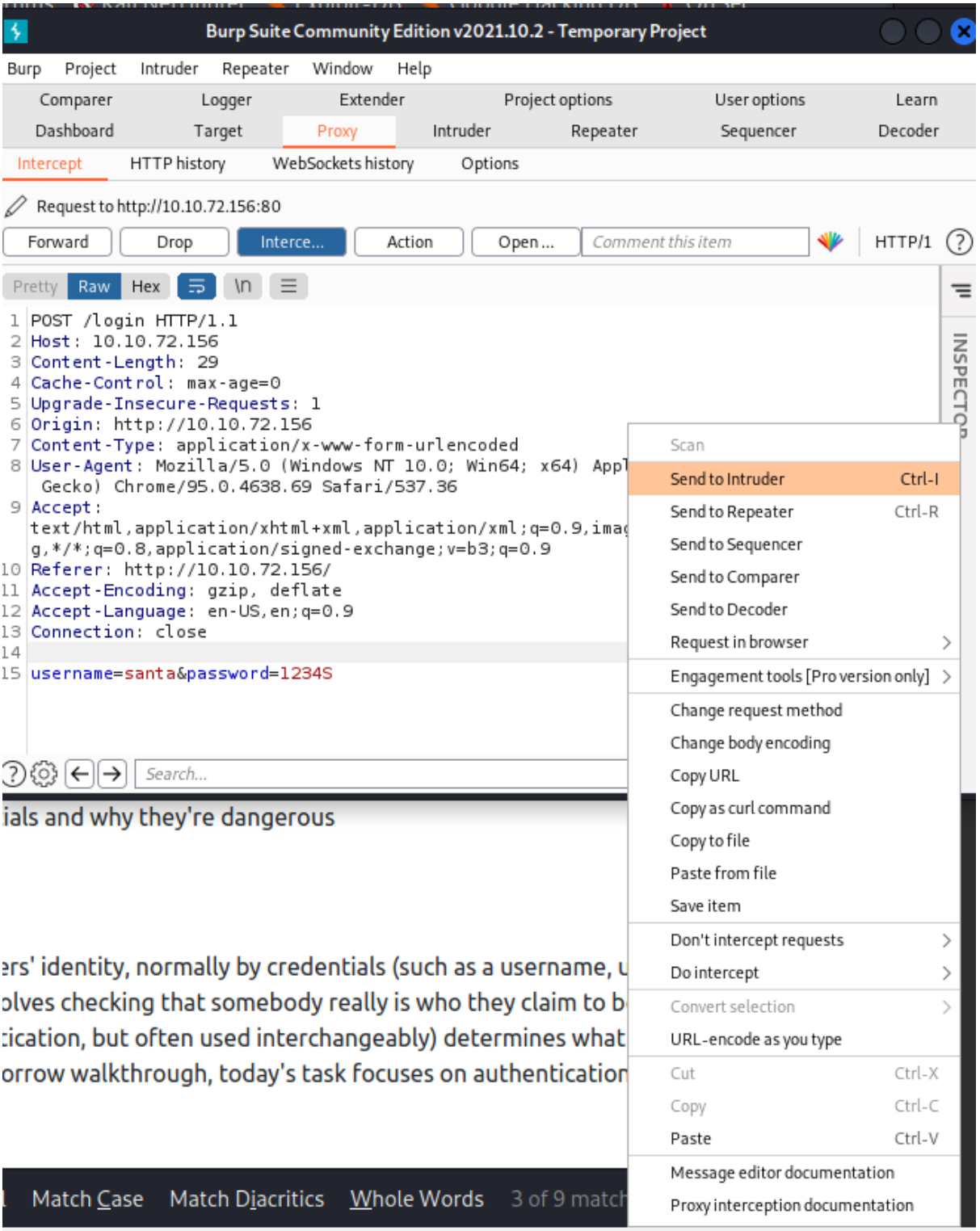
Uses multiple payload sets. Different payload for each defined position up to maximum 20. Iterates through each payload set in turn, so all permutations of payload combinations are tested.

Cluster bomb

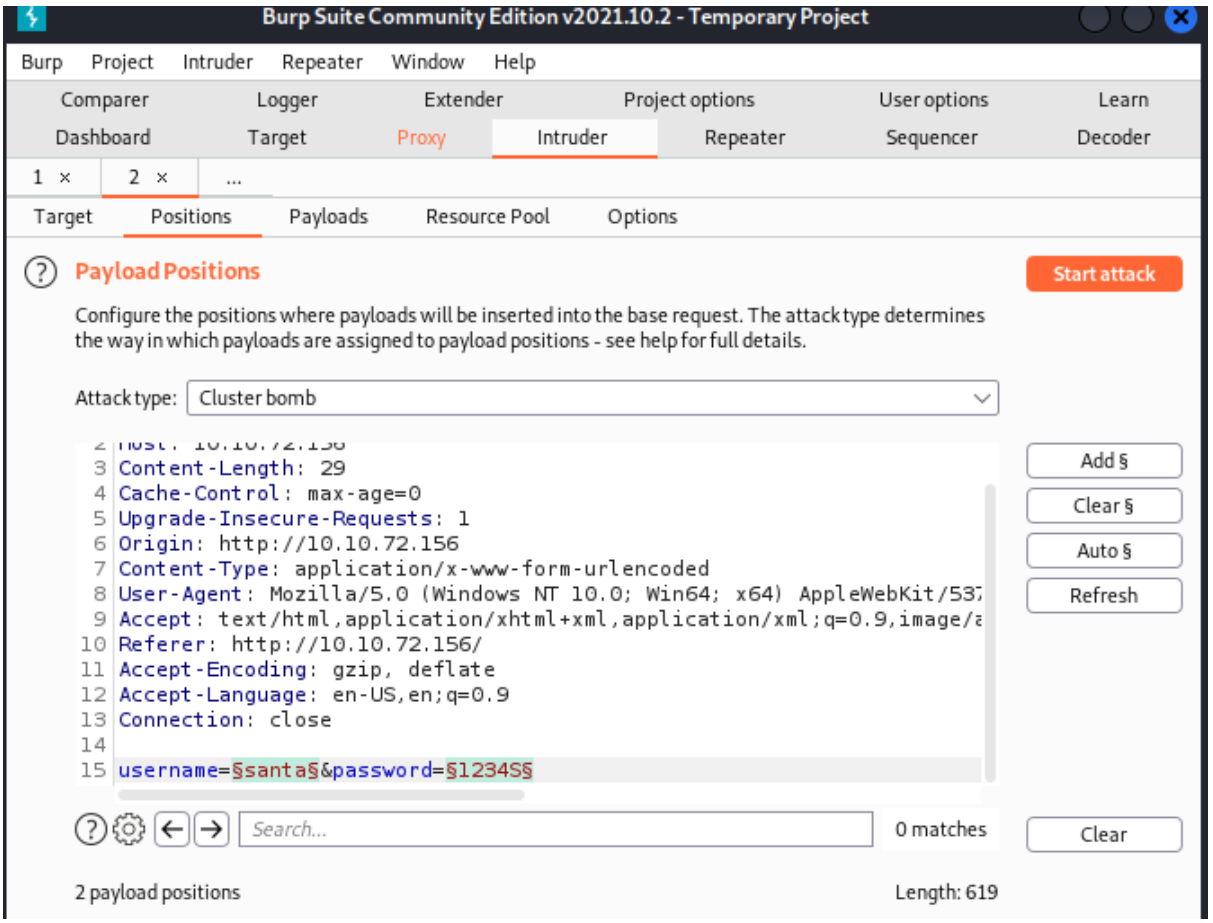


open burp suite> proxy> open browser> search the link “http://10.10.72.156/”

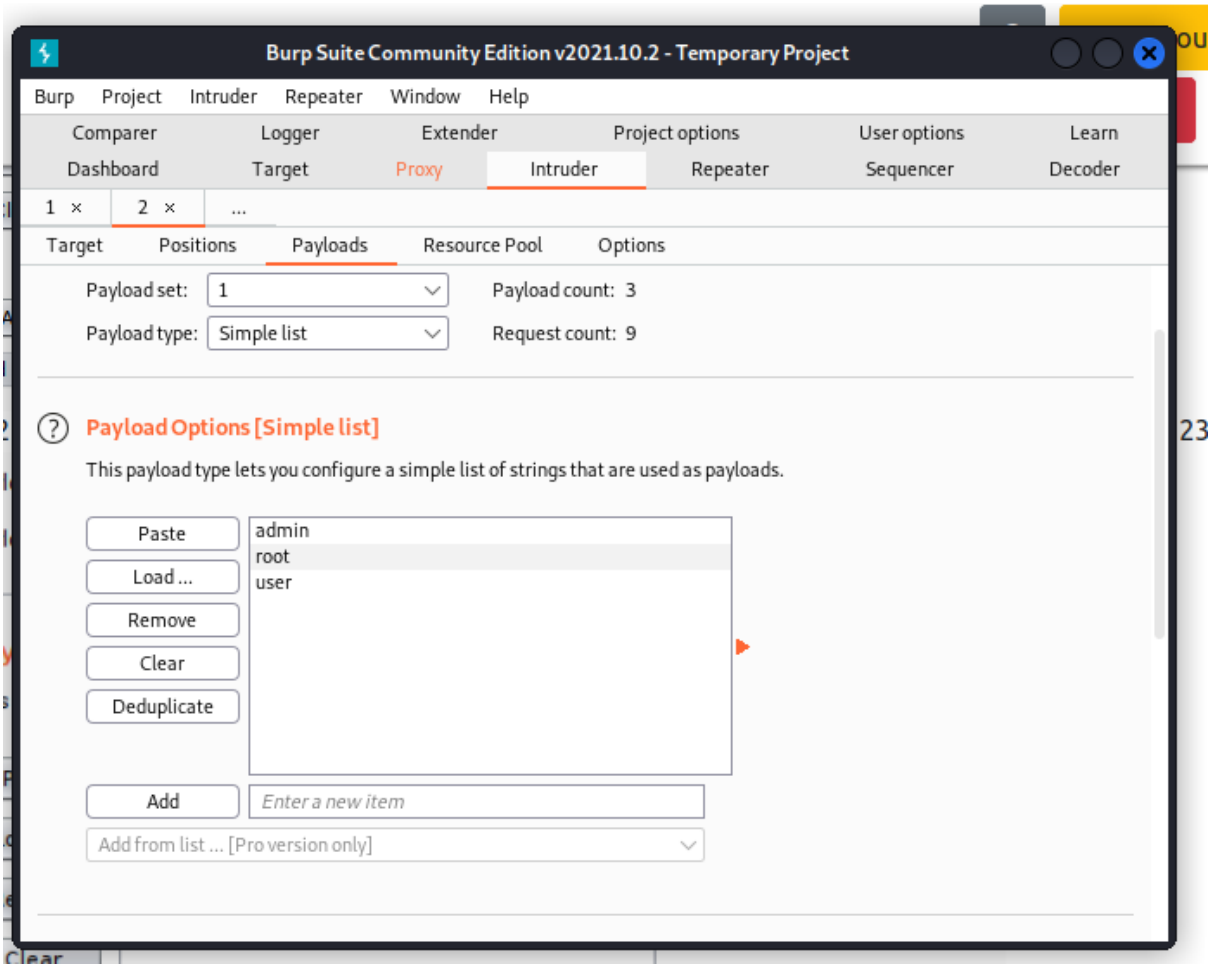
key whatever username and password then press “login”



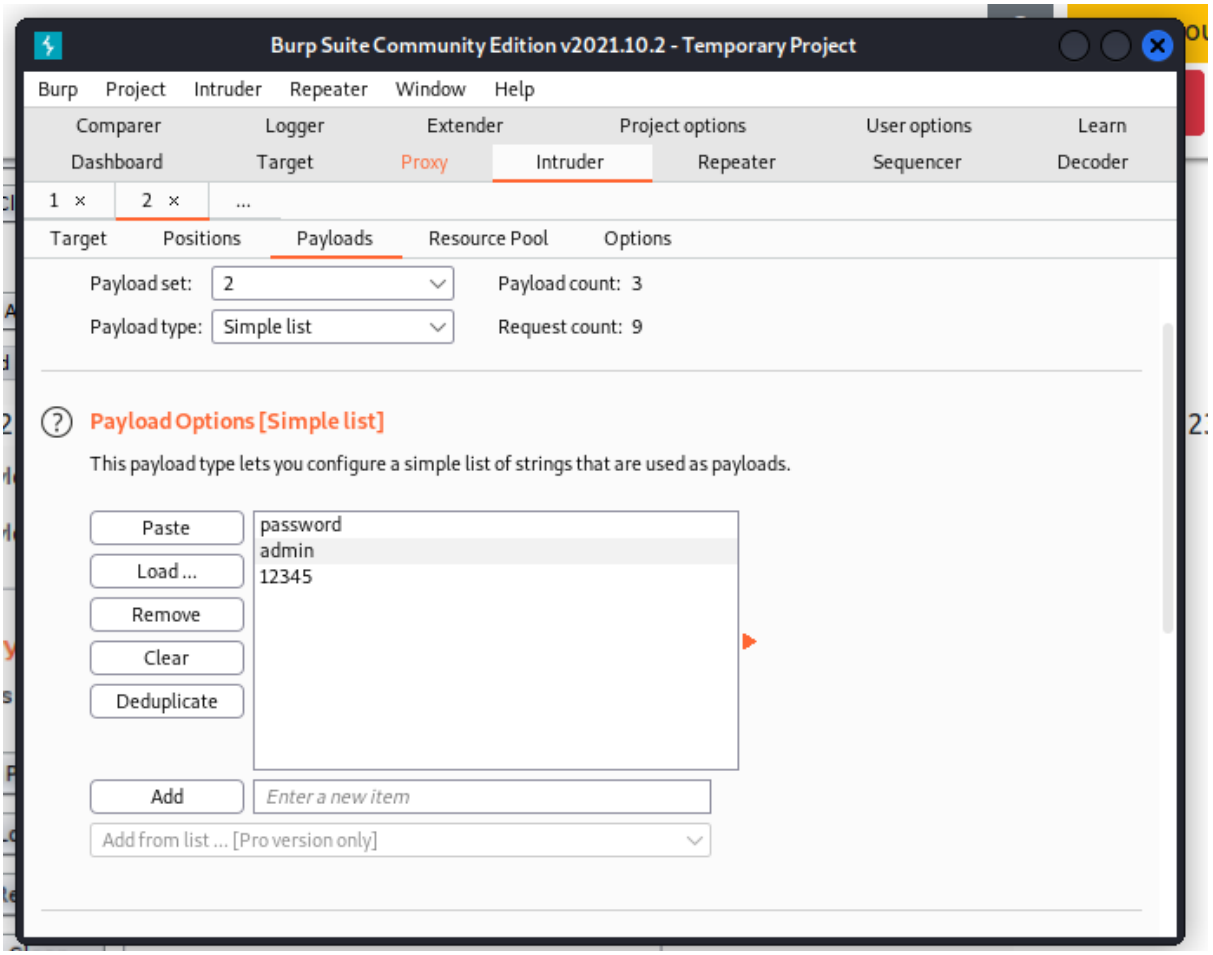
Send to intruder



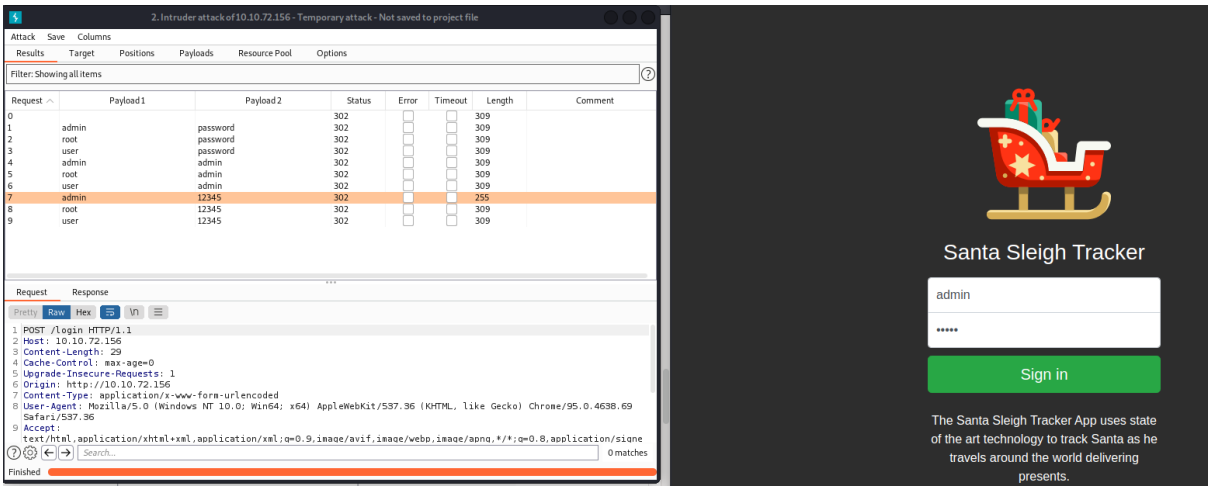
change the attack type to cluster bomb



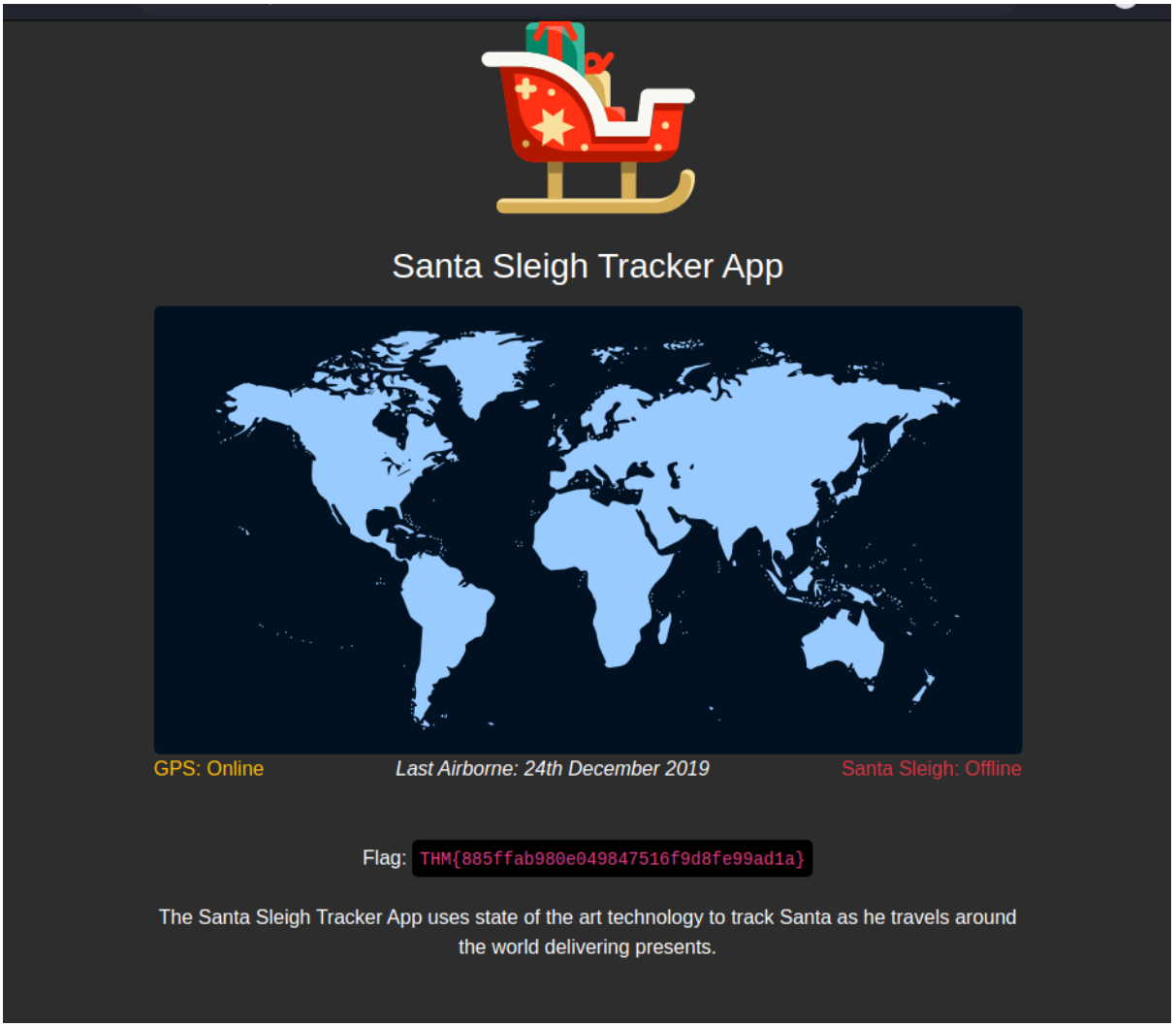
To the “Payloads” tab. For payload set 1, it’s username, add “admin”,”root”, “user” in the payload options.



For payload set 2, it's password, add "password","admin", "12345" in the payload options. Then press "start attack".



Find the difference length and status, then key in the username ("admin") and password ("12345") into the website



Back to Burp Suite> Proxy>Forward, then there's the flag in the website!

Q8: What is the flag?

THM{885ffab980e049847516f9d8fe99ad1a}

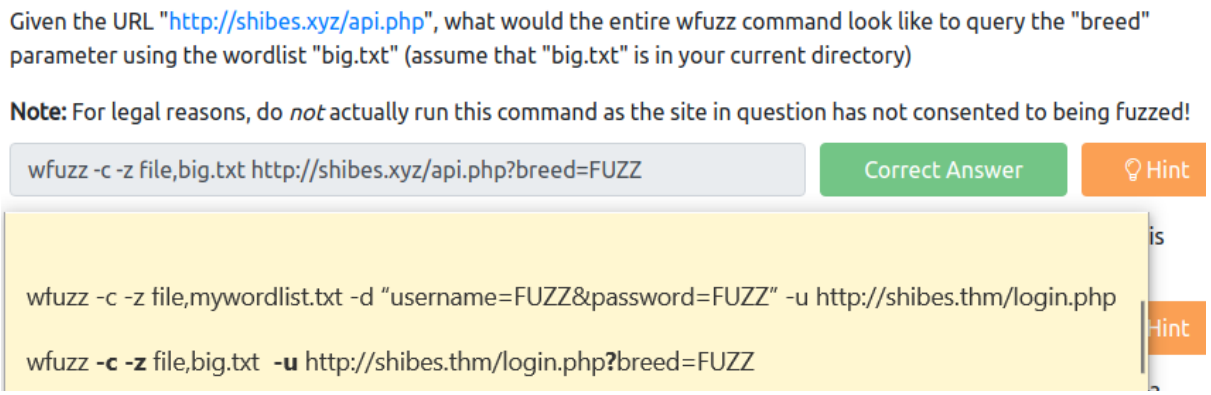
Thought Process/Methodology:

First, having accessed the target machine. Then we opened burp suite and went to proxy and then opened browser and searched the link "http://10.10.72.156/". Second, we had key whatever username and password then pressed "login". Third, we opened the burpsuite and the captured request was showing up in Proxy tab. We right-click on it and "send to intruder". Forth, we go to Intruder tab, we saw our request. At Payload Positions, we added the username and password values as positions. We changed the attack type to "Cluster Bomb". Fifth, we clicked the "Payloads" tab and added a few command default username in Payload set 1 and default passwords in Payload set 2. Sixth, we clicked the "Start Attack" button, this loop through each position list in every combination. We sorted by the "Length" or "Status" to identify a successful login. Finally, we used the username and password to sign in the websitewhich in turn showed the flag.

Day 4: [Web Exploitation] Santa's watching

Tools used: Kali Linux, Firefox, Burp Suite Community Edition

Solution/walkthrough:



Change the txt file name and “breed=FUZZ” to query the "breed" parameter using the wordlist "big.txt"

-c :Shows the output in color

-z :Specifies what will replace FUZZ in the request. For example -z file,big.txt. We're telling wfuzz to look for files by replacing "FUZZ" with the words within "big.txt"

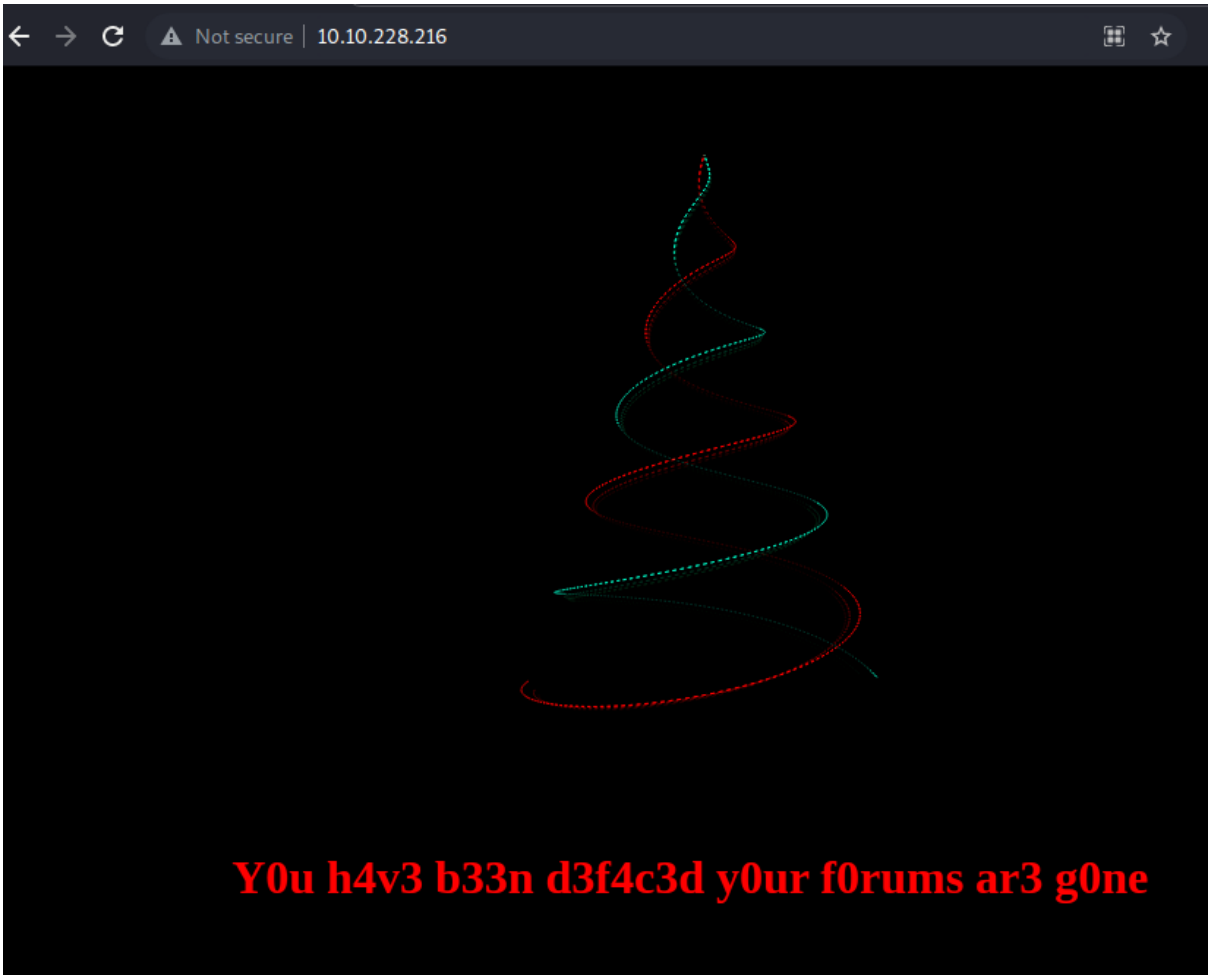
-u :used to specify which url to enumerate

? **:**used to specify that a GET parameter is forthcoming.

Q1: 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)

Select the proper words in the proper place of the command: [a] -c -z file,[b] http://[c].xyz/api.[d]?[e]=FUZZ

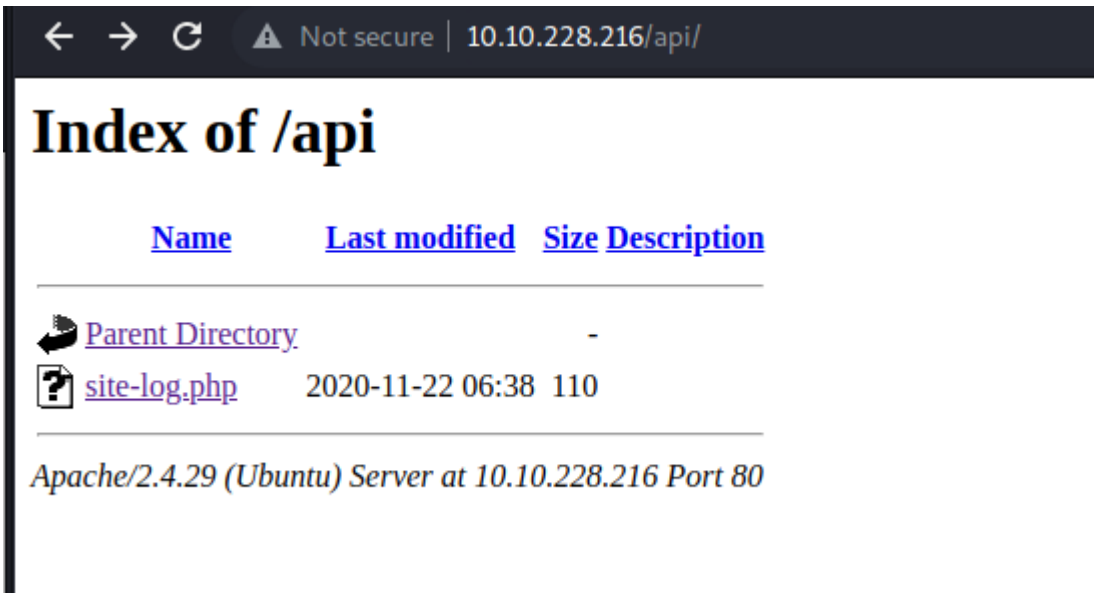
wfuzz -c -z file,big.txt http://shibes.xyz/api.php?breed=FUZZ



open the link: http://10.10.228.216/

```
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.100.98
[+] 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
[+] Timeout:        10s
===============================================================
2022/06/23 04:01:29 Starting gobuster
===============================================================
/.htpasswd (Status: 403)
/.htpasswd.php (Status: 403)
/.htaccess (Status: 403)
/.htaccess.php (Status: 403)
/LICENSE (Status: 200)
/api (Status: 301)
/server-status (Status: 403)
===============================================================
2022/06/23 04:03:39 Finished
===============================================================
root@ip-10-10-45-237:~#
```

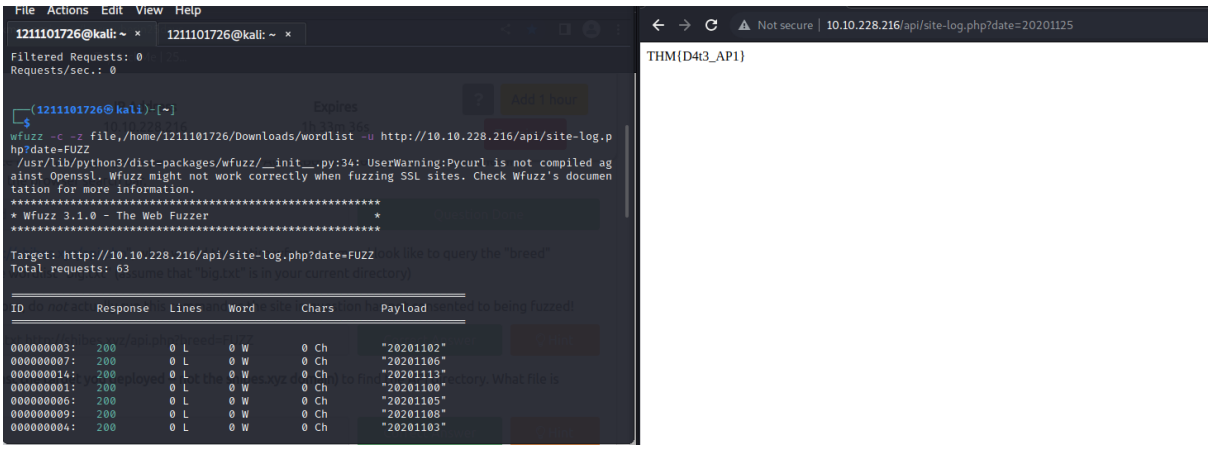
Use gobuster to find the API directory. (/api)

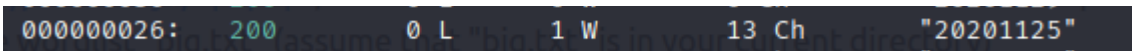


Add in the **/api** into the link and open the site-log.php.

Q2: Use GoBuster (against the target you deployed -- not the shibes.xyz domain) to find the API directory. What file is there?

site-log.php





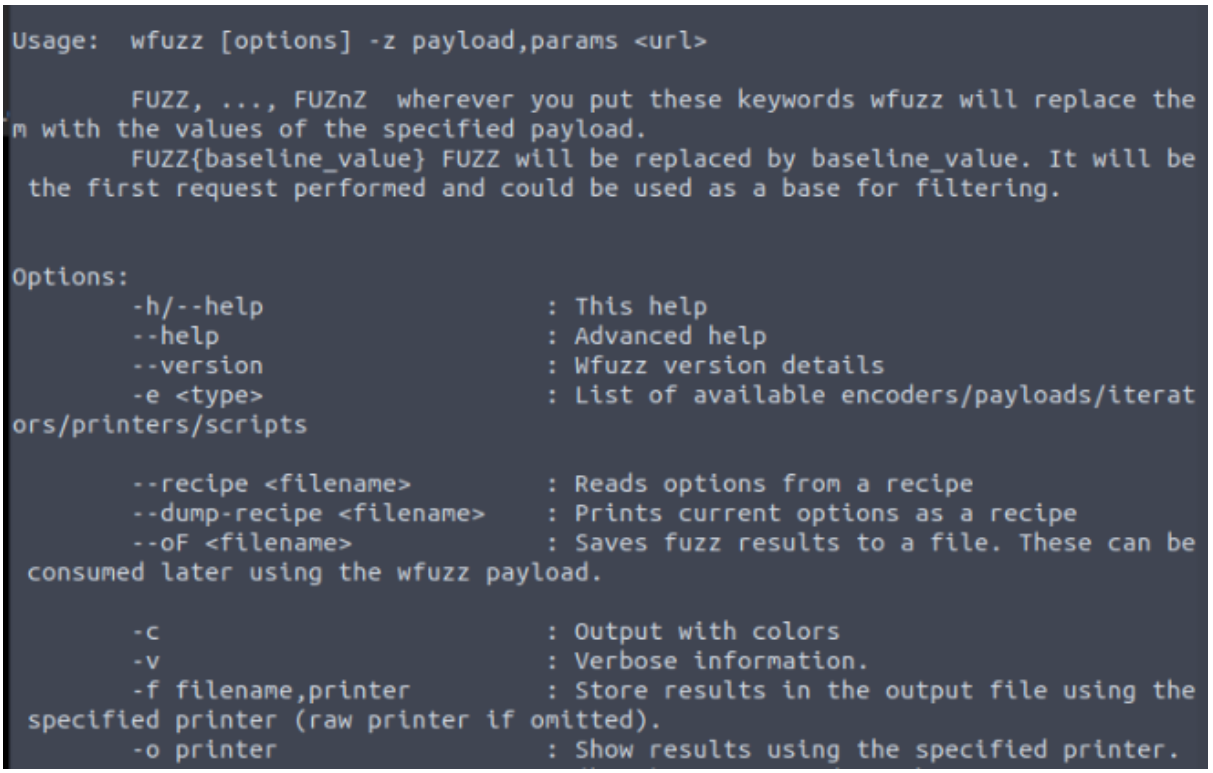
Open terminal and use wfuzz command: wfuzz -c -z file,/home/1211101726/Downloads/wordlist -u http://IP address/api/site-log.php?date=FUZZ

to find the date which has different characters, then add the date into the link.

And there's is the flag!

Q3: Fuzz the date parameter on the file you found in the API directory. What is the flag displayed in the correct post?

THM{D4t3_AP1}



Q4: Look at wfuzz's help file. What does the -f parameter store results to?

filename

printer

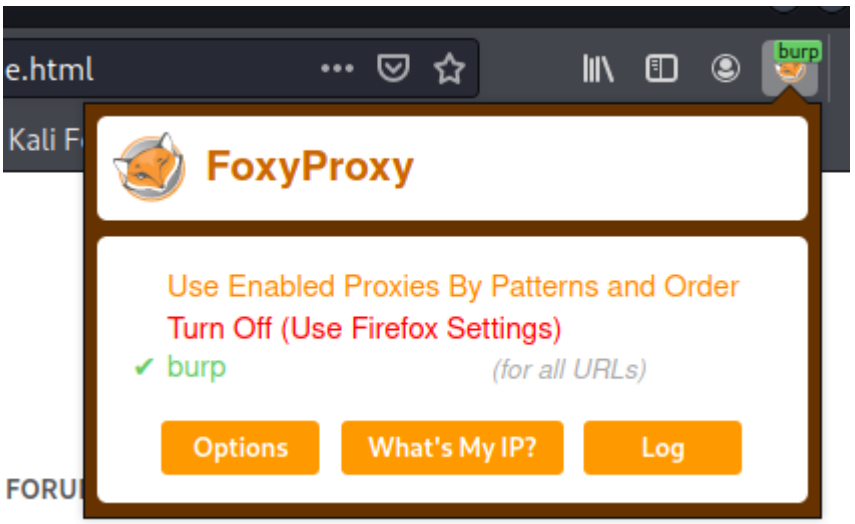
Thought Process/Methodology:

First, having accessed the target machine, we were shown an animation christmas tree page. We used **gobuster** to find the API directory then added in the **/api** into the link and a file, **site-log.php** is shown. We opened site-log.phph. After that, we opened terminal and used the command **:wfuzz -c -z file,/home/1211101726/Downloads/wordlist -u http://IP_address/api/site-log.php?date=FUZZ** to find the date which has different characters, then added the date into the link(wfuzz -c -z file,/home/1211101726/Downloads/wordlist -u http://IP_address/api/site-log.php?date=**20201125**). The flag appeared.

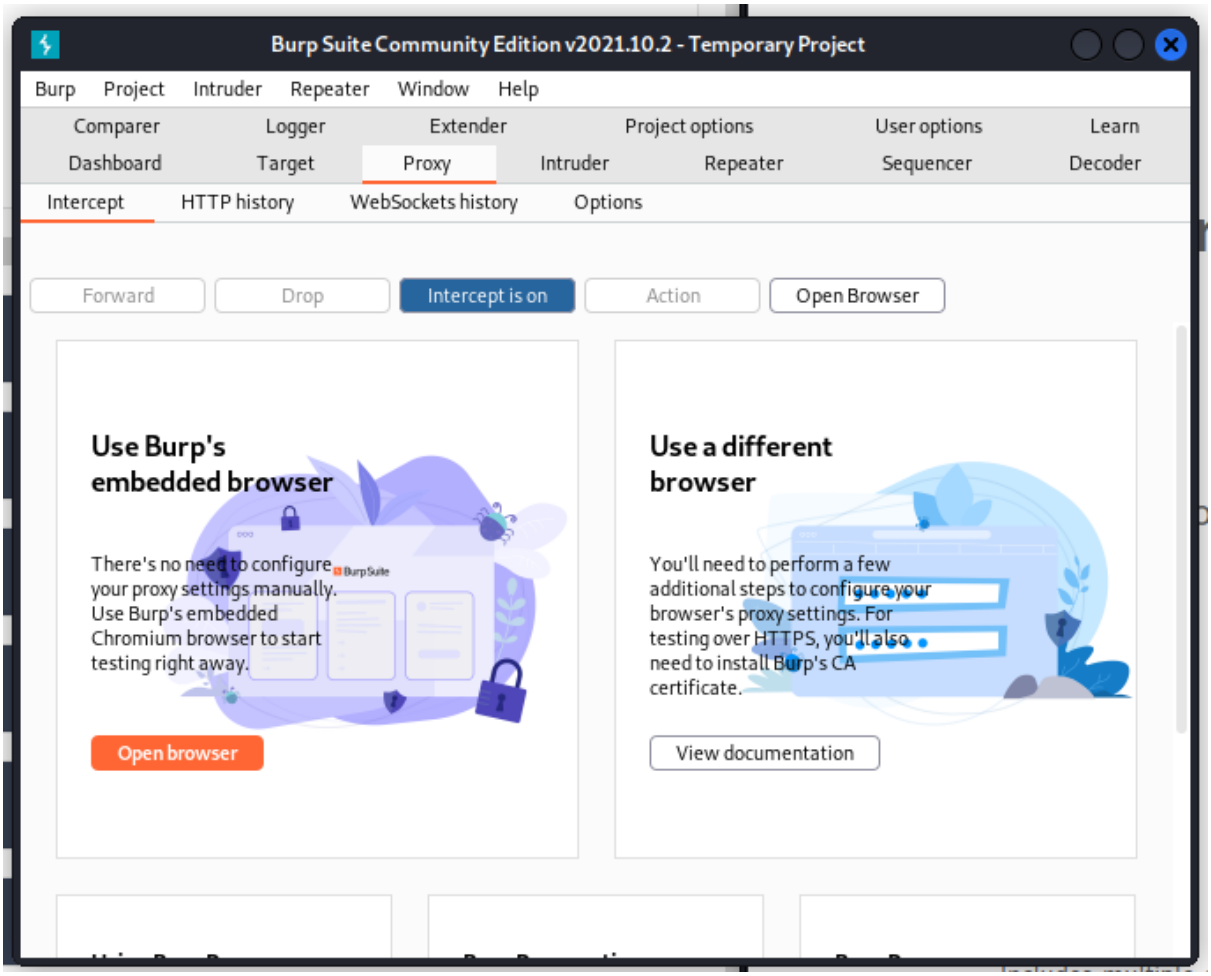
Day 5: [Web Exploitation] Someone stole Santa's gift list!

Tools used: Kali Linux, Firefox, FoxyProxy, Burp Suite Community Edition, Terminal

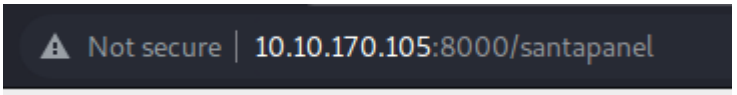
Solution/walkthrough:



Enable FoxyProxy in Firefox.



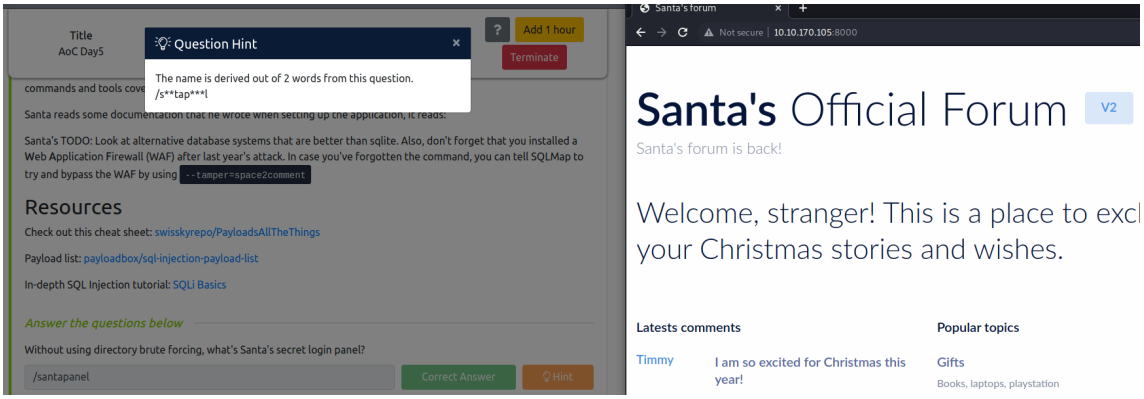
Open Burp Suite Community Edition>Proxy tab> make sure Intercept is **ON.**



Get into the link:

Q1: What is the default port number for SQL Server running on TCP?

8000

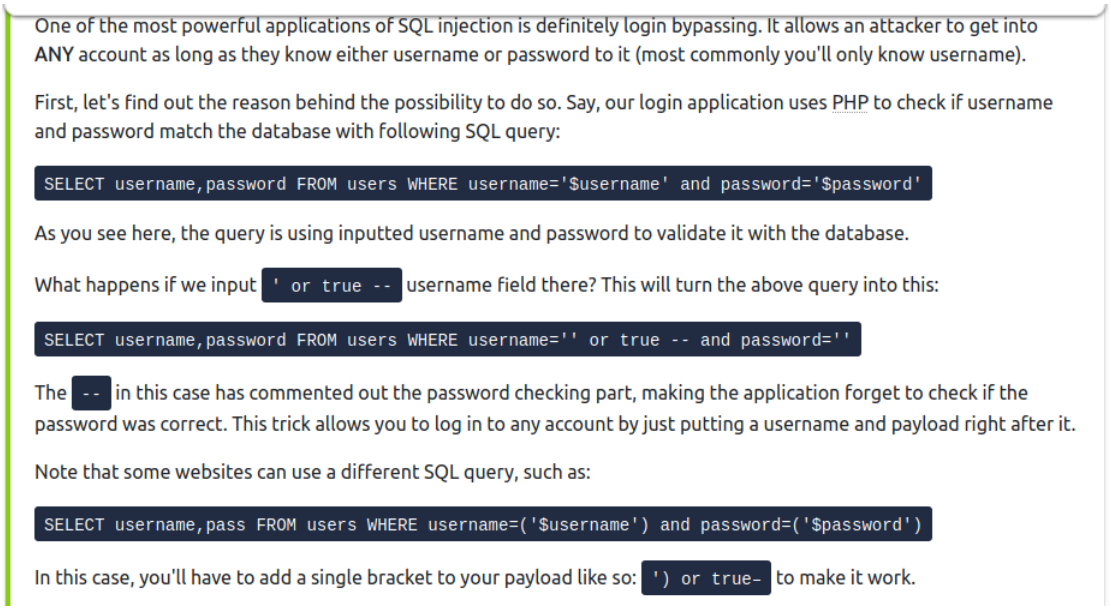


Get into the link: http://10.10.170.105:8000/

then questions 1, from the hint, we can guess the Santa's secret login panel, which is **/santapanel**.

Q2: Without using directory brute forcing, what's Santa's secret login panel?

/santapanel



One of the most powerful applications of SQL injection is definitely login bypassing. It allows an attacker to get into **ANY** account as long as they know either username or password to it (most commonly you'll only know username).

First, let's find out the reason behind the possibility to do so. Say, our login application uses PHP to check if username and password match the database with following SQL query:

```
SELECT username,password FROM users WHERE username='$username' and password='$password'
```

As you see here, the query is using inputted username and password to validate it with the database.

What happens if we input `' or true --` username field there? This will turn the above query into this:

```
SELECT username,password FROM users WHERE username='' or true -- and password=''
```

The `--` in this case has commented out the password checking part, making the application forget to check if the password was correct. This trick allows you to log in to any account by just putting a username and payload right after it.

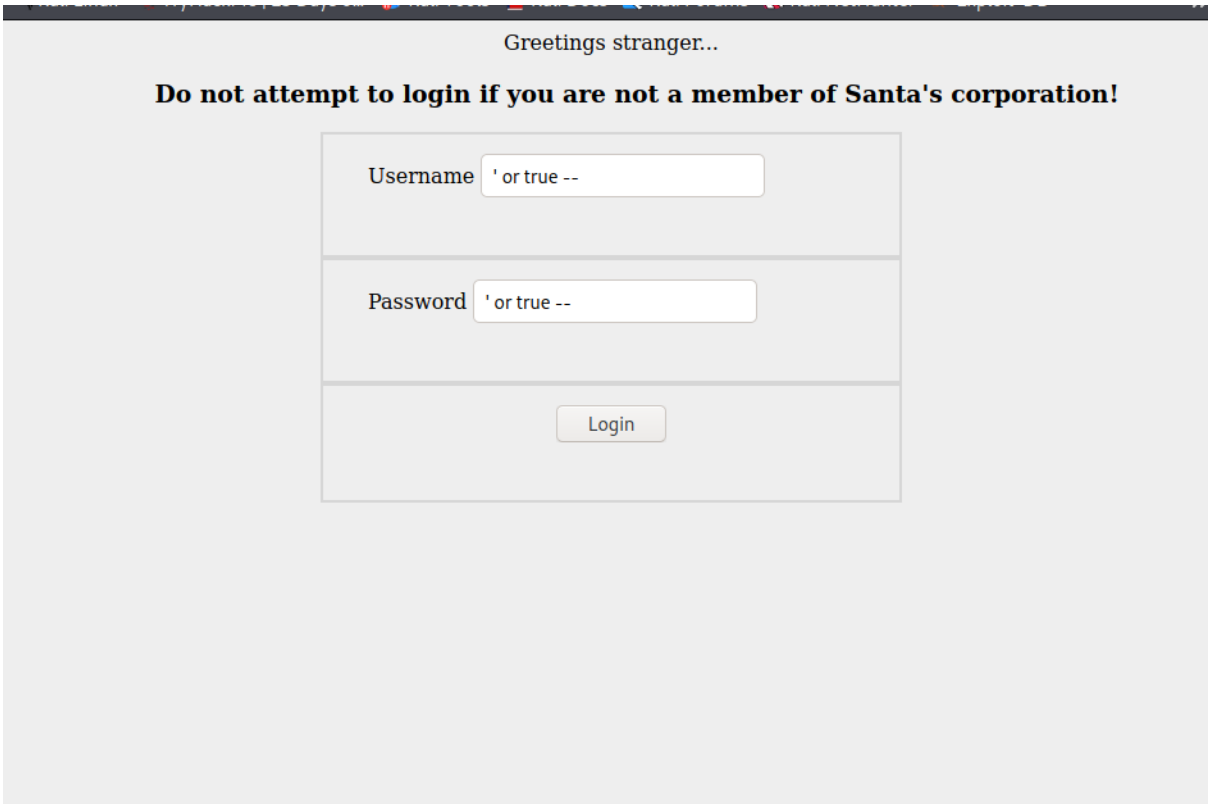
Note that some websites can use a different SQL query, such as:

```
SELECT username,pass FROM users WHERE username=('$username') and password=('$password')
```

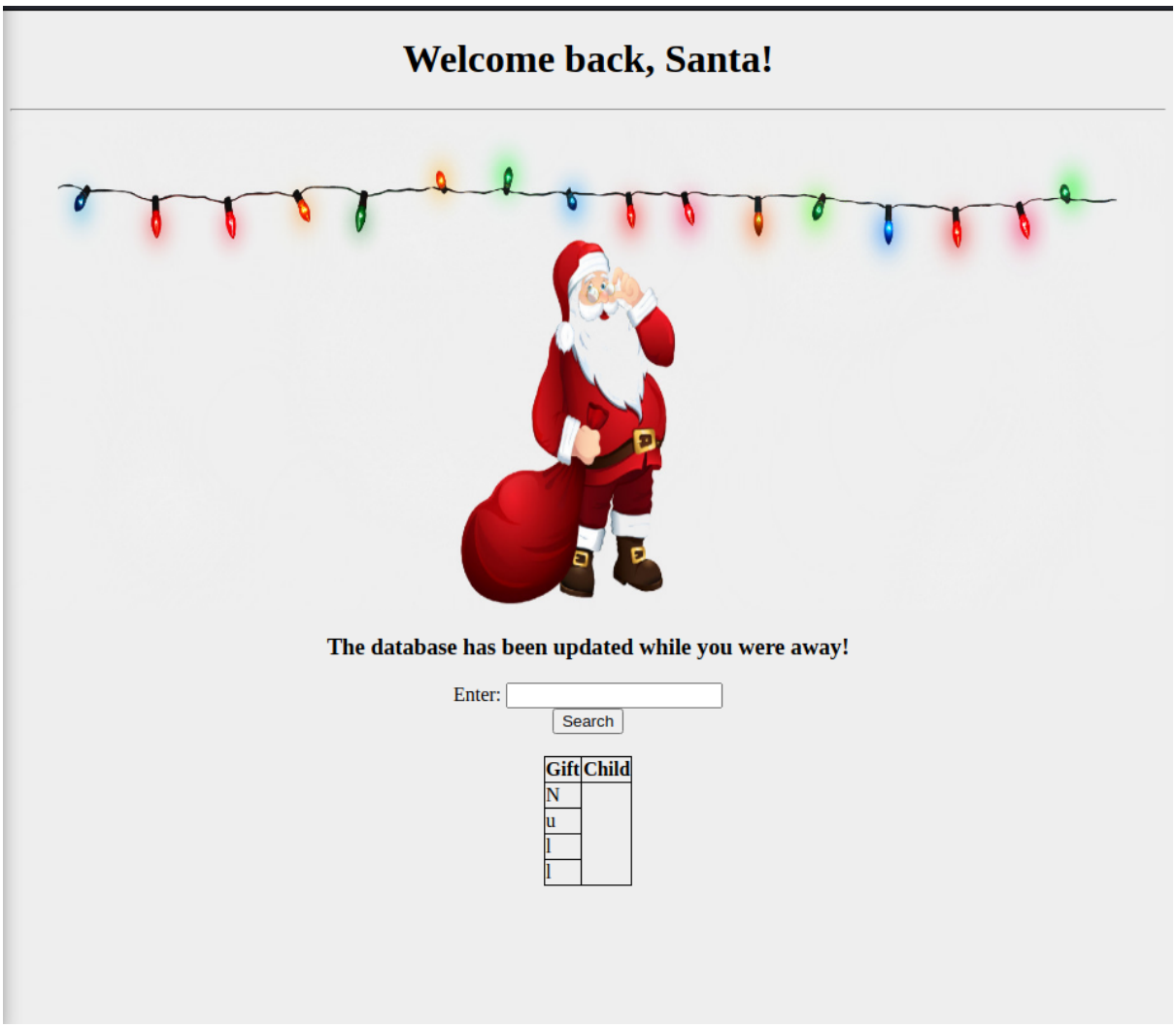
In this case, you'll have to add a single bracket to your payload like so: `') or true--` to make it work.

Q3: What is the database used from the hint in Santa's TODO list?

sqlite



To bypass the login using SQLi, username & password = **' or true –**

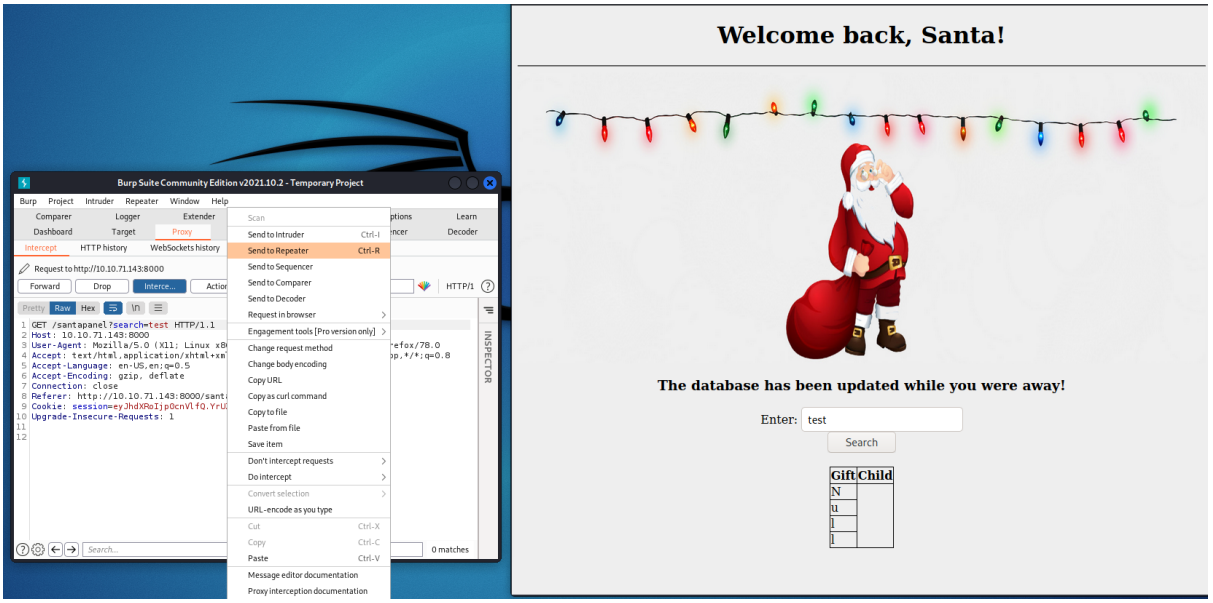


We are logged in successfully when we see this page.

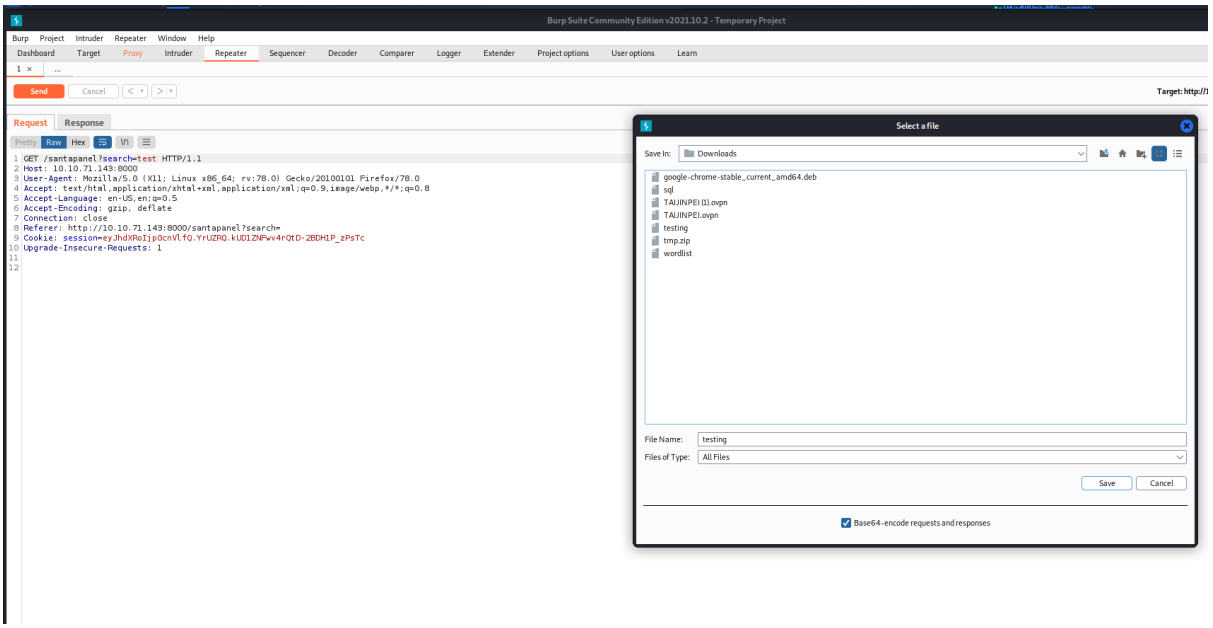
Santa's TODO: Look at alternative database systems that are better than sqlite. Also, don't forget that you installed a **W**eb **A**pplication **F**irewall (WAF) after last year's attack. In case you've forgotten the command, you can tell SQLMap to try and bypass the WAF by using `--tamper=space2comment`

Q3: What is the database used from the hint in Santa's TODO list?

sqlite



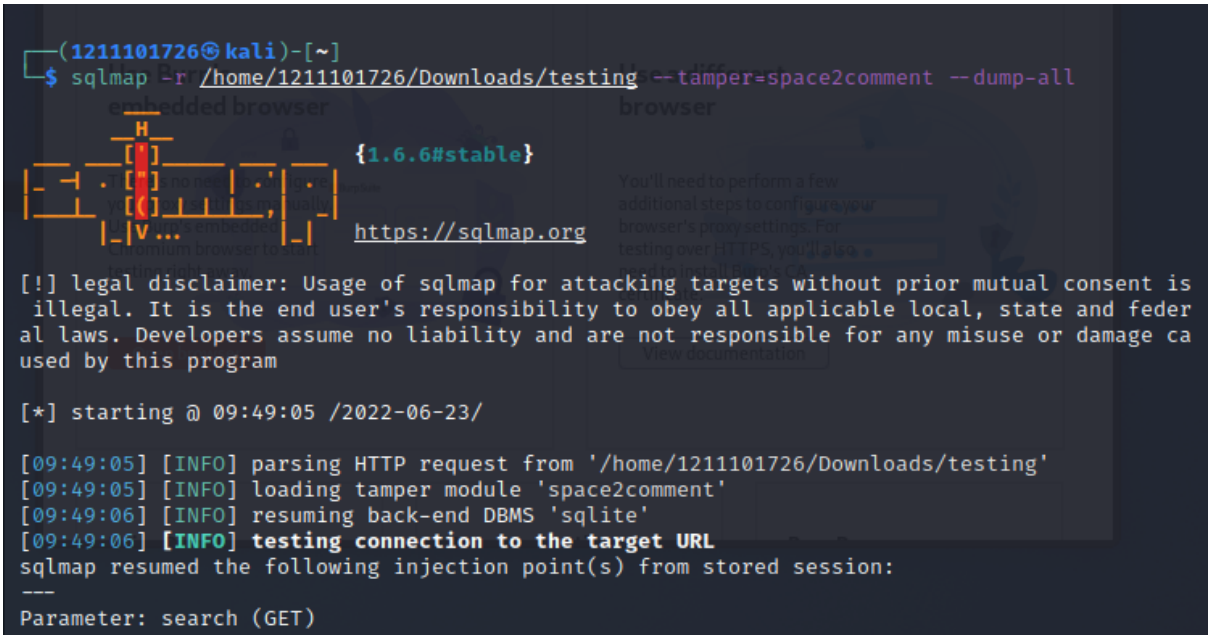
Enter a word then press “Search”, then send the request from Burp Suite Proxy tab to Repeater.



Go to Repeater tab, then right click the request, press “Save item”. Now we can use this request in SQLMap.

SQLMap command:

Command	
--url	Provide URL for the attack
--dbms	Tell SQLMap the type of database that is running
--dump	Dump the data within the database that the application uses
--dump-all	Dump the ENTIRE database
--batch	SQLMap will run automatically and won't ask for user input

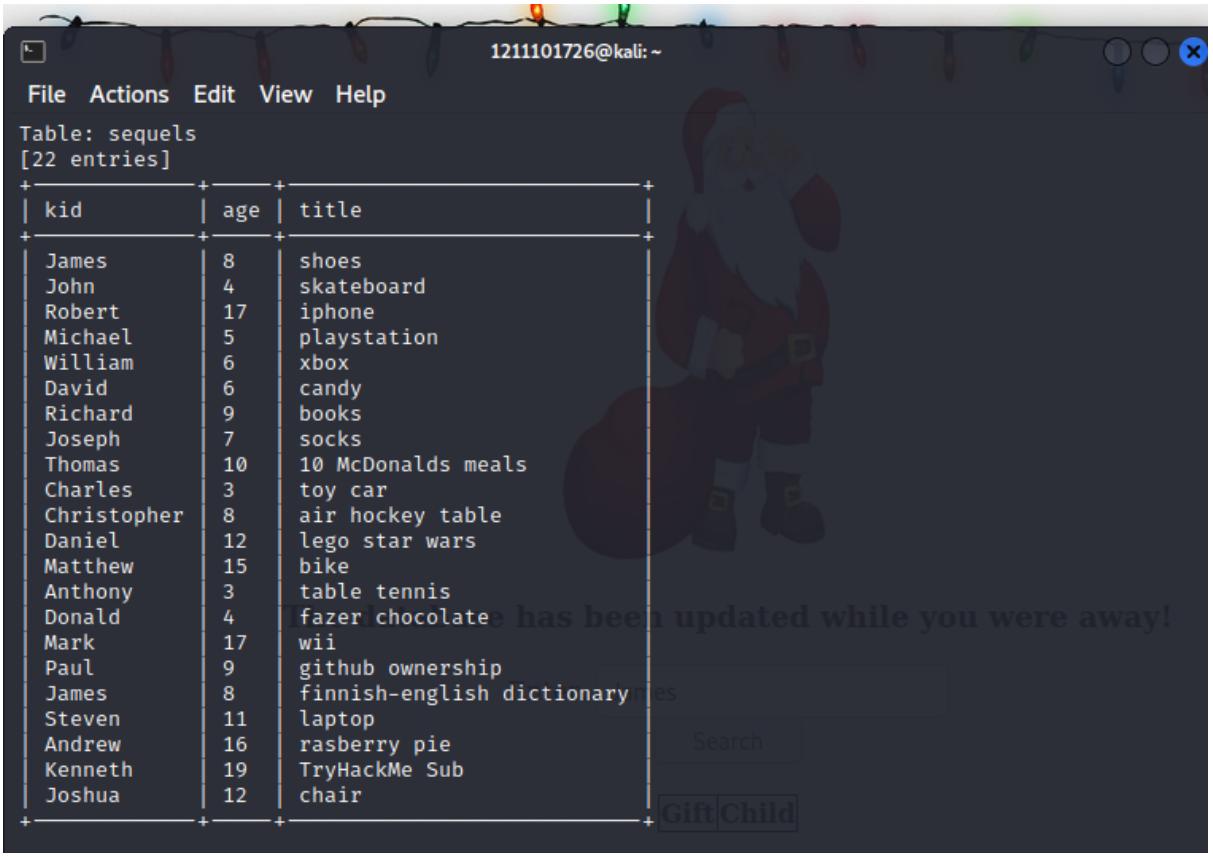


Open terminal and run sqlmap command.

sqlmap -r filename : request and exploit the database.

--tamper=space2comment : tell SQLMap to try and bypass the WAF

--dump-all : Dump the ENTIRE database



Now we can get hidden information.

Q4: How many entries are there in the gift database?

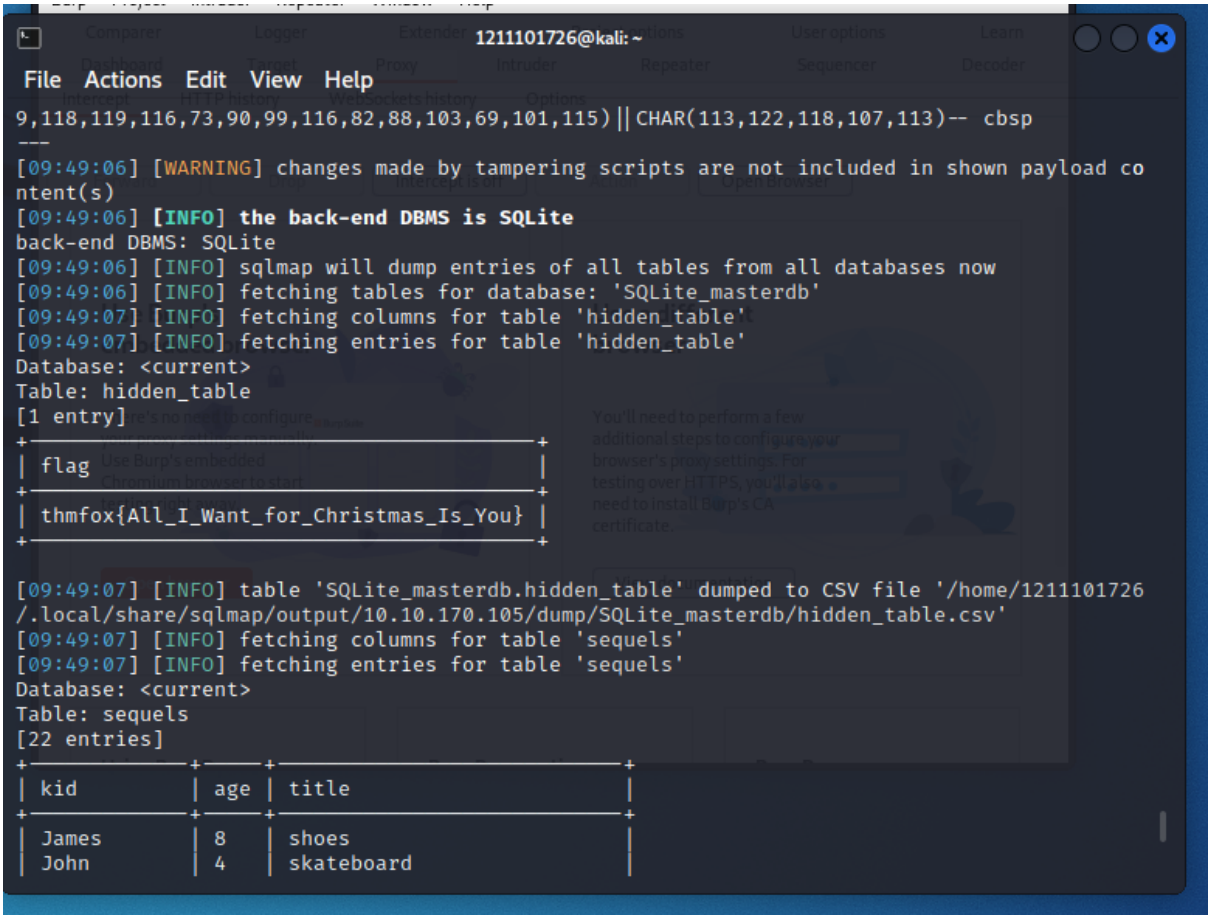
22

Q5: What is James' age?

8

Q6: What did Paul ask for?

github ownership



Q7: What is the flag?

thmfox{All_I_Want_for_Christmas_Is_You}

Q8: What is admin's password?

EhCNSWzzFP6sc7gB

Thought Process/Methodology:

First, we enabled FoxyProxy in Firefox. Second, we opened Burp Suite Community Edition, went to Proxy tab and made sure Intercept was ON**.** Third, we having accessed the target machine with the port: 8000 and we were shown a Santa's Official Forum. Then from the hint of question 1, we guessed the Santa's secret login panel, which is /santapanel and added into the link. A login page were shown. To bypass the login using SQLi, we inputted **' or true –** as the username and password field. After logging in, we searched "test" and the request appeared in Burp Suite Proxy tab, then we sent the request to repeater. We went to the Repeater tab, then right click the request, pressed "Save item". Now we can use this request in SQLMap. We opened terminal and run sqlmap command to get hidden information and the flags.