Black box penetrating attack (Jangow01)

23/24, Systems and Cyber Security

Group: 1

Student ID: 4008609

15th April 2024

London South Bank University

Department of Computer Science, Engineering

Table of Contents

Abstract

The report executes a penetration test that uncovers two significant security issues. One is an insecure HTTP service which allows retrieval of files containing login credentials which were used to gain unauthorised access. Secondly, a successful Berkeley Packet Filter (BPF) exploit enabled memory manipulation to escalate privileges to root. These findings demonstrate the critical need for improved security protocols. It also includes insights gained from OpanVas, Nmap and Nessus scans.

Acknowledgements

I would like to give thanks to George Bamfo, Ioannis and Naimul for the guidance and supervision of this project.

List of Figures

List of tables

Chapter 1 Introduction

1.1 Project Overview

This report works on performing traditional black-box penetration testing on a provided target machine to explore vulnerabilities and get a deeper understanding of mitigating and managing cybersecurity risks.

1.2 Aims and Objectives

The primary **aim** is to conduct Black-Box Penetration Testing and to execute a thorough security assessment testing on a jangow01 target system. This involves approaching the jangow01 system as an external attacker, with no prior knowledge of the internal mechanisms, to uncover potential security vulnerabilities. The ultimate aim is to gain access to a file name proof.txt within the root directory.

Objectives:

1. Perform Passive Attack Information Gathering: To collect comprehensive information about the target without direct interaction, using passive techniques to avoid detection and gather essential data for subsequent stages.

2. Perform Network Discovery: To discover and map the target network, identifying active devices, network services, and their configurations, laying the groundwork for more targeted attacks.

3. Conduct Port Scanning: To scan the target's network ports, identifying open ports and the services running on them, as well as any associated protocols and application versions. This helps in pinpointing potential entry points for exploitation.

4. Conduct Vulnerability Scanning and Analysis: To systematically scan the target system for known vulnerabilities, and analyse the scan results to prioritise vulnerabilities based on their severity and potential impact on the target system.

5. Leverage Identified Issues for Exploitation: To use the vulnerabilities uncovered during the scanning phase to attempt exploitation, aiming to gain unauthorised access or retrieve sensitive information from the target system.

6. Achieve and Demonstrate Proof of Exploitation: The final objective is to gain access to a specific file, “proof.txt”, located in the target system's root directory or to provide equivalent evidence of root access. This serves as conclusive proof of successful system compromise.

7. Optional Objectives - Bonus Tasks: Additional tasks such as pilfering, covering tracks, and backdoor creation are considered bonus objectives. These actions simulate advanced attacker techniques for maintaining access, evading detection, and ensuring persistence within the compromised system.

1.3. Machine Setup

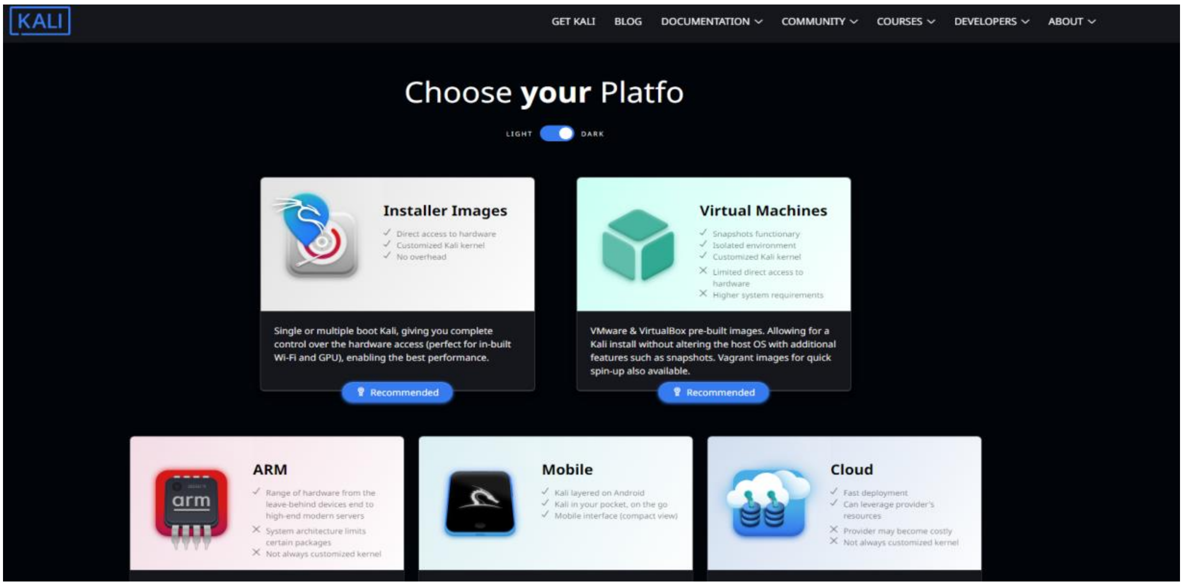


Figure 1: Kali Download

This setup involved downloading the latest Kali Linux ISO from the official (Kali Linux, 2022) website, its checksum was also verified to ensure the file's authenticity and security.

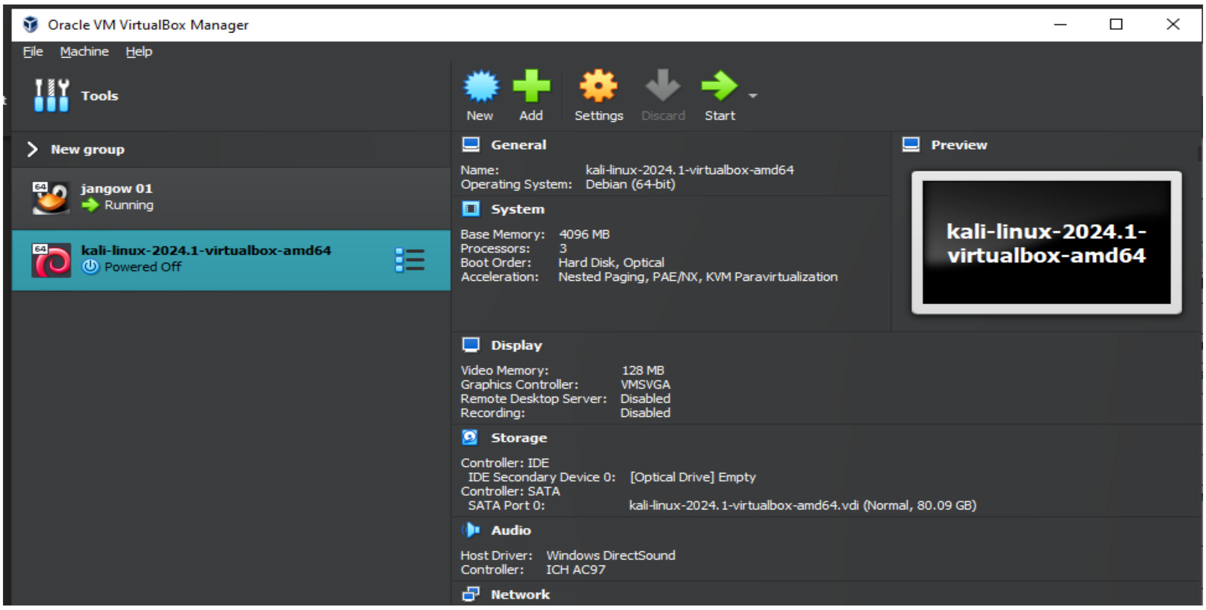


Figure 2: Importing Kali VM

A new virtual machine (VM) was then created in VirtualBox using the provided .iso file. The system was assigned 4GB RAM, 3 CPU cores and 32GB mounded storage.

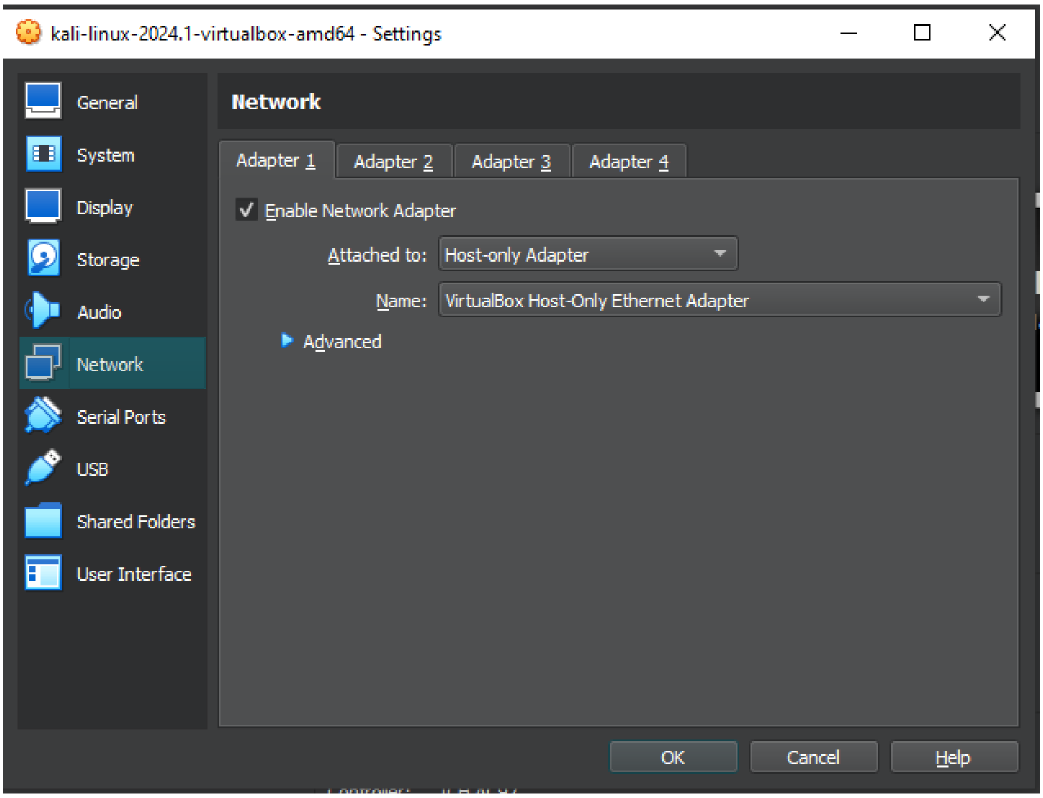


Figure 3: Kali adapter 1

The Kali Linux VM's network is set to a Host-only Adapter in VirtualBox, enabling network interactions solely between the host and the VM, and among VMs on the same host. This isolated network setup allows for a secure, contained environment where the attacker's activities are confined, preventing external network exposure and facilitating safe, controlled testing and analysis.

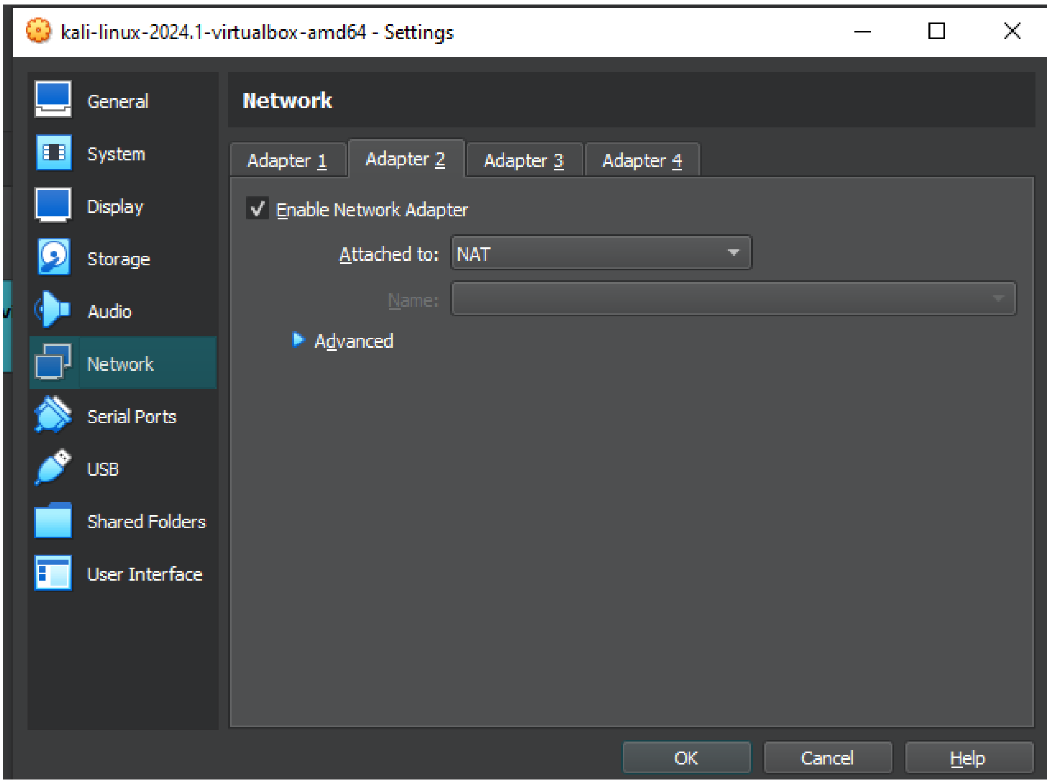


Figure 4: Kali's second adapter

The second adapter is set to NAT which means the virtual machine will use the host's internet connection, which could be through Wi-Fi, to access external networks.

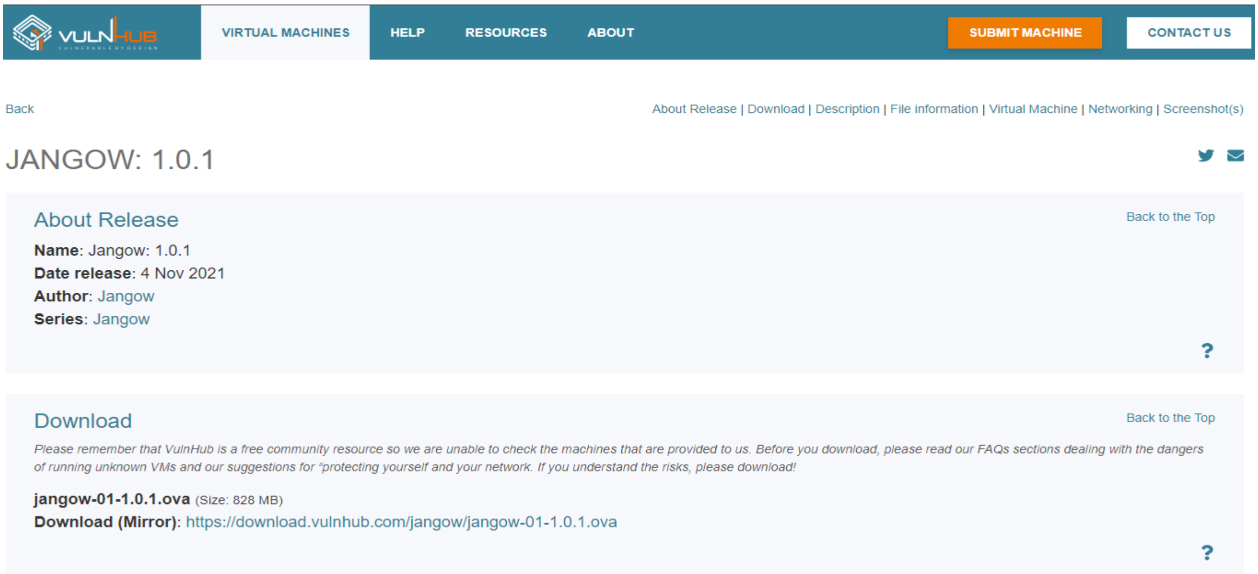


Figure 5: Target (Jangow01) download

The target machine can be downloaded from the (Vulnhub, 2022) website. This specific scenario involved downloading a .oba file provided by lecturer (Bamfo.G, 2024).

Chapter 2: Summary and recommendations

2.1 Summary

While conducting the internal penetration test, there were alarming vulnerabilities that significantly compromised the system's security. Initially, the HTTP service was found to be improperly secured, allowing the unauthorised retrieval of sensitive files (containing login credentials) by passing Linux commands via the URL. These credentials were leveraged to facilitate unauthorised access to the system.

Subsequently, by exploiting a vulnerability associated with the Berkeley Packet Filter (BPF), a malicious BPF map was created. Malicious bytecode was injected to bypass the verifier, leading to the creation of a socket pair. The attached BPF program to this socket enabled arbitrary memory reads and writes. This manipulation of memory ultimately allowed for the modification of credential structures within the system.

The outcome was the escalation to root privileges, which granted complete control over the system. This exploit outlines the need for security protocols and regular system audits. Consider the below recommendations.

2.2. Recommendations

- Conduct thorough code reviews and audits, focusing on systems involving kernel-level operations like BPF. Ensure adherence to security best practices, such as input validation and error handling.
- Regularly update and patch systems, applications, and kernel versions to mitigate known vulnerabilities exploitable by BPF programs.
- Safeguard sensitive files, like those containing credentials, from HTTP access. Employ proper file permissions and access controls.
- Deploy intrusion detection systems (IDS) to detect and alert on abnormal activities, such as unusual system calls or attempts to attach BPF programs to sockets.

• Provide regular security training for staff on secure coding practices and emerging threat vectors. Emphasise securing inter-process communication and preventing privilege escalation.

• Develop and maintain a robust incident response plan that addresses BPF and command injection exploits. Train the security team for effective incident response.

• Enforce strict firewall rules to control inbound and outbound network traffic, preventing exposure of sensitive endpoints and services.

Chapter 3 Methodology

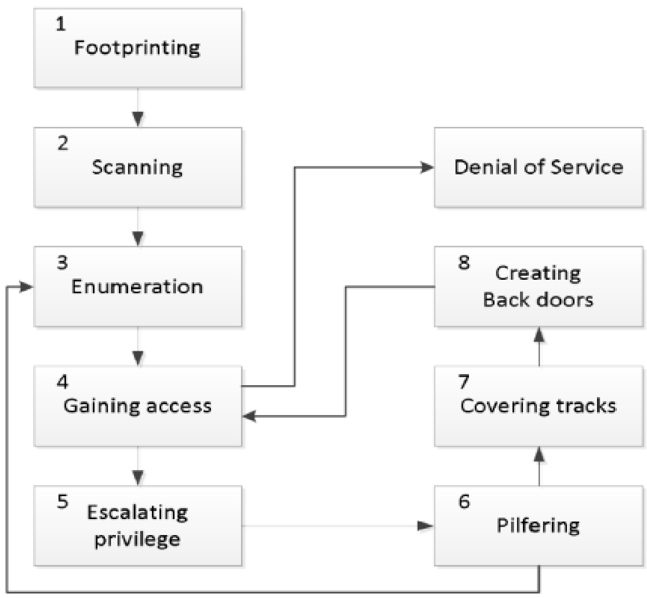


Figure 6: Methodology

The diagram outlines a structured penetration testing methodology provided by (Bamfo.G, 2024) designed to identify and exploit vulnerabilities in a system to gain root access, cover tracks, and create a back door. The stages involved are:

• Footprinting: Gather information about the target system, including public records, domain details, network structure, and services.

• Scanning: Actively engage with the system to identify live hosts, open ports, and services, crucial for discovering vulnerabilities.

• Enumeration: Extract detailed information from identified services, such as user accounts and network shares.

• Gaining Access: Exploit vulnerabilities to gain unauthorised access to the system, utilising various attack vectors.

• Escalating Privilege: Focus on escalating privileges to gain root or administrative access, exploiting system or application flaws.

• Covering Tracks: Erase or alter logs and evidence to avoid detection/maintain access.

• Creating Backdoors: Establish backdoors for long-term access, bypassing normal authentication procedures.

Chapter 4 Information Gathering

4.1 Attacker/Target IP info

```
┌──(essa09㉿kali)-[/home]
└─$ ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.101  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::bd9c:a6b3:66ad:30bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1e:36:4a  txqueuelen 1000  (Ethernet)
        RX packets 88  bytes 36158 (35.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59  bytes 21739 (21.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Table 1: Kali (attacker) IP address

For the attacker machine (Kali), the primary network interface, eth0, is active and configured with the IPv4 address 192.168.56.101, with a subnet mask of 255.255.255.0, indicating a standard Class C network.

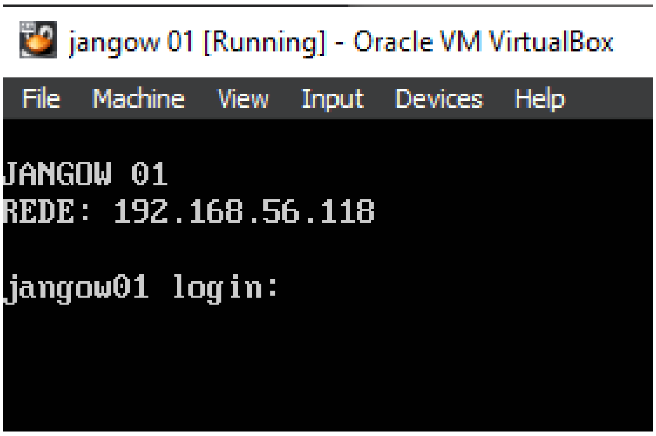


Figure 7:Target IP address

The text "REDE: 192.168.56.118" is likely indicative of the network configuration, with "REDE" translating to "network" in Portuguese, implying the IP address of this virtual machine is set to 192.168.56.118.

4.2. NetDiscover

(Netdiscover Project, 2023) explains that Netdiscover is a network discovery tool that uses ARP (Address Resolution Protocol) to identify active hosts on a local Ethernet network, either actively by sending ARP requests or passively by sniffing for ARP traffic.

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo netdiscover -i eth0
```

```
Currently scanning: 192.168.143.0/16   |   Screen View: Unique Hosts

 4 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 240
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.56.1    0a:00:27:00:00:14      1      60  Unknown vendor
 192.168.56.100  08:00:27:6f:5b:35      2     120  PCS Systemtechnik GmbH
 192.168.56.118  08:00:27:ed:b2:28      1      60  PCS Systemtechnik GmbH
```

Table 2: NetDiscover to ensure both are on the same network

A NetDiscover scan was done to ensure the target machine (Jangow01) was on the same network as the attacker (Kali) machine. Upon scanning the ethernet port on the Kali machine, I was able to confirm that the target was accessible by the attacker, as the IP address was discoverable.

4.3 Initial ping

```
┌──(essa09㉿kali)-[/home]
└─$ ping 192.168.56.118
```

```
PING 192.168.56.118 (192.168.56.118) 56(84) bytes of data.
64 bytes from 192.168.56.118: icmp_seq=1 ttl=64 time=1.79 ms
64 bytes from 192.168.56.118: icmp_seq=2 ttl=64 time=0.791 ms
64 bytes from 192.168.56.118: icmp_seq=3 ttl=64 time=0.542 ms
64 bytes from 192.168.56.118: icmp_seq=4 ttl=64 time=0.909 ms
```

Table 3: Pinging target jangow01 machine

The ping command was used to test the reachability of the target machine along with the round-trip time by echoing ICMP requests, waiting for a reply then measuring the RT (round-trip) time. The ping was successful in sending and receiving 4 packets (before manually interrupted) at a low latency (since both Kali and jangow01 are within the same network), indicating the target machine is reachable.

4.4 Foot printing

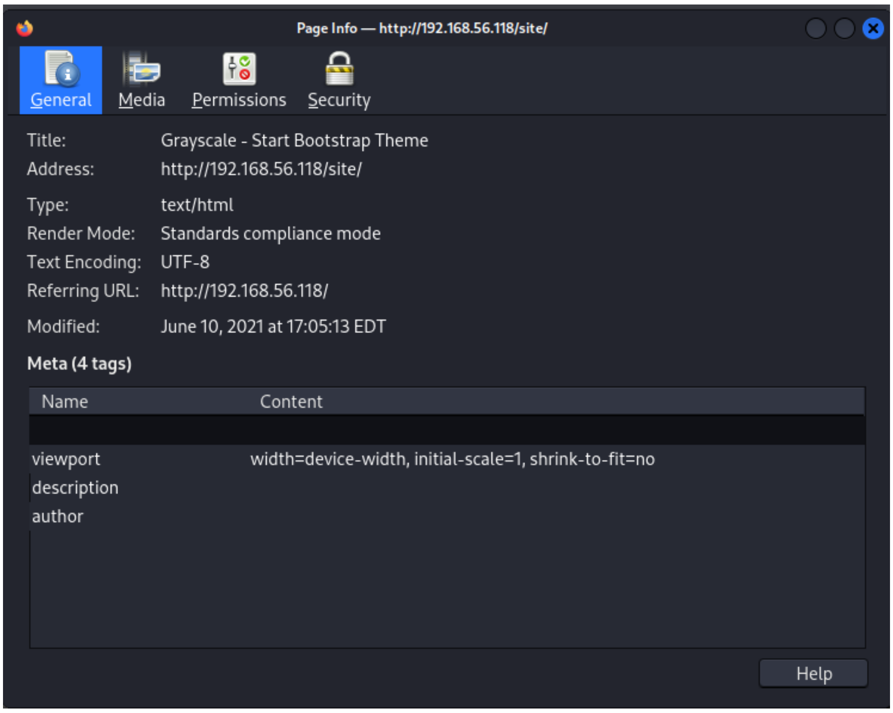
<table>
<tr><th colspan="2">General Details: Target-1</th></tr>
<tr><td>Domain Name</td><td>Grayscale - Start Bootstrap Theme</td></tr>
<tr><td>IP Address</td><td>http://192.168.56.118</td></tr>
<tr><td>Content Type</td><td>Text/HTML</td></tr>
<tr><td>Audience</td><td>Web Developers</td></tr>
<tr><td>Status</td><td>Running</td></tr>
<tr><td>Location</td><td>Local VM</td></tr>
<tr><td>Name Server(s)</td><td>N/A</td></tr>
<tr><td>Trace</td><td>N/A</td></tr>
<tr><td>Administration Page</td><td>http://192.168.56.118/site/busque.php?buscar=</td></tr>
<tr><th colspan="2">Technical Details</th></tr>
<tr><td>Certificate Information</td><td>
</td></tr>
<tr><td>Programming Language(s)</td><td>Html</td></tr>
</table>

Table 4: Footprint info of target

Chapter 5: Scanning and Mapping

5.1 OpenVas/GreenBone Scan

This section will outline the starting and vulnerability scanning of the target machine using GreenBone, which is formally known as OpenVas. It is an open-source vulnerability scanning and management suite. (Greenbone Networks, 2023) outline that it provides comprehensive scanning, assessment, and management of vulnerabilities.

5.1.1. Checking installation

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo gvm-check-setup
[sudo] password for essa09:
gvm-check-setup 23.11.0
 Test completeness and readiness of GVM-23.11.0
Step 1: Checking OpenVAS (Scanner)...
    OK: OpenVAS Scanner is present in version 22.7.9.
•
It seems like your GVM-20.8.0 installation is OK.
```

Table 5: Ensuring OpenVas installation

OpenVas was installed through GreenBone. Errors were encountered when adding a scanning task. The command above (table 4) ensured that GreenBone was installed accurately with no missing resources. The error encountered was solved by simply allowing OpenVas to install its resources (took around three hours since VM is in HDD and not SSD.

5.1.2. Starting Service

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo gvm-start
[sudo] password for essa09:
[>] Please wait for the GVM services to start.
[>] You might need to refresh your browser once it opens.
[>]  Web UI (Greenbone Security Assistant): https://127.0.0.1:9392
[>] Opening Web UI (https://127.0.0.1:9392) in: 5... 4... 3... 2... 1...
```

Table 6: Launching OpenVas service

The OpenVas/Greenbone was started, meaning it could be accessed through a conventional browser (i.e. Firefox, preinstalled with Kali, due to its low resource requirements when compared to Chrome OS)

5.1.3. Defining target machine creating a new task

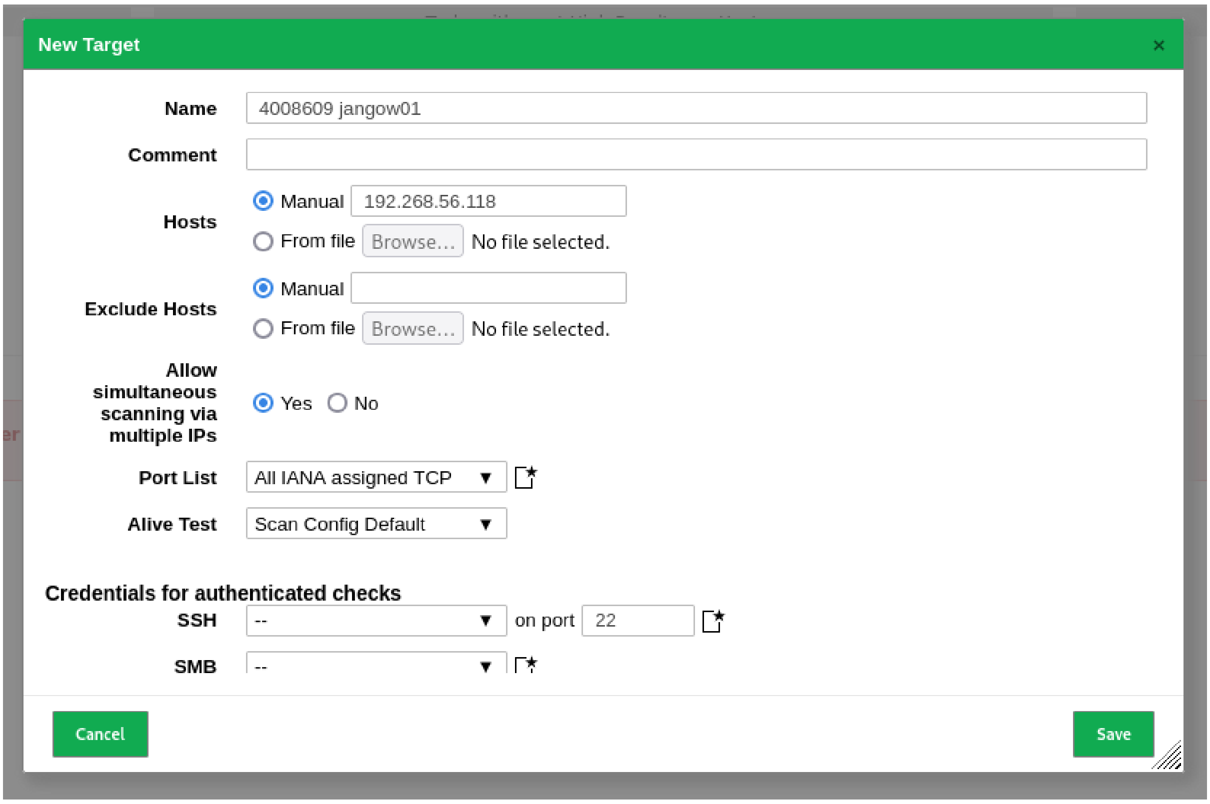


Figure 8: Creating a new target in OpenVas

After the GreenBone./OpenVas was installed, the target machine was defined within OpenVas with the name' 4008609 jangow01'. The target host is set to the IP address of the target jangow01 machine (192.168.56.118). The port list setting 'All IANA assigned TCP' , where IANA stands for Internet Assigned Numbers Authority, scans every TCP port for potential vulnerabilities.

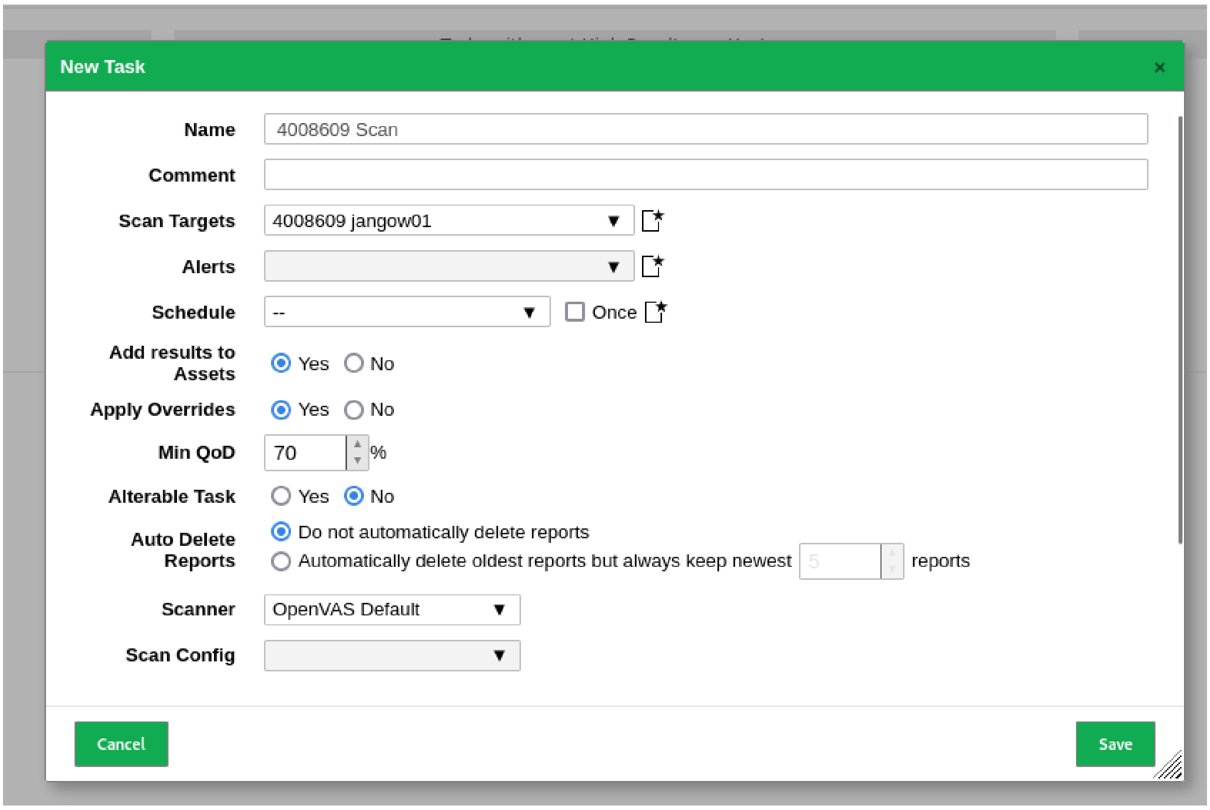


Figure 9: Creating a new task in OpenVas

A new vulnerability task scan is created within OpenVas with the name '4008609 Scan'. "Add results to Assets" is set to "Yes," meaning that after the scan, the results will be added to the asset management database for tracking and past comparison. The "Apply Overrides" is selected as "Yes" with a "Min QoD" (Minimum Quality of Detection) set to 70%. which allows for prioritisation of certain vulnerabilities over others based on the reliability of the vulnerability's detection.

5.1.4. Scan Results

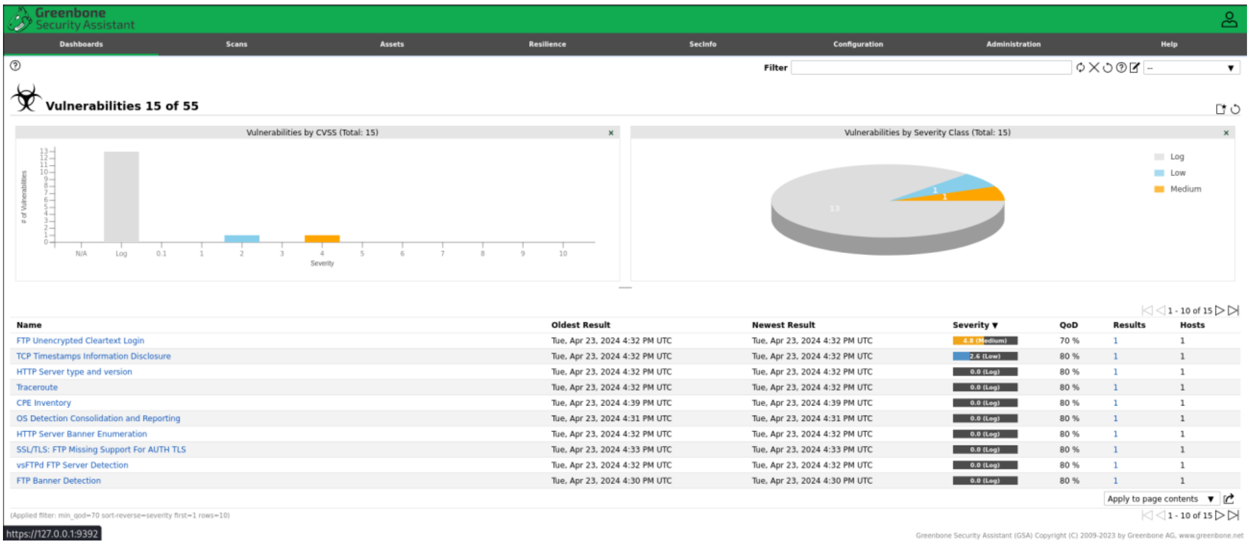


Figure 10: OpenVas vulnerability scan results

The Greenbone Security Assistant dashboard highlights two vulnerabilities, including "FTP Unencrypted Cleartext Login" and "HTTP Server type and version" which are both associated with significant security risks. These vulnerabilities suggest unencrypted login credentials are reasonably easily accessible. A further nessus scan was conducted to gain a deeper insight.

5.2 NMAP scan

Nmap (Network Mapper) is an open-source tool used for network discovery and security auditing. (Nmap Project, 2023) explain that it works by sending packets to network hosts and analysing their responses to discover host availability, services, operating systems, and types of packet filters/firewalls. It's useful for both network inventory and vulnerability detection.

```
┌──(essa09㉿kali)-[/home/kali]
└─$ nmap -A -p- -T4 192.168.56.118
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-22 23:56 EDT
Nmap scan report for 192.168.56.118
Host is up (0.0013s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
80/tcp open  http    Apache httpd 2.4.18
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2021-06-10 18:05  site/
|_
|_http-title: Index of /
Service Info: Host: 127.0.0.1; OS: Unix

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 108.98 seconds
```

Table 7: Nmap scan of the target machine

The Nmap network scanning tool was used along with option '-A' which enables OS detection, security auditing, etc. The '-p- options ensures that all 65535 ports are scanned. The -T4 option sets the timing template to aggressive which speeds up the scan through making certain timing assumptions. The output shows that two ports are

Chapter 6: Enumeration & Gaining access

6.1 Dirb content scan

DIRB, developed by (The Dark Raver, 2022), is a web content scanner that detects hidden files and directories on servers. It sends HTTP requests based on a predefined wordlist and analyses server responses to uncover potential vulnerabilities or misconfigurations. It's widely used in security testing to assess web server security by revealing inaccessible or unprotected content.

```
┌──(essa09㉿kali)-[/home/kali]
└─$ dirb http://192.168.56.118/site/
/usr/share/dirb/wordlists/common.txt
```

```
-----------------
DIRB v2.22
By The Dark Raver
-----------------
START_TIME: Tue Apr 23 00:02:37 2024
URL_BASE: http://192.168.56.118/site/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----------------
GENERATED WORDS: 4612
---- Scanning URL: http://192.168.56.118/site/ ----
==> DIRECTORY: http://192.168.56.118/site/assets/
==> DIRECTORY: http://192.168.56.118/site/css/
+ http://192.168.56.118/site/index.html (CODE:200|SIZE:10190)
==> DIRECTORY: http://192.168.56.118/site/js/
==> DIRECTORY: http://192.168.56.118/site/wordpress/
```

Table 8: Scanning for common words

The target machine was then scanned using the dirb web content scanner which launched a dictionary attack against the web server to search for hidden and non-hidden web objects. It scans the target machine using the common.txt wordlists file. The scan was able to identify a word press file which could be potentially exploited as it may contain hidden credentials of the target VM.

6.2 Nessus Scan

Nessus is a widely used cybersecurity tool developed by (Tenable, Inc., n.d.). It is designed for the assessment and management of vulnerabilities. It scans computing systems for known vulnerabilities, i.e. weak firewalls, outdated software, etc that could be exploited by attackers. Nessus works by first performing a network discovery to identify devices and their operating systems, services, and open ports on a network. Then, it examines these components against a database of known vulnerabilities.

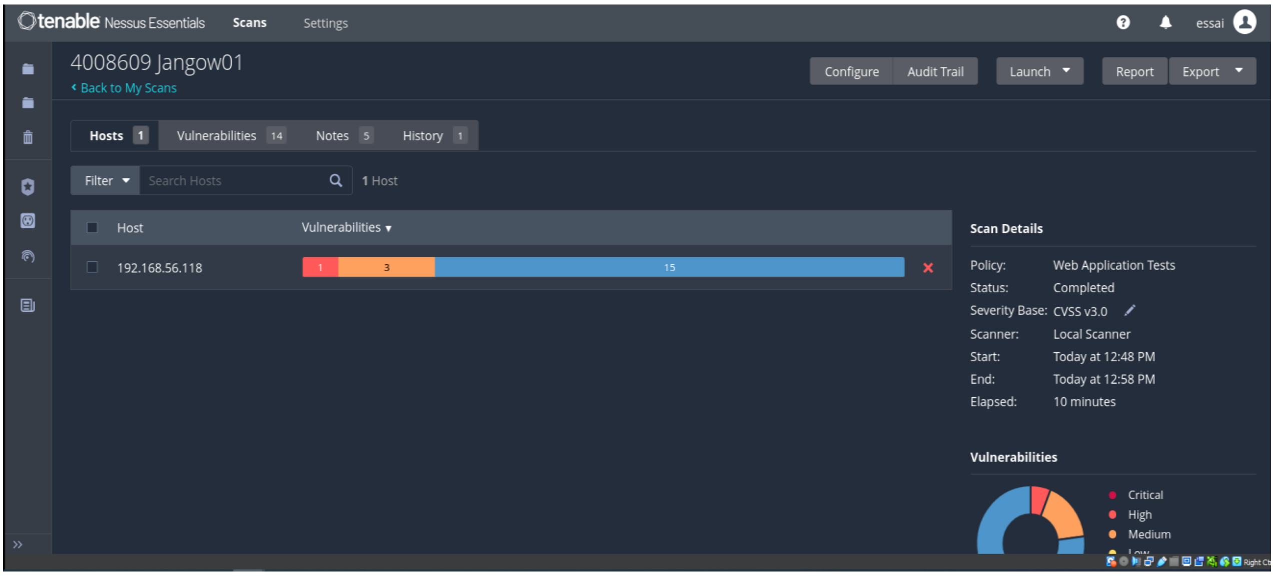


Figure 11: Nessus scan basics stats

The screenshot displays a completed vulnerability scan from Nessus with the jangow01 host IP address 192.168.56.118 where 14 vulnerabilities were found. Of these, 1 is critical, 3 are high. The rest are of medium and low severity. The scan was performed with a policy tailored for web application tests and took 10 minutes to complete using a local scanner.

tenable Nessus

Report generated by Nessus™

4008609 Jangow01

Tue, 23 Apr 2024 12:58:33 EDT

TABLE OF CONTENTS

Vulnerabilities by Host

Collapse All | Expand All

192.168.56.118

0	1	3	0	14
CRITICAL	HIGH	MEDIUM	LOW	INFO

Severity	CVSS v3.0	VPR Score	Plugin	Name
HIGH	7.5*	-	39465	CGI Generic Command Execution
MEDIUM	5.3	-	40984	Browsable Web Directories
MEDIUM	5.0*	-	57640	Web Application Information Disclosure
MEDIUM	4.3*	-	85582	Web Application Potentially Vulnerable to Clickjacking

Figure 12: Nessus scan in-depth info of vulnerabilities

The high-severity issue is a CGI Generic Command Execution vulnerability with a CVSS v3.0 score of 7.5, which is quite serious and should be addressed promptly. The medium severity issues include vulnerabilities such as Browsable Web Directories and Web Application Information Disclosure, indicating areas where sensitive information could potentially be accessed by unauthorised users. There's also a medium severity issue listed as Web Application Potentially Vulnerable to Clickjacking, which could allow an attacker to trick a user into clicking on something different from what the user perceives, potentially revealing confidential information or allowing unauthorised actions.

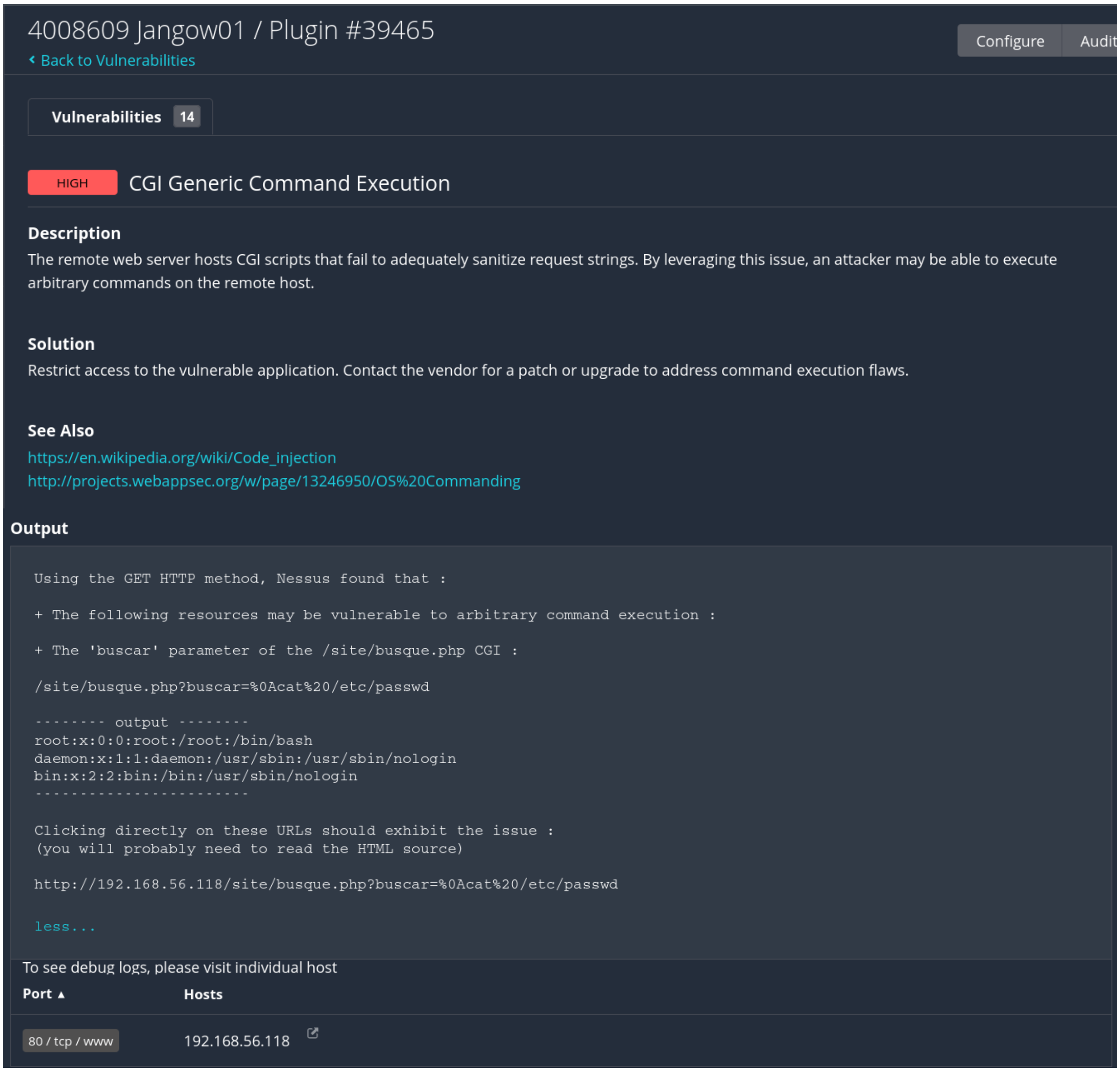


Figure 13: CGI Generic command execution vulnerability insight (High severity)

The high vulnerability indicated in the Nessus report is a "CGI Generic Command Execution" flaw on a web server, which allows an attacker to execute arbitrary commands due to inadequate input sanitisation in CGI scripts. The specific issue demonstrated in the report shows an attacker could leverage the 'buscar' parameter in a PHP script to gain access to sensitive system files like /etc/passwd, posing a serious security risk that requires immediate attention, including restricting access and seeking a patch from the software vendor.

6.3 Accessing webpage

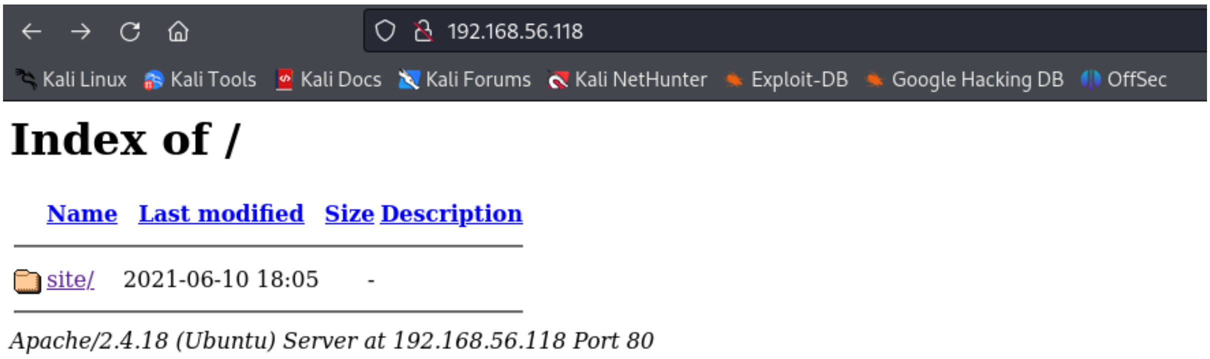


Figure 14: Accessing HTTP using Firefox

The Firefox web browser was used to enter the IP address of the target machine. The page displayed the directory listing for the site, indicating that the web server was functioning and served the expected directory as its homepage. This initial access to the server's content directory confirmed the server was responsive and provided a starting point for further investigation.

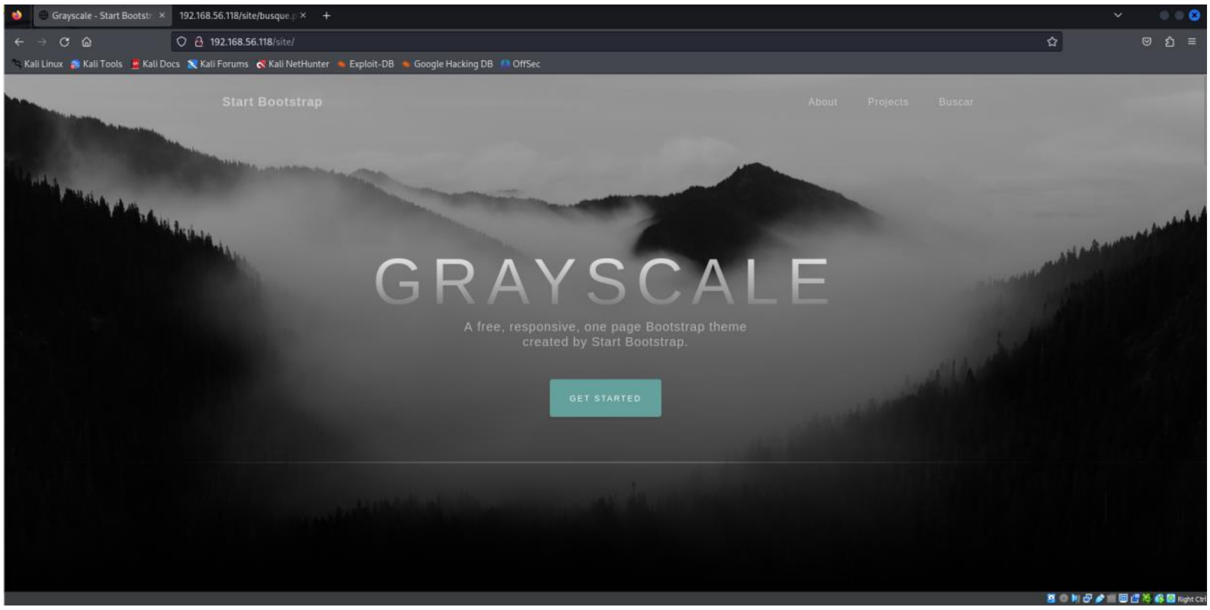


Figure 15: Accessing the main site/page

Upon reaching the main directory index of the target's website, various hyperlinks associated with the site's structure were explored. This interaction was crucial to understanding the layout and potential entry points of the website. It offered insight into the organisation of the server's directories and files, presenting avenues for deeper exploration of the site's structure. As the Nmap scan showed the site/file, I was able to access it using the URL and interacting with the HTML.

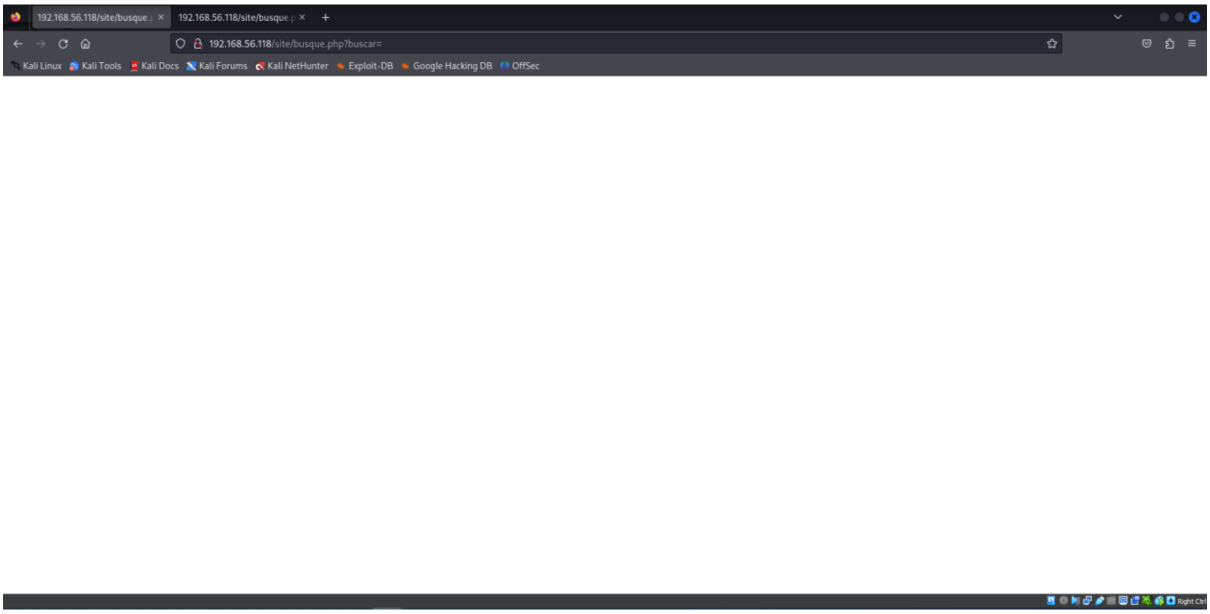


Figure 16: Identifying blank page

The Nessus vulnerability scan directed attention to a non-functional 'search' (buscar translation) feature within the site. When attempting to use this search a blank page showed. This suggested a possible fault or misconfiguration in the page's PHP code. This was also pointed out in previous scans. This blank response was an anomaly, that encouraged further exploration.

6.4 Executing Linux commands via URL

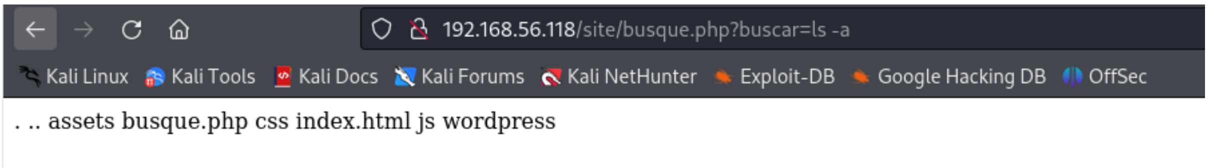


Figure 17: Listing hidden files in buscar php

By manipulating the website's URL (through Linux commands), I was able to reveal hidden server files, while bypassing the usual user interface constraints. The listed files included a WordPress directory previously discovered during the Nmap scan

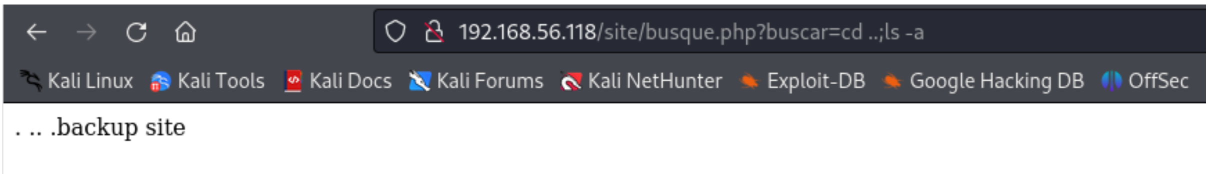


Figure 18: Accessing parent directory relative to busque.php

Exploiting the vulnerability identified in the previous step, directory traversal was used to access higher-level directories. This revealed more files and directories, allowing a more granular examination of the server's file structure. While the 'site' file did not yield any useful information, this method demonstrated the ability to navigate the server's file system beyond the intended limits of the web interface.

6.5. Getting Credentials

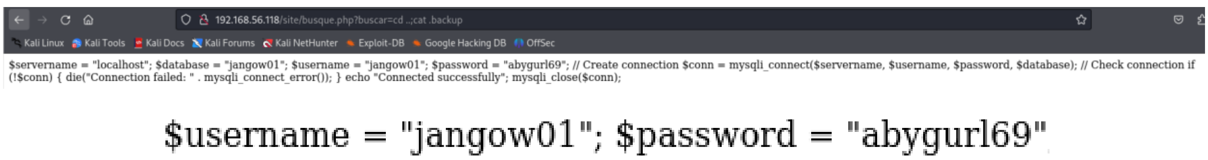


Figure 19: Identifying credentials within old SQL backup trace

A hidden file, likely an old SQL database backup, was found. The contents, when viewed, displayed plain text credentials. The username 'jangow01' paired with the password 'abygurl69' were clearly outlined, representing a significant security lapse. Such sensitive information should not have been stored in an unsecured manner, and its presence indicated a disregard for proper security practices regarding sensitive data. It was also recognised by the OpenVas scan.

6.6 Gaining Access

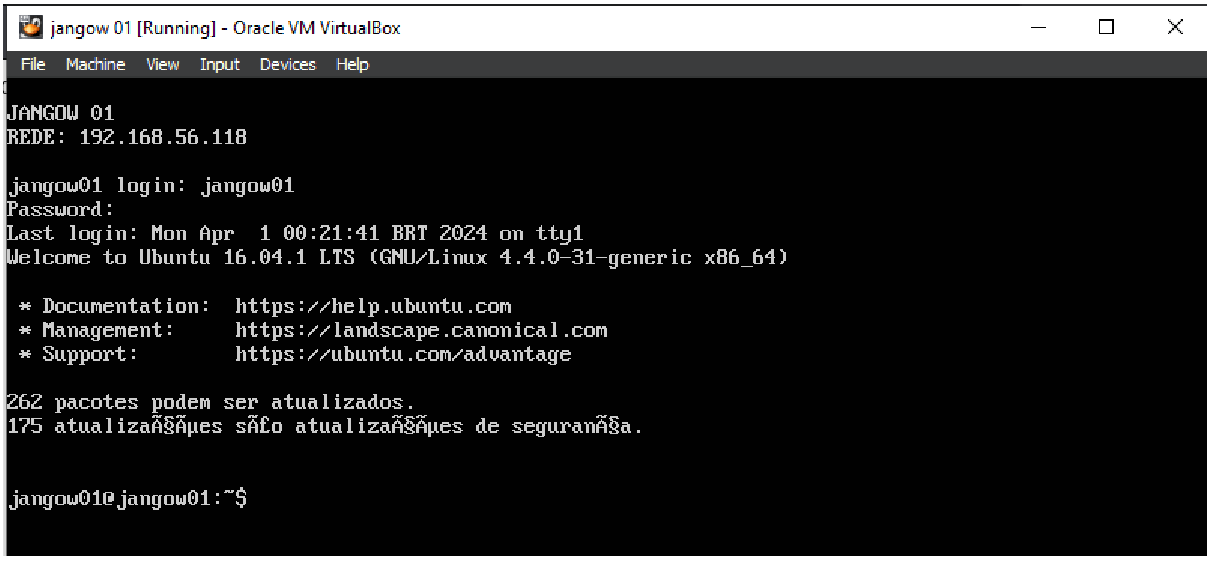


Figure 20: Logging in to the target machine using log-ins

Utilising credentials extracted from a plain text SQL backup aided through insights gained from Nessus and OpenVAS scans, the penetration testing was able to progress. The SQL backup provided essential login details, which allowed for initial access to the system.

6.6.1. Getting User.txt

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
ftp> cd /home/jangow01
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||34487|)
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000           33 Jun 10  2021 user.txt
226 Directory send OK.
ftp> get user.txt
local: user.txt remote: user.txt
229 Entering Extended Passive Mode (|||44636|)
150 Opening BINARY mode data connection for user.txt (33 bytes).
100%
|*******************************************************************
********|    33        2.74 KiB/s    00:00 ETA
226 Transfer complete.
33 bytes received in 00:00 (2.25 KiB/s)
```

Table 9: Establishing an FTP connection and getting user.txt file in jangow01 directory

A ssh connection was attempted but was unsuccessful. This was expected as the previous scans revealed that only port 80 (http) and port 21 (FTP) were open and. The method used to remotely access the target machine was through FTP (file transfer protocol). Upon exploring the directories (that didn't require root access). I was able to identify a file named user.txt. . The discovery of user.txt provided an additional avenue for investigation without the need for elevated privileges just yet.

6.6.2. Reading User.txt

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo ls
BurpSuiteCommunity  Desktop    Downloads       Music     Public
server.csr  Templates  Videos
cmd2html.sh         Documents  ls_output.html  Pictures  server.crt
server.key  user.txt

┌──(essa09㉿kali)-[/home/kali]
└─$ sudo cat user.txt
d41d8cd98f00b204e9800998ecf8427e
```

Table 10: Reading user.txt

The user.txt file was expected to contain important data, but it turned out to hold a known MD5 hash indicative of an empty file. The MD5 hash, **d41d8cd98f00b204e9800998ecf8427e**, is commonly used to represent a zero-length string. This means the file, while present, contained no data. This lack of content suggests that either the file was a placeholder or a result of a system error.

The discovered credentials from the SQL backup were used to gain entry into the target system. The successful login provided an interactive session with the target machine, laying the groundwork for deeper system exploration and potential privilege escalation.

Chapter 7: Escalating Privileges

7.1 Transferring Root Access Exploit

```
┌──(essa09㉿kali)-[/home/kali]
└─$ sudo ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /home/jangow01
250 Directory successfully changed.
ftp> put attack.c
local: attack.c remote: attack.c
229 Entering Extended Passive Mode (|||16917|)
150 Ok to send data.
100%
|*************************************************************
**************| 13241        17.25 MiB/s    00:00 ETA
226 Transfer complete.
13241 bytes sent in 00:00 (4.19 MiB/s)
ftp>
```

Table 11: Putting (transferring) root access attack to target machine

I determined that the jangow01 machine was running Linux kernel version 4.4.0-31-generic after executing the uname -a command. (Exploit Database, 2023) revealed that this kernel version was susceptible to a known privilege escalation vulnerability. I found an exploit by (Exploit Database, 2023) written in C that targets this specific vulnerability, which could allow a user to escalate their privileges to root on the affected system.

7.2 Compiling and executing exploit (Gaining Root)

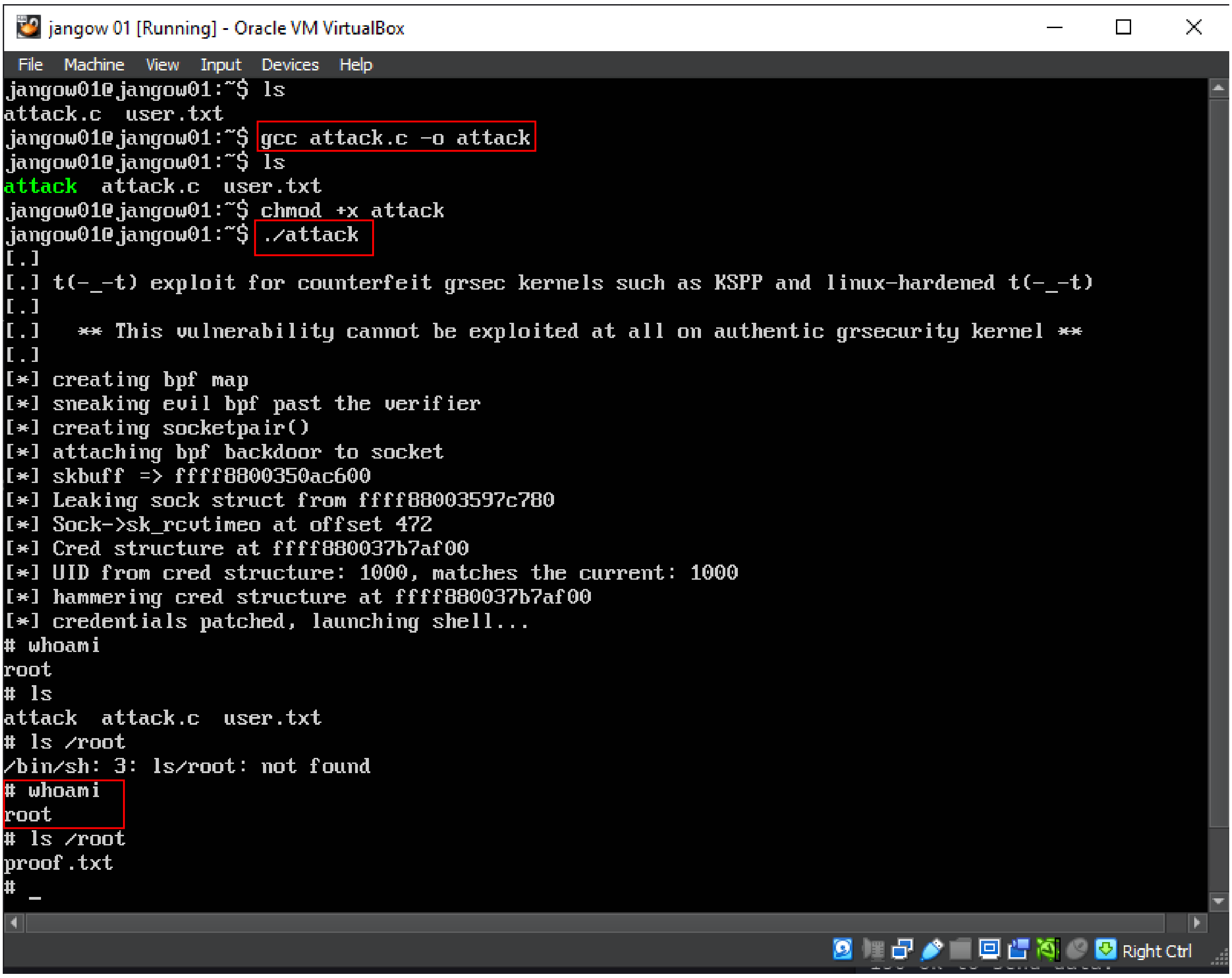


Figure 21: Compiling and executing root access attack

Through analysing the exploit file, it seems to take advantage of the flaw in BPF (Berkeley Packet Filter). It works by initially creating a BPF map and massing malicious bytecodes in order to skip the verifier. It then creates a socket pair which attaches the BPF program to the socket which initiates memory reads and writes. This then leads to modified credentials structures which allow root privilege escalation.

7.3 Getting access to Proof.txt

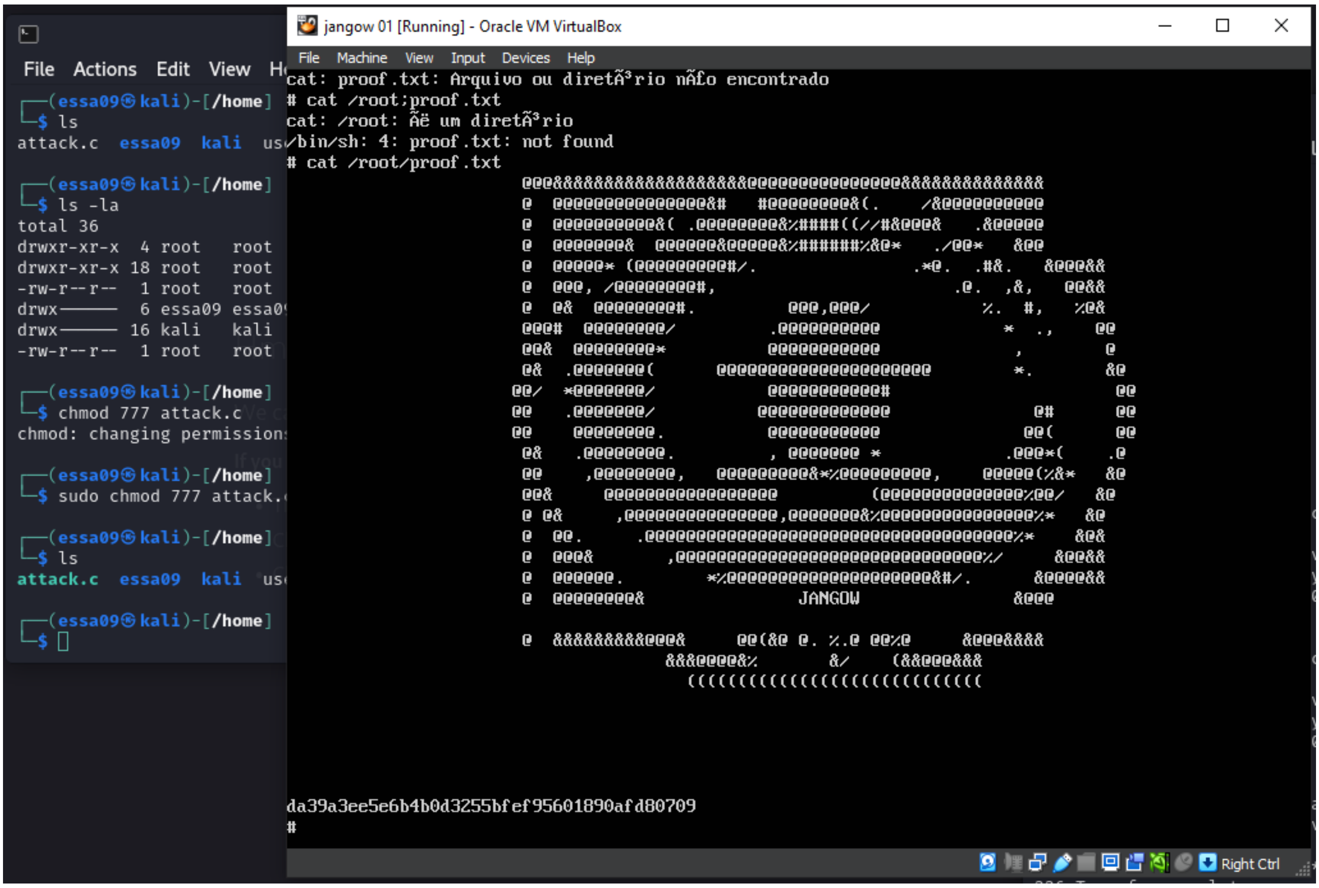


Figure 22: Outputting proof.txt file in the root directory

After successfully exploiting the vulnerability in the Linux kernel and gaining root privileges, I accessed the root directory of the jangow01 machine. Within this directory, I located the file named **proof.txt**, which was critical to the objectives of the penetration test. Accessing this file as the root user confirmed the effectiveness of the exploit and demonstrated complete control over the target system. The presence of proof.txt in the root directory typically serves as definitive **evidence that administrative-level access has been achieved**, validating the success of the penetration testing process.

Chapter 8: Bonus

The bonus section involves creating a backdoor, pilfering and covering tracks.

8.1 Creating backdoor (Reverse ssh)

8.1.1. Ensuring ssh server is installed and running

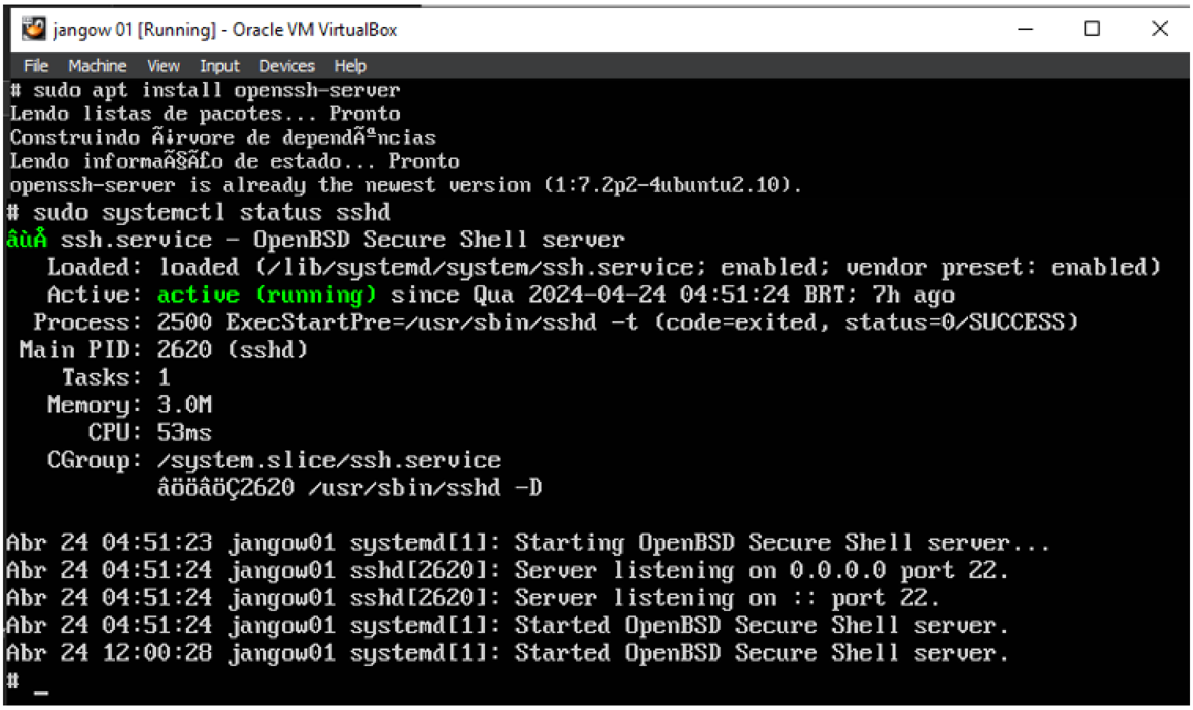


Figure 23: Checking SSH Status

To prepare for a reverse SSH backdoor, the OpenSSH server is installed on the target machine using sudo apt install OpenSSH-server. This enables secure remote access. The status of the SSH service is then confirmed with sudo systemctl status sshd, ensuring it's active and ready to accept connections. These steps establish the foundation for remote access to the target system.

8.1.2. Adjusting firewall

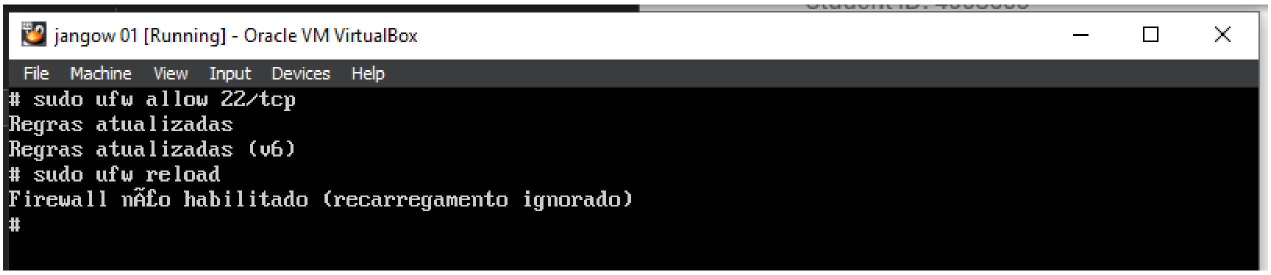


Figure 24: Allowing SSH port (22)

Executing’ sudo ufw allows 22/tcp’ configures the firewall to allow SSH traffic on port 22, allowing for remote access to the target system. This command ensures that the reverse SSH backdoor can establish a secure connection for remote management and control.

8.1.3. Attempting to reverse SSH

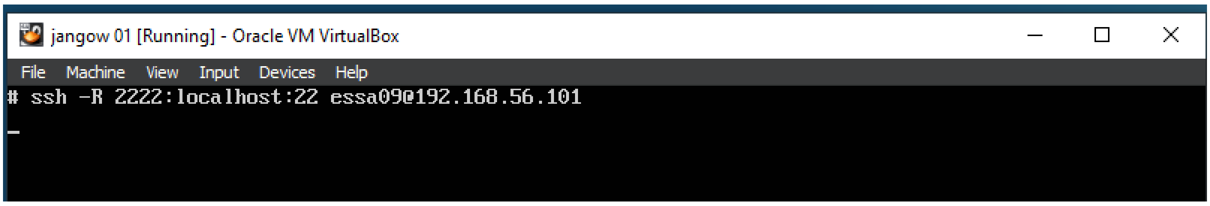


Figure 25: Reverse SSH attempt

The reverse SSH command establishes a connection from a remote machine to a local machine, allowing remote access to the local machine's services. However, despite attempting to execute the reverse SSH command, it did not work as expected. This failure is likely attributed to network issues within the virtual machine environment, such as misconfiguration of network settings or firewall rules. Another idea/a way around would be SQL injection.

```
┌──(essa09㉿kali)-[/home/kali]
└─$ ssh -p 2222 localhost
ssh: connect to host localhost port 2222: Connection refused
```

Table 12: Attempting SSH from attacker to target

The attempted SSH connection to localhost on port 2222 failed with a "Connection refused" error as expected. This prevented SSH access from the attacker to the target machine, preventing the implementation of the reverse SSH backdoor.

8.2 Pilfering

During the pilfering phase, I obtained login credentials through web vulnerabilities, aided by Nmap, Nessus, and OpenVAS. Despite initial setbacks with an empty user.txt, escalation to root privileges was achieved, leading to the extraction of proof.txt. These successes outline the effectiveness of the pilfering process in uncovering critical system vulnerabilities and accessing sensitive data. Data already on the system (user.txt and proof.txt) were left as is in their original directories to avoid raising suspicion or doubt.

8.3 Covering Tracks

8.3.1. Deleting exploits

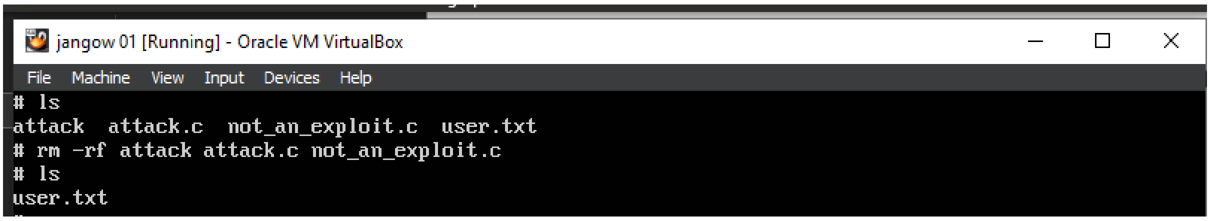


Figure 26: Deleting root access exploits

Exploits used during the penetration test were deleted from the target machine to cover tracks. This involved removing exploit files and associated (including pre and post-compiled C. It is worth noting that tampering with evidence raises legal and ethical implications. Attacks were done locally and on a permitted test machine.

8.3.2.Deleting SSH Logs

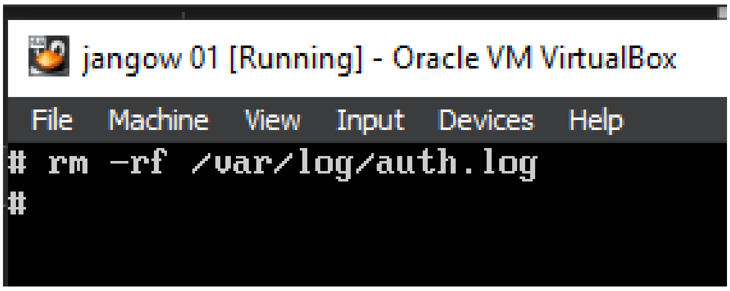


Figure 27: Deleting SSH Logs

SSH logs were deleted to conceal unauthorised access to the system and obscure traces of the intrusion, which would hinder forensic analysis and maintain the anonymity of the attacker.

8.3.3.Deleting System Logs

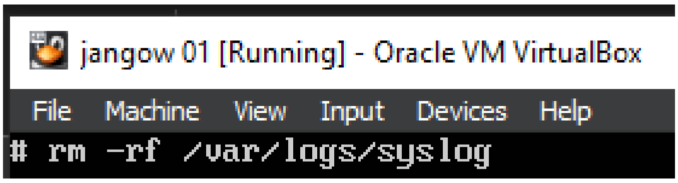


Figure 28: Deleting System Logs

System logs were deleted to obscure evidence of unauthorised access, effectively concealing the attacker's digital trail, and making it harder to trace the intrusion. This was essential for maintaining anonymity and evading detection.

8.3.4.Deleting Apache Logs

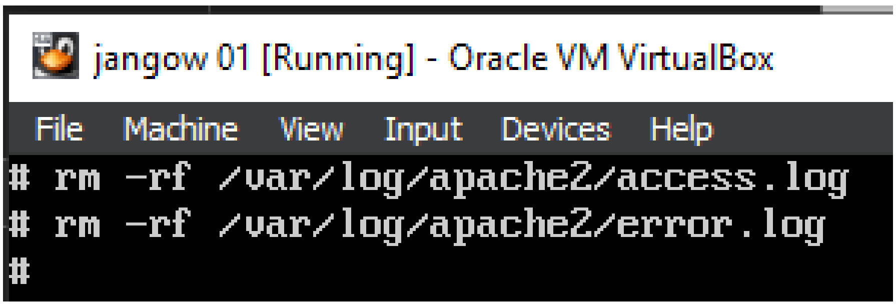


Figure 29: Deleting Apache access and error Logs

Deleting Apache logs was essential to conceal any records of HTTP requests and server interactions, thereby masking the attacker's activities and impeding forensic investigation. This action helped maintain the anonymity of the attacker and minimised the risk of detection.

8.3.5.Deleting Bash History (Attempt)

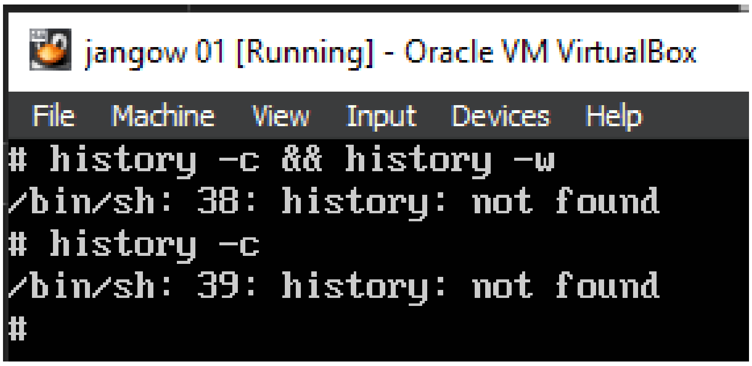


Figure 30: Deleting command history (attempt)

Attempting to delete the Bash history was attempted to remove any records of commands executed by the attacker, which would conceal their actions. However, the attempt was unsuccessful as the "history" command was not found. This is likely due to misconfiguration or restrictions on the system.

Conclusion

This report on reveals significant vulnerabilities within the target system, notably an insecure HTTP service and a Berkeley Packet Filter exploit that allowed escalation to root privileges. These vulnerabilities highlight the urgent need for rigorous security protocols and regular system audits. Recommendations include thorough code reviews, systematic vulnerability scanning, and stringent access controls to safeguard sensitive files and endpoints. The successful exploitation and subsequent access to sensitive files underscore the critical importance of proactive security measures in protecting against sophisticated cyber threats.

Self-Reflection

Through this cyber security black box testing project, I gained insights into system vulnerabilities and potential legal risks. It was an invaluable experience. This project outlines and gives insight in to the critical importance of robust security measures in safeguarding digital assets.

References

[1] Netdiscover Project (2023) *Netdiscover: Active/Passive ARP Reconnaissance Tool*, version [insert version number], Available at: https://github.com/netdiscover-scanner/netdiscover (Accessed: 9th April 2024).

[2] Greenbone Networks, 2023. *Greenbone Vulnerability Management (GVM) version* Available at: https://www.greenbone.net/en/community-edition/ (Accessed 9th April 2024).

[3] Tenable, Inc., n.d. *Nessus Vulnerability Scanner*. Available at: https://www.tenable.com/products/nessus (Accessed: 11th April 2024).

[4] Nmap Project, 2023. *Nmap: Network Mapper, version 7.93*. Available at: https://nmap.org (Accessed: 14th April 2024).

[5] The Dark Raver, 2022. *DIRB v2.22: Web Content Scanner*. Available at: https://www.kali.org/tools/dirb/ (Accessed: 14th April, 2024).

[6] Vulnhub, 2022. *Jangow 101*. Available at: https://www.vulnhub.com/entry/jangow-101,754/ (Accessed: 14th April, 2024).

[7] Kali Linux, 2022. *Get Kali Linux*. Available at: https://www.kali.org/get-kali/ (Accessed: 14th April, 2024).

[8] Exploit Database. 2023. Linux Kernel 4.4.0-31 - 'Dirty COW' Race Condition Privilege Escalation (Metasploit). [online] Available at: https://www.exploit-db.com/exploits/45010 [Accessed 17th April 2024].

Bibliography

[Walkthrough] Ibeakanma, Chioma. "Jangow 101 Walkthrough (Vulnhub)." Chioma Ibeakanma - Hashnode. Accessed 16th April 2024. URL: https://chiomaibeakanma.hashnode.dev/jangow-101-walkthrough-vulnhub