FINAL VAPT REPORT

PREPARED BY: Almotasem Bellah Mahsoun

Submitted To: Neue Fische

Submission Date: 16/09/2025

neue fische
School and Pool for Digital Talent

TABLE OF CONTENTS

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

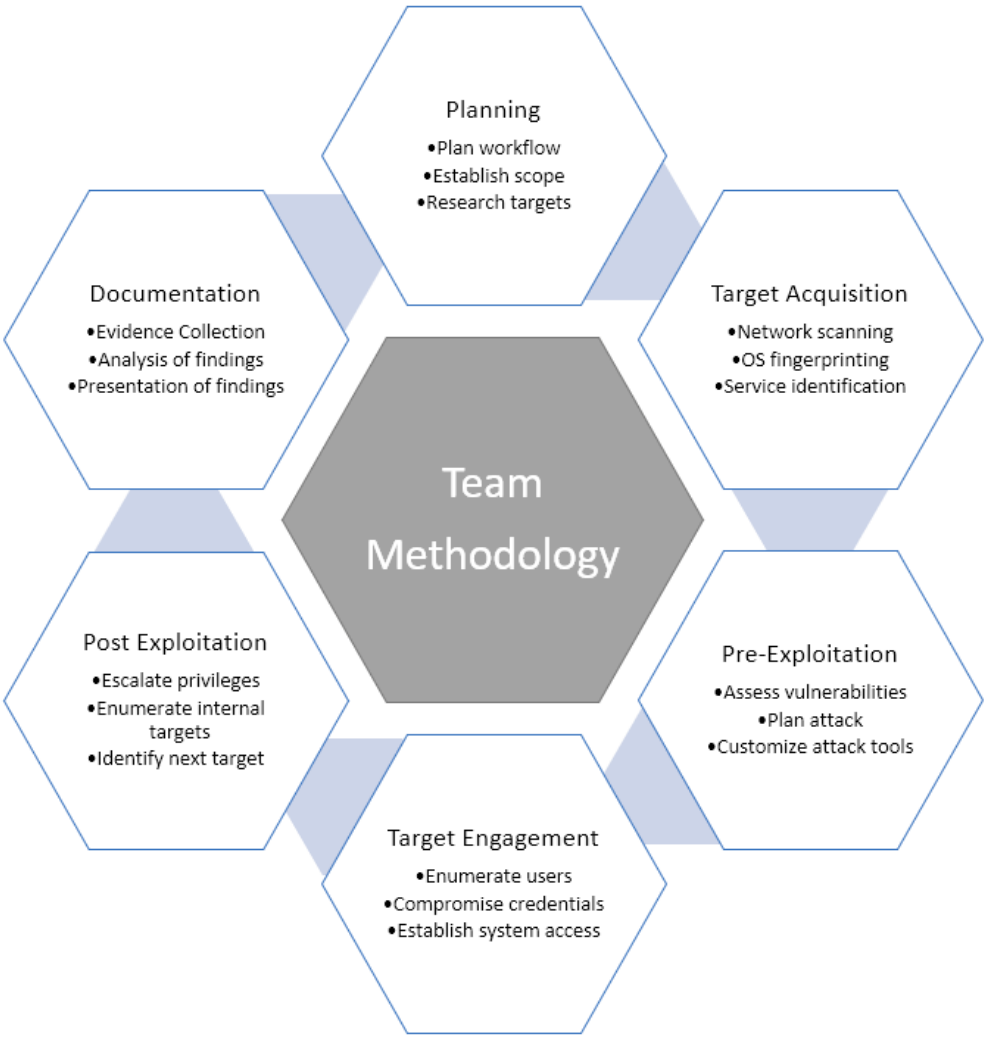
Network Information

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.

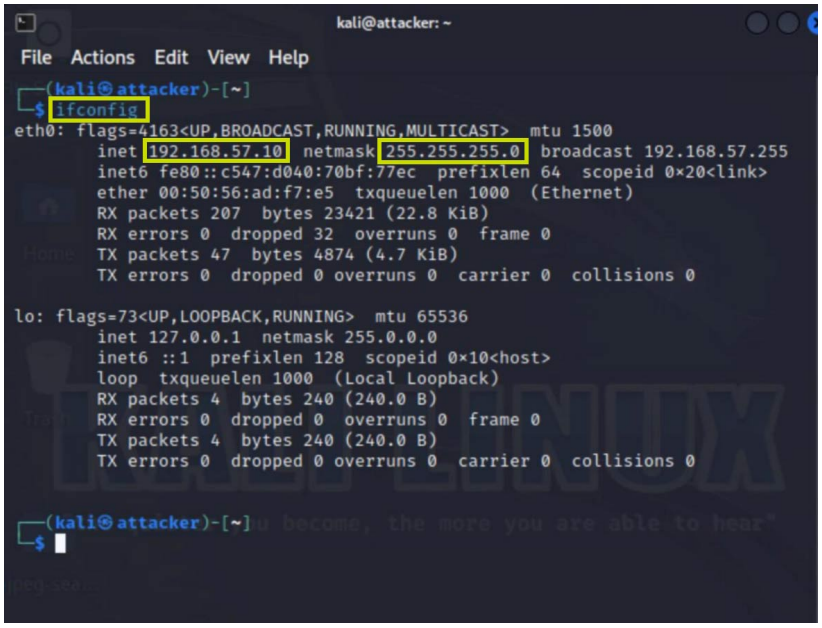


Figure 1.1 – Output of command

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.

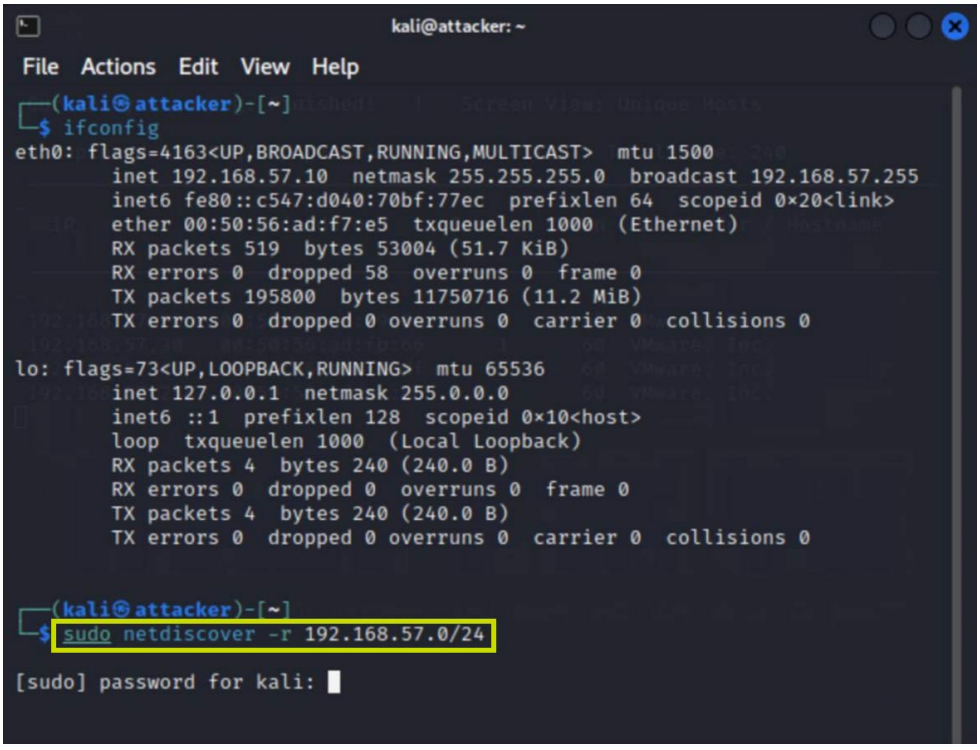


Figure 1.2.1 – Netdiscover scan for subnet 192.168.57.0/24

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.

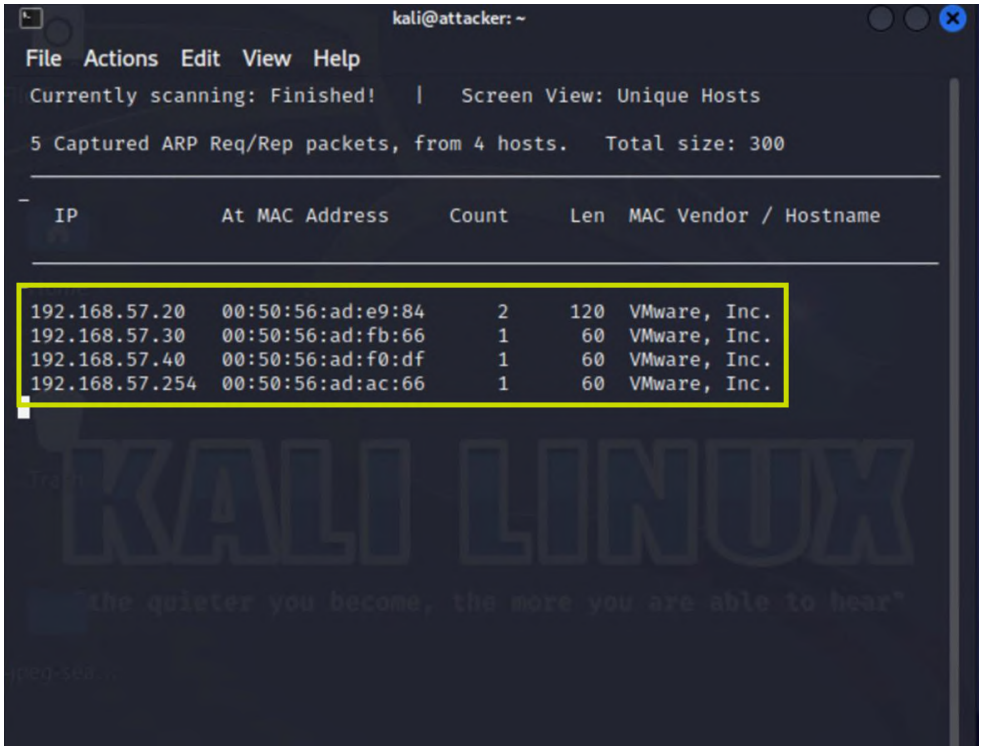


Figure 1.2.2 – Netdiscover scan results showing active hosts

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:

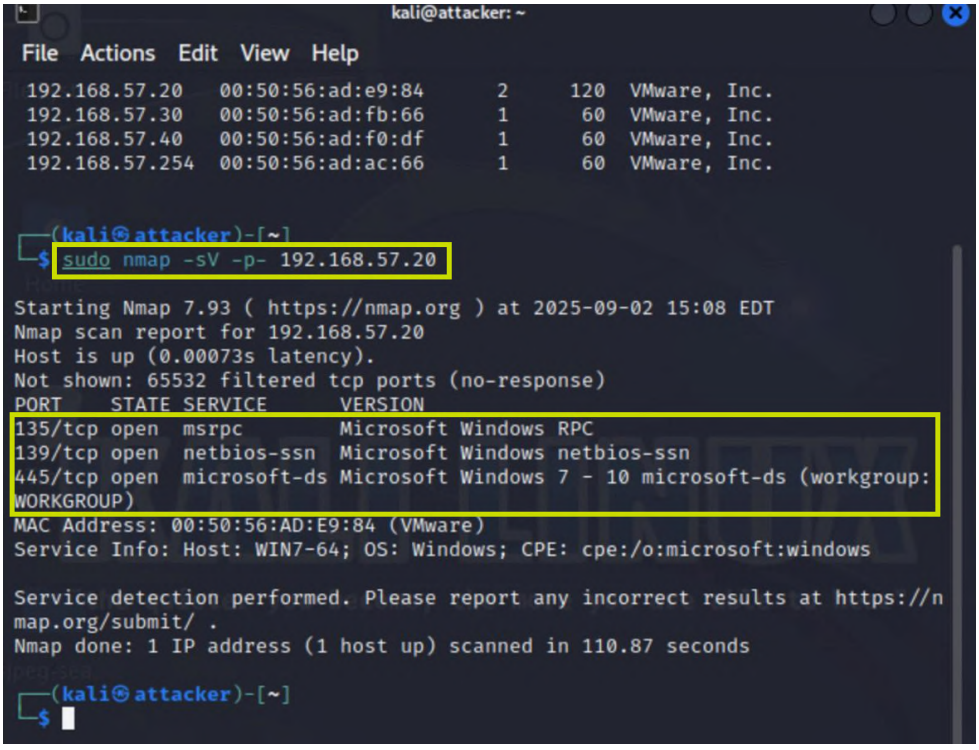


Figure 1.3.1 – Nmap scan results for 192.168.57.20

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 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:

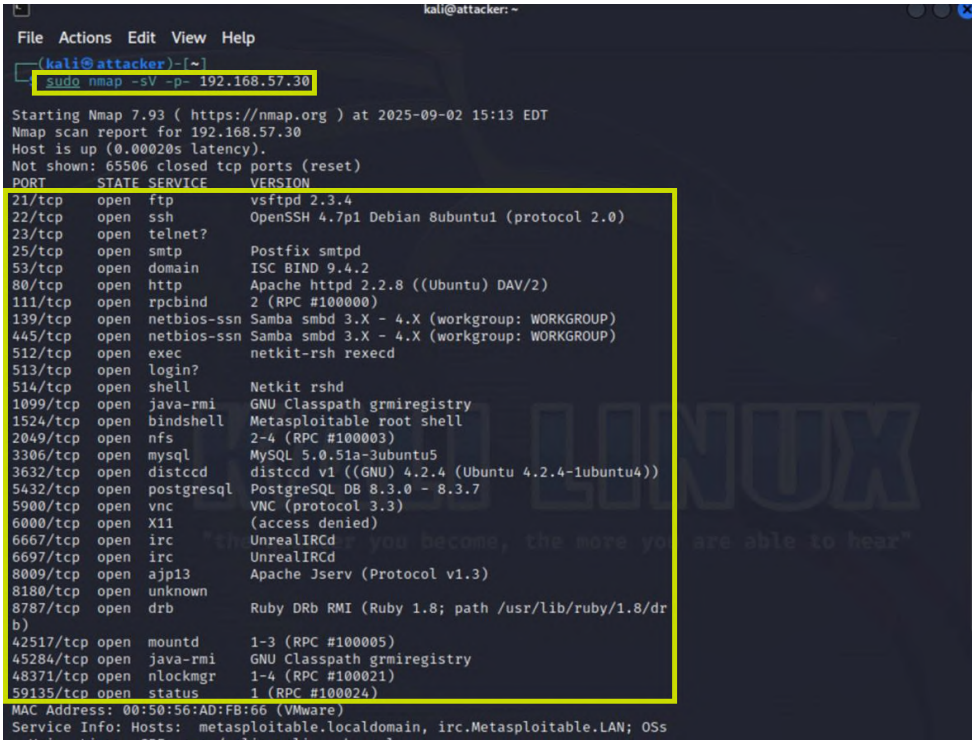


Figure 1.3.2 – Nmap scan results for 192.168.57.30

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:

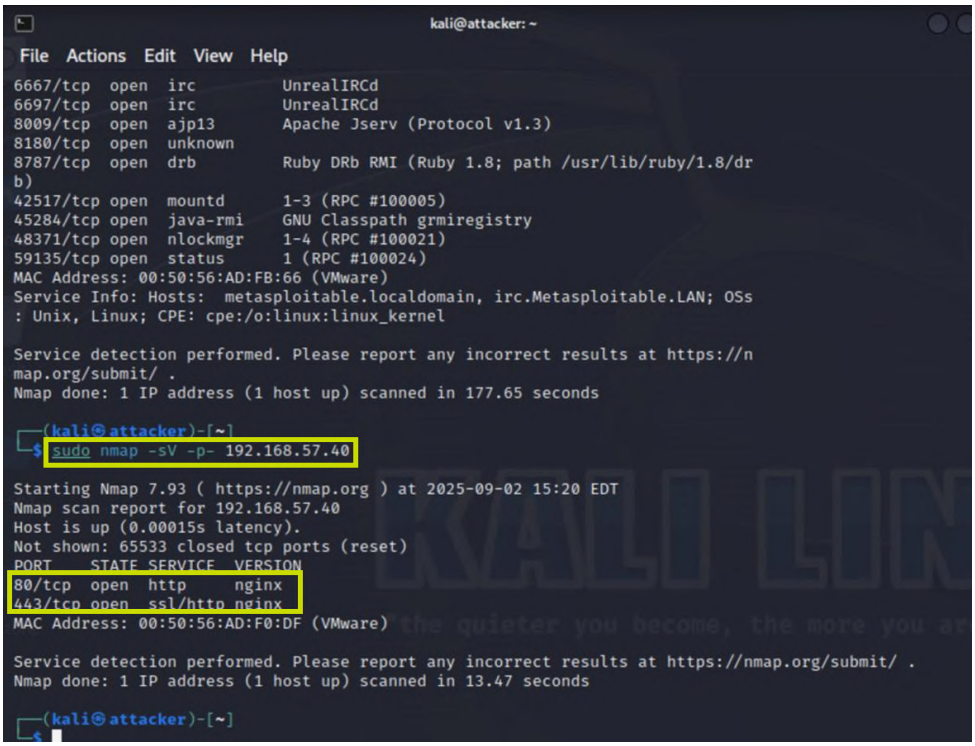


Figure 1.3.3 – Nmap scan results for 192.168.57.40

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:

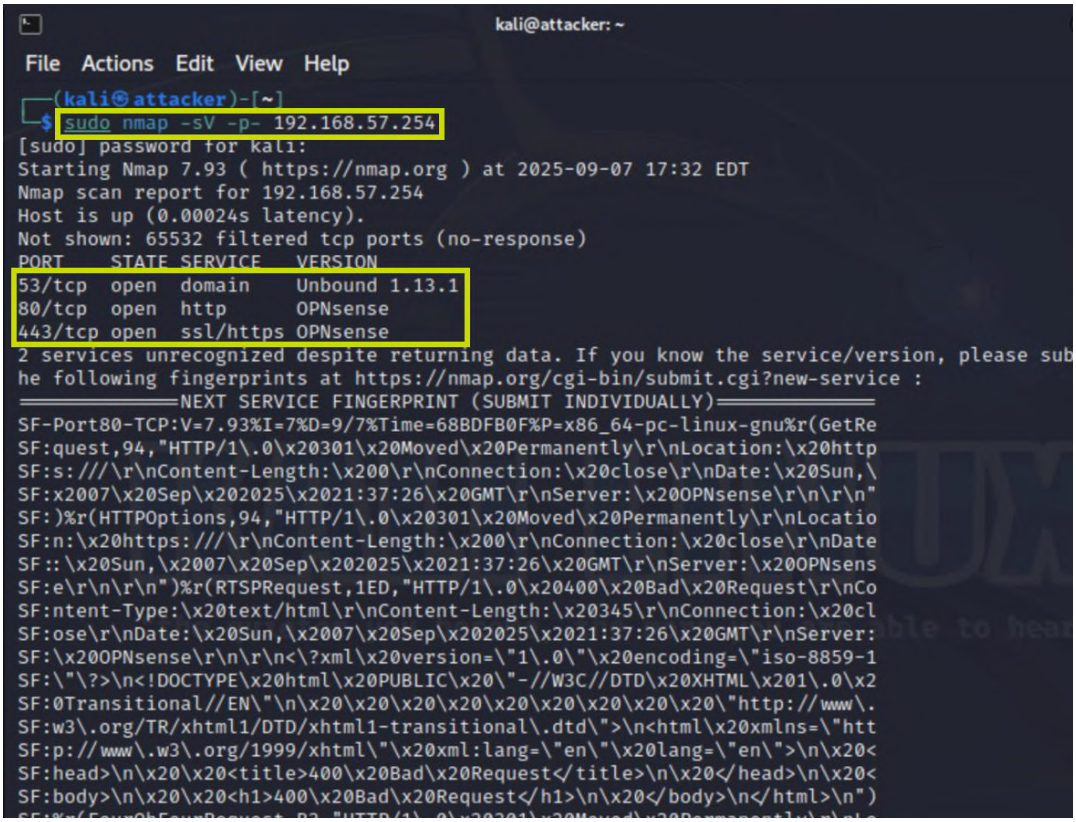


Figure 1.3.4– Nmap scan results for 192.168.57.254

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.

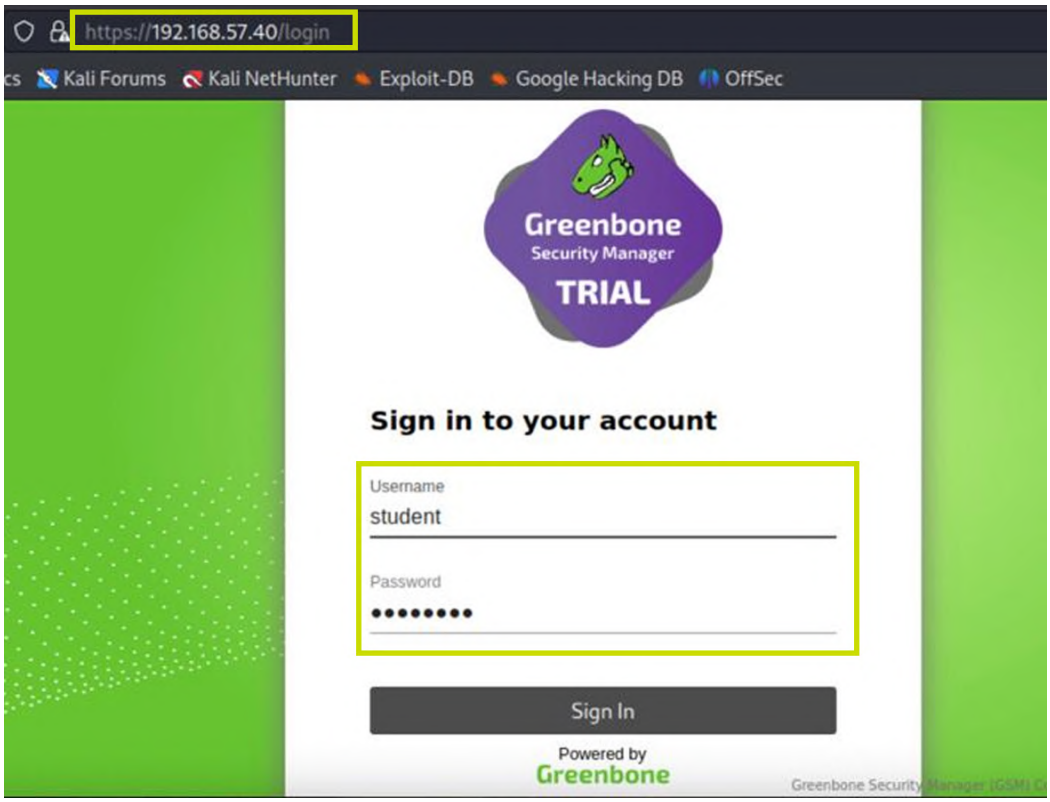


Figure 2.1 – Greenbone login interface

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.

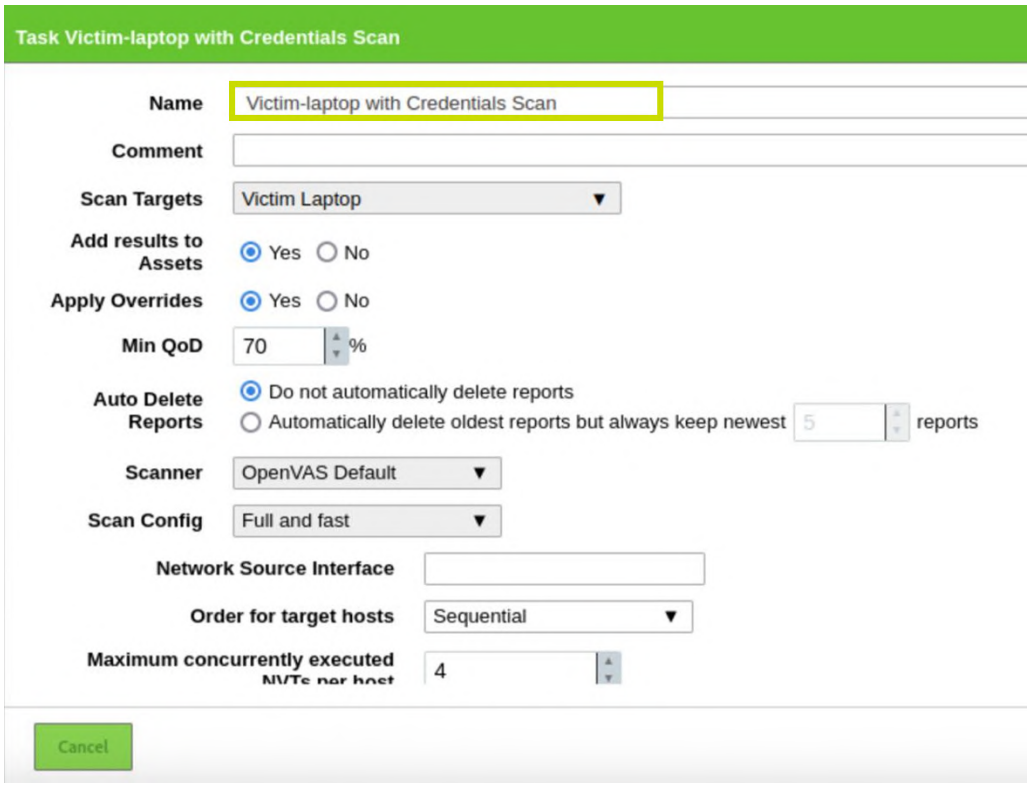


Figure 2.2 – Scan configurations for 192.168.57.20 with credentials

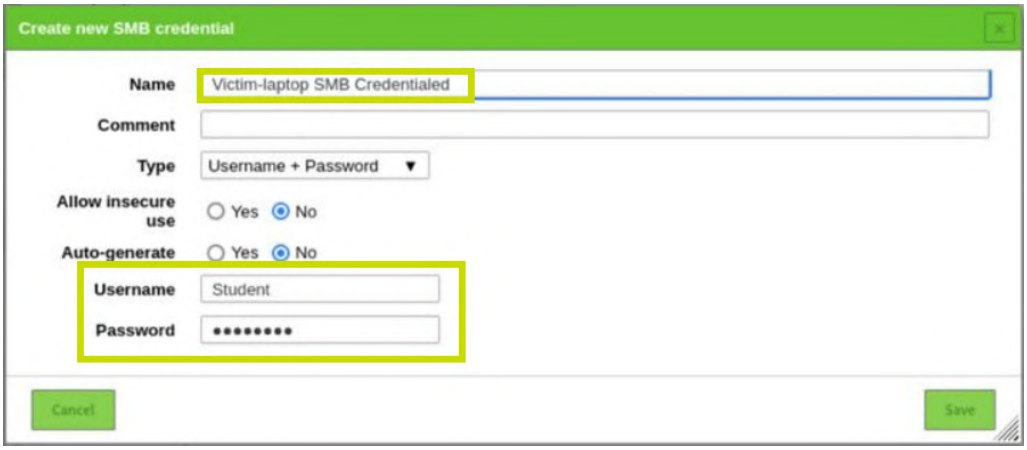


Figure 2.2.1 – Adding credentials for victim laptop

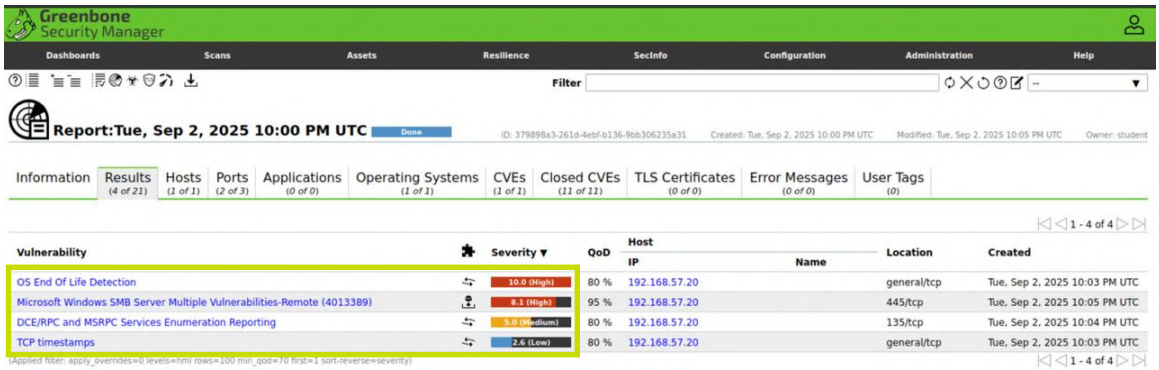


Figure 2.2.2 – Scan results for 192.168.57.20 with credentials

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 credentialed access.

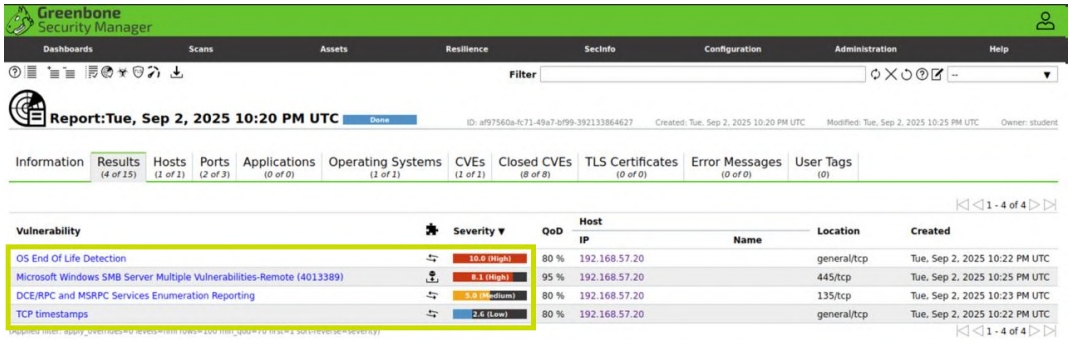


Figure 2.3 – Non-credentialed scan results for 192.168.57.20

the results were nearly identical to those obtained from the credentialed 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.

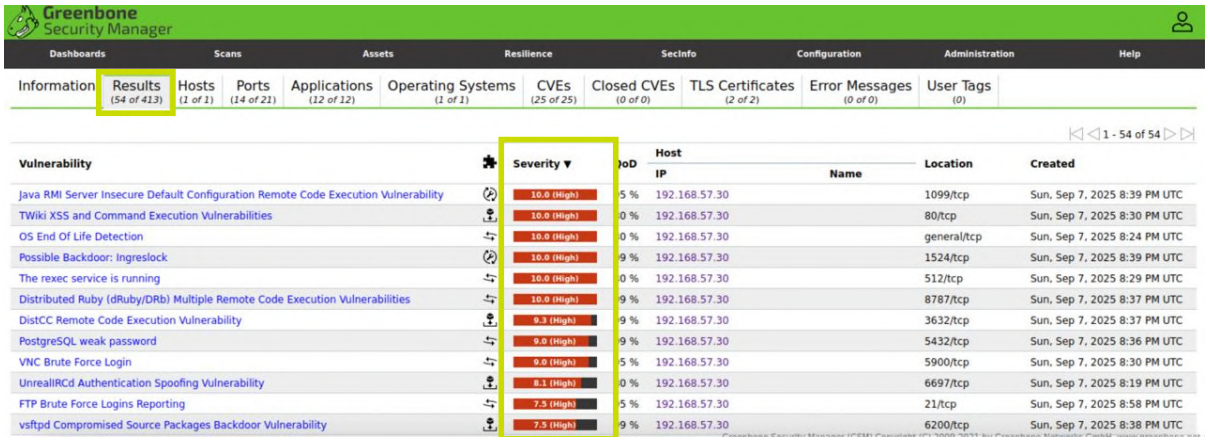


Figure 2.4 – First page of scan results for 192.168.57.30

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.

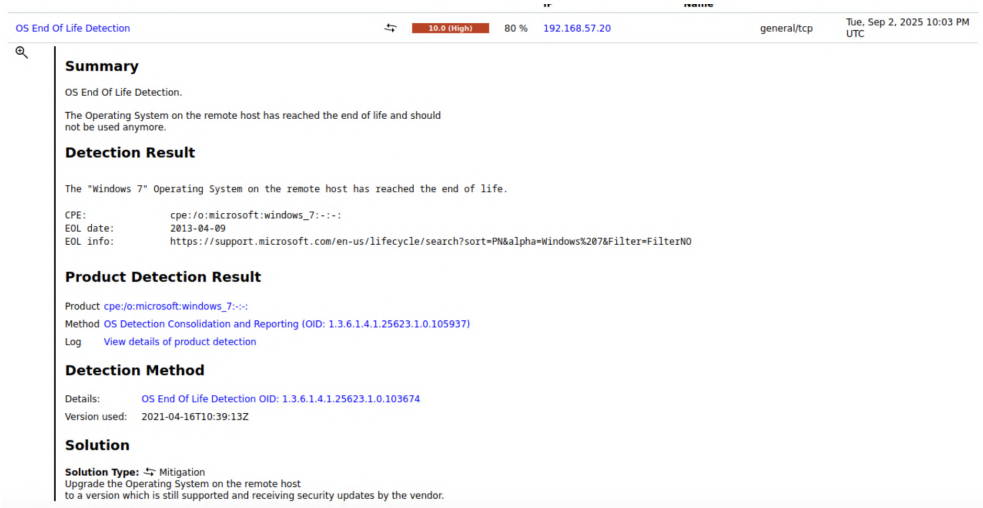


***Figure 2.3.1**: OS End Of Life Detection vulnerability*

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.

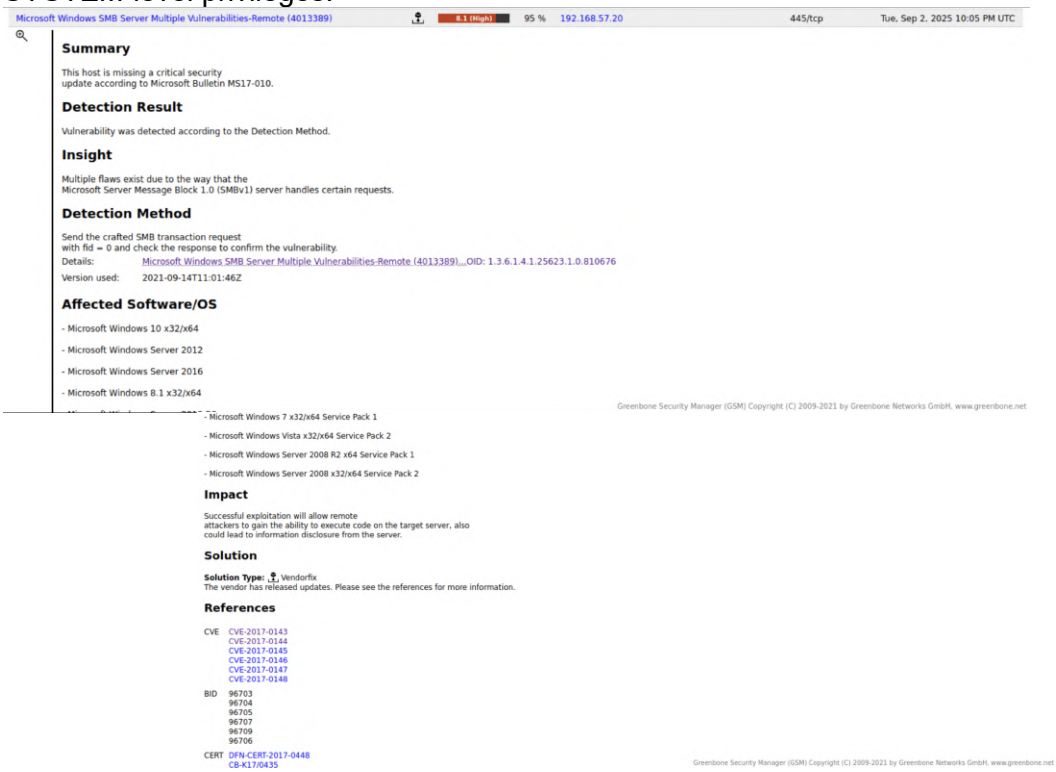


***Figure 2.3.1**: OS End Of Life Detection vulnerability*

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.

***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.

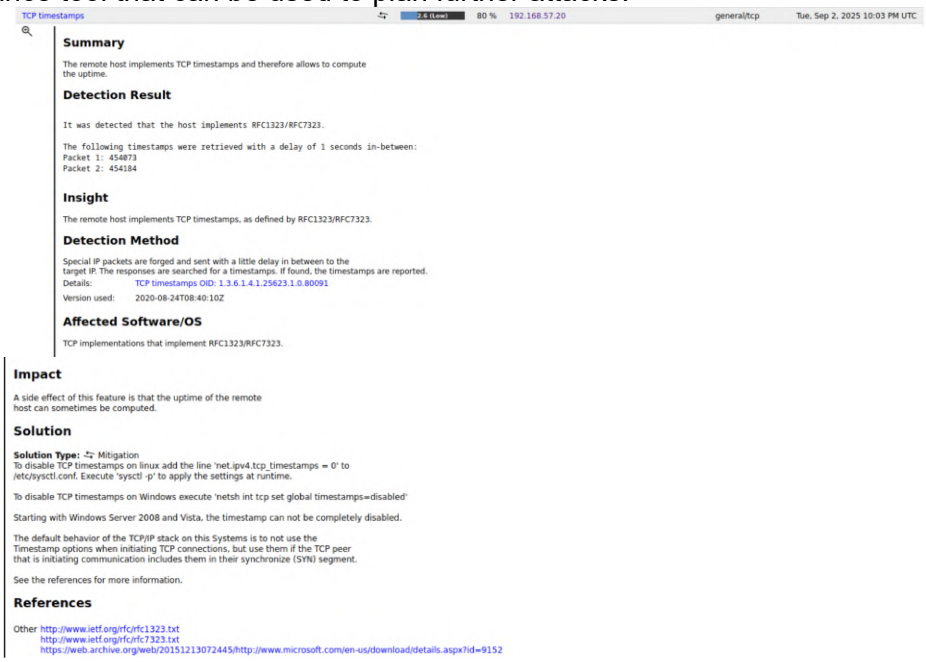


***Figure 2.3.1**: OS End Of Life Detection vulnerability*

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.

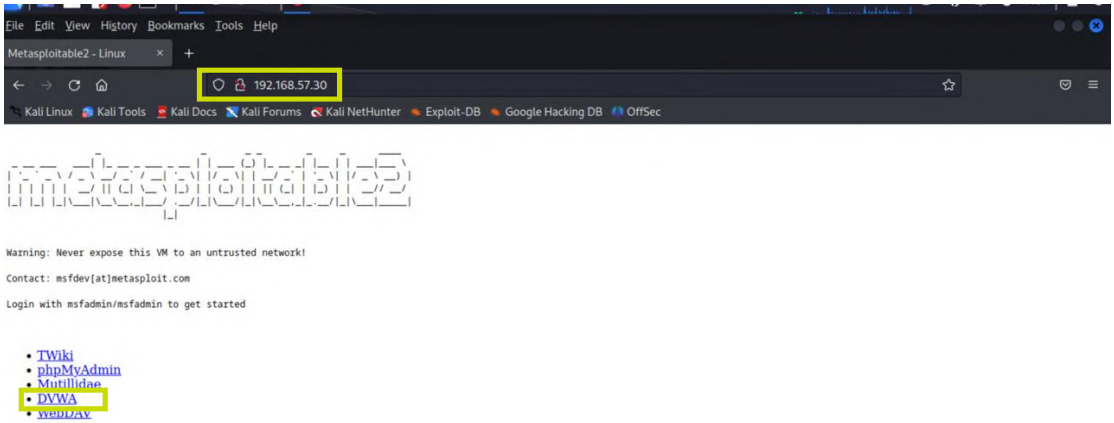


Figure 3.1 – Metasploitable2 web interface on 192.168.57.30

Step 3.2 – Logging into DVWA

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

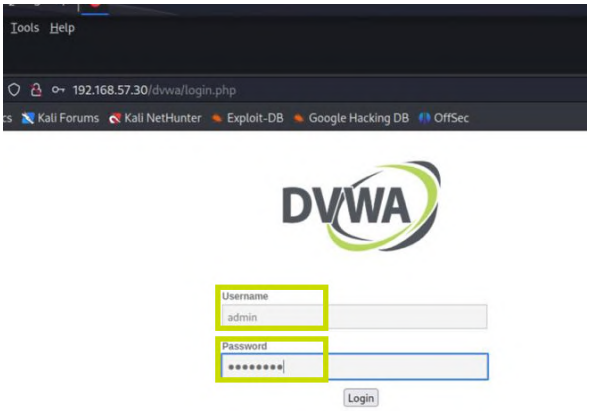


Figure 3.2 – DVWA login screen with credentials entered

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

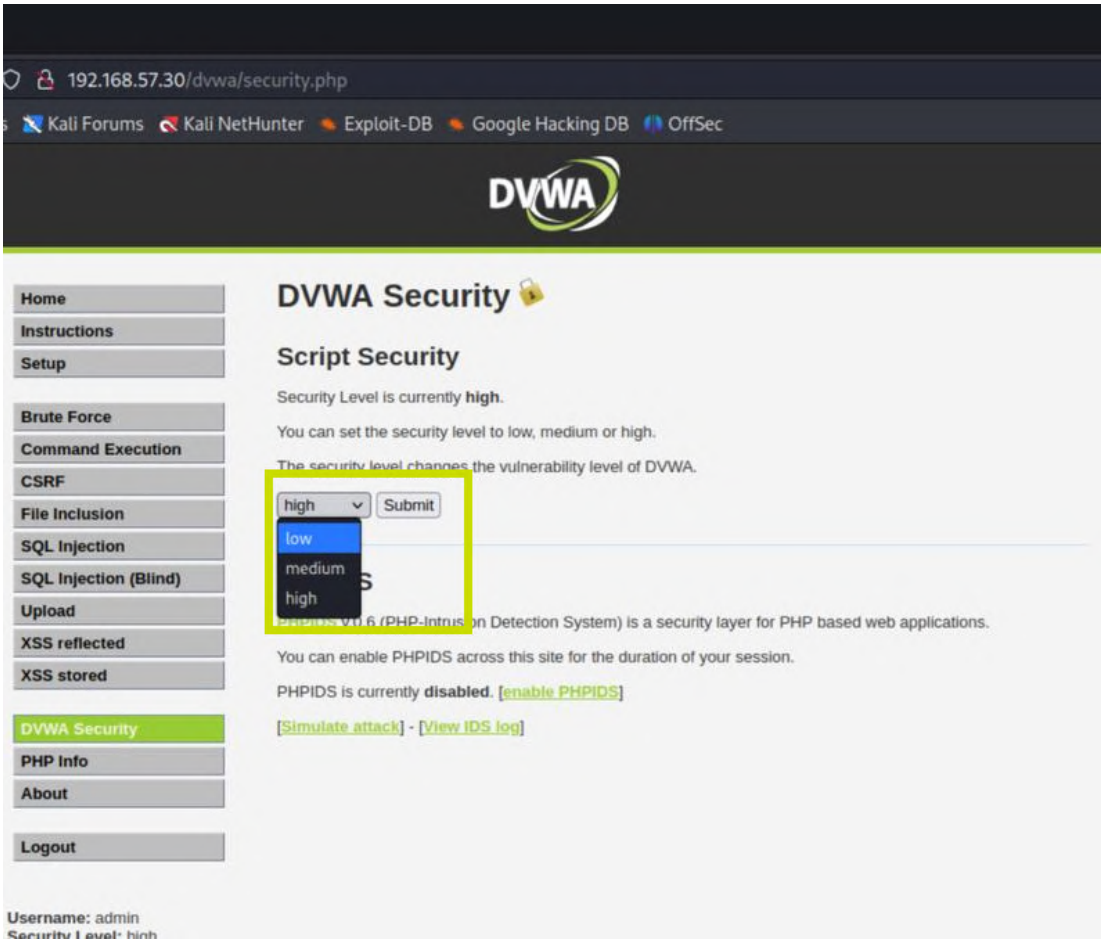


Figure 3.3 – DVWA security level set to Low

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

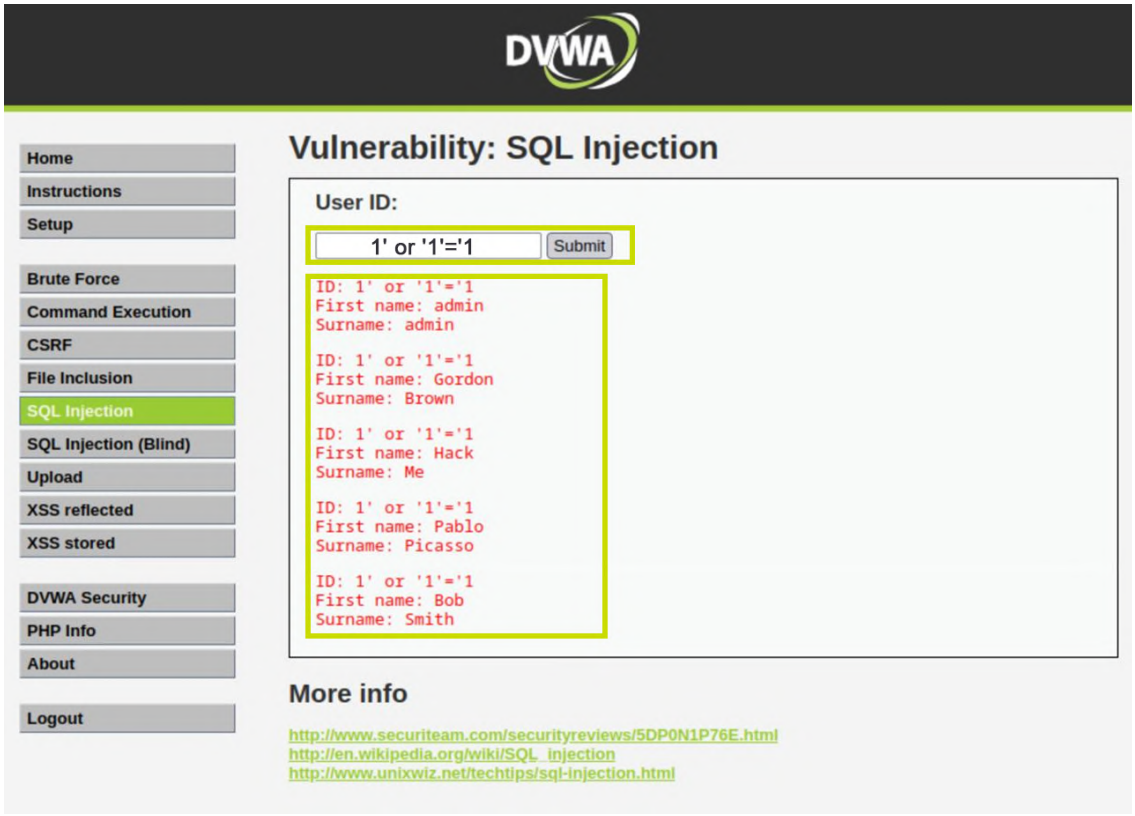


Figure 3.4 – SQL Injection payload entered and successful result displayed

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.

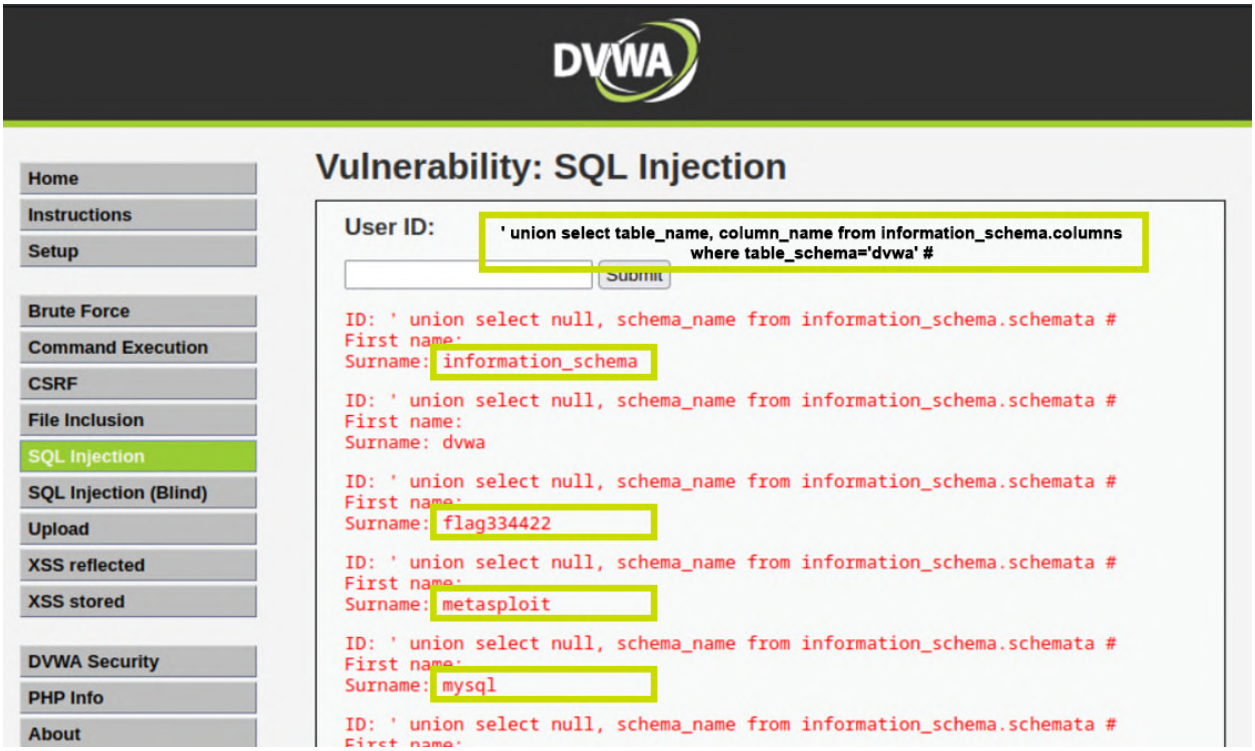


Figure 3.5 – SQL Injection - Database Enumeration via information_schema

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:

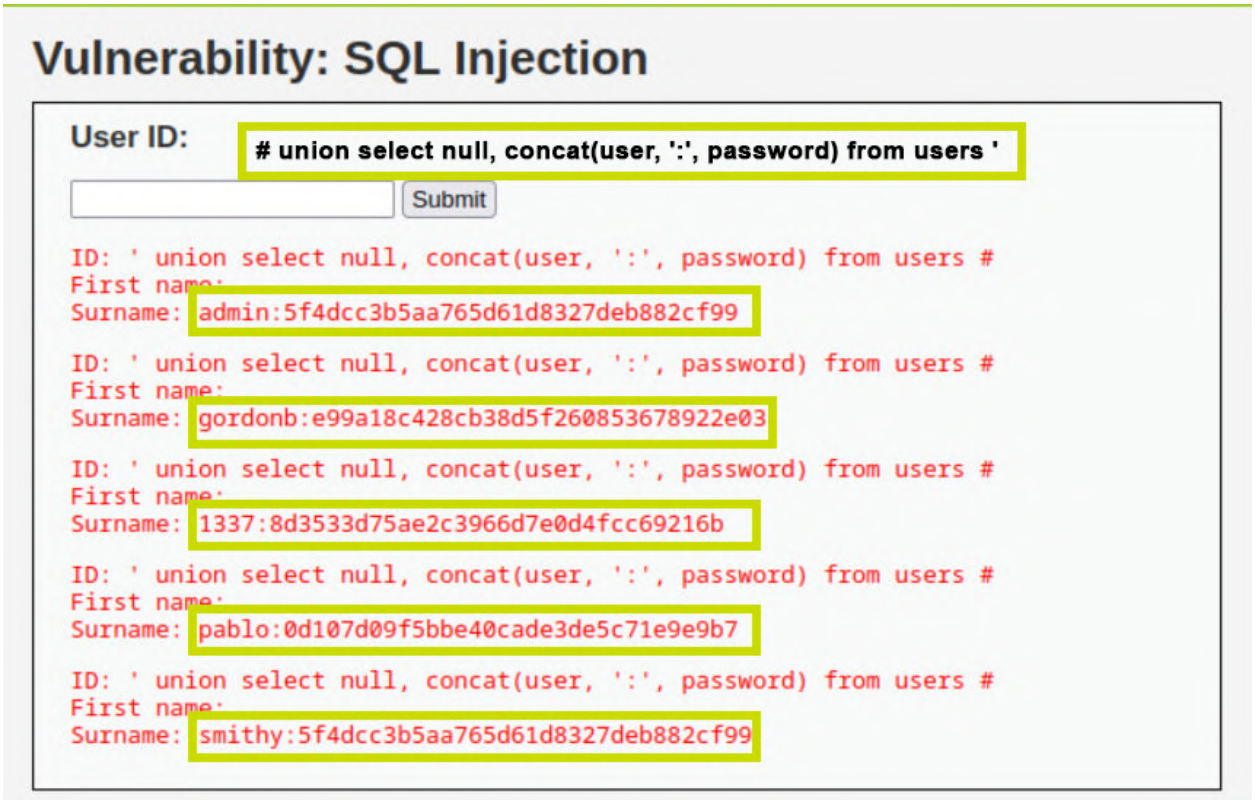


Figure 3.5 –Extracted User Credentials from DVWA via SQL Injection

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.

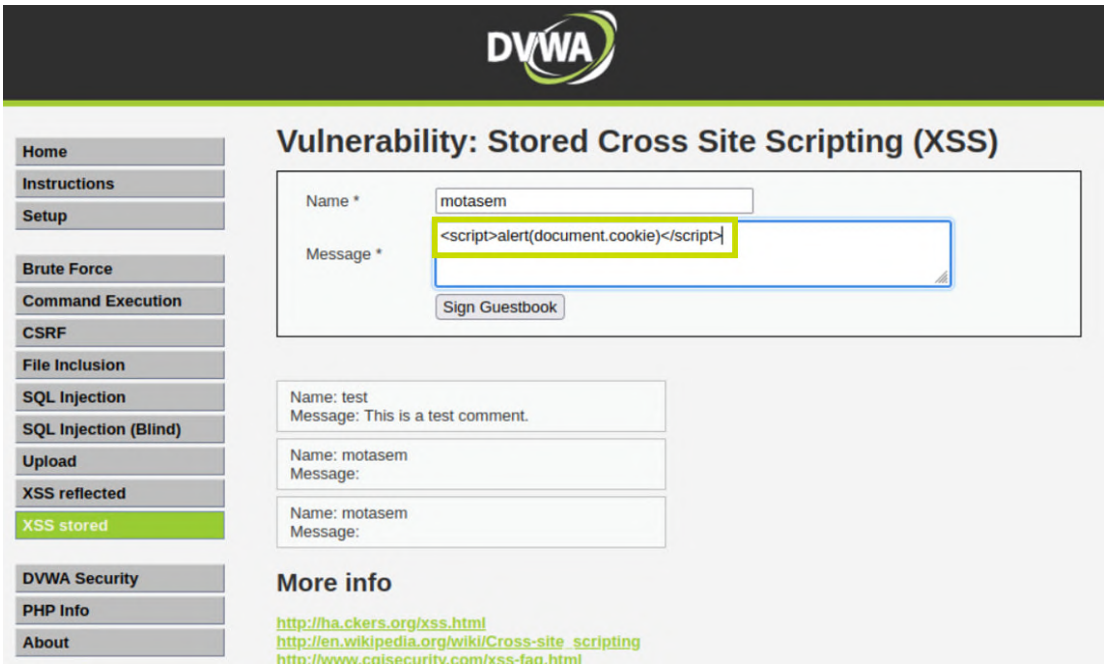
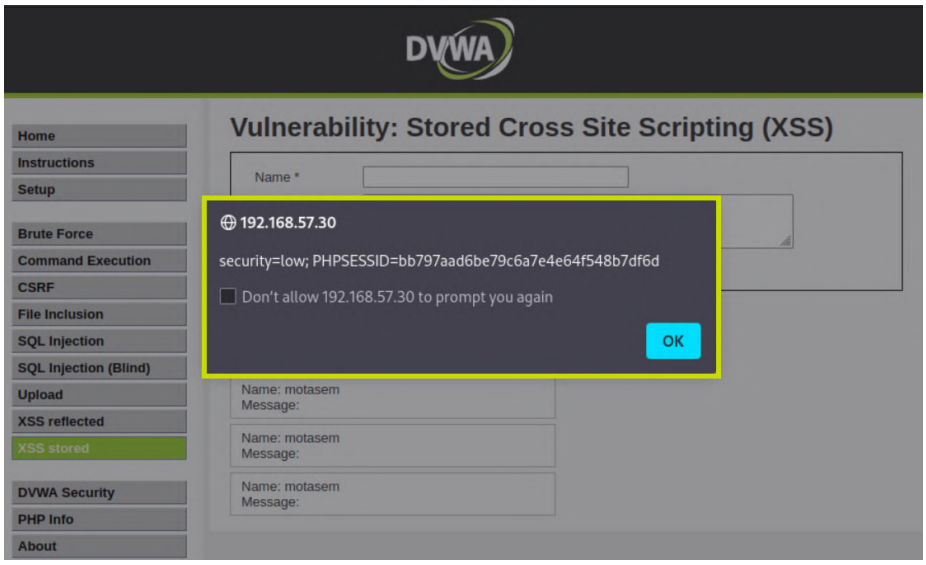


Figure 3.5 –Extracted User Credentials from DVWA via SQL Injection

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.

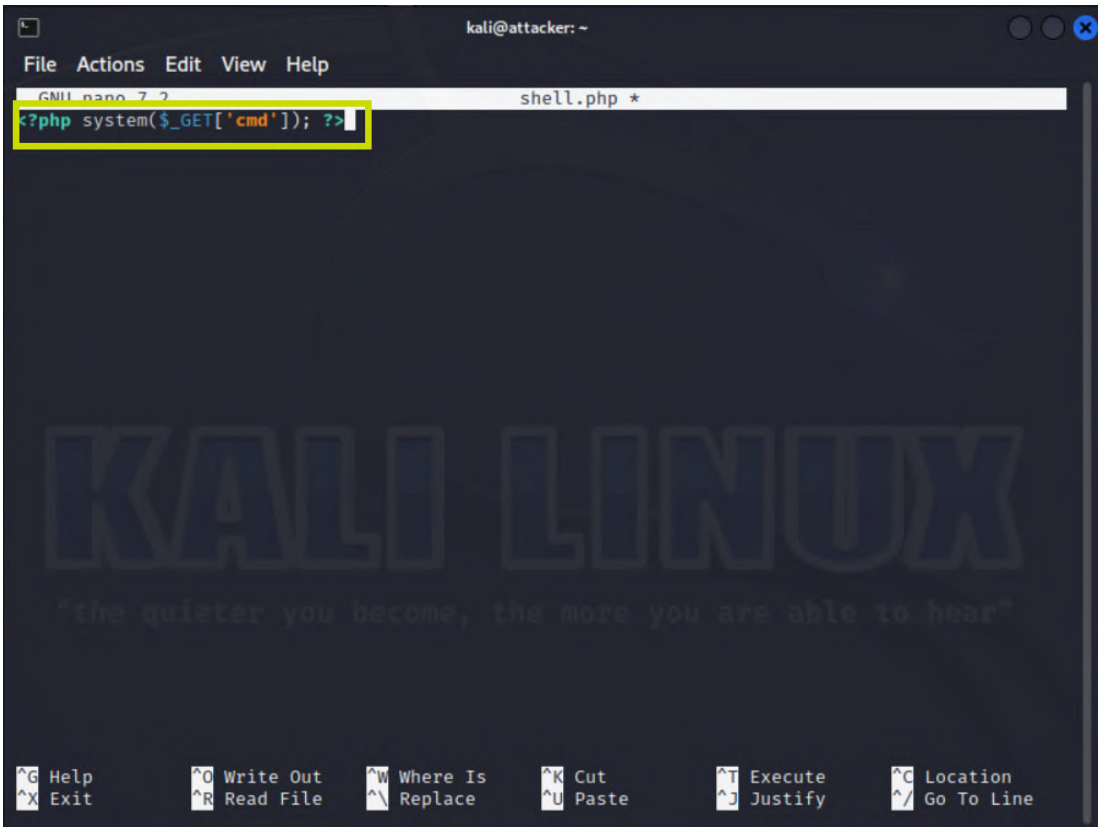


Figure 3.6.1 – Creating PHP shell using the nano editor

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.

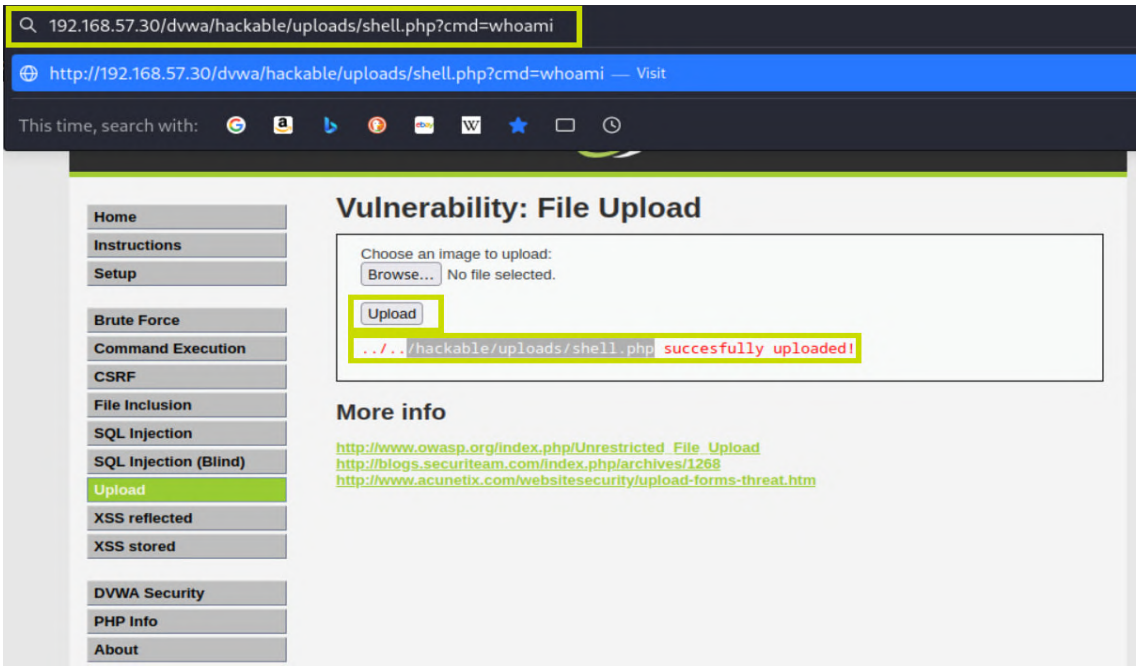


Figure 3.6.2 – Uploading Malicious PHP Webshell through DVWA File Upload

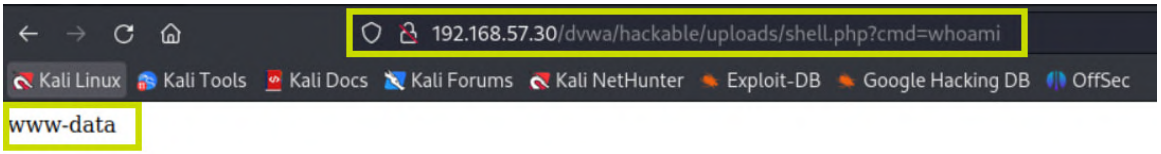


Figure 3.6.3 – Execution Output retrieved via Webshell.

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

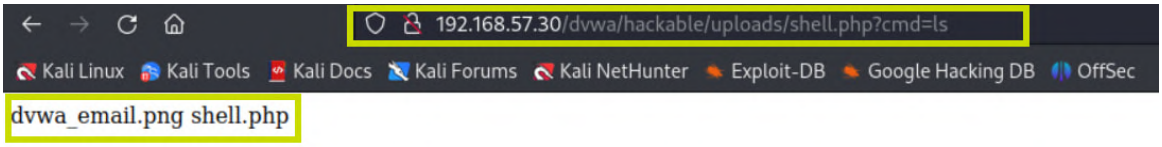


Figure 3.6.4 – Execution Output retrieved via Webshell.

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:

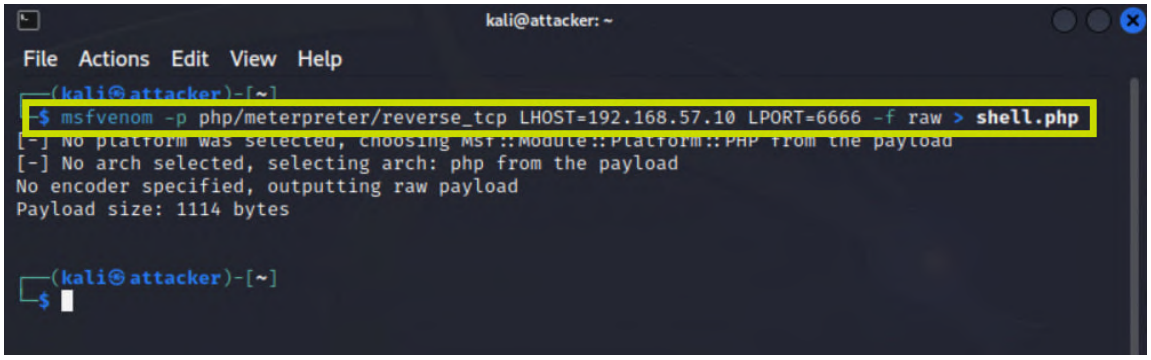


Figure 3.7.1 – Generating PHP Reverse Shell Payload using Msfvenom

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

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

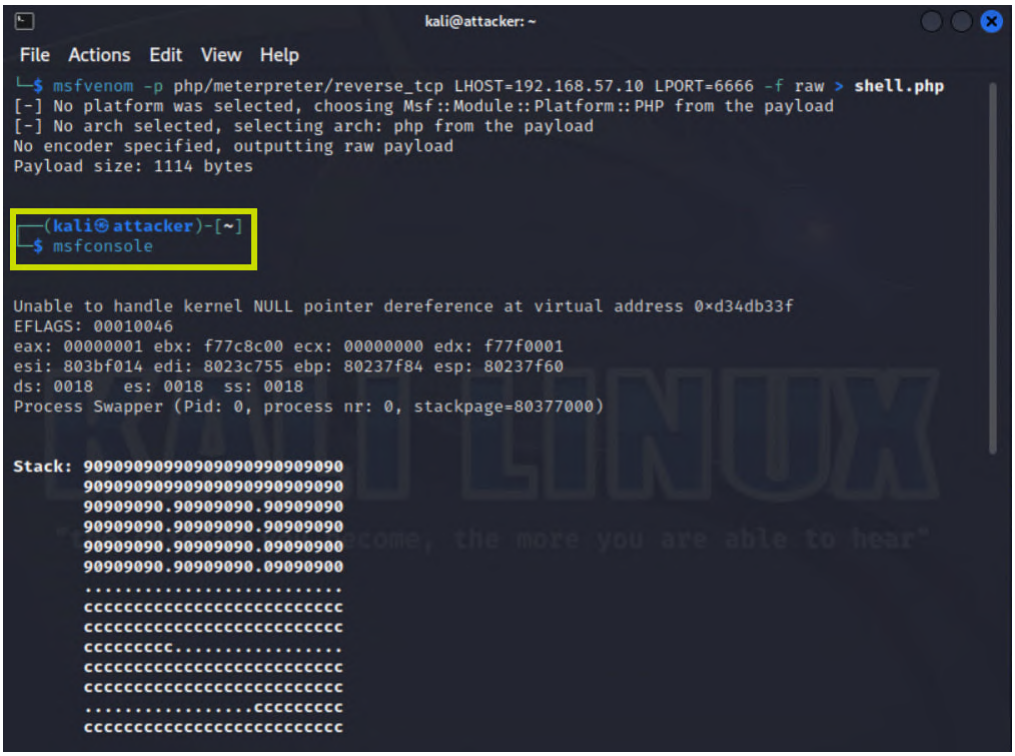
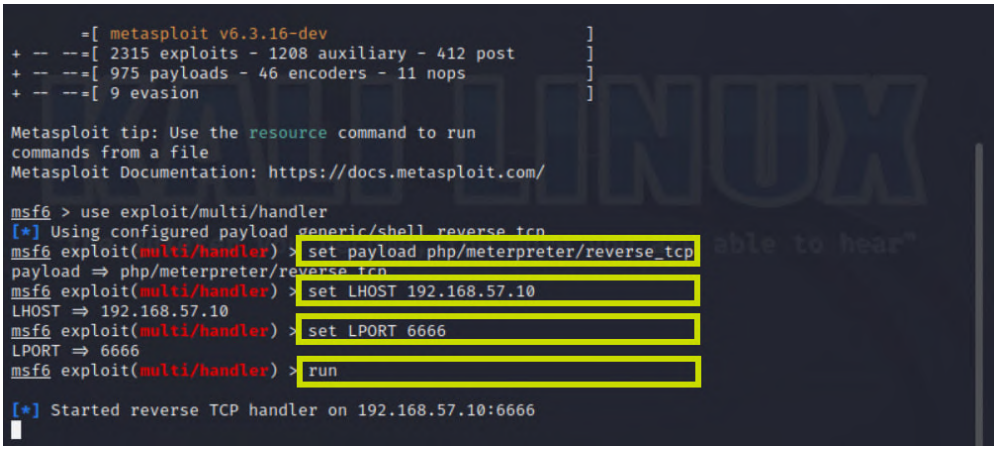


Figure 3.7.3 – Launching Metasploit Console on (pen-tester) Machine

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



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

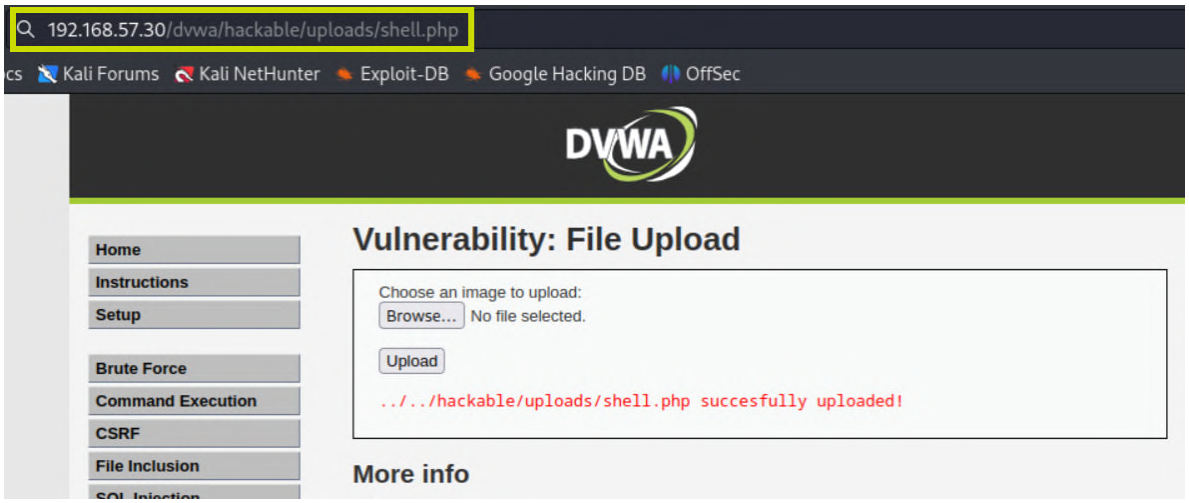
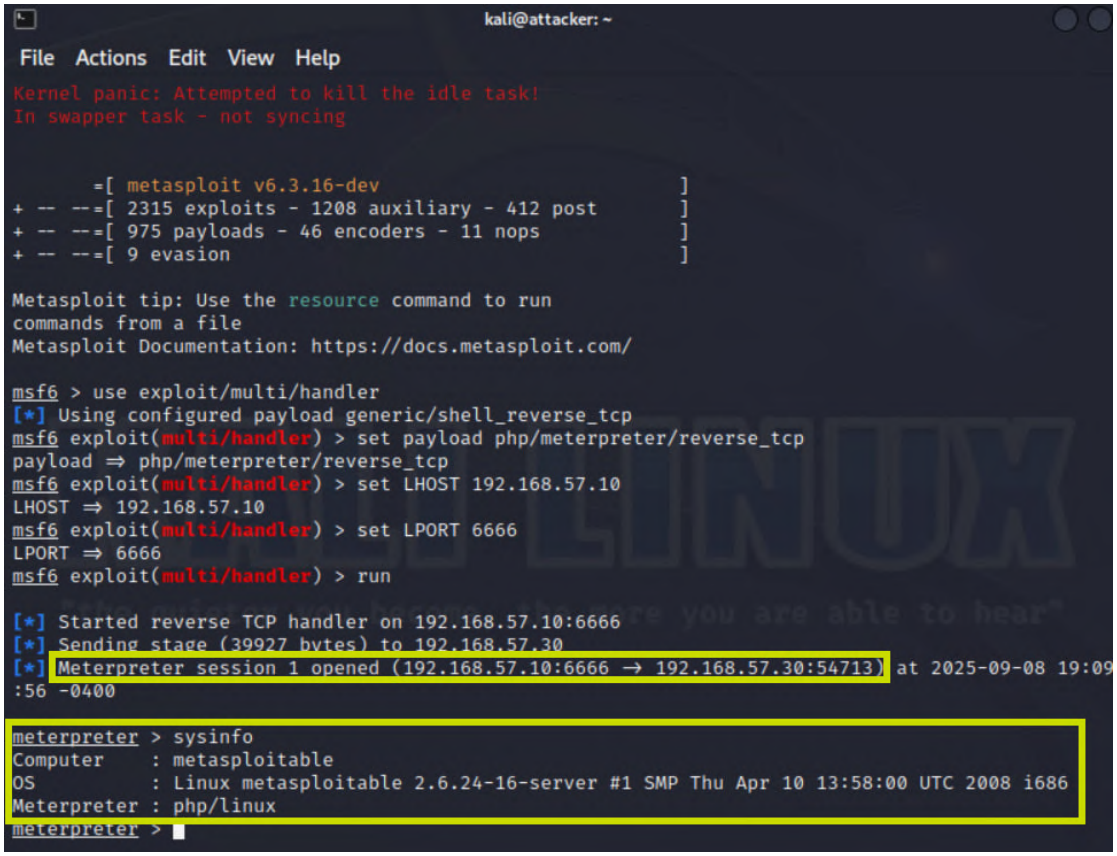


Figure 3.7.4 – Triggering the Webshell via Browser to Initiate Reverse Connection

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



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).

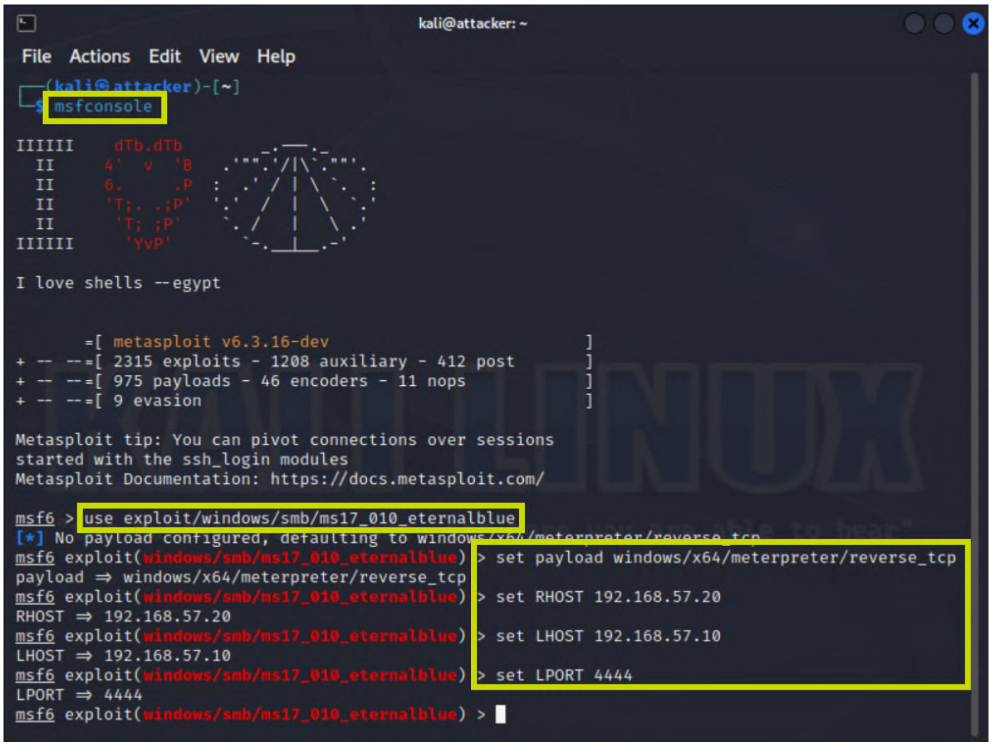
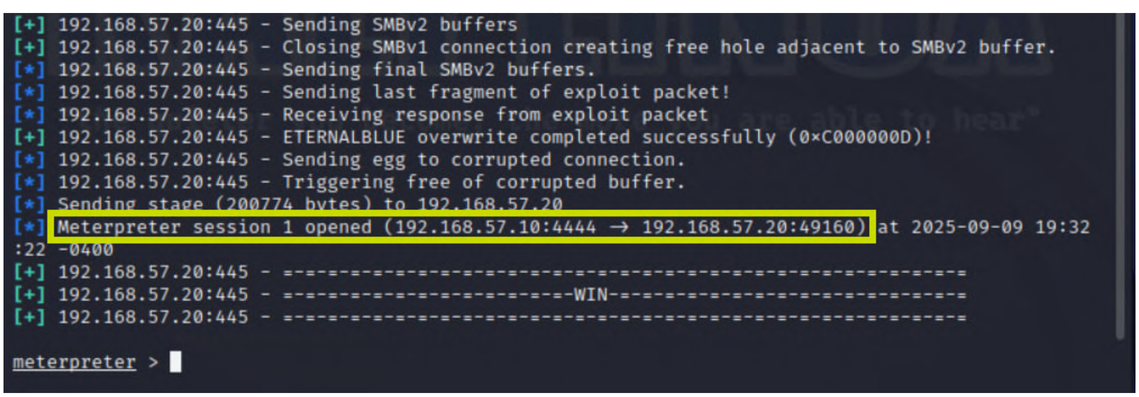


Figure 4.1 – Successful Meterpreter Session via EternalBlue Exploit

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



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.

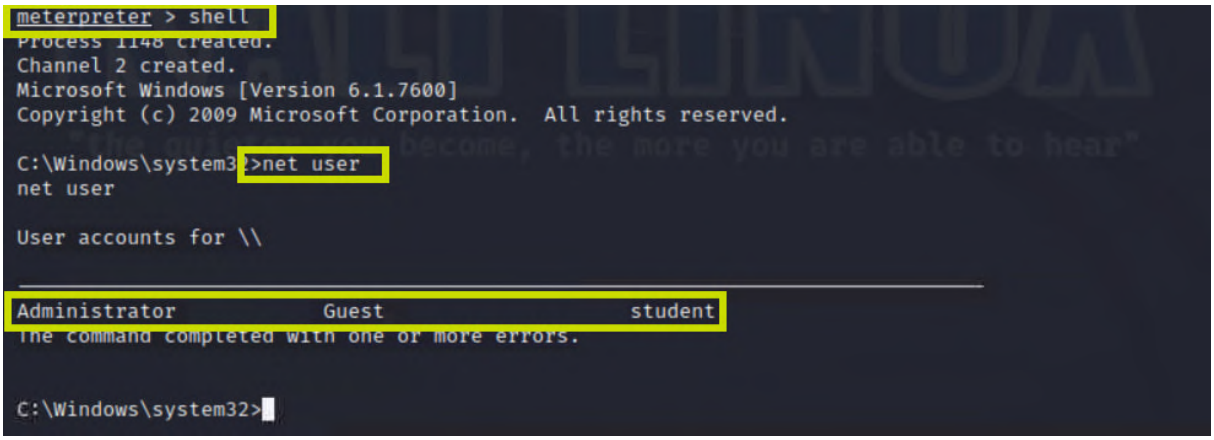


Figure 4.2.1 – Enumerating Local Users via Windows Shell

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.

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.

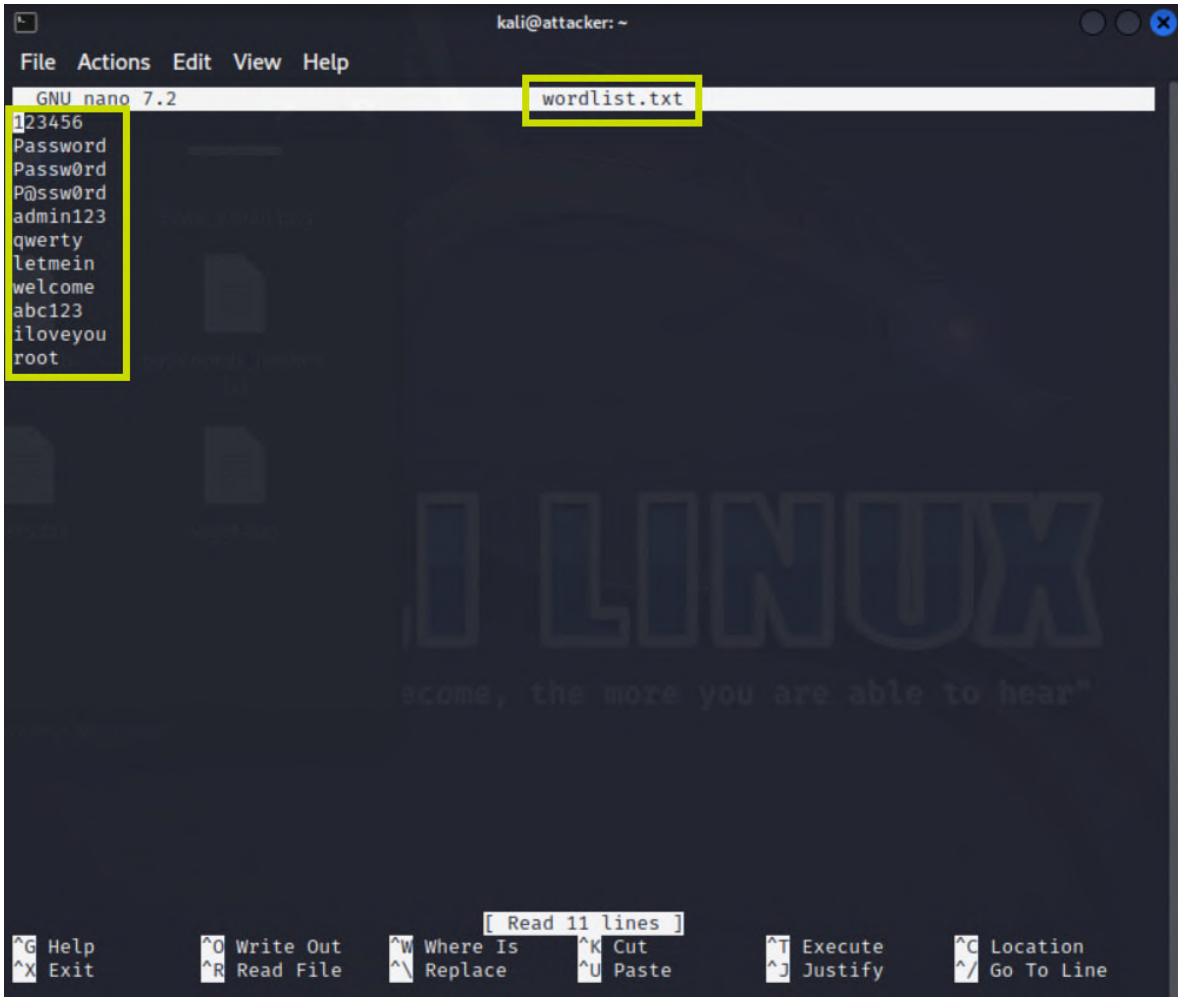


Figure 4.3– Custom Password Wordlist

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

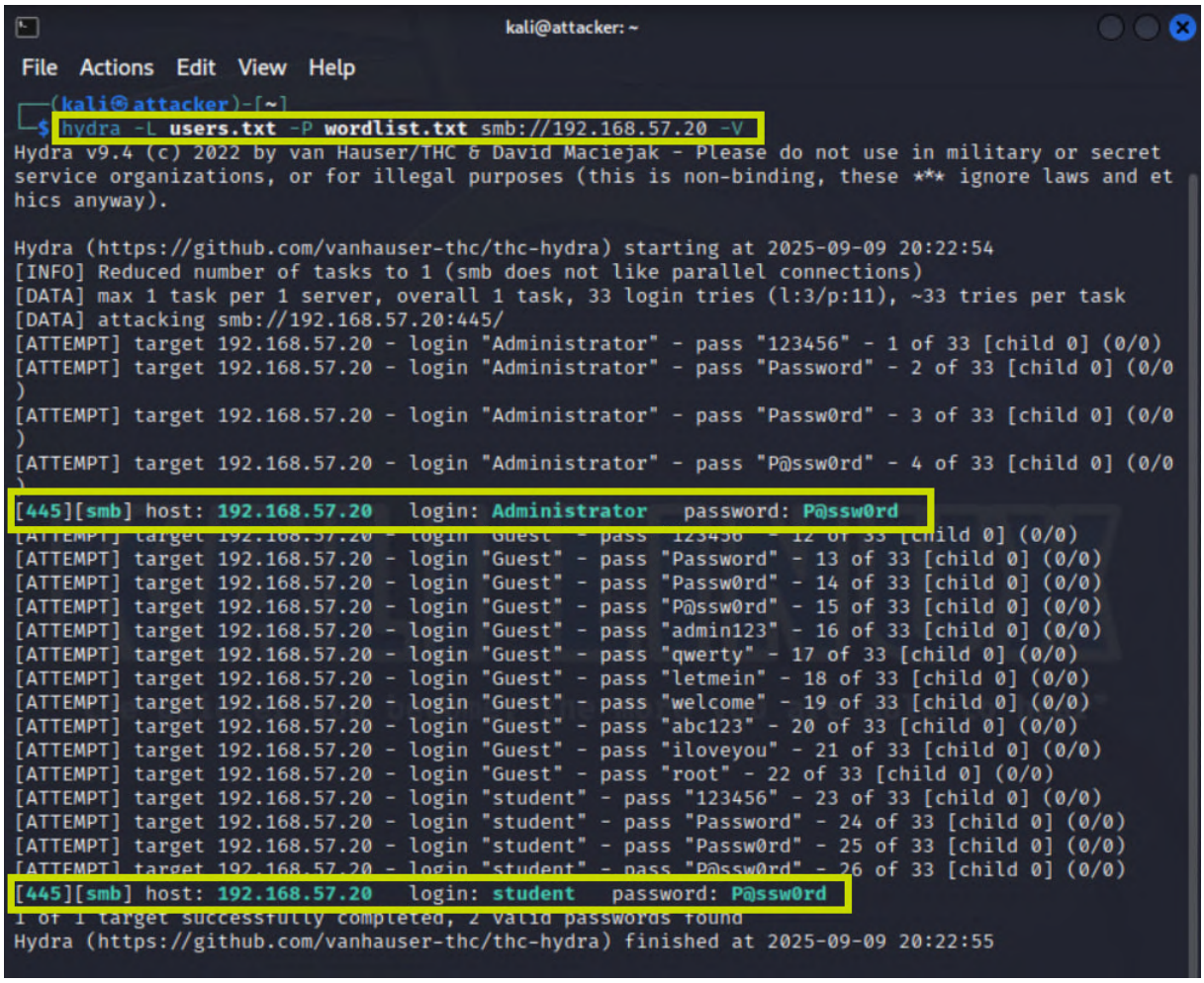


Figure 4.4– Hydra SMB Attack Output Showing Cracked Credentials

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.

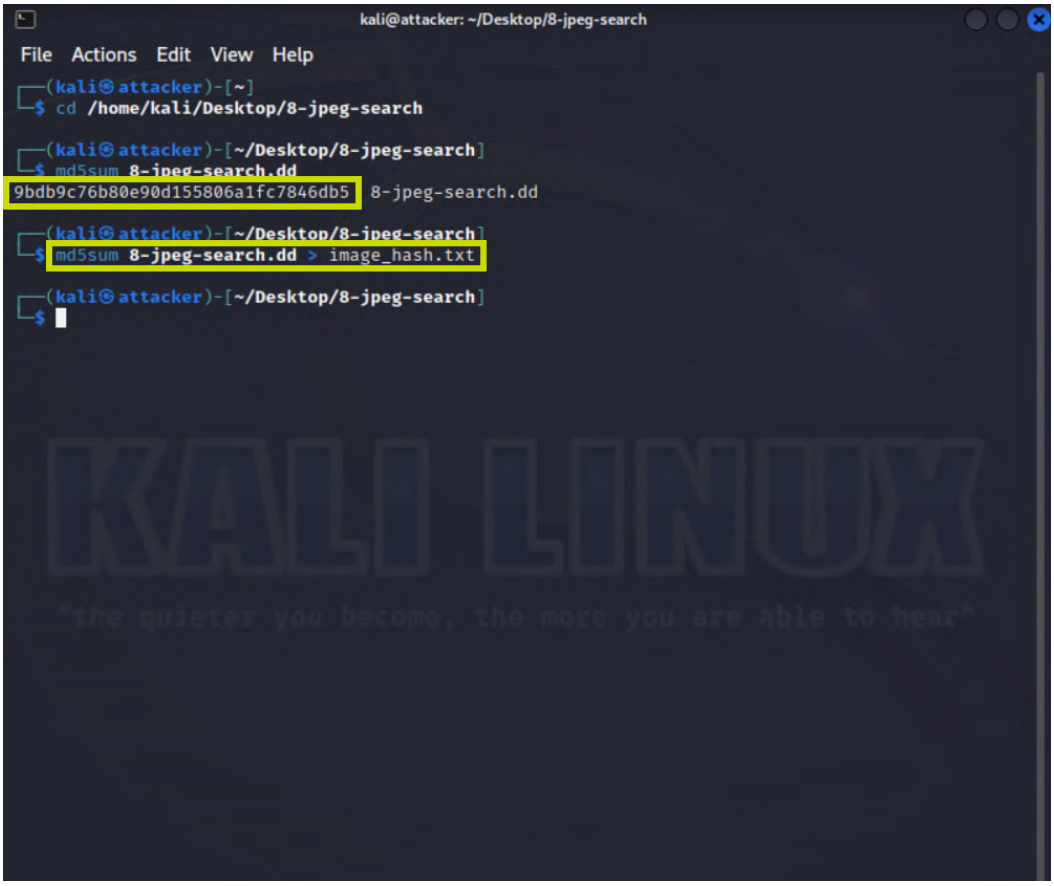


Figure 5.1 – MD5 Hash Generated and Saved to File

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.

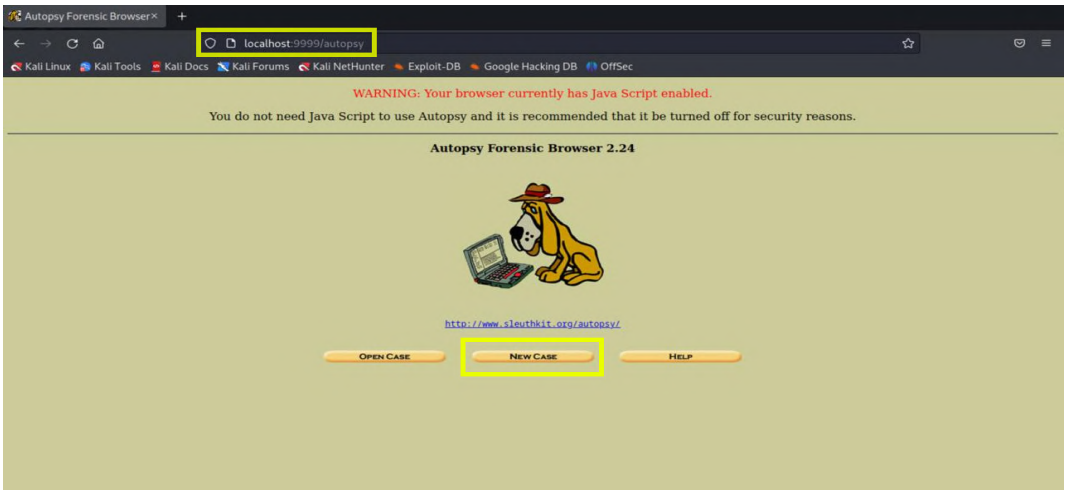


Figure 5.2.1 – Initial Autopsy Interface

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

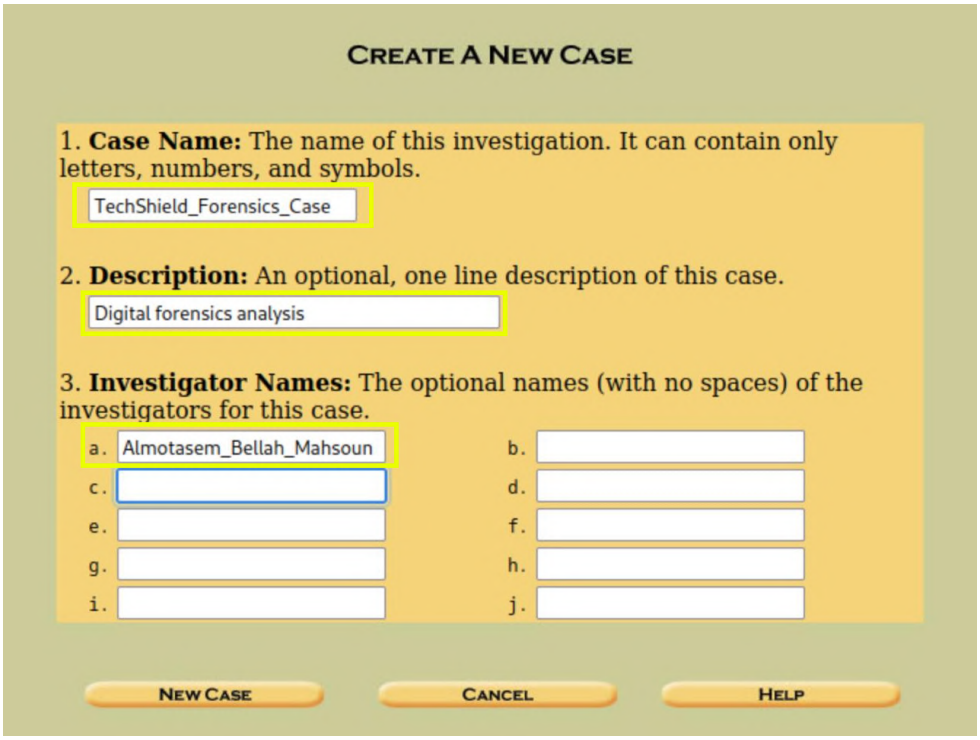


Figure 5.2.2 – Autopsy Case Creation

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.

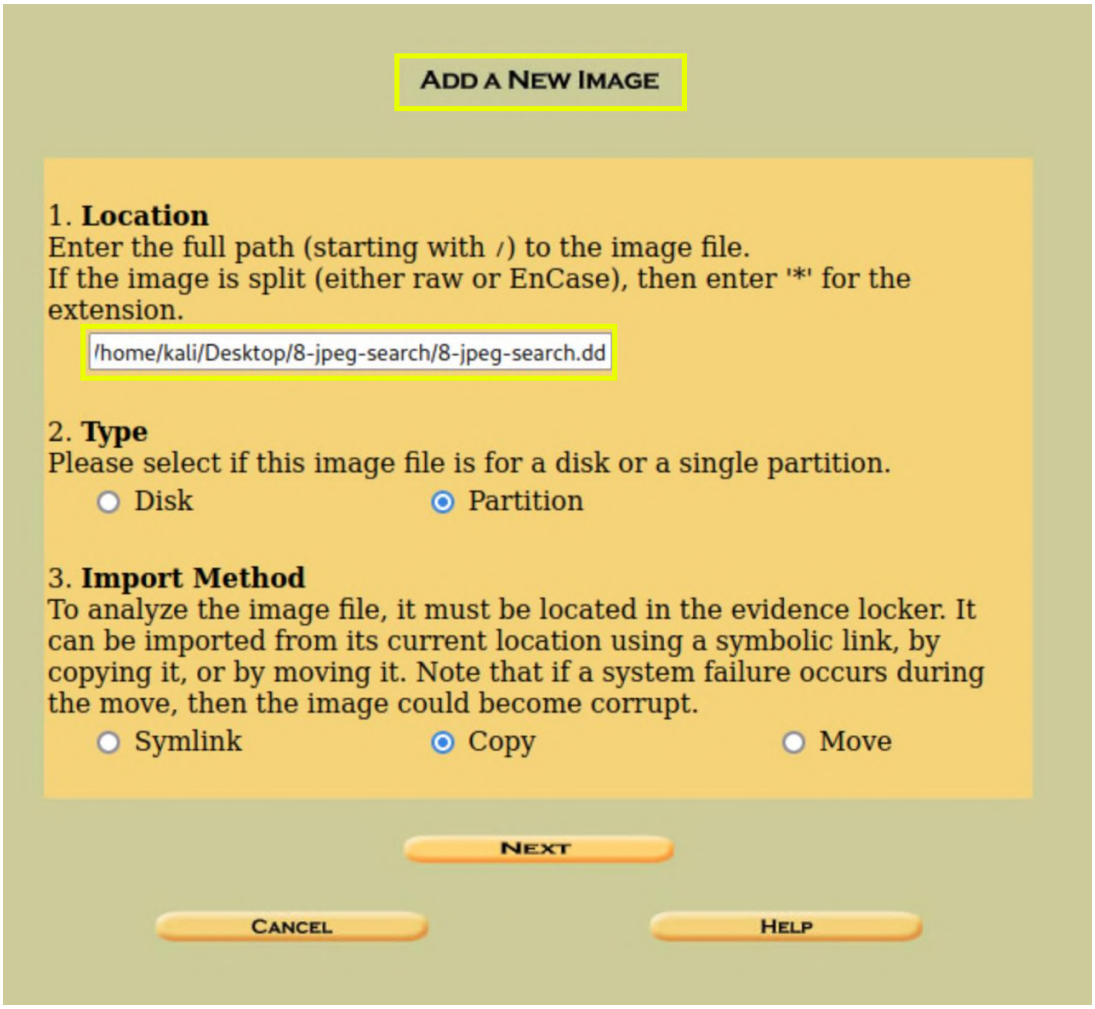


Figure 5.2.2 – Forensic Image Added

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.

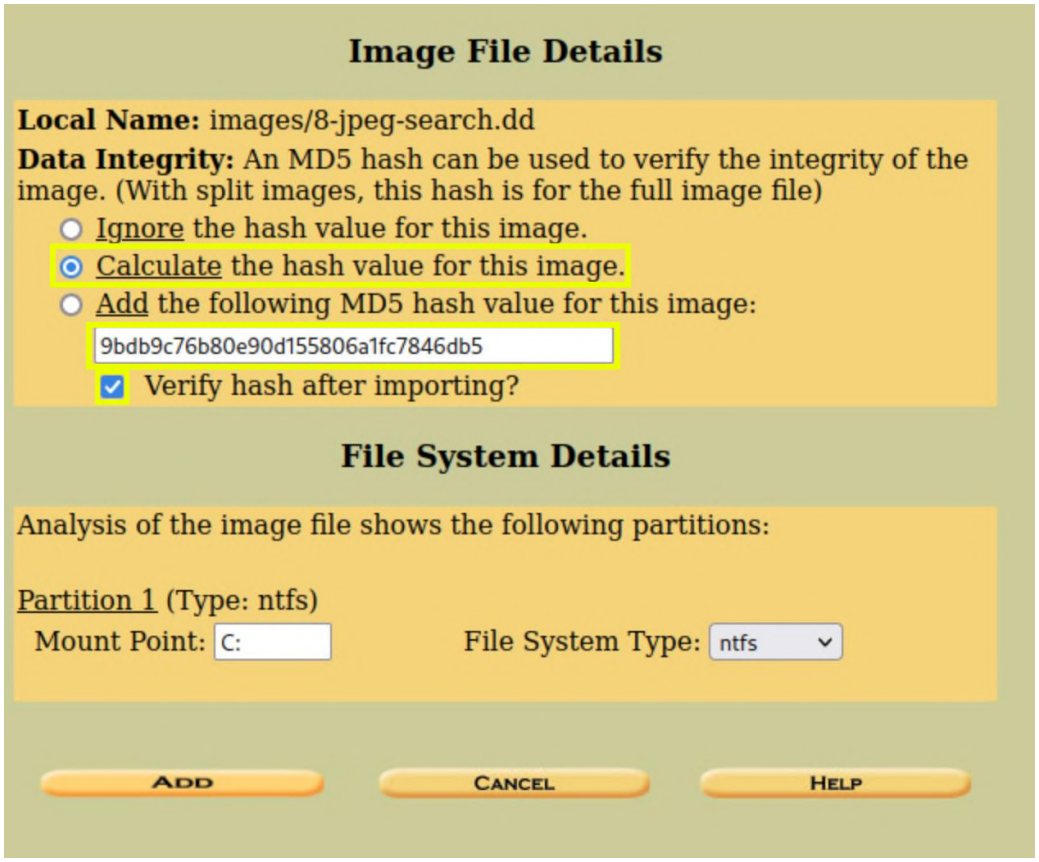


Figure 5.3.1 – Hash Input for Calculation and Verification

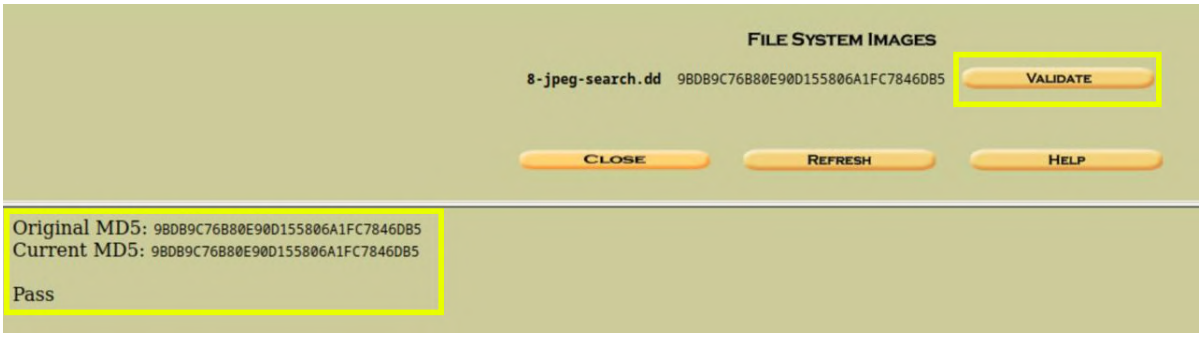


Figure 5.3.2 – MD5 Hash Verified in Autopsy

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.

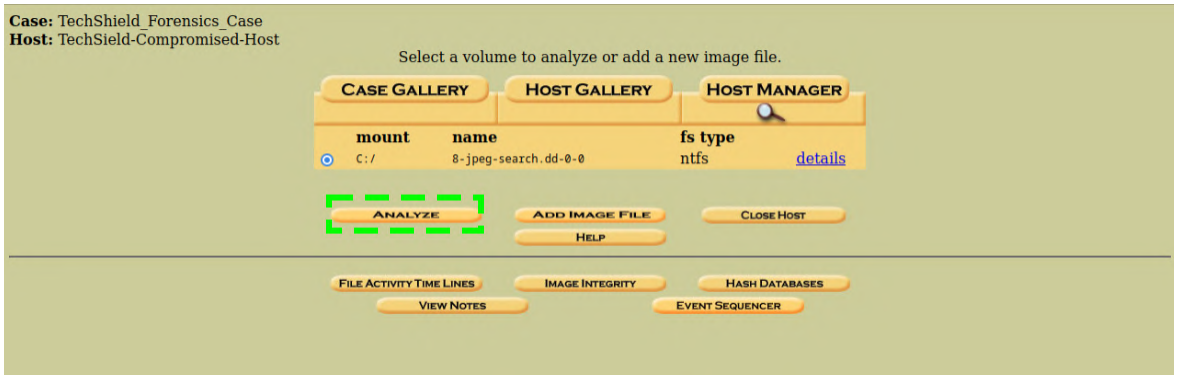


Figure 5.4.1 – MD5 Hash Verified in Autopsy

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.

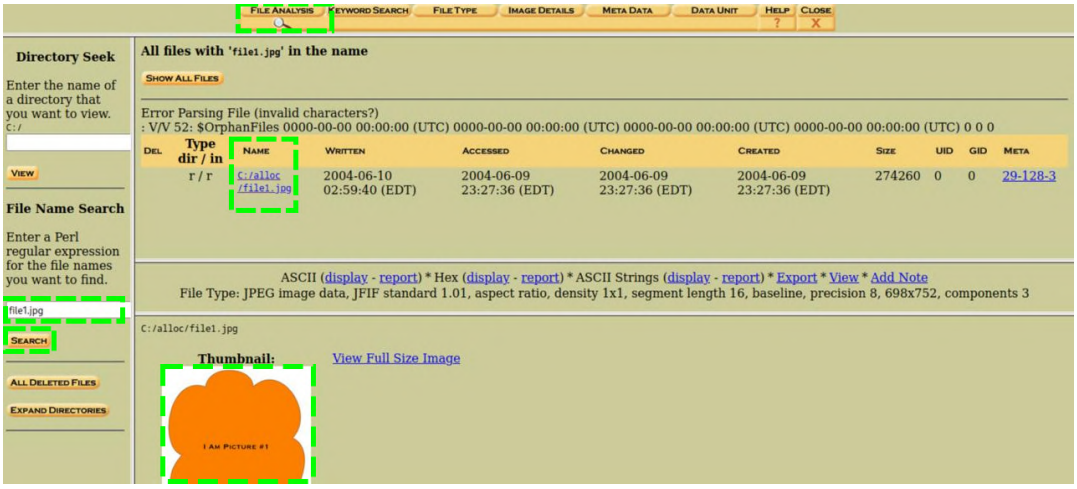


Figure 5.4.2 – file1.jpg Found via Keyword Search

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

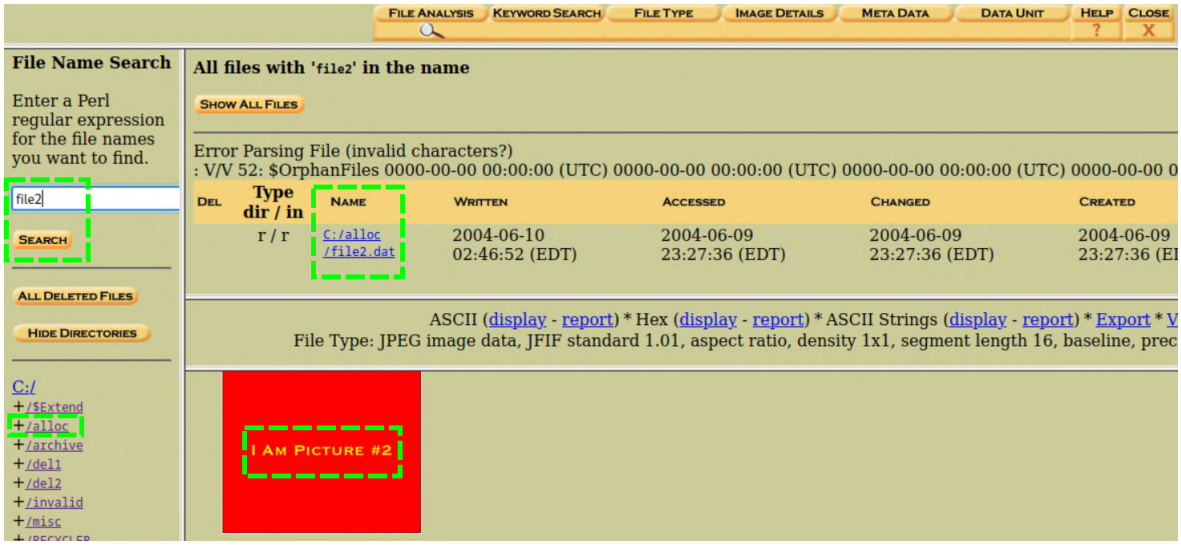


Figure 5.4.3 – file2.dat Exported and Opened as Image

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.

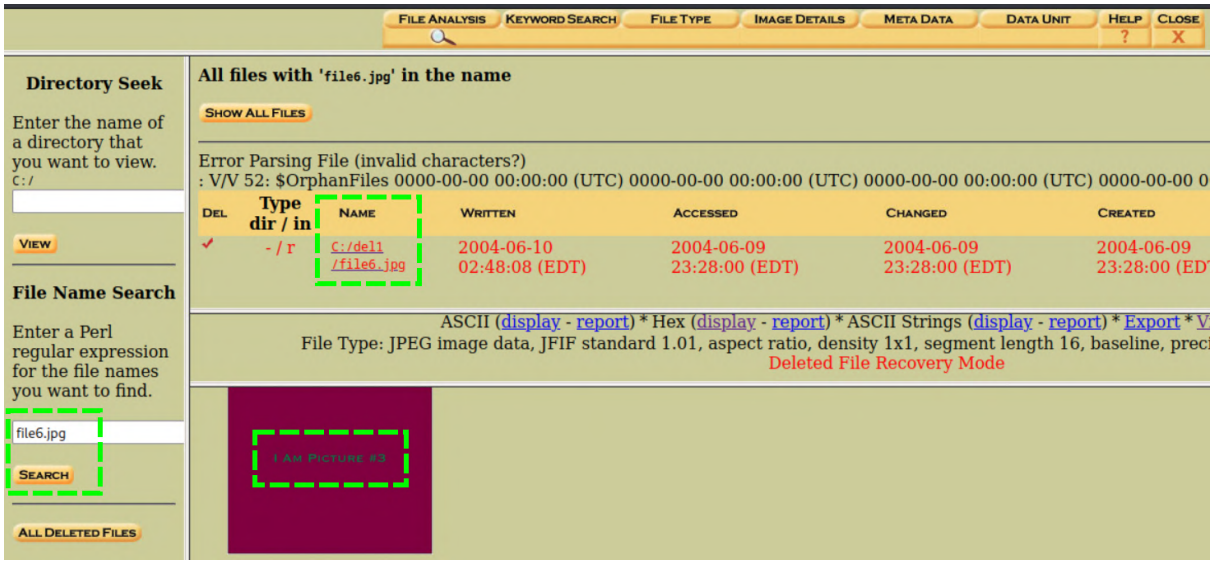


Figure 5.4.4 – file2.dat Exported and Opened as Image

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.

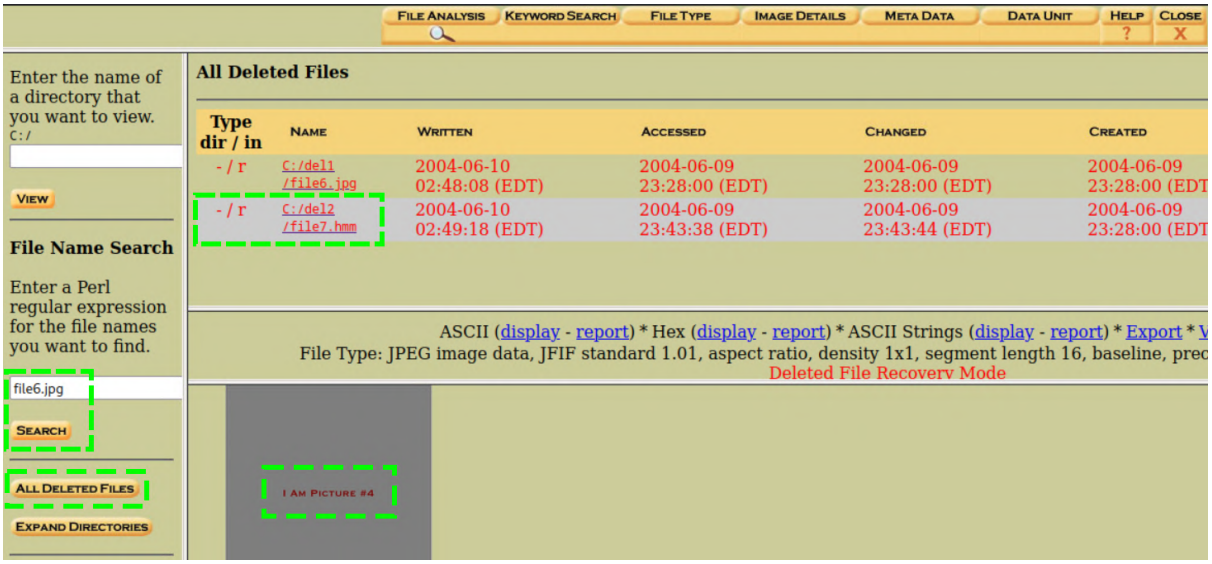


Figure 5.4.5 – file7.hmm Exported and Opened as Image

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.

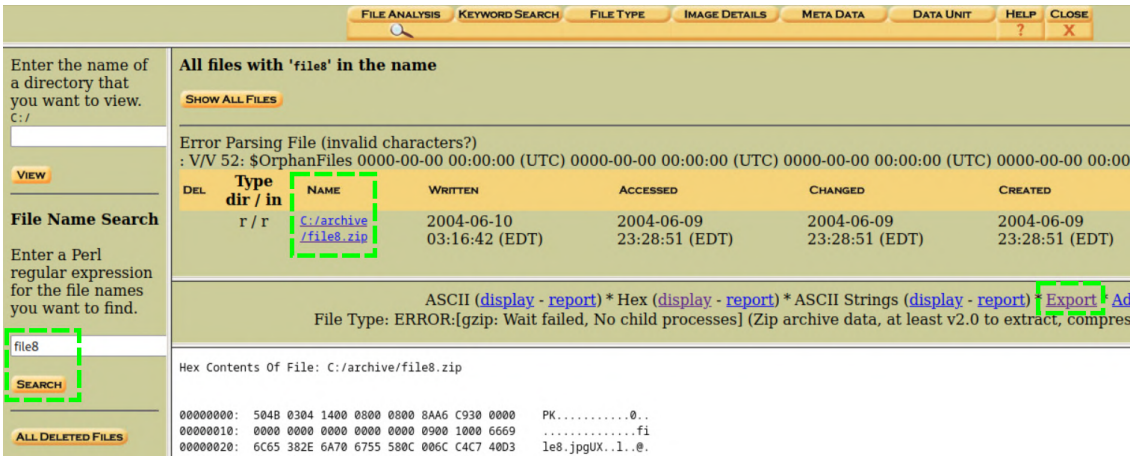


Figure 5.4.6.1 – file8.zip

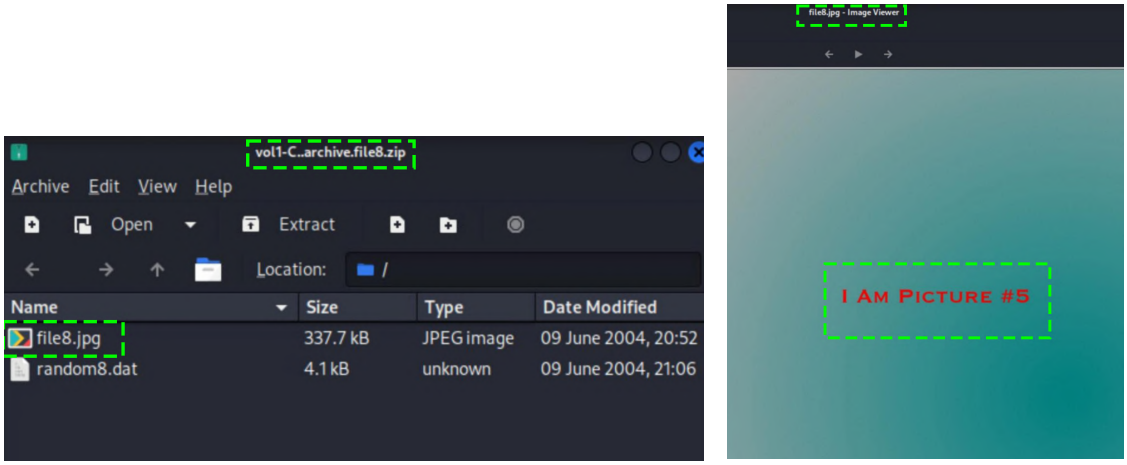


Figure 5.4.6.2 – file8.zip Extracted to Reveal file8.jpg

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.

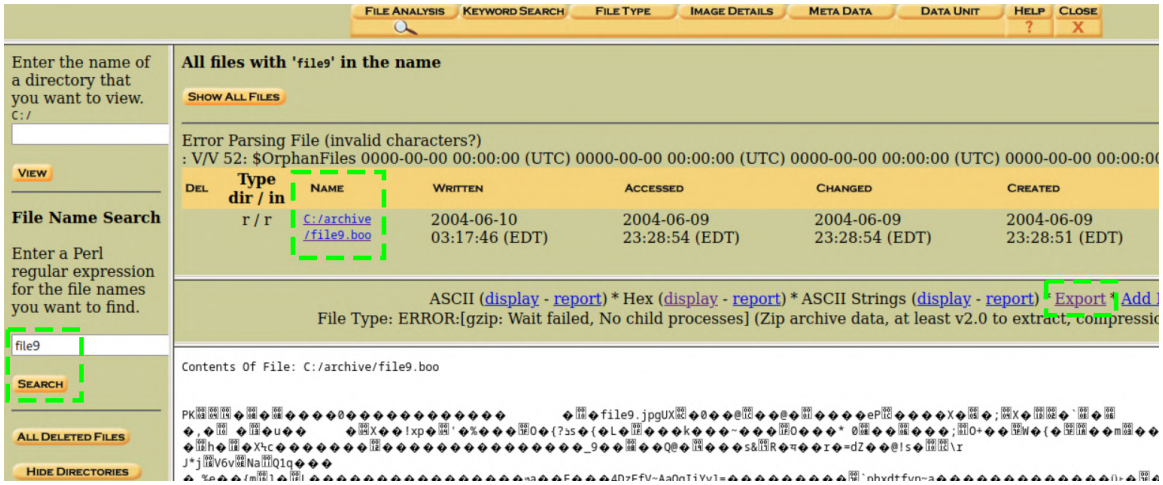


Figure 5.4.7 – file9.boo

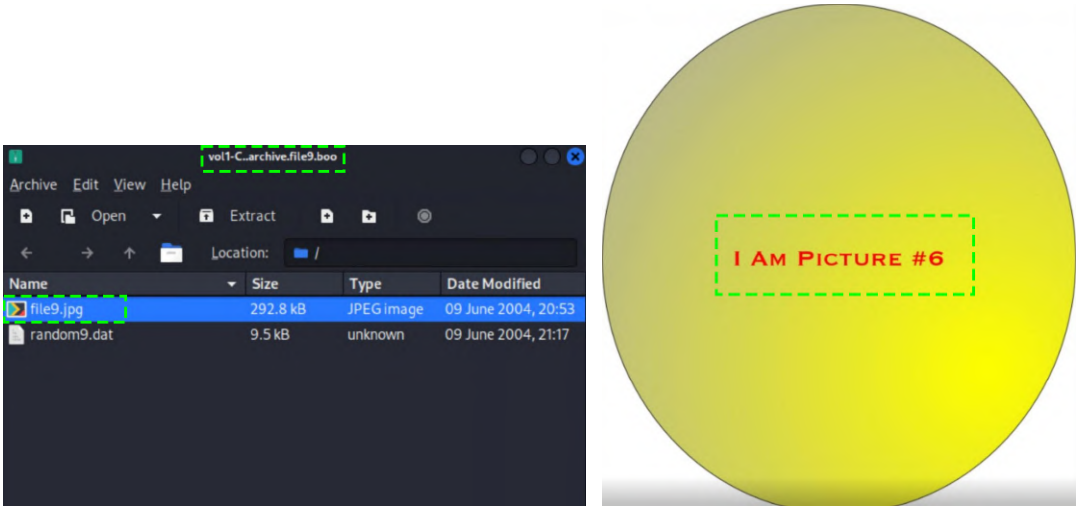


Figure 5.4.8 – file9.boo Extracted to Reveal file9.jpg

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.

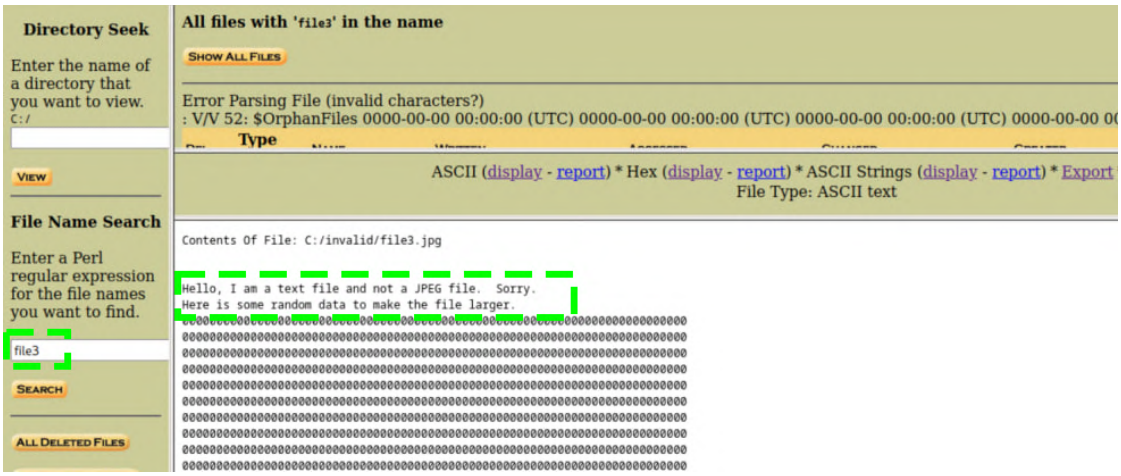


Figure 5.4.9 –Hex view of file3.jpg showing plain text

Consolidated Evidence Overview

The figure bellow 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.

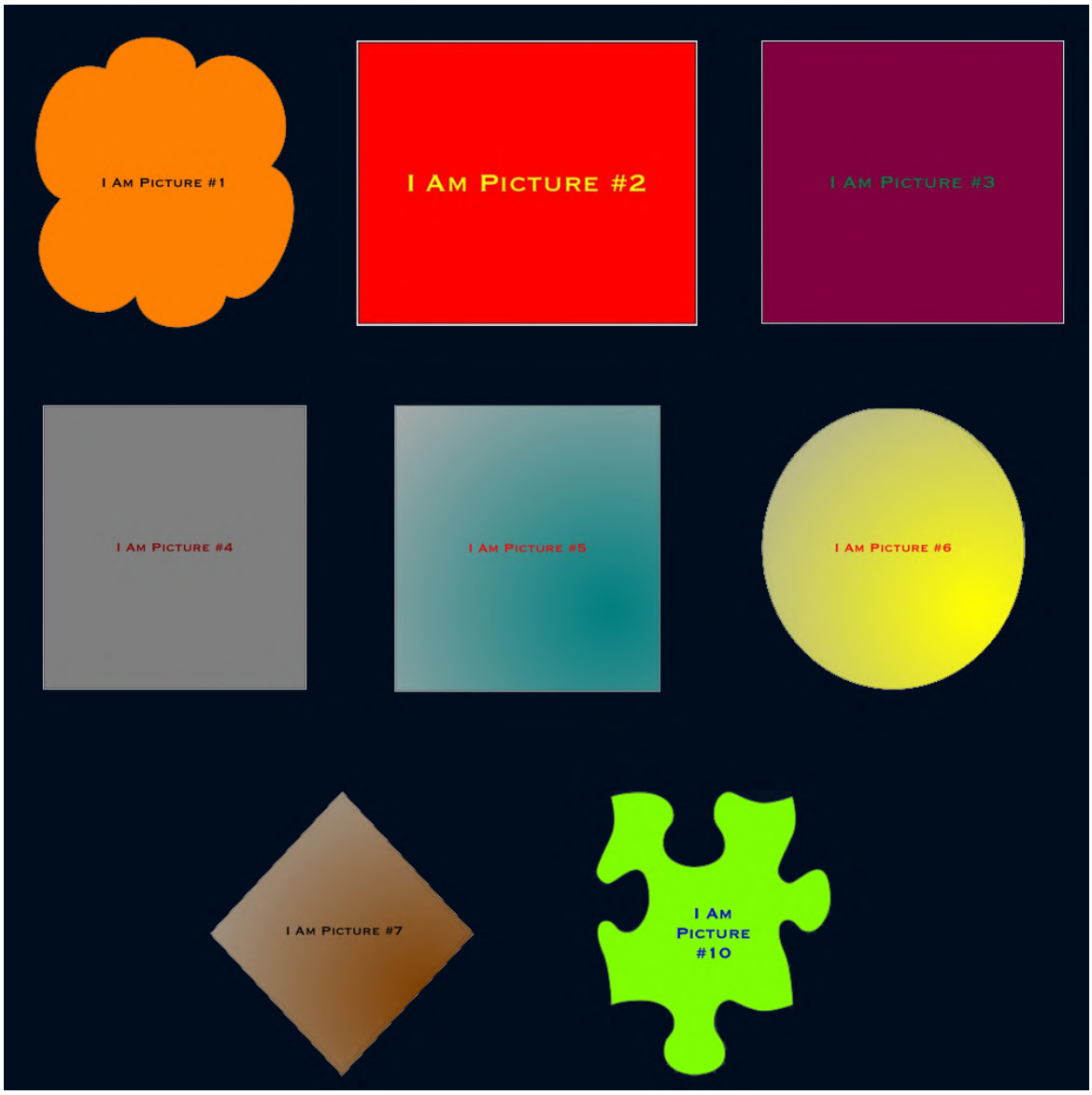


Figure 5.4.10 – Summary of Hidden JPG Evidence Recovered from Forensic Image

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 ● 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