INTERNSHIP PROJECT REPORT

Project 1 — Basic Vulnerability Assessment for a Small Business Network

TEAM MEMBERS:

1.VISHNU SUDARSAN.E.P

2.PALLAVI.S

3.NIDHI KATHAYAT

4.KARAN

Problem Statement

Simulate a real-world vulnerability assessment for a small business IT infrastructure. Identify security

gaps, prioritize risks, and provide mitigation strategies to improve the overall security posture.

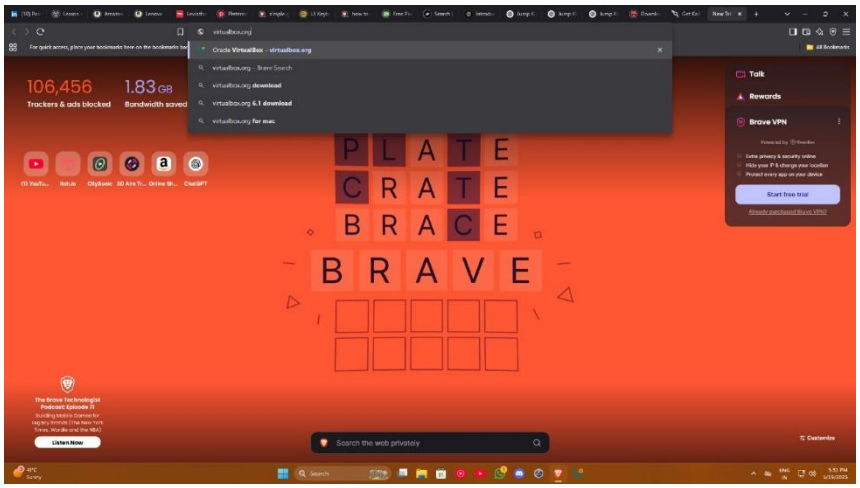
Week 1

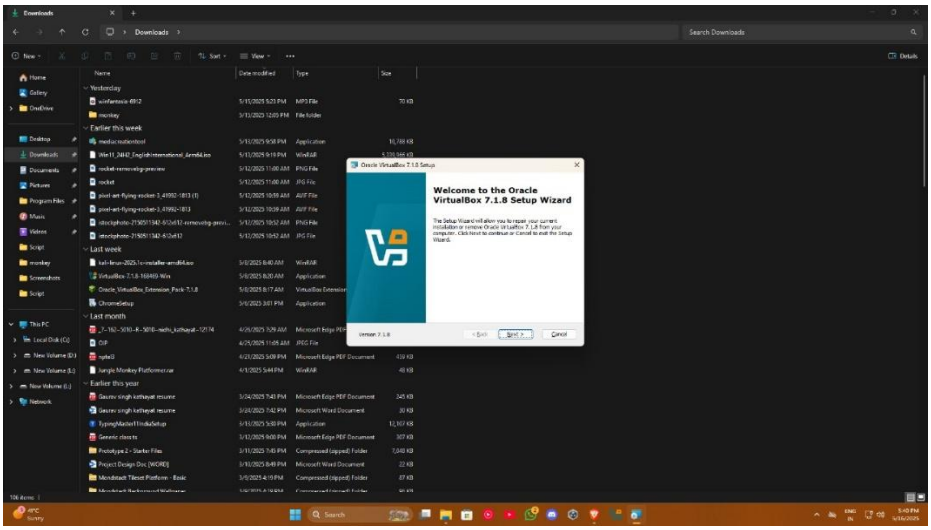
• Set up a virtual lab using VirtualBox/VMware.

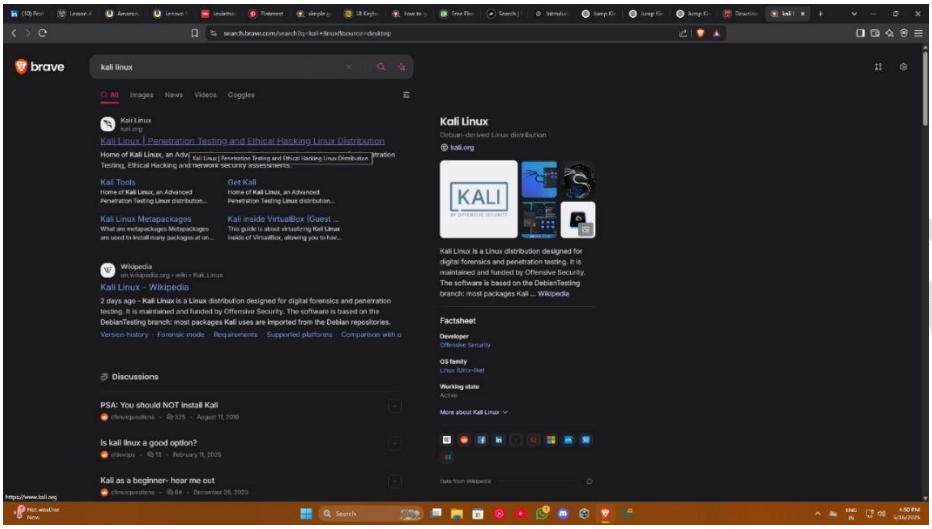
• Install Kali Linux and vulnerable machines (Metasploitable, OWASP Juice Shop).

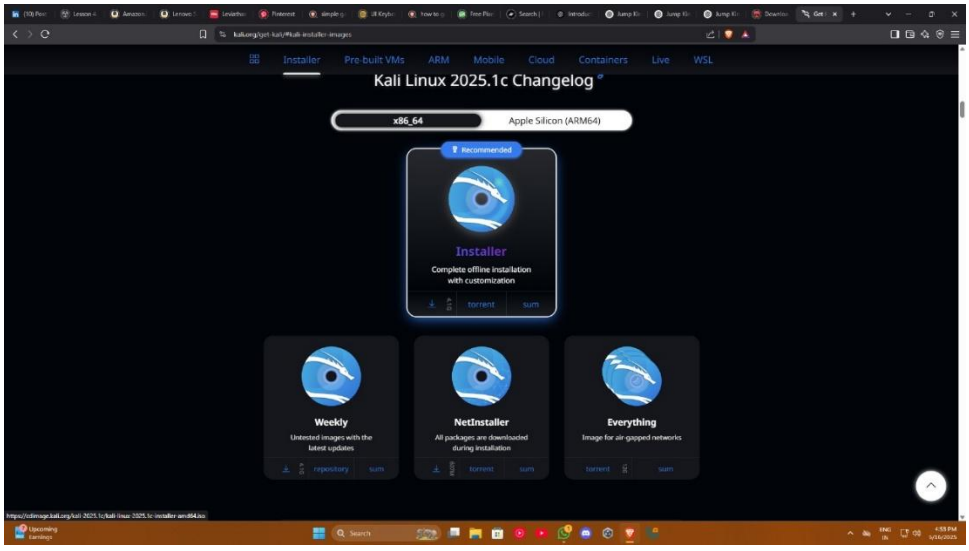
• Learn and practice basic scanning with Nmap and OpenVAS.

Below Screen Short of virtual lab Setup



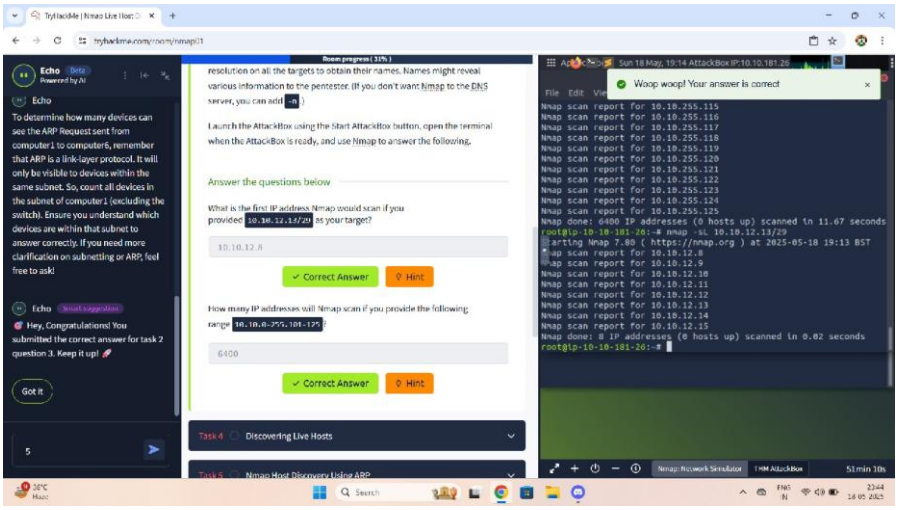

Welcome to the Oracle VirtualBox 7.1.8 Setup Wizard

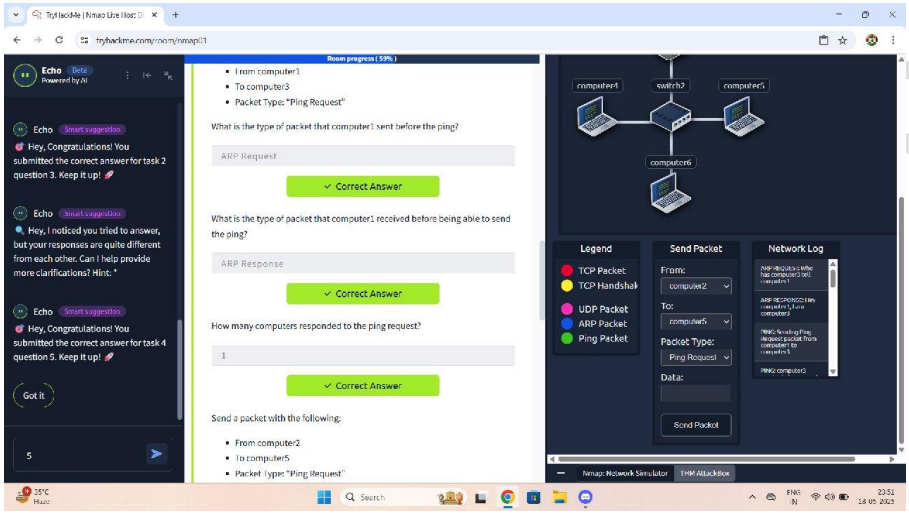

brave
kali linux
Kali Linux
Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution
Kali Linux - Wikipedia
Discussions
PSA: You should NOT install Kali
Is kali linux a good option?
Kali as a beginner- hear me out
Factsheet

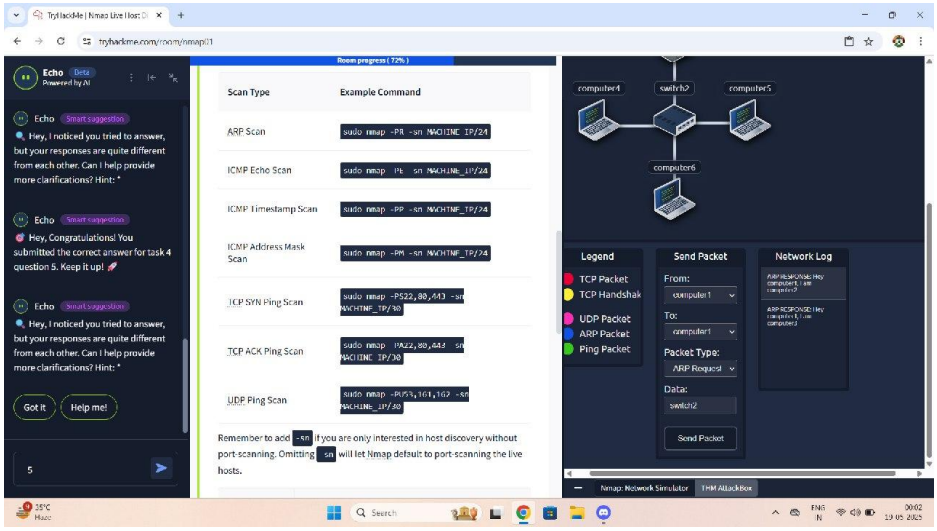

Installer
Pre-built VMs
ARM
Mobile
Cloud
Containers
Live
WSL
Kali Linux 2025.1c Changelog
x86_64
Apple Silicon (ARM64)
Recommended
Installer
Complete offline installation with customization
Weekly
Untested images with the latest updates
NetInstaller
All packages are downloaded during installation
Everything
Image for air-gapped networks

We have practiced Nmap Scanning using the “TRYHACKME”

Below Screen Shorts for example







WEEK 2 : Network Scanning & Vulnerability Scanning

Nmap

Nmap (Network Mapper) is an open-source tool used for **network discovery** and **security auditing**. It is widely used by cybersecurity professionals and system administrators to map networks, detect devices, find open ports, and identify services running on hosts.

To **discover hosts** on a network.

To **identify open ports** on devices.

To **detect services** and **operating systems**.

To **assess vulnerabilities** in a system.

Steps :

Define the Target

Host Discovery (Ping Scan)

Port Scanning

Service and Version Detection

Operating System Detection

Aggressive Scanning (Optional)

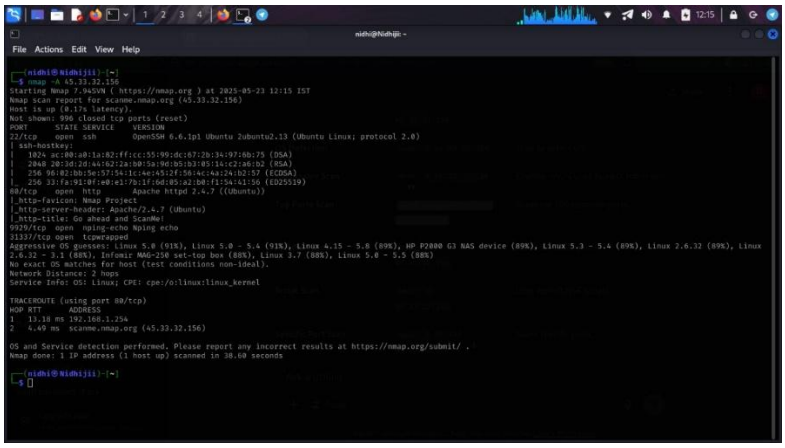
Script Scanning (Nmap Scripting Engine - NSE)

Analyze and Save the Results

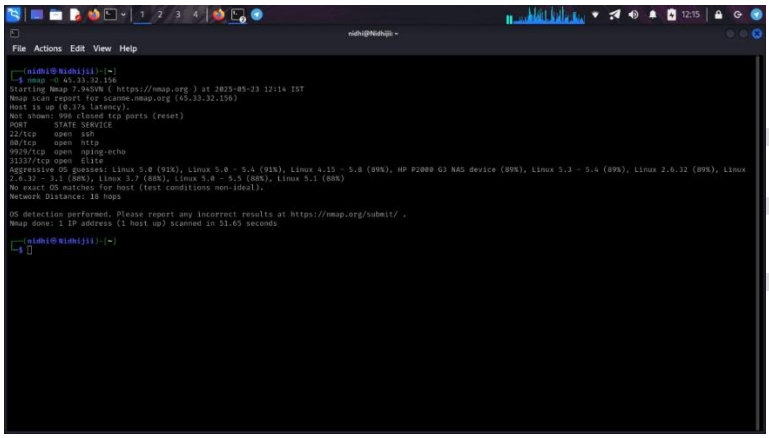
Take Action Based on Results

Below Screen Shorts of Nmap Scanning

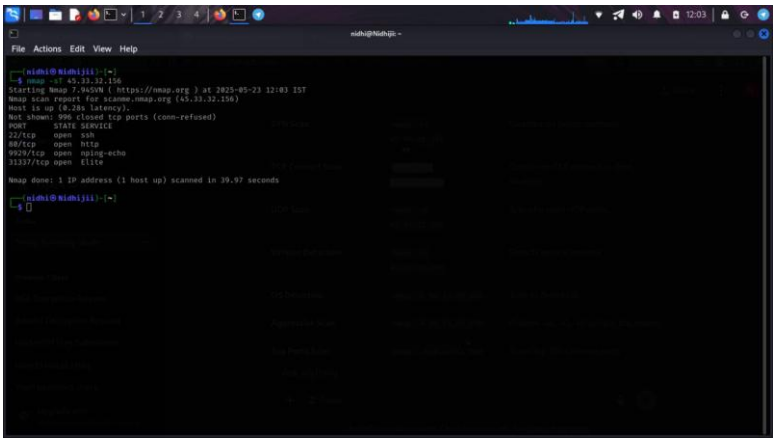
Command : nmap -A 45.33.32.156



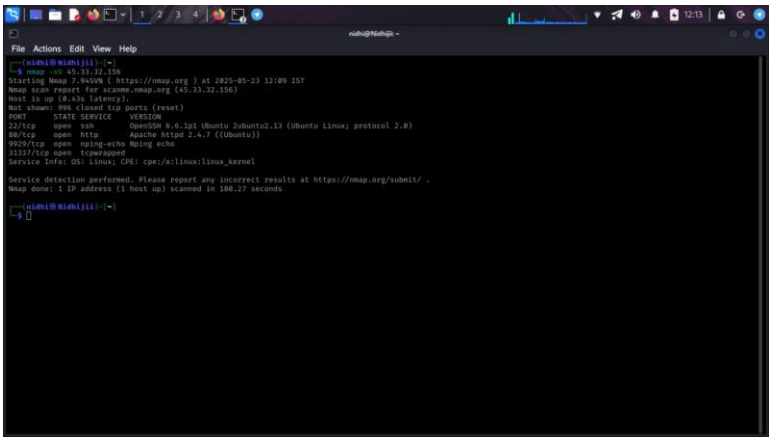
Command : nmap -O 45.33.32.156



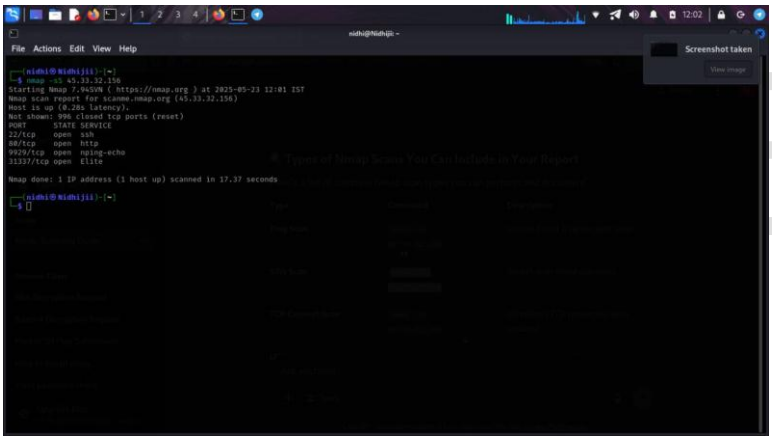
Command : nmap -sT 45.33.32.156



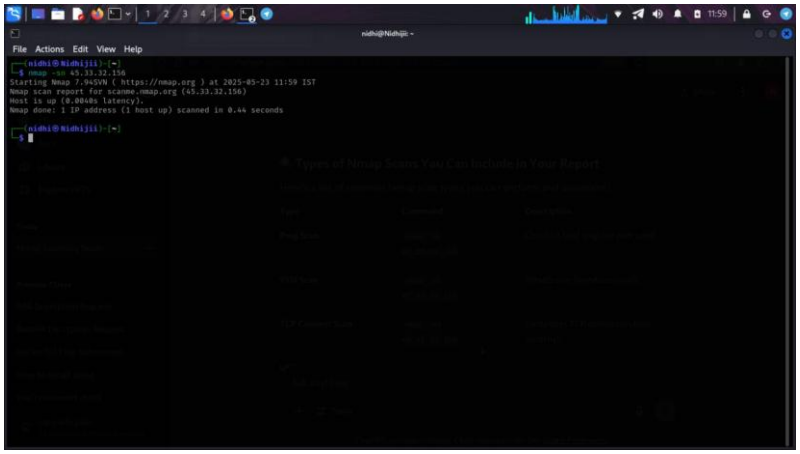
Command : nmap -sV 45.33.32.156



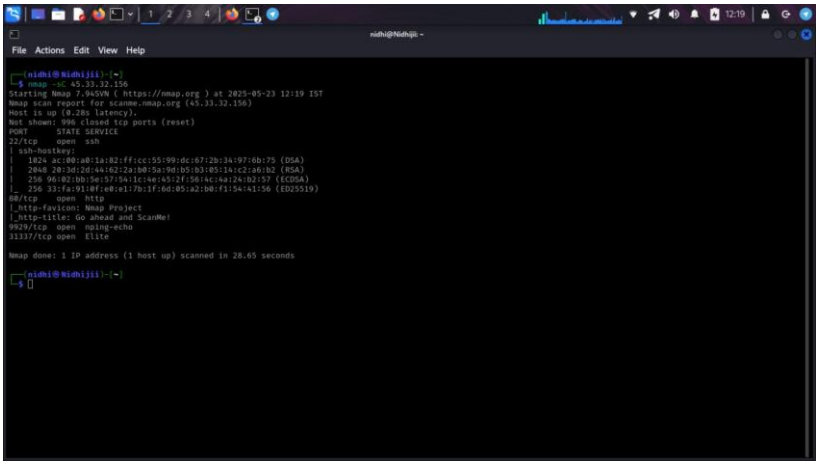
Command : nmap -sS 45.33.32.156



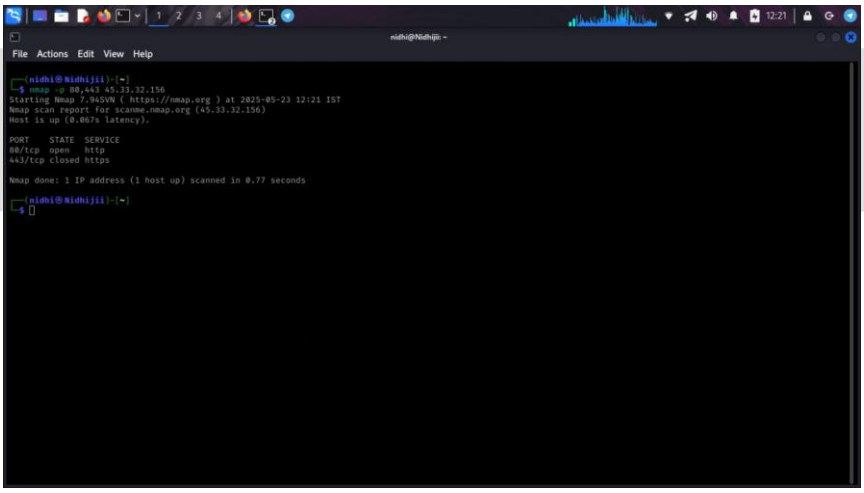
Command : nmap -sn 45.33.32.156



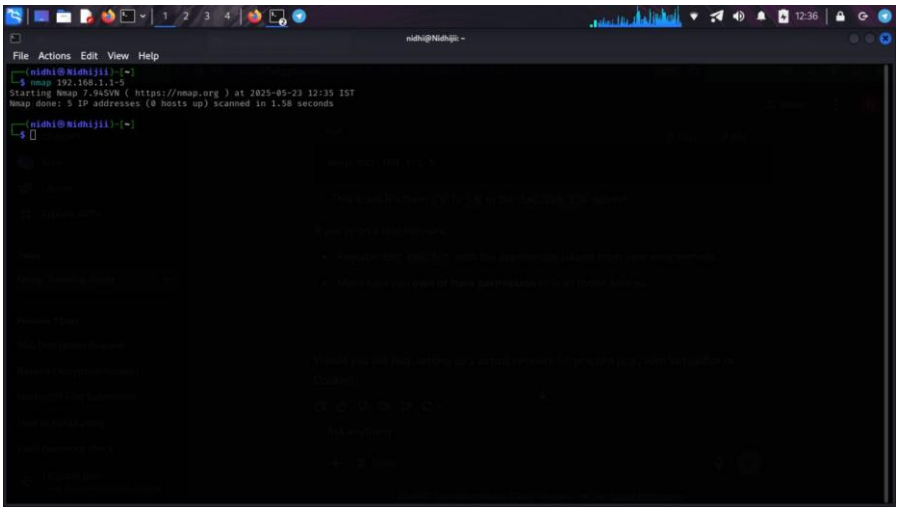
Command : nmap -sC 45.33.32.156



Command : nmap -p 80,443 45.33.32.156



Command : nmap 192.168.1.1-5



Explaination :

nmap -A 45.33.32.156
Performs an aggressive scan with OS detection, version info, scripts, and traceroute.
Useful for comprehensive analysis but easily detectable by firewalls**.**

nmap -O 45.33.32.156
Detects the target's operating system using TCP/IP fingerprinting.
Helps identify the OS to tailor further attacks or audits.

nmap -sT 45.33.32.156
Performs a full TCP connection (3-way handshake) on each port.
Less stealthy but reliable, especially for non-root users.

nmap -sV 45.33.32.156
Detects open ports and determines service versions.
Useful for identifying software vulnerabilities**.**

nmap -sS 45.33.32.156
Performs a stealthy SYN scan without completing TCP handshakes.
Commonly used for fast and quiet port discovery.

nmap -sn 45.33.32.156
Ping scan to check if the host is online without scanning ports.
Quickly discovers live hosts on a network.

nmap -sC 45.33.32.156
Runs default Nmap scripts for common service checks and vulnerabilities.
Provides additional detail beyond simple scans.

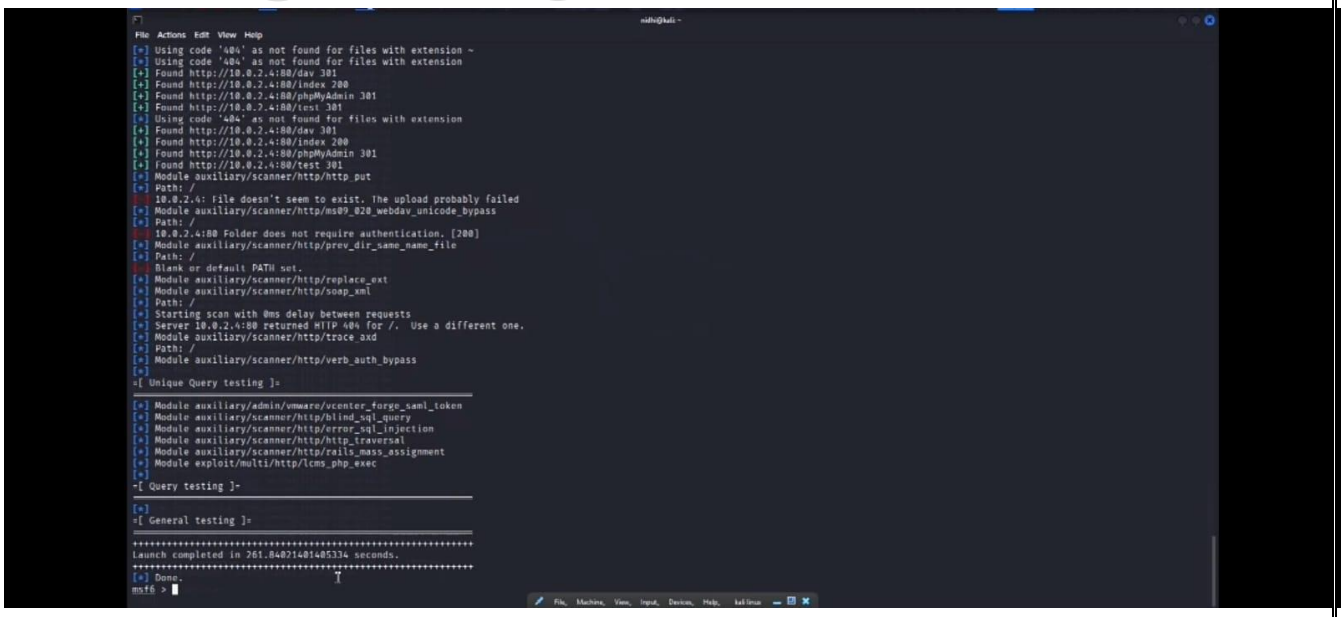
nmap -p 80,443 45.33.32.156
Scans only ports 80 (HTTP) and 443 (HTTPS) on the target IP.
Used to quickly check if web services are running on the host.

nmap 192.168.1.1-5

Scans IPs from 192.168.1.1 to 192.168.1.5 using default settings.
Useful for scanning a small range of hosts in a local network.

Vulnerability Scanning

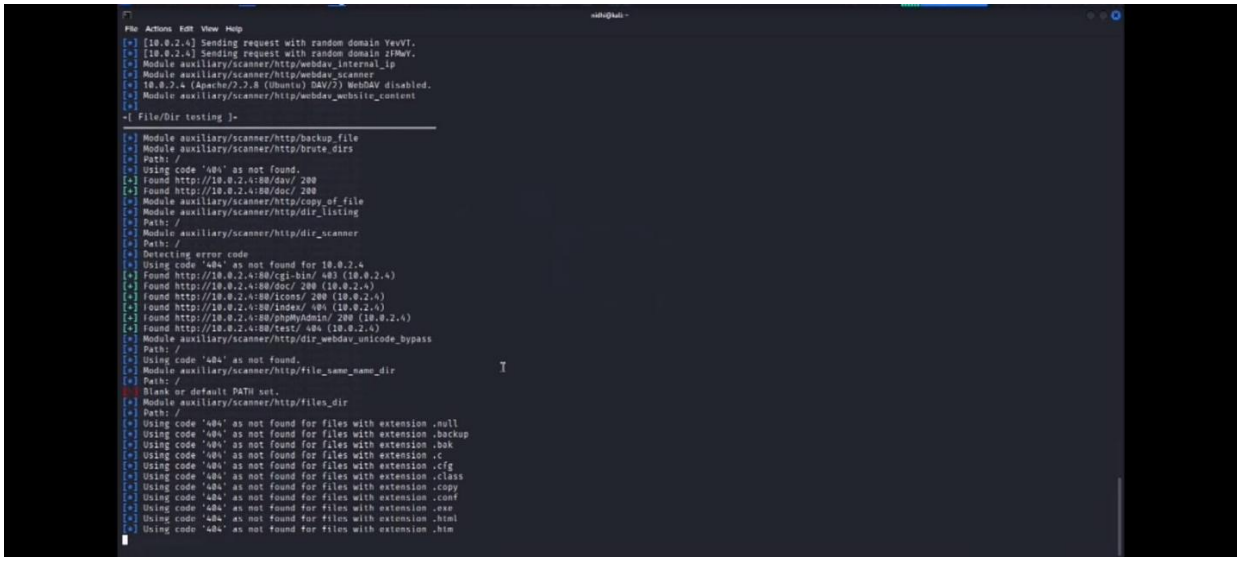
- **Vulnerability scanning uses automated tools to identify security weaknesses in systems, networks, and applications.**
- **It detects issues such as outdated software, misconfigurations, open ports, and exposed files or directories.**
- **Scanners actively probe targets and generate detailed reports with vulnerability descriptions and severity levels.**
- **The results help prioritize vulnerabilities based on risk and guide remediation efforts effectively.**
- **Vulnerability scanning is typically repeated regularly to track new vulnerabilities and maintain security posture.**
- **It supports compliance with security standards by ensuring continuous risk assessment.**
- **Scanning is often the first step before more comprehensive testing like penetration testing.**
- **Common tools include Nessus, OpenVAS, Qualys, and Rapid7 Nexpose.**



Vulnerability (1)

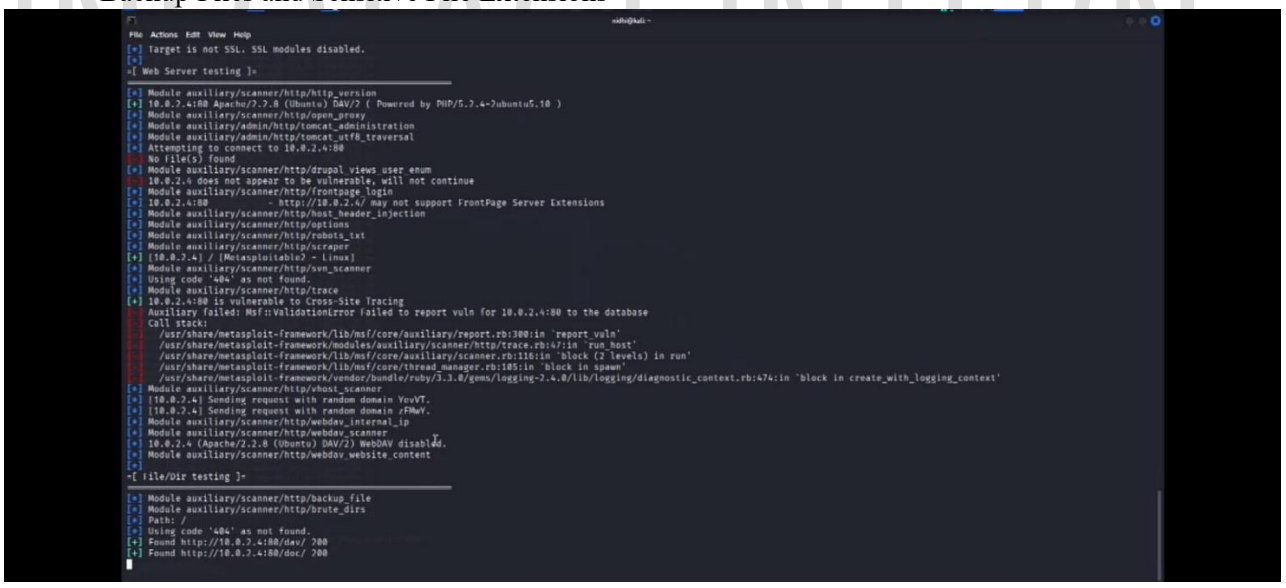
- Unauthenticated Directory Access
- WebDAV Unicode Bypass (ms09_020) – Failed.
- HTTP PUT Upload – Failed.
- HTTP Verb Authentication Bypass – Tested, but result unclear.

- SQL Injection (blind/error-based) – Tested, but not confirmed.
- CMS PHP Code Execution – Attempted, no success shown.



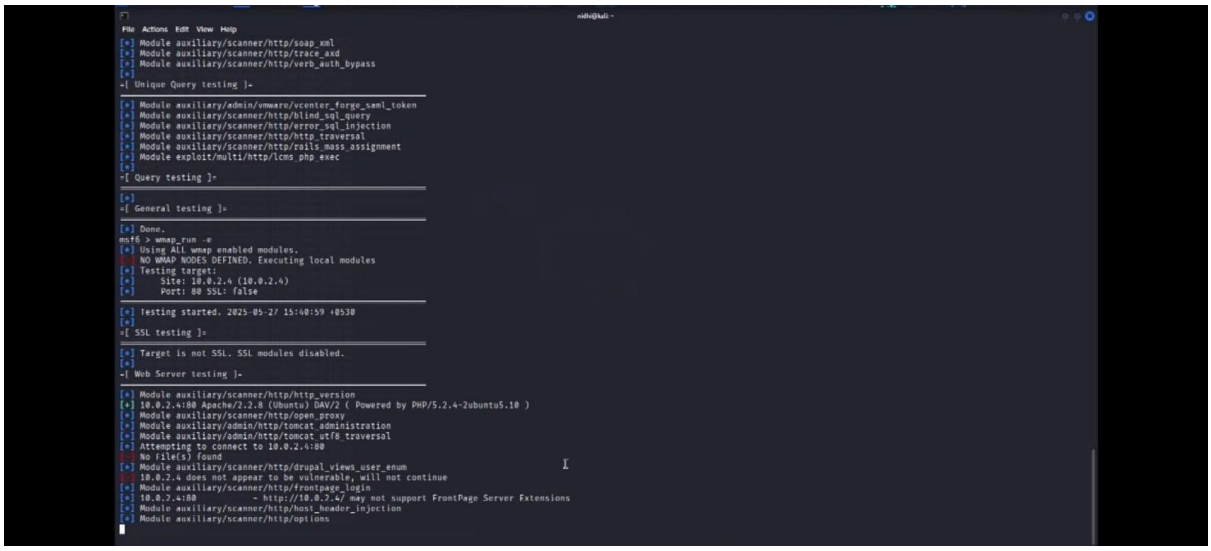
Vulnerability(2)

- Directory Listing / Unrestricted Directory Access
- Exposed phpMyAdmin Interface
- Backup Files and Sensitive File Extensions



Vulnerability(3)

- Web Server Trace Method Enabled
- Outdated Server Version
- Unprotected /dav/ and /doc/ Paths



Vulnerability(4)

- HTTP TRACE Enabled
- Unsecured WebDAV Access
- Outdated Web Server Stack
- No SSL/TLS Encryption

WEEK 3 : Research mitigation Strategies – Prioritize Vulnerabilities by Severity – Draft Security Recommendation

Vulnerability (1):

Prioritize the vulnerability

Unauthenticated Directory Access- High
WebDAV Unicode Bypass (ms09_020) – Medium to High
SQL Injection (blind/error-based) – Medium to High
CMS PHP Code Execution – High

🔐 **Mitigation Strategies**

1. Restrict Access to Sensitive Directories

- Use .htaccess or web server config (Apache/Nginx) to **block access** or require **authentication** for:
  - /phpMyAdmin
  - /dav
  - /admin or any sensitive path

2. Disable or Remove Unused Services

- **Disable WebDAV** if not required. It's often a target for upload-based attacks.

- Remove or disable **phpMyAdmin** from production environments, or use only locally.

3. Implement Authentication and Authorization

- Enforce **strong credentials** for all administrative interfaces.
- Use **HTTP Basic Auth** or **IP whitelisting** to restrict access to admin tools like phpMyAdmin.

4. Keep Software Updated

- Regularly update:
  - **phpMyAdmin**
  - Web servers (Apache/Nginx/IIS)
  - CMS platforms and plugins

5. Use a Web Application Firewall (WAF)

- Deploy a WAF to detect and block:
  - Directory enumeration
  - SQL injection
  - HTTP method tampering
  - Common Metasploit scans

Vulnerability(2):

Prioritize the Vulnerability

Directory Listing / Unrestricted Directory Access – High

Exposed phpMyAdmin Interface- High

Backup Files and Sensitive File Extensions – Medium – High

Mitigation Strategies:

1. Disable Directory Listing

- In Apache: Options -Indexes
- Prevents users from seeing directory contents when index file is missing.

2. Restrict phpMyAdmin Access

- Remove it from production servers if not needed.
- Restrict access via:
  - .htaccess
  - IP whitelisting
  - VPN-only access

3. Remove Backup/Sensitive Files

- Delete unnecessary .bak, .conf, .cfg, and .copy files from the web root.
- Move sensitive configuration files **outside public directories**.

4. Harden the Web Server

- Use WAF to block automated scanning.
- Set proper permissions and sanitize exposed paths.

5. Log and Monitor Access

- Enable logging for /phpMyAdmin and unusual requests.
- Set up alerts for directory brute-forcing or suspicious enumeration.

Vulnerability(3):

Prioritize the Vulnerability

Web Server Trace Method Enabled - High

Outdated Server Version - Critical
Unprotected /dav/ and /doc/ Paths – Medium

Mitigation Strategies

1. Disable TRACE Method

- **Apache**:

Apache : TraceEnable off

- Prevents XST attacks.

2. Upgrade Server Software

- Update Apache to latest LTS (e.g., Apache 2.4+)
- Upgrade PHP to currently supported versions (e.g., PHP 8.x).
- Patch known CVEs for Apache 2.2.8 and PHP 5.2.4.

3. Restrict Access to Sensitive Paths

- Use .htaccess or server config to block access to:

Bash
/dav/
/doc/
/phpMyAdmin/

- If not needed, remove the directories from the web root.

4. Disable or Harden WebDAV

- If not needed:
  - Disable WebDAV module completely.
- If needed:
  - Restrict file types.
  - Enforce authentication and access controls.

5. Enforce HTTPS

- Currently: **"Target is not SSL"**
- Deploy SSL/TLS (HTTPS) to protect traffic from interception and tampering.

Vulnerability (4):

Prioritize the Vulnerability
HTTP TRACE Enabled - High
Unsecured WebDAV Access - High
Outdated Web Server Stack – Critical
No SSL/TLS Encryption – Medium

Mitigation Strategies

1. HTTP TRACE Enabled
Disable the TRACE method in your web server settings.
Use firewall or WAF rules to block TRACE requests.
Regularly scan to ensure TRACE stays disabled.
2. Unsecured WebDAV Access
Disable WebDAV if not needed or restrict access.
Enforce strong authentication and HTTPS for WebDAV.
Monitor and patch WebDAV server regularly.
3. Outdated Web Server Stack

Keep web server and components updated with patches.
Use automated tools for consistent version control.
Test updates before applying in production.

4. No SSL/TLS Encryption

Install valid SSL/TLS certificates and enable HTTPS.
Redirect all HTTP traffic to HTTPS.
Disable old protocols; use TLS 1.2 or higher.

WEEK 4: Finalize the Vulnerability report

Vulnerability Assessment Report

1. Introduction

This report summarizes the key vulnerabilities discovered during the security assessment of the target web environment. It categorizes vulnerabilities by severity and provides an overview of preventive measures necessary to reduce risk exposure.

2. Identified Vulnerabilities and Severity

Vulnerability Set 1

- **Unauthenticated Directory Access** – High severity: Sensitive directories are accessible without authentication, increasing the risk of unauthorized access and data exposure.
- **WebDAV Unicode Bypass (ms09_020)** – Medium to High severity: This vulnerability could allow attackers to bypass WebDAV restrictions using Unicode encoding.
- **SQL Injection (blind/error-based)** – Medium to High severity: Potential for database compromise exists, although not conclusively confirmed.
- **CMS PHP Code Execution** – High severity: Risks of arbitrary code execution on the CMS platform were detected but not successfully exploited.

Vulnerability Set 2

- **Directory Listing / Unrestricted Directory Access** – High severity: Directory contents are exposed, allowing attackers to enumerate files and directories easily.
- **Exposed phpMyAdmin Interface** – High severity: phpMyAdmin is accessible without proper restrictions, increasing the risk of database attacks.
- **Backup and Sensitive Files Exposure** – Medium to High severity: Backup files and configuration files with sensitive information are present in publicly accessible locations.

Vulnerability Set 3

- **Web Server TRACE Method Enabled** – High severity: TRACE method can be exploited in Cross-Site Tracing (XST) attacks to steal sensitive data.
- **Outdated Server Version** – Critical severity: The server is running outdated software with known vulnerabilities, posing significant risk.
- **Unprotected /dav/ and /doc/ Paths** – Medium severity: These directories lack adequate access controls, potentially exposing sensitive data.

Vulnerability Set 4

- **HTTP TRACE Enabled** – High severity: Enabled TRACE method introduces risks of sensitive information disclosure.
- **Unsecured WebDAV Access** – High severity: WebDAV access is not properly secured, allowing unauthorized operations.
- **Outdated Web Server Stack** – Critical severity: The server software and components are outdated, increasing exposure to known exploits.
- **No SSL/TLS Encryption** – Medium severity: Lack of encryption exposes data to interception and manipulation during transmission.

3. Prevention Overview

The vulnerabilities identified highlight the need for comprehensive security hygiene, including:

- Restricting access to sensitive directories and administrative interfaces to prevent unauthorized entry.
- Disabling unnecessary services like WebDAV and TRACE to reduce attack surface.
- Keeping all server components, CMS, and related software up to date with security patches.
- Enforcing encrypted communication channels through SSL/TLS to protect data in transit.
- Limiting exposure of backup and configuration files by proper file management and permissions.

4. Summary

The assessment reveals several high and critical severity vulnerabilities that could allow attackers to compromise sensitive data, execute arbitrary code, or gain unauthorized access. The presence of outdated software and insecure configurations further increases risk.

Immediate attention is required to secure directory access, disable insecure HTTP methods, and ensure encryption is properly implemented. Maintaining up-to-date software and restricting unnecessary services will significantly improve the security posture and resilience against common web-based attacks.

Project 2 — **Web Application Penetration Testing – OWASP Top 10 Focus**

Problem Statement

The objective of this project is to conduct a comprehensive penetration test on a sample vulnerable web application (such as DVWA or OWASP Juice Shop), identifying and exploiting vulnerabilities listed in the **OWASP Top 10**. The ultimate goal is to document the vulnerabilities, demonstrate exploitation, and provide actionable recommendations for remediation.

Week 1: Setup and Reconnaissance

Tasks Completed:

- Studied the **OWASP Top 10 (2021)** vulnerabilities:
    1. Broken Access Control
    2. Cryptographic Failures
    3. Injection (e.g., SQLi)
    4. Insecure Design
    5. Security Misconfiguration
    6. Vulnerable and Outdated Components
    7. Identification and Authentication Failures
    8. Software and Data Integrity Failures
    9. Security Logging and Monitoring Failures
    10. Server-Side Request Forgery (SSRF)
- Set up local environment:
    - Installed and configured **DVWA (Damn Vulnerable Web Application)** on XAMPP.
    - Configured **OWASP Juice Shop** using Node.js for extended testing.
- Reconnaissance using:
    - **Burp Suite** (Spidering, Repeater, Scanner)
    - Identified endpoints, request parameters, cookies, and headers.

Evidence:

- Screenshots of Burp Suite target scope and spidering results.
- Localhost test environment running DVWA and Juice Shop.

Setup on DVWA

Commands to setup the Dvwa :

STEP 1: Move to the web server root directory

cd /var/www/html/ # (HTML directory to clone)

STEP 2: Clone DVWA from GitHub

sudo git clone 'url' # Replace 'url' with actual DVWA GitHub link:

https://github.com/digininja/DVWA.git

STEP 3: List the folder to check DVWA is cloned

ls # (Check if 'DVWA' directory is there)

STEP 4: Give all permissions to the DVWA folder

sudo chmod -R 777 DVWA # (Read, Write, Execute permissions)

STEP 5: Go inside DVWA folder

cd DVWA

STEP 6: Go inside config folder

cd config

STEP 7: List files inside config

Ls

STEP 8: Copy the config file template to actual config

cp config.inc.php.dist config.inc.php

STEP 9: Edit the config file to set DB username & password

sudo nano config.inc.php

In nano:

→ Change the DB user to 'user'

→ Change the DB password to 'pass'

Save with: Ctrl + O

Exit with: Ctrl + X

CONFIGURE THE DATABASE FOR DVWA

STEP 10: Start MySQL service

sudo service mysql start

STEP 11: Login to MySQL as root

sudo mysql -u root -p

STEP 12: Create user for DVWA

CREATE USER 'user'@'127.0.0.1' IDENTIFIED BY 'pass';

STEP 13: Grant all privileges to the user

GRANT ALL PRIVILEGES ON dvwa.* TO 'user'@'127.0.0.1' IDENTIFIED BY 'pass';

STEP 14: Exit MySQL

Exit

STEP 15: Go to PHP configuration folder (adjust version if different)

cd /etc/php/8.2/apache2/

STEP 16: Check the contents

ls

STEP 17: Edit the PHP configuration file

sudo nano php.ini

#; Fopen wrappers

allow_url_fopen = On

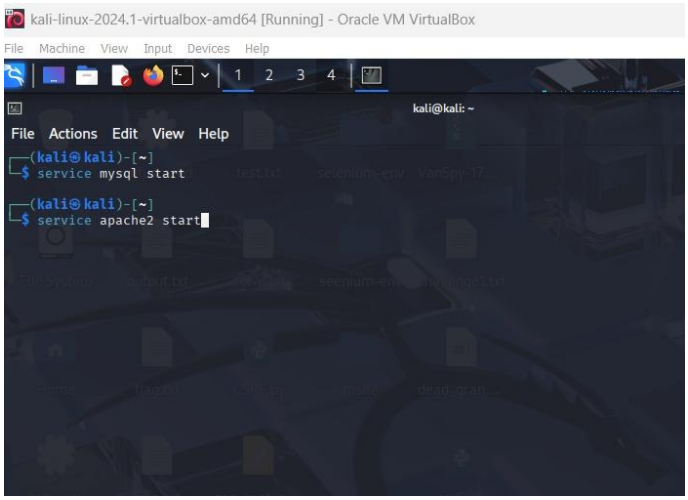
allow_url_include = On

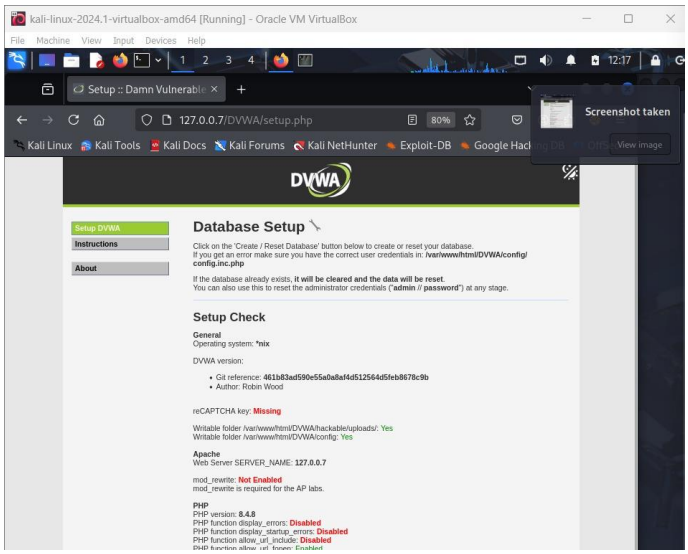
STEP 18: Restart Apache to apply changes

sudo service apache2 start

FINAL DVWA SETUP LINK

http://127.0.0.8/dvwa/setup.php





Example screenshorts of Testing

Week 2: Injection & Cross-Site Scripting (XSS)

Tasks Completed:

1. **SQL Injection (SQLi):**
    - Identified vulnerable input in DVWA's "SQL Injection" module.
    - Payload used:
      ' OR '1'='1
    - Demonstrated data extraction (e.g., user credentials).
    - Automated detection using **sqlmap**.
2. **Cross-Site Scripting (XSS):**

- Reflected XSS:
  Injected: <script>alert('XSS')</script>
- Stored XSS:
  Injected: <img src=x onerror=alert('XSS Stored')>
- Demonstrated execution in victim's browser.

3. **Broken Authentication:**
   - Bypassed login with default credentials.
   - Session fixation via predictable PHPSESSID.
4. **Sensitive Data Exposure:**
   - Intercepted credentials over HTTP (no HTTPS).
   - Found hardcoded API keys and secrets.

Evidence:

- Screenshots of alert boxes, intercepted responses, sqlmap outputs.
- Source code snippets highlighting vulnerability points.

Example Screenshorts:

PERFORMING XXS ON DVWA VULNERABLE WEBSITE

Make sure:

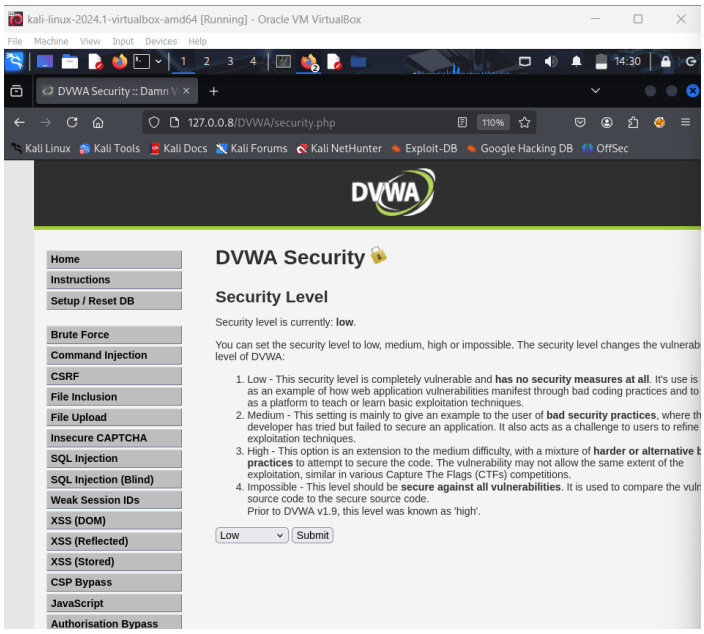
- DVWA is running (http://127.0.0.8/dvwa/)
- You are logged in as an admin (admin / password by default)
- Database is setup via: http://127.0.0.8/dvwa/setup.php
- Security Level is set to Low or Medium:
  - Go to: http://127.0.0.8/dvwa/security.php
  - Choose: Low or Medium
  - Click Submit

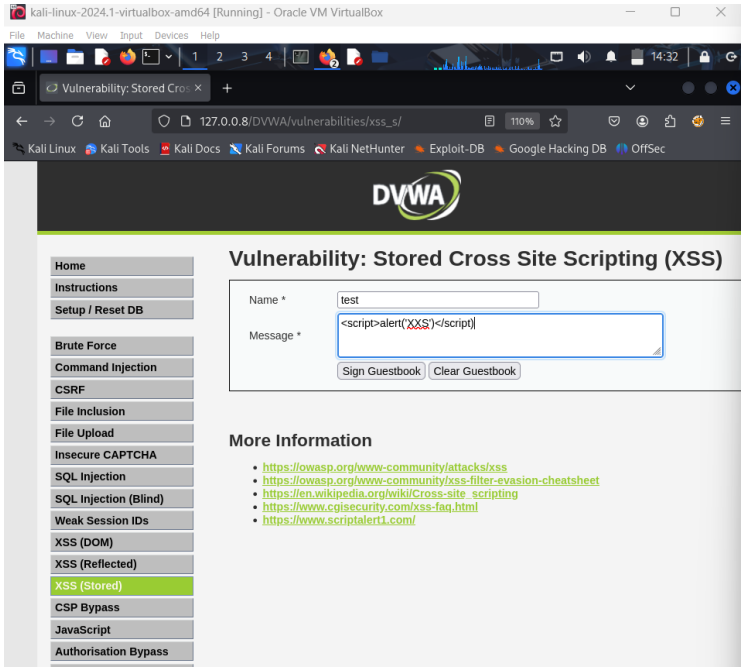
On LOW Security

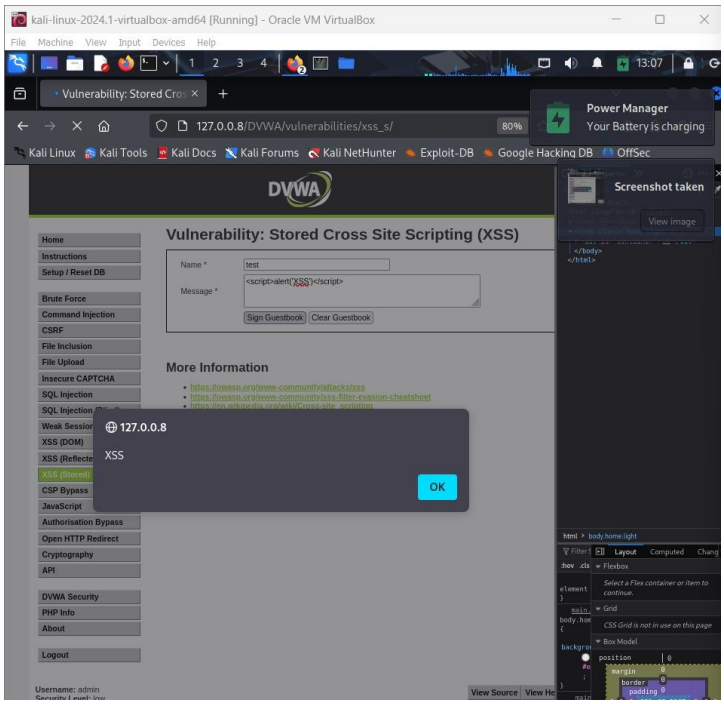
Navigate to http://127.0.0.8/dvwa/vulnerabilities/xss_s/

Name : test

Message : <script>alert('XSS')</script>


kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
DVWA Security :: Damn
127.0.0.8/DVWA/security.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
DVWA Security
Security Level
Security level is currently: low.
You can set the security level to low, medium, high or impossible. The security level changes the vulnerab
level of DVWA:
1. Low - This security level is completely vulnerable and has no security measures at all. It's use is
as an example of how web application vulnerabilities manifest through bad coding practices and to
as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security practices, where th
developer has tried but failed to secure an application. It also acts as a challenge to users to refine
exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative b
practices to attempt to secure the code. The vulnerability may not allow the same extent of the
exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vuln
source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.
Low
Submit


kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
Vulnerability: Stored Cros
127.0.0.8/DVWA/vulnerabilities/xss_s/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
test
Message *
<script>alert('XXS')</script>
Sign Guestbook
Clear Guestbook
More Information
https://owasp.org/www-community/attacks/xss
https://owasp.org/www-community/xss-filter-evasion-cheatsheet
https://en.wikipedia.org/wiki/Cross-site_scripting
https://www.cgisecurity.com/xss-faq.html
https://www.scriptalert1.com/

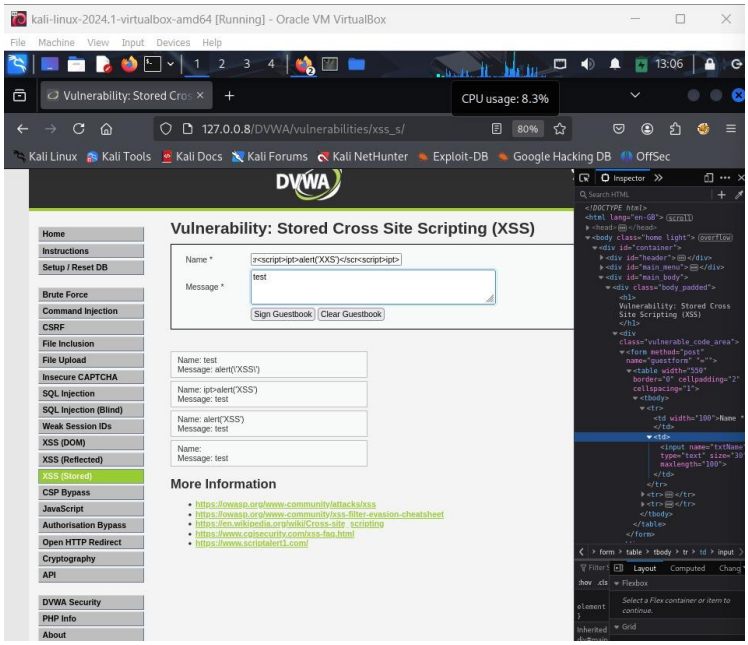


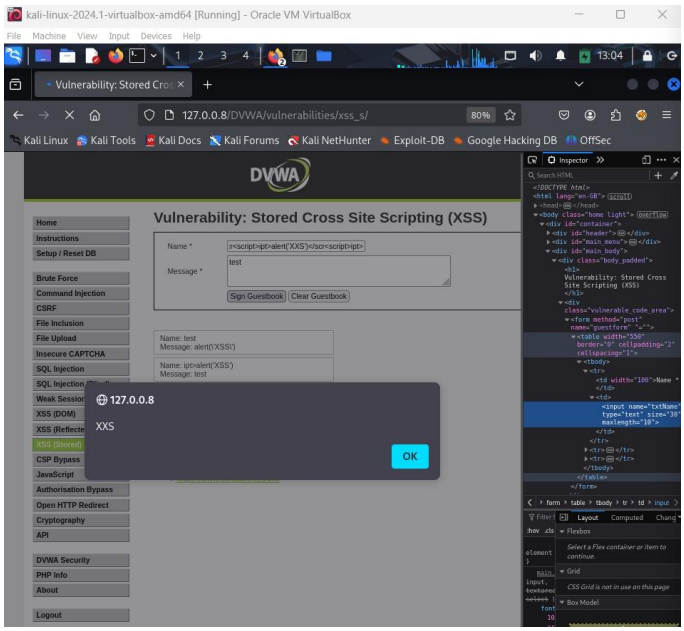
On MEDIUM Security

Navigate to http://127.0.0.8/dvwa/vulnerabilities/xss_s/

Name: **<scr<script>ipt>alert('XSS')</scr<script>ipt>** (Note: we need to increase the size on inspecting)

Message : test



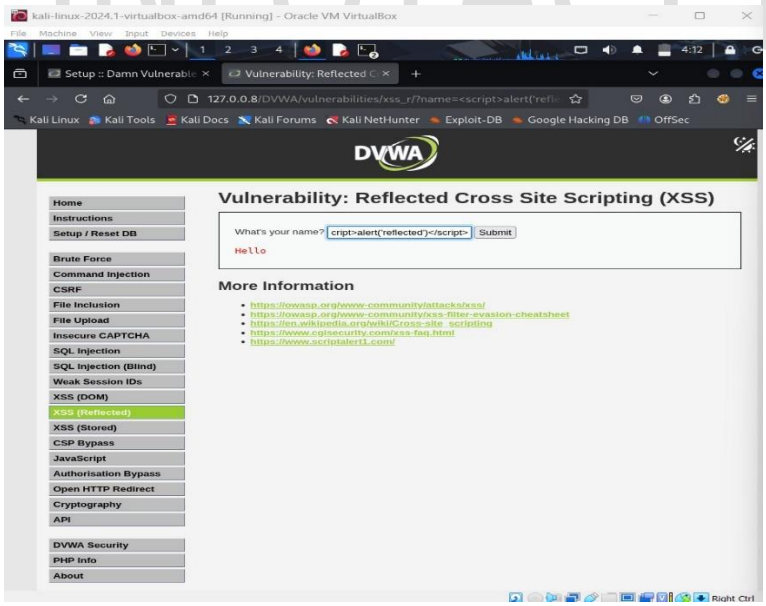


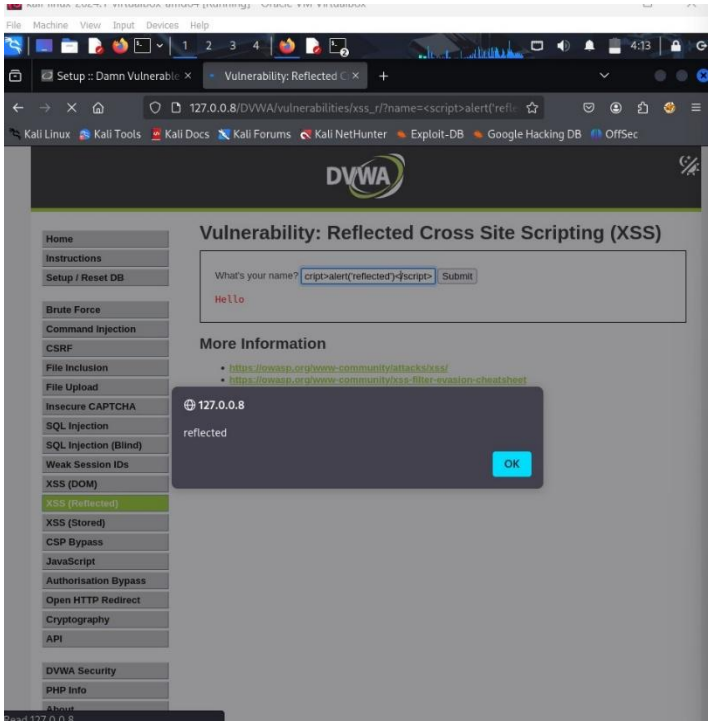
On LOW Security

Navigate to http://127.0.0.8/dvwa/vulnerabilities/xss_r/

Name : test

Message : <script>alert('XSS')</script>



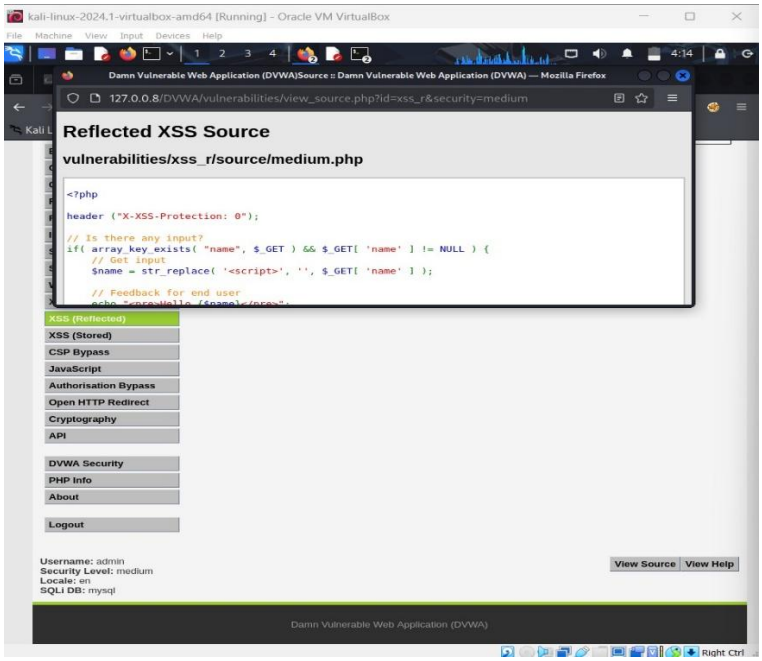


On MEDIUM Security

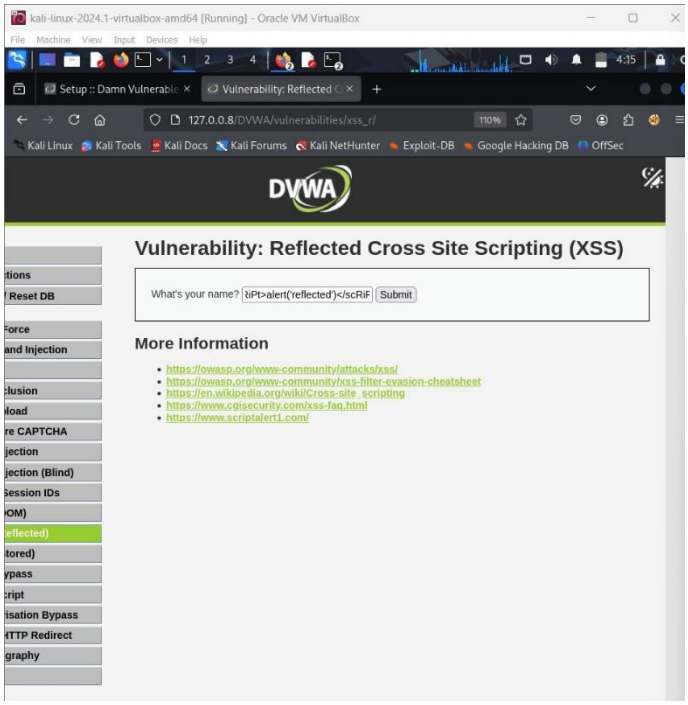
Navigate to http://127.0.0.8/dvwa/vulnerabilities/xss_s/

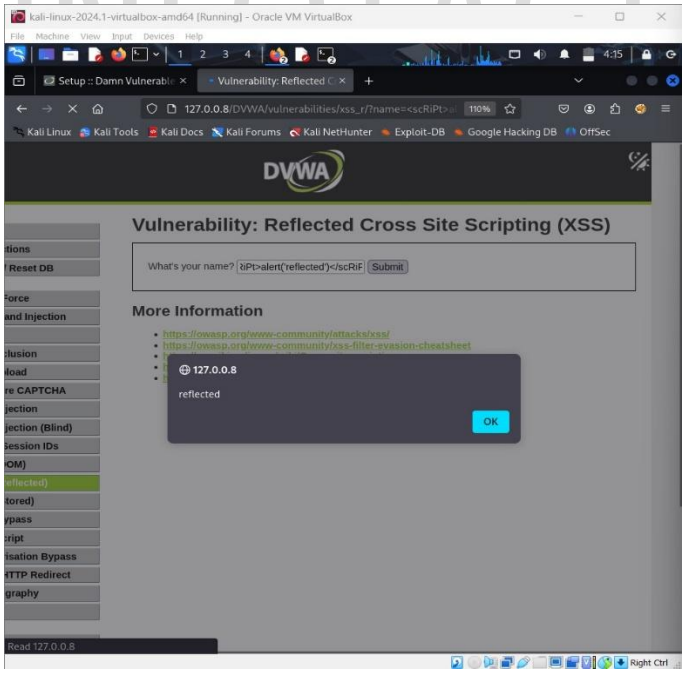
Name: <scRiPt>alert('reflected')</scRiPt)

Message : test



View the source page : where <script> will be replaced with space
so we Capitalize some word in <script> → <scRiPt>





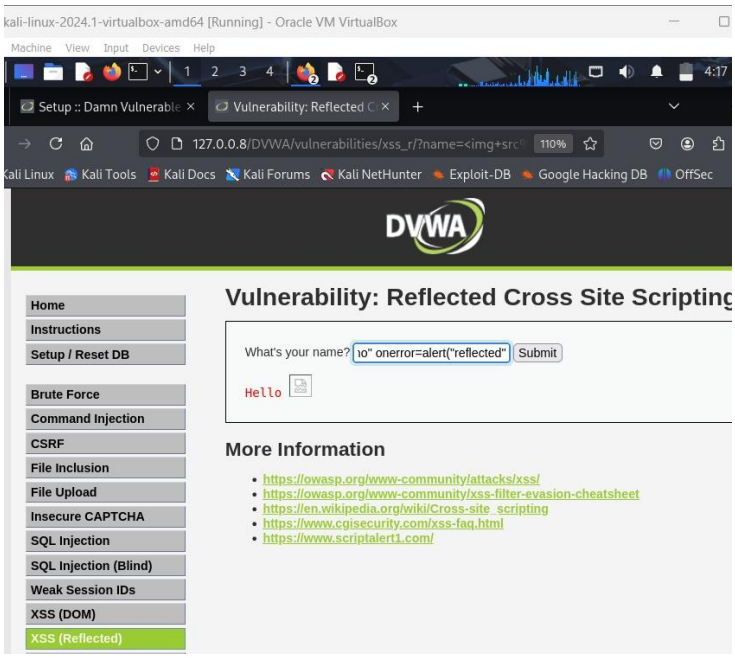
On HIGH Security

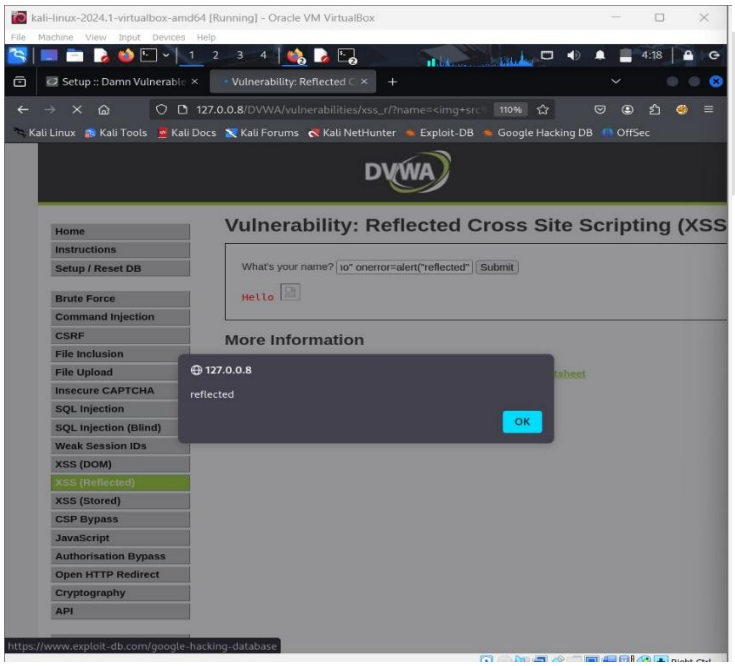
Navigate to http://127.0.0.8/dvwa/vulnerabilities/xss_s/

Name: <img = 'x' onerror= alert('reflected')>

Message : test

On view source page the script are give in Case Sensitive so we use img tag and give with error test


kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
Machine View Input Devices Help
Setup :: Damn Vulnerable
Vulnerability: Reflected
127.0.0.8/DVWA/vulnerabilities/xss_r/?name=<img+src
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
Vulnerability: Reflected Cross Site Scripting
What's your name?
Submit
Hello
More Information
https://owasp.org/www-community/attacks/xss/
https://owasp.org/www-community/xss-filter-evasion-cheatsheet
https://en.wikipedia.org/wiki/Cross-site_scripting
https://www.cgisecurity.com/xss-faq.html
https://www.scriptalert1.com/


kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
DVWA
Vulnerability: Reflected Cross Site Scripting (XSS
What's your name?
Submit
Hello
More Information
127.0.0.8
reflected
OK
XSS (Reflected)
https://www.exploit-db.com/google-hacking-database

Week 3 :

1. Security Misconfiguration

Objective:

To identify insecure default configurations, unnecessary features, and misconfigured permissions that can expose the web application to external attacks.

Methodology:

- Manual inspection of exposed URLs and directories
- Burp Suite scanning for misconfigured headers
- File inclusion and access testing
- Directory brute-forcing using tools like dirb and gobuster

Vulnerabilities Identified:

a. Directory Listing Enabled

Observation:
Accessing http://127.0.0.1/DVWA/config/ revealed a directory listing of sensitive configuration files.

Impact:
An attacker can access files like config.inc.php, which may contain database credentials.

Proof-of-Concept (PoC):

URL Accessed: http://127.0.0.1/DVWA/config/

Exposed Files: config.inc.php, setup.php

Screenshot:
🖼️ Directory listing exposing internal files (attached separately in the report)

Remediation:

- Disable directory listing via Apache/Nginx config.
- Restrict access to internal directories via .htaccess or web server rules.

b. Exposed Admin Panel

Observation:
The /admin/ directory was accessible without authentication.

Impact:
Attackers can gain access to admin functionalities (e.g., user delete, config updates).

PoC URL:
http://127.0.0.1/DVWA/admin/delete_user.php?id=1

Exploit Steps:

1. Navigated directly to the admin URL without logging in.

2. Was able to delete user records.

Remediation:

- Implement role-based access controls.
- Enforce session-based authentication checks.

c. Missing Security Headers

Findings from Burp Suite Scanner:

- X-Content-Type-Options not set
- Content-Security-Policy missing
- X-Frame-Options missing

Impact:
Increased risk of clickjacking, MIME-type attacks, and XSS.

Remediation:
Add the following headers in server configuration:

Header set X-Content-Type-Options "nosniff"

Header set X-Frame-Options "DENY"

Header set Content-Security-Policy "default-src 'self'"

2. Broken Access Control

Objective:

To test if unauthorized users can perform actions or access resources that should be restricted.
Methodology:

- Manual URL manipulation
- Role-based testing (admin vs user)
- IDOR testing
- Testing for forced browsing of protected resources

a. Insecure Direct Object Reference (IDOR)

Observation:
Users could change the id parameter in the URL to access or modify other users' data.

PoC Example:

1. Logged in as user ID = 2

2. Accessed: http://127.0.0.1/DVWA/user_profile.php?id=1

3. Was able to view user ID 1's profile details

Impact:
Unauthorized data access leading to privacy violations.

Remediation:

- Enforce server-side access control checks.
- Avoid using predictable user IDs in URLs.

b. Unauthorized Role Escalation

Observation:
Non-admin users could access admin functionalities by directly accessing admin endpoints.

PoC:
Accessed http://127.0.0.1/DVWA/admin/view_users.php as a regular user.

Impact:
Privilege escalation without proper authentication.

Remediation:

- Check roles and privileges in the backend before serving admin resources.
- Hide admin UI elements for non-admin users.

c. Forced Browsing

Observation:
Restricted pages like reset_db.php and setup.php were accessible after logout.

Impact:
Could allow attackers to reset the application state or view system setup info.

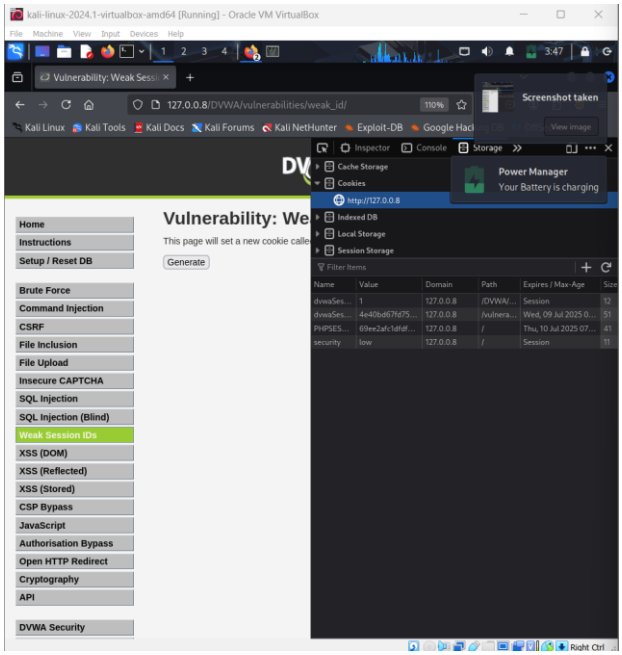
Remediation:

- Enforce login session checks on all sensitive pages.
- Redirect unauthorized access to login or error pages.

Proof-of-Concept (PoC) Deliverables

- Annotated screenshots of exploitation
- Payloads and manipulated URLs
- Video clips (optional) demonstrating attacks
- Logs and captured requests via Burp Suite

Screeshorts (Weak session IDs)



Week 4:

1. Consolidate findings from all prior weeks.
2. Finalize and compile a comprehensive penetration testing report.
3. Include step-by-step exploitation procedures and recommendations.

1. Retesting and Validation

Before finalizing the report, all previously discovered vulnerabilities were **retested** to:

- Ensure they are still exploitable.
- Validate consistency across different user roles.
- Confirm that exploitation steps work without any changes to the environment.

Validated Vulnerabilities:

Vulnerability	Application	Confirmed
SQL Injection	DVWA	✅ Yes
Stored XSS	DVWA	✅ Yes
Reflected XSS	DVWA	✅ Yes

Vulnerability	Application	Confirmed
IDOR	DVWA	✅ Yes
Misconfiguration (Admin Panel)	DVWA	✅ Yes
Insecure Cookies	DVWA	✅ Yes

2. Final Penetration Testing Report

The final report was structured to meet professional standards and industry practices.

Report Sections:

Executive Summary:

- Brief overview of the engagement.
- Objectives of the test.
- Summary of vulnerabilities found and risk ratings.

Scope:

- Target applications: **DVWA** and **OWASP Juice Shop**
- Local testing environment.
- OWASP Top 10 vulnerabilities as the core focus.

Methodology:

- Manual and automated testing.
- Tools used:
    - **Burp Suite** for interception and manipulation
    - **sqlmap** for automated SQL injection
    - **Browser DevTools** for source code inspection and XSS
    - **dirb/gobuster** for directory brute-forcing
    - **Postman** for testing API endpoints

Vulnerability Findings:

Each vulnerability included:

- **Title**
- **Description**
- **Proof-of-Concept (PoC)** with screenshots

- **Impact**
- **Affected components**
- **CVSS rating**
- **Mitigation/Recommendation**

Examples:

Vulnerability	**Description**	**CVSS Score**	**Status**
SQL Injection	Input fields vulnerable to unsanitized SQL queries	9.1 (Critical)	Confirmed
Stored XSS	Payload stored in database and executed on load	8.8 (High)	Confirmed
Security Misconfig	Unrestricted access to admin dashboard	8.2 (High)	Confirmed

Risk Ratings:

Each vulnerability was rated based on:

- Exploitability
- Impact
- Likelihood
- Business risk

Recommendations:

Clear, actionable mitigation steps for each issue:

- Use input validation & output encoding
- Implement proper session management
- Configure web server securely
- Avoid using default or guessable endpoints
- Enable HTTPS and set security headers

We have successfully performed and demonstrated both Stored and Reflected XSS attacks, and an example demo video showcasing these attacks has been posted on LinkedIn for reference.

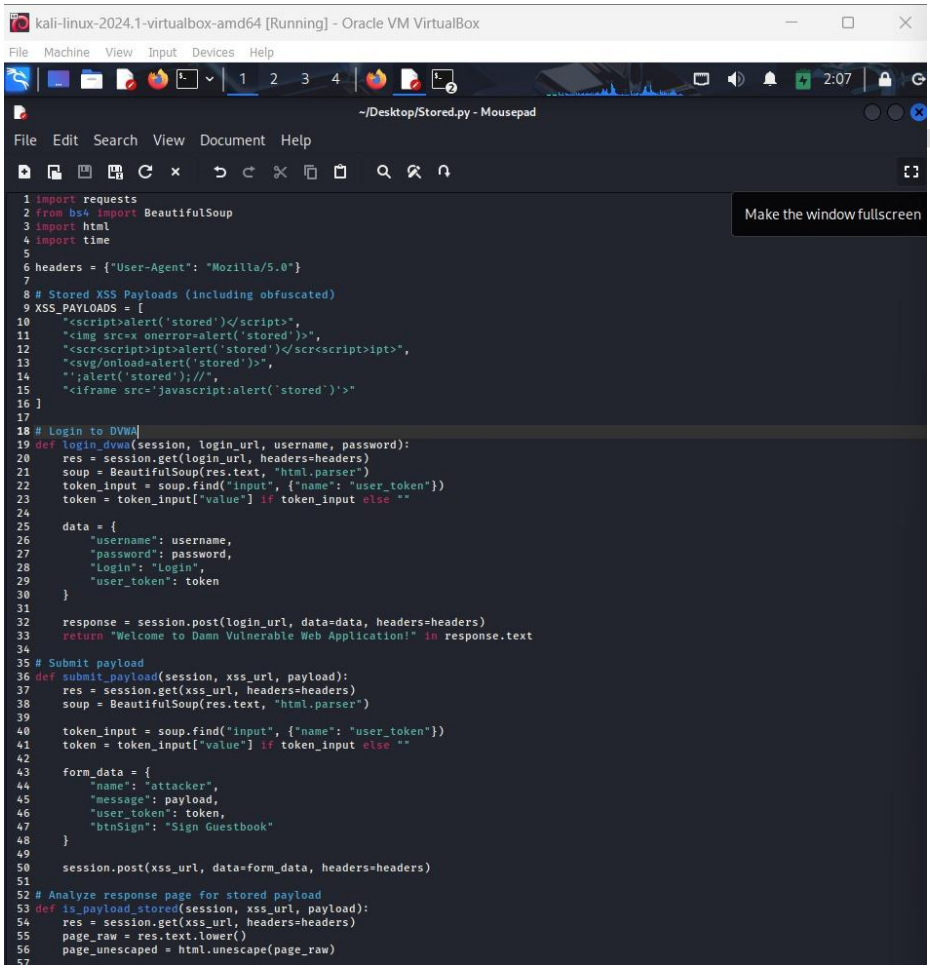
link: https://www.linkedin.com/posts/vishnu-sudarsan-e-p-9965a6292_cybersecurity-internshipexperience-project2-activity-7348630002092920833-PEAq?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEbsNAIBgDhs07cvmT6ZFFdYWZ3cRAwRS8k

As an additional finding, the application was detected to be vulnerable to **Stored Cross-Site Scripting (XSS)**.
Malicious scripts injected into input fields were permanently stored in the database.

PERFORMING XXS DETECTION ON DVWA

STORED XXS DETECTION

Code Snippet



Output Snippet