

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

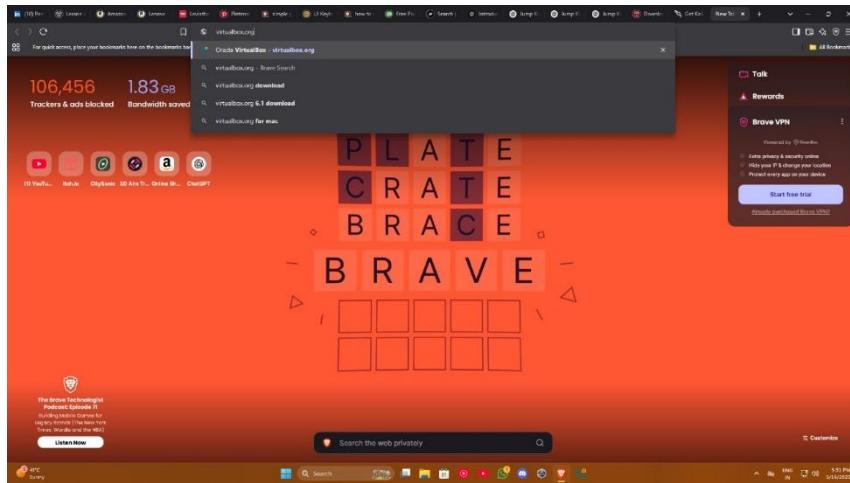
Simulate a real-world vulnerability assessment for a small business IT infrastructure. Identify security

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

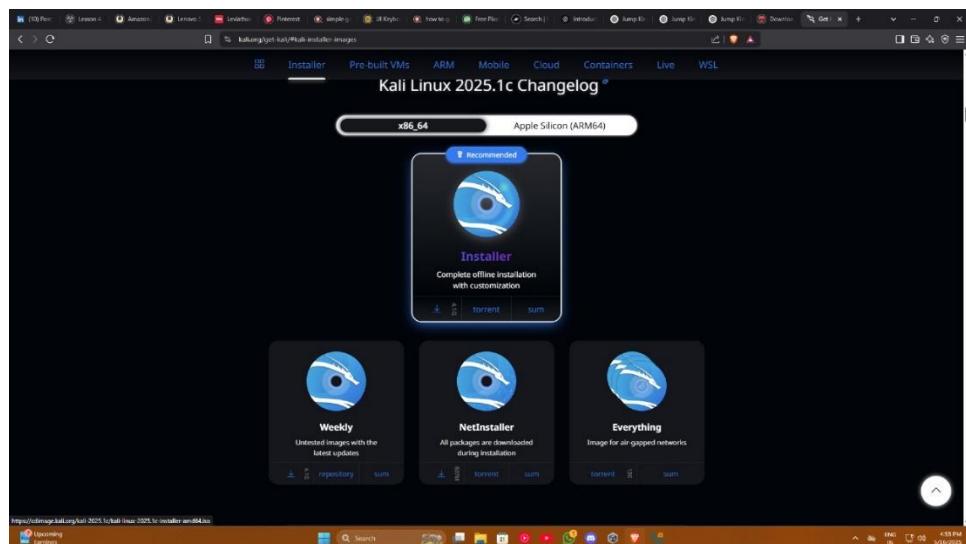
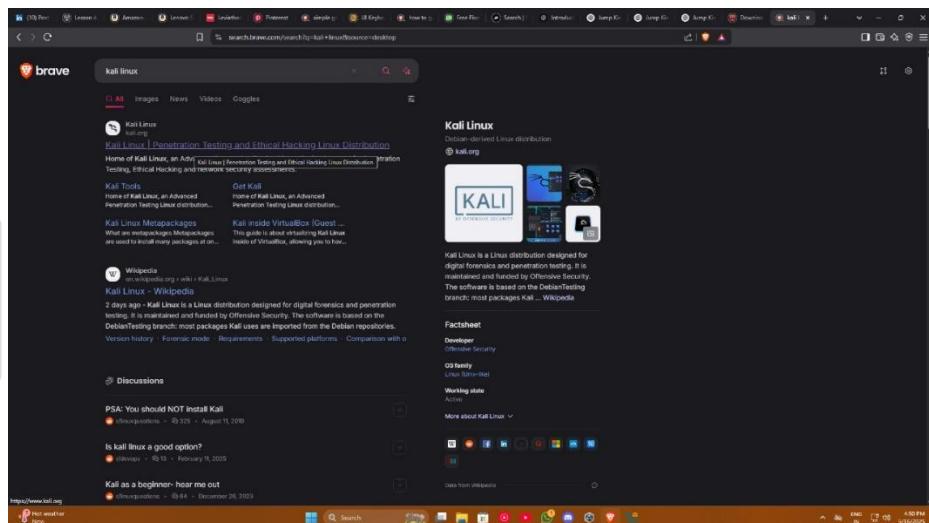
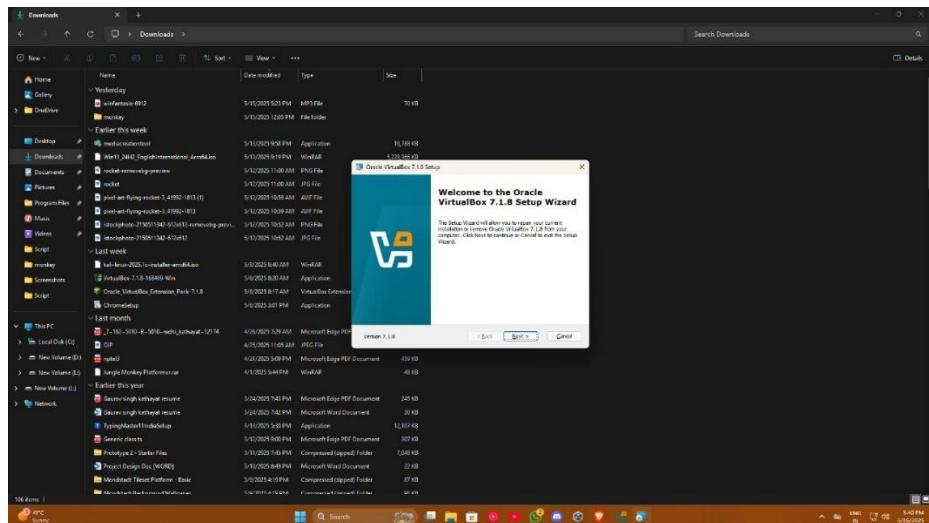
Week 1

- 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



INTERNSHIP PROJECT REPORT



INTERNSHIP PROJECT REPORT



We have practiced Nmap Scanning using the “TRYHACKME”

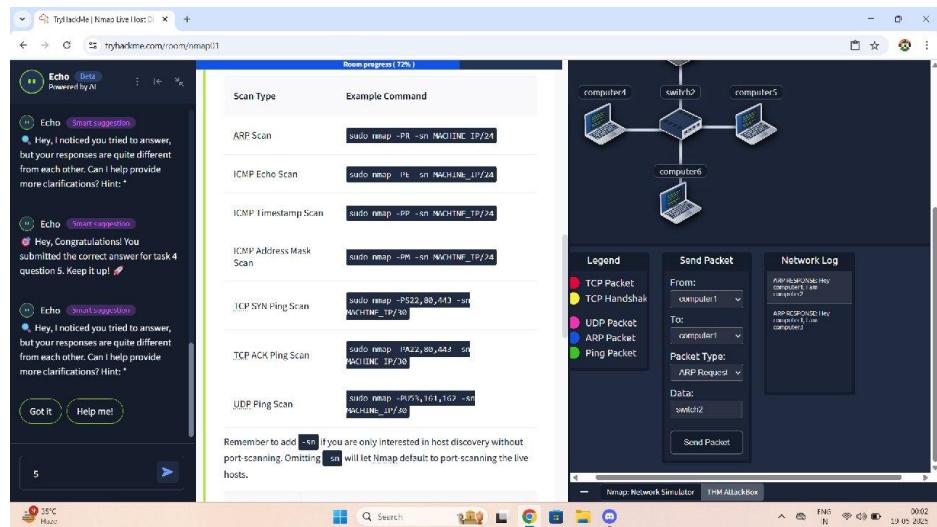
Below Screen Shorts for example

The screenshot shows a browser window for the TryHackMe Nmap Lab. On the left, there's a sidebar with a green 'Echo' icon and a progress bar at 33%. The main area contains a question about ARP requests and a terminal window showing Nmap scan reports for multiple hosts. The terminal output includes:

```
Nmap scan report for 10.10.255.115
Nmap scan report for 10.10.255.116
Nmap scan report for 10.10.255.117
Nmap scan report for 10.10.255.118
Nmap scan report for 10.10.255.119
Nmap scan report for 10.10.255.120
Nmap scan report for 10.10.255.121
Nmap scan report for 10.10.255.122
Nmap scan report for 10.10.255.123
Nmap scan report for 10.10.255.124
Nmap scan report for 10.10.255.125
Nmap done: 6400 IP addresses (0 hosts up) scanned in 11.67 seconds
root@tp-10-10-181-26:~# nmap -sn 10.10.12.13/29
Nmap scan report for 10.10.12.13
Nmap scan report for 10.10.12.14
Nmap scan report for 10.10.12.15
Nmap scan report for 10.10.12.16
Nmap scan report for 10.10.12.17
Nmap scan report for 10.10.12.18
Nmap scan report for 10.10.12.19
Nmap scan report for 10.10.12.20
Nmap scan report for 10.10.12.21
Nmap scan report for 10.10.12.22
Nmap scan report for 10.10.12.23
Nmap scan report for 10.10.12.24
Nmap done: 8 IP addresses (8 hosts up) scanned in 0.02 seconds
root@tp-10-10-181-26:~#
```

The screenshot shows a browser window for the TryHackMe Nmap Lab. On the left, there's a sidebar with a green 'Echo' icon and a progress bar at 59%. The main area contains a question about ping requests and a network simulation interface. The simulation interface shows a network topology with four computers connected to a switch. A legend identifies packet types: TCP Packet (red), TCP Handshake (yellow), UDP Packet (pink), ARP Packet (blue), and Ping Packet (green). The 'Send Packet' section shows a 'From' dropdown set to 'computer2', a 'To' dropdown set to 'computer5', a 'Packet Type' dropdown set to 'Ping Request', and a 'Data:' input field containing 'PING computer3'. The 'Network Log' section shows a log entry for an ARP REQUEST from computer2 to computer5.

INTERNSHIP PROJECT REPORT



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

INTERNSHIP PROJECT REPORT



Command : nmap -A 45.33.32.156

```
nmap -A 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 12:15 IST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.00s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.6.1p1 Ubuntu 2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 ac:00:a0:1a:b2:ff:cc:55:90:dc:87:2b:34:97:0b:75 (RSA)
|   2048 30:3d:3e:4f:5c:53:55:5b:5d:5e:5f:5a:5c:4a:24:02:57 (ECDSA)
|_  256 96:02:bd:5e:57:54:1c:ac:45:0f:58:4c:4a:24:02:57 (ED25519)
Nmap done: 1 IP address (1 host up) scanned in 38.69 seconds
nmap@NidhiJi:~$
```

Command : nmap -O 45.33.32.156

```
nmap -O 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 12:14 IST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.00s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE OS
22/tcp    open  ssh     Linux 5.0 - 5.4 (91%)
22/tcp    open  ssh     Linux 4.15 - 5.0 (9%)
80/tcp    open  http    Aggressive OS guesses: Linux 5.0 (91%), Linux 5.0 - 5.4 (91%), Linux 4.15 - 5.0 (9%)
993/tcp   open  https   Aggressive OS guesses: Linux 5.0 (91%), Linux 5.0 - 5.4 (91%), Linux 4.15 - 5.0 (9%)
No exact OS guess. For most (test conditions non-ideal).
Network Distance: 20 hops
OS detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 51.65 seconds
nmap@NidhiJi:~$
```

Command : nmap -sT 45.33.32.156

```
nmap -sT 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 12:03 IST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.28s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
993/tcp   open  https
31337/tcp open  Elite
Nmap done: 1 IP address (1 host up) scanned in 39.97 seconds
nmap@NidhiJi:~$
```

Command : nmap -sV 45.33.32.156

INTERNSHIP PROJECT REPORT



```
nmap -sS 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 12:09 IST
Map scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.43s latency).
Nmap scan timing rules are being applied.
PORT      STATE SERVICE VERSION
22/tcp    open  ssh   OpenSSH 8.0.1p1 Ubuntu 2ubuntu0.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http  Apache/2.4.7 (Ubuntu)
9929/tcp  open  nginx-echo  Nginx echo
9137/tcp  open  tcpcrypt
Service info: OS: Linux; CPU: generic; OS:Ubuntu,Kernel: 5.4.0-132-generic
Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 608.27 seconds
nmap@nmapjali:[~]
```

Command : nmap -sS 45.33.32.156

```
nmap -sn 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 12:01 IST
Map scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.43s latency).
Nmap scan timing rules are being applied.
PORT      STATE SERVICE
22/tcp    closed
80/tcp    closed
9929/tcp  closed
9137/tcp  closed
Nmap done: 1 IP address (1 host up) scanned in 17.37 seconds
nmap@nmapjali:[~]
```

Command : nmap -sn 45.33.32.156

```
nmap -sC 45.33.32.156
Starting Nmap 7.94SN ( https://nmap.org ) at 2025-05-23 11:59 IST
Map scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.0046s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds
nmap@nmapjali:[~]
```

Command : nmap -sC 45.33.32.156

INTERNSHIP PROJECT REPORT



```
[nmap@nmap1 ~]$ nmap -sS 45.33.32.156
Starting Nmap 7.94SVE ( https://nmap.org ) at 2025-05-23 12:19 IST
Nmap scan Report for scanner.nmap.org (45.33.32.156)
Host scanner.nmap.org (45.33.32.156) is up.
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
ssh-hostkey:
 18000000:00:1a:82:ff:cc:50:99:dc:b7:2b:3b:97:6b:75 (RSA)
  948:20:3d:20:44:62:24:1b:5a:96:05:03:05:14:c7:a6:62 (RSA)
  25:96:02:00:56:07:59:41:c1:ce:65:2f:f5:64:c4:a4:24:b2:57 (ECDSA)
  256:83:7a:91:9e:0e:e1:7b:1f:bd:05:a2:0b:f1:54:a1:56 (ED25519)
80/tcp    open  http
|_http-favicon: Nmap Project
|_http-title: Go Home and ScanMe!
|_http-robots.txt: /robots.txt
|_http-server-header: Apache/2.4.41 (Ubuntu)
31337/tcp open  Elite

Nmap done: 1 IP address (1 host up) scanned in 28.65 seconds
[nmap@nmap1 ~]$
```

Command : nmap -p 80,443 45.33.32.156

```
[nids@NidhiJi11:~] $ nmap -p 80,443 -oN scan.txt 45.13.32.156
Starting Nmap 7.91 ( https://nmap.org ) at 2025-05-23 12:21 IST
Nmap scan report for scanner imap.org (45.13.32.156)
Host is up (0.067s latency).

PORT      STATE SERVICE
80/tcp    open  http
443/tcp   closed https

Nmap done: 1 IP address (1 host up) scanned in 0.77 seconds
[nids@NidhiJi11:~]
```

Command : nmap 192.168.1.1-5

INTERNSHIP PROJECT REPORT

A screenshot of a terminal window titled 'nmap@Nikhiljii ~'. The window displays the output of an Nmap scan for the IP address 45.33.32.156. The output shows that the scan started at 12:35 IST on May 23, 2025, and completed in 1.58 seconds, scanning 5 IP addresses. The results indicate that 8 hosts are up. The terminal also shows a message from the user 'nmap' asking for permission to scan ports 22, 23, 25, 53, 80, 110, 143, 443, and 465.

Explanation :

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

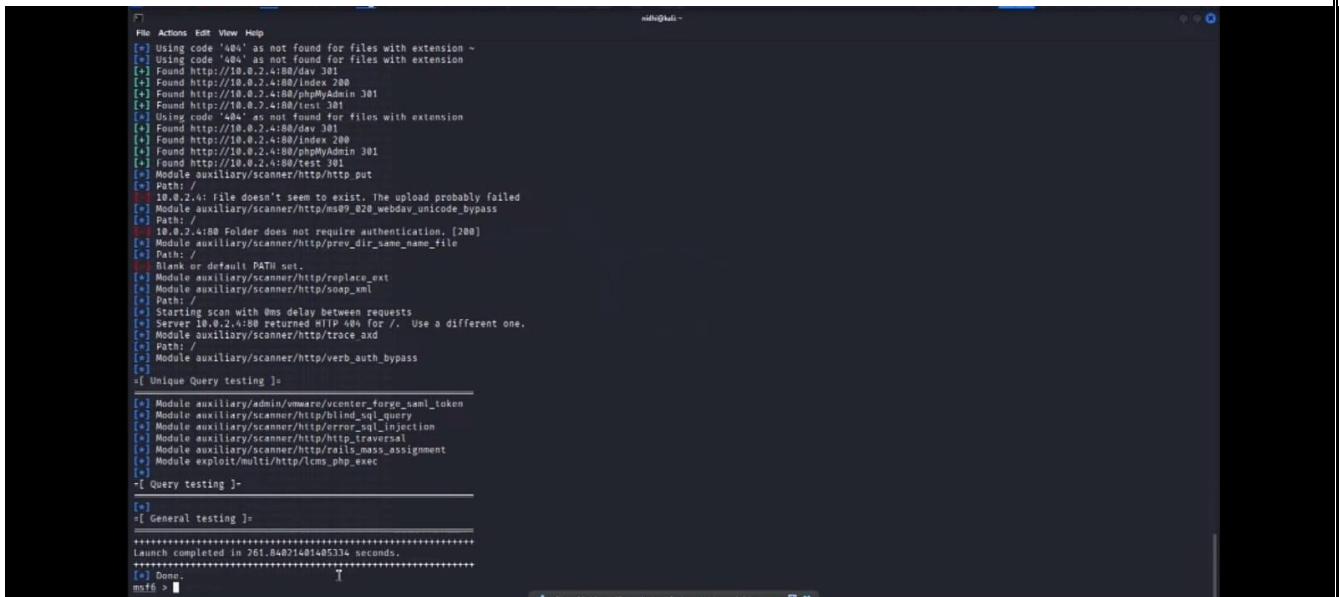
Scans IPs from 192.168.1.1 to 192.168.1.5 using default settings.

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 Nmapse.

INFOTACT INTERN



```

File Actions Edit View Help
[*] Using code '404' as not found for files with extension ~
[*] Found http://10.0.2.4:80/dav 301
[*] Found http://10.0.2.4:80/index 200
[*] Found http://10.0.2.4:80/index.html 200
[*] Found http://10.0.2.4:80/index.htm 301
[*] Using code '404' as not found for files with extension .
[*] Found http://10.0.2.4:80/dav 301
[*] Found http://10.0.2.4:80/index 200
[*] Found http://10.0.2.4:80/phpMyAdmin 301
[*] Found http://10.0.2.4:80/test 301
[*] Module auxiliary/scanner/http/http_put
[*] Path: /
[*] 10.0.2.4: File doesn't seem to exist. The upload probably failed
[*] Module auxiliary/scanner/http/ms09_020_webdav_unicode_bypass
[*] Path: /ms09_020_webdav_unicode_bypass
[*] 10.0.2.4:80 Folder does not require authentication. [200]
[*] Module auxiliary/scanner/http/prev_dir_same_name_file
[*] Path: /
[*] Blah or default PATH set.
[*] Module auxiliary/scanner/http/replace_ext
[*] Module auxiliary/scanner/http/soap_xml
[*] Path: /
[*] Starting scan with 0ms delay between requests
[*] Server 10.0.2.4:80 returned HTTP 404 for /. Use a different one.
[*] Module auxiliary/scanner/http/trace_aid
[*] Path: /
[*] Module auxiliary/scanner/http/verb_auth_bypass
[*] Unique Query testing :
[*] Module auxiliary/admin/vmware_forge_saml_token
[*] Module auxiliary/scanner/http/blind_sql_query
[*] Module auxiliary/scanner/http/error_sql_injection
[*] Module auxiliary/scanner/http/http_traversal
[*] Module auxiliary/scanner/http/rails_mass_assignment
[*] Module exploit/multi/http/lcms_php_exec
[*] Unique Query testing :
[*] General testing :
=====
Launch completed in 261.84821401405334 seconds.
[*] Done.
nmap > 
  
```

Vulnerability (1)

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

INTERNSHIP PROJECT REPORT



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

```
[*] Actions Edit View Help
[*] [10.0.2.4] Sending request with random domain YevVt.
[*] [10.0.2.4] Sending request with random domain zFMwY.
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] [10.0.2.4] (Apache/2.4 (Ubuntu) PHP/7.2.17) Webdav disabled.
[*] Module auxiliary/scanner/http/webdav_unicode_content
[*] -[ FileDir testing ]-
[*] Module auxiliary/scanner/http/backup_file
[*] Module auxiliary/scanner/http/brute_dirs
[*] Path: /
[*] Using code '404' as not found.
[*] Found http://10.0.2.4:80/dav/ 200
[*] Found http://10.0.2.4:80/doc/ 200
[*] Module auxiliary/scanner/http/copy_of_file
[*] Module auxiliary/scanner/http/dir_listing
[*] Path: /
[*] Module auxiliary/scanner/http/dir_scanner
[*] Path: /
[*] Detecting error code
[*] Using code '404' as not found for 10.0.2.4
[*] Found http://10.0.2.4:80/cgi-bin/ 403 (10.0.2.4)
[*] Found http://10.0.2.4:80/doc/ 200 (10.0.2.4)
[*] Found http://10.0.2.4:80/icons/ 200 (10.0.2.4)
[*] Found http://10.0.2.4:80/images/ 200 (10.0.2.4)
[*] Found https://10.0.2.4:80/phpMyAdmin/ 200 (10.0.2.4)
[*] Found https://10.0.2.4:80/test/ 404 (10.0.2.4)
[*] Module auxiliary/scanner/http/diwebdav_unicode_bypass
[*] Path: /
[*] Using code '404' as not found.
[*] Module auxiliary/scanner/http/file_name_nano_dir
[*] Path: /
[*] Blank or default PATH set.
[*] Module auxiliary/scanner/http/files_dir
[*] Path: /
[*] Using code '404' as not found for files with extension .null
[*] Using code '404' as not found for files with extension .backup
[*] Using code '404' as not found for files with extension .lock
[*] Using code '404' as not found for files with extension .c
[*] Using code '404' as not found for files with extension .cfg
[*] Using code '404' as not found for files with extension .class
[*] Using code '404' as not found for files with extension .ppp
[*] Using code '404' as not found for files with extension .conf
[*] Using code '404' as not found for files with extension .exe
[*] Using code '404' as not found for files with extension .html
[*] Using code '404' as not found for files with extension .htm
```

Vulnerability(2)

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

```
[File Actions Edt View Help -neth@ali -]
[*] Target is not SSL. SSL modules disabled.
[*]
[*] [ Web Server testing ]:
[*] Module auxiliary/scanner/http/http_version
[*] 10.0.2.4:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Module auxiliary/scanner/http/enum_http_headers
[*] Module auxiliary/admin/http/webmin_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Attempting to connect to 10.0.2.4:80
[*] No file(s) found
[*] Module auxiliary/scanner/http/drupal_views user enum
[*] 10.0.2.4 does not appear to be vulnerable, will not continue
[*] Module auxiliary/scanner/http/frontpage_login
[*] 10.0.2.4:80 - http://10.0.2.4/ may not support FrontPage Server Extensions
[*] Module auxiliary/scanner/http/host_header_injection
[*] 10.0.2.4:80 - http://10.0.2.4/
[*] Module auxiliary/scanner/http/reboot
[*] Module auxiliary/scanner/http/scrapbox
[*] [10.0.2.4] / Metasploitable2 - Linux
[*] Module auxiliary/scanner/http/svn_scanner
[*] Using code 404 as not found.
[*] Module auxiliary/scanner/http/trace
[*] 10.0.2.4:80 is vulnerable to Cross-Site Tracing
[*] Auxiliary failed: Msf::Validation::Error failed to report vuln for 10.0.2.4:80 to the database
Call stack:
[*] /usr/share/metasploit-framework/lib/msf/core(auxiliary/report.rb:30:in `report_vuln'
[*] /usr/share/metasploit-framework/modules/auxiliary/scanner/http/trace.rb:47:in `run_host'
[*] /usr/share/metasploit-framework/lib/msf/core/auxiliary/scanner/rb116:in `block (2 levels) in run'
[*] /usr/share/metasploit-framework/lib/msf/core/thread_manager.rb:105:in `block in spawn'
[*] /usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/logging-2.4.0/lib/logging/diagnostic_context.rb:474:in `block in create_with_logging_context'
[*] [ File/DIR testing ]:
[*] [*] [10.0.2.4] Sending request with random domain rFWV.
[*] [10.0.2.4] Sending request with random domain rFWV.
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] 10.0.2.4 (Apache/2.2.8 (Ubuntu) DAV/2) WebDAV disabled.
[*] Module auxiliary/scanner/http/webdav_website_content
[*]
[*] [ File/DIR testing ]:
[*] [*] Module auxiliary/scanner/http/backup_file
[*] [*] Module auxiliary/scanner/http/brute_dirs
[*] Path: /
[*] Using code '404' as not found.
[*] Found http://10.0.2.4:80/dev/ 200
[*] Found http://10.0.2.4:80/doc/ 200
```

Vulnerability(3)

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

INTERNSHIP PROJECT REPORT



```
[File Actions Edit View Help]
[+] Module auxiliary/scanner/http/soap_xml
[+] Module auxiliary/scanner/http/trace_axd
[+] Module auxiliary/scanner/http/verb_auth_bypass
[-] Unique Query testing :
[+] Module auxiliary/admin/http/recenter_forge_semi_token
[+] Module auxiliary/scanner/http/blind_xss
[+] Module auxiliary/scanner/http/error_sql_injection
[+] Module auxiliary/scanner/http/http_traversal
[+] Module auxiliary/scanner/http/rolls_mass_assignment
[+] Module exploit/multi/http/lcms_php_exec
[+] Query testing :
[+]
[-] General testing :
[+]
[-] Done.

msf6 > wmap.run -e
[*] Using All WMAP enabled modules.
[*] NO TARGET NODES DEFINED. Executing local modules
[*] Testing module: http
[*]   Site: 10.0.2.4 (10.0.2.4)
[*]   Port: 80 SSL: false

[*] Testing started: 2025-05-27 15:48:59 +0530
[*]
[-] SSL testing :

[*] Target is not SSL. SSL modules disabled.

[*]
[-] Web Server testing :
[+]
[-] Unique Query testing :
[+] Module auxiliary/scanner/http/http_version
[*] 10.0.2.4:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.18 )
[*] 10.0.2.4:80 ModSecurity/2.9.1.3 (http://www.modsecurity.org/projects/modsecurity/modsecurity.html)
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Attempting to connect to 10.0.2.4:80

[*] Module auxiliary/scanner/http/drupal_views_user_enum
[*] 10.0.2.4 does not appear to be vulnerable, will not continue
[*] Module auxiliary/scanner/http/aspnet_injection
[*] 10.0.2.4:80 - http://10.0.2.4/ does not support FrontPage Server Extensions
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
```

Vulnerability(4)

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

INFOTACT INTERN

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.

INFOTACT INTERN

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

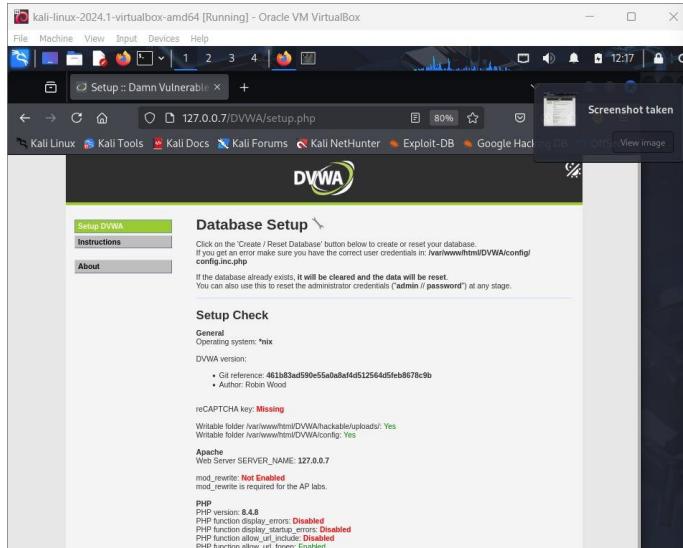
FINAL DVWA SETUP LINK

<http://127.0.0.8/dvwa/setup.php>

INTERNSHIP PROJECT REPORT



```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ service mysql start
test.txt selenium-env VanSpy-17...
(kali㉿kali)-[~]
$ service apache2 start
```



Example screenshots 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):

INTERN

- Reflected XSS:
Injected: <script>alert('XSS')</script>
- Stored XSS:
Injected:
- 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 Screenshots:**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>

INTERNSHIP PROJECT REPORT

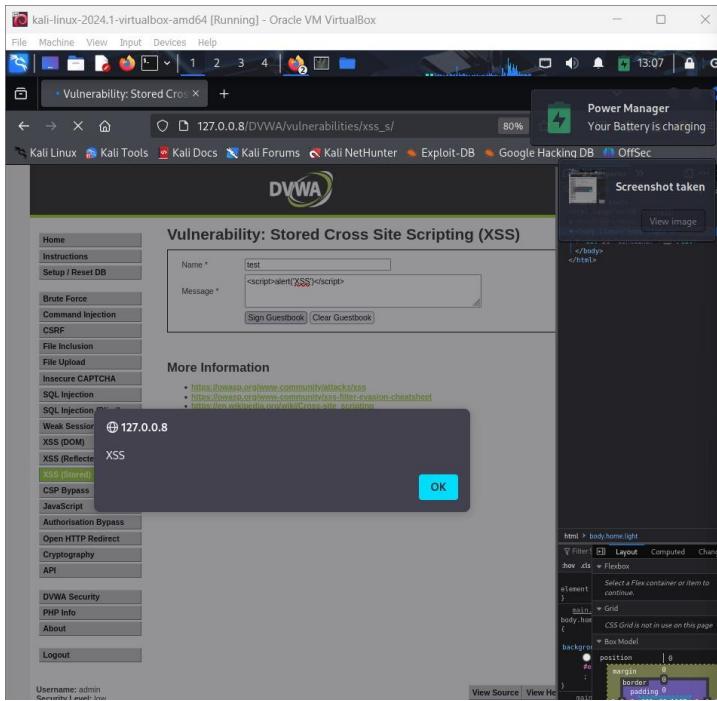


A screenshot of the DVWA (Damn Vulnerable Web Application) Security page. The URL is 127.0.0.8/DVWA/security.php. The security level is currently set to 'low'. A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, and Authorisation Bypass. The 'XSS (Stored)' option is highlighted in green.

A screenshot of the DVWA XSS (Stored) vulnerability page. The URL is 127.0.0.8/DVWA/vulnerabilities/xss_s/. The user has entered 'test' in the 'Name' field and '' in the 'Message' field. Below the form, there are links for 'Sign Guestbook' and 'Clear Guestbook'. The sidebar on the left shows the 'XSS (Stored)' option highlighted in green.

INTERN

INTERNSHIP PROJECT REPORT



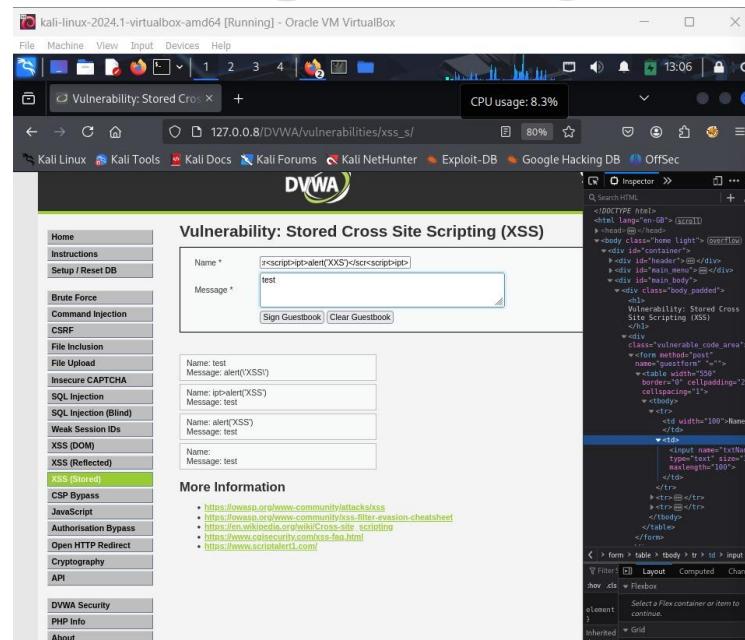
A screenshot of a Kali Linux VM running DVWA. The browser window shows the 'Vulnerability: Stored Cross Site Scripting (XSS)' page. In the 'Name' field, the user has entered '<script>alert('XSS')</script>'. The 'Message' field contains 'test'. Below the form, a 'More Information' section lists several XSS resources. On the right, a developer tools panel shows the DOM structure of the page, specifically focusing on the body element which contains the injected script.

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



A screenshot of a Kali Linux VM running DVWA. The browser window shows the 'Vulnerability: Stored Cross Site Scripting (XSS)' page. In the 'Name' field, the user has entered '<script>ipt<alert('XSS')</script>'. The 'Message' field contains 'test'. Below the form, a 'More Information' section lists several XSS resources. On the right, a developer tools panel shows the DOM structure of the page, specifically focusing on the body element which contains the injected script. The inspection results show the raw HTML code: <script>ipt<alert('XSS')</script>.

INTERNSHIP PROJECT REPORT



A screenshot of the DVWA application running in a Kali Linux VM. The user has selected the 'XSS (Stored)' vulnerability. In the 'Message' field, they have entered the payload: <script>alert('XSS')</script>. The 'OK' button is highlighted. The browser's developer tools are open, showing the source code of the page. The 'Inspector' tab is active, displaying the HTML structure of the page, including the injected script tag.

On LOW Security

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

Name : test

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

A screenshot of the DVWA application running in a Kali Linux VM. The user has selected the 'XSS (Reflected)' vulnerability. In the 'What's your name?' field, they have entered the payload: <script>alert('reflected')</script>. The 'Submit' button is highlighted. The browser's developer tools are open, showing the source code of the page. The 'Inspector' tab is active, displaying the HTML structure of the page, including the injected script tag.

INTERNSHIP PROJECT REPORT



A screenshot of the DVWA application. The URL is `http://127.0.0.8/DVWA/vulnerabilities/xss_r/?name=<script>alert('reflected')</script>`. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, there's a sidebar with various security testing options. In the main area, there's a form with a placeholder "What's your name?". A user has entered "`<script>alert('reflected')</script>`". Below the form, there's a "Submit" button. Underneath the form, the text "Hello" is displayed. A modal dialog box is open, showing the IP address "127.0.0.8" and the word "reflected", with an "OK" button.

On MEDIUM Security

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

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

Message : test

A screenshot of the DVWA application. The URL is `http://127.0.0.8/DVWA/vulnerabilities/view_source.php?id=xss_r&security=medium`. The page title is "Reflected XSS Source". The main content area shows the source code of a PHP file named `vulnerabilities/xss_r/source/medium.php`. The code includes logic to check if there is any input and replace it. At the bottom of the code, there is a comment: // Feedback for end user echo "<script>alert('reflected')</script>";. The sidebar on the left is identical to the previous screenshot. At the bottom, there are links for "View Source" and "View Help".

INTERNSHIP PROJECT REPORT



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

A screenshot of a Kali Linux VM running DVWA. The browser window shows the 'Reflected Cross Site Scripting (XSS)' page. In the input field, the user has entered: What's your name? <scRiPt>alert('reflected')</scRiPt>. Below the input field is a 'Submit' button. To the right of the input field, there is an error message: 'Parse error: syntax error, unexpected '<scRiPt>', expecting '<script>' at line 1, column 1'. The DVWA sidebar on the left lists various vulnerabilities, with 'Reflected' highlighted.

A screenshot of the same DVWA setup, but now showing a successful exploit. The input field contains the same payload: What's your name? <scRiPt>alert('reflected')</scRiPt>. The error message is no longer present. Instead, a modal dialog box appears with the text 'reflected' and an 'OK' button. The DVWA sidebar remains visible on the left.

On HIGH Security

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

Name:

Message : test

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

INTERNSHIP PROJECT REPORT



kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

Machine View Input Devices Help

Setup :: Damn Vulnerable DVWA Vulnerability: Reflected

→ C 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

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/>

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)

kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Setup :: Damn Vulnerable DVWA 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

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

More Information

127.0.0.8 reflected

OK

https://www.exploit-db.com/google-hacking-database

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 Open HTTP Redirect Cryptography API

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:

INTERNSHIP PROJECT REPORT



- 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

INTERNSHIP PROJECT REPORT



Screeshorts (Weak session IDs)

A screenshot of a Kali Linux VM running DVWA. The browser window shows the 'Weak Session IDs' page at 127.0.0.8/DVWA/vulnerabilities/weak_id/. The sidebar menu is visible on the left, and the main content area displays a table of session cookies. The table has columns for Name, Value, Domain, Path, Expires / Max-Age, and Size. Several session cookies are listed, including 'dvsess...', 'PHPSESSID...', and 'security'. The 'Expires / Max-Age' column shows values like 'Session', 'Wed, 09 Jul 2025 0...', and 'Thu, 10 Jul 2025 0...'. The 'Size' column shows values like '12', '51', and '11'.

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	<input checked="" type="checkbox"/> Yes
Stored XSS	DVWA	<input checked="" type="checkbox"/> Yes
Reflected XSS	DVWA	<input checked="" type="checkbox"/> Yes

INTERNSHIP PROJECT REPORT



Vulnerability	Application	Confirmed
IDOR	DVWA	<input checked="" type="checkbox"/> Yes
Misconfiguration (Admin Panel)	DVWA	<input checked="" type="checkbox"/> Yes
Insecure Cookies	DVWA	<input checked="" type="checkbox"/> 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

INTERNSHIP PROJECT REPORT



- 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

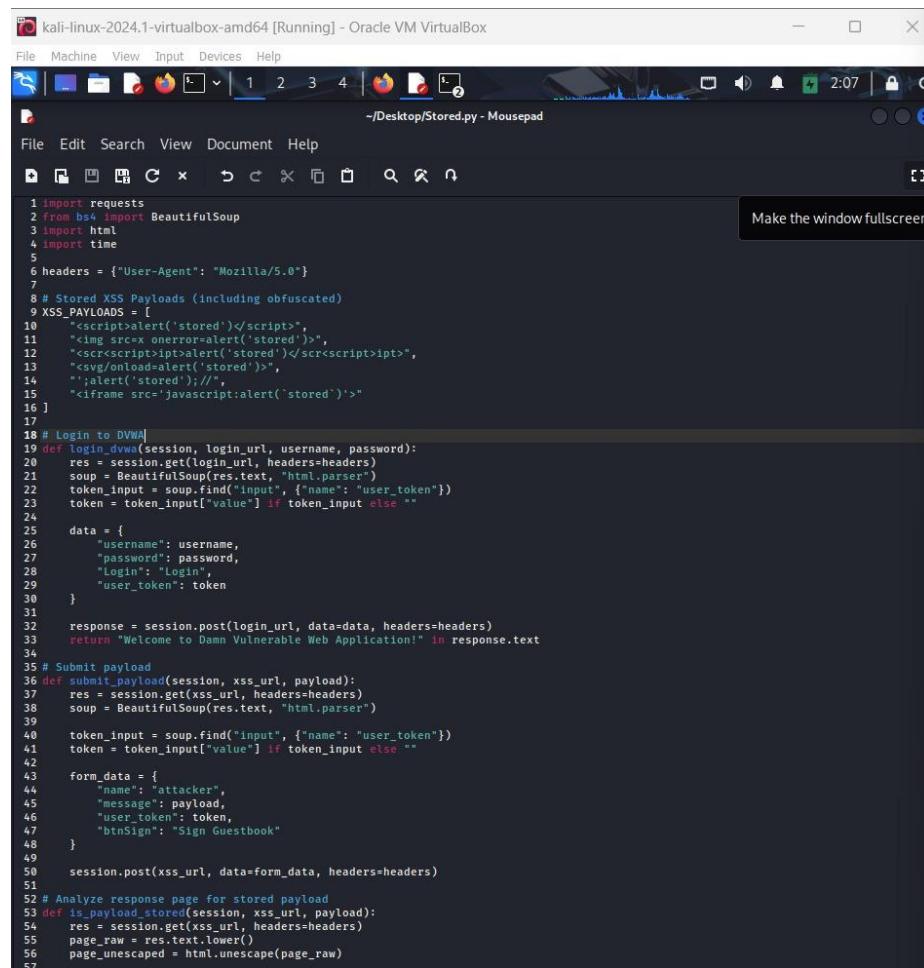
As an additional finding, the application was detected to be vulnerable to **Stored Cross-Site Scripting (XSS)**.

Malicious scripts injected into input fields were permanently stored in the database.

PERFORMING XXS DETECTION ON DVWA

STORED XXS DETECTION

Code Snippet



```

kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit Search View Document Help
File Manager Terminal ~Desktop/Stored.py - Mousepad
File Edit Search View Document Help
File Manager Terminal ~Desktop/Stored.py - Mousepad
1 import requests
2 from bs4 import BeautifulSoup
3 import html
4 import time
5
6 headers = {"User-Agent": "Mozilla/5.0"}
7
8 # Stored XSS Payloads (including obfuscated)
9 XSS_PAYLOADS = [
10     "<script>alert('stored')</script>",
11     "<img src=x onerror=alert('stored')>",
12     "<scr>script<ipt>alert('stored')</scr><script>ipt>",
13     "<svg/onload=alert('stored')>",
14     "&#x31;<alert('stored')//",
15     "<i>frame src='javascript:alert('stored')'>"
16 ]
17
18 # Login to DVWA
19 def login_dvwa(session, login_url, username, password):
20     res = session.get(login_url, headers=headers)
21     soup = BeautifulSoup(res.text, "html.parser")
22     token_input = soup.find("input", {"name": "user_token"})
23     token = token_input["value"] if token_input else ""
24
25     data = {
26         "username": username,
27         "password": password,
28         "Login": "Login",
29         "user_token": token
30     }
31
32     response = session.post(login_url, data=data, headers=headers)
33     return "Welcome to Damn Vulnerable Web Application!" in response.text
34
35 # Submit payload
36 def submit_payload(session, xss_url, payload):
37     res = session.get(xss_url, headers=headers)
38     soup = BeautifulSoup(res.text, "html.parser")
39
40     token_input = soup.find("input", {"name": "user_token"})
41     token = token_input["value"] if token_input else ""
42
43     form_data = {
44         "name": "attacker",
45         "message": payload,
46         "user_token": token,
47         "btnSign": "Sign Guestbook"
48     }
49
50     session.post(xss_url, data=form_data, headers=headers)
51
52 # Analyze response page for stored payload
53 def is_payload_stored(session, xss_url, payload):
54     res = session.get(xss_url, headers=headers)
55     page_raw = res.text.lower()
56     page_unescaped = html.unescape(page_raw)
57

```

Output Snippet

```
└─(selenium-env)─(kali㉿kali)─[~/Desktop]
└─$ python3 Stored.py
Enter DVWA login URL: http://127.0.0.8/DVWA/login.php
Enter Stored XSS (guestbook) URL: http://127.0.0.8/DVWA/vulnerabilities/xss_s/
Enter DVWA username: admin
Enter DVWA password: password
[+] Login successful.
[+] Submitting payload: <script>alert('stored')</script>
✗ Stored XSS Detected with payload: <script>alert('stored')</script>
[+] Submitting payload: <img src=x onerror=alert('stored')>
[+] Submitting payload: <svg/onload=alert('stored')>
[+] Submitting payload: ';alert('stored');//'
[+] Submitting payload: <iframe src='javascript:alert(`stored`)'>
```

INFOTACT INTERN