Stage-1

Understanding various vulnerabilities

TOP 5 VUNERABILITIES EXPLOITATION:

S. No	Vulnerability Name	CWE-No
1.	Remote Code Execution	CWE-94
2.	XML External Entity	CWE-611
3.	Cross-Site Scripting	CWE-79
4.	LDAP Injection	CWE-90
5.	SNMP Exploits	CWE-287

Vulnerability assessment Report

Vulnerability Name: Remote Code Execution (RCE)

- **CWE Number**: CWE-94 (Improper Control of Code Generation)
- **OWASP/SANS Category**:
    - OWASP: A03:2021 (Injection)
    - SANS Top 25: Improper Neutralization of Special Elements Used in an OS Command
- **Description**: RCE vulnerabilities allow attackers to execute arbitrary code on a target system. This typically happens due to improper input validation, deserialization issues, or command injection vulnerabilities.
- **Business Impact**:
    - Complete system takeover
    - Data breaches and loss of sensitive information
    - Financial loss due to ransomware deployment
    - Regulatory and legal consequences (GDPR, HIPAA fines)

Vulnerability Name: XML External Entity (XXE) Injection

- **CWE Number**: CWE-611 (Improper Restriction of XML External Entity Reference)
- **OWASP/SANS Category**:
    - OWASP: A04:2021 (Insecure Design)
    - SANS Top 25: Improper Input Validation
- **Description**: XXE occurs when an application processes XML input that contains references to external entities. An attacker can use this to read local files, cause Denial of Service (DoS), or even achieve SSRF (Server-Side Request Forgery).
- **Business Impact**:
    - Exposure of sensitive files (e.g., /etc/passwd)
    - Server-side request forgery (SSRF) leading to data leaks
    - Application denial of service (DoS)
    - Potential remote code execution in some cases

Vulnerability Name:- Cross-Site Scripting (XSS)

- **CWE No:** CWE-79
- **OWASP/SANS Category:** Top 5
- **Description:** Cross-Site Scripting (XSS) is a critical web vulnerability where an attacker injects malicious JavaScript into a website, which is then executed in a victim's browser. This happens when a web application fails to properly validate or sanitize user input before displaying it. XSS attacks can be classified into Stored XSS, where the malicious script is permanently stored on the website and executes when a user visits the affected page; Reflected XSS, where the script is embedded in a malicious link and runs when a victim clicks it; and DOM-based XSS, which occurs due to insecure Javascript execution on the client side. The impact of XSS can be severe, allowing attackers to steal cookies, session tokens, and login credentials, potentially leading to account hijacking and phishing attacks. Additionally, it can be used to inject fake content, deface websites, or spread malware

Business Impact: -

- Attackers can steal user credentials, session cookies, or authentication tokens through malicious scripts.
- XSS can be used to manipulate forms, redirect payments, or steal financial details.
- In e-commerce or banking platforms, it can lead to direct financial losses for both businesses and customers.
- XSS attacks that leak sensitive information can result in heavy fines and legal action.
- XSS can be leveraged to create fake login pages, tricking users into entering Their credentials on a malicious site.
- They may use persistent XSS to create backdoors, leading to long-term security risks.

Vulnerability Name: LDAP Injection

- **CWE Number:** CWE-90 (Improper Neutralization of Special Elements in LDAP Query)
- **OWASP/SANS Category:**
  - OWASP: A03:2021 (Injection)
  - SANS Top 25: Improper Neutralization of Special Elements in Data Queries
- **Description:** LDAP injection occurs when an attacker manipulates input fields to alter LDAP queries. This can lead to unauthorized access, privilege escalation, and information disclosure.
- **Business Impact:**
  - Unauthorized access to sensitive data
  - Bypassing authentication mechanisms
  - Privilege escalation in directory services (e.g., Active Directory)
  - Exposure of employee/customer PII

    SNMP Exploits

Vulnerability Name: SNMP Exploitation (Weak Community Strings & misconfigurations)

- **CWE Number:** CWE-287 (Improper Authentication)
- **OWASP/SANS Category:**
  - OWASP: Not directly listed but falls under A05:2021 (Security Misconfiguration)
  - SANS Top 25: Use of Hard-coded Credentials
- **Description:** SNMP (Simple Network Management Protocol) vulnerabilities arise due to default or weak community strings (like "public"), misconfigured SNMP services, and buffer overflow vulnerabilities in SNMP agents. Attackers can gain unauthorized access to network devices, leak system information, or execute arbitrary code.
- **Business Impact:**
  - Unauthorized access to network configurations
  - Potential network takeover by modifying router settings
  - Information disclosure leading to further attacks
  - Denial of service (DoS) by flooding SNMP services

STAGE-2

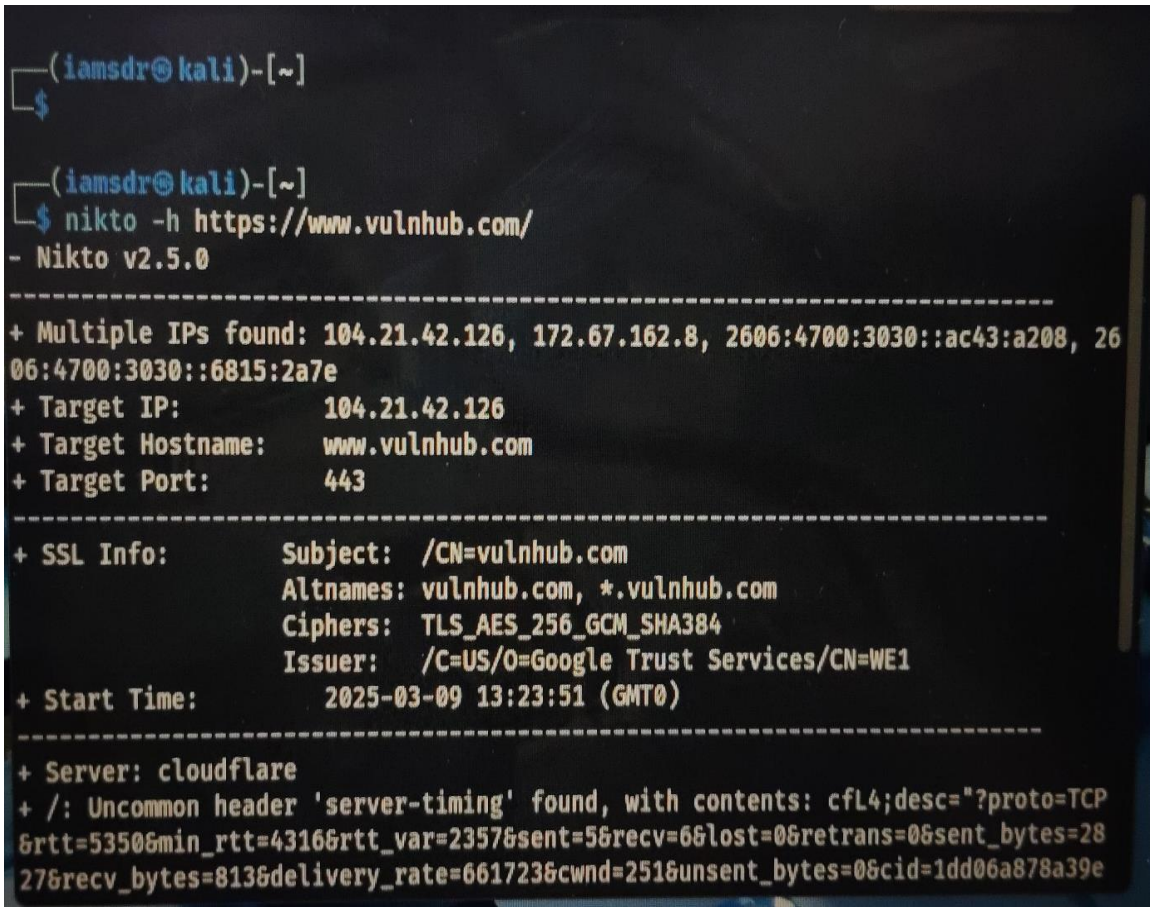
FUNCTIONAL AND PERFORMANCE TESTING

vulnerability report

Target Website: https://www.vulnhub.com/

Target IP address: 104.21.42.126

Target port:443



s.no	Vulnerability name	CWE.no	Severity	Status
1	SQL injection	89	High	confirmed
2	Command injection	77	Medium	confirmed
3	Insecure Deserialization	502	High	confirmed

PROCEDURE FOR FINDING THE VULNERABILITY

Step 1: Set Up the Environment

Install vulnhub

• If not already installed, download vulnhub from https://www.vulnhub.com/.

• Install it on:

o Virtual Machine (e.g., Kali Linux, Ubuntu)

Install Burp Suite

• Download Burp Suite (Community/Professional) from https://portswigger.net/burp.

• Open Burp Suite and set up the proxy.

Step 2: Configure Burp Suite

Set Up Proxy

• Open Burp Suite → Go to Proxy → Options.

• Ensure Burp is listening on 127**.0.0.1:8080.**

• In your browser:

o Set proxy to 127.0.0.1 and port 8080.

o Install Burp CA Certificate to avoid SSL/TLS warnings.

Enable Interception (Optional)

• Go to Proxy → Intercept → Click Intercept is on.

• This allows capturing live requests.

Step 3: Capture and Analyze Requests

Navigate bWAPP

• Open http://localhost/bWAPP/

• Login using default credentials:

Username: bee

Password: bug

• Browse different vulnerable pages to identify entry points.

3.2 Monitor Requests in Burp

• Open Burp Suite → Proxy → HTTP history.

• Identify URLs with GET/POST parameters.

• These parameters can be manipulated for testing.

Step 4: Finding Vulnerabilities

Now, let's test for different vulnerabilities.

SQL Injection (SQLi)

Objective: Inject malicious SQL queries to bypass authentication or extract data.

Identify Vulnerable Input Fields

• Go to vulnhub → Choose "SQL Injection (GET/POST/Search)" from the security level dropdown.

• Submit a normal query like test and capture the request in Burp.

Command injection

Objective: Inject malicious command queries to bypass authentication or extract data.

Identify Vulnerable Input Fields

• Go to bWAPP → Choose "command Injection (GET/POST/Search)" from the security level dropdown.

• Submit a normal query like test and capture the request in Burp.

step 5: Automating with Burp Suite Scanner

• (Professional version required) Go to Target → Issues.

• Run Burp scanner to detect vulnerabilities automatically.

Step 6: Document and Report Findings

For each vulnerability found:

• Description (SQLi, XSS, CSRF, etc.).

• Affected URL & Parameter.

• Proof of Concept (PoC) Payload.

• Impact & Remediation Recommendations.

RESULTS

Findings and Reports

1) SQL Injection (SQLi)

- **CWE No**: CWE-89
- **OWASP/SANS Category**: A03:2021 (Injection), SANS Top 25 (Improper Input Handling)

Description
SQL Injection allows attackers to manipulate database queries by injecting malicious SQL code through unsanitized user input.

Business Impact

- Unauthorized Data Access
- Loss of Data Integrity
- Data Breaches (fines, reputation damage)
- Legal/Compliance Violations (GDPR, PCI-DSS, HIPAA)
- Full Database Compromise

Real-World Example

- *2019*: A US government agency's site was breached via SQLi, leaking sensitive citizen records.

Steps to Identify

1. Insert SQL meta-characters (e.g., ' OR 1=1 --) and observe responses.
2. Use automated tools like **SQLMap**.
3. Analyze application logs for DB error messages.
4. Test for Blind SQL Injection (time-based or Boolean-based).
5. Inspect source code for unvalidated query inputs.

---

2) Command Injection

- **CWE No**: CWE-77
- **OWASP/SANS Category**: A03:2021 (Injection), SANS Top 25 (OS Command Injection)

Description
Command Injection occurs when user input is executed as a system-level command without proper validation, enabling attackers to run arbitrary commands on the host.

Business Impact

- Remote Code Execution (RCE)
- Data Manipulation or Deletion
- Privilege Escalation
- Business Disruption (service downtime)
- Sensitive Information Disclosure (system files, environment variables)

Real-World Example

- *2021*: A cloud hosting provider was breached via a command injection flaw, giving attackers root access.

Steps to Identify

1. Input fuzzing with special characters (;, &&, |).
2. Look for unexpected output or system responses.
3. Use **Burp Suite** or **OWASP ZAP** for scanning.
4. Check system logs for suspicious commands.
5. Review source code for improper command executions.

---

3) Insecure Deserialization

- **CWE No**: CWE-502
- **OWASP/SANS Category**: A08:2021 (Insecure Deserialization), SANS Top 25 (Deserialization Issues)

Description
Insecure Deserialization happens when untrusted data is deserialized, allowing attackers to modify objects and potentially execute arbitrary code or escalate privileges.

Business Impact

- Unauthorized Code Execution
- Data Corruption
- Full System Compromise
- Privilege Escalation

- Denial of Service (via malformed data)

Real-World Example

- *2018*: An enterprise Java app had an insecure deserialization flaw used to install cryptocurrency miners on servers.

Steps to Identify

1. Locate endpoints that deserialize user-supplied data.
2. Decode and analyze serialized objects.
3. Check logs for serialization-related errors.
4. Use tools like **Burp Suite**, **OWASP ZAP**, or custom fuzzers.
5. Modify serialized objects with malicious payloads.
6. Review code handling deserialization (e.g., unserialize() in PHP.

Title: Understanding Cyber Threats: Exploring Nessus Beyond the Scanning Tools