Experiment No. 10

Aim: Generating a penetration testing report with CVSS score and suggested mitigations.

Theory:

This report provides a comprehensive overview of the vulnerabilities identified during the penetration test of the target web application and infrastructure. The assessment focused on real-world attack scenarios, including but not limited to SQL Injection, Local File Inclusion (LFI), Cross-Site Scripting (XSS), weak session management, and outdated software. Each vulnerability is evaluated using the Common Vulnerability Scoring System (CVSS) to enable informed remediation prioritization. Robust mitigation strategies are provided for all findings.

Testing Methodology

1. Reconnaissance and Enumeration

- Identified public-facing assets, endpoints, and input vectors using automated scanners and manual exploration.
- Gathered intelligence on running software versions, exposed services, and potentially sensitive information leaks.

2. Vulnerability Assessment

- Combined automated scanning tools with in-depth manual testing to uncover vulnerabilities in input handling, authentication, authorization, and configuration.

3. Proof-of-Concept (PoC) Exploitation

- Where safe and authorized, demonstrated practical exploitability, focusing on risk exposure and potential business impact.

4. Reporting and Scoring

- Documented validated findings with evidence, PoC steps, and impact analysis.
- Assigned CVSS v3.1 scores considering attack complexity, impact, exploitability, and environmental context.

Findings Summary

Vulnerability	CVSS Score	Severity	Description	Suggested Mitigation
SQL Injection	8.2	High	Allows arbitrary SQL execution	Parameterized queries, input validation, least privilege users
Local File Inclusion	7.5	High	Unauthorized sensitive file access	Input sanitization, allowlist inclusion, strict server permissions
Cross-Site Scripting	6.1	Medium	Client-side script injection	Output encoding, input sanitization, CSP headers
Weak Session Management	5.6	Medium	Predictable or insecure sessions	Secure/HttpOnly/SameSite flags, token regen, short timeouts
Outdated Server Software	4	Low	Unpatched components, known exploits	Routine updates, security advisories, minimize attack surface

Severity Levels:

- Low (0.0–3.9)
- Medium (4.0–6.9)
- High (7.0–8.9)
- Critical (9.0–10.0)

Detailed Vulnerability Descriptions and Remediation

1. SQL Injection

- **Description:**
  Application input fields are insufficiently validated, letting attackers manipulate backend SQL logic. Exploitation may expose sensitive databases, allow credential extraction, authentication bypass, or even server compromise.
- **Proof-of-Concept:**
  Successful injection of operators or union statements in URL/query/post variables led to unauthorized data access.
- **CVSS Details:**
  - **Attack Vector:** Network
  - **Privileges Required:** None
  - **User Interaction:** None
  - **Impact:** High on confidentiality, integrity, and availability
- **Mitigation Steps:**
  - Enforce prepared statements and parameterized queries throughout the codebase.
  - Filter and validate all user-supplied data.
  - Restrict database account permissions to only essential operations.
  - Periodically audit database logs and anomalous queries.

2. Local File Inclusion (LFI)

- **Description:**
  User-controlled path parameters enable attackers to access internal files such as /etc/passwd or sensitive application configs. In applications with file upload features, LFI may be chained for code execution.
- **Proof-of-Concept:**
  Manipulated file path parameters resulted in the disclosure of the server's environment files.
- **CVSS Details:**
  - **Attack Vector:** Network
  - **Privileges Required:** None
  - **Impact:** High on confidentiality; possible integrity/availability risks
- **Mitigation Steps:**
  - Strictly validate and sanitize input used in file operations.
  - Allow only specific, hard-coded file paths and types.
  - Implement server filesystem permissions minimizing access outside intended directories.
  - Disable unnecessary PHP functions (allow_url_fopen, allow_url_include).

3. Cross-Site Scripting (XSS)

- **Description:**
  The application echoes unsanitized and unencoded user data in HTML responses, which allows attacker-crafted JavaScript to execute in user browsers. This could result in session hijack, user impersonation, or malicious redirects.
- **Proof-of-Concept:**
  Injection of harmless <script>alert(1)</script> tags triggered code execution in victim browser sessions.
- **CVSS Details:**
  - **Attack Vector:** Network
  - **Privileges Required:** None
  - **Impact:** Moderately high on confidentiality and integrity
- **Mitigation Steps:**
  - Filter and encode all user input before rendering output.
  - Employ strict Content Security Policy (CSP) headers.
  - Utilize security libraries/frameworks with in-built XSS protections.
  - Update legacy code to ensure state-of-the-art input/output handling.

4. Weak Session Management

- **Description:**
  Session tokens are easily guessable or not properly invalidated on logout, increasing the risk of session hijacking and unauthorized access.
- **Proof-of-Concept:**
  Analysis revealed static or repetitive session patterns without necessary security flags.
- **CVSS Details:**
  - **Attack Vector:** Network
  - **Privileges Required:** None
  - **Impact:** Medium on confidentiality and integrity
- **Mitigation Steps:**
  - Generate session IDs using cryptographically-secure methods.
  - Always use Secure, HttpOnly, and SameSite cookie attributes.
  - Automatically expire sessions after specified inactivity periods or logouts.
  - Rotate session tokens on privilege changes or authentication events.

5. Outdated Server Software

- **Description:**
  Multiple server components are running outdated versions with publicly known vulnerabilities, increasing systemic risk of compromise.
- **Proof-of-Concept:**
  Banner grabbing and fingerprinting tools revealed unpatched versions of open-source components.
- **CVSS Details:**
  - **Attack Vector:** Network
  - **Privileges Required:** None
  - **Impact:** Low to moderate depending on exploitability
- **Mitigation Steps:**
  - Routinely update and patch all OS and software components as per vendor recommendations.
  - Regularly monitor security advisories for critical vulnerabilities.
  - Disable or remove unnecessary services/software from running in production.

CVSS Scoring and Prioritization

CVSS assigns a quantitative score (0–10) for each vulnerability based on its exploitation complexity, the attacker's required privileges, user interaction necessities, and the impact on data confidentiality, integrity, and availability. Scores enable rapid triage and remediation, ensuring that "high" and "critical" severity flaws are addressed most urgently to prevent exploitation.

Recommendations and Next Steps

- Urgently remediate all High and Critical vulnerabilities.
- Deploy comprehensive monitoring for anomalous activities related to the findings (e.g., SQL injection patterns, LFI exploitation attempts).
- Upon completing fixes, conduct a follow-up penetration test ("retest") to confirm closure and check for regression vulnerabilities.
- Foster regular security training and robust code review practices for development and operations teams.
- Establish a routine for timely patch management and continuous vulnerability scanning.

Conclusion:

……

……

……

……

Name:

Class: BE-CSE

Roll No.:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				