

ELEVATE LABS (CYBER SECURITY

INTERNSHIP)

Task 8: SQL Injection Practical Exploitation

1. Objective

The objective of this document is to analyze a web application for potential **SQL Injection vulnerabilities**, assess the possible impact, and recommend appropriate mitigation techniques. All activities are assumed to be performed in an **authorized and controlled environment**.

2. Identification of Injectable Parameters

The first step involves identifying user-supplied inputs that interact with the backend database.

Common input points include:

- URL query parameters
- Login and registration forms
- Search fields
- Cookies and HTTP headers

Indicators of vulnerability:

- Absence of input validation
- Use of dynamic SQL queries
- Verbose database error messages

3. Vulnerability Validation (Tool-Assisted Testing)

Automated security testing tools are used to validate suspected SQL injection points.

Purpose:

- Confirm the presence of SQL injection vulnerabilities
- Reduce false positives

Precautions:

- Testing is restricted to approved scope
- Request rate is controlled to avoid service disruption
- All activities are logged

4. Database Enumeration Risk

If a vulnerability exists, an attacker may be able to identify backend database information.

Potential exposure includes:

- Database type (MySQL, MSSQL, Oracle, etc.)
- Names of available databases

This step helps determine the **severity level** of the vulnerability.

5. Table and Schema Exposure Analysis

Further analysis determines whether database structure can be inferred.

Risk factors:

- Enumeration of table names
- Identification of schema structure
- Presence of sensitive tables such as user or payment records

6. User Data Exposure Assessment

This phase evaluates the potential exposure of sensitive information.

Possible impacted data:

- Usernames and passwords
- Personally Identifiable Information (PII)
- Financial or confidential records

The sensitivity of the data directly affects the **business and legal impact**.

7. Impact Analysis

The impact is analyzed based on the **CIA Triad**:

- **Confidentiality:** Unauthorized access to sensitive data
- **Integrity:** Modification or deletion of records
- **Availability:** Database crashes or service downtime

Additional impacts:

- Reputational damage
- Regulatory non-compliance
- Financial loss

8. Attack Flow Documentation (Conceptual)

A high-level attack flow is documented without revealing exploit details.

Attack flow includes:

1. Attacker identifies a vulnerable input parameter
2. Malicious input is injected into an SQL query
3. Database executes the unintended query
4. Sensitive data or system behavior is exposed

9. Recommended Fixes and Mitigations

9.1 Secure Coding Practices

- Use **prepared statements / parameterized queries**
- Avoid dynamic SQL query construction
- Validate and sanitize all user inputs

9.2 Database Security

- Apply **least privilege** to database users
- Disable detailed error messages in production

9.3 Application & Network Security

- Deploy a **Web Application Firewall (WAF)**
- Enable logging and monitoring for suspicious queries
- Conduct regular security audits

10. Conclusion

SQL Injection remains a critical web application vulnerability. Proper secure coding practices, layered security controls, and regular assessments significantly reduce the risk of exploitation. Addressing these vulnerabilities protects user data, maintains system integrity, and ensures regulatory compliance.