Task 8: SQL Injection Practical Exploitation

Tools: SQLMap Alternatives: Manual testing

Hints / Mini Guide:

1.Identify injectable parameters.

2.Run SQLMap.

3.Extract database names.

4.Extract tables.

5.Extract user data.

6.Analyze impact.

7.Document attack flow.

8.Suggest fixes

1. Identify Injectable Parameters

Injectable parameters are input fields where user-supplied data is passed to backend SQL queries without proper validation or sanitization. These parameters may exist in URL query strings, form inputs, cookies, or request headers. Identifying such parameters is essential because they represent potential entry points through which an attacker could manipulate database queries.

2. Run SQLMap

SQLMap is used to automate the detection of SQL injection vulnerabilities after a potentially injectable parameter is identified. It sends structured SQL payloads to the application and analyzes responses to confirm whether the input is insecurely handled by the database. This step helps validate findings and reduces false positives identified during manual testing.

3. Extract Database Names

Extracting database names helps identify the backend databases used by the application. This information provides insight into the system architecture and the scope of exposure. Knowledge of database names demonstrates how an attacker could gain awareness of sensitive backend components if SQL injection vulnerabilities are present.

4. Extract Tables

Table extraction reveals how data is structured within a database. Tables often contain sensitive information such as user credentials, transactions, or configuration data. Identifying accessible tables highlights the severity of the vulnerability and shows how improper access controls can expose internal application data.

5. Extract User Data

Extracting user-related data illustrates the real-world impact of SQL injection vulnerabilities. It demonstrates how attackers could retrieve sensitive information such as usernames, email addresses,

or password hashes. This step is typically performed only in authorized testing environments to assess the potential risk to user privacy and data confidentiality.

6. Analyze Impact

Impact analysis evaluates the consequences of the identified vulnerability on confidentiality, integrity, and availability. Potential impacts include data breaches, unauthorized access, account compromise, and regulatory non-compliance. Understanding impact helps prioritize remediation based on business and security risk.

7. Document Attack Flow

Documenting the attack flow involves recording each phase of the testing process, including the vulnerable parameter, testing method, observed behavior, and results. Clear documentation ensures that vulnerabilities can be reproduced, understood, and addressed by development teams and stakeholders.

8. Suggest Fixes and Mitigations

Suggesting fixes focuses on preventing SQL injection vulnerabilities through secure development practices. Common mitigations include using parameterized queries, input validation, least-privilege database access, secure error handling, and regular security testing. Providing mitigation strategies strengthens the application’s overall security posture.