Secure Programming
Injection

Dr. Fatma ElSayed

Computer Science Department
fatma.elsayed@fci.bu.edu.eg

Introduction to Injection

Injection: is a type of security vulnerability that occurs when untrusted data is inserted into a program or query, allowing attackers to insert malicious code or commands.

Common Types

- SQL Injection
- Cross-site Scripting (XSS)
- Command Injection
- Local File Inclusion

SQL Injection (SQLI)

- Injects malicious SQL code to manipulate database queries.
- This vulnerability arises when a web application fails to validate user input correctly.

Steps of an SQL Injection Attack:

- Identifying Vulnerable Points: Forms, URL parameters, cookies, or headers.
- Crafting Malicious Input: It could bypass authentication by tricking the SQL query into thinking the condition is always true.
- Executing Malicious Queries: Application processes the input, leading to unintended behavior.

Types of SQL Injection

Union-Based SQL Injection

Combines results of two or more queries.

Error-Based SQL Injection

Uses database error messages to extract data.

Blind SQL Injection

Relies on observing application behavior.

Time-Based Blind SQL Injection

Uses time delays to infer information.

Preventing SQL Injection

- **Parameterized Queries:** Prepared statements that separate user inputs from SQL commands.
- **Input Validation:** Ensure all data provided by users meets expected formats and reject unexpected or dangerous characters
- **Database Access Controls:** Limit permissions for queries.
- **Using ORM Tools:** Like Hibernate or Django ORM.
- **Security Testing:** Regular scans for vulnerabilities.

Command Injection

- Command Injection is a security vulnerability that occurs when an attacker manipulates input to execute arbitrary commands on the operating system or server or application.

Types of command Injection

1. Blind OS Command Injection
2. Classic (or Non-Blind) OS Command Injection
3. Out-of-Band (OOB) Command Injection

Preventing Command Injection

- **Avoid Calling the Shell:**
  - Use language-native functions instead of executing shell commands.
- **Whitelist Valid Input**
  - Only allow expected and valid input values.
- **Use Safe APIs**
  - Use APIs that don't involve shell interpretation.
- **Least Privilege**
  - Run apps with limited permissions.
- **Use a Web Application Firewall**
  - Detects and blocks known attack patterns.

Tools For Detection

- **Static Analysis**
  - Analyzes source code without executing it *(SonarQube, Checkmarx)*
- **Dynamic Testing**
  - Tests the application during execution, simulating user behavior *(OWASP ZAP, Burp Suite)*
- **Penetration Testing**
  - Manual or automated simulated attacks on a system to discover real-world vulnerabilities.

THANK YOU