APPSEC FUNDAMENTALS FOR MODERN DEVOPS

LISA 2021

Suchakra Sharma
Staff Scientist
ShiftLeft

Vickie Li
Developer Evangelist
ShiftLeft
WHAT YOU'LL LEARN IN THIS SESSION
OWASP TOP TEN: WHAT COULD GO WRONG?

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access Control
- Security Misconfiguration
- Cross-Site Scripting (XSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging & Monitoring
SQL INJECTION

#631051

SQL Injection Extracts Starbucks Enterprise Accounting, Financial, Payroll Database

State: Resolved (Closed)
Disclosed: August 6, 2019 12:51am -0500
Reported to: Starbucks
Reported at: April 8, 2019 5:38am -0500
Asset: Other non domain specific items
CVE ID
Weakness: SQL Injection

Severity: Critical (9.3)
Participants: 
Visibility: Disclosed (Limited)

Starbucks SQL injection: https://hackerone.com/reports/531051
THE SDLC: SOFTWARE DEVELOPMENT LIFE CYCLE

- Requirements
- Design
- Code
- Testing
- Release
THE SDLC: SOFTWARE DEVELOPMENT LIFE CYCLE

- Requirements
- Design
- Code
- Testing
- Release

Shift left
THE SDLC: SOFTWARE DEVELOPMENT LIFE CYCLE

- Requirements
- Design
- Code
- Testing
- Release

Shift left
THE SDLC: SOFTWARE DEVELOPMENT LIFE CYCLE

- Requirements
- Design
- Code
- Testing
- Release

Shift left
SHIFT LEFT: REQUIREMENTS

- Ask security questions from the very start.
- Include security folks in requirement planning.
SHIFT LEFT: DESIGN

- Plan application design around security requirements.
- Consider building in security mechanisms like input validation, output encoding, and prepared statements from the start.
• Choose a secure programming language and framework.
• Handle untrusted data safely via validation, sanitization, and output encoding.
• Implementing proper error handling and logging.
SHIFT LEFT: TESTING

- Manual code review
- SAST (Static Analysis Security Testing)
- SCA (Software Composition Analysis)
- DAST (Dynamic Analysis Security Testing)
- Pentests + bug bounty programs
SHIFT LEFT: RELEASE

- Pay attention to the security of your CICD pipeline.
- Build security tests into the pipeline, such as dependency monitoring and SAST scans.
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```java
String queryString = 
    "SELECT * FROM USER WHERE
    USERNAME = '' + Username + ''
    AND PASSWORD = '' + Password + ''";

sql.executeQuery(QueryString)
```
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```java
String queryString = 
    "SELECT * FROM USER WHERE 
    USERNAME = '' + Username + '' 
    AND PASSWORD = '' + Password + ''";

sql.executeQuery(QueryString)
```

**HTTP request:**

```plaintext
POST /login

Username=Vickie
Password=password123
```
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```
String queryString = "SELECT * FROM USER WHERE USERNAME = '" + Username + "' AND PASSWORD = '" + Password + "'";

sql.executeQuery(queryString)
```

HTTP request:

POST /login

Username=Vickie
Password=password123
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```
SELECT Id FROM Users
WHERE Username='vickie' AND Password='password123';
```
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```java
String queryString = "SELECT * FROM USER WHERE USERNAME = '" + Username + "' AND PASSWORD = '" + Password + "";

sql.executeQuery(queryString)
```

HTTP request:
```
POST /login
Username=admin';--
Password=password123
```
SQL INJECTION

A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```sql
SELECT Id FROM Users
    WHERE Username='admin';-- ' AND Password='password123';
```
A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

```sql
SELECT Id FROM Users
  WHERE Username='admin';-- ' AND Password='password123';
```
PREVENTING SQL INJECTION

• A SQL Injection attack is when an attacker can inject arbitrary SQL code into SQL statements that an application uses to access its database.

SELECT Id FROM Users
    WHERE Username='admin';-- ' AND Password='password123';
PREVENTING SQL INJECTION

- How will sensitive data be stored and transported?
- When does this app need to take in user input?
- Where does this app make database calls?
- Are user input needed in database calls?

Requirements  Design  Code  Testing  Release
PREVENTING SQL INJECTION

- What mechanisms should we use to handle user input safely?
- Where are input validation, sanitization, and escaping needed?
- How do we secure database calls?
- How do we store sensitive data safely to minimize damage in case of a breach?
- What is the least privilege needed for the application to run?
- How do we backup data and code?
- How should we log potential attacks and errors?
PREVENTING SQL INJECTION

- Implement input validation.
- Escape or reject dangerous characters.
- Implement prepared statements.
- Implement the principle of least privilege.
- Store data securely.
PREVENTING SQL INJECTION

- Manual code review of dangerous functions.
- SAST scanning for signatures of SQL injection.
- SCA to ensure third-party components are secure.
PREVENTING SQL INJECTION

- Build security tests into the pipeline, such as SCA and SAST scans.
- Bug bounty programs.
- Regularly back up important data and code.
- Monitor the application for potential attacks.
THANK YOU!

Feel free to connect:
@vickieli7
@tuxology