# SQL Injection (SQLi)

## Introduction

This repository focuses on common web application vulnerabilities, including parameter tampering, Cross-Site Scripting (XSS), and SQL Injection. This section provides an overview of SQL Injection, examples of how it can be exploited, and guidance on how to prevent it.

## SQL Injection

### Explanation

SQL Injection (SQLi) is a type of web application attack where an attacker manipulates SQL queries to execute arbitrary SQL code. This can lead to unauthorized access, data leakage, and other security issues.

### Common Techniques

- Error-based SQL Injection
- Union-based SQL Injection
- Blind SQL Injection

### Example Scenarios

1. **Error-based SQL Injection**
   - Original Query: `SELECT * FROM users WHERE username = 'admin' AND password = 'password'`
   - Tampered Query: `SELECT * FROM users WHERE username = 'admin' AND password = 'password' OR '1'='1'`
   - Impact: An attacker can bypass authentication by injecting a condition that always evaluates to true.
2. **Union-based SQL Injection**
   - Original Query: `SELECT name, email FROM users WHERE id = 1`
   - Tampered Query: `SELECT name, email FROM users WHERE id = 1 UNION SELECT credit_card_number, cvv FROM credit_cards`
   - Impact: An attacker can retrieve sensitive information from another table by using the UNION operator.
3. **Blind SQL Injection**
   - Original Query: `SELECT * FROM users WHERE username = 'admin' AND password = 'password'`
   - Tampered Query: `SELECT * FROM users WHERE username = 'admin' AND password = 'password' AND 1=IF(1=1, SLEEP(5), 0)`
   - Impact: An attacker can infer information from the database by observing the response time.

### Code Snippets

1. **Error-based SQL Injection**

```
# Original Query
query = "SELECT * FROM users WHERE username = 'admin' AND password = 'password'"

# Tampered Query
tampered_query = query + " OR '1'='1'"
```

2. **Union-based SQL Injection**

```
# Original Query
query = "SELECT name, email FROM users WHERE id = 1"

# Tampered Query
tampered_query = query + " UNION SELECT credit_card_number, cvv FROM credit_cards"
```

3. **Blind SQL Injection**

```
# Original Query
query = "SELECT * FROM users WHERE username = 'admin' AND password = 'password'"

# Tampered Query
tampered_query = query + " AND 1=IF(1=1, SLEEP(5), 0)"
```

### Prevention Techniques

- Use prepared statements and parameterized queries to prevent SQL injection.
- Validate and sanitize all user inputs on the server side.
- Implement least privilege principle for database access.
- Use ORM (Object-Relational Mapping) frameworks that provide built-in protection against SQL injection.

## Conclusion

SQL Injection (SQLi) is a serious security threat that can lead to unauthorized access and data manipulation. By understanding common techniques and implementing prevention measures, developers can protect their applications from such attacks.

For further reading, refer to the following resources: - OWASP: SQL Injection - OWASP: SQL Injection Prevention Cheat Sheet