ЛАБОРАТОРНА РОБОТА 6

Етичний хакінг власного застосунку

Виконав Холоша Сергій

Дослідження SQL-ін’єкцій

LAB | APPRENTICE | SQL injection vulnerability in WHERE clause allowing retrieval of hidden data → | ✓ Solved

LAB | APPRENTICE | SQL injection vulnerability allowing login bypass → | ✓ Solved

LAB | PRACTITIONER | SQL injection attack, querying the database type and version on Oracle → | ✓ Solved

LAB | PRACTITIONER | SQL injection attack, querying the database type and version on MySQL and Microsoft → | ✓ Solved

LAB | PRACTITIONER | SQL injection attack, listing the database contents on non-Oracle databases → | ✓ Solved

- 5 лабораторна робота була найскладнішою в виконанні та об’єднувала усі що були раніше. В цій лабораторній роботі я досліджував вразливість веб-додатку до SQL-ін'єкцій через параметр category. Спочатку мені потрібно було з'ясувати, скільки стовпців повертає SQL-запит, тому я підставив у параметр '+UNION+SELECT+'abc','def'-- і побачив, що на сторінці відобразилися обидва значення, значить запит повертає два текстових стовпці.
- Далі я вирішив подивитися, які взагалі таблиці є в базі даних, використовуючи системну таблицю information_schema.tables. Я передав запит '+UNION+SELECT+table_name,+NULL+FROM+information_schema.tables-- і отримав список усіх таблиць. Серед них я знайшов таблицю з назвою типу users_abcdef, яка, очевидно, містила дані користувачів.
- Потім мені треба було дізнатися структуру цієї таблиці, тобто які в ній стовпці. Я запустив запит '+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_abcdef'--, замінивши users_abcdef на реальну назву таблиці. У відповіді я побачив назви стовпців і знайшов ті, що відповідали за імена користувачів і паролі, на кшталт username_abcdef та password_abcdef.
- Маючи всю цю інформацію, я зробив фінальний запит '+UNION+SELECT+username_abcdef, +password_abcdef+FROM+users_abcdef--, підставивши реальні назви таблиці і стовпців. У результаті отримав список усіх користувачів з їхніми паролями. Знайшов там користувача administrator з його паролем і використав ці дані для входу в систему, чим успішно завершив лабораторну роботу.

Аналіз принципів SQL-ін’єкцій

Уразливий варіант: SELECT firstname FROM users WHERE username='admin' AND password='admin'

Безпечний варіант: SELECT firstname FROM users WHERE username=? AND password=?

```python
import sqlite3

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

cursor.execute("CREATE TABLE users (username TEXT, password TEXT, firstname TEXT)")
cursor.execute("INSERT INTO users VALUES ('admin', 'admin', 'Petro')")
cursor.execute("INSERT INTO users VALUES ('user', 'pass', 'Ivan')")
conn.commit()

username = "admin' OR '1'='1"
password = "anything"

print("Уразливий запит:")
query = f"SELECT firstname FROM users WHERE username='{username}' AND password='{password}'"
print(query)
result = cursor.execute(query).fetchall()
print(f"Результат: {result}\n")

print("Безпечний запит:")
query = "SELECT firstname FROM users WHERE username=? AND password=?"
print(query)
result = cursor.execute(query, (username, password)).fetchall()
print(f"Результат: {result}")

conn.close()
```

```
Уразливий запит:
SELECT firstname FROM users WHERE username='admin' OR '1'='1' AND password='anything'
Результат: [('Petro',)]

Безпечний запит:
SELECT firstname FROM users WHERE username=? AND password=?
Результат: []
```

Уразливий варіант використовує конкатенацію рядків, тому зловмисник може вставити admin' OR '1'='1 замість логіна і отримати доступ до всіх записів, бо умова '1'='1' завжди істинна.

Безпечний варіант використовує плейсхолдери (?), які автоматично екранують спецсимволи, тому введення admin' OR '1'='1' сприймається як звичайний текст, а не як частина SQL-коду.

Тестування на практиці

Розширимо попередній код

```python
import sqlite3

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

cursor.execute("CREATE TABLE users (username TEXT, password TEXT, firstname TEXT)")
cursor.execute("INSERT INTO users VALUES ('admin', 'admin', 'Petro')")
cursor.execute("INSERT INTO users VALUES ('user', 'pass', 'Ivan')")
conn.commit()

username = "admin' OR '1'='1"
password = "anything"

print("Уразливий запит:")
query = f"SELECT firstname FROM users WHERE username='{username}' AND password='{password}'"
print(query)
try:
    result = cursor.execute(query).fetchall()
    print(f"Результат: {result}")
    if result:
        print("УВАГА! SQL-ін'єкція успішна - отримано несанкціонований доступ!")
except sqlite3.OperationalError as e:
    print(f"Помилка SQL: {e}")
print()

print("Безпечний запит:")
query = "SELECT firstname FROM users WHERE username=? AND password=?"
print(query)
try:
    result = cursor.execute(query, (username, password)).fetchall()
    print(f"Результат: {result}")
    if not result:
        print("Доступ заборонено: невірний логін або пароль")
except sqlite3.OperationalError as e:
    print(f"Помилка SQL: {e}")

conn.close()
```

```
Уразливий запит:
SELECT firstname FROM users WHERE username='admin' OR '1'='1' AND password='anything'
Результат: [('Petro',)]
УВАГА! SQL-ін'єкція успішна - отримано несанкціонований доступ!

Безпечний запит:
SELECT firstname FROM users WHERE username=? AND password=?
Результат: []
Доступ заборонено: невірний логін або пароль
```

Тепер програма також обробляє помилки, при запуску ми зможемо побачити яка типова помилка виникає при спробі ін’єкції на коректній версії.

В даному випадку ми бачимо результат «невірний логін або пароль» адже запит обробив ін’єкцію як звичайний текст.

Технічне завдання

```
============================================================
 ДЕМОНСТРАЦІЯ SQL-ІН'ЄКЦІЙ ТА МЕТОДІВ ЗАХИСТУ
============================================================
База даних створена успішно!


РЕЖИМИ РОБОТИ:
1. Автоматична демонстрація (всі сценарії)
2. Інтерактивний режим (ручне тестування)

Виберіть режим (1/2): 1
```

```
======================================================================
 ДЕМОНСТРАЦІЯ 1: Вразлива авторизація
======================================================================

--- Спроба 1: Звичайна авторизація ---

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM users WHERE username = 'admin' AND password = 'admin123'

✓ Вхід успішний!
ID: 1, Користувач: admin, Роль: administrator

--- Спроба 2: SQL-ін'єкція (обхід пароля) ---
Ввід: username = admin'--
Пояснення: Символ ' закриває рядок, -- коментує перевірку пароля

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM users WHERE username = 'admin'--' AND password = 'anything'

✓ Вхід успішний!
ID: 1, Користувач: admin, Роль: administrator

--- Спроба 3: SQL-ін'єкція (універсальний обхід) ---
Ввід: username = ' OR '1'='1'--
Пояснення: Умова завжди істинна, отримуємо першого користувача

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM users WHERE username = '' OR '1'='1'--' AND password = 'anything'

✓ Вхід успішний!
ID: 1, Користувач: admin, Роль: administrator
```

```
======================================================================
 ДЕМОНСТРАЦІЯ 2: Захищена авторізація
======================================================================

--- Спроба 1: Звичайна авторізація ---

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM users WHERE username = ? AND password = ?
[ПАРАМЕТРИ]: username=admin, password=admin123

✓ Вхід успішний!
ID: 1, Користувач: admin, Роль: administrator

--- Спроба 2: Спроба SQL-ін'єкції (ЗАБЛОКОВАНО) ---
Ввід: username = admin'--

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM users WHERE username = ? AND password = ?
[ПАРАМЕТРИ]: username=admin'--, password=anything

✗ Нічого не знайдено

--- Спроба 3: Спроба SQL-ін'єкції (ЗАБЛОКОВАНО) ---
Ввід: username = ' OR '1'='1'--

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM users WHERE username = ? AND password = ?
[ПАРАМЕТРИ]: username=' OR '1'='1'--, password=anything

✗ Нічого не знайдено
```

```
======================================================================
 ДЕМОНСТРАЦІЯ 3: Вразливий пошук студентів
======================================================================

--- Спроба 1: Звичайний пошук ---

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE '%Іван%'

✓ Знайдено 1 студентів:
  • Іван Петренко (Курс: 3, GPA: 4.5)

--- Спроба 2: SQL-ін'єкція (витік всіх даних) ---
Ввід: %' OR '1'='1
Пояснення: Умова завжди істинна, отримуємо всі записи

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE '%%' OR '1'='1%'

✓ Знайдено 5 студентів:
  • Іван Петренко (Курс: 3, GPA: 4.5)
  • Марія Коваленко (Курс: 2, GPA: 4.8)
  • Олег Шевченко (Курс: 4, GPA: 4.2)
  • Анна Бондаренко (Курс: 1, GPA: 4.9)
  • Дмитро Ткаченко (Курс: 3, GPA: 3.8)

--- Спроба 3: SQL-ін'єкція (витік даних з іншої таблиці) ---
Ввід: %' UNION SELECT id, username, password, role, username FROM users--
Пояснення: UNION дозволяє об'єднати результати з таблиці users

[ВРАЗЛИВИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE '%%' UNION SELECT id, username, password, role, username FROM users--%'

✓ КРИТИЧНА УРАЗЛИВІСТЬ! Витік даних користувачів:
  • ID: 1, Логін: admin, Пароль: admin123, Роль: administrator
  • Іван Петренко (Курс: 3, GPA: 4.5)
  • ID: 2, Логін: teacher, Пароль: teach456, Роль: teacher
  • Марія Коваленко (Курс: 2, GPA: 4.8)
  • ID: 3, Логін: student, Пароль: stud789, Роль: student
  • Олег Шевченко (Курс: 4, GPA: 4.2)
  • Анна Бондаренко (Курс: 1, GPA: 4.9)
  • Дмитро Ткаченко (Курс: 3, GPA: 3.8)
```

```
======================================================================
 ДЕМОНСТРАЦІЯ 4: Захищений пошук студентів
======================================================================

--- Спроба 1: Звичайний пошук ---

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE ?
[ПАРАМЕТРИ]: search_term=%Іван%

✓ Знайдено 1 студентів:
  • Іван Петренко (Курс: 3, GPA: 4.5)

--- Спроба 2: Спроба SQL-ін'єкції (ЗАБЛОКОВАНО) ---
Ввід: %' OR '1'='1

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE ?
[ПАРАМЕТРИ]: search_term=%%' OR '1'='1%

✗ Нічого не знайдено

--- Спроба 3: Спроба SQL-ін'єкції UNION (ЗАБЛОКОВАНО) ---
Ввід: %' UNION SELECT id, username, password, role, username FROM users--

[ЗАХИЩЕНИЙ ЗАПИТ]: SELECT * FROM students WHERE name LIKE ?
[ПАРАМЕТРИ]: search_term=%%' UNION SELECT id, username, password, role, username FROM users--%

✗ Нічого не знайдено
```

```
================================================================================
 ПОРІВНЯННЯ: Вразлива vs Захищена версія
================================================================================

📌 ВРАЗЛИВА ВЕРСІЯ:
  ✗ Пряме підставлення користувацького вводу в SQL-запит
  ✗ Можливість виконання довільних SQL-команд
  ✗ Витік конфіденційних даних
  ✗ Можливість обходу авторизації
  ✗ Можливість модифікації/видалення даних

📌 ЗАХИЩЕНА ВЕРСІЯ:
  ✓ Використання параметризованих запитів (prepared statements)
  ✓ Автоматичне екранування спеціальних символів
  ✓ Неможливість виконання довільного SQL-коду
  ✓ Захист від всіх типів SQL-ін'єкцій
  ✓ Збереження цілісності даних

📌 МЕХАНІЗМ ЗАХИСТУ:
  • Prepared statements відокремлюють SQL-код від даних
  • Параметри передаються окремо і обробляються як значення
  • БД знає структуру запиту до підстановки параметрів
  • Спеціальні символи (', --, ;) розглядаються як дані, а не код

================================================================================
 ДЕМОНСТРАЦІЮ ЗАВЕРШЕНО
================================================================================
```

Дякую за увагу!

Висновок: в ході лабораторної роботи я навчився виявляти та усувати уразливості типу SQL-ін’єкція у програмних застосунках через створення та тестування власного вразливого застосунку