**Доступ к БД с уровня приложений. SQL-инъекции. Защита и фильтрация данных, получаемых от пользователя.**

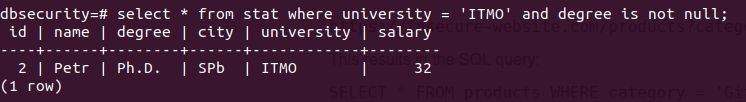
В общем виде SQL-инъекция – это уязвимость на уровне приложений, при которой злоумышленник может создавать запросы, позволяющие извлекать конфиденциальные данные из базы данных. Эта уязвимость основана на ошибке в исходном коде, который обрабатывает пользовательские данные и передает введенные пользователем значения в качестве параметров запроса. В этом случае злоумышленник может манипулировать данными в таблицах базы данных. Функции, используемые злоумышленником во время атаки, основаны на синтаксисе языка SQL. Есть несколько типов SQL-инъекций.

*Получение скрытых данных*

Во время этой атаки злоумышленник может получить доступ к скрытым данным в базе данных. Для этого злоумышленник просто добавляет комментарий (-), который позволяет пропустить следующее за ним условие из SQL-запроса.

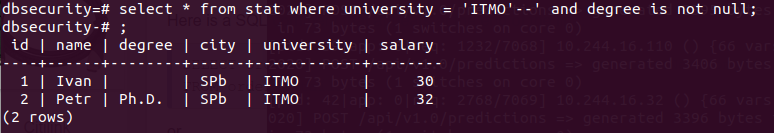
Исходный SQL-запрос показан ниже.

**select \* from stat where university = 'ITMO' and degree is not null;**

****

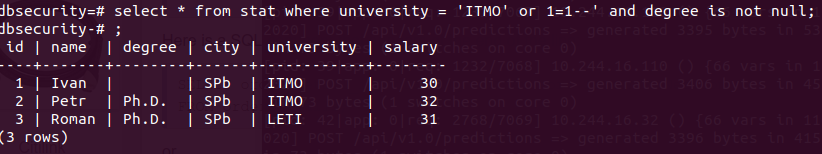
Атакующий может использовать знак комментария для выполнения SQL-инъекции.

**select \* from stat where university = 'ITMO'--' and degree is not null;**

****

Также злоумышленник может использовать всегда верное логическое условие внутри входных данных для того, чтобы получить доступ к пространству скрытых значений.

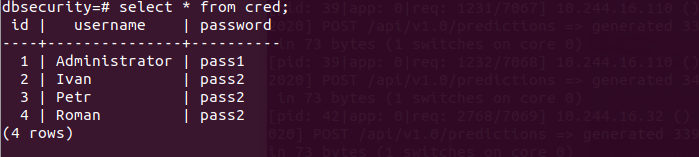
**select \* from stat where university = 'ITMO' or 1=1--' and degree is not null;**



*Изменение логики приложения*

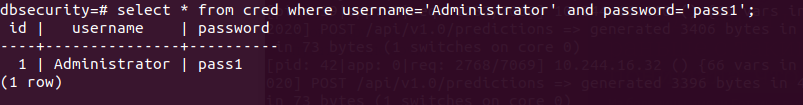
Часто SQL-инъекция используются для изменения логики обработки запросов в приложении. Типичный пример здесь связан с процедурой входа в систему.

**select \* from cred;**

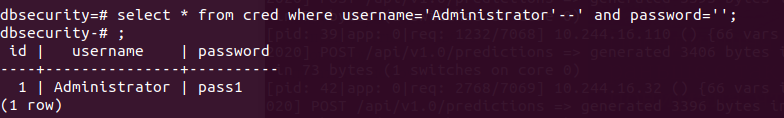
****

Злоумышленник может обойти процедуру аутентификации в системе, которая основана на SQL-запросах, используя уже упомянутые ранее подходы.

**select \* from cred where username='Administrator' and password='pass1';**

****

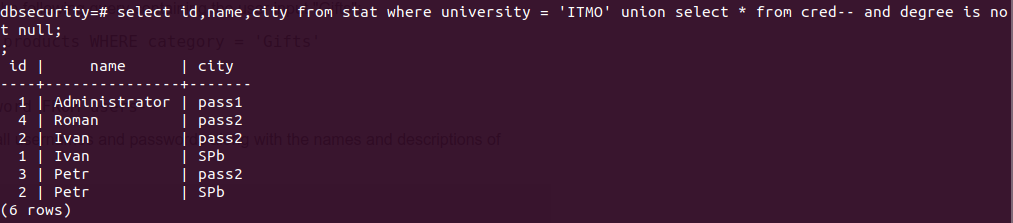
**select \* from cred where username='Administrator'--' and password='';**



*Доступ к другим таблицам в базе данных*

Во время построения SQL-запроса к базе данных злоумышленнику может быть интересна не только текущая таблица БД, поскольку она может и не содержать конфиденциальных данных. Для доступа к другой таблице необходимо использовать ключевое слово UNION.

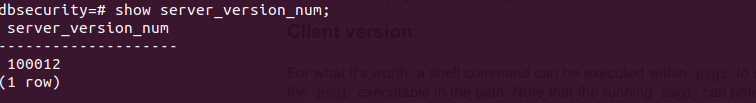
**select id,name,city from stat where university = 'ITMO'' union select \* from cred-- and degree is not null;**



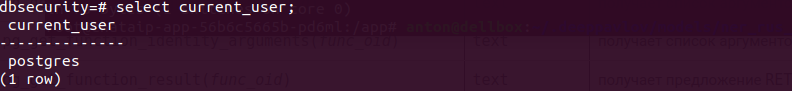
*Анализ базы данных*

Атакующий может проанализировать базу данных, используя встроенные функции.

**show server\_version\_num;**



**select current\_user;**



**Подготовка параметров**

Один из популярных методов предотвращения SQL-инъекций – использование параметризованных запросов. Такой подход позволяет вместо конкатенации строк использовать специальную структуру с входными данными в качестве параметров.

Все известные языки программирования могут использовать подготовленные параметры.

Пример для языка программирования Java вы можете увидеть ниже:

***public static void*** *main(String[] args){*

***if****(args.****length****!= 3){*

*System.****out****.println(****"No enough input values: username password type SQl query "****);*

*System.exit(0);*

*}*

*String username = args[0];*

*System.****out****.println(****"Input username:"****+username);*

*String password = args[1];*

*System.****out****.println(****"Input password:"****+password);*

***int*** *isPrepared = Integer.parseInt(args[2]);*

*System.****out****.println(****"Sqltype :"****+isPrepared);*

*System.****out****.println();*

***try****{*

*Class.forName(****"org.postgresql.Driver"****);*

*String url =* ***"jdbc:postgresql://localhost:5432/dbsecurity"****;*

*Connection conn = DriverManager.getConnection(url,****"postgres"****,****"pass"****);*

*PreparedStatementpstmt;*

*String sql;*

***if*** *(isPrepared == 1){*

*sql =* ***"SELECT id,username,password from cred where password = ? and username = ? "****;*

*pstmt = conn.prepareStatement(sql);*

*pstmt.setString(1,password);*

*pstmt.setString(2,username);*

*}*

***else*** *{*

*sql =* ***"SELECT id,username,password from cred where password ='"*** *+ password+****"'"****+****" and username = '"****+username+****"'"****;*

*pstmt = conn.prepareStatement(sql);*

*}*

*System.****out****.println(****"Executed SQL : "****+sql);*

*ResultSetrs = pstmt.executeQuery();*

*System.****out****.println(****"\nId : Username : Password \n"****);*

***while****(rs.next()){*

*System.****out****.println(rs.getInt(****"id"****)+****" "****+rs.getString(****"username"****)+****" "****+rs.getString(****"password"****));*

*}*

*rs.close();*

*pstmt.close();*

*conn.close();*

*}* ***catch*** *(Exception e){*

*System.****out****.println(e);*

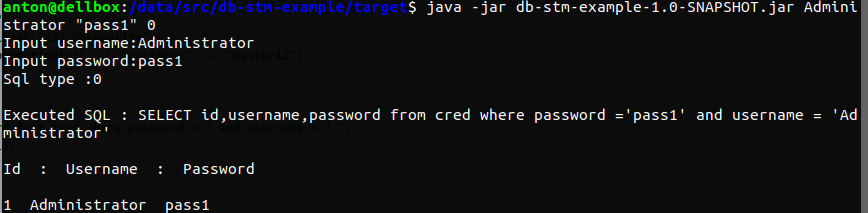
*}*

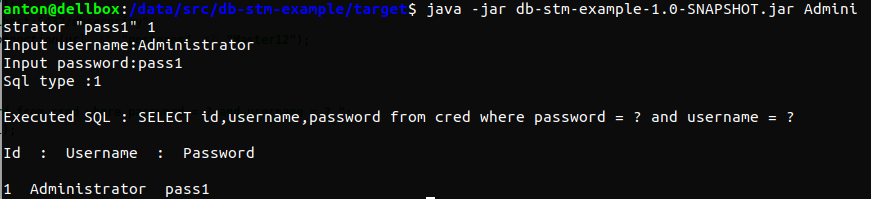
*}*

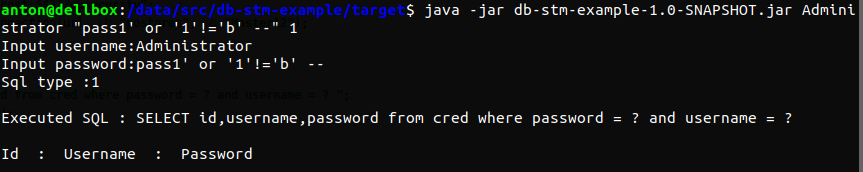
*Пример 1. Для изучения простейших SQL инъекций.*

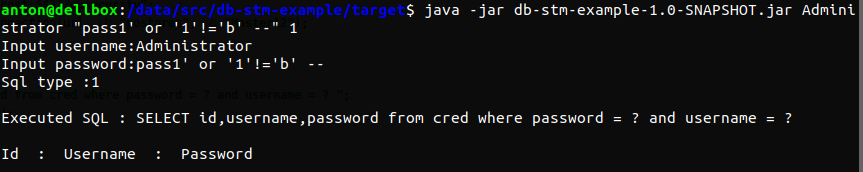
Есть два варианта: использовать подготовленный оператор с обработкой символов-пропусков и просто конкатенацию строк.

Обычное выполнение может заключаться в получении записи из таблицы, которая соответствует условиям пользователя и пароля.









**Задачи для практической работы:**

1. Для данной лабораторной работы создайте БД как минимум из 2-3 отношений и заполните ее данными (достаточно 5-6 кортежей в каждой таблице). Можно использовать отношения из прошлых лабораторных. Отношения должны быть составлены таким образом, чтобы была возможность выполнить объединение таблиц (заданы связи через внешние ключи; имелись атрибуты в таблицах, по которым возможно выполнить объединение вида inner join, left join и др)

2. В рамках ЛР опишите и продемонстрируйте один из способов взаимодействия с БД с уровня приложения. Для изучения можно выбрать любой язык программирования и любой фреймворк/ORM систему/интерфейс для доступа к базе данных. Для демонстрации функций фреймворка/ORM системы/интерфейса для доступа к БД покажите, как минимум, следующие действия с БД: выборка, вставка, удаление данных из вашей БД с помощью выбранного вами фреймворка или языка программирования. Составьте как минимум 2 сложных запроса, в которых выполняется выборка/модификация данных в одних таблицах на основании данных из других таблиц.

3. Для изучения проблемы фильтрации данных подготовьте пример аналогичный, заданному в указаниях к данной лабораторной работе (Пример 1.). Пример может быть подготовлен на любом языке программирования. Предусмотрите в примере два случая подготовки SQL запросов (подготовленные запросы, конкатенация параметров со строкой запроса).

4. Для варианта конкатенации параметров, вводимых пользователем, со строкой запроса продемонстрируйте возможные варианты проведения SQL-инъекций. Например, покажите как в случае объединения таблиц злоумышленник может узнать количество столбцов второй таблицы. Предложите подход для получения структуры базы данных (включая название столбцов таблицы). Покажите устойчивость или уязвимость варианта с подготовленными параметрами к выбранным вами вариантам проведения SQL-инъекций.

**Вопросы к защите:**

1. Опишите разницу между динамическими и статическими SQL выражениями.
2. Опишите хотя бы один из методов анализа исходного кода, который позволяет обнаружить уязвимые места для внедрения SQL инъекции
3. Приведите хотя бы один пример как ' или '1'='1 могут быть использованы для тестирование на возможность внедрения SQL инъекции
4. Приведите хотя бы один пример как UNION может быть использован для внедрения SQL инъекции
5. Требования к базам данных (адекватность предметной области, удобство использования, производительность, защищенность данных)