Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И. Носова»

Кафедра информатики и информационной безопасности

**ЛАБОРАТОРНАЯ РАБОТА №2**

По дисциплине: Безопасность систем баз данных

На тему: Задание 8.2

Исполнитель: Минязев Т.Э, студент 3 курса, группа АИБ-22-1

Руководитель: Григоренко Л.А., преподаватель кафедры ИиИБ

Магнитогорск, 2024

**Цель работы**

Задача 1:

* Перейти на задание: <https://ringzer0ctf.com/challenges/1>
* Используя экранирование запроса(комментирование) получить следующий запрос для отправки логина администратора без пароля

Задача 2:

* Перейти на задание: <https://ringzer0ctf.com/challenges/3>
* Проверить работу уже изученных инъекций.
* Придумать запрос позволяющий обойти защиту от управляющих символов в запросе.
* Объяснить работу вашего запроса.

**Контрольный вопрос:**

* Основные требования к фильтрации пользовательского ввода.

**Ход работы:**

В первой части задания планировалось использовать запрос такого вида: ' UNION SELECT username, password FROM users WHERE username="admin" -- ", где одинарная кавычка закрывает ожидаемое строковое значение, что позволяет внедрить свой SQL-код.

UNION — оператор объединения двух SELECT-запросов. Он используется для обхода стандартных ограничений запроса, добавляя новые результаты в вывод.

SELECT null, login, password FROM users WHERE login='admin' в свою очередь получает логин и пароль пользователя с логином 'admin' из таблицы users. Первый null стоит на месте первого столбца, так как количество и порядок столбцов должны совпадать с основным запросом.

Однако запрос не подошел. При этом было выяснено, что сайт выдает подробную информацию при попытке ввести SQL-инъекцию (рисунок 1).



Рисунок 1 – информация об ошибке

Это позволяет составить примерную картину системы базы данных на основе введенных запросов. Было решено провести «разведку» на основе вывода ошибок. Если SQL-запрос выполняется без ошибки, но система возвращает «Wrong username or password», это значит, что SQL-инъекция сработала, но данные не отобразились.

Первым делом было решено проверить количество столбцов при помощи команды ORDER BY. Так было выяснено, что в таблице содержится два столбца для логина и пароля (рисунок 2).



Рисунок 2 – Ошибка сортировки

Далее было решено опробовать эту команду «' UNION SELECT login, password FROM users WHERE login="admin" -- "». Она тоже не подошла, из-за того, что один из столбцов был указан неверно. Как выяснилось этим столбцом оказался login. Методом перебора (а именно login, text, user, username) было выяснено, что названием столбца оказался username.

Итоговый вариант SQL-инъекции выглядит следующим образом: ' UNION SELECT username, password FROM users WHERE username="admin" -- ". Данный вариант подошел и значение пароля администратора было получено, а первое задание выполнено (рисунок 3).



Рисунок 3 – Значение пароля администратора

Сайт на второй задаче имеет систему защиты от ввода «опасных» символов, а также не выдает излишней информации (рисунок 4).



Рисунок 4 – Обнаружение ввода «опасных» символов

Однако, несмотря на скудность информации, выводящийся при ошибке все езе можно составить представление о базе данных. Аналогично заданию 1, если при запросе выводится «Wrong username / password» - это значит, что что SQL-инъекция сработала, но данные не отобразились.

Так было выяснено, что символы --, # и = сайт расценивает как попытку инъекции. Сайт не блокирует символы, а значит можно воздержаться от сложны конструкций на подобии «' UNION SELECT 1,2,3 FROM users WHERE username=CHAR(97,100,109,105,110) –», где функция CHAR(97,100,109,105,110) эквивалентна строке "admin". Способ обхода пробелов при помощи /\*\*/ тоже не нужен, ибо сайт не считает одинарную кавычку и пробел угрозой.

Затем было опробован вариант URL-кодирования инъекции. Выглядит он следующим образом: %27%20%55%4E%49%4F%4E%20%53%45%4C%45%43%54%20username,password%20%46%52%4F%4D%20users%20%57%48%45%52%45%20username=%22admin%22%20%23.

Здесь каждый символ, записанный как %XX, представляет собой HEX-код ASCII-символа. Многие веб-приложения автоматически экранируют спецсимволы, но иногда можно обойти фильтрацию, передав символы в URL-кодировке. Если декодировать сообщение, то будет: ' UNION SELECT username, password FROM users WHERE username="admin" #. Однако запрос не подошел и система все еще жаловалась на «опасные» символы, а именно на знак решетки.

Далее использовался такой вариант: ' UNION SELECT username, password FROM users WHERE username LIKE admin' OR '1' LIKE '1. Но и он не подошел и выдавал ошибку «Wrong username / password». Однако после этого был опробован укороченный вариант инъекции, а именно: admin' OR '1' LIKE '1. Данный способ подошел, и пароль был получен (рисунок 5).



Рисунок 5 – Удачный вариант инъекции

Данный вариант подошел, из-за того, что запрос на авторизацию как правило выглядит следующим образом: SELECT \* FROM users WHERE username = 'ВХОДНЫЕ\_ДАННЫЕ' AND password = '...'.

Чтобы сохранить синтаксис, после 'admin' уже закрыта строка, поэтому дальше идет OR и снова '1' LIKE '1' – отдельное логическое выражение. В итоге запрос для сайта выглядит следующим образом: SELECT \* FROM users WHERE username = 'admin' OR '1' LIKE '1' AND password = '...'. Так и было выполнено второе задание.

**Вывод:**

В ходе выполнения задания были успешно проведены SQL-инъекции для обхода аутентификации в уязвимых веб-приложениях. Анализ ошибок вывода позволил выявить структуру базы данных и подобрать корректный вариант инъекции. Основными методами атаки стали использование оператора UNION SELECT для извлечения данных из базы и логические выражения с оператором OR для обхода проверки пароля. Второе задание было сложнее из-за наличия фильтрации специальных символов, но удалось подобрать рабочий вариант инъекции.

В результате выполнения работы можно сделать следующие выводы:

* Ошибочные сообщения могут помочь атакующему составить представление о структуре базы данных.
* Фильтрация специальных символов усложняет атаку, но ее можно обойти с использованием альтернативных техник, таких как URL-кодирование или логические выражения.

Для предотвращения SQL-инъекций необходимо соблюдать следующие меры безопасности:

* Использование параметризованных запросов
* Вместо подстановки пользовательских данных непосредственно в SQL-запрос следует использовать подготовленные выражения (prepared statements) и параметризованные запросы.
* Ограничение прав пользователей базы данных
* Ограничить права учетных записей базы данных, используемых веб-приложением, запрещая выполнение опасных команд (DROP, DELETE, UPDATE и др.).
* Очищать ввод пользователя от SQL-специфичных символов, таких как одинарная кавычка ('), двойная кавычка (") и комментарии (--, #).
* Проверка и валидация данных на стороне сервера
* Разрешать ввод только допустимых значений (например, для полей логина разрешать только буквы и цифры, исключая пробелы и специальные символы).
* Ограничение вывода ошибок