

УДК 004.056.53

Исследование уязвимостей серверов, использующих SQL-запросы путем SQL-инъекций

Сосулев Лев Геннадьевич

slg21ki234@student.bmstu.ru

Носов Георгий Дмитриевич

ngd21ki185@student.bmstu.ru

Ильичев Владимир Юрьевич

ilyichev.vyu@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

В современном мире, где базы данных играют ключевую роль в функционировании веб-приложений, SQL-инъекции остаются одной из наиболее распространенных и опасных угроз безопасности. Рассмотрена уязвимость, связанную с недостаточной фильтрацией пользовательского ввода (CWE-89), а также последствия успешных атак, которые могут привести к компрометации данных, потере конфиденциальности и финансовым убыткам. Исследование направлено на выявление основных причин уязвимостей, анализ существующих методов защиты и разработку практических рекомендаций по предотвращению SQL-инъекций в программном обеспечении. Цель статьи — повысить осведомленность разработчиков и организаций о значимости безопасного программирования и важности внедрения эффективных мер контроля ввода данных.

Ключевые слова: SQL-инъекции, аутентификация, авторизация, информационная безопасность

Что такое SQL-инъекция. SQL-инъекция [1] — это вид атаки, при которой злоумышленник внедряет SQL-запросы через входные данные от пользователя в приложение. Если атака успешна, она позволяет получить доступ к конфиденциальной информации из базы данных, изменять данные (добавлять, обновлять или удалять записи), выполнять административные команды в базе данных (например, выключать СУБД), восстанавливать содержимое файлов, которые доступны СУБД, и, в некоторых случаях, даже выполнять команды операционной системы. SQL-инъекции являются одной из форм атак внедрения, при которых SQL-команды вводятся в пользовательские данные для манипуляции заранее заданными SQL-командами.

Модель угрозы SQL-уязвимости можно описать следующим образом:

- атаки с использованием SQL-инъекций позволяют злоумышленникам подделывать идентификационные данные, изменять существующую информацию, создавать проблемы с невозможностью подтверждения действий (например, отмена транзакций или изменение балансов), полностью раскрывать все данные системы, уничтожать их или делать недоступными, а также получать права администратора сервера базы данных;

- SQL-инъекции особенно распространены в приложениях на PHP [2] и ASP [3], так как они часто используют старые функциональные интерфейсы. В случае J2EE [4] и ASP.NET приложений такие уязвимости встречаются

реже, поскольку их программные интерфейсы позволяют лучше защититься от инъекций;

- степень тяжести SQL-инъекций зависит от уровня навыков и изобретательности атакующего, а также от глубины защиты системы, например ограничения привилегий соединений с сервером базы данных и других мер. В целом, SQL-инъекция — это угроза высокой степени риска.

Цель работы — создание средств для изучения механизма обнаружения SQL уязвимостей путем SQL-инъекции. На данный момент существует множество способов обнаружения SQL уязвимостей:

- тестирование с использованием стека запросов. Метод стека запросов предполагает добавление нового SQL-запроса после завершения текущего. Разработчики и тестировщики должны проверять, чтобы приложение не поддерживало стековые запросы (по возможности). Например, следует избегать использования многоэтапных запросов, которые позволяют выполнять стековые запросы;

- тестирование на основе ошибок. Этот метод использует SQL-ошибки, отображаемые пользователям. Злоумышленник вызывает ошибку и извлекает информацию из сообщения об ошибке, включая имена таблиц и другую служебную информацию, что облегчает компрометацию базы данных;

- тестирование на основе булевых условий. Метод булевых условий заключается в добавлении условий в запросы, где проверяется истинность или ложность результата. Таким образом злоумышленники могут получать данные о структуре базы данных через несколько условных запросов. Тестировщики могут использовать данный метод для выявления уязвимостей на основе булевых инъекций;

- тестирование Out-of-Band (слепая эксплуатация) [5]. Методы Out-of-Band применимы для слепых SQL-инъекций, когда атакующий не имеет доступа к результату запроса. Этот метод использует функции системы управления базами данных (СУБД) для выполнения внешних соединений и передачи результатов запросов на сервер атакующего;

- тестирование с использованием задержки по времени. Метод задержки по времени полезен для слепых SQL-инъекций. Он заключается в отправке запросов с инъекцией и измерении времени отклика сервера. Если ответ задерживается, это означает, что результат условного запроса верен.

Созданная авторами программа для обучения выявления SQL-уязвимостей состоит из следующих модулей:

- реляционная база данных, в которой хранится информация о пользователях приложения формата «Логин; Пароль»;

- сервер для связи приложения и базы данных, написанный на Flask — фреймворк для создания веб-приложений на языке программирования Python;

- Android приложение (клиент) для проведения тестирования базы данных на наличие SQL-уязвимости, написанное на Android Studio IDE.

Эксплуатация SQL уязвимости с помощью созданных средств состоит из следующих этапов.

Допустим, существует 2 пользователя — admin и user. Пароль пользователя user известен — user. Нужно узнать пароль admin — это будет ответом. Попробуем произвести вход в систему (рис. 2).

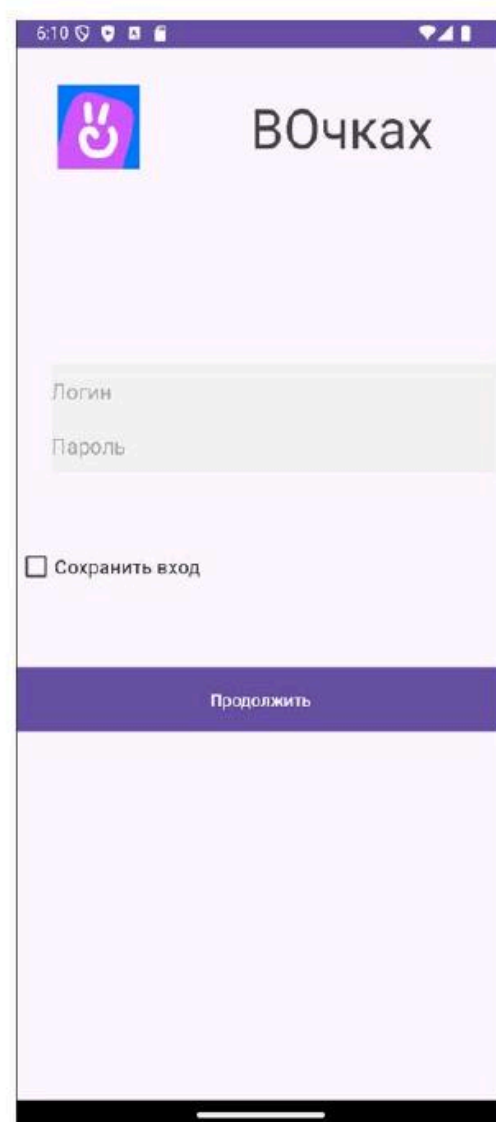


Рис. 1. Android приложение



Рис. 2. Android приложение после входа в систему за пользователя user

После входа в систему появляется кнопка «Показать пароль», нажав на которую высвечивается текст с паролем пользователя, под которым выполнен вход в систему. Можно сделать вывод, что если войти от пользователя `admin`, то получится узнать его пароль.

Приложение протестировано на наличие SQL-уязвимости на основе булевых условий. Вместо пароля пользователя `admin`, в соответствующее поле программы вписывается условие: `anything' OR '1'='1'`.



Рис. 3. Попытка входа в качестве пользователя `admin`

Данная конструкция обеспечит вход в аккаунт пользователя `admin`, если на сервере содержится булева SQL-уязвимость. В ответ на запрос к базе, она возвращает значение. Если это значение не нулевое, то внутри базы данных существует пользователь с такой учетной записью, тогда пользователю разрешено войти в систему, в противном случае доступ запрещен.

Проверяя пароль с конструкцией `"anything' OR '1'='1'"`, база данных считает, что пароль существует, так как возвращаемое значение от запроса пароля всегда равно 1. Таким образом, можно войти в систему с любым паролем, если введенное имя пользователя присутствует в базе.

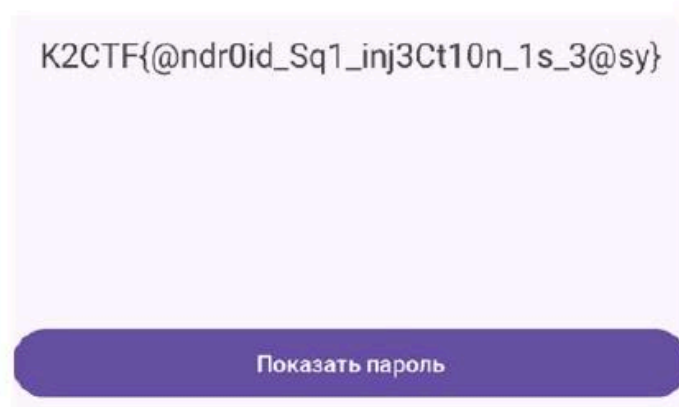


Рис. 4. Пароль пользователя admin

Вход в систему оказался успешным, это означает, что на сервере присутствует булева SQL-уязвимость. Нажав на кнопку «Показать пароль», высвечивается текст с паролем пользователя admin — “K2CTF{@ndr0id_Sq1_inj3Ct10n_1s_3@sy}”.

Таким образом, в ходе описанного исследования, созданы средства для изучения механизма обнаружения SQL-уязвимостей путем SQL-инъекции. В ходе проведения тестирования android приложения на наличие SQL-уязвимостей, был получен пароль пользователя, что свидетельствует об успешном обнаружении уязвимости.

Литература

- [1] Кошкарева Ю.И., Колесникова К.И. SQL-инъекции. *Молодежный научно-технический вестник*, 2015, № 10, 32 с.
- [2] Сорин Д.Б. PHP: the best web-programming language. *Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики*, 2006, № 27, с. 113–121.
- [3] Игнатов Ю.Ю. Технология ASP.Net в веб-проектах организаций. *Математические методы в технике и технологиях*, 2014, № 4 (63), с. 59–63.
- [4] Князев М.В. Абстрактные компоненты в проектировании информационных систем на базе платформы j2ee. *Алгоритмы, методы и системы обработки данных*, 2003, № 8, с. 105–115.
- [5] Мунтянов Д.Н., Додонов М.В. Особенности SQL-инъекций в системе управления базой данных oracle. *Молодежный научный форум: технические и математические науки*, 2017, № 5 (45), с. 63–67.

Research on vulnerabilities of servers using SQL queries through SQL injections

Sosulev L.G. slg21ki234@student.bmstu.ru
Nosov G.D. ngd21ki185@student.bmstu.ru
Ilyichev V.Y. ilyichev.vyu@bmstu.ru

BMSTU, Moscow, Russia

In the modern world, where databases play a crucial role in the functioning of web applications, SQL injections remain one of the most common and dangerous security threats. This article examines the vulnerability associated with insufficient user input filtering (CWE-89), as well as the consequences of successful attacks that can lead to data compromise, loss of confidentiality, and financial losses. The research aims to identify the main causes of vulnerabilities, analyze existing protection methods, and develop practical recommendations to prevent SQL injections in software. The article's goal is to raise awareness among developers and organizations about the importance of secure programming and the significance of implementing effective data input control measures.

Keywords: SQL injections, authentication, authorization, information security