УДК 004

Проведение простой SQL-инъекции на локальном сайте

Языков Кирилл Александрович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье на учебном примере рассматривается эксплуатация уязвимости типа «SQL-инъекция». В локальной среде OpenServer Panel (OSPanel) развёрнут простой сайт с формой авторизации и база данных MySQL. Показано, как из-за небезопасной подстановки пользовательских данных в SQL-запрос становится возможным обойти проверку пароля, войти под учётной записью администратора и получить полный список учётных записей. Приведён способ устранения уязвимости.
Ключевые слова: SQL-инъекция, информационная безопасность, веб-уязвимости, PHP, MySQL, OSPanel, авторизация, подготовленные выражения.

Performing a simple SQL injection on a local website

Yazykov Kirill Alexandrovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The paper uses an educational example to examine the exploitation of an SQL injection vulnerability. A simple website with a login form and a MySQL database are deployed in a local OpenServer Panel (OSPanel) environment. It is shown how the insecure insertion of user-supplied data into an SQL query makes it possible to bypass the password check, log in as the administrator, and retrieve the full list of accounts. A way to fix the vulnerability is also described.
Keywords: SQL injection, information security, web vulnerabilities, PHP, MySQL, OSPanel, authentication, prepared statements.

1 Введение

1.1 Актуальность

Большинство современных веб-приложений хранит данные в реляционных базах и формирует к ним SQL-запросы на основе введённой пользователем информации. Если эти данные подставляются в запрос без должной обработки, возникает уязвимость внедрения SQL-кода. Внедрение SQL-кода на протяжении многих лет остаётся одним из наиболее критичных рисков безопасности веб-приложений. Понимание механизма такой атаки на простом примере необходимо как разработчикам для построения защиты, так и специалистам по информационной безопасности. Локальный запуск

учебного стенда через OSPanel позволяет изучить уязвимость безопасно, не затрагивая сторонние ресурсы.

1.2 Обзор исследований

Вопросам разработки веб-приложений и веб-интерфейсов посвящён ряд работ. В.А. Борисенко и С.И. Белозерова рассмотрели применение современных информационных технологий при разработке дизайна веб-интерфейсов, подчеркнув важность удобства и функциональности для пользователей [1]. А.А. Леванов и А.Ю. Винокуров предложили подход к разработке интерактивных мультимедийных веб-приложений с использованием языка программирования JavaScript [2]. К.А. Тапелина и М.И.Жевалкина провели сравнительный анализ современных веб-систем учёта и обработки данных [3]. Непосредственно вопросам безопасности посвящена работа П.В. Полухина, в которой рассмотрены современные технологии поиска уязвимостей в веб-приложениях [4]. Наиболее полно методы поиска и эксплуатации уязвимостей веб-приложений, включая SQL-инъекции, описаны в зарубежной работе Д. Статтарда и М. Пинто [5].

1.3 Цель исследования

Целью работы является демонстрация на локальном учебном стенде механизма простой SQL-инъекции: обхода авторизации и получения списка всех учётных записей из базы данных без предварительного знания пароля администратора, а также описание способа устранения уязвимости.

2 Материалы и методы

Для выполнения работы использовались Windows 10, локальная серверная среда OpenServer Panel (OSPanel), включающая веб-сервер Apache, интерпретатор PHP и СУБД MySQL/MariaDB, а также веб-интерфейс HeidiSQL для управления базой данных. Проект размещён в каталоге домена, которому в настройках OSPanel сопоставлено доменное имя 1234.com, поэтому сайт открывается в браузере по адресу http://1234.com.

Учебный стенд состоит из трёх файлов: database.sql — дамп базы данных с таблицей учётных записей; db.php — модуль подключения к базе данных; index.php — страница с формой авторизации. Запрос авторизации в файле index.php намеренно собран конкатенацией строк без экранирования и без подготовленных выражений — именно это создаёт уязвимость. Проверка проводилась локально в браузере.

3 Результаты

На первом этапе создаётся база данных. В HeidiSQL локального сервера OSPanel выполняется импорт файла database.sql, который создаёт базу sqlinj_demo и таблицу users с полями id, username, password и role. Для наглядности пароли в учебной базе хранятся в открытом виде (рис. 1).

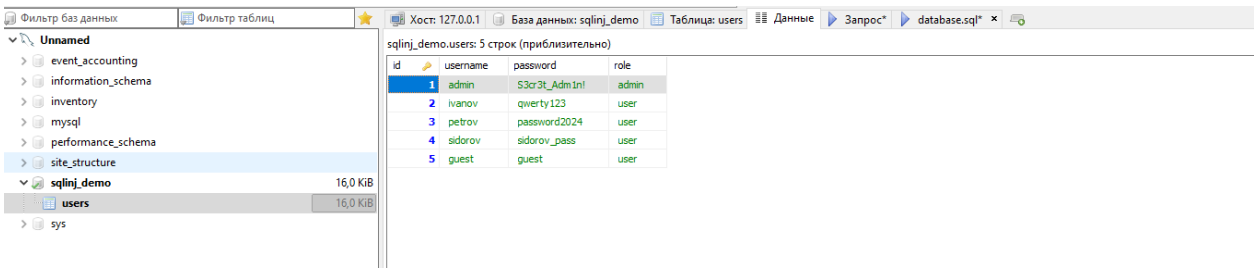


Рисунок 1 – Импорт учебной базы данных в HeidiSQL

После импорта таблица users содержит несколько учётных записей, включая запись администратора с ролью admin.

Файл db.php содержит параметры подключения к базе данных, соответствующие настройкам OSPanel по умолчанию: хост localhost, пользователь root, пустой пароль (рис. 2).



Рисунок 2 – Код файла db.php

Ключевым является файл index.php. В нём значения логина и пароля из формы подставляются напрямую в текст SQL-запроса. Итоговый запрос имеет вид: SELECT id, username, password, role FROM users WHERE username = '<логин>' AND password = '<пароль>'. Именно отсутствие экранирования делает возможной инъекцию (рис. 3).

index.php

```
<head>
    <style>
        .debug {
            background: #f7f9fc; border: 1px dashed #c3ccd6;
            border-radius: 6px; padding: 10px; word-break: break-word;
        }
        .wide { max-width: 760px; }
    </style>
</head>
<body>
    <div class="card <?php echo $rows ? 'wide' : ''; ?>">
        <h1>Личный кабинет</h1>
        <p class="sub">Учебный стенд. Вход для сотрудников.</p>

        <form method="post" autocomplete="off">
            <label for="username">Логин</label>
            <input type="text" id="username" name="username" value="">

            <label for="password">Пароль</label>
            <input type="password" id="password" name="password" value="">

            <button type="submit">Войти</button>
        </form>

        <?php if ($welcome): ?>
            <div class="msg-ok"><?php echo $welcome; ?></div>
        <?php endif; ?>

        <?php if ($error): ?>
            <div class="msg-err"><?php echo htmlspecialchars($error); ?></div>
        <?php endif; ?>

        <?php if ($rows): ?>
            <table>
                <tr><th>ID</th><th>Логин</th><th>Пароль</th><th>Роль</th></tr>
                <?php foreach ($rows as $r): ?>
                    <tr>
                        <td><?php echo htmlspecialchars($r['id']); ?></td>
                        <td><?php echo htmlspecialchars($r['username']); ?></td>
                        <td><?php echo htmlspecialchars($r['password']); ?></td>
                        <td><?php echo htmlspecialchars($r['role']); ?></td>
                    </tr>
                <?php endforeach; ?>
            </table>
        <?php endif; ?>

        <?php if ($queryShown): ?>
            <div class="debug"><strong>Выполненный запрос:</strong><br><?php echo htmlspecialchars($queryShown); ?></div>
        <?php endif; ?>
    </div>
</body>
</html>
```

Рисунок 3 – Уязвимый фрагмент кода index.php

Открыв в браузере адрес домена, пользователь видит форму входа. При вводе произвольных неверных данных система сообщает «Неверный логин или пароль» — это штатное поведение (рис. 4).

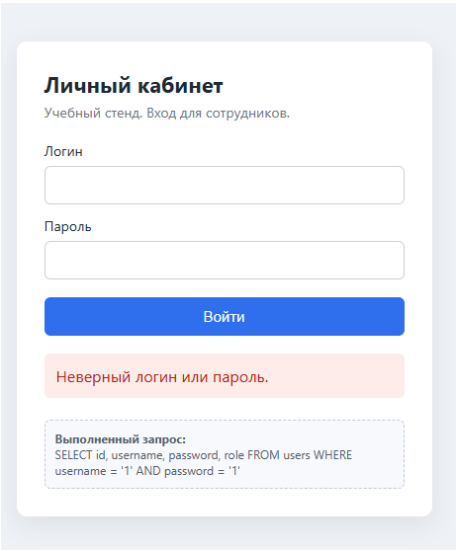


Рисунок 4 – Форма авторизации и неудачная попытка входа

Первый сценарий — обход авторизации. В поле «Логин» вводится строка admin'-- (после двойного дефиса стоит пробел), а поле «Пароль» заполняется любым значением. После подстановки запрос принимает вид: ... WHERE username = 'admin'-- ' AND password = '...'. Последовательность -- является началом комментария в SQL, поэтому остаток запроса с проверкой пароля отбрасывается. База возвращает строку администратора, и выполняется вход под его учётной записью без знания пароля (рис. 5-6).

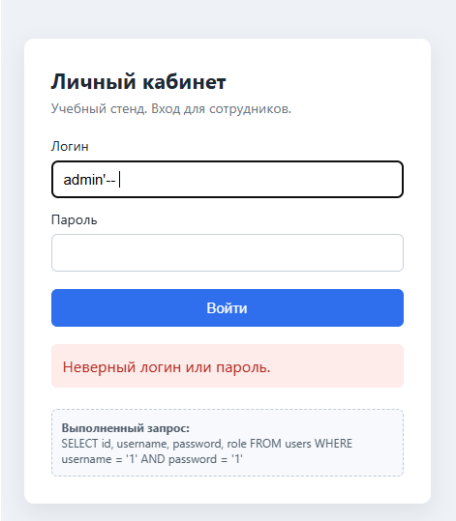


Рисунок 5 – Обход авторизации и вход под учётной записью администратора (часть 1)

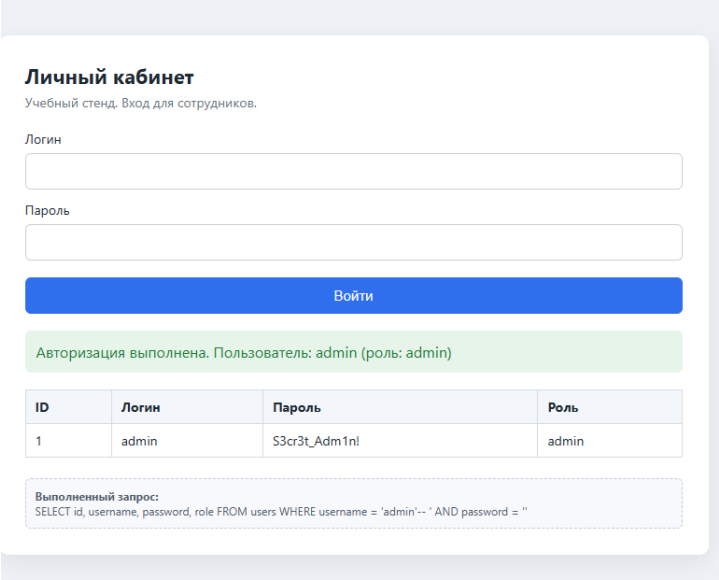


Рисунок 6 – Обход авторизации и вход под учётной записью администратора (часть 2)

Второй сценарий — получение всех учётных записей. В поле «Логин» вводится строка ' OR '1'='1'-- (с пробелом после дефисов), пароль — любой. Условие '1'='1' истинно для каждой строки таблицы, поэтому запрос возвращает все записи. На странице отображается таблица со всеми логинами,

паролями и ролями пользователей, то есть происходит выгрузка всех учётных данных из базы (рис. 7).

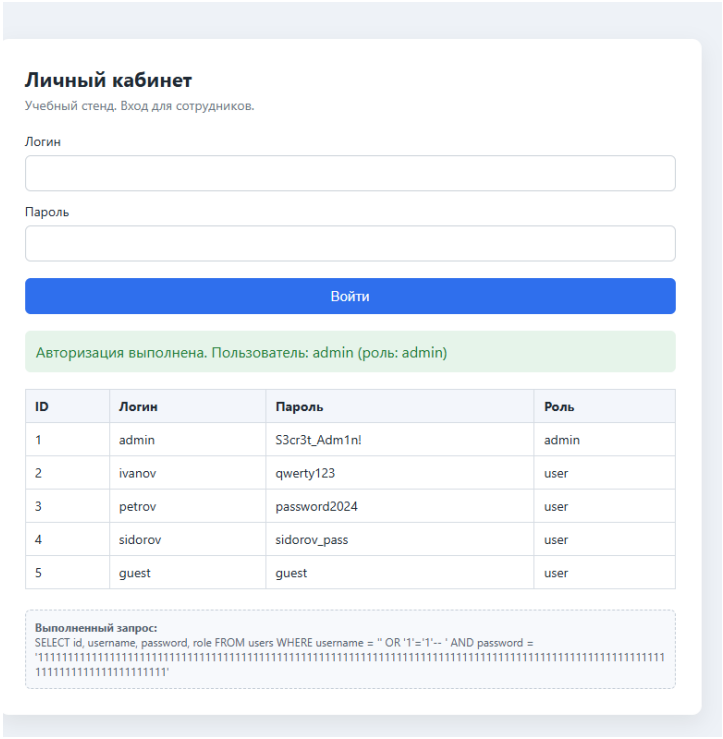


Рисунок 7 – Выгрузка всех учётных записей из базы данных

Таким образом, не зная заранее ни одного пароля, можно получить полный перечень учётных записей, в том числе пароль администратора, и войти в систему с его правами. Причина уязвимости — прямая конкатенация пользовательского ввода в SQL-запрос.

Для устранения уязвимости применяются подготовленные выражения (prepared statements). Пользовательский ввод передаётся в запрос как параметр, а не как часть SQL-кода, поэтому символы кавычек и комментариев теряют своё управляющее значение. Безопасный вариант использует mysqli_prepare, bind_param и выполняет запрос SELECT ... WHERE username = ? AND password = ? (рис. 8-9). После такой доработки приведённые выше строки воспринимаются как обычный текст и инъекция становится невозможной.

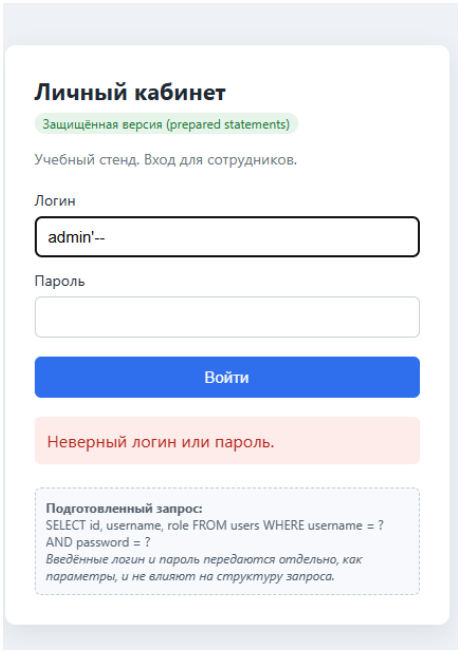


Рисунок 8 – Безопасный вариант запроса с использованием подготовленного выражения (часть 1)

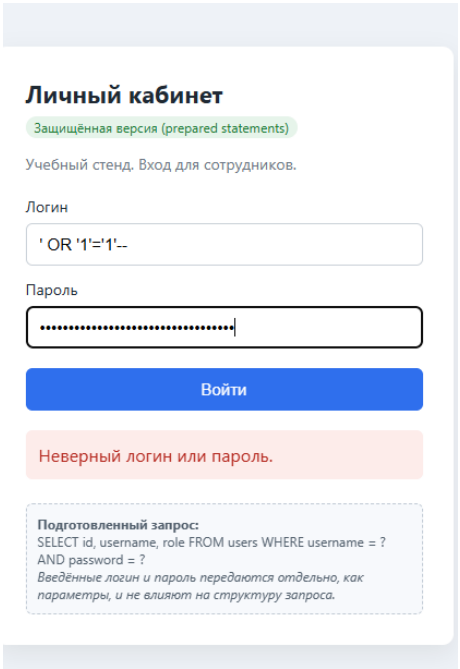


Рисунок 9 – Безопасный вариант запроса с использованием подготовленного выражения (часть 2)

4 Выводы

В ходе работы на локальном учебном стенде, развёрнутом средствами OSPanel, продемонстрирован механизм простой SQL-инъекции. Показано, что небезопасная подстановка пользовательских данных в SQL-запрос позволяет обойти проверку пароля, войти под учётной записью администратора и получить полный список учётных записей, не зная паролей заранее. Рассмотрен способ устранения уязвимости — применение подготовленных выражений. Материал может использоваться как учебный пример при

изучении безопасности веб-приложений и демонстрирует важность корректной обработки пользовательского ввода.

Важно: все описанные действия выполнялись исключительно в образовательных целях на собственном локальном сайте, принадлежащем автору. Проведение подобных действий в отношении информационных систем и сайтов без согласия их владельца является противоправным. В соответствии со статьёй 272 Уголовного кодекса Российской Федерации «Неправомерный доступ к компьютерной информации» неправомерный доступ к охраняемой законом компьютерной информации, повлёкший её копирование, предусматривает уголовную ответственность [6]. Автор не несёт ответственности за использование материалов статьи в противоправных целях.

Библиографический список

1. Борисенко В. А., Белозерова С. И. Применение современных информационных технологий при разработке дизайна веб-интерфейсов // Научно-техническое и экономическое сотрудничество стран АТР в XXI веке. 2021. Т. 2. С. 232–235. URL: https://science.vvsu.ru/files/F1CD7FA9-3C32-40F3-B646-AB6780447825.pdf#page=233 (дата обращения: 24.10.2026).
2. Леванов А. А., Винокуров А. Ю. Новый подход к разработке интерактивных мультимедийных электронных учебников // Электронное обучение в непрерывном образовании. 2015. № 1. С. 99–102. URL: https://www.elibrary.ru/download/elibrary_23152177_98884560.pdf (дата обращения: 24.10.2026).
3. Тапелина К. А., Жевалкина М. И. Анализ современных методик и web-систем учета достижений студентов вузов // Современные научные исследования и инновации. 2014. № 10-1. С. 60–67. URL: https://web.snauka.ru/issues/2014/10/38828 (дата обращения: 24.10.2026).
4. Полухин П. В. Фаззинг – современная технология поиска уязвимостей в веб-приложениях // Наука и современность. 2012. URL: https://cyberleninka.ru/article/n/fazzing-sovremennaya-tehnologiya-poiska-uyazvimostey-v-veb-prilozheniyah (дата обращения: 24.10.2026).
5. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. 2nd ed. Indianapolis: Wiley, 2011. 912 p.
6. Уголовный кодекс Российской Федерации от 13.06.1996 № 63-ФЗ. Статья 272. Неправомерный доступ к компьютерной информации. URL: https://www.consultant.ru/document/cons_doc_LAW_10699/ (дата обращения: 24.10.2026).
7. Материалы разработки веб-приложения для проведения SQL-инъекции: https://drive.google.com/drive/folders/1q5cBpsnsiTZa1yCtSmtNViFuf9yaVA6b