Отчет по аудиту безопасности веб-приложения

Введение

В данном отчете представлен анализ безопасности веб-приложения для управления пользователями. Были исследованы основные уязвимости и предложены методы их устранения.

1. Уязвимости XSS (Cross-Site Scripting)

Найденные проблемы:

- В нескольких местах (например, в adminpage.php) данные пользователей выводятся без экранирования
- В edit.php и update.php используются htmlspecialchars(), но не везде

Методы защиты:

1. Всегда использовать htmlspecialchars() при выводе пользовательских данных
2. Установка заголовка Content-Type с charset
3. Использование HTTP-заголовка X-XSS-Protection

Исправления:

php

Copy

Download

// В adminpage.php заменяем простой вывод на экранированный

<td><?php echo htmlspecialchars($user['fio'], ENT_QUOTES, 'UTF-8'); ?></td>

2. Information Disclosure

Найденные проблемы:

- В db_connection.php отображаются детали ошибок подключения к БД
- В некоторых случаях показываются технические детали ошибок

Методы защиты:

1. Отключение вывода подробных ошибок в production
2. Логирование ошибок вместо вывода пользователю
3. Настройка корректных HTTP-заголовков

Исправления:

php

Copy

Download

// В db_connection.php заменяем die на более безопасный вариант

} catch (PDOException $e) {

error_log("Database connection error: " . $e->getMessage());

die("Произошла ошибка подключения к базе данных");

}

3. SQL Injection

Найденные проблемы:

- В delete_user.php параметр user_id передается напрямую в запрос
- В некоторых местах используются подготовленные выражения, но не везде

Методы защиты:

1. Повсеместное использование подготовленных выражений (PDO)
2. Валидация всех входящих параметров
3. Принцип минимальных привилегий для пользователя БД

Исправления:

php

Copy

Download

```
// В delete_user.php уже используется подготовленное выражение - это правильно
$stmt = $pdo->prepare("DELETE FROM users1 WHERE user_id = ?");
$stmt->execute([$user_id]);
```

4. CSRF (Cross-Site Request Forgery)

Найденные проблемы:

- Отсутствие CSRF-токенов в формах
- Возможность выполнения важных действий (удаление, изменение) без подтверждения

Методы защиты:

1. Добавление CSRF-токенов во все формы
2. Проверка HTTP Referer
3. Использование SameSite cookies

Исправления:

php

Copy

Download

```
// В начале скриптов добавляем генерацию токена
if (empty($_SESSION['csrf_token'])) {
   $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

// В формах добавляем скрытое поле
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
```

```
// При обработке POST проверяем токен
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("Недействительный CSRF-токен");
}
```

5. File Include

Найденные проблемы:

- В edit_user.php используется прямой include файла
- Возможность локального включения файлов (LFI)

Методы защиты:

1. Ограничение включаемых файлов белым списком
2. Проверка путей перед включением
3. Использование абсолютных путей

Исправления:

php

Copy

Download

```
// В edit_user.php лучше использовать require с полным путем
$allowed_includes = ['edit.php'];
if (in_array(basename($_GET['file']), $allowed_includes)) {
    require __DIR__ . '/' . basename($_GET['file']);
} else {
    die("Недопустимый файл");
}
```

6. File Upload

Найденные проблемы:

- В текущем коде нет функционала загрузки файлов, но стоит предусмотреть защиту

Методы защиты (на будущее):

1. Проверка MIME-типов
2. Ограничение расширений файлов
3. Хранение загруженных файлов вне корневой директории
4. Генерация случайных имен файлов

Пример защиты:

php

Copy

Download

```
$allowed_types = ['image/jpeg', 'image/png'];
$upload_dir = __DIR__ . '/../uploads/';

if (in_array($_FILES['file']['type'], $allowed_types)) {
    $filename = uniqid() . '.' . pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION);
    move_uploaded_file($_FILES['file']['tmp_name'], $upload_dir . $filename);
}
```

Заключение

В ходе аудита были выявлены и устранены основные уязвимости. Для поддержания безопасности приложения рекомендуется:

1. Регулярно обновлять PHP и используемые библиотеки
2. Внедрить систему мониторинга безопасности
3. Проводить периодические аудиты кода
4. Использовать статический анализ кода

Приложение стало значительно более защищенным после внедрения описанных мер.