Аудит безопасности веб-приложения (PHP + MySQL)
Автор: Афонин Владислав Александрович
Дата: 15.05.2025
Репозиторий: https://github.com/Kemposonik/6

 1. Защита от XSS (Cross-Site Scripting)
Уязвимость: внедрение вредоносного HTML/JS через формы и URL-параметры.
Пример атаки: <script>alert('XSS')</script> при редактировании имени.

Методы защиты:

Использование htmlspecialchars() для вывода пользовательских данных:

<?= htmlspecialchars($user['name']) ?>
Запрет ввода HTML/JS в полях:

$name = strip_tags($_POST['name']); // опционально

 2. Защита от Information Disclosure (утечка служебной информации)
Уязвимость: отображение технических ошибок пользователю (например, SQL ошибки, stack trace).

Методы защиты:

Выключение display_errors в php.ini в продакшене:

display_errors = Off
log_errors = On
Обработка ошибок вручную:

mysqli_report(MYSQLI_REPORT_OFF);
Использование пользовательских сообщений:

if (!$stmt->execute()) {
    error_log("Ошибка запроса: " . $stmt->error);
    die("Произошла ошибка. Попробуйте позже.");
}
 3. Защита от SQL Injection
Уязвимость: внедрение произвольных SQL-запросов через форму/URL.

Методы защиты:

Использование подготовленных выражений (prepared statements):

$stmt = $conn->prepare("SELECT * FROM users WHERE login = ?");
$stmt->bind_param("s", $login);
$stmt->execute();
Никаких $_GET и $_POST напрямую в SQL:

// ❌ Нельзя:
$sql = "SELECT * FROM users WHERE id = " . $_GET['id'];
 4. Защита от CSRF (Cross-Site Request Forgery)
Уязвимость: злоумышленник может отправить вредоносный POST-запрос от имени пользователя.

Методы защиты:

Генерация и проверка CSRF-токена:

```
// В форме
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));

html
<input type="hidden" name="csrf_token" value="<?= $_SESSION['csrf_token'] ?>">


// При обработке POST
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("CSRF token mismatch");
}
```

 5. Защита от Include-уязвимостей

Уязвимость: подключение непроверяемых файлов через include/require с параметрами от пользователя.

Методы защиты:

Никогда не включать файлы, путь к которым приходит от пользователя.

Использовать whitelisting:

```
$pages = ['home', 'admin', 'profile'];
$page = in_array($_GET['page'], $pages) ? $_GET['page'] : 'home';
include "pages/{$page}.php";
```

 6. Защита от Upload-уязвимостей

Уязвимость: загрузка вредоносных .php/.exe файлов.

Методы защиты:

Проверка MIME-типа и расширения:

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

Переименование файла перед сохранением:

```
$ext = pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION);
$filename = uniqid() . '.' . $ext;
move_uploaded_file($_FILES['file']['tmp_name'], 'uploads/' . $filename);
```

Хранение вне веб-доступной директории.

  Заключение

В результате аудита были выявлены и устранены следующие уязвимости:

| Уязвимость | Метод защиты | Реализовано |
|---|---|---|
| XSShtml | specialchars(),  strip_tags() | yes |
| Information Leak | error_log, отключение display_errors | yes |
| SQL Injection | Prepared Statements | yes |
| CSRF | CSRF-токены в форме | yes |
| Include | Белый список допустимых файлов | yes |
| Upload | MIME-проверка, переименование файлов | no |

  Приложение

Ссылка на GIT-репозиторий: https://github.com/Kemposonik/6

Примеры исправленного кода: admin_panel.php, add_user.php, edit_user.php, upload.php, config.php