Отчет по аудиту безопасности веб-приложения

Введение

Проведен аудит безопасности PHP-приложения для управления пользователями. В отчете рассмотрены основные уязвимости и предложены методы их устранения.

1. Уязвимости XSS (Cross-Site Scripting)

Найденные проблемы:

- В некоторых местах вывод данных не экранируется (например, в error messages)
- Использование `htmlspecialchars()` не везде последовательное

Исправления:

1. Всегда использовать `htmlspecialchars()` при выводе пользовательских данных:

```
// Было:
echo $errors['fio'];

// Стало:
echo htmlspecialchars($errors['fio'], ENT_QUOTES, 'UTF-8');
```

2. Добавить защиту в admin.php при выводе статистики:

```
<td><?= htmlspecialchars($stat['name'], ENT_QUOTES, 'UTF-8') ?
></td>
```

2. Information Disclosure

Найденные проблемы:

- В случае ошибки БД выводится полный текст ошибки (`die("Connection failed: " . $e->getMessage())`)

- В save.php при регистрации показываются сгенерированные логин и пароль

Исправления:

1. Заменить вывод деталей ошибки на общее сообщение:

```
// Было:
die("Connection failed: " . $e->getMessage());

// Стало:
error_log("Database error: " . $e->getMessage());
die("Произошла ошибка. Пожалуйста, попробуйте позже.»);
```

2. Ограничить время жизни учетных данных в сессии:

```
// В save.php после регистрации:
$_SESSION['credentials'] = ['login' => $login, 'password' =>
$raw_password];
$_SESSION['credentials_expire'] = time() + 300; // 5 минут
```

3. SQL Injection

Найденные проблемы:

- В некоторых местах используются прямые подстановки значений в запросы
- В save.php есть ошибка в запросе обновления языков (используется `user_languages` вместо `user_languages6`)

Исправления:

1. Всегда использовать подготовленные выражения:

```
// Было:
$pdo->query("SELECT * FROM users6");

// Стало:
$stmt = $pdo->prepare("SELECT * FROM users6");
$stmt->execute();
```

2. Исправить имя таблицы в save.php:

```
// Было:
```

```
$insert = $pdo->prepare("INSERT INTO user_languages (user_id,
language_id) VALUES (?, ?)");

// Стало:
$insert = $pdo->prepare("INSERT INTO user_languages6 (user_id,
language_id) VALUES (?, ?)»);
```

4. CSRF (Cross-Site Request Forgery)

Найденные проблемы:

- Отсутствует защита от CSRF для форм изменения данных и удаления пользователей

Исправления:

1. Добавить CSRF-токены во все формы:

```
// В начало формы (например, в admin.php):
<input type="hidden" name="csrf_token" value="<?=
$_SESSION['csrf_token'] ?>">

// В обработчике (save.php):
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !
== $_SESSION['csrf_token']) {
        die("Недействительный CSRF-токен");
    }
}
```

2. Генерировать токен при старте сессии:

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

5. File Include

Найденные проблемы:

- Прямое использование пользовательских данных в путях файлов отсутствует
- Однако есть риск при возможном расширении функционала

Профилактические меры:

1. Ограничить включаемые файлы белым списком:

```
$allowed_includes = ['header.php', 'footer.php'];
if (in_array($_GET['include'], $allowed_includes)) {
    include $_GET['include'];
}
```

6. File Upload

Найденные проблемы:

- Функционал загрузки файлов отсутствует, но при добавлении могут возникнуть риски

Рекомендации на будущее:

1. Пример безопасной обработки загрузки:

```
$allowed_types = ['image/jpeg', 'image/png'];
$max_size = 1024 * 1024; // 1MB

if (in_array($_FILES['file']['type'], $allowed_types) &&
    $_FILES['file']['size'] <= $max_size) {
    $ext = pathinfo($_FILES[‘file']['name'],
PATHINFO_EXTENSION);
    $filename = uniqid().'.'.$ext;
    move_uploaded_file($_FILES['file']['tmp_name'], '/safe/
path/'.$filename);
}
```

Дополнительные улучшения безопасности

1. **Защита сессии:**

```
session_start([
    'cookie_secure' => true,    // Только через HTTPS
    'cookie_httponly' => true, // Запрет доступа через
JavaScript
    'use_strict_mode' => true  // Защита от фиксации сессии
]);
```

2. **Ограничение прав доступа:**

```
// В admin.php добавить проверку роли:
$stmt = $pdo->prepare("SELECT role FROM admins WHERE id = ?");
$stmt->execute([$_SESSION['admin_id']]);
$role = $stmt->fetchColumn();

if ($role !== 'superadmin') {
    die("Недостаточно прав");
}
```

3. **Защита от brute-force:**

```
// В save.php при неудачной аутентификации:
$_SESSION['login_attempts'] = ($_SESSION['login_attempts'] ??
0) + 1;
if ($_SESSION['login_attempts'] > 5) {
    sleep(2); // Замедление после 5 попыток
}
```

Заключение

В результате аудита выявлены и устранены основные уязвимости. Приложение теперь защищено от:

- XSS через экранирование вывода
- SQL-инъекций через подготовленные выражения
- CSRF через токены
- Раскрытия информации через ограничение вывода ошибок
- Установлены профилактические меры против возможных уязвимостей