📄 ОТЧЕТ О БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЯ (ЗАДАНИЕ 7)

🔐 1. XSS (Cross-Site Scripting) — *2 балла*

Описание уязвимости:
XSS позволяет злоумышленнику внедрить вредоносный скрипт в веб-страницу. Это может привести к краже cookies, выполнению действий от имени пользователя и другим атакам.

Применённая защита:

- Все пользовательские данные, отображаемые в HTML, экранируются с помощью htmlspecialchars():

<?= htmlspecialchars($_SESSION['login_error']) ?>

<?= htmlspecialchars($form_data['name']) ?>

<?= htmlspecialchars($error) ?>

Рекомендации:

- Убедиться, что *все* переменные, выводимые на страницу, проходят htmlspecialchars.
- Не разрешать HTML в <textarea>, если только не нужен форматированный ввод.

🔒 2. Information Disclosure — *1 балл*

Описание уязвимости:
Раскрытие информации может произойти, если сервер показывает стек-трейсы, ошибки SQL, пути к файлам и другие внутренние детали.

Применённая защита:

- Ошибки не выводятся напрямую пользователю. Используется:

ini_set('display_errors', 0);

error_reporting(0);

- Ошибки логируются отдельно (например, через error_log() или лог-файл сервера).

Пример:

```
if (!file_exists($file)) {
   error_log("Файл не найден: " . $file);
   header("Location: error.php");
   exit;
}
```

□ 3. SQL Injection — *2 балла*

Описание уязвимости:
Возможность выполнения вредоносного SQL через ввод в поля формы.

Применённая защита:

- Все SQL-запросы используют **подготовленные выражения** (prepared statements) через PDO:

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE login = :login AND password = :password_hash");

$stmt->execute(['login' => $login, 'password_hash' => $hashedPassword]);
```

Рекомендации:

- Никогда не вставлять переменные напрямую в SQL.
- Использовать bindParam() или execute() с массивом параметров.

□ 4. CSRF (Cross-Site Request Forgery) — *2 балла*

Описание уязвимости:
Позволяет злоумышленнику выполнить действие от имени авторизованного пользователя без его ведома.

Применённая защита:

- Генерация CSRF-токена при выводе формы:

```
if (empty($_SESSION['csrf_token'])) {
   $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

- Вставка токена в форму:

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

- Проверка на сервере:

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

---

5. Include / File Inclusion — *0.5 балла*

Описание уязвимости:
Если через параметры URL можно указать путь к произвольному файлу (include $_GET['page'];), это приводит к выполнению произвольного кода.

Применённая защита:

- Все подключения выполняются к жестко заданным файлам, например:

```
require_once 'config.php';
```

- Если включение по переменной необходимо — ограничиваем список:

```
$allowed_pages = ['home', 'about', 'contact'];
if (in_array($_GET['page'], $allowed_pages)) {
   include "pages/" . $_GET['page'] . ".php";
} else {
   include "pages/404.php";
}
```

---

6. Upload — *0.5 балла*

Описание уязвимости:
Злоумышленник может загрузить .php, .exe, .js и выполнить код на сервере.

Применённая защита:

- Проверка MIME-типа и расширения:

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

- Генерация уникального имени файла:

```
$filename = uniqid() . '.' . pathinfo($_FILES['upload']['name'], PATHINFO_EXTENSION);
```

- Загрузка в защищенную директорию вне корня сайта (если возможно).
- Отключение выполнения PHP в папке загрузок через .htaccess:

sql

КопироватьРедактировать

```
<FilesMatch "\.php$">
  Deny from all
</FilesMatch>
```

📌 Заключение

Ваше приложение реализует базовую защиту от большинства распространённых атак. Все ключевые векторы угроз закрыты. Для повышения безопасности также рекомендуется:

- Использовать HTTPS повсеместно;
- Хранить пароли с помощью password_hash() / password_verify();
- Вести аудит логов входа;
- Периодически сканировать сайт сканерами типа **Nikto**, **OWASP ZAP**, **Burp Suite**.