**Аудит безопасности**

CSRF

Механизм CSRF токенов гарантирует, что лишь аутентифицированные пользователи могут инициировать запросы к серверу. В моем коде применяется генерация уникальных токенов, проверяемых при каждом POST запросе.

//*Создание нового токена*, *если он еще не был создан*

if (!isset($\_SESSION['csrf\_token\_login'])) {

$\_SESSION['csrf\_token\_login'] = bin2hex(random\_bytes(*32*));

}

$csrfTokenLogin = $\_SESSION['csrf\_token\_login'];

//*………………*

//*Проверка правильности токена*

if (!checkCsrfToken($\_POST['csrf\_token\_login'])) { die('CSRF token validation failed.');

}

SQL Injection

Для предотвращения SQL инъекций применяются подготовленные выражения с параметризацией. Это исключает возможность интерпретации пользовательских данных как части SQL запроса. Пример из моего кода:

$stmt = $db->prepare("SELECT personId FROM personAuthentificationData WHERE login = :login AND pass = :pass");

$stmt->execute([':login' => $login, ':pass' => $md5Pass]);

XSS

Применение функции htmlspecialchars в моем коде обеспечивает защиту от XSS атак. Эта функция преобразует специальные символы в HTML-сущности, предотвращая выполнение вредоносного кода. Пример:

htmlspecialchars(trim($data));

Information Disclosure

Для избежания разглашения информации, конфиденциальные данные, например, данные для доступа к базе данных, хранятся в переменных окружения, которые не отображаются в коде. Например, данные для подключения к серверу хранятся в файле .env, и к ним обращение осуществляется в коде следующим образом:

$env = file\_get\_contents( *DIR* . '/.env');

$lines = explode("*\n*", $env);

foreach ($lines as $line) {

if (strpos($line, '=') !== *false*) {

list($name, $value) = explode('=', $line, *2*);

$\_ENV[$name] = trim($value, "*\" \r*");

}

}

$dbUser = $\_ENV['DB\_USER'];

$dbPassword = $\_ENV['DB\_PASSWORD'];

$dbName = $\_ENV['DB\_NAME'];

Upload

На моем сайте отсутствует возможность загрузки файлов, следовательно, уязвимость в области загрузки файлов также отсутствует.

Include

Для предотвращения уязвимостей, связанных с использованием include, необходимо гарантировать, что включаемые файлы не могут быть подменены или указаны пользователем.

$allowed\_paths = [ 'form.php',

];

function safe\_include($file) { global $allowed\_paths;

if (in\_array($file, $allowed\_paths)) { include($file);

} else {

echo *"Ошибка: попытка включения неразрешенного файла.";*

}

}

safe\_include('form.php');