Переделаны задания № 5 и 6

1. Защита от XSS:

Используются функции htmlspecialchars и strip\_tags при выводе данных в форму. Это помогает предотвратить внедрение HTML- и JavaScript-кода, защищая от XSS-атак.

Пример:

$values['name'] = empty($\_COOKIE['name\_value']) ? '' : htmlspecialchars(strip\_tags($\_COOKIE['name\_value']));

if (!empty($dates['name'])) {

$values['name'] = htmlspecialchars(strip\_tags($dates[0]['name']));

}

<input name="name" value="<?php echo $values['name']; ?>"/>

1. Защита от SQL Injection:

Используюься подготовленные запросы ($db->prepare и $stmt->execute) для выполнения запросов к базе данных. Это позволяет избежать SQL-инъекций путем корректной обработки параметров запроса.

Пример:

$stmt = $db->prepare("SELECT application\_id FROM users WHERE login = ?");

$stmt->execute([$login]);

1. Защита от CSRF:

Генерируется токен в сессии и проверяете его наличие и соответствие при обработке запросов. Это помогает предотвратить CSRF-атаки, гарантируя, что запросы отправлены только с моего веб-сайта.

Код:

$\_SESSION['token'] = bin2hex(random\_bytes(32));

<?php if (!empty($\_SESSION['login'])) {echo '<input type="hidden" name="token" value="' . $\_SESSION["token"] . '">'; } ?>

if (!empty($\_POST['token']) && hash\_equals($\_POST['token'], $\_SESSION['token'])) {

// сохранение изменённых данных в базу

} else {

die('Ошибка CSRF: недопустимый токен');

}

1. Защита от Include и Upload уязвимости.

Указывается путь к файлу в операторе include вручную. Это помогает предотвратить включение нежелательных файлов.