МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ОБ АУДИТЕ**

Выполнил студент группы 210                                       Е.К. Клыкова

Направление подготовки  09.03.03  Прикладная информатика Курс    2

Отчет приняла кандидат физико-математических наук, доцент                                                                                       Е.П. Лукащик

Краснодар

2022 г.

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc98113292)

[2 Защита от XSS 3](#_Toc98113293)

[3 Защита от SQL-INJECTION 4](#_Toc98113294)

[4 Защита от CSRF 4](#_Toc98113295)

[5 Защита от Upload и Include уязвимости 5](#_Toc98113296)

[6 Вывод 5](#_Toc98113297)

# 1 Цель работы

Провести аудит безопасности вашего приложения и исправить уязвимости. В нем должны быть разделы, посвященные уязвимостям XSS, SQL Injection, CSRF, Include, Upload. В отчете указать по каждой уязвимости примененные методы защиты с примерами вашего кода.

# 2 Защита от XSS

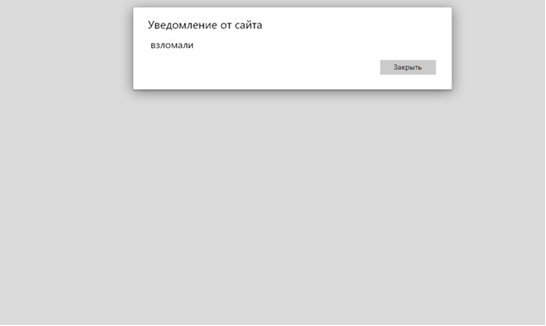
XSS - это когда злоумышленник пытается через формы на сайте добавить свой javascript-код, который затем выполнится в браузере админа/менеджера сайта или других пользователей и натворит дел.

Изначально, при загрузке в форму данные из базы не приводились к безопасному формату и была возможность провести XSS-атаку.

При заполнении любого текстового поля формы пользователь может ввести следующие данные:

(\*) ха'></textarea><script>alert("взломали")</script>

В результате перед каждым выводом формы с полем вида (\*) на экране будет всплывать окно с текстом, переданным как параметр функции alert. Атака также затронет и администратора, так как данные пользователя отображаются на его панели.

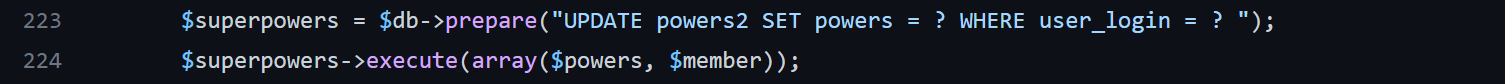


Решение: использование функции htmlspecialchars() перед каждым выводом текстового поля на экран. Когда мы запускаем функцию htmlspecialchars(), она берёт нашу строку и заменяет некоторые символы в ней (кавычки, угловые скобки и т.д.) на мнемоники, чтобы браузер гарантированно вывел нашу строку на экран, не пытаясь выполнять её как код. Т.е. когда в форму введут текст (\*), функция htmlspecialchars() превратит его в следующую строку: ха'></textarea><script>alert("взломали")</script >. Браузер уже не воспримет такой код как javascript и просто выведет на экран текст.

# 3 Защита от SQL-INJECTION

SQL-INJECTION - атаки, позволяющие злоумышленнику производить различные несанкционированные действия над базой данных. Они могут затрагивать как сами данные, так и структуру базы. Защита от sql-инъекций в веб-приложении предусмотрена использованием подготовленных запросов.

# 



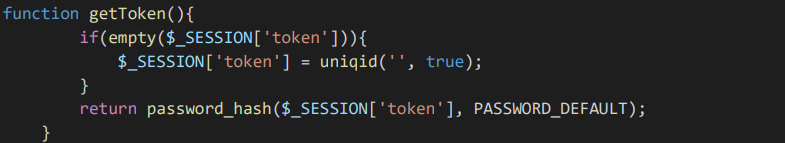
В данном случае для этого используются Prepare и Execute

# 4 Защита от CSRF

CSRF —вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер.

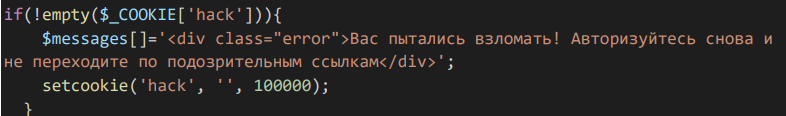
Защитим важные поля форм, (изменение записи в базе данных и удаление записи) добавив токен, привязанный к сессии пользователя и проверку токена. Под токеном имеется в виду случайный набор байтов, который сервер передает клиенту, а клиент возвращает серверу. Защита сводится к проверке токена, который сгенерировал сервер, и токена, который прислал пользователь. Токен находится в скрытом поле, должен быть непредсказуемым случайным числом и иметь небольшое время жизни, без возможности переиспользования.

Добавим в form.php функцию генерации токена и скрытое поле формы



Добавим в index.php функцию проверки токена и проверку на совпадение токенов:







Теперь при переходе пользователя по ссылке злоумышленника произойдет выход из сессии и удаление cookies. На экране отобразится сообщение о попытке взлома и просьба повторно авторизоваться.

# 5 Защита от Upload и Include уязвимости

Данных уязвимостей в моём случае нет, однако знать о них стоит.

 Внедрение PHP-кода (**PHP**-**Include**) — это **уязвимость**, заключающаяся в возможности внедрения и выполнения произвольного кода на языке PHP. **Уязвимость** возникает вследствие недостаточной проверки и контроля переменных, используемых внутри функций, осуществляющих подключение кода на этапе выполнения.

Безопасный upload файлов — это не только проверки в скриптах, но и грамотно решенный вопрос о местонахождении и обработке загруженных файлов. При этом также не стоит забывать и об особенностях самого PHP, связанных с массивом $\_FILES..

# 6 Вывод

Мы защитили сайт от различного рода уязвимостей в коде, атак. Теперь при загрузке в форму данные из базы приводятся к безопасному формату.