Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

Безопасность WEB приложений

Отчет по заданию

I-ой аттестации

Выполнил студент группы IA1802:

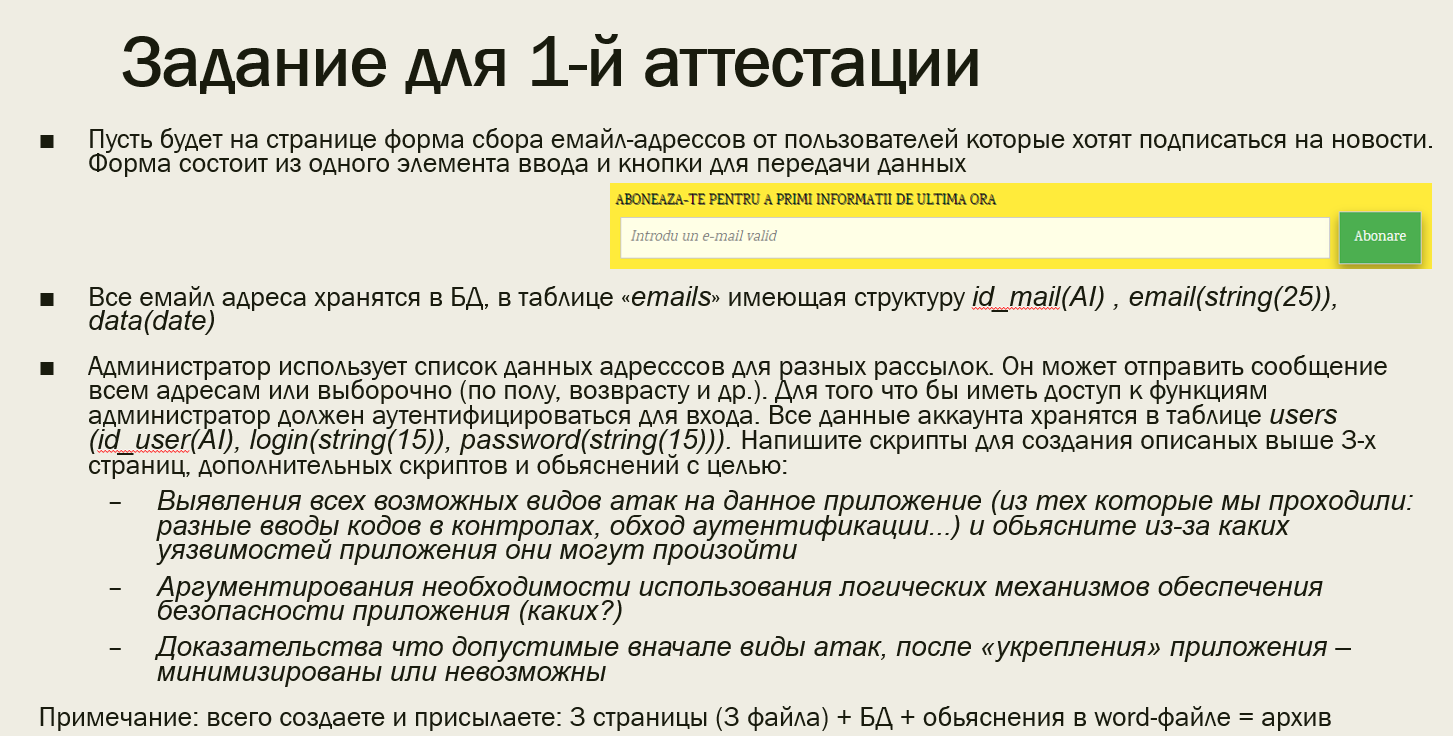
Паненко Е.

Проверил преподаватель:

Плешка Н.

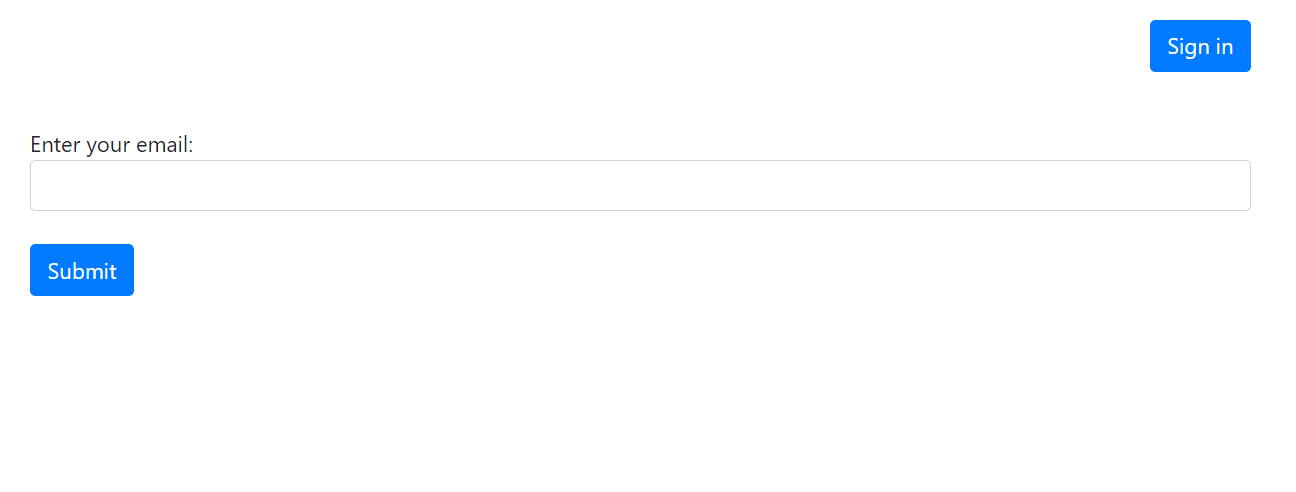
Кишинёв, 2020

**Условие задания:**

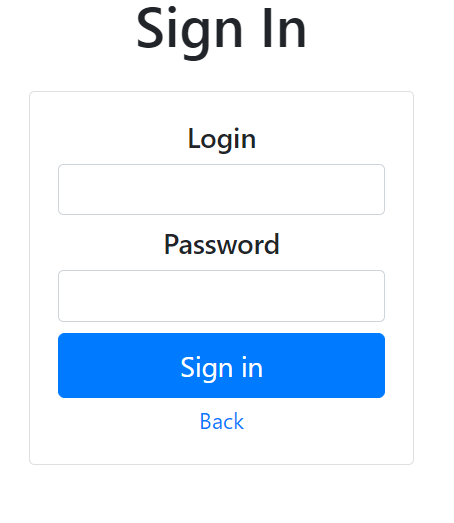


**Демонстрация функционала первоначального примера**:

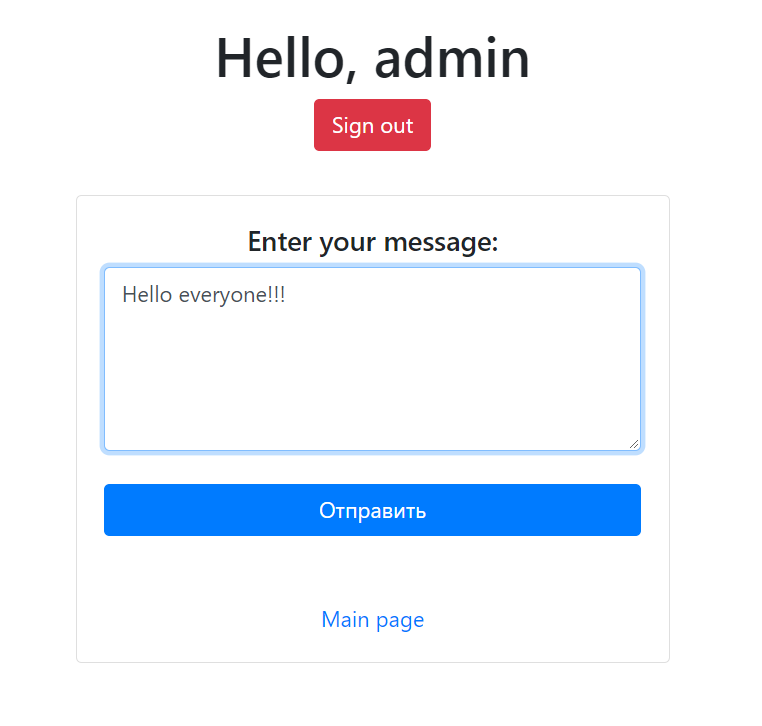
Первая страница, которая нас встречает – страница сбора email-ов



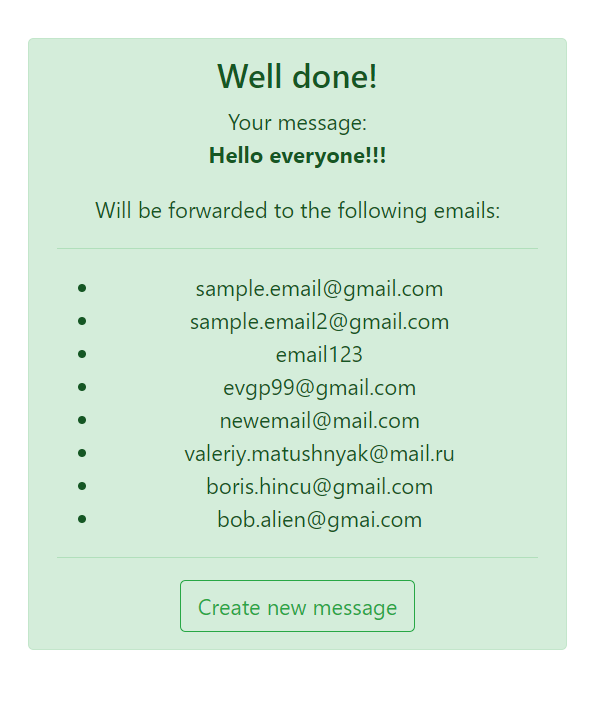
При переходе по ссылке Sign in переходим на страницу входа



При успешной авторизации пользователя перенаправит на пано, где можно будет написать сообщение которое будет отправлено другим пользователям, email-ы которых сохранены в БД.



При успешной отправке сообщения нас перенаправит на страницу отображающую статус отправки.



Здесь нам покажут наше сообщения и на какие email-ы оно было отправлено (для дальнейшей демонстрации XSS).

**Далее будем одновременно рассматривать незащищенный вариант и то как мы исправили его.**

Начнем с главной страницы сбора email-ов.

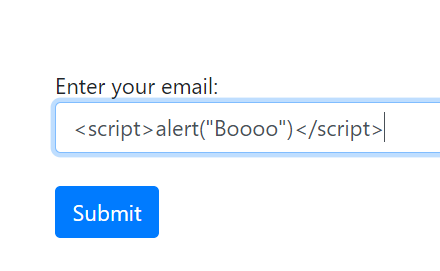
Первая проблема в том, что данный код уязвим к XSS атаке и так же к некорректному использованию.

Например, пользователь может ввести туда скрипт который не даст нормально пользоваться системой.

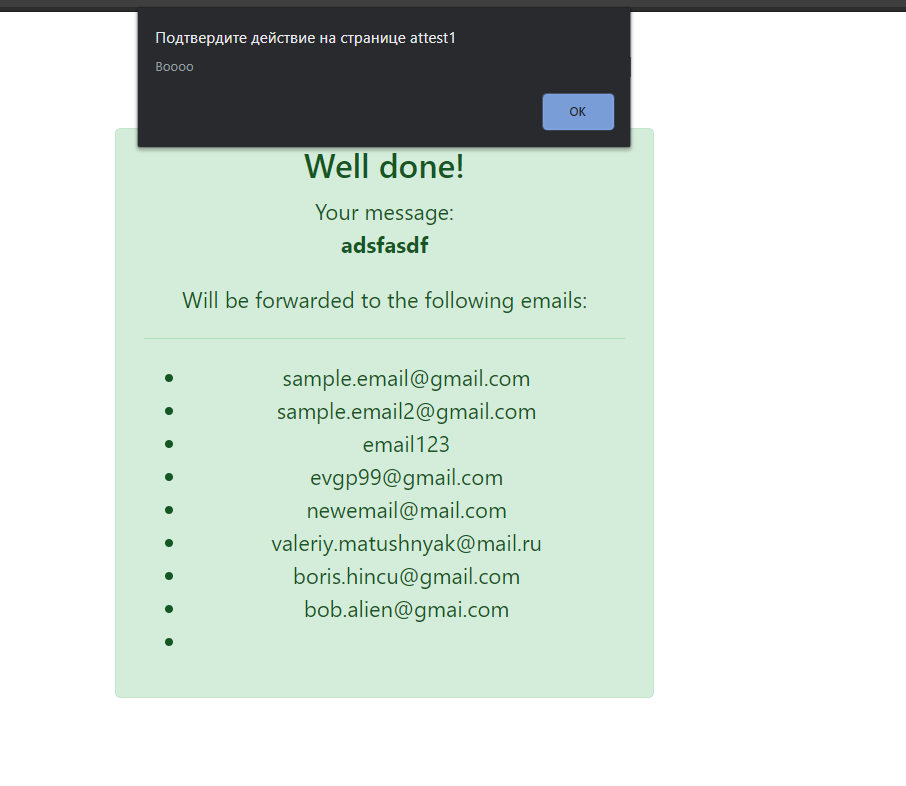
*Пример*:

Пользователь вводит следующее в поле email:

*<script>while(true)alert("Boooo");</script>*



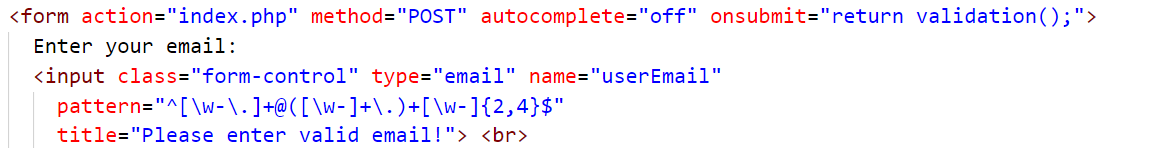
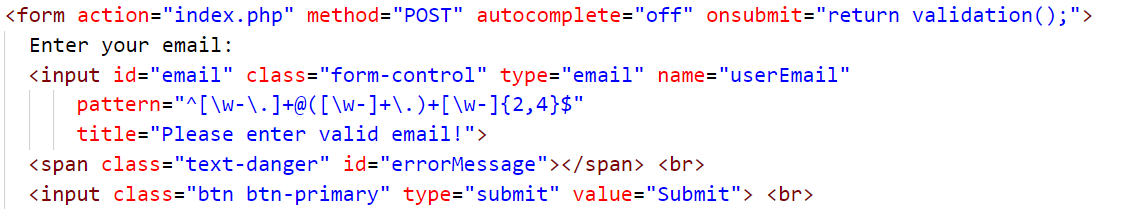
Теперь когда администратор будет отсылать сообщения он будет видеть следующее вместо статуса отправки



В следствие чего он не сможет ничего сделать

Решение проблемы просто – добавление валидации на сторону клиента и сервера, чтобы избежать добавления такой записи в БД.

Исправим html код



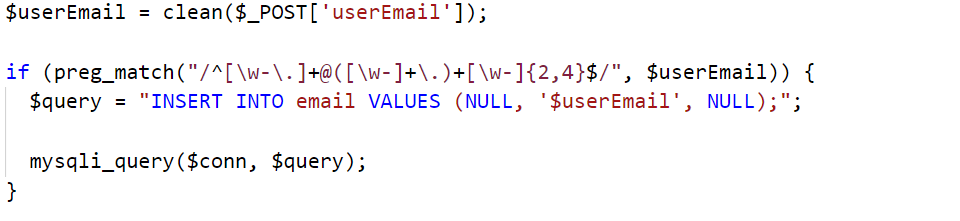
Отключим автозаполнение, поставим input’у email верный тип и добавим ещё дополнительный паттерн

Так же добавим javascript функцию которая будет дополнительно проверять email

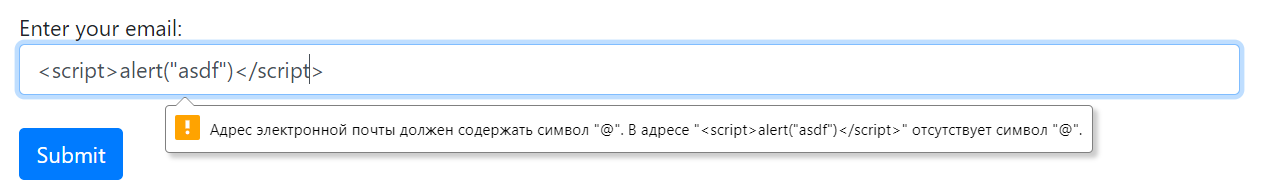
Следующим шагом добавим валидацию на стороне сервера

Для начала добавим функцию которая будет очищать наш input.

И добавим валидацию на стороне сервера



После проделанных действий в input email’а уже нельзя добавить скрипт, а в случае отключении защиты на стороне клиента сработает защита на стороне сервера.

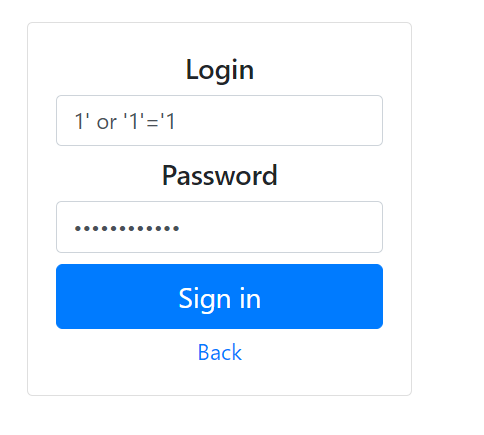


**Далее переходим на станицу входа**

Здесь присутствует 2 уязвимости, первая по аналогии с прошлым пример отсутствует валидация данных, что позволяет использовать SQL injection для доступа к странице menu. Вторая заключается в том что отсутствует контроль сессий, что позволяет прописав в пути верный путь попасть на страницу menu без входа в систему.

Демонстрация обеих уязвимостей:

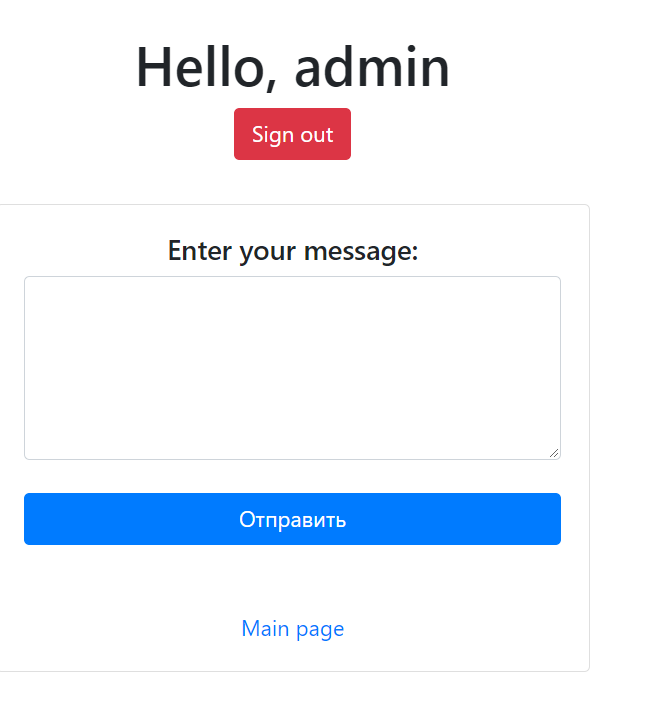
Вход с помощью SQL injection



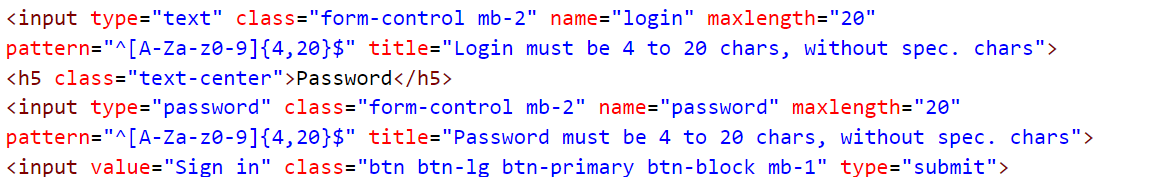
Вход с помощью отсутствия сессий



В обеих случаях получим доступ к сервису рассылки сообщений

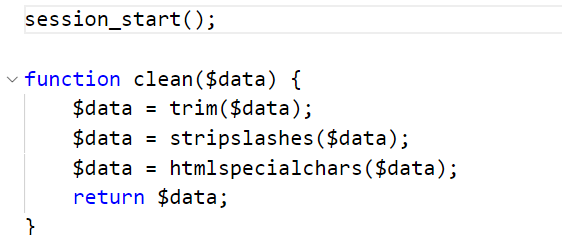


По аналогии с предыдущим пунктом исправляем наш html

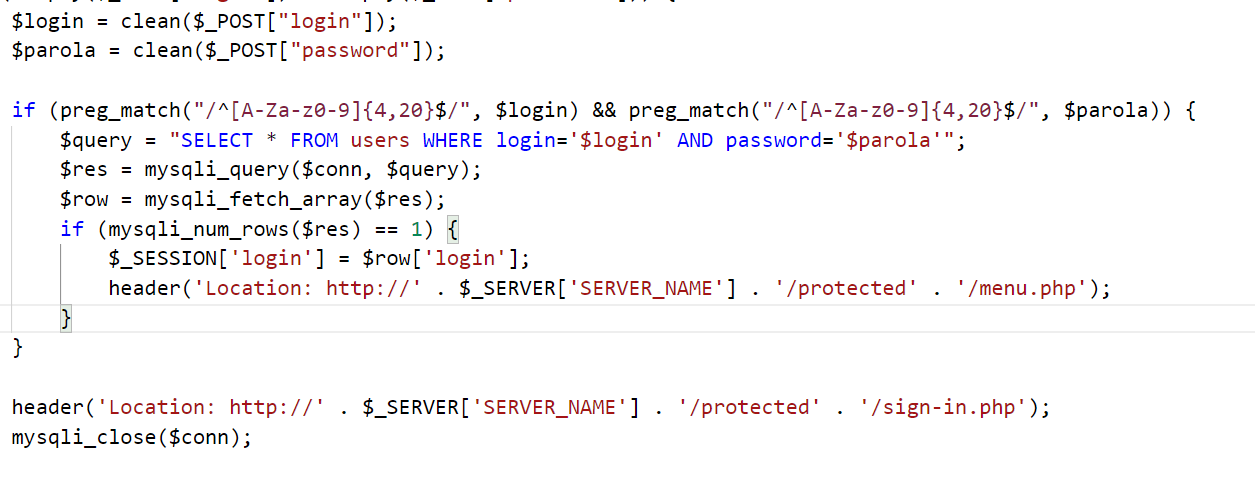


Далее добавим защиту на стороне сервера

Добавим ф-ию очистки и так же добавим поддержку сессий

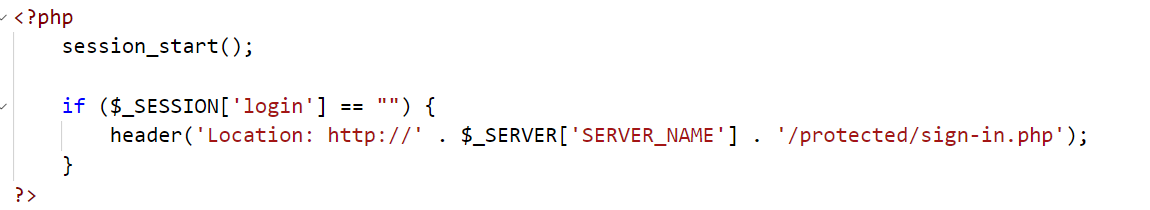


Во-первых очищаем наши input’ы, далее проверяем их на соответствие регулярному выражению, далее проверяем вернет ли наш запрос только одну строку и если результат положительный сохраняем переменную login в сессию и перенаправляем пользователя на страницу menu.



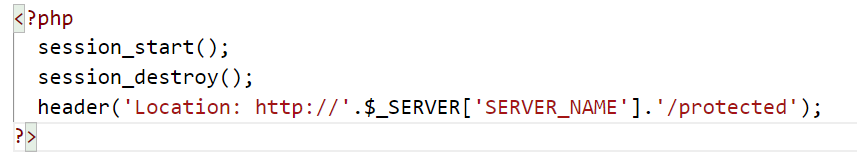
Уязвимость с переходом на menu.php осталась, но сейчас мы ее исправим

При переходе на menu.php будем проверять если пользователь вошел в систему, в противном случае будет перенаправлять его на страницу sign-in.php

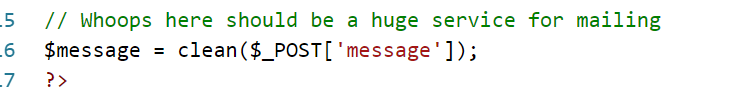


Аналогичный код добавим в файл send-message.php

Для завершения работы сессии добавим следующий код в sign-out.php, который будет уничтожать нашу сессию.



Так же хоть мы и не пишем код для отправки сообщений, но добавим код очищающий содержимое сообщения.



**Анализ проделанной работы:**

Изначальный вид нашего сервиса со стороны безопасности был уязвим к следующим атакам:

* XSS
* SQL Injection
* Отсутствие контроля сессий

Благодаря чему пользователь мог обойти систему аутентификации двумя способами, а также внедрить зловредные скрипты в нашу базу данных.

В ходе улучшения безопасности приложения мы использовали следующие приемы:

* Валидация данных на стороне клиента
* Валидация данных на стороне сервера
* Внедрение сессий

В результате мы получили сервис защищенный от вышеописанных атак.