Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

Безопасность WEB приложений

Индивидуальная работа

Выполнил студент группы IA1802:

Паненко Е.

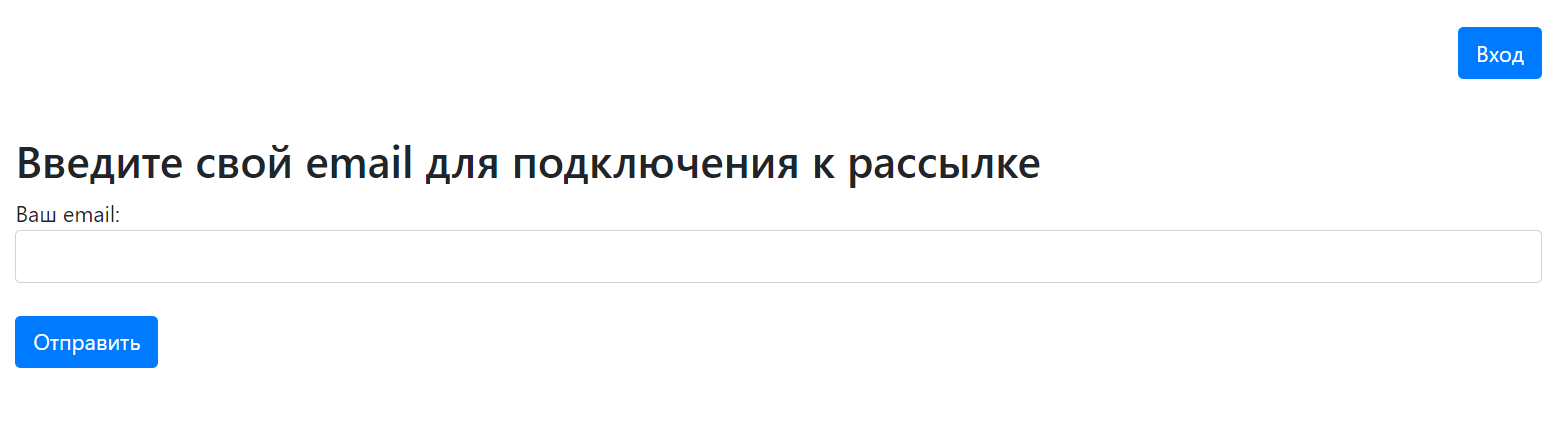
Проверил преподаватель:

Плешка Н.

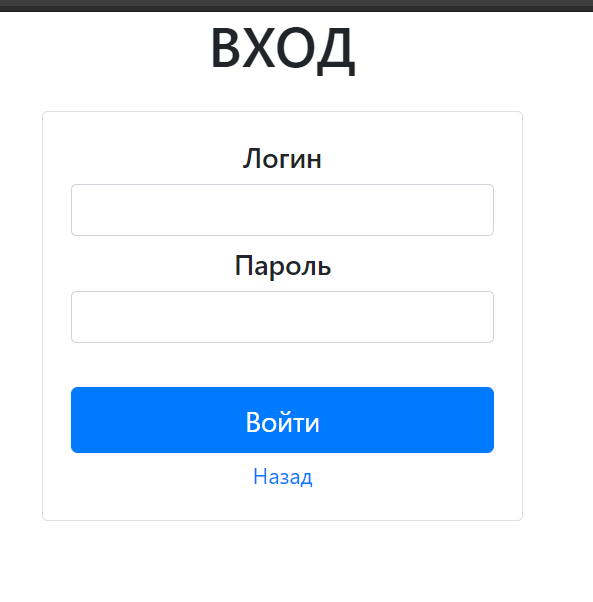
Кишинёв, 2020

**Демонстрация функционала первоначального примера**:

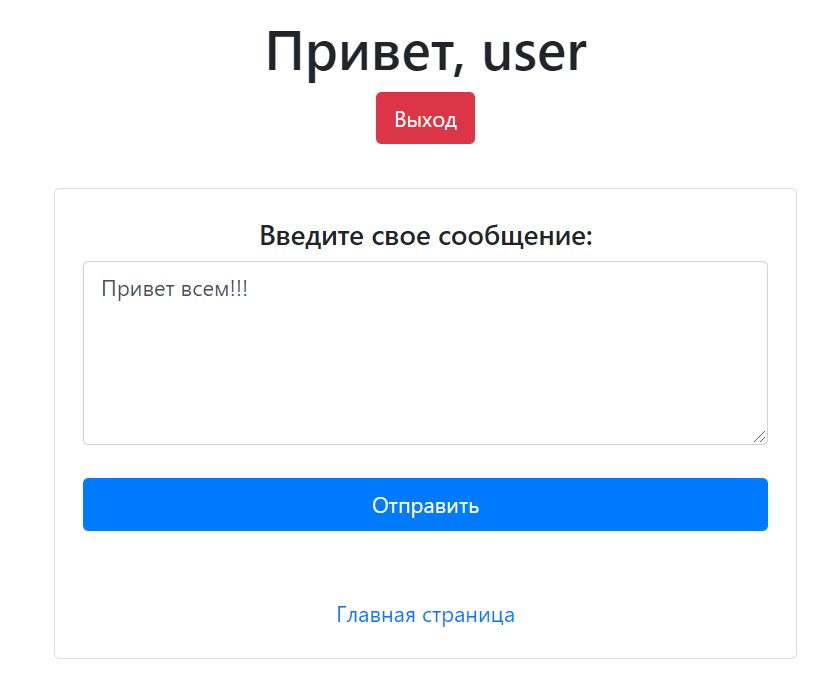
Первая страница, которая нас встречает – страница сбора email-ов



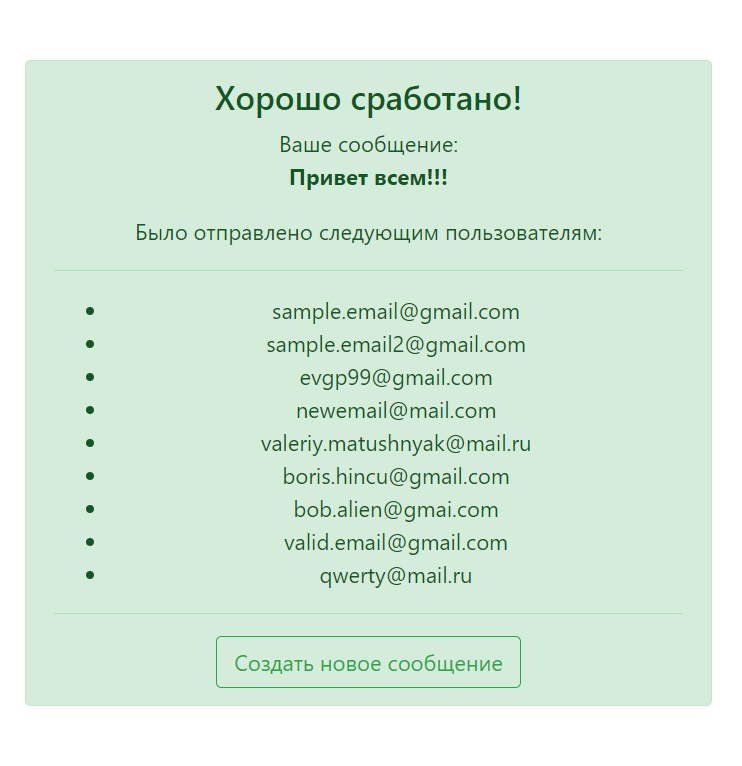
При переходе по ссылке Sign in переходим на страницу входа



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. (пока что работаем только с уязвимостями SQL Injection и XSS атакой)**

Начнем с главной страницы сбора email-ов.

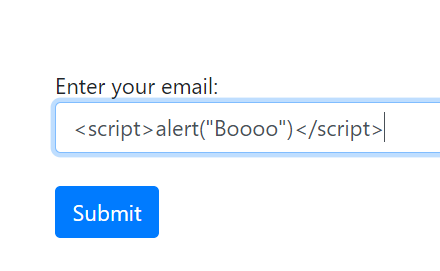
Первая проблема в том, что данный код уязвим к XSS атаке и так же к некорректному использованию.

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

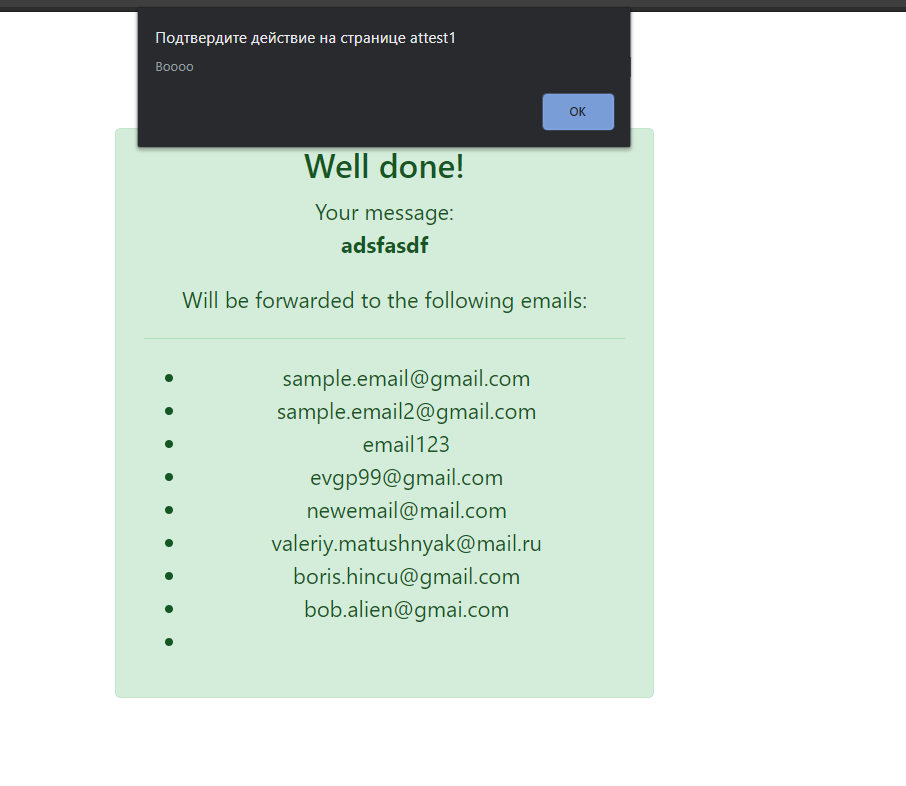
*Пример*:

Пользователь вводит следующее в поле email:

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



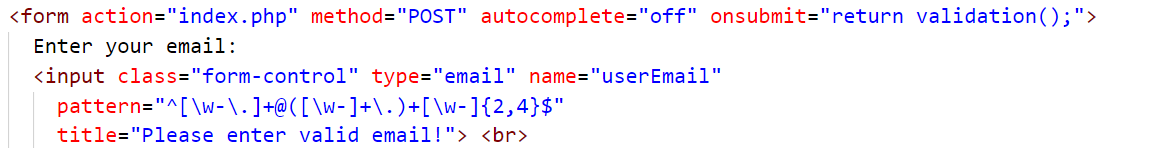
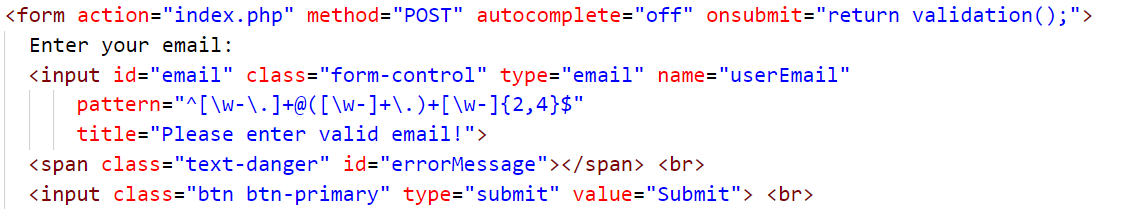
Теперь когда администратор будет отсылать сообщения он будет видеть следующее вместо статуса отправки



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



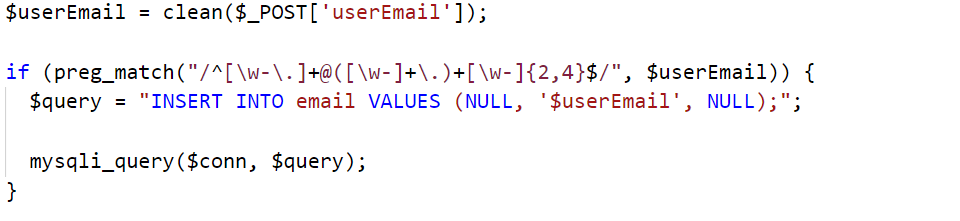
Отключим автозаполнение, поставим input’у email верный тип и добавим ещё дополнительный паттерн

Так же добавим javascript функцию которая будет дополнительно проверять email

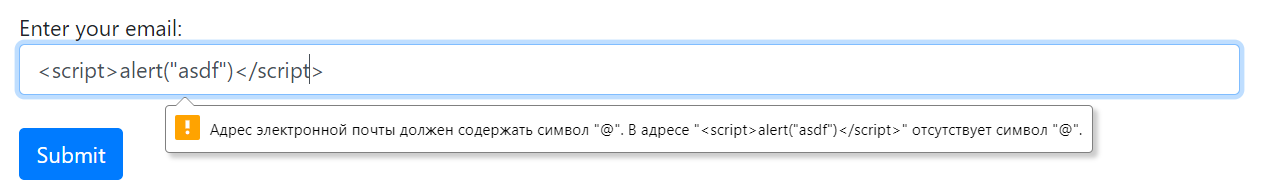
Следующим шагом добавим валидацию на стороне сервера

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.

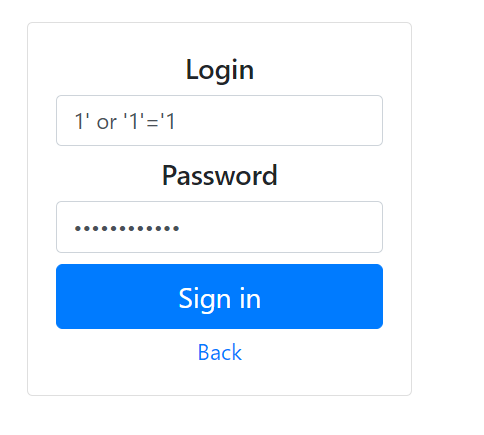


**Далее переходим на станицу входа**

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:

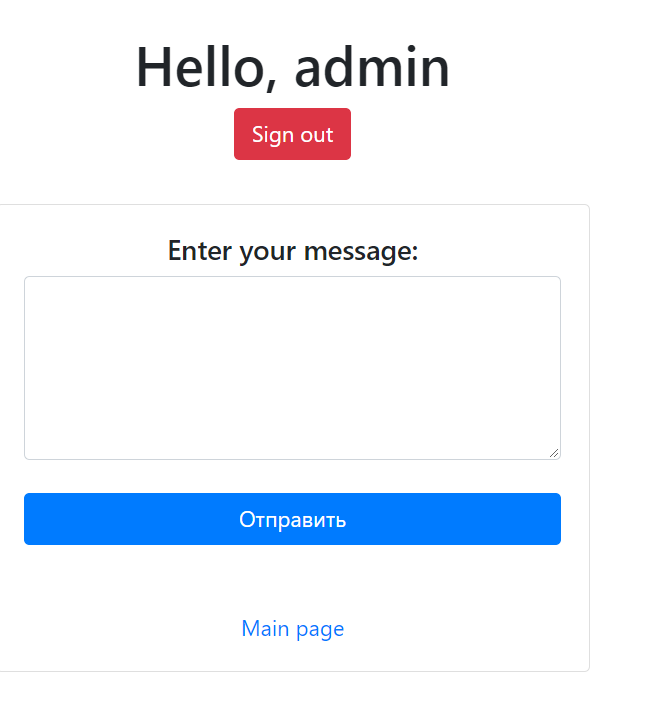
Вход с помощью SQL injection



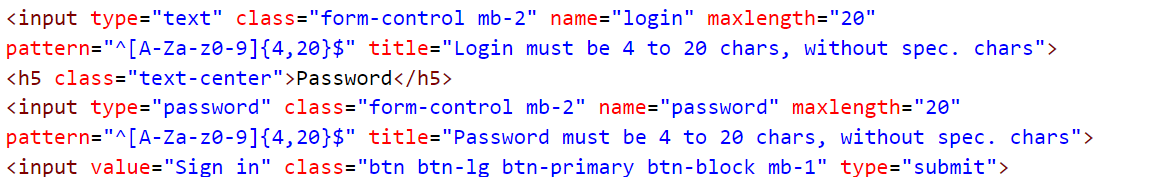
Вход с помощью отсутствия сессий



В обеих случаях получим доступ к сервису рассылки сообщений

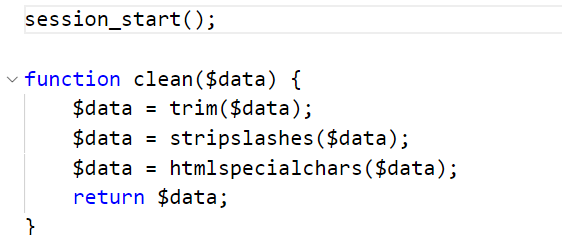


По аналогии с предыдущим пунктом исправляем наш html

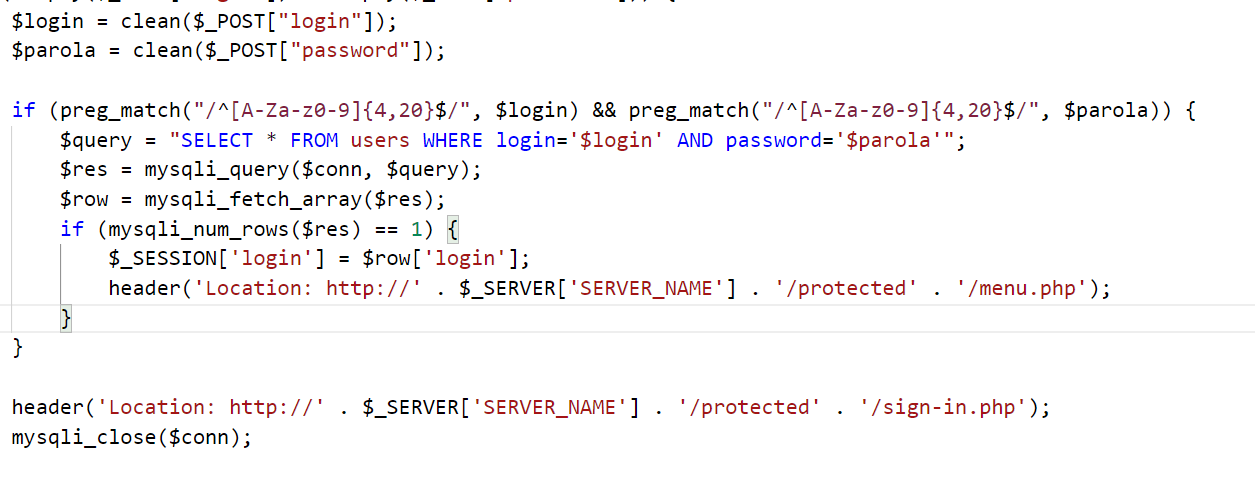


Далее добавим защиту на стороне сервера

Добавим ф-ию очистки и так же добавим поддержку сессий

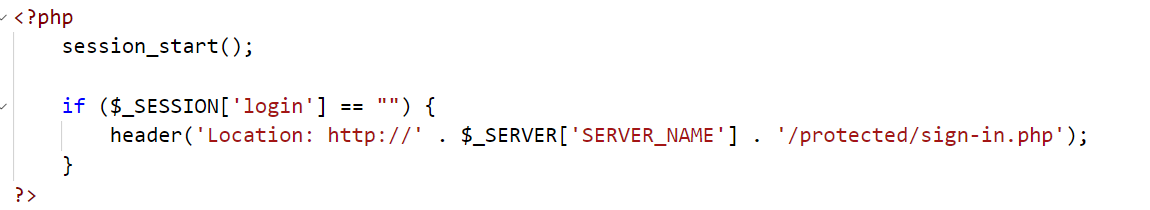


Во-первых очищаем наши input’ы, далее проверяем их на соответствие регулярному выражению, далее проверяем вернет ли наш запрос только одну строку и если результат положительный сохраняем переменную login в сессию и перенаправляем пользователя на страницу menu.



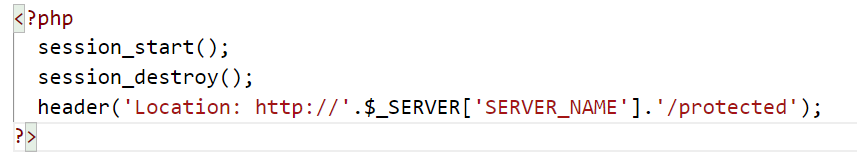
Уязвимость с переходом на menu.php осталась, но сейчас мы ее исправим

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

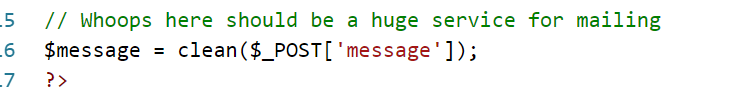


Аналогичный код добавим в файл send-message.php

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.



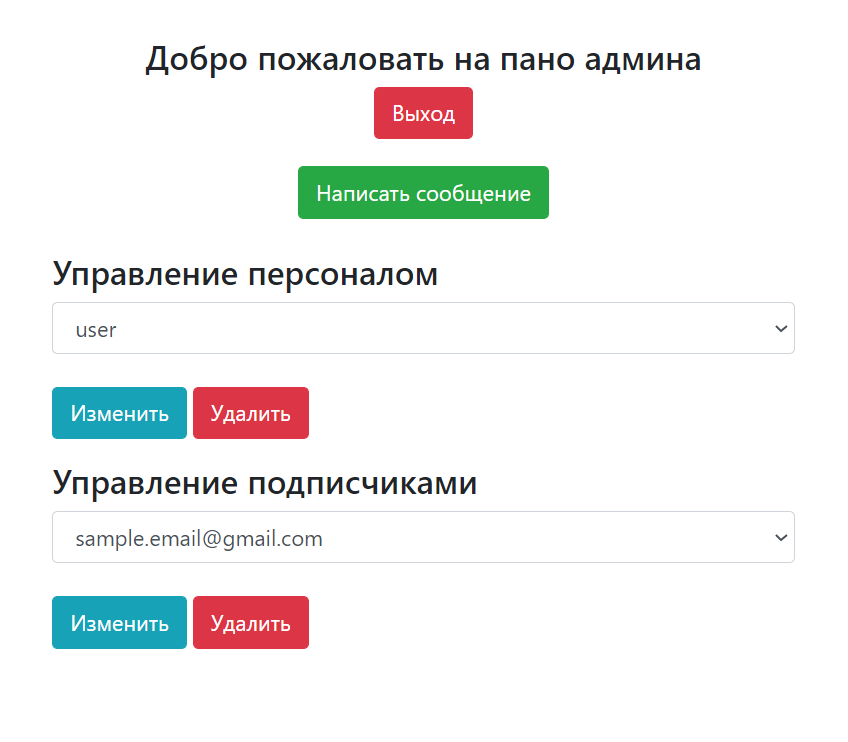
Далее добавим поддержку различных ролей в нашем приложении

Вначале модифицируем нашу базу данных

Добавим таблицу с ролями и свяжем ее с таблицей users



Далее создадим панель управления предназначенную для администратора (admin-menu.php)



Из данного меню он сможет перейти в обычное меню для отправки сообщений, управлять персоналом и подписчиками.

К данной панели обычный менеджер не имеет доступа.

Код необходимый для корректной работы и контроля сессии

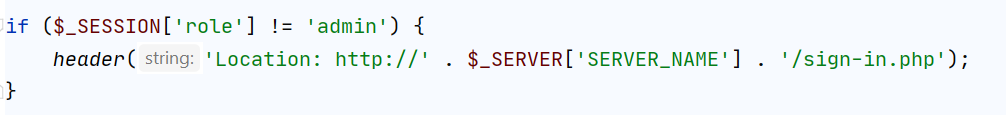
Вначале при регистрации будем там же добавлять роль пользователя



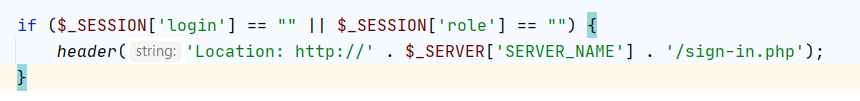
В зависимости от типа пользователя будем перенаправлять его на нужную панель управления

Теперь добавим проверку на принадлежность пользователя к нужной группе

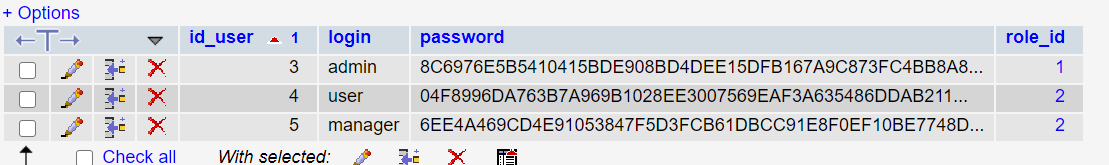
К admin-menu должен иметь доступ только администратор, добавим следующий код



К menu имеет доступ, как администратор так и обычный пользователь.



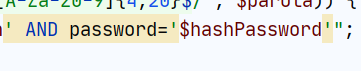
Далее займемся защитой паролей. В данный момент мы храним исходный пароль в базе, тем самым при утечке злоумышленник может получить доступ ко всем аккаунтам. Чтобы решить эту проблему будем хешировать пароль. Для начала изменим нашу БД



Далее изменим страницу аутентификации, будем хешировать пароль алгоритмом sha256



И далее при запросе в базу искать уже не хешу



Далее добавим журнализацию действий пользователей

Для начала напишем функцию, которая будет писать логи в специальный файл

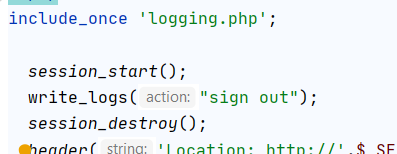


Теперь в нужных нам местах будем вызывать эту ф-ию, передавая действие которое производит пользователь

В месте входа в систему

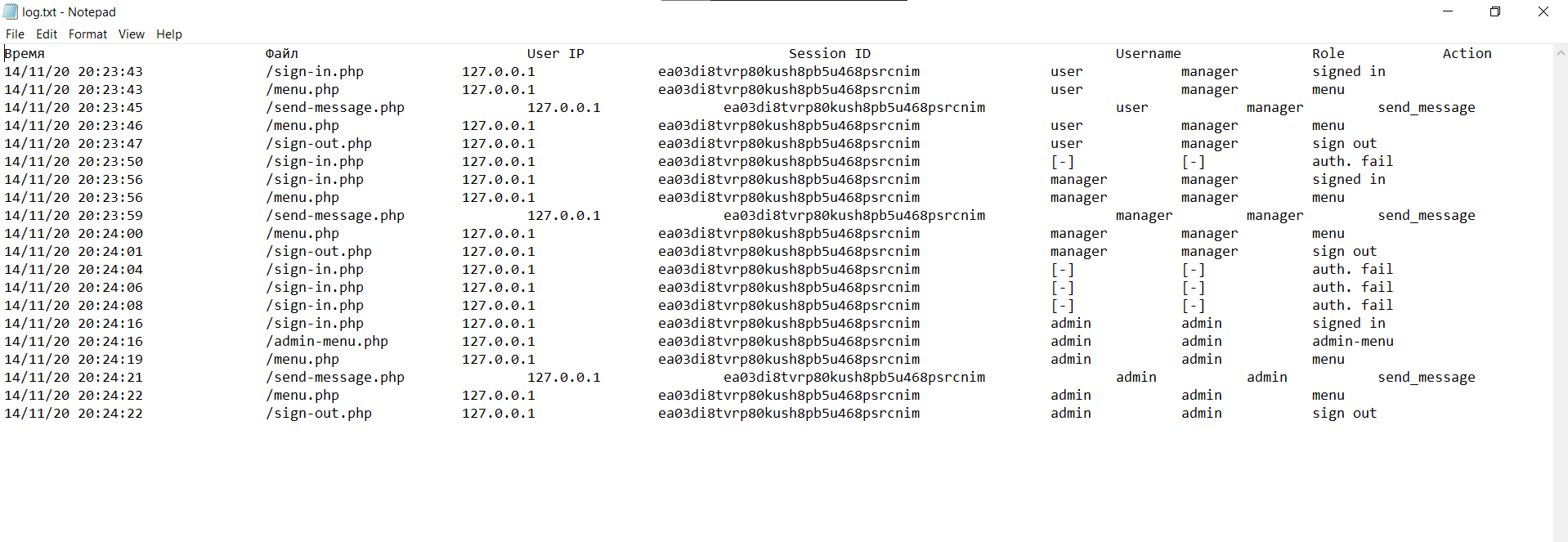


При выходе из системы



Аналогично добавим для входа на различные пано и отправку сообщений, при подписке на рассылку.

Пример log файла



Теперь займемся обработкой ошибок

В случае, если мы не сможем подключится к базе обработаем исключительную ситуацию, запишем лог, что именно пошло не так, а пользователя перенаправим на статичную страницу с ошибкой

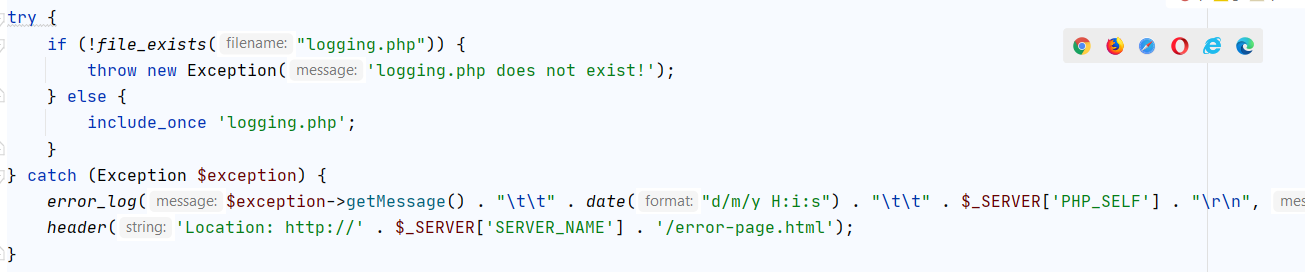
В нашем примере обработаем 2 ошибки – ошибка подключения к БД и ошибка подключения файла logging.php

Для этого добавим следующий код

Код файла connection.php



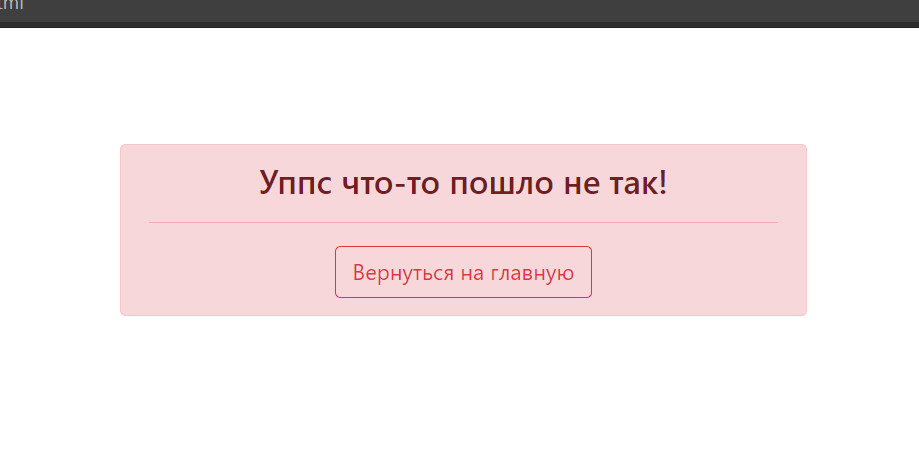
Код файлов использующих logging.php



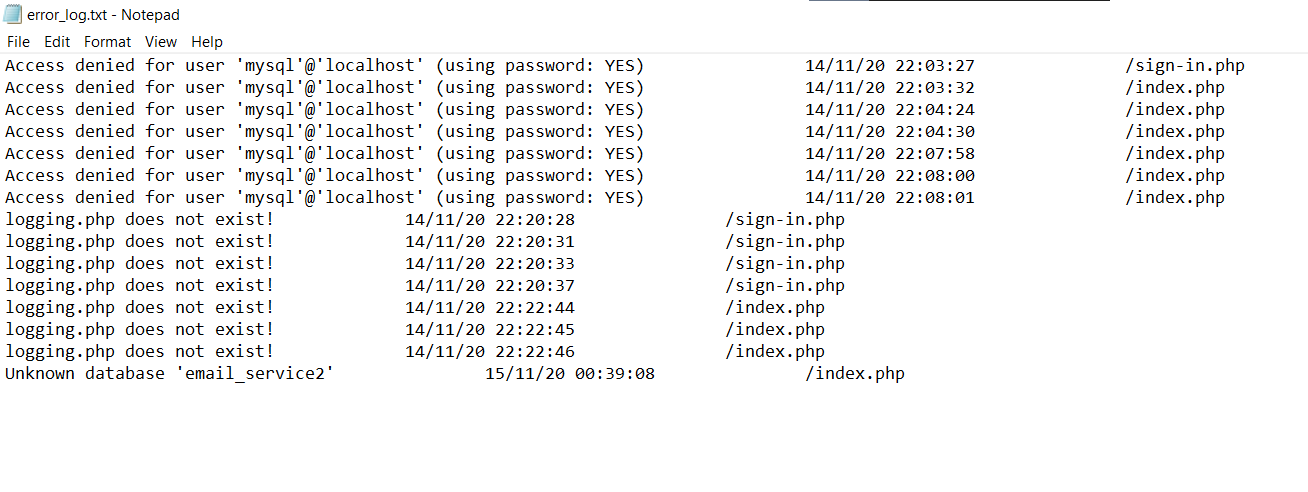
Демонстрация примера

В случае ошибки коннекта к базе

Пользователь увидит следующую страницу



И запишется соответствующий log



На скриншоте так же изображена ошибка при отсутствии файла logging.php

**Анализ проделанной работы:**

Изначально наше приложении имело следующие уязвимости:

* XSS
* SQL Injection
* Отсутствие контроля сессий
* Отсутствие ролей
* Незащищенные пароли
* Отсутствие логирования действий пользователя
* Отсутствие логирования ошибок

Благодаря чему пользователь мог обойти систему аутентификации двумя способами, а также внедрить зловредные скрипты в нашу базу данных. При краже данных из БД имел доступ ко всем аккаунтам, а так же все действия никак не журнализировались.

В ходе улучшения безопасности приложения мы использовали следующие приемы:

* Валидация данных на стороне клиента
* Валидация данных на стороне сервера
* Внедрение сессий
* Внедрение доступа по ролям
* Хеширование паролей
* Логирование действий пользователя
* Обработка исключений и их логирование

В результате мы получили сервис защищенный от вышеописанных атак.