|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное  учреждение высшего образования | | | | | | | | | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК** | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
| **Кафедра информационной безопасности** | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
| **О Т Ч Е Т** | | | | | | | | | | | | | | | | | | | | |
| о прохождении учебной практики (учебно-лабораторного практикума) | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | |  |  | Выполнил студент  гр. С8117-10.05.01ммзи | | | | | | | | |
|  |  |  | | | | | | Ставицкий М.Н. | | |
|  | | | | | | | | | |  |  | (подпись) | | | | | |  | | |
|  | | | | | | | | | | | | | | | | | | | | |
| Отчет защищен с оценкой | | | | | | | | | |  |  | Руководитель практики | | | | | | |  | |
|  | | | | | | | | | |  |  | Старший преподаватель кафедры информационной безопасности ШЕН | | | | | | | | |
|  | | | |  | С.С. Зотов | | | | |  |  |  | | | | |  | С.С. Зотов | | |
| (подпись) | | | |  | (И.О. Фамилия) | | | | |  |  | (подпись) | | | | |  | (И.О. Фамилия) | | |
| « | 26 | » | июня | | | | | | 2021 г. |  |  |  | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
| Регистрационный № | | | | | | | |  | |  |  | Практика пройдена в срок | | | | | | | | |
| « | 26 | » | июня | | | | | | 2021 г. |  |  | с | « | 22 | » | февраля | | | | 2021 г. |
|  | | | | | | | | | |  |  | по | « | 26 | » | июня | | | | 2021 г. |
|  | | | | | |  | Е.В. Третьяк | | |  |  | на предприятии | | | | | | | | |
| (подпись) | | | | | |  | (И.О. Фамилия) | | |  |  |
|  | | | | | | | | | |  |  | Кафедра информационной | | | | | | | | |
|  | | | | | | | | | |  |  | безопасности ШЕН ДВФУ | | | | | | | | |
|  | | | | | | | | | |  |  |  | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | | | | | | | | | |
| 2021 | | | | | | | | | | | | | | | | | | | | |

**Содержание**

Задание на практику 3

Введение 4

1.Описание основных уязвимостей веб-приложений 5

2. Реализация А1, А6, А7 8

Заключение 14

Список использованных источников 15

**Задание на практику**

- Проведение исследования в области безопасности веб-приложений.

- Написание отчета по практике о проделанной работе.

**Введение**

Учебная практика (учебно-лабораторный практикум) проходила на кафедре информационной безопасности ШЕН ДВФУ в период с 22 февраля 2021 года по 26 июня 2021 года.

Целью прохождения практики является приобретение практических и теоретических навыков по специальности, а также навыков оформления проведенного исследования в отчетной форме.

Задачи практики:

1. Исследовать нормативную базу по информационной и компьютерной безопасности в области уязвимостей веб-приложений.
2. Реализовать в команде 9 веб-приложений по уязвимостям OWASP TOP 10(A1 – A8), где каждой уязвимости соответствует одно веб-приложение, за исключением A1. Где случай A1 – делится на два: SQL и NOSQL.
3. Составить отчёт о проделанной работе.

## Описание основных уязвимостей веб-приложений

## А1. Injections

Возникают, когда ненадежные данные отправляются интерпретатору как часть команды или запроса. Вредоносные данные атакующего могут заставить интерпретатор сделать непреднамеренные команды или получить доступ к хранящимся данным без надлежащей авторизации.

Для реализации данной уязвимости создавалось два веб-приложения, в одном использовалась NOSQL инъекция, а во втором – SQL. Инъекция проводится через форму с логином и паролем на клиенте, где уже обрабатывается на сервере.

В случае успешного проведения атаки будет высвечен флаг.

## А2. Broken auth

Реализация функций, связанных с аутентификацией и управлением сеансами, часто делается неправильно, что дает возможность злоумышленникам взломать пароли, ключи или токены сеансов, а также использовать другие недостатки реализации, чтобы временно или навсегда получить идентификационные данные других пользователей.

Для реализации данной уязвимости создавалось веб-приложение, в котором в форме регистрации были низкие требования к паролю, а именно то, что он должен состоять только из 6 цифр. Таким образом в форме для входа можно перебрать пароль брутфорсом.

## А3. Xss (stored)

Данная уязвимость появляется, когда в генерируемые сервером страницы по той или иной причине внедряются пользовательские скрипты.

В данном типе уязвимостей на клиенте имеется форма для отзывов к продукту с возможностями комментирования, где поле комментария уязвимо для проведения XSS атаки. Пользователь отправляет данные с JavaScript-кодом, данные отправляются на сервер, где они не проверяются, не обрабатываются, и сохраняются в базу данных. Таким образом любой пользователь, зайдя на страницу с этим продуктом и отзывами к нему, получит отзыв с JS – кодом, который запустится на его клиенте.

## А4. Insecure dor

Клиент веб приложение состоит из трёх частей: форма входа, отображение данных пользователя и страница 404.

Случай заключается в том, что пользователь по URL может просмотреть данные всех людей, которые пользуются данным приложением, если же такого пользователя не существует, то показывается страница 404.

## А5. Miscofiguration

В данном случае пользователь при входе в веб-приложение видит строку ввода для поиска файлов в системе, воспользоваться которой может только после авторизации. Уязвимость реализовывается в данных, вводимых пользователем, которые могут быть интерпретированы как код, способный в дальнейшем запуститься на сервере.

## А6. Sensitive data exposure

В данном случае имеется два клиента и один сервер. Случай подразумевает клиент пользователя, которому необходимо войти в систему и клиент жертвы.

Клиент жертвы работает следующим образом: каждые 5 секунд делается POST запрос на авторизацию с валидными данными, уязвимость заключается в том, что данные передаются по HTTP.

Пользователю необходимо прослушать траффик внутри сети Wi-Fi и авторизироваться, узнав данные жертвы.

## А7. Access controls

Веб приложение показывает список пользователей, зарегистрированных в системе веб приложения с дезактивированной кнопкой удаления пользователя. Пользователю необходимо сделать запрос на удаление пользователя из списка. Уязвимость заключается в том, что на сервере не проверяются права пользователя, удаляющего пользователя из базы данных.

## А8. CSRF

В данном случае используется клиент пользователя, клиент злоумышленника и сервер. Клиент пользователя состоит из 2 частей: форма для входа с данными и новостная страница. Форма для входа необходима для того, чтобы понять, как идентифицирует сервер. Идентификация проходит по JWT, хранящиеся в cookie. Новостная страница состоит из следующих элементов: баланс пользователя, форма перевода средств, новость, комментарий и форма добавления комментария к новости. Форма перевода средств деактивировна, но из этой формы можно узнать какие данные нужны для перевода средств, а также API endpoint.

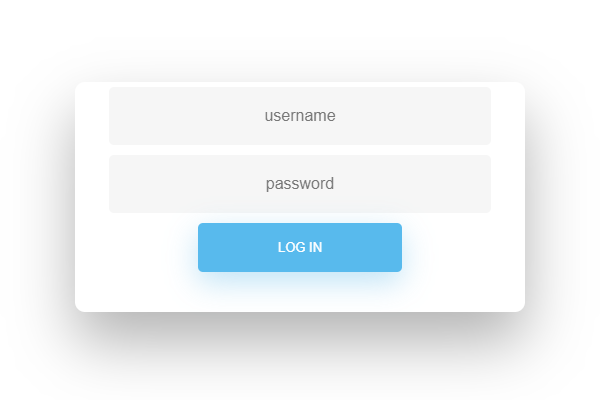
Задача пользователя состоим в том, чтобы создать клиент, который произведёт CSRF-атаку над жертвой, перешедшей по ссылке, а также в добавлении комментария со ссылкой.

## Реализация А1, А6, А7

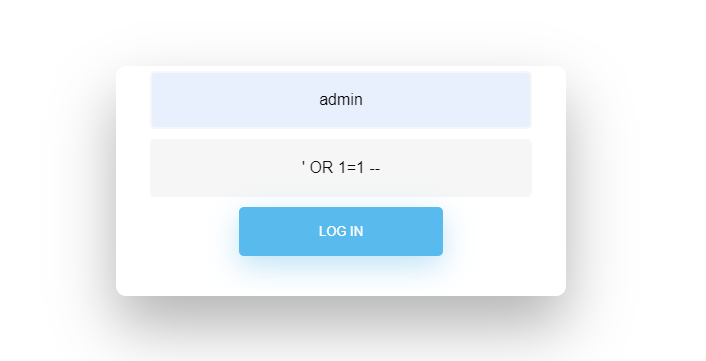
Моей частью работы была полноценная реализация уязвимости A1 с использованием SQL-инъекции, а также реализация API для уязвимостей A6 и А7.

## А1

В данной уязвимости пользователь заходит на страницу авторизации, где находится форма с вводом логина и пароля. Инъекция находится в поле для ввода пароля.



Так как все атаки нацелены на пользователя admin, то в поле username вводим admin, а в поле password нашу инъекцию.

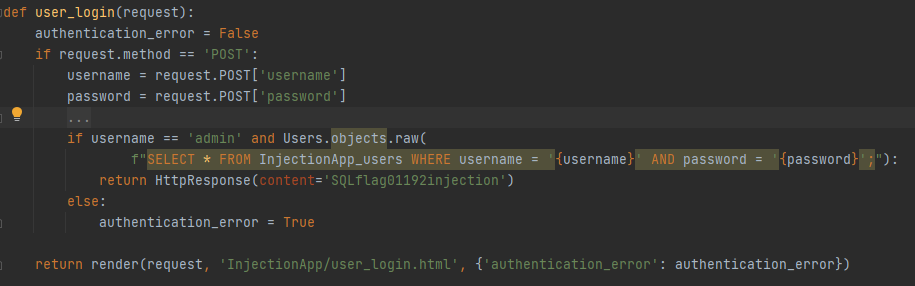


В случае успешной атаки появится флаг.



Так будет выглядить запрос к базе данных, при проведении инъекции

Реализация такой инъекции была в неправильном использовании чистого SQL запроса. Использовалось форматирование строк внутри SQL запроса, а также использовались скобки.

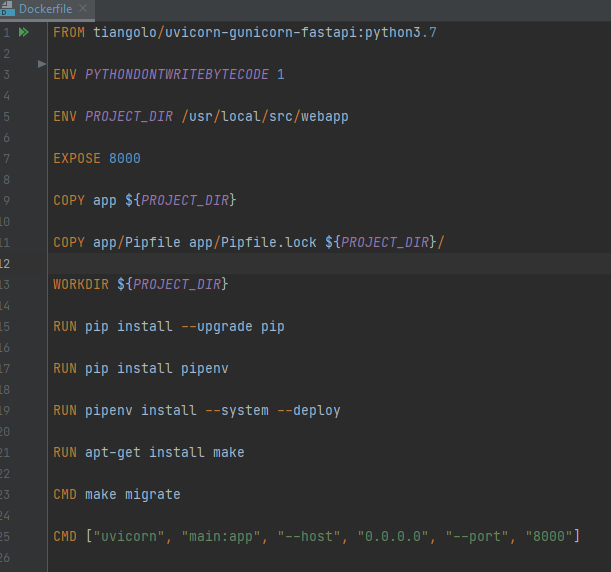


Избежать такой оплошности можно использовав SQL-запроса с передачей в него аргументов следующим образом:

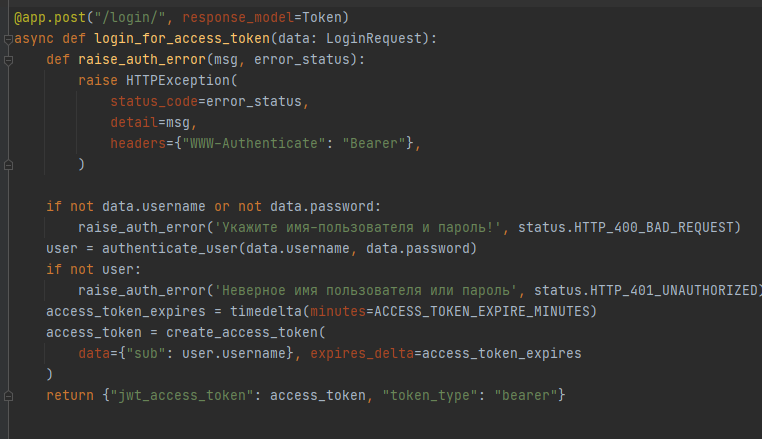


### А6

Ключевым моментом в реализации уязвимости является использование сервера uvicorn, который использует HTTP по умолчанию и тем самым дает возможность злоумышленнику перехватывать незашифрованные данные пользователя при авторизации.



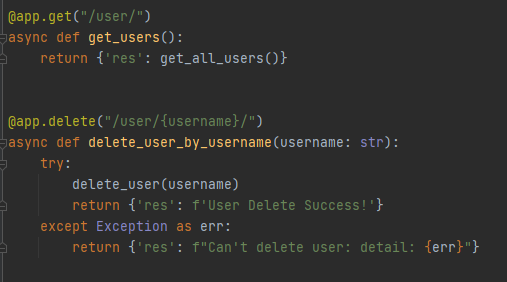
Авторизация на сервере реализована следующим образом:



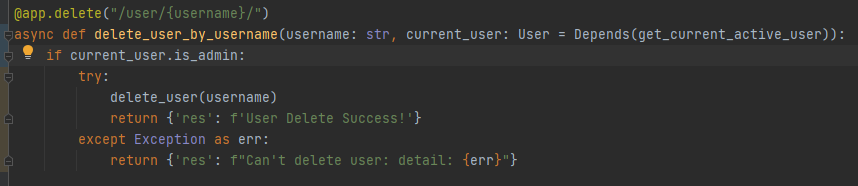
В login\_for\_access\_token передается data, в последствии проверяется наличие следующих атрибутов data: username, password. В случае если хотя бы одного из них нет вызывается ошибка 400. При наличии необходимых атрибутов, ищется пользователь из базы данных и в случае успешной авторизации выдается jwt токен. В противном случае вызывается ошибка 401.

## A7

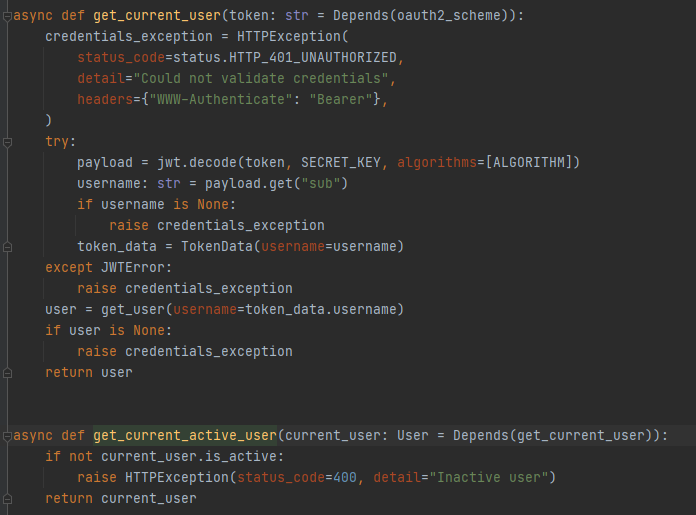
В API есть два endpoint ’а: получить список всех пользователей и удалить какого-то пользователя. За получение списка всех пользователей отвечает функция get\_users(), а за удаление - delete\_user\_by\_username(username). Уязвимость заключается в том, что функция для удаления пользователей не проверяет пользователя, который посылает запросы. Таким образом воспользоваться этой функцией может кто угодно.



Чтобы не допустить такой уязвимости, стоит передавать функции delete\_user\_by\_username переменную current\_user и использовать ее для условия проверки необходимых привилегий. Переменная current\_user объявляется как обязательная, с использованием класса Dependns из FastAPI. Тем самым не авторизованный пользователь не может делать запросы на этот endpoint.



Переменную current\_user можно получить из функии get\_current\_user.



Эта функция в свою очередь принимает jwt токен, декодирует его, достает из него необходимые данные (username и прочие) и пытается найти конкрентного пользователя в базе данных и в случае успеха, возращает объект user, иначе вызывает ошибку.

## Технологии

Для реализации использовался язык программирования Python.

Для хранения данных использовалась база данных SQLite.

На стороне сервера для реализации уязвимости A1 использовался Django, для A6 и A7 – FastAPI.

**Заключение**

Для достижения данной цели, в процессе работы над учебно-лабораторной практикой) познакомился с нормативной базой по информационной и компьютерной безопасности в области уязвимостей веб-приложений.

Также были изучены требования к написанию отчета по практике. В результате прохождения практики был составлен отчет по практике, соответствующий предъявленным требованиям.

В ходе прохождения практики все задачи были выполнены, а цель достигнута.

**Список используемых источников**

1. Performing raw SQL queries [Электронный ресурс]. – Режим доступа: <https://docs.djangoproject.com/en/3.2/topics/db/sql/>
2. OWASP Top 10 [Электронный ресурс]. – Режим доступа: <https://owasp.org/www-pdf-archive/OWASP_Top_10_-_2013.pdf>
3. FastAPI [Электронный ресурс]. – Режим доступа: <https://fastapi.tiangolo.com/>