**Содержание**:

[1. Введение 2](#__RefHeading___Toc779_1887482898)

[2. Краткие теоретические сведения 4](#__RefHeading___Toc781_1887482898)

[3. Блок-схема алгоритма 9](#__RefHeading___Toc783_1887482898)

[4. Демонстрация работы программы 11](#__RefHeading___Toc785_1887482898)

[5. Вывод 14](#__RefHeading___Toc787_1887482898)

# **Введение**

Цель лабораторной работы – познакомиться с концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз.

Научиться разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей. Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

# **Краткие теоретические сведения**

Для выполнения данной лабораторной работы было использовано много технологий, в частности:

Python3 как основной язык разработки.

Django как фрэймворк для создания web приложения

PeeWee – ORM для использования базы данных

PostgreSQL в качестве технологии для БД

Gunicorn – HTTP сервер, более безопасная и надежная альтернатива Django серверу

Heroku – web сервис для разворачивания приложения в глобальной сети

PyQt5 - фреймворк для написания клиент части под десктоп

Fbs – библиотека для упаковки и распространения клиента в виде исполняемого файла

Также для данной лабораторной работы была написана отдельная библиотека, которая может быть использована как и в Веб версии приложения, так и через консольный интерфейс. Приложение является таск трекер системой.

**2.1 Django приложение**

Django **-** [свободны](https://ru.wikipedia.org/wiki/Свободное_программное_обеспечение)й фреймворк для [веб-приложени](https://ru.wikipedia.org/wiki/Веб-приложение)й на языке [Python](https://ru.wikipedia.org/wiki/Python), использующий шаблон проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller). Проект поддерживается организацией [Django Software Foundation](https://ru.wikipedia.org/wiki/Django_Software_Foundation).

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, [Ruby on Rails](https://ru.wikipedia.org/wiki/Ruby_on_Rails)). Один из основных принципов фреймворка **-** [DR](https://ru.wikipedia.org/wiki/DRY)Y ([анг](https://ru.wikipedia.org/wiki/Английский_язык)л. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики [UR](https://ru.wikipedia.org/wiki/URL)L в Django конфигурируются явно при помощи [регулярных выражений](https://ru.wikipedia.org/wiki/Регулярные_выражения).

Для работы с [базой данны](https://ru.wikipedia.org/wiki/База_данных)х Django использует собственный [ORM](https://ru.wikipedia.org/wiki/ORM), в котором модель данных описывается классами Python, и по ней генерируется схема [базы данных](https://ru.wikipedia.org/wiki/БД). В примере данной лабораторной работы была использована ORM PeeWee – очень выразительная и легковесная ORM, использование которой – просто сказка.

Архитектура Django похожа на «[Модель-Представление-Контроллер](https://ru.wikipedia.org/wiki/Модель-Представление-Контроллер)» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление, а презентационная логика Представления реализуется в Django уровнем Шаблонов. Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Первоначальная разработка Django как средства для работы новостных ресурсов достаточно сильно отразилась на его архитектуре: он предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

В дистрибутив Django также включены приложения для системы комментариев, синдикации [RS](https://ru.wikipedia.org/wiki/RSS)S и [Atom](https://ru.wikipedia.org/wiki/Atom), «статических страниц» (которыми можно управлять без необходимости писать контроллеры и представления), перенаправления URL и другое.

Также Django предоставляет множество возможностей по защите, в частности средства по шифрованию отправляемых web form для защиты от злоумышленников, которые могут попытаться послать неверные данные от лица пользователя.

**2.2 Возможные атаки и защита от них**

XSS (межсайтовый скриптинг) атаки – один из самых распространенных вариантов атаки на веб-сервер или компьютер пользователя, суть которой заключается во встраивании определенных html элементов и javaScript кода в отданную сервером разметку, который выполнится на стороне клиента и может запустить какой-либо вирус или же произвести несанкционированное действие от имени пользователя на сервере. Принцип атаки приведен на рисунке 1.

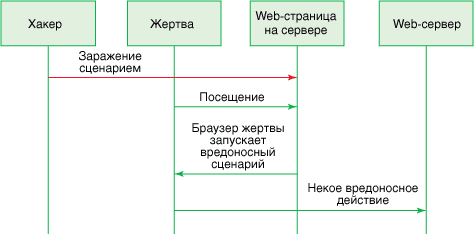


Рисунок 1. Принцип XSS атак

Если говорить немного подробнее, то можно выделить следующие виды XSS атак:

Хранимые (Постоянные)

Отражённые (Непостоянные)

Пример постоянных:

- Введённое злоумышленником специально сформированное сообщение в гостевую книгу (комментарий, сообщение форума, профиль) которое сохраняется на сервере, загружается с сервера каждый раз, когда пользователи запрашивают отображение этой страницы.

- Злоумышленник получил доступ к данным сервера, например, через SQL инъекцию (смешение двух типов атак – SQL injections и XSS атак), и внедрил в выдаваемые пользователю данные злонамеренный JavaScript код.

Пример непостоянных:

- На сайте присутствует поиск, который вместе с результатами поиска показывает что-то вроде «Вы искали: [строка поиска]», при этом данные не фильтруются должным образом. Поскольку такая страница отображается только для того, у кого есть ссылка на неё, то пока злоумышленник не отправит ссылку другим пользователям сайта, атака не сработает. Вместо отправки ссылки жертве, можно использовать размещение злонамеренного скрипта на нейтральном сайте, который посещает жертва.

Для предотвращения XSS-атак приложение должно шифровать выходные данные страницы перед передачей их конечному пользователю. При шифровании выходных данных разметка HTML заменяется альтернативными представлениями - объектами. Браузер отображает эти объекты, но не запускает их. Например, *<script>* преобразуется в *&lt;script&gt;.*

Когда браузер встречает объекты, они преобразуются обратно в HTML и распечатываются, но они не запускаются. Например, если атакующий внедрит в переменное поле Web-страницы сервера строку *<script>alert("вы атакованы")</script>*, то при использовании описанной стратегии сервер возвратит строку *&lt;script&gt;alert("вы атакованы")&lt;/script&gt;.*

Когда браузер загрузит зашифрованный сценарий, он преобразует его к виду *<script>alert("вы атакованы")</script>* и отобразит сценарий в составе Web-страницы, но не запустит его.

В случае приложения реализованного в данной лабораторной работе, защита была реализована прямо на сервере gunicorn, который благодаря выше описанной процедуре не позволяет запустить вредоносный скрипт на стороне пользователя.

SQL инъекции – также один из очень распространенных видов атак на приватные пользовательские данные или же непосредственно на устройство пользователя, а также и на сам сервер с целью уничтожения или распространения данных. Суть заключается в том, что некоторые веб формы могут быть связаны с SQL запросами и если пользователь введет в такую форму SQL запрос, он может исполниться прямо на стороне сервера и принести очень много проблем.

В реализации приложения этой лабораторной работы была использована ORM PeeWee. Сама по себе технология ORM позволяет обернуть все SQL запросы в классы и функции языка программирования, тем самым не давая простым SQL запросам исполняться вовсе.

Также одной из атак является web crawling засекреченной информации. В частности такое можно увидеть при web crawling'е, когда по различным сайтам ходят программы, которые собирают различные данные, иногда к которым доступа у них не должно быть. Или же злоумышленник может ввести определенную ссылку исходя из логических именований и получить доступ к редактированию или эксплуатации чужих данных.

Основной защитой от такого вида атаки является постоянная проверка аутентификации и подлинности текущего пользователя на данную страницу. В примере 2 лабораторной работы при реализации Kerberos протокола мы обеспечивали такую защиту. Также одним из видом защиты может стать генерация уникальных ссылок и соответственно вероятность перехода по такой ссылке может быть крайне мала (если брать Sha-256 для примера, то вероятность коллизии будет равна 1/(16 ^ 256), т е 5.56е-309).

DoS атаки также популярный вид атак, который заключается в перегрузке сети запросами (рисунок 2). По сути это  [хакерская атака](https://ru.wikipedia.org/wiki/Хакерская_атака) на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён. Отказ «вражеской» системы может быть и шагом к овладению системой (если в нештатной ситуации ПО выдаёт какую-либо критическую информацию — например, версию, часть программного кода и т. д.). Но чаще это мера экономического давления: потеря простой службы, приносящей доход, счета от провайдера и меры по уходу от атаки ощутимо бьют «цель» по карману. В настоящее время DoS и DDoS-атаки наиболее популярны, так как позволяют довести до отказа практически любую систему, не оставляя юридически значимых улик.

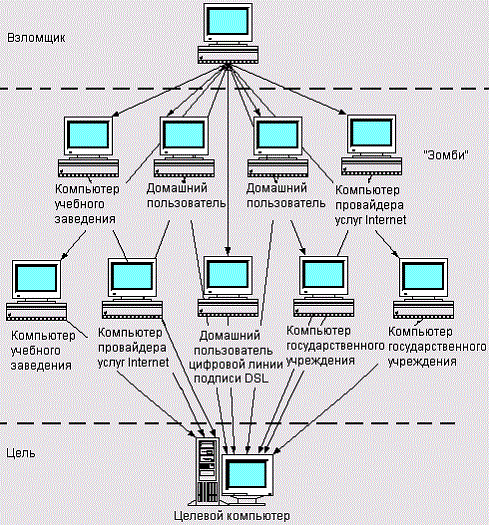


Рисунок 2. Принцип DoS атаки

Защититься от DoS атаки является очень трудной задачей. Для защиты от сетевых атак применяется ряд фильтров, подключенных к интернет-каналу с большой пропускной способностью. Фильтры действуют таким образом, что последовательно анализируют проходящий [трафик](https://ru.wikipedia.org/wiki/Сетевой_трафик), выявляя нестандартную сетевую активность и ошибки. В число анализируемых шаблонов нестандартного трафика входят все известные на сегодняшний день методы атак, в том числе реализуемые и при помощи распределённых бот-сетей. Фильтры могут реализовываться как на уровне [маршрутизаторов](https://ru.wikipedia.org/wiki/Маршрутизатор), управляемых [свичей](https://ru.wikipedia.org/wiki/Сетевой_коммутатор), так и специализированными аппаратными средствами. Также возможна установка задержек на запросы от пользовательских машин (такой способ и был реализован в лабораторной работе), однако при большом количестве устройств данная защита начинает заметно проседать.

**2.3 Инсталлятор**

Для постановки клиентской части приложения будет использоваться специальная программа-инсталлятор, которая позволяет установить программу на десктоп. Для создания инсталлятора используется библиотека fbs, которая требует создания проекта, создание кода внутри него и по команде fbs installer библиотека может создавать инсталляторы для приложений. Однако данная команда платформозависимая и в случае с windows нам требуется софт NSIS. NSIS - система создания установочных программ для [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) с открытыми исходными кодами, созданная компанией [Nullsoft](https://ru.wikipedia.org/wiki/Nullsoft), основанной создателями [Winamp](https://ru.wikipedia.org/wiki/Winamp). NSIS создан как альтернатива [InstallShield](https://ru.wikipedia.org/wiki/InstallShield), предназначенного для коммерческих продуктов. Также данный инсталлятор создает автора и идентифицирует его таким образом, чтобы никто другой, кто будет распространять приложение не мог бы подделать авторство и от нашего имени распространять софт. Также в NSIS возможны скрипты, которые могут кастомизировать сам инсталлятор и делать установку более гибкой.

# **Блок-схема алгоритма**

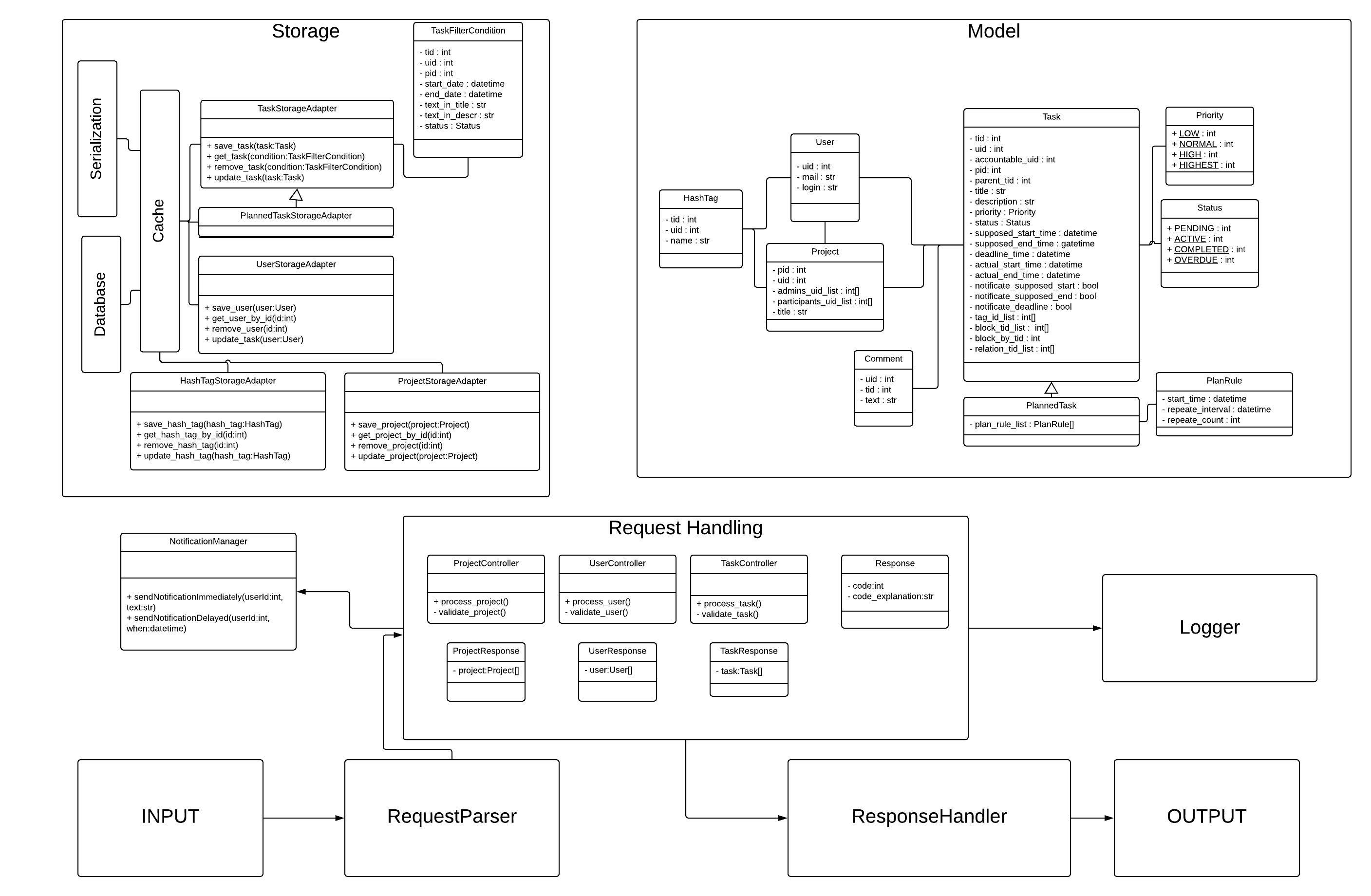


Рисунок 3.1. Блок-схема алгоритма программы

# **Демонстрация работы программы**

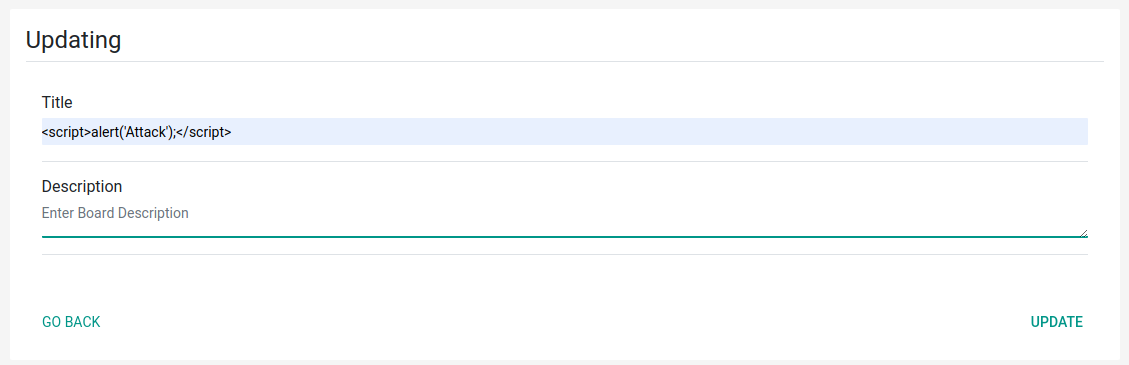


Рисунок 4.1. XSS атака.



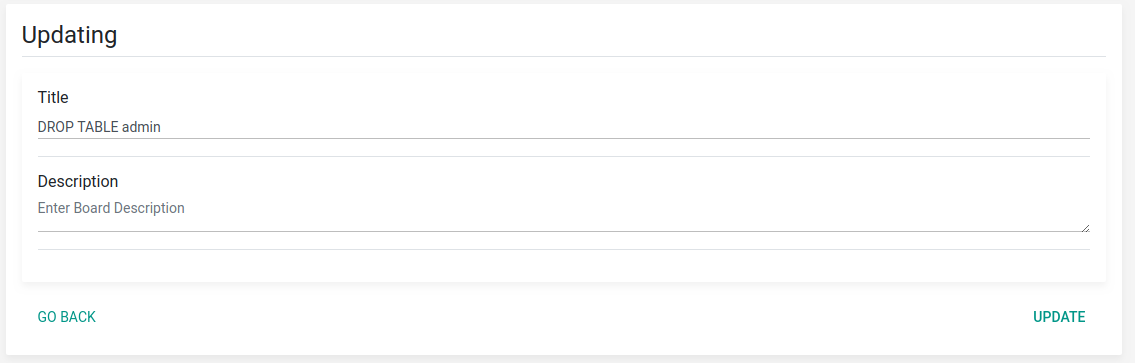
Рисунок 4.2. Подавленая XSS атака.

Рисунок 4.3. SQL-инъекция

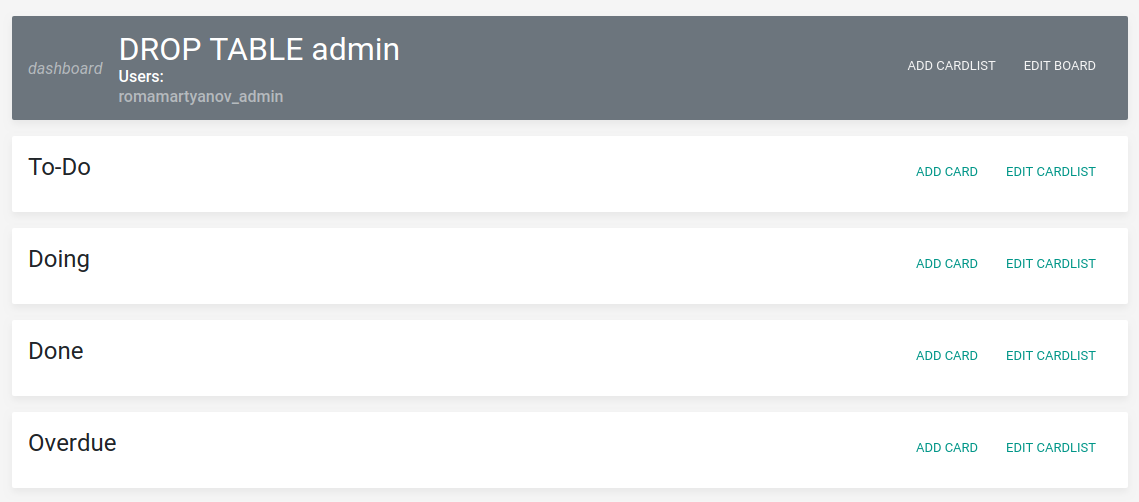


Рисунок 4.4. Принцип минимизации привилегий

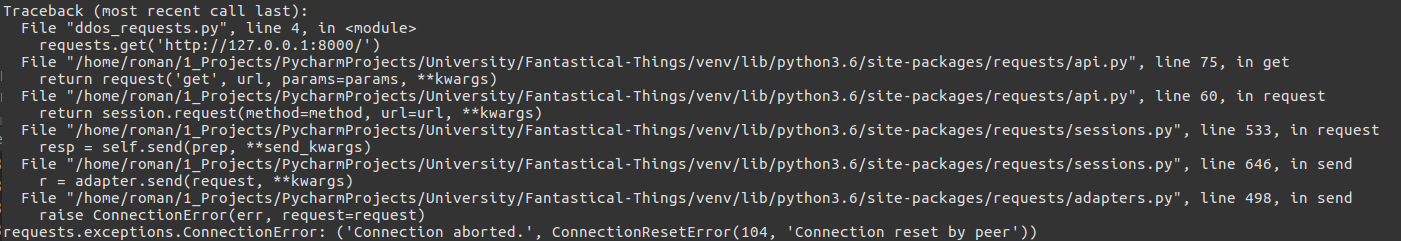


Рисунок 4.5. Подавленная DdoS атака

# **Вывод**

В ходе выполнения лабораторной работы были реализована концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз. Было разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей. Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.