|  |
| --- |
| МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ города Костромы «Лицей № 17» |
| Индивидуальный итоговый проект |
| Создание ПО для системного инжиниринга |
|  |
| **Выполнил: Белоконь Д, учащийся 10а класса Руководитель: Исаева Мария Владимировна, учитель информатики** |

|  |
| --- |
| Кострома, 2020 |

Оглавление

[Введение 3](#_Toc59683097)

[Концепция 3](#_Toc59683098)

[Актуальность 3](#_Toc59683099)

[Польза 3](#_Toc59683100)

[Цель 3](#_Toc59683101)

[Задачи 3](#_Toc59683102)

[Глава I. Теоретическая часть. 5](#_Toc59683103)

[Взаимодействие устройств 5](#_Toc59683104)

[Виды уязвимостей 5](#_Toc59683105)

[Взаимодействие с пользователем 6](#_Toc59683106)

[Выбор языка программирования 6](#_Toc59683107)

[Глава II. Реализация. 7](#_Toc59683108)

[Планирование 7](#_Toc59683109)

[Структура файлов 7](#_Toc59683110)

[Написание кода 7](#_Toc59683111)

[Классы 7](#_Toc59683112)

[Консоль взаимодействия 7](#_Toc59683113)

[Уязвимости 8](#_Toc59683114)

[Подпрограмма анализа 8](#_Toc59683115)

[Глава III. Результаты. 9](#_Toc59683116)

[Главные сложности 9](#_Toc59683117)

[Работоспособность модели 9](#_Toc59683118)

[Анализ уязвимостей 9](#_Toc59683119)

[Заключение 10](#_Toc59683120)

# Введение

## Концепция

Идея проекта заключается в создании способной анализировать возможные угрозы программы по заданной пользователем модели. Модель создаётся с помощью вводимых в консоль приложения команд. После этого запускается проходящая по прописанному алгоритму подпрограмма. Она собирает все учтённые уязвимости и выводит результаты на экран. Плюсом должна являться возможность легко добавлять новый код к уже существующему проекту.

## Актуальность

На данный момент существует большое количество приложений для визуализации устройства локальных сетей. Чаще всего их функционал заканчивается на рисовании объектов и связей между ними. Примерами можно обозначить CADE, Concept Draw Pro, Dia, Diagram Designer, eDraw Max, LanFlow и другие. Встречаются программы с дополнительным функционалом, а именно: NetProbe сам анализирует сеть и составляет схемы; Network Notepad в дополнение к автоматическому анализу способен удалённо подключаться к устройствам и проверять их доступность. Но я не нашёл способных анализировать уязвимости моделей приложений. Поэтому я решил реализовать прототип такого проекта самостоятельно.

## Польза

Данная идея способна в дальнейшем развиться в полезный продукт. При полной реализации задуманного и добавлении из сторонних приложений существующего функционала получится полноценная рабочая программа для использования в реальных целях.

Лично для меня этот проект является способом закрепить уже полученные навыки и приобрести новые. Работа включает в себя нужные мне в запланированном будущем знания.

## Цель

Разработка и реализация среды для создания схем локальных сетей и дополнительной программы для анализа уязвимостей данных моделей.

## Задачи

Полноценный продукт включает в себя разработку комплексной системы с большим количеством функций и графическим интерфейсом. Также он подразумевает наличие различных настроек и удобного добавления новых модулей. Я выделил следующие этапы:

* Создание базовых представляющих собой реальные устройства классов.
* Написание функционала для манипуляций этими классами:
  + Создание и удаление объектов.
  + Их настройка.
  + Установление связей между ними.
* Ввод возможности хранения моделей в базе данных.
* Разработка обобщённых видов уязвимостей.
* Добавление подпрограммы для анализа создаваемых моделей.
* Перевод консоли в графический интерфейс:
  + Создание основного интерфейса программы.
  + Определение к каждому базовому классу отдельной иконки.
  + Добавление в базу данных полей для хранения координат объектов.
* Создание функции добавления собственных классов.
* Дальнейшее дополнение функционала и изменения.

Для своего проекта я выбрал первые пять основных задач и раскрыл их:

* Разработка нескольких классов:
  + Написание «модулей» связи (Bluetooth, USB, Remote, LAN).
  + Объединение в классах различных сочетаний данных модулей (Computer, Phone, Server, Router, Laptop).
* Создание функционала для настройки объектов и связей между ними.
* Сохранение моделей в базе данных и получения их оттуда.
* Описание передачи пакетов данных между устройствами.
* Ввод 2-3 уязвимостей и возможности их добавления экземплярам классов.
* Написание анализирующего 2 или 3 ситуации обработчика моделей.

# Глава I. Теоретическая часть.

Для реализации задуманной идеи необходимы знания в области коммуникации устройств в сети, основах компьютерной безопасности и программировании. Перед написанием программы необходимо изучить некоторые основы первых двух дисциплин.

## Взаимодействие устройств

Устройства в сети постоянно обмениваются данными между собой. Это происходит по стандартизированным правилам. Основным таким стандартом принято считать протокол обмена данными TCP/IP[[1]](#footnote-1).

Основой данного механизма является присвоение каждому устройству адреса. Тогда любое сообщение разбивается на небольшие части и отправляется к цели с пометкой конечного устройства. Оно проходит несколько этапов преобразований и обработок. Эти этапы прописаны в общепризнанной модели OSI[[2]](#footnote-2).

Данная модель включает в себя 7 уровней: от физического до программного. На каждом уровне существуют различные правила взаимодействия. TCP/IP работает на 4 из них, а именно: канальный, сетевой, транспортный уровни и уровень приложений.

Сначала данные из приложений разбиваются на отдельные части. Затем они передаются на транспортный уровень. Он отвечает за надёжную доставку данных и их целостность. Именно на нём работает Transport Control Protocol – TCP. Далее взаимодействие переходит на сетевой уровень. Здесь определяются адреса получателя и отправителя. Это происходит с помощью иного протокола – IP (Internet Protocol). После сборки информации для отправки пакет[[3]](#footnote-3) отправляется по проводам к пункту назначения. Там уже происходит его сборка и передача в нужные приложения.

В моём проекте нет необходимости описывать работу систем так подробно. В своей программе для каждого устройства пользователем будет задаваться имя. Оно заменяет реальный IP-адрес. Далее переходящий от устройства к устройству пакет добавляет в свой путь уже пройденные объекты. Благодаря этому пакет никогда не попадёт в уже пройденное устройство. Это сильно сокращает работу алгоритма.

## Виды уязвимостей

Устройства не лишены недостатков. Разработчики постоянно закрывают случайные пробелы в безопасности своих продуктов. Но в дополнение к этому также существует человеческий фактор. Его не исправить и вся ответственность лежит на людях. Эти и пользуются злоумышленники.

Существуют классические позволяющие удалённо контролировать и получать данные уязвимости. К ним относятся: различные инъекции, подделка запросов, ошибки памяти, открытый удалённый доступ и другие. Инъекции – это исполнение команд или запросов злоумышленника. Подделка запросов подразумевает отправку данных на сервер злоумышленника вместо сервера сайта. Ошибки памяти включают в себя ряд уязвимостей, которые могут привести к завершению работы программы.

Для своего проекта я решил выбрать такие уязвимости, как: незащищённость трафика, удалённый доступ, открытые базы данных. В своём проекте я сделал из них отдельные классы, которые добавлялись к моделям устройств. На основе их в подпрограмме я анализировал случаи перехвата трафика и удалённого доступа к данным серверов.

## Взаимодействие с пользователем

В программах существует два способа взаимодействия с пользователями. Первый – через консоль. Второй – через графику. Более простым и понятным для пользователя является присутствие хотя бы минимальной графики в программе. Но в моём случае это понесёт добавление достаточно большого спектра задач: от хранения координат расположенных пользователем устройств до рисования иконок всех видов классов. Поэтому в моём проекте я использовал средство ввода команд в консоль, которые затем шли на обработку.

## Выбор языка программирования

Для выбора языка программирования необходимо установить для себя определённые особенности реализации проекта. Для меня важны скорость разработки, возможность использовать как ООП, так и функциональное программирование и обычные скрипты. Поэтому выбором стал Python. Также он имеет большое количество подключаемых дополнительных модулей. Они позволяют легче выполнить поставленные задачи.

Для работы с базами данных я использовал такой модуль Python как SQLite3. Это перенесённая в Python реализация системы управления базами данных SQLite.

# Глава II. Реализация.

## Планирование

Среди своих задач я уже чётко выделил необходимые этапы для достижения поставленной цели. Во время всей разработки я придерживался их. Это достаточно облегчило работу.

## Структура файлов

Любой проект начинается с «main.py». Он содержит код основной программы и запускается в первую очередь. В нём я хранил все функции настройки, создания, инициализации программы, выхода из неё и другие. Для классов я создал отдельную папку. В ней каждая директория является вкладкой с файлом «\_\_init\_\_.py». Это помогло не путаться в коде программы и легко импортировать необходимые классы, когда это было необходимо.[[4]](#footnote-4)

## Написание кода

## Классы

Первым делом я реализовал 4 класса связи между устройствами: Bluetooth, USB, Remote и LAN. Все эти обозначения известны за исключением Remote. Под Remote подразумевается удалённый обмен данными.

После этого я выделил 5 основных видов устройств, которые необходимо реализовать: Computer, Laptop, Server, Router, Phone. Computer включает в себя модули LAN и USB; Laptop – Remote, Bluetooth, LAN, USB; Server – LAN, USB; Router – LAN, Remote, Phone – Remote, USB, Bluetooth. Классы сервера и компьютера имеют одинаковый набор модулей. Но я выделил их в отдельные классы для упрощённой работы анализатора. Ему необходимо определить нахождение данных на сервере под угрозой. Для этого при сканировании легче сразу отделить тип сервера от обычного компьютера.

Сначала в приложении была обязательная настройка количества максимальных подключений для устройств. Без установления лимита нельзя было добавить хотя бы одно соединение. Но затем я отказался от данной задумки и разрешил пользователям по своему усмотрению ставить ограничение на количество подключений.

## Консоль взаимодействия

В главном файле я сделал лёгкий обработчик команд. Он определяет функцию назначения для введённой пользователем команды. Далее последовала реализация самих функций. В их число изначально входили: добавление устройств, удаление устройств, настройка устройств, добавление и удаление разрешений для удалённого доступа, вывод всех устройств и их подключений, вывод лимитов на подключения.

Настройка устройств включает в себя изменение названия объекта, лимитов подключений, имеющихся уязвимостей.

## Уязвимости

Я выделил 3 вида необходимых мне уязвимостей: PHP-инъекция, SQL-инъекция и Remote Control. PHP-инъекция добавляется к предполагаемому web-серверу. SQL-инъекция – к SQL-серверу. Remote Control могут иметь любые устройства. Последняя уязвимость имеет функцию управления доступом к устройству со стороны других.

Инъекции не означают возможность исполнения команд. Это маркеры для анализатора. По ним он определяет вид сервера и выводит в зависимости от этого определённый результат.

Сначала я хотел добавить уязвимостям функцию «do». Она должна была реализовывать их главные особенности. Но данный код не пригодился в проекте – анализатор сам просматривает все данные.

## Подпрограмма анализа

В конце я создал предлагающий пользователю ввести имя устройства скрипт. Введённое устройство обозначается как атакующее. От него происходит дальнейший анализ всей сети.

Сначала я хотел написать что-то более обширное и общее. Но спустя некоторое время для лёгкости решения реализовал анализ по определённой схеме. Она может дополняться в будущем. [[5]](#footnote-5)

Сначала программа по цепочке анализирует подключённые по LAN и Remote устройства и составляет список. Затем она перебирает каждое устройство и оценивает его на возможность получения прямого удалённого доступа или удалённого доступа через другие устройства. После этого проверяется наличие среди данных устройств серверов или нахождение в одном ограниченном маршрутизаторами блоке сети рядом с различными SQL- и PHP-серверами одного из подконтрольных устройств. При нахождении в консоль выводится предупреждение.

# Глава III. Результаты.

## Главные сложности

Главными сложностями проекта являлись сами реализуемые модели. При возникновении ошибки в передаче данных приходилось достаточно долго смотреть итерации и прохождение разных путей пакетом. Кроме того эти модели надо было сохранять в базах данных и доставать их оттуда. Основной трудностью в этом были пробелы в полях или хранение самих экземпляров классов и необходимость записать в таблицу именно буквенные значения.

В итоге это привело к большому количеству переборов разных типов устройств, что значительно увеличило размеры кода.

## Работоспособность модели

В программе присутствует возможность составлять модели устройств локальных сетей. Это является тем минимумом, что присутствует во всех программах данного назначения за исключением наличия графики. Модели удобно сохраняются в базу данных и вынимаются оттуда при инициализации.

Сами базы данных необходимо реструктурировать. Они хранят большое количество пустых полей. Это удобно только для чтения человеком – с точки зрения памяти и скорости работы программы это достаточно плохая реализация.

## Анализ уязвимостей

Анализ проходит по чётко заданной схеме. По этой причине он всегда найдёт заданную в коде ситуацию. Но это вынуждает работать над каждым случаем наличия уязвимости отдельно, что заставляет пересмотреть подход к реализации данной функции.

# Заключение

Анализируя полученные результаты, я могу сказать, что проект реализован на необходимом минимуме. Проект уже способен выполнять главную задачу по хранению и отображению связей между устройствами и проверки на наличие нарушений безопасности.

Графическая оболочка для данного приложения подходит в качестве темы для отдельного проекта и требует длительной и тщательной проработки с изменением самого кода.

Дополнительная подпрограмма для анализа моделей задаёт вектор развития данной идеи в будущем и является её главной отличительной особенностью. Над ней придётся ещё много работать. Тем не менее, есть более универсальное и простое решение, чем то, которое я начал развивать. Хорошей идеей в следующей версии будет переделывание чёткой программы анализатора в некое подобие непрерывного тестировщика всех устройств. Это сделает проект более логичным с точки зрения трат на развитие и фиксирование всех возможных вариантов уязвимостей.

Иллюстрации, демонстрирующие результаты находятся в приложении к проекту. Видеозаписи и код можно найти в репозитории по ссылке: <https://github.com/Denittka/SecondSchoolProject>

# Источники

Документация SQLite3 для Python:

<https://docs.python.org/3/library/sqlite3.html>

Ресурс IT-BLACK:

<https://it-black.ru/vidy-kompyuternykh-uyazvimostey/>

Ресурс CIT Forum:

<http://citforum.ru/nets/services/services02.shtml>

Интернет-учебник по языку запросов SQL:

http://moonexcel.com.ua/уроки-sql1-язык-основные-понятия\_ru

Компьютерные сети, Э. Таненбаум, Д. Уэзеролл

Python Programming: A Step By Step Guide For Beginners», Брайан Дженкинс

1. TCP/IP – Transfer Control Protocol/Internet Protocol. Протокол управления передачей/Интернет протокол. [↑](#footnote-ref-1)
2. OSI – The Open Systems Interconnection model. Открытая модель взаимодействия систем. Изображение со всеми уровнями находится в приложении. [↑](#footnote-ref-2)
3. Пакет – передаваемый по сети блок данных. [↑](#footnote-ref-3)
4. Структура файлов прикреплена в приложении. [↑](#footnote-ref-4)
5. Алгоритм работы анализатора в общем представлении прикреплён в приложении. [↑](#footnote-ref-5)