**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики - процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Системное программирование в Linux»**

**на тему «Обнаружение и блокировка подозрительного сетевого трафика»**

**Студент гр. 23Б15-пу**

**Кубякин Н.А.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

***Оглавление***

[1. Цель работы 3](#_Toc178797423)

[2. Описание задачи (формализация задачи) 3](#_Toc178797424)

[3. Теоретическая часть 4](#_Toc178797431)

[4. Основные шаги программы 4](#_Toc178797431)

[5. Описание программы 4](#_Toc178797442)

[6. Рекомендации пользователя 5](#_Toc178797443)

[7. Рекомендации программиста 6](#_Toc178797444)

[8. Исходный код программы 6](#_Toc178797445)

[9. Контрольный пример 6](#_Toc178797447)

[10. Вывод 7](#_Toc178797447)

[11. Источники 7](#_Toc178797448)

# Цель работы

# Разработать систему мониторинга сети, способную выявлять и блокировать подозрительный трафик. Создать скрипт на Python для прослушивания и анализа входящего трафика с использованием Scapy. Система анализирует параметры пакетов для обнаружения подозрительной активности, такой как аномальные объемы данных, и при необходимости блокирует подозрительные IP.

# Описание задачи (формализация задачи)

1. **Сбор и анализ данных о сетевом трафике**: фиксирование основных параметров: IP-адрес, порт и размер пакета.
2. **Определение критериев подозрительного трафика**:
   * Обнаружение IP-адресов, передающих аномально большие объемы данных.
   * Обнаружение IP-адресов, часто повторяющих запросы в короткий промежуток времени.
   * Распознавание подозрительных действий, таких как порт-сканирование (много соединений на разные порты от одного IP).
3. **Выявление подозрительного трафика**: Занесение IP-адреса в список подозрительных IP.
4. **Блокировка подозрительных IP-адресов**: при нажатии пользователем кнопки блокировки для конкретного подозрительного IP программа должна добавить правило для блокировки данного IP-адреса, предотвращая дальнейшую передачу данных с него.

# Теоретическая часть

Мониторинг сетевого трафика — это процесс сбора и анализа данных, передаваемых по сети, для обеспечения безопасности и эффективности сетевой инфраструктуры. Одной из основных задач мониторинга является обнаружение аномалий и угроз, таких как сканирование портов, чрезмерная передача данных от определенных источников или повторяющиеся запросы, которые могут свидетельствовать о сетевых атаках.

### **Методы обнаружения подозрительного трафика**

1. **Анализ сигнатур**. Этот метод основывается на заранее известных шаблонах поведения, которые характерны для определенных типов атак. Недостаток метода — ограниченная способность обнаруживать новые виды атак, которые не имеют сигнатур.
2. **Поведенческий анализ**. Ищет аномалии, отклоняющиеся от нормы, например, непропорционально большие объемы трафика от одного IP или частое сканирование портов. Поведенческий анализ эффективен против новых атак, так как он ориентируется на необычные изменения в сетевом поведении.
3. **Использование статистических методов**. Статистический анализ позволяет отслеживать метрики сетевого трафика, такие как частота запросов, объем передаваемых данных и распределение пакетов по времени.

**Библиотека Scapy для анализа трафика**

**Scapy** — это библиотека для Python, которая позволяет работать с сетевыми пакетами. Она предоставляет возможности для создания, отправки, получения и анализа пакетов на различных уровнях сетевой модели OSI.

Основные функции, которые предоставляет Scapy, включают:

* **Перехват пакетов**: позволяет прослушивать трафик в режиме реального времени.
* **Создание и отправка пакетов**: Scapy поддерживает формирование и отправку пакетов.

### **Блокировка подозрительного трафика с помощью iptables**

Для блокировки подозрительных IP-адресов часто используются, например iptables — утилита, которая позволяет создавать правила фильтрации на уровне ядра. iptables предоставляет возможности для блокировки и разрешения трафика на основе адресов, портов и других параметров.

# Основные шаги программы

* **Интерфейс**:
* Создаётся графический интерфейс с помощью tkinter.
  + "Received" — отображает все входящие пакеты
  + "Flagged" — отображает IP, с которых посылается подозрительные пакеты.
  + "Blocked" — отображает заблокированные IP.
* Кнопки управления позволяют начинать и останавливать мониторинг, а также блокировать и разблокировать подозрительные IP.
* **Обработка пакетов**:
* Вызывается функция для каждого пакета, перехваченного с помощью библиотеки Scapy. Она анализирует IP-адрес источника и размер пакета.
* Если суммарный размер пакетов от одного IP превышает пороговое значение, адрес добавляется в таблицу "Flagged".
* **Запуск и остановка**:
* Кнопка "Start" управляет процессом мониторинга.
* При нажатии "Stop" процесс отслеживания трафика приостанавливается.
* **Блокировка**:
* Если IP признан подозрительным, его можно выбрать из таблицы "Flagged" и заблокировать нажатием кнопки "Block".
* Блокировка IP-адреса выполняется с помощью команды iptables. Адрес затем переносится в таблицу "Blocked".
* **Разблокировка**:
* Для разблокировки IP его необходимо выбрать из таблицы "Blocked" и нажать кнопку "Unblock".
* При разблокировке правила iptables вызывает разблокировку IP на системном уровне, восстанавливая возможность соединения для IP.

# Описание программы

Описание основных компонентов программы представлено в таблице 4.

*Таблица 4. Описание функций*

|  |  |  |
| --- | --- | --- |
| Имя | Параметры | Описание |
| packets | packet: Переменная, содержащая размер пакета | Функция для определения подозрительных IP |
| begin\_mon | - | Запуск мониторинга |
| stop\_mon | - | Остановка мониторинга |
| block\_ip | - | Блокировка IP в интерфейсе |
| unblock\_ip | - | Разблокировка IP в интерфейсе |
| add\_ip | ip: IP адрес | Блокировка трафика по IP |
| remove\_ip | ip: IP адрес | Разблокировка трафика по IP |

# Рекомендации пользователя

# **1.** sudo python3 main.py

**2.** Нажмите кнопку **start** чтобы начать мониторинг входящего сетевого трафика.

* 1. 3. При обнаружении подозрительных IP-адресов, они появятся в колонке Flagged с описанием причины.
  2. 4. Чтобы заблокировать IP-адрес из списка Flagged, выберите нужный IP и нажмите кнопку **block**. Чтобы разблокировать IP-адрес, выберите его в колонке Blocked и нажмите кнопку **Unblock**.
  3. 5. Для завершения мониторинга нажмите кнопку **stop**.

# Рекомендации программиста

# Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с python 3.1 и библиотеки scapy[1], tkinter[2], threading[3], subprocess[4]

# Минимальное необходимое место на диске: 1 МБ. Минимальное необходимое количество оперативной памяти: 25 МБ.

# Исходный код программы

# Исходный код программы и необходимые текстовые файлы доступны по ссылке: <https://github.com/NikiTaku1/spbu_linux/tree/main/lab2>

# Контрольный пример

В данном разделе представлены контрольные примеры для каждой из задач, демонстрирующие способность выполнять комплексную обработку данных.

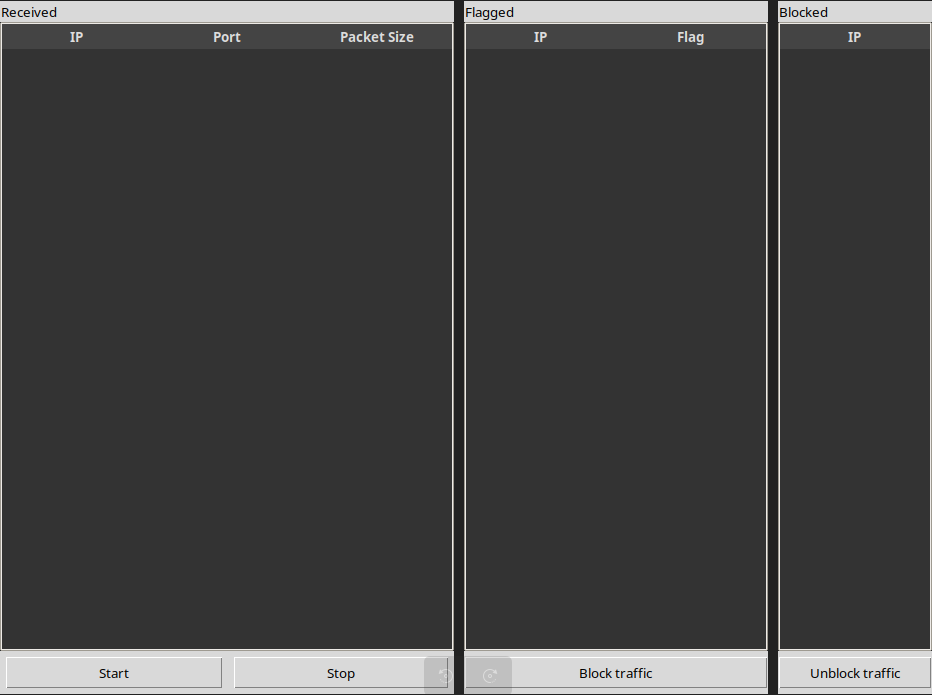


Рис. 9.1 Интерфейс.

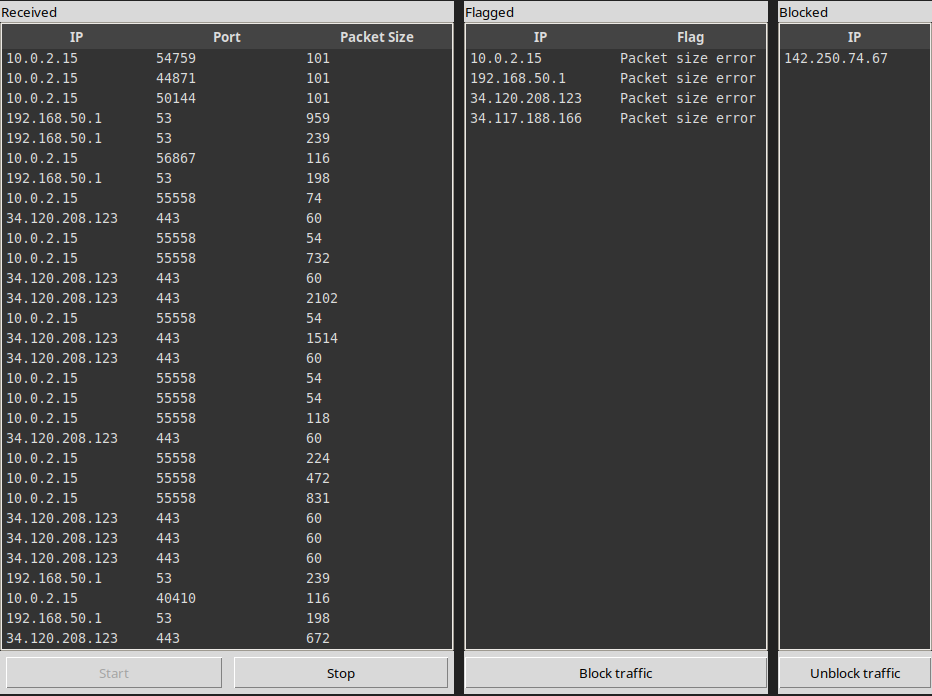


Рис. 9.2 Работа программы

# Вывод

# В ходе разработки программы был реализован функционал для отслеживания входящих сетевых пакетов, выявления подозрительных IP-адресов и их блокировки с помощью утилиты iptables. Основной функционал программы заключается в обеспечении безопасности сети, позволяя своевременно выявлять и блокировать нежелательные подключения. Реализация выполнена на языке Python с использованием библиотеки scapy для захвата трафика и tkinter для создания графического интерфейса.

# Источники

# scapy — scapy documentation // Documentation URL: [https://scapy.readthedocs.io/en/latest/index.html](https://scapy.readthedocs.io/en/latest/index.html%20) (дата обращения: 02.12.2024).

# tkinter — tkinter documentation // Documentation URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 02.12.2024).

# threading — threading documentation // Documentation URL: <https://docs.python.org/3/library/threading.html> (дата обращения: 02.12.2024).

# subprocess — subprocess documentation // Documentation URL: <https://docs.python.org/3/library/subprocess.html> (дата обращения: 02.12.2024).