**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**

**Шарабарин М.С.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc182212872)
2. [Описание задачи (формализация задачи) 4](#_Toc182212873)
3. [Теоретическая часть 5](#_Toc182212874)

**[Методы обнаружения подозрительного трафика](#_Toc182212875)** [5](#_Toc182212875)

**[Библиотека Scapy для анализа трафика](#_Toc182212876)** [5](#_Toc182212876)

**[Блокировка подозрительного трафика с помощью iptables](#_Toc182212877)** [6](#_Toc182212877)

1. [Основные шаги программы 7](#_Toc182212878)
2. [Описание программы 9](#_Toc182212879)
3. [Рекомендации пользователя 11](#_Toc182212880)
4. [Рекомендации программиста 12](#_Toc182212881)

**[1. Установка необходимых библиотек Python](#_Toc182212882)**[: 12](#_Toc182212882)

**[2](#_Toc182212883)**[.](#_Toc182212883) **[Установка утилит для работы с iptables:](#_Toc182212883)** [12](#_Toc182212883)

1. [Исходный код программы 12](#_Toc182212884)
2. [Контрольный пример 13](#_Toc182212885)
3. [Вывод 14](#_Toc182212886)
4. [Источники 15](#_Toc182212887)

# Цель работы

Цель работы — разработать систему мониторинга и защиты сети, способную выявлять и блокировать подозрительный сетевой трафик. В рамках работы создается скрипт на Python для прослушивания и анализа входящего трафика с использованием библиотеки **Scapy**. Система анализирует параметры пакетов для обнаружения потенциально вредоносной активности, такой как аномальные объемы данных или частые повторяющиеся запросы, и при необходимости блокирует подозрительные IP-адреса.

# Описание задачи (формализация задачи)

Задача предполагает следующие этапы:

1. **Сбор и анализ данных о сетевом трафике**: программа должна уметь перехватывать входящие сетевые пакеты, фиксировать их основные параметры, такие как IP-адрес источника, номер порта и размер пакета. Эта информация будет отображаться в графическом интерфейсе, что позволит визуализировать активные соединения.
2. **Определение критериев подозрительного трафика**: необходимо разработать и внедрить правила обнаружения аномального поведения, которые позволят отфильтровывать подозрительные пакеты. Такие правила могут включать:
   * 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**: при обнаружении пакета, удовлетворяющего условиям подозрительности, IP-адрес должен быть занесен в список подозрительных IP. В графическом интерфейсе для этих IP-адресов должно указываться, какая именно аномалия была выявлена.
4. **Блокировка подозрительных IP-адресов**: при нажатии пользователем кнопки блокировки для конкретного подозрительного IP программа должна добавить правило в **iptables** для блокировки данного IP-адреса, предотвращая дальнейшую передачу данных с него. При этом заблокированный IP должен отобразиться в отдельном списке, и для него должна быть доступна функция разблокировки.

Таким образом, задача заключается в создании программного обеспечения, которое будет обеспечивать анализ сетевого трафика в реальном времени, выявлять подозрительные соединения по заданным правилам и оперативно блокировать их по мере необходимости.

# 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**

Для анализа сетевого трафика применяются различные методы и подходы:

1. **Анализ сигнатур**. Этот метод основывается на заранее известных шаблонах поведения, которые характерны для определенных типов атак. Например, DDoS-атаки часто включают в себя большое количество запросов с одного или нескольких IP-адресов. Недостаток метода — ограниченная способность обнаруживать новые виды атак, которые не имеют сигнатур.
2. **Поведенческий анализ**. В отличие от сигнатурного, 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. На основе статистики можно строить пороговые значения, за **рамки которых выходят подозрительные активности.**

### **Библиотека Scapy для анализа трафика**

**Scapy** — это мощная 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. **Scapy** может использоваться для сниффинга (прослушивания) трафика, что делает ее идеальным инструментом для обнаружения аномалий в сетевом поведении.

Основные функции, которые предоставляет Scapy, включают:

* **Перехват пакетов**: позволяет прослушивать трафик в режиме реального времени, обрабатывая входящие пакеты и извлекая из них полезную информацию (например, IP-адреса источника и назначения, порты, размеры пакетов и заголовки).
* **Создание и отправка пакетов**: **Scapy** поддерживает формирование и отправку пакетов, что может быть полезно для активного сетевого тестирования и отправки управляющих ICMP-сообщений для блокировки.

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

Для блокировки подозрительных IP-адресов часто используются сетевые фильтры, такие как **iptables** — утилита, которая позволяет создавать правила фильтрации на уровне ядра **Linux**. **iptables** предоставляет гибкие возможности для блокировки и разрешения трафика на основе адресов, портов и других параметров. В рамках данной задачи блокировка подозрительного трафика осуществляется путем добавления правил для запрета входящих соединений с определенных IP-адресов, что предотвращает их взаимодействие с сетевой инфраструктурой.

# Основные шаги программы

* **Инициализация интерфейса**:
* Создаётся графический интерфейс с помощью tkinter. Интерфейс включает три таблицы:
  + "All Incoming IPs" — для отображения всех входящих IP-адресов и информации о соединениях.
  + "Suspicious IPs" — для фиксации IP-адресов, превышающих пороговое значение по объёму данных.
  + "Blocked IPs" — для отображения заблокированных IP-адресов.
* Кнопки управления позволяют начинать и останавливать мониторинг, а также блокировать и разблокировать подозрительные IP-адреса.
* **Обработка пакетов**:
* Функция packet\_callback вызывается 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 (например, 200 байт), адрес классифицируется как подозрительный и добавляется в таблицу "Suspicious IPs".
* IP-адреса, которые не блокируются, также отображаются в таблице "All Incoming IPs" с указанием порта и размера пакета.
* **Запуск и остановка мониторинга**:
* Кнопки "Start" и "Stop" управляют процессом мониторинга. При нажатии "Start" запускается поток, в котором Scapy отслеживает трафик и вызывает packet\_callback для каждого пакета.
* При остановке мониторинга поток завершает работу, и можно проанализировать накопленные данные.
* **Блокировка подозрительных IP-адресов**:
* Если IP-адрес признан подозрительным, его можно выбрать из таблицы "Suspicious IPs" и заблокировать нажатием кнопки "Block".
* Блокировка IP-адреса выполняется добавлением правила в iptables с помощью команды subprocess.run. Адрес затем переносится в таблицу "Blocked IPs".
* **Разблокировка IP-адресов**:
* Для разблокировки IP-адреса его необходимо выбрать из таблицы "Blocked IPs" и нажать кнопку "Unblock".
* При разблокировке правила iptables удаляются, восстанавливая возможность соединения для IP-адреса.

# Описание программы

Программа мониторинга сетевого трафика написана на языке Python с использованием библиотеки Scapy для перехвата и анализа сетевых пакетов. Графический интерфейс создан с помощью tkinter, что позволяет пользователю взаимодействовать с программой и управлять мониторингом трафика. Основные функции включают анализ входящих пакетов для выявления подозрительных IP-адресов, возможность блокировки и разблокировки IP через iptables, а также отображение информации о подозрительных и заблокированных IP. Программа организована в классе NetworkTrafficMonitor, который объединяет все основные функции, такие как обработка пакетов, запуск мониторинга и управление блокировкой IP.

Таблица 1. net\_scanner.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| \_\_init\_\_ | Инициализирует GUI и таблицы для отображения IP-адресов. | None |
| packet\_callback | Обрабатывает входящие пакеты, обновляет статистику по IP и отмечает подозрительные IP. | None |
| start\_monitoring | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Начинает мониторинг сетевого трафика, создавая новый поток для анализа пакетов. | | | None |
| monitor\_traffic | |  | | --- | |  |  |  | | --- | | Запускает функцию захвата пакетов с помощью Scapy. | | None |
| stop\_monitoring | Останавливает мониторинг, завершает поток захвата пакетов. | None |
| block\_ip | Блокирует выбранный подозрительный IP, добавляя его в таблицу блокировки и iptables. | None |

|  |  |  |
| --- | --- | --- |
| unblock\_ip | Разблокирует IP, удаляя его из таблицы блокировки и iptables. | None |
| add\_iptables\_rule | Добавляет правило в iptables для блокировки IP-адреса. | None |
| remove\_iptables\_rule | Удаляет правило в iptables, позволяя IP снова взаимодействовать с системой. | None |

# Рекомендации пользователя

1. **Запуск программы**:
   * Откройте и запустите файл программы. После запуска откроется графический интерфейс с тремя колонками: All Incoming IPs, Suspicious IPs и Blocked IPs.
2. **Начало мониторинга**:
   * Нажмите кнопку **Start** в левой колонке All Incoming IPs, чтобы начать мониторинг входящего сетевого трафика. Программа начнёт отображать информацию о каждом IP-адресе, отправляющем пакеты на вашу систему, включая IP-адрес, порт и размер пакета.
3. **Просмотр подозрительных IP**:
   * При обнаружении подозрительных IP-адресов (например, если объём пакетов от одного IP превышает порог в 200 байт), они появятся в колонке Suspicious IPs с описанием причины.
4. **Блокировка подозрительных IP**:
   * Чтобы заблокировать IP-адрес из списка Suspicious IPs, выберите нужный IP и нажмите кнопку **Block**. IP-адрес будет перемещён в колонку Blocked IPs и добавлен в правила iptables, чтобы заблокировать дальнейшие подключения с этого адреса.
5. **Разблокировка IP**:
   * Чтобы разблокировать IP-адрес, выберите его в колонке Blocked IPs и нажмите кнопку **Unblock**. Это удалит правило из iptables, и IP-адрес будет разблокирован.
6. **Остановка мониторинга**:
   * Для завершения мониторинга нажмите кнопку **Stop** в колонке All Incoming IPs. Программа прекратит захват пакетов, и кнопка **Start** станет снова доступной.

Программа рассчитана на непрерывный мониторинг, поэтому можно в любой момент запустить или остановить мониторинг, а также блокировать или разблокировать IP-адреса по мере необходимости.

# Рекомендации программиста

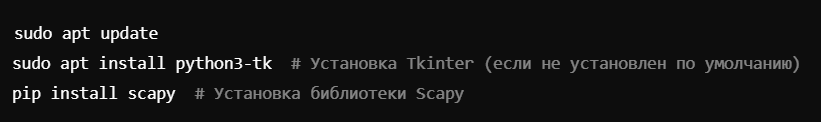
Для успешной работы программы **net\_scanner** потребуется установить несколько библиотек и утилит. Ниже представлены шаги, которые **необходимо выполнить:**

### **1. Установка необходимых библиотек Python:**

Программа использует библиотеку **Scapy** для захвата и анализа сетевых пакетов и **Tkinter** для создания графического интерфейса. Обе библиотеки могут быть установлены с помощью менеджера пакетов pip.

* **Scapy** — библиотека для работы с пакетами на сетевом уровне.
* **Tkinter** — стандартная библиотека Python для создания графических интерфейсов.

Для установки **Scapy** и **Tkinter** выполните следующие команды в терминале:



### **2**. **Установка утилит для работы с iptables:**

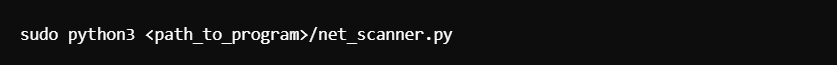
Программа использует утилиту iptables для блокировки и разблокировки IP-адресов. Убедитесь, что она установлена в вашей системе.

Для установки iptables выполните команду:

****

**3.** **Запуск программы с правами администратора:**

Для блокировки и разблокировки IP-адресов через **iptables** программе могут потребоваться права администратора. Запустите программу с использованием команды **sudo**, чтобы избежать ошибок доступа:



# Исходный код программы

<https://github.com/Kliooo/System-Programming-Linux>

# Контрольный пример

* 1. **Запуск исходного файла:** запустите файл **net\_scanner.py**, появится графический интерфейс (Рис. 1):

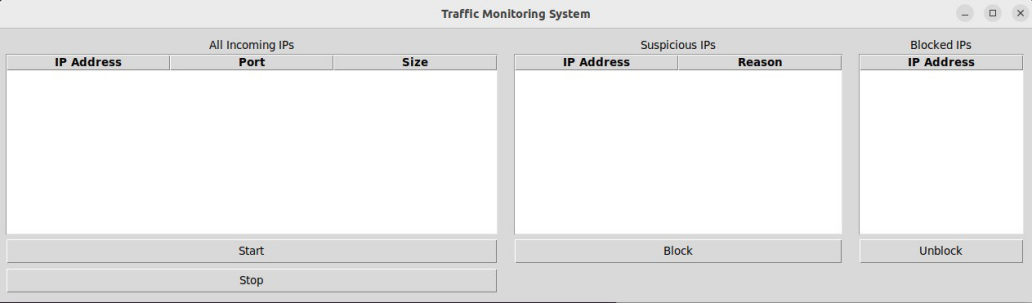


Рис. 1 Графический интерфейс

* 1. **Запуск мониторинга входящего сетевого трафика:** нажимаем на кнопку – “Start” и ждём, когда появятся первые трафики, а в месте с ними и подозрительные. (Рис. 2):

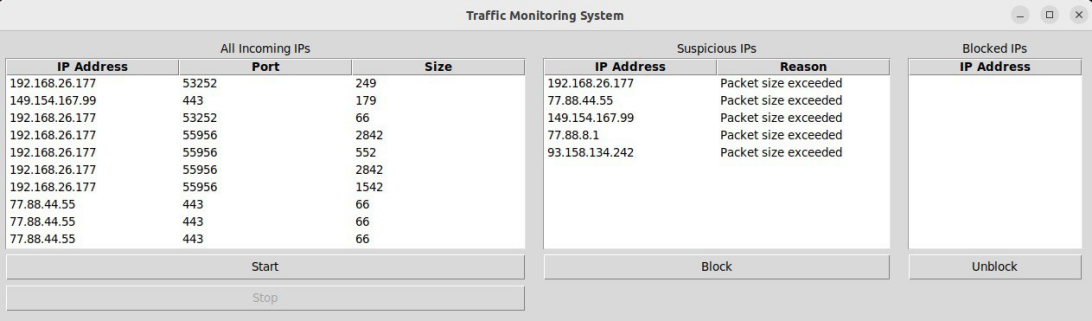


Рис. 2 Мониторинг

* 1. **Блокировка:** выбираем какой-то адрес из таблицы подозрительных и блокируем на кнопку – “Block” (Рис.3):

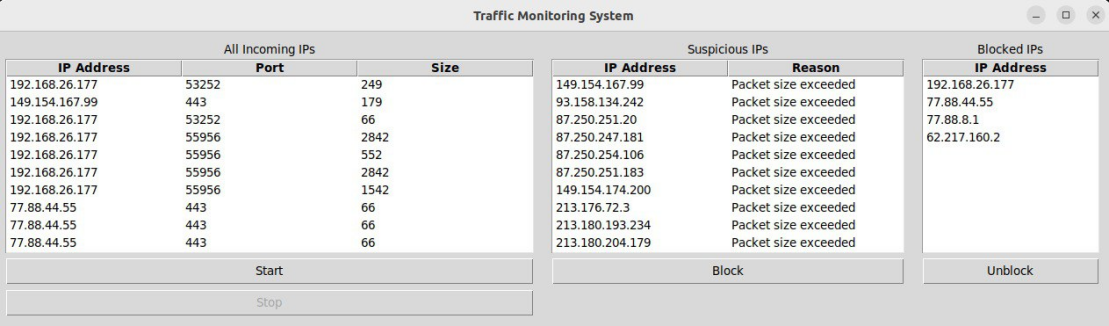


Рис. 3 Блокировка подозрительного трафика

# Вывод

В ходе разработки программы для мониторинга сетевого трафика был реализован функционал для отслеживания входящих сетевых пакетов, выявления подозрительных IP-адресов и их блокировки с помощью утилиты **iptables**. Программа предоставляет пользователю графический интерфейс с возможностью просмотра всех входящих адресов, а также отображает подозрительные и заблокированные IP-адреса. 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. В дальнейшем можно расширить программу, добавив дополнительные функции для анализа трафика или интеграции с другими системами безопасности.

# Источники

• scapy — Для захвата и анализа сетевого трафика.  
<https://ru.wikipedia.org/wiki/Scapy>

*Дата обращения: (11.11.2024)*

• tkinter — Для создания графического интерфейса.  
<https://docs.python.org/3/library/tkinter.html>

*Дата обращения: (11.11.2024)*

• subprocess — Для работы с системными процессами.  
<https://docs.python.org/3/library/subprocess.html>

*Дата обращения: (11.11.2024)*

• threading — Для организации многозадачности и параллельных потоков.  
*<https://docs.python.org/3/library/threading.html>*

*Дата обращения: (11.11.2024)*