**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**

**Овчинников С.А.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc185601365)
2. [Описание задачи (формализация задачи) 3](#_Toc185601366)
3. [Теоретическая часть 4](#_Toc185601367)

[Мониторинг сетевого трафика 4](#_Toc185601368)

[Методы обнаружения подозрительного трафика 4](#_Toc185601369)

[Scapy для анализа трафика 5](#_Toc185601370)

[Блокировка подозрительного трафика 5](#_Toc185601371)

[Основные этапы программы 5](#_Toc185601372)

1. [Описание программы 7](#_Toc185601373)
2. [Рекомендации пользователя 8](#_Toc185601374)

[Примечания 9](#_Toc185601375)

1. [Рекомендации программиста 10](#_Toc185601376)

[**1. Установка необходимых библиотек Python**: 10](#_Toc185601377)

[**2**. **Установка утилит для работы с iptables:** 10](#_Toc185601378)

1. [Исходный код программы 10](#_Toc185601379)
2. [Контрольный пример 10](#_Toc185601380)
3. [Вывод 12](#_Toc185601381)
4. [Источники 14](#_Toc185601382)

# Цель работы

Целью работы является создание системы мониторинга и защиты сети, которая сможет обнаруживать и блокировать подозрительный сетевой трафик. В рамках реализации разрабатывается скрипт на языке Python, использующий библиотеку Scapy для прослушивания и анализа входящих данных. Система выполняет анализ параметров сетевых пакетов для выявления признаков потенциально вредоносной активности, включая аномальные объемы передаваемых данных или частые повторяющиеся запросы, а также блокирует подозрительные IP-адреса при необходимости.

# Описание задачи (формализация задачи)

Задача предполагает следующие этапы:

1. **Сбор и анализ сетевого трафика**:  
   Программа должна перехватывать входящие сетевые пакеты и фиксировать ключевые параметры, такие как IP-адрес источника, номер порта и размер пакета. Эти данные будут отображаться в графическом интерфейсе, позволяя визуализировать активные соединения.
2. **Определение критериев подозрительного трафика**:  
   Разработка правил для выявления аномального поведения с целью фильтрации подозрительных пакетов. Правила могут включать:
   * Обнаружение IP-адресов, отправляющих чрезмерно большие объемы данных.
   * Определение IP-адресов, часто повторяющих запросы за короткий период.
   * Распознавание подозрительных действий, например, порт-сканирования (многочисленные соединения с разными портами от одного IP).
3. **Выявление подозрительного трафика**:  
   При обнаружении пакетов, соответствующих заданным условиям, IP-адрес заносится в список подозрительных. В графическом интерфейсе для каждого такого адреса указывается выявленная аномалия.
4. **Блокировка подозрительных IP-адресов**:  
   При нажатии кнопки блокировки для конкретного IP программа добавляет правило в iptables для его блокировки, предотвращая дальнейшую передачу данных. Заблокированные IP отображаются в отдельном списке, где доступна возможность их разблокировки.

Таким образом, задача заключается в разработке программного обеспечения, способного анализировать сетевой трафик в реальном времени, выявлять подозрительные соединения на основе заданных правил и оперативно блокировать их при необходимости.

# Теоретическая часть

### Мониторинг сетевого трафика

Мониторинг сетевого трафика представляет собой процесс сбора и анализа данных, передаваемых по сети, с целью обеспечения ее безопасности и повышения эффективности. Основная задача мониторинга — выявление аномалий и угроз, таких как порт-сканирование, чрезмерная передача данных с одного источника или частые повторяющиеся запросы, что может свидетельствовать о сетевых атаках.

### Методы обнаружения подозрительного трафика

Для анализа сетевого трафика применяются различные подходы:

1. **Анализ сигнатур**:  
   Метод основан на сравнении данных с известными шаблонами, характерными для определенных атак. Например, DDoS-атаки сопровождаются множественными запросами с одного или нескольких IP. Однако этот метод ограничен и плохо справляется с новыми типами атак, для которых сигнатуры еще не созданы.
2. **Поведенческий анализ**:  
   Данный метод выявляет аномалии, отклоняющиеся от привычного поведения. Это может быть необычно большой объем данных от одного источника или частое сканирование портов. Поведенческий подход более универсален и эффективен против неизвестных атак, так как ориентирован на обнаружение отклонений от нормы.
3. **Статистический анализ**:  
   Используется для отслеживания метрик трафика, таких как частота запросов, объем данных и временное распределение пакетов. На основе собранной статистики устанавливаются пороговые значения, превышение которых сигнализирует о подозрительной активности.

### Scapy для анализа трафика

**Scapy** — мощная библиотека для Python, позволяющая работать с сетевыми пакетами. Она предоставляет инструменты для создания, отправки, получения и анализа пакетов на различных уровнях сетевой модели OSI.

Основные функции Scapy:

* **Перехват пакетов**: позволяет прослушивать сетевой трафик в реальном времени и извлекать данные, такие как IP-адреса, порты, размеры пакетов и их заголовки.
* **Создание и отправка пакетов**: подходит для активного тестирования сети и управления, например, отправки ICMP-сообщений.

Scapy делает процесс анализа трафика и обнаружения аномалий удобным и гибким, что позволяет интегрировать библиотеку в систему мониторинга.

### Блокировка подозрительного трафика

Для блокировки подозрительных IP-адресов используется **iptables** — утилита Linux для управления фильтрацией трафика. Она позволяет добавлять правила, ограничивающие доступ от определенных IP-адресов, чтобы предотвратить потенциально вредоносные соединения.

### Основные этапы программы

1. **Инициализация интерфейса**:  
   Разрабатывается графический интерфейс с использованием **tkinter**, включающий три таблицы:
   * "All Incoming IPs" — отображает все входящие IP-адреса и параметры соединений.
   * "Suspicious IPs" — содержит адреса, превышающие пороговые значения (например, по объему данных).
   * "Blocked IPs" — показывает заблокированные IP-адреса.  
     Управление осуществляется через кнопки для начала/остановки мониторинга, а также блокировки и разблокировки IP-адресов.
2. **Обработка пакетов**:
   * Используется функция packet\_callback, вызываемая для каждого перехваченного пакета. Она анализирует источник пакета и его размер.
   * Если объем трафика от одного IP превышает установленный порог (например, 200 байт), адрес добавляется в список "Suspicious IPs".
   * В таблице "All Incoming IPs" отображается информация обо всех перехваченных соединениях, включая порты и размеры пакетов.
3. **Запуск и остановка мониторинга**:
   * Кнопки "Start" и "Stop" управляют процессом мониторинга. При нажатии "Start" запускается поток для сниффинга трафика через Scapy.
   * Остановка мониторинга завершает поток, сохраняя накопленные данные для анализа.
4. **Блокировка IP-адресов**:
   * Из списка "Suspicious IPs" пользователь может выбрать адрес для блокировки.
   * После нажатия кнопки "Block" в **iptables** добавляется правило, запрещающее доступ с указанного IP. Адрес переносится в таблицу "Blocked IPs".
5. **Разблокировка IP-адресов**:
   * Для восстановления доступа необходимо выбрать адрес в таблице "Blocked IPs" и нажать кнопку "Unblock".
   * Удаление правила из iptables возвращает возможность взаимодействия с этим IP.

# 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 элемент класса | 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

#### **Запуск программы**

* Откройте и запустите файл программы.
* После запуска появится графический интерфейс с тремя колонками:
  + **All Incoming IPs** — для отображения всех входящих IP-адресов.
  + **Suspicious IPs** — для подозрительных адресов.
  + **Blocked IPs** — для заблокированных адресов.

#### 2. **Начало мониторинга**

* Нажмите кнопку **Start** в секции **All Incoming IPs**.
* Программа начнёт мониторить входящий сетевой трафик.
* В колонке **All Incoming IPs** будет отображаться информация о каждом IP-адресе, включая:
  + IP-адрес отправителя.
  + Номер порта.
  + Размер пакета.

#### 3. **Просмотр подозрительных IP**

* Если программа обнаружит IP-адрес, который превышает пороговые значения (например, суммарный объём пакетов превышает 200 байт), этот адрес будет добавлен в колонку **Suspicious IPs**.
* В колонке будет указана причина, по которой адрес признан подозрительным.

#### 4. **Блокировка подозрительных IP**

* Для блокировки подозрительного IP-адреса:
  + Выберите адрес в колонке **Suspicious IPs**.
  + Нажмите кнопку **Block**.
* Заблокированный адрес переместится в колонку **Blocked IPs**.
* Программа автоматически добавит правило в **iptables**, чтобы заблокировать соединения с этого IP-адреса.

#### 5. **Разблокировка IP**

* Чтобы разблокировать ранее заблокированный IP-адрес:
  + Выберите адрес в колонке **Blocked IPs**.
  + Нажмите кнопку **Unblock**.
* Программа удалит соответствующее правило из **iptables**, и адрес снова станет доступным для подключения.

#### 6. **Остановка мониторинга**

* Для завершения мониторинга нажмите кнопку **Stop** в секции **All Incoming IPs**.
* Программа прекратит перехват пакетов.
* Кнопка **Start** станет доступной для повторного запуска мониторинга.

### Примечания

* Программа предназначена для непрерывного мониторинга. Вы можете в любой момент:
  + Запустить или остановить мониторинг.
  + Блокировать или разблокировать IP-адреса.

# 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:**

Для работы библиотек необходимо создать виртуальное пространство, в которое нужно будет добавить необходимые библиотеки Scapy и Tkinter



### **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:

****

# Исходный код программы

https://github.com/Browochka/linux-y2

# Контрольный пример

* 1. **Запуск исходного файла:** запустите файл **net\_scanner.py** с помощью команды и появится появится графический интерфейс (Рис. 1):



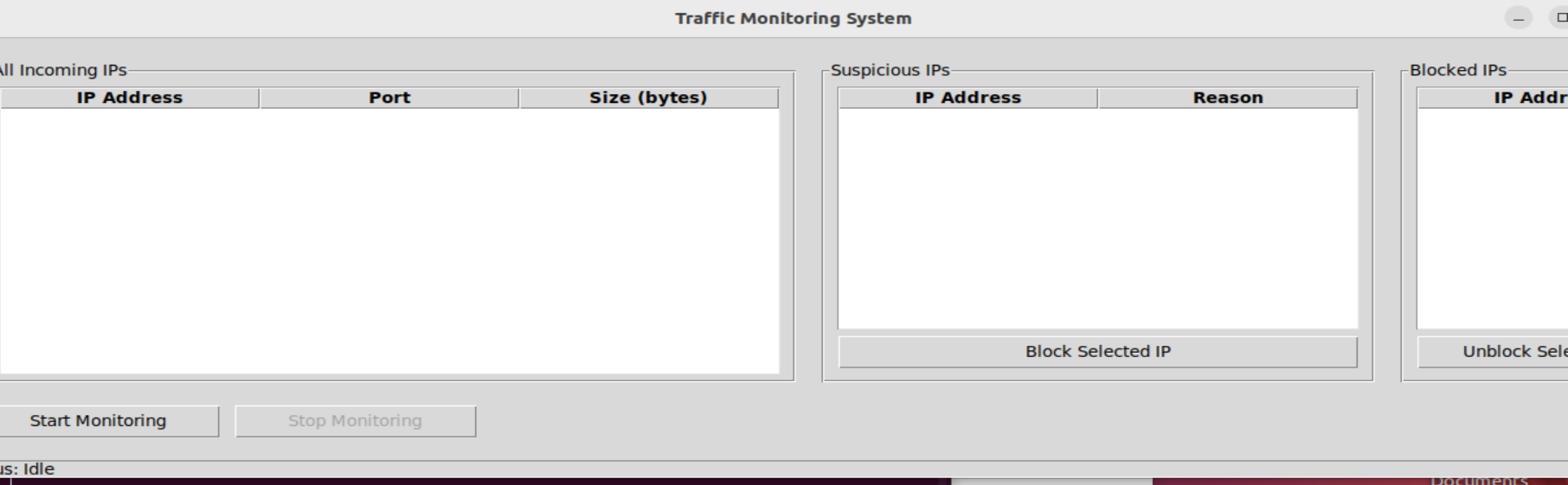


Рис. 1 Графический интерфейс

* 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):

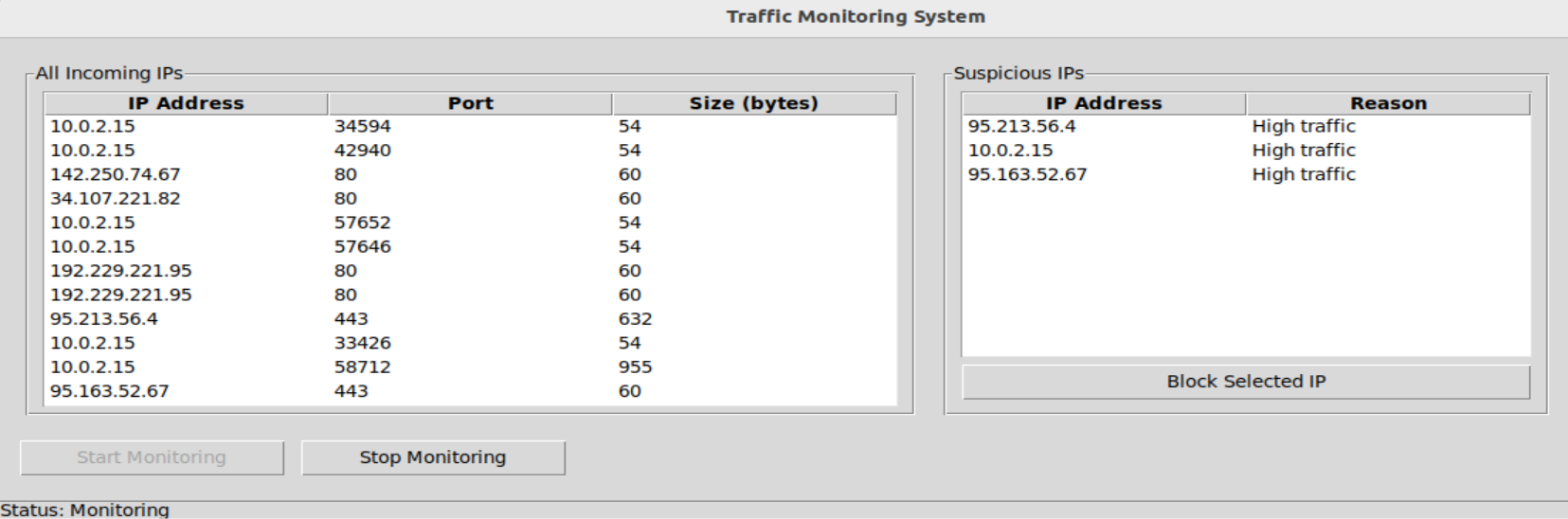


Рис. 2 Мониторинг

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

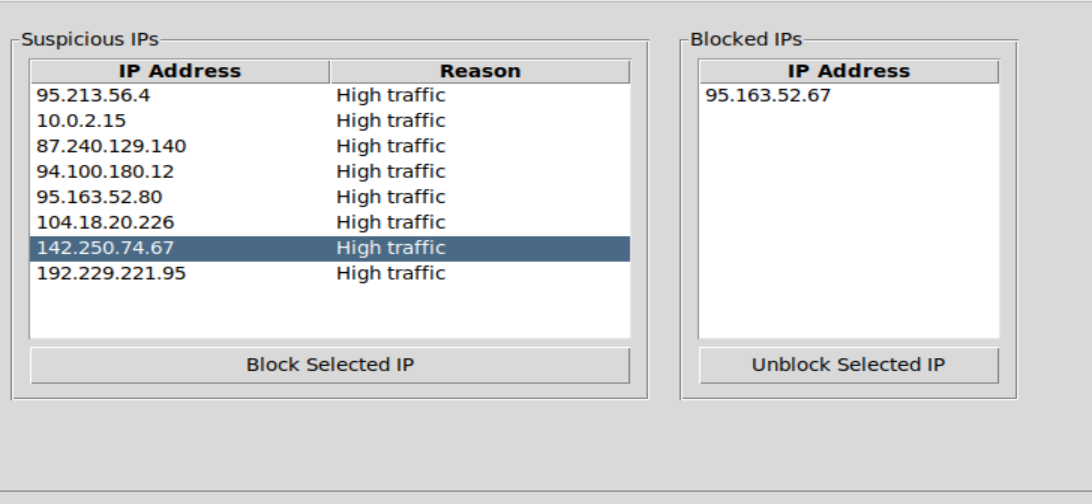


Рис. 3 Блокировка подозрительного трафика

# Вывод

В ходе разработки программы для мониторинга сетевого трафика была реализована система, позволяющая:

* **Отслеживать входящие сетевые пакеты**.
* **Выявлять подозрительные IP-адреса** на основе заданных критериев.
* **Блокировать нежелательные подключения** с помощью утилиты **iptables**.

Программа оснащена удобным графическим интерфейсом, предоставляющим пользователю следующие возможности:

1. Просмотр всех входящих IP-адресов и информации о соединениях.
2. Отображение списка подозрительных IP-адресов с указанием причин их классификации как подозрительных.
3. Управление списком заблокированных адресов, включая добавление и снятие блокировок.

Ключевая цель программы — обеспечение безопасности сети за счёт своевременного выявления и предотвращения потенциально вредоносных подключений.

**Техническая реализация:**

* Программа разработана на языке **Python**.
* Для перехвата и анализа сетевого трафика используется библиотека **Scapy**.
* Для создания графического интерфейса применена библиотека **tkinter**.

**Потенциальное развитие:**  
В будущем функционал программы может быть дополнен:

* Расширением критериев анализа трафика для более точного выявления угроз.
* Интеграцией с другими инструментами и системами безопасности.
* Добавлением отчётности или логирования действий для более детального анализа событий.

Программа уже сейчас является мощным инструментом для мониторинга и защиты сетевой инфраструктуры, обеспечивая удобство использования и высокую функциональность.

# 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*](https://docs.python.org/3/library/threading.html)

*Дата обращения: (11.11.2024)*