**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. 24Б16-пу**

**Шитов И.А.**

**Преподаватель**

**Киямов Ж.У.**

**Санкт-Петербург**

**2025 г.**

Оглавление

[Цель работы 3](#_Toc184594076)

[Теоритическая часть: 3](#_Toc184594077)

[Алгоритм метода: 5](#_Toc184594078)

[Описание программы: 5](#_Toc184594079)

[Рекомендации пользователю: 8](#_Toc184594080)

[Рекомендации для программиста: 8](#_Toc184594081)

[Контрольный пример: 10](#_Toc184594082)

[Вывод: 12](#_Toc184594083)

[Источники: 13](#_Toc184594084)

### **Описание задачи**

Задача состоит из следующих этапов:

1. **Сбор и обработка данных о сетевом трафике**  
   Программа перехватывает входящие пакеты с использованием библиотеки Scapy, извлекая ключевые параметры, такие как IP-адрес отправителя, номер порта и размер пакета. Эти данные обрабатываются и отображаются в графическом интерфейсе, что позволяет пользователю видеть активные соединения и подозрительные активности в реальном времени.
2. **Установление критериев для подозрительного трафика**  
   Для анализа сетевого трафика используются следующие критерии:
   * Пакеты, превышающие установленный пользователем размер, считаются подозрительными.
   * IP-адреса, генерирующие множество соединений с разными портами (подозрительное сканирование портов).
   * Источники, которые повторяют запросы слишком часто в заданный интервал времени.  
     Эти параметры настраиваются через графический интерфейс, что позволяет пользователю адаптировать правила под свои нужды.
3. **Обнаружение подозрительного трафика**  
   Пакеты, соответствующие установленным критериям, заносятся в список подозрительных IP-адресов. Для каждого такого IP в графическом интерфейсе отображается природа аномалии (например, большой размер пакета, частые запросы или сканирование портов). Это позволяет пользователю быстро выявить потенциальные угрозы.
4. **Блокировка подозрительных IP-адресов**  
   Пользователь может заблокировать выбранные IP-адреса с помощью кнопки "Блокировать". Программа добавляет соответствующее правило в iptables, предотвращая дальнейшую передачу данных от этих IP-адресов. Заблокированные IP-адреса отображаются в интерфейсе, и для них предусмотрена возможность разблокировки.

### **Теоретическая часть**

Мониторинг сетевого трафика — это процесс анализа данных, передаваемых по сети, с целью обнаружения аномалий, угроз и обеспечения сетевой безопасности. Среди основных задач мониторинга выделяют:

* Выявление подозрительных действий, таких как сканирование портов.
* Обнаружение источников, генерирующих чрезмерный объем данных.
* Отслеживание повторяющихся запросов за короткий промежуток времени, что может указывать на сетевую атаку (например, DDoS).

В данной программе для перехвата пакетов используется библиотека **Scapy**, которая предоставляет мощные инструменты для работы с сетевым трафиком, а для блокировки IP-адресов применяется утилита **iptables** (или **pfctl** для macOS).

**Описание программы**

Программа реализована в одном Python-файле analyzer.py, который предоставляет графический интерфейс для управления процессом анализа и блокировки сетевого трафика.

Основные функции программы:

1. **Настройка параметров анализа:**  
   Пользователь может включать или отключать проверки на размер пакета, сканирование портов и частоту повторяющихся запросов. Для каждой проверки предусмотрены настраиваемые пороговые значения.
2. **Перехват трафика:**  
   Программа в течение заданного времени перехватывает входящие пакеты и анализирует их на основе установленных критериев.
3. **Отображение подозрительных IP:**  
   В интерфейсе отображаются IP-адреса, которые были определены как подозрительные. Для каждого IP выводится информация о типе нарушения и провайдере (получается через внешний API).
4. **Блокировка IP-адресов:**  
   Пользователь может выбрать один или несколько IP-адресов для блокировки. Программа добавляет правила в брандмауэр (iptable) для предотвращения передачи данных от этих адресов.

Интерфейс программы построен на **Tkinter** и включает удобные элементы управления, такие как чекбоксы, поля ввода и кнопки.

|  |  |  |
| --- | --- | --- |
| **Название функции** | **Переменные** | **Краткое описание** |
| **init** | self | Конструктор класса TrafficMonitor. Инициализирует интерфейс и параметры мониторинга |
| **setup\_ui** | self | Создает виджеты интерфейса: чекбоксы, поля ввода, кнопки управления, список IP и лог событий |
| **start** | self | Запускает процесс сканирования трафика в отдельном потоке на 15 секунд |
| **packet\_handler** | self, packet | Анализирует пакеты на основе заданных критериев (размер, сканирование портов, повторяющиеся запросы) |
| **get\_isp\_info** | self, ip | Отправляет запрос к внешнему API для получения информации об интернет-провайдере IP-адреса |
| **stop** | self | Останавливает процесс сканирования трафика |
| **on\_sniff\_end** | self | Вызывается при завершении захвата трафика, показывает результаты сканирования |

Рекомендации программисту

1. **Используемые библиотеки:**
   * **Scapy:** для перехвата и анализа сетевых пакетов. Позволяет гибко работать с сетевыми протоколами, но требует административных прав для запуска.
   * **Tkinter:** для создания графического интерфейса. Простая и встроенная в Python библиотека для разработки интерфейсов с элементами управления.
   * **Requests:** для выполнения HTTP-запросов, например, получения информации о провайдерах IP-адресов через внешние API.
   * **Threading:** для запуска перехвата трафика в отдельном потоке, чтобы не блокировать интерфейс пользователя.
2. **Как модифицировать код:**
   * **Добавление новых критериев для анализа:**  
     Для расширения функциональности можно добавить новые правила анализа трафика, например, определение странного поведения в ICMP-пакетах (ping-атаки) или блокировка по географическому положению IP.
   * **Работа с другими ОС:**  
     Для Windows можно интегрировать использование встроенных средств блокировки, таких как netsh. Это расширит кроссплатформенность программы.
   * **Поддержка базы данных:**  
     Добавление базы данных (например, SQLite) для хранения истории подозрительных IP и их активности. Это позволит сохранять данные между запусками программы.
3. **Оптимизация кода:**
   * Разделение программы на модули для повышения читаемости: например, отдельные файлы для интерфейса, логики обработки пакетов и взаимодействия с системой.
   * Логирование с использованием библиотеки logging для отслеживания событий и ошибок во время работы программы.

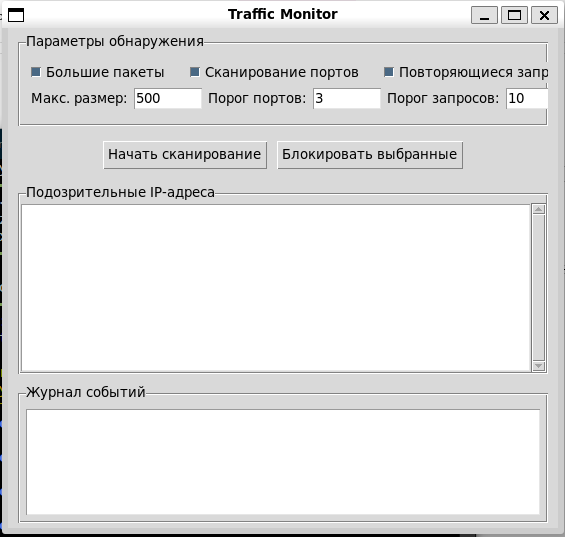
Рекомендации пользователю

1. **Запуск программы:**
   * Для корректной работы программы требуется запуск от имени администратора. Это необходимо для перехвата пакетов и добавления правил в брандмауэр.
   * Запустите программу с помощью команды:

Sudo python main.py

1. **Подготовка системы:**
   * Убедитесь, что на вашей системе установлен брандмауэр (iptables для Linux).
2. **Использование программы:**
   * После запуска программы выберите критерии для анализа: размер пакетов, сканирование портов или частые запросы. Настройте соответствующие пороговые значения в интерфейсе.
   * Нажмите **"Начать сканирование"**, чтобы запустить процесс анализа трафика. Подождите завершения сканирования.
   * После завершения программы в интерфейсе отобразятся подозрительные IP-адреса. Вы можете выбрать их для блокировки и нажать кнопку **"Блокировать выбранные"**.
3. **Работа с брандмауэром:**
   * После завершения работы программы проверьте правила брандмауэра на наличие заблокированных IP, чтобы избежать нежелательной блокировки доверенных источников.

Контрольный пример

Запуск скрипта с помощью sudo python main.py (необходма установленная библиотека scapy). На рисунке 1 изображен интерфейс программы:

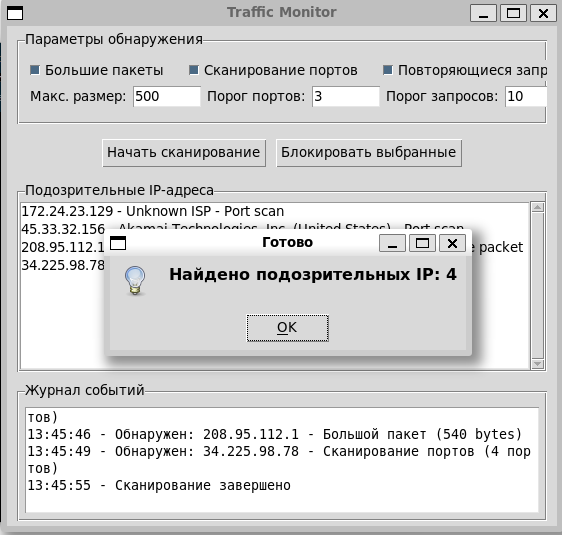
Рисунок 1. Начальное окно программы

Рисунок 2. Сканирование пакетов

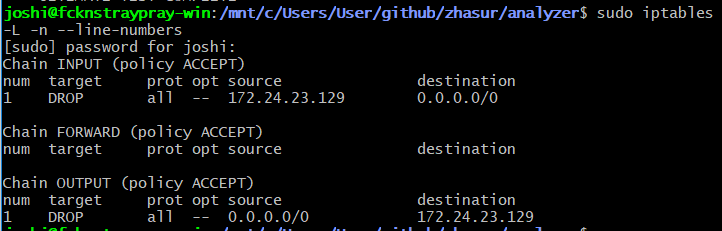
В результате iptables заблокирует выбранный ip:

Рисунок 3. Заблокированный с помощью программы IP

**Вывод**

В рамках данной работы была реализована программа для обнаружения и блокировки подозрительного сетевого трафика на основе заданных критериев. Программа позволяет:

* Перехватывать сетевые пакеты, анализируя ключевые параметры, такие как размер, источник и количество соединений.
* Определять подозрительное поведение, включая сканирование портов, передачу больших пакетов и частые повторяющиеся запросы.
* Обеспечивать гибкость настроек для пользователей, предоставляя параметры конфигурации через графический интерфейс.
* Блокировать подозрительные IP-адреса с использованием системных инструментов (iptables) для предотвращения дальнейших угроз.

Использование библиотеки Scapy позволило эффективно работать с сетевыми пакетами, а графический интерфейс на основе Tkinter сделал программу удобной для конечных пользователей. Разработка продемонстрировала, что мониторинг сетевого трафика является важным инструментом для повышения безопасности локальных сетей и защиты от сетевых атак.

Перспективы развития программы включают добавление поддержки других ОС, внедрение базы данных для хранения истории подозрительных IP, использование более мощных инструментов анализа, таких как машинное обучение, и улучшение пользовательского интерфейса.

Исходный код:

<https://github.com/joshixap/linux_autumn_2025/tree/master/analyzer>

Список источников

 Official Python Documentation. URL: <https://docs.python.org>

 Official Scapy Documentation. URL: <https://scapy.readthedocs.io>

 Tkinter Documentation. URL: <https://wiki.python.org/moin/TkInter>

 Linux iptables Documentation. URL: <https://netfilter.org>