**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

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

**ОТЧЁТ**

**По лабораторной работе №34 По дисциплине “Системное программирование в линукс”**

**На тему “Обнаружение и блокировка подозрительного сетевого трафика”**

**Студент гр. 23Б16-пу**

**Пушкарев Н.П.**

**Преподаватель**

**Киямов Ж.У.**

**Санкт-Петербург**

**2024 г.**

[Цель работы 3](#__RefHeading___Toc1293_325581465)

[Описание задачи 3](#__RefHeading___Toc1295_325581465)

[2. Определение подозрительного трафика 3](#__RefHeading___Toc1285_325581465)

[3. Блокировка подозрительного трафика 3](#__RefHeading___Toc1287_325581465)

[4. Тестирование и настройка 3](#__RefHeading___Toc1289_325581465)

[5. Непрерывное обновление и мониторинг 3](#__RefHeading___Toc1291_325581465)

[Теоретическая часть 3](#__RefHeading___Toc1313_325581465)

[Описание программы 6](#__RefHeading___Toc1311_325581465)

[Основные шаги программы 8](#__RefHeading___Toc1309_325581465)

[Рекомендации пользователя 8](#__RefHeading___Toc1307_325581465)

[Рекомендации программиста 9](#__RefHeading___Toc1305_325581465)

[Контрольный пример 9](#__RefHeading___Toc1303_325581465)

[Вывод 10](#__RefHeading___Toc1301_325581465)

[Исходный код 11](#__RefHeading___Toc1299_325581465)

[Список источников 11](#__RefHeading___Toc1297_325581465)

# Цель работы

Разработать простой механизм обнаружения и блокировки подозрительного сетевого трафика на основе определенных правил и сигнатур.

# Описание задачи

1. Создание скрипта для обнаружения и анализа трафика

2. Определение подозрительного трафика

3. Блокировка подозрительного трафика

4. Тестирование и настройка

5. Непрерывное обновление и мониторинг

# Теоретическая часть

**Сетевое сканирование**

Сканирование портов — это процесс проверки сетевых портов на наличие открытых соединений на целевом устройстве. Это метод широко используется как в легитимных целях, например, для тестирования безопасности, так и злоумышленниками для поиска уязвимостей в системах. В процессе сканирования определяется, какие порты на целевом устройстве открыты для связи, что может помочь в выявлении уязвимых сервисов, доступных для атак.

Существует несколько видов сканирования портов:

* **Полное сканирование** — проверяются все порты устройства (от 1 до 65535), что может занять значительное время.
* **Сканирование по выборке портов** — проверяются только наиболее часто используемые порты, например, 22 (SSH), 80 (HTTP), 443 (HTTPS).
* **SYN-сканирование** (полусоединение) — отправляются только пакеты с SYN-флагом, что позволяет быстро и эффективно выявить открытые порты, не устанавливая при этом полные соединения.

В нашем случае, сканирование портов используется для обнаружения потенциально подозрительной активности. Мы устанавливаем **порог сканирования**, который определяет, сколько уникальных портов должен попытаться подключить один и тот же IP-адрес за определённый промежуток времени, чтобы это считалось сканированием. Это помогает фильтровать легитимные запросы от агрессивных попыток исследовать уязвимости системы.

**Логирование**

Одним из важных элементов защиты сети является **логирование** подозрительной активности. В нашем проекте мы используем модуль Python logging для ведения журнала событий. Логирование позволяет администраторам сети отслеживать действия, которые могут быть признаками вредоносной активности, и анализировать инциденты безопасности, такие как попытки несанкционированного доступа или сканирования.

При обнаружении потенциальной угрозы, такой как сканирование портов, в лог файл записывается информация о соответствующем IP-адресе и характере действия (например, попытка сканирования). Это дает возможность:

* Поддерживать полную историю инцидентов для последующего анализа.
* Идентифицировать повторяющиеся угрозы или массовые атаки.
* Быстро реагировать на новые угрозы, обнаруженные в сети.

Логирование является критически важным компонентом в системе безопасности, так как оно даёт администраторам информацию для принятия обоснованных решений о действиях, таких как блокировка или ограничение доступа.

**Обработка пакетов**

Для мониторинга сетевого трафика используется функция **packet\_handler**, которая анализирует входящие пакеты. Эта функция проверяет каждый пакет на наличие определённых слоёв данных, таких как **IP-** и **TCP-слои**, которые содержат важную информацию о сетевых соединениях.

1. **IP-слой**: Он содержит исходный и целевой IP-адреса, что позволяет идентифицировать источник и получателя пакета.
2. **TCP-слой**: В нем содержится информация о порте назначения, который может быть использован для определения, пытается ли устройство подключиться к определенному сервису.

При анализе трафика наша программа фиксирует все попытки подключения к различным портам. Если количество уникальных портов, к которым пытается подключиться один и тот же IP-адрес, превышает установленный порог, это интерпретируется как возможное **сканирование портов**.

Этот подход позволяет эффективно отслеживать аномальную активность, которая может быть связана с попытками взлома или разведки системы. Если, например, с одного IP-адреса происходит множество попыток подключения к различным портам за короткий промежуток времени, это является явным признаком сканирования.

**Блокировка IP-адресов**

Для защиты от подозрительных попыток подключения мы реализуем механизм **блокировки IP-адресов**, которые пытаются провести сканирование. Как только подозрительная активность обнаружена (например, IP-адрес пытается подключиться к большому количеству портов), наша система автоматически блокирует этот IP-адрес.

Для блокировки мы используем механизм отправки **ICMP-сообщений** (Internet Control Message Protocol). ICMP — это протокол, используемый для обмена диагностической информацией, включая сообщения об ошибках и недоступности хоста. В случае блокировки подозрительного IP, отправляется ICMP-сообщение с кодом **"Destination Unreachable"** (код 1), что означает, что хост не может быть достигнут. Это действие эффективно прерывает дальнейшие попытки сканирования со стороны злоумышленника.

Применение ICMP для блокировки позволяет не только прекратить нежелательные соединения, но и сигнализировать злоумышленнику о том, что его действия были замечены и заблокированы, что может побудить его отказаться от дальнейших атак.

# Описание программы

Программа написана на python и разделена на модули, выполняющие отдельные функции:

исходный код: traffic\_blocking.py

Таблица 1. Описание исходного кода

|  |  |  |
| --- | --- | --- |
| Название функции | Входные данные | Описание |
| log\_suspicious\_activity | ip (str): IP-адрес, с которого поступила подозрительная активность.  activity (str): Описание подозрительного действия (например, "Port scan detected"). | Эта функция записывает в лог файл информацию о подозрительной активности, такой как обнаружение сканирования портов. Функция использует модуль logging для записи сообщений в файл traffic\_logs.log. |
| packet\_handler | packet (Packet): Сетевой пакет, полученный от функции sniff() из Scapy. | Функция обрабатывает входящие пакеты, проверяя наличие IP- и TCP-слоев. Если в пакете обнаружен IP-слой, она анализирует исходный IP-адрес. Если присутствует TCP-слой, то проверяется порт назначения. Для каждого IP-адреса отслеживаются попытки подключения к различным портам, и если количество уникальных портов превышает заданный порог (в данном случае port\_scan\_threshold), это расценивается как попытка сканирования, и IP-адрес блокируется. |
| block\_ip | ip\_address (str): IP-адрес, который необходимо заблокировать. | Функция отправляет ICMP-сообщение о недостижимости (код 3, тип 1) для заблокированного IP-адреса. Это предотвращает дальнейшие попытки сканирования со стороны этого IP. Блокировка осуществляется с помощью функции send() из библиотеки Scapy, которая отправляет ICMP-пакет с соответствующими параметрами. |
| start\_monitoring | () | Эта функция запускает процесс мониторинга сетевого трафика с помощью Scapy. Функция sniff() используется для перехвата пакетов, которые передаются в функцию packet\_handler для обработки. Функция будет работать в фоновом режиме, пока не будет завершена вручную или программно. |
| reset\_scan\_attempts | () | Функция сбрасывает все отслеживаемые попытки сканирования портов. Она очищает словарь scan\_attempts, в котором хранятся IP-адреса и соответствующие порты. Эта функция может быть полезна для периодической очистки данных, чтобы предотвратить избыточное использование памяти и перезапуск отслеживания активности. |

# 

# Основные шаги программы

1. Запуск цикла мониторинга: Программа использует библиотеку Scapy для перехвата пакетов, анализирует трафик и блокирует IP-адреса с подозрительными активностями (например, попытки сканирования портов).
2. Остановка при сигнале: Программа завершает свою работу при получении сигнала для корректного выхода и освобождения ресурсов.

# Рекомендации пользователя

1. Настройка окружения:

Убедитесь, что у вас установлена библиотека Scapy. Для этого выполните команду pip install scapy в терминале. Также рекомендуется запускать программу с правами администратора, чтобы обеспечить доступ к сетевым интерфейсам.

2. Конфигурация порога сканирования:

Порог для обнаружения сканирования портов можно настроить в переменной port\_scan\_threshold. Рекомендуется установить значение, соответствующее вашему сетевому окружению. Например, для небольших сетей можно установить порог на уровне 1-2, а для более крупных сетей — 3-5.

3. Мониторинг логов:

Программа записывает подозрительную активность в файл traffic\_logs.log. Регулярно проверяйте этот файл для анализа инцидентов и выявления потенциальных угроз. Это поможет вам лучше понять поведение сети и реагировать на возможные атаки.

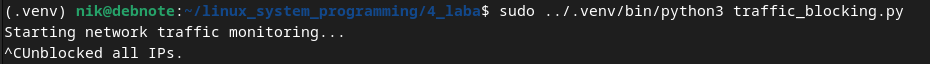
4. Тестирование программы:

Перед использованием в реальной сети протестируйте программу в контролируемой среде. Это позволит вам убедиться в корректности работы и настроить параметры под ваши нужды.

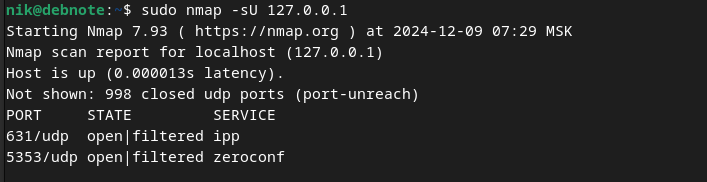
# Рекомендации программиста

* Следите за актуальностью используемых библиотек
* Очищайте лог, когда это необходимо

# Контрольный пример

Рис1. Пример запуска

Пример генерации подозрительного трафика.

Рис2. Пример использования nmap

Пример заблокированных ip через iptables.

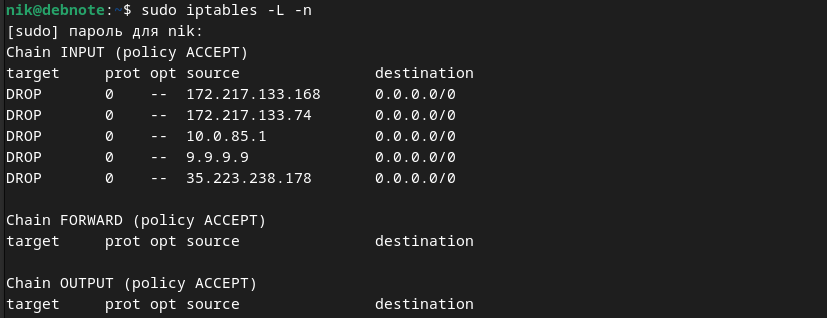
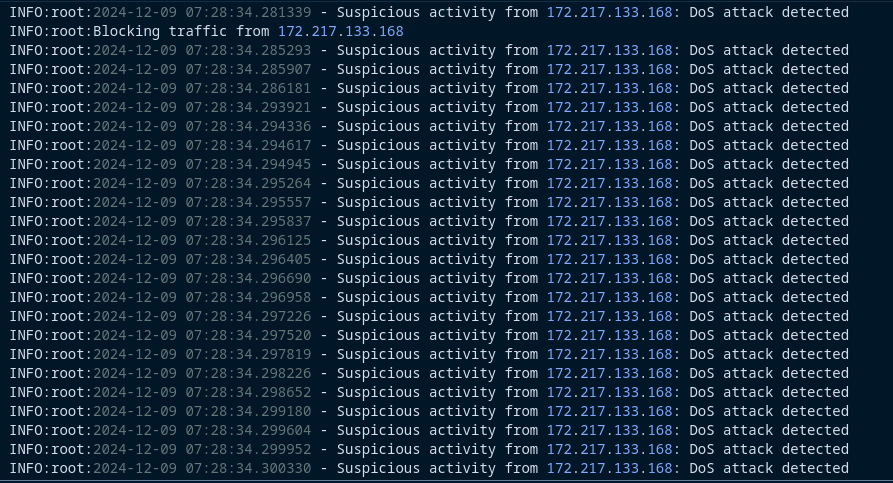


Рис3. Пример вывода заблокированных источников

Пример логов для сгенерированного трафика.

Рис4. Пример traffic\_logs.log

# Вывод

В ходе выполнения лабораторной работы была разработана программа для мониторинга сетевого трафика с использованием библиотеки Scapy. Основной целью работы было создание инструмента, способного обнаруживать подозрительную активность, такую как сканирование портов, и автоматически реагировать на нее путем блокировки IP-адресов.

Результаты работы показали, что программа успешно выполняет поставленные задачи. Она корректно фиксирует попытки сканирования, записывает информацию о подозрительной активности в лог-файл и блокирует IP-адреса, с которых происходят атаки. Проведенные контрольные примеры подтвердили эффективность алгоритмов обнаружения и блокировки, а также правильность работы логирования.

# Исходный код

Посмотреть все используемые файлы можно на GitHub.  
https://github.com/nikitopus/functional\_programming\_24/tree/master/4\_laba

# Список источников

1. Документация Scapy

Scapy — это мощный инструмент для манипуляции сетевыми пакетами в Python, который используется для перехвата, создания и анализа сетевого трафика. Официальная документация Scapy содержит подробную информацию о функциях и методах, доступных для работы с пакетами.

Источник: Scapy Documentation

1. RFC 791 - Internet Protocol

В этом документе описаны основные принципы работы IP-протокола, включая структуру IP-адресов и обработку пакетов. Понимание структуры и особенностей IP-пакетов важно для правильного анализа сетевого трафика.

Источник: RFC 791

1. "Linux Firewalls" by Robert Ziegler

Эта книга предоставляет подробное руководство по настройке фаерволов в Linux с использованием iptables. В ней рассматриваются способы фильтрации и блокировки трафика, что полезно для расширения функционала программы, например, для блокировки подозрительных IP-адресов.

Источник: Linux Firewalls: Enhancing Security with nftables and Beyond