**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**

**Абрахин Е.Д.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc178975364)
2. [Описание задачи (формализация задачи) 4](#_Toc178975365)
3. [Теоретическая часть 6](#_Toc178975366)
4. [Основные шаги программы 9](#_Toc178975367)
5. [Описание программы 11](#_Toc178975368)
6. [Рекомендации пользователя 13](#_Toc178975369)
7. [Рекомендации программиста 14](#_Toc178975370)
8. [Исходный код программы 15](#_Toc178975371)
9. [Контрольный пример 16](#_Toc178975372)
10. [Вывод 16](#_Toc178975373)

# Цель работы

Целью данной работы является разработка механизма для обнаружения и блокировки подозрительного сетевого трафика на основе заранее определенных правил и сигнатур. Программа должна эффективно отслеживать сетевой трафик, анализировать его параметры и при выявлении аномалий блокировать подозрительную активность, тем самым обеспечивая базовую защиту сети от потенциальных угроз.

# Описание задачи (формализация задачи)

Необходимо разработать программное обеспечение, которое будет анализировать сетевой трафик в реальном времени, выявлять подозрительные действия на основе заданных параметров и блокировать их.

Задача делится на несколько подзадач:

1. Установка и настройка инструментов:

Установить и настроить библиотеку Scapy для работы с сетевыми пакетами на Python. Scapy будет служить основным инструментом для захвата и анализа сетевого трафика.

1. Обнаружение и анализ трафика:

Разработать скрипт, который будет пассивно слушать сетевой трафик, захватывая пакеты для анализа.

Программа должна проверять сетевые пакеты на соответствие заданным правилам и сигнатурам, таким как определенные IP-адреса, порты, заголовки, типы пакетов и т. д.

1. Определение подозрительного трафика:

Реализовать механизм для выявления аномалий в сетевом трафике. Например, определять попытки сканирования портов, большое количество пакетов от одного источника (флуд), подозрительно крупные пакеты и повторяющиеся запросы.

Программа должна содержать гибкие правила для идентификации различных типов подозрительных действий.

1. Блокировка подозрительного трафика:

После обнаружения подозрительных пакетов, программа должна автоматически принимать меры по блокировке. Возможные действия включают отправку ICMP-сообщений о недостижимости для блокировки IP-адресов или определенных портов.

# Теоретическая часть

Современные вычислительные системы и сети ежедневно сталкиваются с разнообразными угрозами безопасности, включая сетевые атаки, сканирование уязвимостей и попытки несанкционированного доступа. В условиях увеличивающегося числа и сложности атак ручной мониторинг сетевого трафика становится неэффективным. Автоматизация обнаружения и блокировки подозрительного трафика с помощью специализированных механизмов помогает защитить систему и минимизировать человеческое участие в процессе защиты сети.

Анализ сетевого трафика

Сетевой трафик — это поток данных, передаваемый между устройствами в сети. Анализ трафика позволяет выявлять аномальные и подозрительные действия, такие как несанкционированный доступ, попытки вторжения или сетевые атаки. Для выполнения этой задачи широко применяются инструменты для захвата и анализа сетевых пакетов. Одним из таких инструментов является библиотека Scapy, позволяющая на Python анализировать пакеты на низком уровне, предоставляя полную информацию о содержимом трафика.

Сетевые угрозы и виды атак

Существует множество типов атак, которые можно обнаружить на основе анализа сетевого трафика:

* Сканирование портов: попытка определить открытые порты на хосте с целью поиска уязвимостей.
* Флуд-атаки (DDoS): перегрузка системы с огромным количеством запросов для нарушения её работы.
* Атаки с подменой IP (IP Spoofing): изменение IP-адресов пакетов с целью маскировки источника атаки.
* Аномальные пакеты: слишком большие пакеты или пакеты, не соответствующие стандартам протоколов, могут свидетельствовать об атаке.

Эти и другие угрозы можно распознавать с помощью анализа ключевых параметров сетевых пакетов, таких как IP-адреса, порты, заголовки, протоколы и частота отправки пакетов.

Автоматизация процесса обнаружения и блокировки трафика

Автоматизация защиты от сетевых угроз позволяет эффективно и непрерывно контролировать сетевую активность без постоянного участия администратора. Создание скрипта для автоматического анализа и блокировки подозрительного трафика упрощает этот процесс и снижает риски от атак. Основные задачи, которые решаются при разработке системы обнаружения и блокировки трафика:

1. Сбор и анализ сетевого трафика:

Система должна захватывать все входящие и исходящие сетевые пакеты для их анализа. С использованием библиотеки Scapy можно просматривать содержимое пакетов на уровне сетевых заголовков и данных, что позволяет выявлять аномалии или сигнатуры атак.

1. Обнаружение подозрительного трафика:

Программа должна уметь распознавать подозрительные действия, такие как частые повторяющиеся запросы с одного IP-адреса, флуд с целью перегрузки системы или сканирование портов. Правила для определения таких событий настраиваются в зависимости от специфики работы сети.

1. Блокировка сетевых атак:

При обнаружении аномального трафика программа должна блокировать дальнейшие попытки взаимодействия.

Программа для анализа и блокировки сетевого трафика должна следовать следующим этапам:

1. Инициализация и настройка:

Сначала система инициализируется и настраивается для захвата трафика с сетевого интерфейса. Scapy предоставляет возможности для пассивного прослушивания всех пакетов, проходящих через сеть.

1. Обработка пакетов:

После захвата каждый пакет анализируется на предмет соответствия определенным правилам и сигнатурам. Например, проверяются заголовки на наличие аномалий или соответствие известным паттернам атак.

1. Выявление угроз:

При обнаружении пакетов, соответствующих правилам для подозрительного трафика, система фиксирует событие и принимает меры по блокировке дальнейших действий.

1. Блокировка подозрительного трафика:

В зависимости от типа угрозы программа отправляет ICMP-сообщения о недостижимости или добавляет IP-адреса в список блокировки. Это предотвращает дальнейшее взаимодействие с подозрительными хостами.

Таким образом, автоматизация процесса обнаружения и блокировки подозрительного трафика с использованием Scapy и правил анализа пакетов позволяет существенно повысить безопасность сети и минимизировать риски, связанные с сетевыми атаками.

# Основные шаги программы

1. Запуск программы

Программа запускается в графическом интерфейсе с помощью библиотеки Tkinter. Интерфейс позволяет пользователю запускать/останавливать процесс сниффинга сетевого трафика, а также управлять подозрительными и заблокированными IP-адресами. При нажатии на кнопку "Запустить Sniffing" вызывается функция toggleSniffing(), которая активирует или деактивирует процесс захвата трафика.

1. Анализ сетевого трафика

Функция sniffTraffic() с использованием библиотеки Scapy производит захват сетевых пакетов с сетевого интерфейса. Пакеты передаются на обработку в функцию packetCallback(), которая анализирует содержимое пакета и проверяет его на соответствие критериям подозрительности.

1. Обработка пакетов

В функции packetCallback() программа:

* Добавляет информацию о каждом захваченном пакете в список capturedPackets и обновляет графический элемент Listbox, отображающий все захваченные пакеты.
* Проверяет, является ли пакет подозрительным, с помощью функции isSuspicious(). Если пакет соответствует критериям (большой размер или использование подозрительных портов), его IP-адрес добавляется в список подозрительных IP-адресов и обновляется соответствующий интерфейс.

1. Определение подозрительных пакетов

Функция isSuspicious() проверяет пакеты на наличие потенциальных угроз. Подозрительным считается пакет, если:

* Он использует подозрительные порты.
* Его размер превышает определенное значение.

1. Обновление интерфейса

Функции updatePacketList(), updateSuspiciousList() и updateBlockedList() отвечают за обновление соответствующих списков в пользовательском интерфейсе (захваченные пакеты, подозрительные IP-адреса и заблокированные IP-адреса).

При добавлении новых данных списки обновляются в реальном времени, что позволяет пользователю видеть изменения.

1. Блокировка подозрительных IP-адресов

Пользователь может выбрать подозрительный IP-адрес в интерфейсе и нажать кнопку "Заблокировать". Функция blockIP() добавляет этот IP-адрес в список заблокированных и выполняет системную команду iptables, чтобы заблокировать трафик от этого IP.

Если IP-адрес уже был заблокирован, программа уведомляет пользователя.

1. Разблокировка IP-адресов

Если пользователь решает разблокировать IP-адрес, выбирается соответствующий IP в списке заблокированных, при нажатии на кнопку "Разблокировать" вызывается функция unblockIP(). Программа удаляет IP-адрес из списка заблокированных и разблокирует его с помощью iptables.

1. Цикл захвата трафика

Процесс захвата сетевого трафика продолжается до тех пор, пока пользователь не остановит его. Функция sniffTraffic() запускается в отдельном потоке с использованием библиотеки threading, что позволяет программе не блокировать основной интерфейс и продолжать работу в фоновом режиме.

1. Интерактивное управление процессом

Пользователь имеет полный контроль над процессом мониторинга трафика через графический интерфейс. Он может запускать/останавливать захват пакетов, просматривать подозрительные IP-адреса и блокировать или разблокировать их.

# Описание программы

Программа написана на языке Python 3.12.6 и предназначена для мониторинга сетевого трафика с возможностью блокировки подозрительных IP-адресов в реальном времени. Она реализует графический интерфейс с помощью библиотеки Tkinter и использует библиотеку Scapy для захвата и анализа сетевых пакетов. Пользователь может наблюдать за трафиком, выявлять подозрительные IP-адреса, блокировать их через iptables и разблокировать, если потребуется.

Основные библиотеки:

* os — для выполнения системных команд (блокировка/разблокировка IP через iptables).
* threading — для параллельного выполнения захвата пакетов в фоне.
* scapy.all — для захвата и анализа сетевых пакетов.
* tkinter — для создания графического интерфейса пользователя.

Программа организована в единый модуль TrafficScanner.py, который включает в себя основные функции по захвату трафика, управлению списками подозрительных и заблокированных IP-адресов, а также взаимодействие с пользователем через интерфейс.

Таблица 1. TrafficScanner.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| isSuspicious | Определяет, является ли сетевой пакет подозрительным (по порту или размеру пакета). | True/False |
| packetCallback | Вызывается при каждом захваченном пакете. Обрабатывает пакет, обновляет списки пакетов и подозрительных IP-адресов. | None |
| blockIP | Блокирует выбранный IP-адрес с помощью команды iptables и обновляет список заблокированных IP-адресов. | None |
| unblockIP | Разблокирует выбранный IP-адрес, удаляя его из правил iptables, и обновляет список. | None |
| updatePacketList | Обновляет список захваченных пакетов в графическом интерфейсе. | None |
| updateSuspiciousList | Обновляет список подозрительных IP-адресов в графическом интерфейсе. | None |
| updateBlockedList | Обновляет список заблокированных IP-адресов в графическом интерфейсе. | None |
| sniffTraffic | Захватывает сетевой трафик с помощью Scapy и передает пакеты для анализа. | None |
| toggleSniffing | Управляет началом и остановкой процесса захвата трафика. | None |

# Рекомендации пользователя

1. Запуск программы:

Для запуска программы выполните следующую команду в терминале:

python3 TrafficScanner.py

Программа откроет окно с интерфейсом для мониторинга сетевого трафика, обнаружения подозрительных IP-адресов и их блокировки.

1. Мониторинг сетевого трафика:

Для начала мониторинга трафика нажмите кнопку "Запустить Sniffing". Программа начнет захват пакетов.

1. Блокировка подозрительных IP-адресов:

Программа автоматически определяет подозрительные пакеты по их размеру или портам. Подозрительные IP-адреса добавляются в список "Подозрительные IP". Для блокировки IP выберите адрес из списка и нажмите кнопку "Заблокировать". Это добавит правило блокировки в iptables.

1. Разблокировка IP-адресов:

Заблокированные IP-адреса отображаются в списке "Заблокированные IP". Чтобы разблокировать IP-адрес, выберите его и нажмите кнопку "Разблокировать". Правило блокировки будет удалено из iptables.

1. Остановка сниффинга:

Чтобы остановить захват трафика, нажмите кнопку "Остановить Sniffing". Программа прекратит захват пакетов.

# Рекомендации программиста

Программа использует Scapy для анализа сетевого трафика и Tkinter для графического интерфейса. Убедитесь, что эти библиотеки установлены:

sudo apt-get install python3-tk

sudo pip install scapy

Программа производит по умолчанию захват пакетов с интерфейса enp8s0. При необходимости измените сетевой интерфейс на нужный в функции sniffTraffic():

# Исходный код программы

[https://github.com/FasterXaos/System\_programming\_in\_Linux.git](https://github.com/FasterXaos/System_programming_in_Linux.git%20)

# Контрольный пример

1. Запуск программы

Для запуска программы необходимо запустить выполнить следующую команду в терминале:

python3 TrafficScanner.py

1. Начало мониторинга трафика

После запуска программы откроется окно с графическим интерфейсом. Для начала мониторинга сетевого трафика нажмите кнопку "Запустить Sniffing". Программа начнет захват сетевых пакетов с интерфейса, указанного в коде (по умолчанию enp8s0).

1. Просмотр захваченных пакетов

В левой части интерфейса будет отображен список всех захваченных пакетов. Каждая строка представляет краткое описание захваченного пакета (например, IP-адреса источника и назначения, протокол и размер).

1. Обнаружение подозрительных IP-адресов

Программа автоматически добавляет IP-адреса, отправляющие подозрительные пакеты, в список "Подозрительные IP" (в центральной части окна). Подозрительные пакеты определяются по размеру (более 1000 байт) или по порту (22, 23, 80, 443).

1. Блокировка IP-адресов

Чтобы заблокировать IP-адрес, выберите его в списке "Подозрительные IP" и нажмите кнопку "Заблокировать". Программа выполнит команду iptables, чтобы заблокировать весь входящий трафик с этого IP-адреса. Заблокированные IP-адреса появятся в списке "Заблокированные IP" (справа).

1. Разблокировка IP-адресов

Для разблокировки IP-адреса выберите его в списке "Заблокированные IP" и нажмите кнопку "Разблокировать". Это удалит правило блокировки для этого IP-адреса в iptables.

1. Остановка мониторинга трафика

Чтобы остановить захват трафика, нажмите кнопку "Остановить Sniffing". Программа прекратит захват сетевых пакетов и обновление списка.

1. Закрытие программы

Чтобы выйти из программы, просто закройте окно, нажав на кнопку "Закрыть" в правом верхнем углу окна.

# Вывод

В ходе выполнения данной работы была разработана программа-демон для автоматизированного создания резервных копий с использованием системных средств Linux. Основной задачей было создать эффективную и надежную систему для регулярного резервного копирования данных, работающую в фоновом режиме и требующую минимального вмешательства со стороны пользователя.

Программа реализована на языке C++ с использованием многопоточной обработки и системных вызовов для создания резервных копий с помощью утилиты rsync. Основные функции программы включают в себя создание резервных копий по заданному расписанию, мониторинг состояния демона, а также ведение логов для отслеживания статуса и возможных ошибок.

Был реализован механизм автоматического запуска и остановки демона через систему systemd, что позволило интегрировать решение в стандартную инфраструктуру Linux для управления службами. Программа поддерживает параллельную обработку данных и гарантирует надежное сохранение копий, исключая из копирования системные директории, которые могут содержать динамические или временные файлы.

Разработанное решение предоставляет пользователю удобный инструмент для автоматизации процессов резервного копирования, обеспечивает сохранность данных и повышает надежность системного администрирования.

# Литература

1. os — Miscellaneous operating system interfaces // docs.python.org URL: https://docs.python.org/3/library/os.html (дата обращения: 22.10.2024).
2. threading — Thread-based parallelism // docs.python.org URL: https://docs.python.org/3/library/threading.html (дата обращения: 22.10.2024).
3. scapy.all — Sniffing and dissecting network packets // scapy.readthedocs.io URL: https://scapy.readthedocs.io/en/latest/usage.html (дата обращения: 22.10.2024).
4. tkinter — Python interface to Tcl/Tk // docs.python.org URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 22.10.2024).
5. iptables — Administration tool for IPv4 packet filtering and NAT // linux.die.net URL: https://linux.die.net/man/8/iptables (дата обращения: 22.10.2024).