Санкт-Петербургский государственный университет

Направление ”Большие данные и распределенная цифровая платформа”

Лабораторная работа по дисциплине
Системное программирование в Linux

”Создание программы для обнаружения и блокировки подозрительного сетевого трафика.”

Выполнил:
Зайнуллин Мансур Альбертович
Группа: 23.Б16-пу

Руководитель:
Киямов Жасур Уткирович

Санкт-Петербург
2024

Оглавление

1 Цель работы **2**

2 Описание задачи **3**

3 Теоретическая часть **5**

3.1 Основы сетевого трафика 5

3.2 IP-адреса и порты 5

3.3 Протоколы и их роль 5

3.4 Обнаружение сетевых угроз 6

3.5 Блокировка сетевого трафика 6

3.6 Демоны в системах Unix/Linux 6

3.7 Управление демонами с помощью systemd 7

3.8 Инструменты для анализа сетевого трафика 7

3.9 Принципы работы сетевого монитора 7

4 Описание программы **9**

4.1 Класс `NetworkMonitor` 9

4.2 Класс `Application` 10

4.3 Алгоритм работы программы 10

5 Рекомендации пользователю **12**

5.1 Инструкция по установке сканера на Ubuntu 24 12

5.2 Инструкция по эксплуатации сканера 13

6 Контрольный пример **16**

7 Вывод по работе **18**

8 Полезные ссылки **19**

1 Цель работы

Цель данной работы разработать простой механизм обнаружения и блокировки подозрительного сетевого трафика на основе определенных правил и сигнатур.

2 Описание задачи

Разработать механизм для обнаружения и блокировки подозрительного сетевого трафика на основе заданных правил и сигнатур. Программа должна выполнять следующие функции:

- **Установка необходимых библиотек:**Установить библиотеку Scapy для анализа сетевого трафика и взаимодействия с сетевыми пакетами на Python.
- **Создание скрипта для обнаружения и анализа трафика** Разработать скрипт на Python, который будет прослушивать сетевой трафик на указанном сетевом интерфейсе и анализировать пакеты с помощью Scapy. Анализ должен основываться на различных параметрах пакетов, таких как IP-адреса источника и назначения, номера портов, размер пакета и заголовки протоколов.
- **Определение подозрительного трафика** Реализовать логику для выявления подозрительных пакетов. Подозрительным трафиком считается:
  - Пакеты с IP-адресов из списка подозрительных.
  - Пакеты, направленные на подозрительные порты.
  - Пакеты, превышающие заданный размер.
  - Повторяющиеся запросы или сканирование портов с одного IP-адреса.
- **Блокировка подозрительного трафика** После обнаружения подозрительного пакета программа должна блокировать соответствующий IP-адрес. Блокировка осуществляется путем отправки ICMP-сообщений типа ”Destination Unreachable” на данный IP-адрес.
- **Тестирование и настройка** Провести тестирование скрипта на различных видах сетевого трафика для проверки корректности

обнаружения и блокировки подозрительных активностей. Настроить параметры скрипта для обеспечения оптимальной работы в различных сетевых условиях.

- **Непрерывное обновление и мониторинг** Обеспечить возможность обновления правил обнаружения и блокировки в соответствии с новыми угрозами и сценариями атак. Реализовать механизм мониторинга работы скрипта и анализа логов для выявления аномалий.

3 Теоретическая часть

3.1 Основы сетевого трафика

Сетевой трафик представляет собой обмен данными между устройствами в компьютерной сети. Он состоит из пакетов данных, которые передаются по различным протоколам, таким как TCP, UDP, ICMP и другим. Каждый пакет содержит заголовки, содержащие информацию о маршрутизации, а также полезную нагрузку.

3.2 IP-адреса и порты

IP-адрес — уникальный идентификатор устройства в сети, позволяющий маршрутизировать пакеты данных между источником и назначением. Существует два основных типа IP-адресов: IPv4 и IPv6.

Порты используются для идентификации конкретных процессов или сервисов на устройстве. Каждый порт ассоциирован с определенным протоколом и службой (например, порт 80 для HTTP, порт 443 для HTTPS).

3.3 Протоколы и их роль

- **TCP (Transmission Control Protocol)**: Обеспечивает надежную передачу данных, устанавливая соединение между источником и получателем. Используется для приложений, требующих гарантированной доставки данных.
- **UDP (User Datagram Protocol)**: Предоставляет менее надежную, но более быструю передачу данных без установления соединения. Применяется для приложений, где важна скорость передачи.
- **ICMP (Internet Control Message Protocol)**: Используется для обмена служебной информацией о состоянии сети, например, для передачи сообщений об ошибках или о недоступности узлов.

3.4 Обнаружение сетевых угроз

Обнаружение сетевых угроз включает в себя идентификацию аномалий и подозрительных активностей в сетевом трафике. Основные методы включают:

- **Анализ сигнатур**: Сравнение пакетов с известными шаблонами атак.
- **Анализ аномалий**: Выявление отклонений от нормального поведения сети, таких как необычный объем трафика или нестандартные порты.
- **Машинное обучение**: Применение алгоритмов для классификации трафика и обнаружения новых типов угроз.

3.5 Блокировка сетевого трафика

Блокировка сетевого трафика осуществляется для предотвращения доступа подозрительных устройств к сети. Основные методы блокировки включают:

- **Отправка ICMP-сообщений**: Информирование источника о недоступности сети или ресурса.
- **Использование iptables**: Настройка правил брандмауэра для блокировки трафика с определенных IP-адресов или портов.
- **Программные фильтры**: Реализация блокировки на уровне приложений с помощью скриптов и библиотек, таких как Scapy.

3.6 Демоны в системах Unix/Linux

Демон — это фоновый процесс в операционных системах Unix и Linux, выполняющий системные задачи без непосредственного взаимодействия с пользователем. Демоны обычно запускаются при старте системы и продолжают работать до ее выключения. Они обеспечивают непрерывную работу сервисов, таких как мониторинг сети, управление файлами, обслуживание запросов и другие системные функции.

3.7 Управление демонами с помощью systemd

В современных системах Linux управление демонами осуществляется с помощью `systemd`. Основные операции включают:

- **Создание файла юнита**: Определение конфигурации демона в файле с расширением `.service`.
- **Установка и запуск демона**: Копирование файла юнита в каталог `/etc/systemd/system/` и использование команд `systemctl` для управления демоном.
- **Мониторинг и управление состоянием**: Проверка статуса демона, его остановка, перезапуск и настройка автозапуска при старте системы.

3.8 Инструменты для анализа сетевого трафика

Для анализа сетевого трафика используются различные инструменты и библиотеки:

- **Scapy**: Библиотека на Python для создания, отправки, захвата и анализа сетевых пакетов. Позволяет гибко настраивать параметры пакетов и автоматизировать процессы анализа.
- **Wireshark**: Графический анализатор сетевых пакетов с поддержкой широкого спектра протоколов и возможностью фильтрации данных.
- **tcpdump**: Командный инструмент для захвата и анализа сетевого трафика в реальном времени. Подходит для быстрого мониторинга и отладки сетевых проблем.

3.9 Принципы работы сетевого монитора

Сетевой монитор, разработанный в рамках данной работы, выполняет следующие действия:

1. **Прослушивание сетевого трафика**: С помощью Scapy скрипт захватывает пакеты, проходящие через указанный сетевой интерфейс.

2. **Анализ пакетов**: Каждый пакет анализируется на соответствие заданным правилам, таким как подозрительные IP-адреса, порты, размер пакета и признаки сканирования портов.

3. **Обнаружение подозрительной активности**: При совпадении параметров пакета с критериями подозрительности IP-адрес добавляется в список заблокированных.

4. **Блокировка IP-адресов**: Для каждого подозрительного IP-адреса отправляется ICMP-сообщение ”Destination Unreachable” и при необходимости добавляются правила в iptables для полного блокирования трафика с этого IP.

5. **Логирование**: Все обнаруженные и заблокированные активности записываются в лог-файл и отображаются в графическом интерфейсе для мониторинга.

6. **Управление и мониторинг**: Пользователь может запускать и останавливать мониторинг через интерфейс, а также просматривать логи для анализа сетевых событий.

4 Описание программы

Программа состоит из двух основных компонентов: класса `NetworkMonitor` для мониторинга сетевого трафика и класса `Application` для графического интерфейса пользователя.

4.1 Класс `NetworkMonitor`

- `sniff_packets(self)`: Начинает прослушивание пакетов на указанном сетевом интерфейсе с использованием библиотеки Scapy. Для каждого пойманного пакета вызывается метод `process_packet`.
- `process_packet(self, packet)`: Обрабатывает каждый пакет, проверяя его на соответствие заданным правилам. Выполняет следующие проверки:
  - **Подозрительные IP-адреса**: Если IP-адрес источника находится в списке подозрительных, блокирует этот IP.
  - **Подозрительные порты**: Проверяет исходящие и входящие порты пакета на соответствие списку подозрительных портов. При совпадении блокирует IP-адрес источника.
  - **Размер пакета**: Если размер пакета превышает заданный порог, блокирует IP-адрес источника.
  - **Сканирование портов**: Отслеживает количество различных портов, на которые отправляются пакеты с одного IP-адреса. Если количество превышает заданный порог, блокирует IP-адрес источника.
- `block_ip(self, ip)`: Блокирует указанный IP-адрес, добавляя его в список заблокированных и отправляя ICMP-сообщение типа ”Destination Unreachable”.
  - **Параметры:**
    - `ip`: IP-адрес, который необходимо заблокировать.

4.2 Класс `Application`

4.3 Алгоритм работы программы

1. **Инициализация**:
    - При запуске программы загружается конфигурационный файл `config.json`, содержащий настройки мониторинга, такие как сетевой интерфейс и правила обнаружения подозрительного трафика.
    - Создается графический интерфейс пользователя с элементами для отображения логов и управления мониторингом.
2. **Запуск мониторинга**:
    - Пользователь нажимает кнопку ”Запустить Монитор”.
    - Создается экземпляр класса `NetworkMonitor` с загруженной конфигурацией и функцией логирования.
    - Начинается прослушивание сетевого трафика на указанном интерфейсе в отдельном потоке.
3. **Обработка пакетов**:
    - Для каждого пойманного пакета выполняется проверка на соответствие подозрительным параметрам (IP-адреса, порты, размер, сканирование портов).
    - При обнаружении подозрительного трафика IP-адрес блокируется путем отправки ICMP-сообщения.
4. **Логирование**:
    - Все обнаруженные и заблокированные активности записываются в область логов интерфейса для мониторинга.
5. **Остановка мониторинга**:

- Пользователь может попытаться остановить мониторинг, однако программа информирует о невозможности этого действия без перезапуска.

5 Рекомендации пользователю

5.1 Инструкция по установке сканера на Ubuntu 24

Следуйте этим шагам для установки и настройки сетевого монитора на системе Ubuntu 24:

1. **Установка Python 3.9**

   Ubuntu 24 может поставляться с Python 3.9 по умолчанию. Проверьте установленную версию:

   ```
   python3 --version
   ```

   Если Python 3.9 не установлен, установите его:

   ```
   sudo apt-get update
   sudo apt-get install python3.9 python3.9-venv python3.9-dev
   ```

2. **Установка pip**

   Убедитесь, что pip установлен:

   ```
   python3.9 -m pip --version
   ```

   Если pip не установлен, установите его:

   ```
   sudo apt-get install python3-pip
   ```

3. **Установка Git**

   Если Git не установлен, установите его:

   ```
   sudo apt-get install git
   ```

4. **Клонирование репозитория проекта**

   Скачайте проект с GitHub:

   ```
   git clone https://github.com/MansurYa/ip_scanner.git
   cd ip_scanner
   ```

5. **Создание виртуального окружения (опционально)**

   Создайте и активируйте виртуальное окружение для изоляции зависимостей:

   ```
   python3.9 -m venv venv
   source venv/bin/activate
   ```

6. **Установка зависимостей**

   Установите необходимые библиотеки с помощью pip:

   ```
   sudo apt-get install build-essential libffi-dev python3-dev
   pip install --upgrade pip
   pip install scapy
   ```

   Примечание: Для блокировки IP-адресов через iptables требуется запуск программы с правами суперпользователя (sudo).

5.2 Инструкция по эксплуатации сканера

Следуйте этим шагам для настройки и использования сетевого монитора:

1. **Настройка файла config.json**

   Откройте файл `config.json` в текстовом редакторе и настройте параметры мониторинга:

```
{
    "interface": "ens3",
    "rules": {
        "suspicious_ips": ["192.168.1.100", "10.0.0.200"],
        "suspicious_ports": [23, 80],
        "max_packet_size": 1500,
        "port_scan_threshold": 5
    }
}
```

Описание полей:

- `interface`: Имя сетевого интерфейса для прослушивания (например, `ens3`). Узнать имя интерфейса можно с помощью команды:

  ```
  ifconfig
  ```

- `suspicious_ips`: Список IP-адресов, которые считаются подозрительными. Пакеты с этими адресами будут блокироваться.
- `suspicious_ports`: Список портов, обращение к которым считается подозрительным. Пакеты, направленные на эти порты, будут блокироваться.
- `max_packet_size`: Максимальный размер пакета в байтах. Пакеты, превышающие этот размер, будут считаться подозрительными.
- `port_scan_threshold`: Порог для определения сканирования портов. Если с одного IP-адреса отправлено более указанного количества пакетов на разные порты, IP будет блокироваться.

2. **Запуск программы**

Запустите программу с правами суперпользователя:

```
sudo python3 main.py
```

Примечание: Использование `sudo` необходимо для обеспечения доступа к сетевым интерфейсам и возможности блокировки IP-адресов через iptables.

3. **Использование графического интерфейса**

   После запуска программы откроется окно графического интерфейса с следующими элементами:

   - **Область логов**: Показывает текущую активность и события, такие как обнаружение и блокировка подозрительных IP-адресов.
   - **Статусный индикатор**: Отображает текущий статус мониторинга (например, “Монитор запущен”).
   - **Кнопки управления:**
     - “Запустить Монитор”: Начинает процесс мониторинга сетевого трафика.
     - “Остановить Монитор”: Попытка остановить мониторинг. Обратите внимание, что остановка невозможна без перезапуска программы.

4. **Просмотр логов**

   Все события мониторинга отображаются в области логов приложения и записываются в файл `network_monitor.log` в директории проекта. Вы можете просматривать этот файл для анализа активности.

5. **Завершение работы**

   Чтобы завершить работу программы, закройте окно графического интерфейса. Для остановки мониторинга потребуется перезапустить программу.

6 Контрольный пример

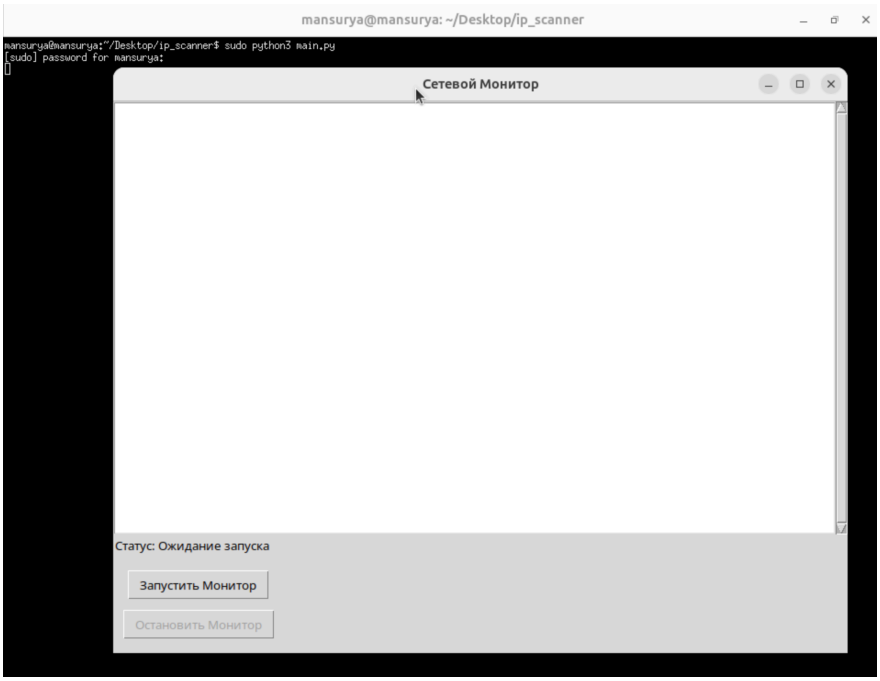

рис. 1 Запуск ip-scanner

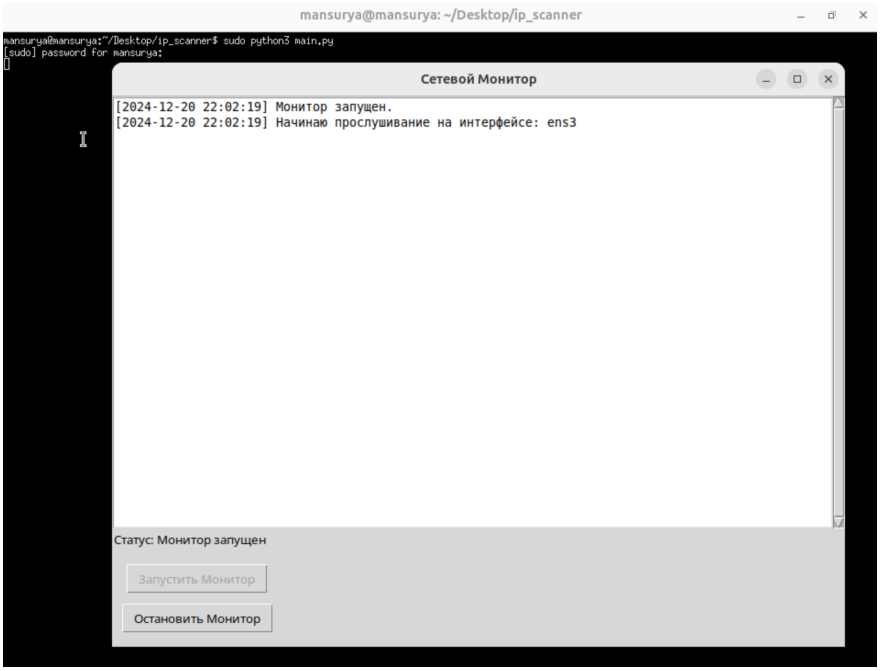

рис. 2 Запуск мониторинга

рис. 3 Запуск скрипта, который отправляет ICMP Echo Request (ping) от подозрительного IP

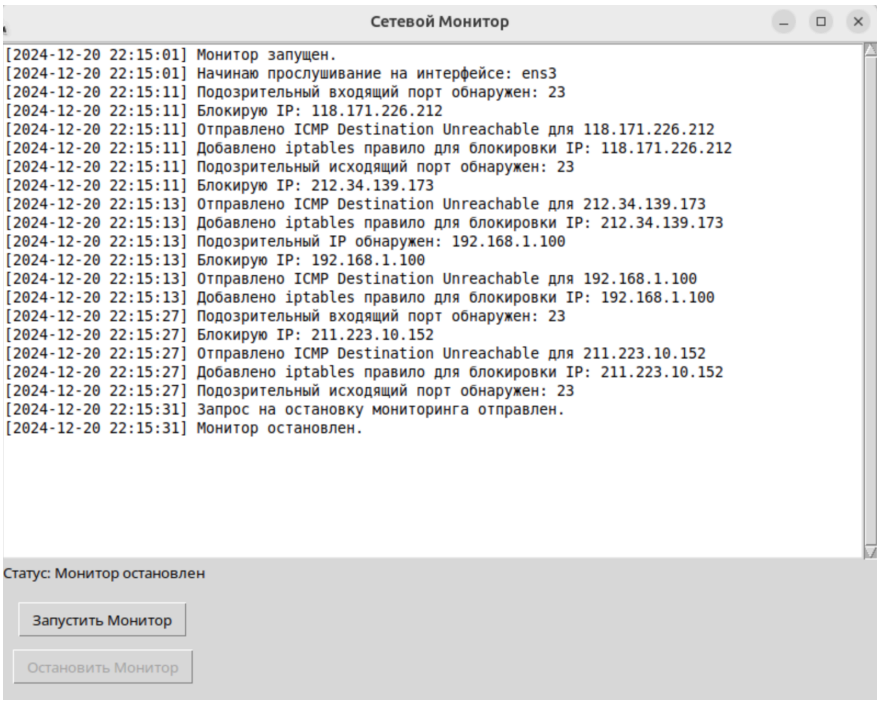

рис. 4 Подозрительный IP был заблокирован

7 Вывод по работе

В ходе работы был разработан скрипт для обнаружения и блокировки подозрительного сетевого трафика с использованием библиотеки Scapy. Реализованы функции мониторинга пакетов, проверки на основе правил и сигнатур, а также блокировки подозрительных IP-адресов. Программа также включает графический интерфейс для удобного управления процессом мониторинга. Тестирование показало, что система успешно обнаруживает и блокирует аномальный трафик в реальном времени.

8 Полезные ссылки

Документация для Scapy: https://scapy.readthedocs.io/en/latest/

Ссылка на репозиторий проекта: https://github.com/MansurYa/ip_scanner.git