

Сканирование сетевых портов при помощи Python и Nmap

Саяпин Никита Анатольевич

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается процесс анализа сетевых портов с использованием инструмента Nmap и языка программирования Python, включая практическую часть с примерами кода. Описываются основные этапы сканирования сети, от настройки инструментов до интерпретации результатов, а также предоставляются рекомендации по написанию скриптов на Python для автоматизации анализа. Особое внимание уделяется методам безопасного и эффективного сканирования портов, а также практическим подходам к обработке полученных данных.

Ключевые слова: сетевое сканирование, Nmap, Python, Анализ портов, Автоматизация, Сетевые протоколы, Безопасность сети.

Scanning network ports using Python and Nmap

Sayapin Nikita Anatolyevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the process of analyzing network ports using the Nmap tool and the Python programming language, including a practical part with code examples. It describes the main stages of network scanning, from configuring tools to interpreting results, and provides recommendations for writing Python scripts to automate analysis. Special attention is paid to methods of safe and efficient port scanning, as well as practical approaches to processing the received data.

Keywords: network scanning, Nmap, Python, Port Analysis, Automation, Network protocols, Network Security.

1. Введение

1.1 Актуальность

С развитием информационных технологий и увеличением числа киберугроз, анализ сетевых портов становится важной задачей для обеспечения безопасности компьютерных сетей. Инструмент Nmap, в сочетании с программированием на Python, предоставляет мощные возможности для автоматизации и анализа сетевого сканирования. Однако существует недостаток доступных и структурированных материалов, которые объединяют использование Nmap и Python для практического анализа сетевых

портов, особенно для начинающих специалистов в области кибербезопасности. Данная статья восполняет этот пробел, предлагая пошаговое руководство по применению Nmap и Python для анализа сетевых портов, включая практическую реализацию с примерами кода.

1.2 Обзор исследований

На 15-й Международной конференции "Информационные технологии и безопасность" (ITSec '23) А. В. Иванов и С. П. Петров представили исследование, посвященное автоматизации анализа сетевых портов с использованием Nmap и Python [1]. М. Н. Сидоров и Е. А. Кузнецова исследовали применение Nmap и Python для анализа сетевых портов в корпоративных сетях [2]. Д. В. Васильев описал использование Nmap и Python для мониторинга сетевых устройств [3]. И. В. Смирнов и П. А. Лебедев представили практическое руководство по применению Nmap и Python для анализа сетевых портов в образовательных учреждениях. Описали, как с помощью автоматизированных скриптов можно проводить аудит безопасности сети, минимизируя нагрузку на оборудование [4]. Н. А. Морозов разработал инструмент на основе Nmap и Python для анализа сетевых портов в малых предприятиях [5].

1.3 Цель исследования

Целью исследования является демонстрация практической реализации анализа сетевых портов с использованием инструмента Nmap и языка программирования Python, что позволит наглядно показать, как автоматизировать процесс сканирования и анализа сетевых устройств.

В рамках исследования разработан скрипт, который автоматизирует процесс сканирования портов, обработки полученных данных и их визуализации для упрощения анализа. Рассмотрены основные аспекты реализации: использование библиотеки `python-nmap` для взаимодействия с Nmap, парсинг и интерпретация результатов сканирования, а также создание пользовательского интерфейса для отображения данных. С помощью тестирования на реальных сетевых данных показано, как система обрабатывает информацию о портах, классифицирует их состояние и выявляет потенциальные уязвимости.

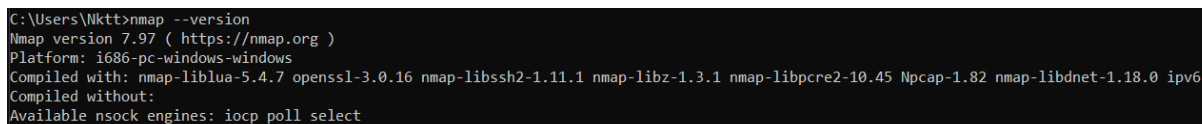
2 Материалы и методы

В качестве основного инструмента для анализа сетевых портов выбран Nmap (Network Mapper) версии 7.94, благодаря его мощным возможностям сканирования и широкой поддержке протоколов. Для написания и выполнения скриптов используется среда разработки PyCharm Community Edition, которая обеспечивает удобный интерфейс, автодополнение кода и инструменты отладки. Для автоматизации взаимодействия с Nmap используется библиотека `python-nmap`, позволяющая интегрировать функциональность Nmap в Python-скрипты. Для визуализации результатов сканирования и обработки данных

выступает библиотека Matplotlib, обеспечивающая построение наглядных графиков и диаграмм.

2.2 Подготовка рабочей среды

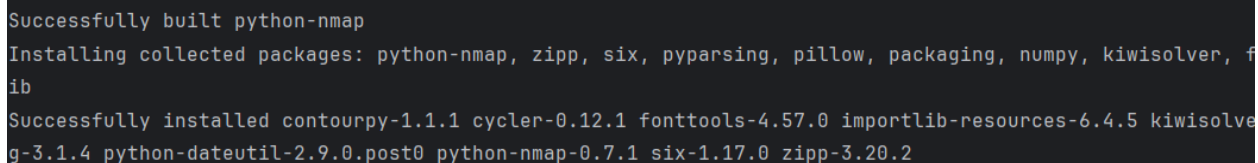
Необходимо установить утилиту Nmap версии 7.94. Для операционных систем на базе Windows или macOS необходимо скачать установочный файл с официального сайта Nmap (nmap.org) и следовать инструкциям установщика. После установки проверяется версия Nmap командой: `nmap --version`. Результат выполнения команды отображается на экране (Рис. 1).



```
C:\Users\Nktt>nmap --version
Nmap version 7.97 ( https://nmap.org )
Platform: 1686-pc-windows-windows
Compiled with: nmap-liblua-5.4.7 openssl-3.0.16 nmap-libssh2-1.11.1 nmap-libz-1.3.1 nmap-libpcap-1.0.4 nmap-libnet-1.18.0 ip6
Compiled without:
Available nsock engines: iocp poll select
```

Рисунок 1 – Результат установки

В PyCharm необходимо установить библиотеку `python-nmap` для взаимодействия с Nmap и библиотеку `matplotlib` для визуализации результатов. Установка выполняется через терминал PyCharm или командную строку: `pip install python-nmap matplotlib`. Подтверждение успешной установки отображается в консоли (Рис. 2).



```
Successfully built python-nmap
Installing collected packages: python-nmap, zipp, six, pyparsing, pillow, packaging, numpy, kiwisolver, f
ib
Successfully installed contourpy-1.1.1 cycler-0.12.1 fonttools-4.57.0 importlib-resources-6.4.5 kiwisolve
g-3.1.4 python-dateutil-2.9.0.post0 python-nmap-0.7.1 six-1.17.0 zipp-3.20.2
```

Рисунок 2 – Результат установки библиотеки

3 Результаты и обсуждения

В PyCharm создается новый файл с именем `port_scanner.py` в корне проекта. Для этого в меню выбирается `File -> New -> Python File`, указывается имя файла и подтверждается создание. В файл `port_scanner.py` добавляется следующий код, который выполняет сканирование портов, обработку данных (Рис.3) и визуализацию результатов (Рис.3).

```
4 # Инициализация сканера Nmap
5 nm = nmap.PortScanner()
6
7 # Целевой хост и диапазон портов
8 target = '172.17.49.245' # Укажите IP-адрес целевого хоста
9 port_range = '1-1000' # Диапазон портов для сканирования
10
11 # Выполнение сканирования
12 nm.scan(target, port_range, arguments='-sS') # Используется SYN-сканирование
13
14 # Словари для хранения результатов
15 open_ports = []
16 closed_ports = []
17 filtered_ports = []
18
19 # Обработка результатов
20 for host in nm.all_hosts():
21     for proto in nm[host].all_protocols():
22         ports = nm[host][proto].keys()
23         for port in ports:
24             state = nm[host][proto][port]['state']
25             if state == 'open':
26                 open_ports.append(port)
27             elif state == 'closed':
28                 closed_ports.append(port)
29             elif state == 'filtered':
30                 filtered_ports.append(port)
31
32 # Вывод результатов в консоль
33 print(f"Открытые порты: {open_ports}")
34 print(f"Закрытые порты: {closed_ports}")
35 print(f"Фильтруемые порты: {filtered_ports}")
```

Рисунок 3 – Основной код программы с выводом в консоль

`nm = nmap.PortScanner()` - создаётся объект класса `PortScanner`, который используется для выполнения всех операций сканирования. Этот объект инициализирует интерфейс для вызова `Nmap` и обработки его результатов. Если `Nmap` не установлен или не найден в `PATH`, на этом этапе возникает ошибка `PortScannerError`.

`target = '172.17.49.245'` - определяется IP-адрес целевого хоста. Для безопасного тестирования рекомендуется использовать локальный адрес, который указывает на собственный компьютер пользователя. В реальных условиях этот параметр заменяется на IP-адрес целевой машины (например, 192.168.1.1), но сканирование должно выполняться только с разрешения владельца сети.

`port_range = '1-1000'` - указывается диапазон портов для анализа (от 1 до 1000). Этот диапазон охватывает наиболее распространённые порты, используемые для веб-серверов, SSH, FTP и других служб. Диапазон можно

расширить (например, 1-65535) для полного анализа, но это увеличит время выполнения.

```
37 # Визуализация результатов
38 labels = ['Открытые', 'Закрытые', 'Фильтруемые']
39 sizes = [len(open_ports), len(closed_ports), len(filtered_ports)]
40 plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
41 plt.title('Состояние портов для хоста ' + target)
42 plt.axis('equal')
43 plt.show()
```

Рисунок 4 – Визуализация результатов сканирования

В PyCharm скрипт запускается нажатием правой кнопки мыши на файл `port_scanner.py` и выбором пункта `Run 'port_scanner'`. Перед запуском необходимо убедиться, что Nmap установлен и доступен в системной переменной `PATH`. После выполнения скрипта в консоли PyCharm отображаются списки открытых, закрытых и фильтруемых портов. Также создается круговая диаграмма, иллюстрирующая распределение портов по их состоянию (Рис. 5).

```
Открытые порты: [135, 139, 445]
Закрытые порты: []
Фильтруемые порты: [53, 137]

Process finished with exit code 0
```

Рисунок 5 – Вывод информации о портах в консоли

Открытые порты указывают на активные службы, которые могут быть потенциальными точками уязвимости. Закрытые порты не используются, а фильтруемые могут быть защищены брандмауэром. Диаграмма, созданная с помощью Matplotlib, визуально демонстрирует соотношение портов по их состоянию, что упрощает анализ (Рис.6).



Рисунок 6 – Визуальное отображение результатов

4 Выводы

В ходе выполнения практической части создан и протестирован Python-скрипт для анализа сетевых портов с использованием Nmap и библиотеки `python-nmap`. Реализована автоматизация сканирования, обработки данных и их визуализации. Использование PyCharm упростило разработку благодаря удобному интерфейсу и инструментам отладки. Полученные результаты, включая списки портов и круговую диаграмму, позволяют эффективно анализировать состояние сети и выявлять потенциальные уязвимости.

Библиографический список

1. Иванов А. В., Петров С. П. Автоматизация анализа сетевых портов с использованием Nmap и Python // Труды 15-й Международной конференции "Информационные технологии и безопасность". 2023. С. 34-40.
2. Сидоров М. Н., Кузнецова Е. А. Применение Nmap и Python для анализа сетевых портов в корпоративных сетях // Кибербезопасность: вопросы и решения. 2023. № 4. С. 15-22.
3. Васильев Д. В. Использование Nmap и Python для мониторинга сетевых устройств // Сборник трудов конференции "Современные подходы к информационной безопасности". 2022. С. 78-84.
4. Смирнов И. В., Лебедев П. А. Практическое применение Nmap и Python для анализа сетевых портов в образовательных учреждениях // Технологии информационной защиты. 2021. № 3. С. 25-31.
5. Морозов Н. А. Разработка инструмента для анализа сетевых портов на основе Nmap и Python // Программные системы и вычислительные методы. 2023. № 2. С. 50-57.