Homework Manual: Packet Analyzer

Due date: 7th February, 2025

Introduction

In this assignment you will write a network packet analyzer called `pktsniffer` that reads packets and produces a detailed summary of those packets. The `pktsniffer` program first reads packets from a specified file (pcap file). Then it extracts and displays the different headers of the captured packets.

Make sure to read the following instructions carefully before starting to write your code, more specific coding details are outlined later in the document.

Instructions

Before starting your work, carefully review all the following instructions to avoid missing any key requirements. Detailed coding instructions and specific requirements are explained later to guide your implementation .

1. **Wireshark**: This tool will be used to capture packets and save them into .pcap files.
    a. Go the link: https://www.wireshark.org
    b. Learn what Wireshark is.
    c. Download and install it.
    d. Use Wireshark to generate a `.pcap` file that contains captured network traffic.
2. **Programming Language:**
    a. You must use **Python** to implement your code (Useful link: https://www.python.org)

   b. Follow the PEP 8 coding guidelines for writing clean and professional Python code. (Here is a link: https://peps.python.org/pep-0008/)
   c. Use `git` for version control of your code
3. **Code Documentation and Comments:**
   a. Properly comment your code to make it clear and understandable
   b. Generate a PDF documentation for your code using a tool like Sphinx that automatically generates it for you. (see this link: https://www.sphinx-doc.org/en/master/)
4. **What to Submit:** All files must be submitted in a **single zip file**. Name the zip file <firstname>_<lastname>_hw1.zip (replace <firstname> and <lastname> with your name). The zip will have **atleast 7 files**:
   a. **Python code**: The main `pktsniffer.py` along with any addition Python files required to run your code.
   b. **Requirements File**: Submit a `requirements.txt` file listing all the Python dependencies required to run your code. (See this link: https://www.geeksforgeeks.org/how-to-create-requirements-txt-file-in-python/)
   c. **Readme**: A `README.md` file that clearly explains: 1- How to compile and run your code. 2- Examples of command-line usage
   d. **Captured Packets**: A `.pcap` file generated using Wireshark that contains the network traffic analyzed by your program.
   e. **Report (PDF)**: The report must be in pdf format. It should include:
      i. Screenshots comparing the first and last few packets from your program's output with the corresponding packets displayed in Wireshark
      ii. Screenshots demonstrating the functionality of each filtering command (filtering commands are explained later). For example pictures of your code showing that certain packets that were in the original file (shown in wireshark) were filtered in the output of your code when certain filtering flags were turned on.
   f. **Code Documentation**: It must be in PDF format. Use a tool like Sphinx to automatically generate documentation
   g. **GIT Log**: Use Git for version control and make commits at appropriate points during development. Submit a git log file showing the commit history. This file should be name `revisions.txt`

Input to the Code:

Capture packets using Wireshark and save them into a `.pcap` file. This file will serve as input to your program.

Output

The `pktsniffer` program should produce an output summarizing the captured packets. The following headers should be parsed and displayed:

- **Ethernet Header:** Packet size, Destination MAC address, Source MAC address, Ethertype.
- **IP Header:** Version, Header length, Type of service, Total length, Identification, Flags, Fragment offset, Time to live, Protocol, Header checksum, Source and Destination IP addresses.
- **Encapsulated Packets:** TCP, UDP, or ICMP headers.

Filtering

> Extend the `pktsniffer` program to support filtering based on the following criteria:

- `host`
- `port`
- `ip`
- `tcp`
- `udp`
- `icmp`
- `net`

> See this link for details: https://www.tcpdump.org/manpages/pcap-filter.7.html

> Your program should also support the `-c` flag to limit the number of packets analyzed.

Examples

```
pktsniffer -r file.pcap host 192.168.0.1
pktsniffer -r file.pcap -c 5
pktsniffer -r file.pcap port 80
```

```
pktsniffer -r file.pcap -net 192.168.1.0
```

FAQ

1. How can I check if my output is correct?

Use Wireshark to open the `.pcap` file and compare its output with the output of your program.

2. What is the purpose of the `net` flag?

It filters packets based on a network address. For example, `pktsniffer -r file.pcap -net 192.168.1.0` will display all packets where either the source or destination IP belongs to the `192.168.1.x` network.

Tips

- Use the **argparse** module in Python to parse command-line arguments.
- Keep your program modular. Separate functions should handle different layers of the packet headers.
- Some of the modules you create might be reusable in future projects.
- Use Git effectively for version control by making commits at key development milestones and providing clear commit messages.

Useful Links

- https://www.wireshark.org
- https://www.python.org
- https://peps.python.org/pep-0008/)
- https://www.sphinx-doc.org/en/master/
- https://www.geeksforgeeks.org/how-to-create-requirements-txt-file-in-python/

Homework Manual: Packet Analyzer