CSC 645 Project #3 Documentation

Spring 2024

Nhan Nguyen

923100929

https://github.com/nhannguyensf/ethernet-frame-capture-with-scapy

Table of Contents

1 Introduction

This project aimed to develop a tool using Python and Scapy for capturing and analyzing Ethernet frames. This tool is intended to help users understand network behaviors and packet data structures by providing insights into the real-time data flowing through their network interfaces.

2 The student's approach to the problem

2.1 Setup and Environment Preparation

- Scapy Installation

Virtual Machine on Windows: Set up a virtual environment using Oracle VirtualBox to install a Linux distribution (Ubuntu). This provides a full-featured Linux environment which often simplifies network tools usage.

- Learning Scapy Basics

Familiarize myself with basic functions of Scapy necessary for packets capture.

Useful resource: Packet Sniffing with Scapy on GeeksforGeeks

2.2 Script Development for Packet Capture

- Writing the Script
- Develop a Python script using Scapy that:
    - Initializes the capture of 15 Ethernet frames.
    - Extracts essential information such as source and destination MAC addresses, and if available, the IP version, source IP, and destination IP addresses.
    - Formats and prints the first 42 bytes of each frame for detailed analysis.

2.3 Frame Analysis

For each frame captured:

- MAC Addresses: Extract and print the source and destination MAC addresses.
- IP Layer Check: Determine if the frame contains an IP layer; if so, extract further details.
- Data Extraction: Extract the first 42 bytes in hex format and format the output for clarity.

2.4 Parallel Capture with Wireshark

- Simultaneously capture packets using Wireshark to validate the results obtained from the Scapy script.
- Compare and document the first 42 bytes of at least one packet from both tools, ensuring they match.

3 Development Environment

3.1 Version of Ubuntu Used

The project was developed and tested on **Ubuntu 18.04.6 LTS**, ensuring compatibility with widely used stable Linux distribution.

3.2 IDE Used

The development was carried out using Visual Studio Code (VS Code), which provided robust Python support and useful extensions for version control integration, and syntax highlighting.

3.3 Special libraries used or special resources

- **Scapy**: Used for packet crafting and capturing, this forms the backbone of the project's functionality.
- **Wireshark**: Used for validating the packet captures done by the Python code with Scapy, ensuring accuracy in packet interpretation.

4 How to run Project

4.1 Import and build the project from GitHub:

Follow these steps:

- git clone https://github.com/nhannguyensf/ethernet-frame-capture-with-scapy.git
- cd ethernet-frame-capture-with-scapy

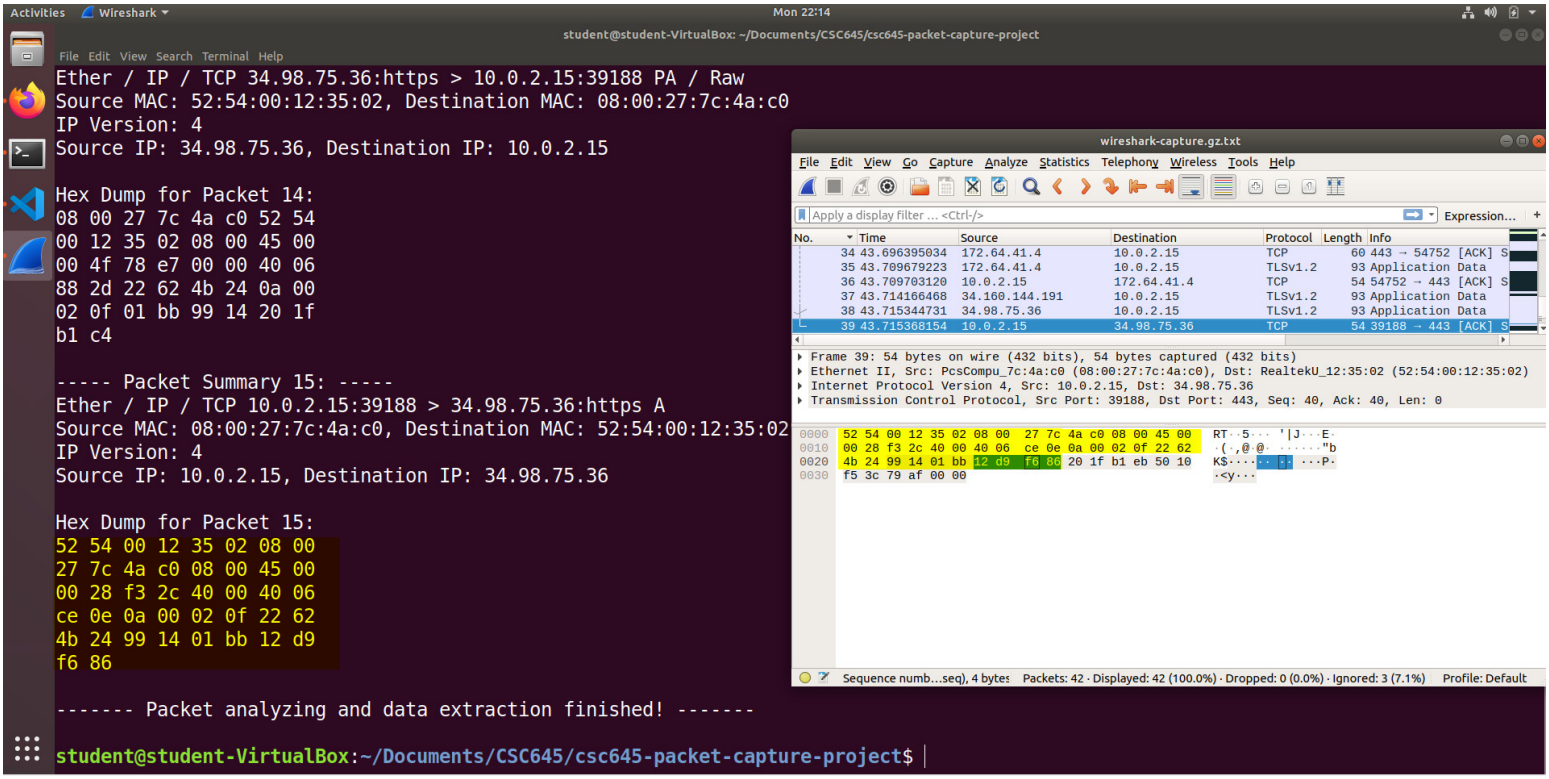
4.2 Command to run the code.

- sudo python3 capture_ethernet.py

5 Key observations from captured data

The purpose of this exercise was to validate the accuracy and consistency of packet captures between a script-based tool (using Scapy) and a GUI-based tool (Wireshark).

Both captures were performed simultaneously to minimize differences in traffic. The packets captured by both tools were consistent (highlighted):

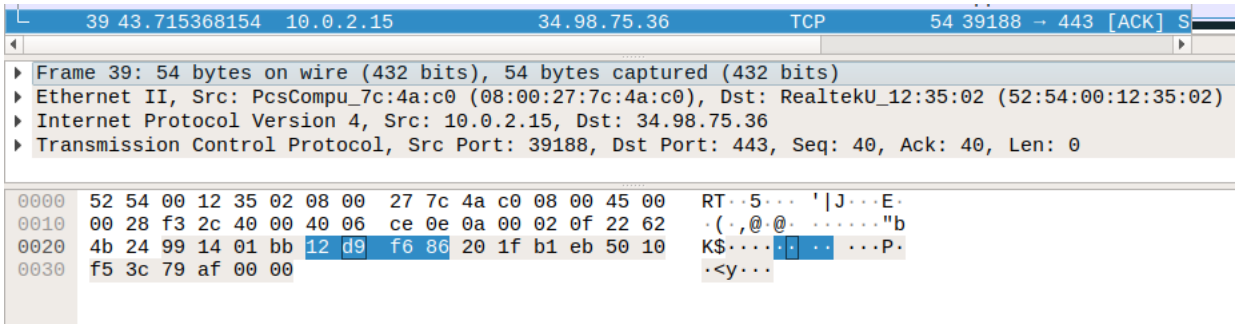


Let's take a look at packet 15, the first 42 bytes of that packet captured by Scapy were exactly the same as the one captured by Wireshark:

- From the hex dump for packet 15:

```
52 54 00 12 35 02 08 00
27 7c 4a c0 08 00 45 00
00 28 f3 2c 40 00 40 06
ce 0e 0a 00 02 0f 22 62
4b 24 99 14 01 bb 12 d9
f6 86
```

- From Wireshark (a closer look from the previous screenshot):



- Similarly, packet 14 from the Scapy code also match with packet number 38 in Wireshark.

6 Challenges faced and how they were overcome.

6.1 Insufficient Network Traffic for Capturing Desired Packet Quantity

Challenges: At first, after writing the Python code using Scapy to capture the packets, it kept running but didn't return the result. I found out that this problem was caused by a low network activity in my VM, where the traffic was insufficient to generate enough packets within that time.

Solution: I manually opened web pages to generate network traffic. I opened several tabs in my web browser to access various websites, thereby ensuring a steady stream of HTTP and HTTPS packets flowing through the network interface.

6.2 Understanding and Documenting Packet Data

Challenges: I need to have good understanding about the structure of Ethernet frames and IP packets.

Solution: I enhanced my understanding by consulting various educational resources, including textbooks and online courses on network protocols.

6.3 Discrepancies Between Scapy and Wireshark Captures

Challenges: Initially, there were discrepancies in the packets captured by Scapy compared to those captured by Wireshark. This issue was crucial to address since the project aimed to demonstrate the reliability of Scapy in capturing and analyzing network packets accurately.

Solution: I conducted multiple test runs to compare data and identified the root causes of discrepancies, which included differences in filters and timing issues. And I also manually look for the packets that match together.

7 Project Conclusion

This project demonstrated the capability to capture and analyze network traffic programmatically using Python and Scapy. It serves as a practical tool for network diagnostics and analysis, providing me foundational knowledge and skills applicable in computing and networking fields.