CSC/ECE 573
Fall 2023
Programming Assignment #2

Instructions

- You must do this assignment in groups as assigned.
- Submit your final codes and other documents (mentioned later in the description) via Moodle.
- The due date and time for this assignment: November 28, 2023, 11:45pm ET.

Packet Sniffer (60 Points: 50 Implementation and Experiments, 10 Report)

1. Objective:

This project aims at giving students an understanding of how raw socket interface and packet header parsing works. A raw socket allows an application to directly access lower-layer protocols, which means a raw socket receives un-extracted packets. There is no need to provide the port and IP address to a raw socket, unlike in the case of the stream (TCP) and the datagram (UDP) sockets. In this project, the students will get the experience of: 1) working with protocol headers at datalink, network, and transport layer, 2) designing data structures for the header of each protocol, and 3) mapping the header fields from the buffer in which packet has been received on to the data structure. After completing this project, students should have developed some confidence in understanding the network stack code of any open-source OS.

2. Description:

In this project, you are going to develop a packet sniffer tool. Your tool should be able to continuously read packets from the raw socket and parse their headers depending on the protocol. You can take help from tutorials on raw sockets available online. However, you are expected to develop your tool without using pre-existing codes on the internet (changing variable names does not amount to originality and it is very easy to spot). The minimum requirements for this project are:

1. Sniffing of all the packets that are received on the raw socket and parsing of headers for ethernet, IP, TCP, UDP, and DNS protocols. Extraction of data from packet is not required. You should only parse headers such that you can identify the protocol used on each layer.

2. Your tool should be able to categorize packets based on protocols (IP, TCP, UDP, and DNS) i.e., for the duration of execution of your tool, it should keep the count of all the packets received for these protocols.
    - **Points distribution**
        - **Successfully implementing this part will secure you 9 points for each protocol. So total points for getting correct packet count for four protocols is 36 out of 50.**

3. When your program is executed, it should run for 30 seconds and then exist gracefully. There should be no error or exception at the time of compilation, execution, during runtime, and at the time of exit.
    - **Points distribution**
        - **Correct implementation of 30 seconds runtime and graceful exit [4 points].**

4. When your program exits after 30 seconds, it should generate a .csv file named as sniffer_<unityid>.csv, e.g., sniffer_mshahza.csv. This file should have the count of packets of each of the four protocols as mentioned in point (2). The format for .csv file *must* be as follows:

    protocol,count
    ip,192
    tcp,61
    udp,47
    dns,22

    For example: if you receive a DNS packet, you will increment count for:
    - DNS as this is the protocol to identify
    - UDP because DNS uses UDP as transport
    - and IP because at network layer IP is used by DNS.

    Note: Every alphabet in .csv file is in lower case. There are no spaces before and after the comma. The counts represent the number of packets received for that protocol. The numbers above are shown as examples. Your counts will vary.

    - **Points distribution**
        - **Each protocol count in the file gives you 2.5 points. So, correct file generation in the specified format with counts for the four protocols will secure you [10 points].**

- **Other reasons for deductions**
    1. **Failure to compile the submitted code [-20 points].**
    2. **Error at the time of execution, during runtime, or exit [-20 points].**
    3. **Exception at the time of execution, during runtime, or exit [-20 points].**
    4. **No file generated after program exits, or is generated but is empty [- 20 points].**
    5. **Not implementing code for any protocol [-12.5 points/protocol].**

    For points 1 through 4, the team will be give a chance to resubmit their code within 24 hours of the notice from the TAs. If the team does not resubmit, or the resubmitted code has errors again, the team will receive 50 point deduction.

3. Experiments:
Once your development is complete, please carryout following experiments with your tool.

Exp 1: In this experiment, you first play any YouTube video at highest resolution possible in a browser. Once video starts playing, run your tool. Once your tool exits itself after 30 seconds, you will get an output .csv file.

Exp 2: In this experiment, first run your tool. Then open a browser, go to YouTube website and quickly click on any video. Let this video play until the tool exits. Once your tool exits itself after 30 seconds, you will get an output .csv file.

Exp 3: In this experiment, run your tool. Then open a browser and randomly search stuff on google and open different websites until the tool exits. Once your tool exits itself after 30 seconds, you will get an output .csv file.

You are going to compare the results of these three experiments and discuss your observations in a report. You will also have to plot the results of these three experiments on a single figure. The format of the report and plot are provided in the template.

4. Submission Instructions:
We will test your code for successful compilation, execution, termination, and .csv file generation, which will contain the counts of packets for the protocols as mentioned earlier. Your results will be compared with our tool for correctness.

Your program will be tested on Ubuntu 22.04 LTS operating system. If you are using Windows or MAC machines, then install VMware or virtual box on your machine and setup a virtual machine (VM) for Ubuntu 22.04 LTS.

If you are using VCL then make sure you do the following:

1. Take backup before turning off VM or before its reservation time expires. Your data *will* be lost if you do not take the backup.
2. If there are packages or dependencies required for this project, then you will have to install those on the VM every time you reserve it. Therefore, maintain a list of packages and dependencies that you have to install every time so that it may save you time.

For submission, students must make a folder named **HW2_<unityid>**. and place the following five files in it:

1. Code file named as sniffer_<unityid>.py, OR sniffer_<unityid>.c, OR sniffer_<unityid>.cc
2. Three CSV files named as sniffer_E1_<unityid>.csv, sniffer_E2_<unityid>.csv, and sniffer_E3_<unityid>.csv, corresponding to the three experiments mentioned above.
3. Readme file as readme_<unityid>.txt
    - Readme file should contain clear instructions about the following:
        - How to compile the code? Provide command.
        - How to execute the compiled code? Provide command.
        - Which packages and dependencies are required by your code for the successful compilation and execution of the code? Provide a list of all those packages and dependencies. Failure to provide this may lead to unsuccessful compilation and execution of your code due to which your points will be deducted as per the criteria mentioned in Section 2 (Description).

When you have placed the five files in the folder* HW2_<unityid>*, go to the* Final Submission *instructions on the next page.

Hints:

- *Take raw sockets, libpcap, or rawsocketpy as a starting point.*
- *Implement in python, C, or C++. Choose from only these three programming languages.*
- *The header of the lower layer has a field that identifies the protocol of the higher layer.*
- *Implementing this tool is easier than you think :)*

Report (10 points)

The template for the report is provided in the attached file. Please follow the instructions below while writing the report:

1. The report must be to-the-point. There is no need of long details.
2. Only follow the format of the template and fill in the required data.
3. There are 10 points for each project in the report.
4. Report must be submitted in .pdf format. Name the report as HW2_report_<unityid>.pdf.
5. Once the report is complete, see below for final submission instructions.

Final Submission

When you have created the folder HW2_<unityid> and added the respective files in it, zip it along with the report into a single .zip file. Name that .zip file as <unityid>.zip and submit it on Moodle. Your submitted .zip file contains following:

- ✓ HW2_<unityid> folder
- ✓ HW2_report_<unityid>.pdf

Good Luck!