WPI

CS 3516 - Computer Networks

Project 2 (100 points)
Assigned: Friday, March 22, 2024
Checkpoint: Tuesday, April 2, 2024 at 11:59pm
Due: Tuesday, April 12, 2024 at 11:59pm

Programming Project 1: Wire View

Programs such as **tcpdump** (`https://www.tcpdump.org/`) and **Wireshark** (`https://www.wireshark.org/`) allow the interception and analysis of packets being transmitted or received over a LAN. One prominent use of this information is in troubleshooting network configuration and reachability. In this project, we will provide you with data captured using these tools. Your task is to write the analysis routines similar to those provided by **tcpdump** and **Wireshark**. Your program must be written in C/C++ and must work on a standard Ubuntu 20.04 Linux virtual machine (which is what the graders will be using).

Please note: WPI's AUP prohibits the use of intercepting other users' traffic. In this project, we will be using our software to read from files, not the network connection to conform to this policy. Please be careful with the tools you use and create to avoid accidentally capturing live network traffic.

Project Specification

Your program, `wireview`, should take a file containing **tcpdump** data as its input. It will then output the statistics detailed later in this document. Since this data is not in human-readable format, you will have to use the Packet Capture Library, `libpcap.a`, and the functions in its header file, `pcap.h` (found in /usr/include on many Linux machines) to read the data. On the provided VMs, run `sudo apt-get install libpcap-dev` to install the libpcap API. When compiling your program, include the `pcap` library by using `-lpcap` as the first argument to your GNU compiler. For example, `gcc -lpcap -o wireview wireview.c` will compile a C program with the `pcap` library support. For C++, simply change the compiler from `gcc` to `g++`.

Please note: The gcc/g++ compilers can be finicky with the order of parameters. If you are getting compilation errors with `gcc -lpcap -o wireview wireview.c`, try using `gcc -o wireview wireview.c -lpcap`.

The other steps you should follow are:

- Open an input file using function `pcap_open_offline()`.
- Check that the data you are provided has been captured from Ethernet using function `pcap_datalink()`.
- Read packets from the file using function `pcap_loop()`. Note that this function needs to be called only once. It takes 4 arguments. Of these, the second and the third arguments are of most interest to you. The second argument lets you specify how many packets to read from the file (and a negative number means "loop forever or until an error occurs"). The third argument, `pcap_handler callback`, is where most of the action happens. Here, `callback` is the function you write to process data from each packet.

  You can pass the callback function to the `pcap_loop()` function simply by giving its name as the appropriate argument to `pcap_loop()`. The callback function must be a void function that takes three arguments, of the types `u_char *, const struct pcap_pkthdr *, const u_char *`. The callback is called by `pcap_loop()` once for each packet. The second argument to the callback is the special libpcap header, which can be used to extract the entire packet length and the packet arrival time (see the `pcap_pkthdr` structure in `/usr/include/pcap.h`). The third argument contains the contents of a single packet (from the Ethernet packet header onward).
- Close the file using function `pcap_close()`.

Packet format

Each packet in the file(s) provided to you is in **tcpdump** format. It contains a tcpdump-specific header, an Ethernet header, followed by network layer headers and their payloads. These packet captures will contain IP and ARP headers at the network layer, and UDP at the transport layer. You will need to understand each of the header formats to accomplish this task. You are encouraged to reuse the structures from the relevant Ethernet, IP, and UDP header files in the `/usr/include/net` and `/usr/include/netinet/` directories on most Linux machines. These files contain structures used by the Linux operating system for actual packets. They can be used to greatly simplify the process of parsing the packets. However, you are still free to implement the structures on your own, if you wish. Note that ARP/Ethernet includes are present in both of these directories. Look at the provided structure and compare them with what you understand about these structures. In the past, students have tried to use the first structure they see that seems close enough without realizing it is for a different protocol (in particular, with ARP).

Please note, only IPv4 header processing is required. If an IPv6 packet is received, you need only process the Ethernet portion of the packet before aborting processing on that packet.

Program output

The `callback` function should gather statistics from each packet to enable your program to print the following information on standard out.

- Print the start date and time of the packet capture.
- Print the duration of the packet capture in seconds with microsecond resolution.
- Print the total number of packets.
- Create two lists, one for unique senders and one for unique recipients, along with the total number of packets associated with each. This should be done at two layers: Ethernet and IP. For Ethernet, represent the addresses in `hex-colon` notation. For IP addresses, use the standard dotted decimal notation.
- Create a list of machines participating in ARP, including their associated MAC addresses and, where possible, the associated IP addresses.
- For UDP, create two lists for the unique ports seen: one for the source ports and one for the destination ports.
- Report the average, minimum, and maximum packet sizes. The packet size refers to everything beyond the **tcpdump** header.

Test files

The following are a few packet captures you can use for testing purposes.

- `https://cerebro.cs.wpi.edu/cs3516/project2-dns.pcap` – Packet Capture 1: Simple DNS request and reply
- `https://cerebro.cs.wpi.edu/cs3516/project2-http.pcap` – Packet Capture 2: HTTP traffic
- `https://cerebro.cs.wpi.edu/cs3516/project2-arp-storm.pcap` – Packet Capture 3: ARP storm from a cable Internet network
- `https://cerebro.cs.wpi.edu/cs3516/project2-other-network.pcap` – Packet Capture 4: Traffic from another network (can you figure out which it is?)

Your program should work on packet captures that have additional protocols your program does not understand (it should be able to parse past them). Additional test files may be used during the demo.

Miscellaneous

- You can install Wireshark onto your virtual machine and experiment with it as a sanity check for your program's functionality.
- Resource: `https://www.tcpdump.org/pcap.htm`
- Use the `ntohs` and `ntohl` functions as appropriate to read values that span multiple bytes. This ensures that the bytes are in proper byte order (Endianness) for use at this host. Students are often confused by Endianness, so please study the topic carefully.

Road Map

When completing the assignment, you may use the following road-map as a guide:

1. Start with examining the TCP dump header for basic statistics.
2. Use an Ethernet header to print out the associated addresses
3. Determine whether ARP or IP is in use. For ARP, print out the protocol fields.
4. Print out the IP sources and destinations.
5. Determine whether a packet carries UDP and, if so, print out the UDP ports.
6. Rather than print per-packet statistics, store things in a data structure to print final statistics. (Hint: look at the C++ STL map data structure).
7. Print out the final statistics using your populated data structures.

Checkpoint Contributions

Students must submit work that demonstrates substantial progress towards completing the project on the checkpoint date. Substantial progress is judged at the discretion of the grader to allow students flexibility in prioritizing their efforts. However, as an example, the checkpoint contributions would be met if the required pcap functionality is present. For example, if students can print out a "hello world" for each packet in the capture, it would likely show a proper base implementation. However, like all checkpoints, equivalent work in other areas would suffice. **Projects that fail to submit a checkpoint demonstrating significant progress will incur a 10% penalty during final project grading.**

Deliverables and Grading

When submitting your project, please include the following:

- All of the files containing the code for all parts of the assignment.
- One file called `Makefile` that can be used by the `make` command for building the executables. It should support the "`make clean`" command and the "`make all`" command.
- A document called `README.txt` explaining your project and anything that you feel the teaching staff should know when grading the project. Only plaintext write-ups are accepted.

Please compress all the files together as a single .zip archive for submission. As with all projects, please **only use standard zip files** for compression; **.rar, .7z, and other custom file formats will not be accepted**.

The project programming is only a portion of the project. Students should use the following checklist in turning in their projects to avoid forgetting any deliverables:

1. Sign up for a project partner or have one assigned (URL: `https://ia.wpi.edu/cs3516/request_teammate.php`),
2. Submit the project code and documentation via InstructAssist (URL: `https://ia.wpi.edu/cs3516/files.php`),
3. Complete your Partner Evaluation (URL: `https://ia.wpi.edu/cs3516/evals.php`)

A grading rubric will be provided separately to give you a guide for how the project will be graded. No points can be earned for a task that has a prerequisite unless that prerequisite is working well enough to support the dependent task. Students will receive a scanned markup of this rubric as part of their project grading feedback. Students in teams are expected to contribute equally; unequal contributions may yield different grades for the team members. Students who work in teams but do not submit a Partner Evaluation by the project deadline will incur up to a 5% penalty on the project grade.