Programming Assignment #6

Packet Sniffing

---

1. Introduction

Packet sniffing is the process of monitoring the raw traffic on a device connected to a network. This is useful for analyzing network traffic which can lead to the discovery of suspicious activity.

There are many great programs for this, including *tcpdump*, *snort*, and *ethereal*. Writing your own program allows you to better understand the structure and processing of traffic on a network.

Code developed for this programming assignment will be used later as building blocks for an Intrusion Detection System.

2. Details

Use *libpcap* if you are programming in a Linux/Unix type of environment, or *WinPcap* if you are programming in a Microsoft Windows environment. They can natively be used with C/C++ and have bindings to Java as well.

For Linux/Unix users, the libpcap library can be downloaded from www.tcpdump.org.

For Windows useres, the winpcap library can be downloaded form winpcap.polito.it.

Whether you use libpcap, or winpcap, read the tutorial from each website. Also, read and understand the examples that come with the source code.

For ease of programming, only worry about IPV4. You get 50% extra credit if you also handle IPV6.

3. What You Have to Do

This assignment is broken down into several parts to help you understand the overall use of packet sniffing. When using pcap, you will find that you need to choose which network device to use.

All programs in this assignment will need to take a command line argument that specifies which network device to use, starting from 0.

3.1 Sniffing All Traffic (15%)

Write a program, named **sniffall**, that sniffs all traffic on a single device (e.g. your ethernet card) and dumps all of the information to the console screen. Make sure to include the source IP address and port, as well as the destination IP address and port in the output as shown in the example below.

Example of output:

```
$ ./sniffall 0
192.168.0.2:2345 192.168.0.1:22
192.168.0.2:5445 192.168.0.1:80
192.168.0.2:5434 192.168.0.1:25
```

```
192.168.0.2:4534 192.168.0.1:21
192.168.0.2:655 192.168.0.1:7070
```

3.2 Sniffing TCP Traffic (15%)

Write a program, named **snifftcp**, that utilizes pcap's filter rules in order to sniff *only* TCP traffic on a single device (e.g. your ethernet card) and dumps all of the information to the console screen. Make sure to include the source IP address and port, as well as the destination IP address and port in the output, use the same format as above.

3.3 Sniffing UDP Traffic (10%)

Write a program, named **sniffudp**, that utilizes pcap's filter rules in order to sniff *only* UDP traffic on a single device (e.g. your ethernet card) and dumps all of the information to the console screen. Make sure to include the source IP address and port, as well as the destination IP address and port in the output, use the same format as above.

3.4 Writing Traffic to File (10%)

Often you will want to be able to save suspicious traffic to file. PCAP has the ability to dump traffic to file that is formatted to be compatible with both tcpdump and snort.

Write a program, named **dumptraffic**, that sniffs all traffic on a single device (e.g. your ethernet card) and dumps all of the information to a file named dump.txt. PCAP will automatically dump all relevent info in a proper format if you use the tools correctly, which means you do not have to worry about formatting.

3.5 Reading Traffic From File (10%)

Often you will want to be read traffic dumps that you saved from your program or other programs. This is also easily done using PCAP. The routine is very similar to the one that reads traffic from a device, except that you will open a local file instead.

Write a program, named **readtraffic**, that reads all traffic from a file named dump.txt.

3.6 Analyzing Traffic (15%)

In order to create an IDS system later on, we are going to need to be able to analyze the incoming packets for certain content. Write a program, named **analyzetraffic**, that takes an extra argument, the string to look for in the packets. You will search for this string in all incoming traffic (TCP and UDP). If you find an exact match in a packet, then print out a line in the same format as before, only prepend it with "Matched: ".

Example of output:

```
$ ./analyze 0 "qwerty"
Matched: 192.168.0.2:2345 192.168.0.1:22
Matched: 192.168.0.2:5445 192.168.0.1:80
Matched: 192.168.0.2:5434 192.168.0.1:25
```

4. Testing Your Application (15%)

To test your code you can use a program called *hping*, a powerful tool for creating arbitrary packets. You can download hping from www.hping.org.

The following questions will help you figure out how to use hping to test your program, these questions must be filled out and turned in with your assignment.

Q1: What would the command line arguments look like for creating a TCP packet that has a source IP of 192.168.0.2, a source port of 3466, a destination IP of 192.168.0.1, and a destination port of 21?

Student name:
Type your content in this table row.

Q2: What would the command line arguments look like for creating a UDP packet that has a source IP of 192.168.0.2, a source port of 35686, a destination IP of 192.168.0.1, and a destination port of 7070?

Student name:
Type your content in this table row.

Q3: What would the command line arguments look like for creating a TCP packet that has a source IP of 192.168.0.2, a source port of 9455, a destination IP of 192.168.0.1, and a destination port of 80, and had a payload that contained the string "qwerty"?

Student name:
Type your content in this table row.

5. Required Document (5%)

In addition to your source code and your output you are required to submit a memo to your instructor that is flawlessly written. No spelling errors. No grammatical errors, etc. If the document is deemed unprofessional, e.g. a significant number of grammatical or spelling errors, then it will be assigned a grade of zero.

If you're not sure about the format for a memo, then just search for "memo format" on Google.

Your memo should state clearly the status of the assignment. Is it done or not? Does it meet all of the requirements? If anything is missing then state clearly what is missing from your work.

Failure to give a status for the assignment will result in a grade of zero for the entire assignment. Providing a status that's false or significantly misleading will also result in a zero for the entire assignment.

Make sure your document is well-written, succinct, and easy to read. If you encountered problems with the assignment, then provide a detailed description of those problems and the solution(s) you found.

6. Assignment Submission (5%)

You are required to submit your work online, using the Blackboard. Late work will be accepted, however it is subject to late penalties as described in the syllabus.

Here are the requirements for your submission:

- Submit your electronic copy using the Blackboard (attach the assignment as a compressed archive file (.zip, .tgz, .tbz2, .rar)
- The name of the compressed archive should be: *fistName-lastName*-PA-*assignmentNumber*.*zip* (e.g. Jane-Doe-PA-6.zip)
- Include (i) your memo, (ii) a completed handout with your results and your name on it, (iii) the source code, (iv) a README file that explains how to build and on how to run each program, (v) compiled executable(s) and (vi) a shell script or batch file that will compile your programs when executed; a Makefile (see GNU Make for details) would be great, however any form of script or building tool will do
- Include your e-mail address in the Comment field when submitting the assignment through the Digital Drop Box
- If for any reason you are submitting the assignment more than once, indicate this in the Comment field by including the word COMPLEMENT
