National Research Tomsk State University

Laboratory 3 – Passive Testing of Communication Protocols

Passive Testing Techniques for Communication Protocols

February 26, 2016

Welcome to your last laboratory! The goal of this laboratory is to introduce you to the concepts of passive testing using network traces. In this laboratory, we will be only using the principles learned and implementations of your own. Your implementations will be using the library which is the holy grail of packet inspection programming – libpcap. You will be making your network monitors. All the methods can be applied to larger (and more complex) network monitors, but given our restrictions in time we will not.

Most of the times, different Operating Systems (OS) or tools within them can provide similar results, for instance, WinPCAP can be used in Windows architectures. As stated before in the course introductory lecture, you are free to use the OS or tool of your choice. However, instructions, hints, commands, libraries, and etc., will be provided only for the a standard Linux environment. Furthermore, you should in any case provide a Makefile to compile your tools in the submission. As usual, I will be able to provide more assistance in the stated lab environment. In short, you're on your own if you decide to choose your own OS or tools, sorry :)

Submissions: Submissions should be done in a single compressed file (zip, or gz, including flavors of .tar.gz, and .tgz) of a directory containing all the necessary files. Regardless of how jealous you are about sharing your code, always include the code of your assignations. Include all your documents, executable files, and a Makefile to compile them. If you feel the need of putting a README.txt file, to clarify what are the contents of your submission feel free to do so. All submissions should be done via The Moodle Cloud Platform, https://pttcc.moodlecloud.com/ Make sure you are able to login and that you assign the course to yourself.

1 Passive Testing – Network Monitoring

I decided that on your last laboratory you will relax a bit (and perhaps catch up with the previous labs). This is a laboratory that covers less material, than the previous.

Nonetheless, the important principles of packet analysis programming can be seen in the code provided. Also, you'll get to play with network packets yourself :)

For this purpose you are given a compressed archive placed on the MoodleCloud and the local FTP, Lab3.tgz. The contents of the file are as follows:

- VSNP: The VSNP implementation that can inject errors
- PTNM: The passive testing network monitoring code

Within the VSNP forlder, the contents of the files are as described in the Laboratory 1. You have already the executable files for both the VSNPServer and VSNPClient from C source files. In case your architecture is not compatible with these executables, re-compile them (make server.c; make client.c).

The VSNP server has a hidden parameter, -e that takes a binary encoded error code. The error codes are as following(represented from the least significant bit at the right to the most significant bit at the left):

- bit 1: Do not check the odd/even, even/odd reply property, i.e., generate a random number from the ID and reply this number, independently if the ID is even or odd.
- bit 2: Insert bugs in the ID number, reading not the correct ID
- bit 3: Insert bugs not replying randomly to client requests.

If the previous was not clear enough, you can run the VSNPServer (sudo ./VSNPServer −s 127.0.0.1 −e N
to inject errors into your code, here is a brief summary of the different NUM possibilities:

- 1 = bit 1 on (insert the errors as described above in bit 1)
- 2 = bit 2 on (insert the errors as described above in bit 2)
- 3 = bit 1 + bit 2 (insert both errors as described above in bit 1 and bit 2)
- 4 = bit 3 on (insert the errors as described above in bit 3)
- 5 = bit 1 + bit 3
- 6 = bit 2 + bit 3
- 7 = bit 1 + bit 2 + bit 3

Try the VSNP server injecting different bugs. For example, injecting ID bugs and non checking of the odd/even, even/odd property can be done like this: sudo ./VSNPServer −s 127.0.0.1 −e 3.

Moving on, the contents of the PTNM are the following:

- ethernet.h: Header file which contains the Ethernet packet data structure (take a look at this file, most probably you won't need to use this, but, it's a good way to get familiar with the data structures involved in the process)

- helpers.c: Implementation of some functions that provide some functionality. Do not worry about this one
- helpers.h: The header file of the previous. Do not worry about this also.
- ip4.h: Header file containing the IPv4 header data structure
- linked_list.c: My linked list implementation, most probably you won't use this
- linked_list.h: The header file for the linked lits
- linked_list_node.h: The linked list node data structure
- Makefile: used to compile the PTNM program
- PTNM: The program that perform the passive testing or network monitoring
- PTNM.c: The source code for PTNM in C
- PTNM.cpp: The source code for PTNM in C++ (well, a bit lazy done this time, but, of course C++ (;)
- PTNM.h: The header file for PTNM, this file is particularly important since it contains the data packet data structure, the data structure for a network packet representation
- sip.h: Header file containing the data structure for SIP packets
- tcp.h: Header file for the TCP protocol header
- udp.h Header file for the UDP protocol header

As usual, to compile your C file, you execute make c; to compile your c++ file , you execute make cpp.

To execute the PTNM you do not need admin privileges since the compiling process does it for you. When you execute the PTNM, you pass as the first argument the interface that you want to capture packets from, and as the second argument a filter enclosed with quotes, for instance "port 53".

Try PTNM with the interface in which you have an IP address that can route to other computers, if you are using the VM, the interface is enp0s3. The filter can be set to "", for example: PTNM enp0s3 "" When you generate some traffic (go to a web page or something) from your computer you might see the first checking property.

Today, you'll be working on your PTNM.c file (or PTNM.cpp as convenient for you). First, make sure the arguments are properly passed with flags -i for the interface and -f for the filter. This can be done in your main method. This is your first task (I'm feeling very kind today)

Second, take a look at how the libpcap library functions work, the most important part is the pcap_loop function, and the specified function for a callback. In our case, we specified that each time a new captured packet comes, the function process_packet should be called.

Take a look at the process_packet function. It in fact places the raw data bytes into the packet data structure and sends the packet for property checking. If you need to add another. In the same function, take a look at how the SIP packets are processed. If the source of destination port is 5060, then we consider this is a SIP packet. If it was the VSNP protocol we would consider it the port 1010.

Take a look at process_sip and related functions, in fact, this are the functions that map from the raw data bytes to the data structures as discussed in class.

Finally, take a look at the check_properties function. In here you can define functions that inspect the packets.

The second task of this lab is to check the VNSP server replies. Check that the packet replies respect the even/odd odd/even constraint. For this you will make your VSNP data structures, your mapping functions, and the checking of your property. Have fun!

P.S. Please note that, if the VSNP server replies with different IDs for the connection you would need to store the request on a queue and check against the replies. For this lab, you won't make it, but, consider how many properties you can check with packet correlation :)

2 Adiós

As you can see your lab today was sweet and short. I hope you learned a thing or two from this class, and that you enjoyed it. It was a very pleasurable experience for myself. I hope to see you all very soon. In the mean time, if you feel like having any project regarding the topics we treated in class (or others), please, do not hesitate to contact me. So, farewell, my dear students :)