Packet Sniffing and Spoofing Lab

1 Overview

Packet sniffing and spoofing are the two important concepts in network security; they are two major threats in network communication. Being able to understand these two threats is essential for understanding security measures in networking. There are many packet sniffing and spoofing tools, such as `Wireshark`, `Tcpdump`, `Netwox`, etc. Some of these tools are widely used by security experts, as well as by attackers. Being able to use these tools is important for students, but what is more important for students in a network security course is to understand how these tools work, i.e., how packet sniffing and spoofing are implemented in software.

The objective of this lab is for students to master the technologies underlying most of the sniffing and spoofing tools. Students will play with some simple sniffer and spoofing programs, read their source code, modify them, and eventually gain an in-depth understanding on the technical aspects of these programs. At the end of this lab, students should be able to write their own sniffing and spoofing programs.

2 Lab Tasks

2.1 Task 1: Writing Packet Sniffing Program

Sniffer programs can be easily written using the `pcap` library. With `pcap`, the task of sniffers becomes invoking a simple sequence of procedures in the `pcap` library. At the end of the sequence, packets will be put in buffer for further processing as soon as they are captured. All the details of packet capturing are handled by the `pcap` library. Tim Carstens has written a tutorial on how to use `pcap` library to write a sniffer program. The tutorial is available at `http://www.tcpdump.org/pcap.htm`. In this task, you need to read the tutorial, play with the program `sniffex` included in the tutorial, read the source code `sniffex.c`, and solve the following problems:

Problem 1: Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial.

Problem 2: Why do you need the root privilege to run `sniffex`? Where does the program fail if executed without the root privilege?

Problem 3: Please turn on and turn off the promiscuous mode in the sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you demonstrate this.

Problem 4: Please write filter expressions to capture each of the followings. In your lab reports, you need to include screendumps to show the results of applying each of these filters.

- Capture the ICMP packets between two specific hosts.
- Capture the TCP packets that have a destination port range from to port 10 - 100.

Problem 5: Please show how you can use `sniffex` to capture the password when somebody is using `telnet` on the network that you are monitoring. You may need to modify the `sniffex.c` a little bit if needed. You also need to start the `telnetd` server on your VM. If you are using our pre-built VM, the `telnetd` server is already installed; just type the following command to start it.

```
% sudo service openbsd-inetd start
```

2.2 Task 2: Spoofing

When a normal user sends out a packet, operating systems usually do not allow the user to set all the fields in the protocol headers (such as TCP, UDP, and IP headers). OSes will set most of the fields, while only allowing users to set a few fields, such as the destination IP address, and the destination port number, etc. However, if the user has the root privilege, he/she can set any arbitary field in the packet headers. This is essentially packet spoofing, and it can be done through *raw sockets*.

Raw sockets give programmers the absolute control over the packet construction, allowing programmers to construct any arbitrary packet, including setting the header fields and the payload. Using raw sockets is quite straightforward; it involves four steps: (1) create a raw socket, (2) set socket option, (3) construct the packet, and (4) send out the packet through the raw socket. There are many online tutorials that can teach you how to use raw sockets in C programming. In this task, we will not focus on any specific tutorial. Your task is to read some of these tutorials, then either write your own packet spoofing program or download a program from other places. After playing with these programs, please solve the following problems:

Problem 6: Please use your own words to describe the sequence of the library calls that are essential for packet spoofing. This is meant to be a summary.

Problem 7: Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?

Problem 8: Please combine your sniffing and the spoofing programs to implement a sniff-and-then-spoof program. This program monitors its local network; whenever it sees an ICMP echo request packet, it spoofs an ICMP echo reply packet. Therefore, even if the victim machine pings a non-existing machine, it will always see that the machine is alive. Please include screendump in your report to show that your program works. Please also attach the code in your report.

3 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.