CMPSC443 Lab - Packet Sniffing and Spoofing Lab

1 Overview

Packet sniffing and spoofing are 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 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 of 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

Task 1: Writing Packet Sniffing Program

a. Understanding the python implementation of a network packet sniffer. Use the link: http://www.binarytides.com/python-packet-sniffer-code-linux/, and try the three examples provided. You do not submit any script for this part but you need to provide screenshots in the lab report.
b. Writing filters. Please implement the following: 1) Capture the ICMP packets between two specific hosts. 2) Capture the TCP packets that have a destination port range from to port 10 - 100. Your filter expression will be put into an external file 'filter.txt', which should only contain a line of your filter statement. Your program will read the filter expression from 'filter.txt' and return the results properly. You can define your own filter syntax. For instance, a string "ICMP host1_IP host2_IP" could be used for 1), and a string "TCP 10 100" can be the second filter. In the example, the first word specifies the protocol, and the following are parameters for the filter.
c. Run command "`telnet games.libreplanet.org`" from your terminal to register a user with username & pswd chosen by you. Now run the script from task 1.b, login the game server and show that the password can be captured by the sniffer in a screenshot.

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, the destination port number, etc. However, if users have the root privilege, they can set any arbitrary field in the packet headers (in theory). This is called 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. Please read the tutorial here: http://www.binarytides.com/raw-socket-programming-in-python-linux/, and learn how to write a packet spoofing program.

a. Write a spoofing program. You can modify the tutorial example to get yours. You need to provide evidences (e.g., Wireshark packet trace) to show us that your program successfully sends out spoofed IP packets.
b. Spoof an ICMP Echo request. Spoof an ICMP echo request packet on behalf of another machine (i.e., using another machine's IP address in the same LAN as its source IP address). This packet should be sent to the host machine from Kali. You should turn on your Wireshark on the host, so if your spoofing is successful, you can see the echo reply to the spoofed address.
c. Spoof an Ethernet Frame. Set 01:02:03:04:05:06 as the source address.

Questions:

Q1: Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?

Q2: Using the raw socket programming, do you have to calculate the checksum for the IP header?

Submission

- A word/pdf file that provides screenshots for each task and also includes answers to the questions.
- Python scripts for all subtasks. Script naming convention is "script_t1a1", in which t1 means task 1, a1 means the first script for subtask a.
- Do not compress your submission.