Sniffing and Spoofing using PCAP Library

Table of Contents:

Lab Environment Setup

Please download the Labsetup(FOR MAC).zip file from the PESU ACADEMY which consists of both the Labsetup and the codes.
Follow the instructions in the **lab setup document** to set up the lab environment.

In this lab, we will use three machines that are connected to the same LAN. We can either use three VMs or three containers. Figure 1 depicts the lab environment setup using containers. We will do all the attacks on the attacker container, while using the other containers as the user machines.

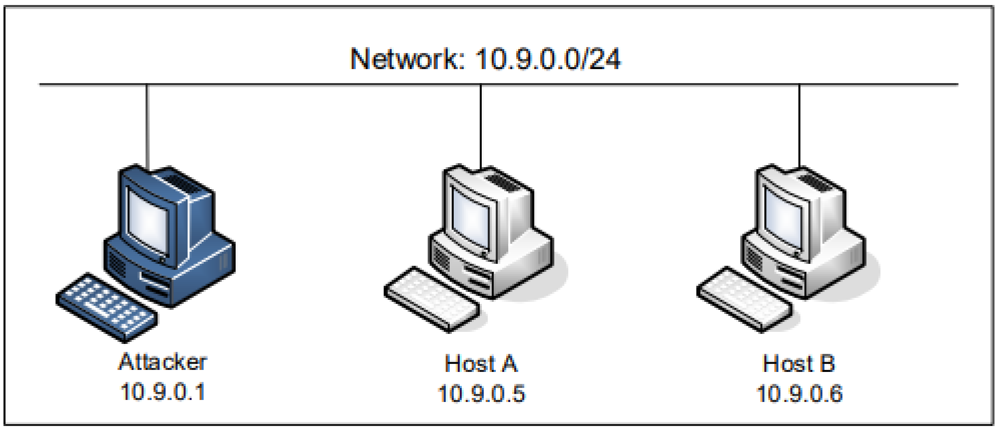

Figure 1 : Lab environment setup

NOTE :

On macOS, you can **directly compile and run the C programs inside the attacker container**. No separate host VM is required.

Helpful Commands:
You can **open a terminal in the attacker container** and **compile C files directly using gcc**.

For example :

```
gcc -o sniff Task2.1A.c -lpcap
./sniff
```

(**No need to use docker cp** since everything can be done inside the container.)

Task 2.1 : Sniffing - Writing Packet Sniffing Program

The objective of this task is to understand the sniffing program which uses the pcap library. With pcap, the task of sniffers becomes invoking a simple sequence of procedures in the pcap library. You should provide screenshots to show that your program runs successfully and produces expected results.

Task 2.1 A : Understanding how a Sniffer Works

In this task, you will execute a sniffer program that uses pcap library to print out the source and destination IP addresses of each captured packet.

- Open the file **Task2.1A.c.**
- Find the interface for the attacker machine. Change the interface value in the code to the interface of the attacker machine.
- Now, compile and run the following commands **directly inside the attacker container**:

```
# gcc -o sniff Task2.1A.c -lpcap

# ./sniff
```

- On Host A terminal :

```
# ping 10.9.0.1
```

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Answer the following questions:

Question 1: Describe the sequence of the library calls that are essential for sniffer programs using pcap.

Question 2: On the Attacker container run the command :

```
# su seed  (this is used to switch from root to normal user)
# ./sniff
```

Why do you need the root privilege to run sniffer? Where does the program fail if executed without the root privilege? Provide a screenshot of the output on the terminal and explain your observation.

Run the command to return to root user on the attacker container:

```
# su root
```

Question 3: The value 1 of the third parameter in the **pcap_open_live() function** turns on the promiscuous mode (use 0 to turn it off). Can you demonstrate the difference when this mode is on and off?

On the Attacker terminal run the command:

```
# gcc -o sniff Task2.1A.c -lpcap
# ./sniff
```

On Host A terminal :

```
# ping 10.9.0.6
```

Provide screenshots of your output on terminal and explain your observations.

Task 2.1 B : Writing Filters

Step 1 : Capture the ICMP packets between two specific hosts

In this task we capture all ICMP packets between two hosts. In order to do that, we need to modify the pcap filter of the sniffer code. The filter will allow us to capture ICMP packets between two hosts.

Change the interface value in the code to the interface of the attacker machine as done in previous tasks. Usually the interface of attacker machine will be of the form br-***.**

On the Attacker terminal run the command:

```
# gcc -o sniff Task2.1B-ICMP.c -lpcap
# ./sniff
```

In the host A machine ping any ip address

```
# ping 10.9.0.6
```

Provide screenshots of your output and explain your observations.

Step 2 : Capture the TCP packets that have a destination port range from to sort 10 - 100.

In this task we capture all TCP packets with a destination port range 10-100. Below we have the filter expression required to filter for TCP packets in a given port range.

We send FTP (runs over TCP) packets to the destination machine. As telnet runs over port 21, we should be able to capture all the packets sent with destination port 21.

Change the interface value in the code to the interface of the attacker machine as done in previous tasks. Usually the interface of attacker machine will be of the form br-***.**

On Attacker Machine terminal :

```
# gcc -o sniff Task2.1B-TCP.c -lpcap
# ./sniff
```

On Host A terminal :

```
# telnet 10.9.0.6
```

Provide screenshots of your output and explain your observations.

Task 2.1 C : Sniffing Passwords

Please show how you can use your sniffer program to capture the password when somebody is using telnet on the network that you are monitoring. It is acceptable if you print out the entire data part, and then manually mark where the password (or part of it) is.

Change the interface value in the code to the interface of the attacker machine as done in previous tasks. Usually the interface of attacker machine will be of the form br-***.**

On the Attacker terminal run the command:

```
# gcc -o sniff Task2.1C.c -lpcap
# ./sniff
```

On Host A terminal :

```
# telnet 10.9.0.6
```

Provide screenshots of your observations.

Task 2.2 Spoofing

The objective of this task is to create raw sockets and send spoof packets to the user/victim machine raw sockets give programmers the absolute control over the packet construction.

Task 2.2 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 as its source IP address). This packet should be sent to a remote machine on the Internet (the machine must be alive).

Open Wireshark before executing the program and select the same interface in Wireshark, as used in the code for each task i.e. the attacker machines interface.

On Attacker Machine terminal :

```
# gcc -o spooficmp Task2.2.c -lpcap

# ./spooficmp
```

Provide screenshots of your output and explain your observations.

Answer the following questions.

Question 4: Using the raw socket programming, do you have to calculate the checksum for the IP header?
Question 5: 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?

Task 2.3 Sniff and then Spoof

In this task, the victim machine pings a non-existing IP address “1.2.3.4”. As the attacker machine is in the same network, it sniffs the request packet, creates a new echo reply packet with IP and ICMP header and sends it to the victim machine. Hence the user will always receive an echo reply from a non-existing IP address indicating that the machine is alive.

We create a buffer of maximum length and fill it with an IP request header. We modify the IP header and ICMP header with our response data. In the new IP header, we interchange the source IP address and destination IP address and send the new IP packet using the raw sockets.

Open Wireshark before executing the program and select the same interface in Wireshark, as used in the code for each task i.e. the attacker machines interface.

Change the interface value in the code to the interface of the attacker machine as done in previous tasks.

On Attacker Machine terminal :

```
# gcc -o sniffspoof Task2.3.c -lpcap
# ./sniffspoof
```

On the Host A terminal ping 1.2.3.4

```
# ping 1.2.3.4
```

Provide screenshots of your observations.

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 explanations 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.