Chapter 12: Packet Sniffing and Spoofing

Problems

12.1. In the pcap-based sniffer program shown in Listing 12.3 (Section 12.2.3), we added a check to see whether `handle` is NULL or not. When running this program, we get an error message, saying `"NULL: No such device"`. What is the cause of the problem?

```
  handle = pcap_open_live("eth1", BUFSIZ, 1, 1000, errbuf);
  if (handle == NULL) {
      perror("NULL");
  }

Error message:
  NULL: No such device
```

12.2. In the pcap-based sniffer program shown in Listing 12.3 (Section 12.2.3), we replace Line ① with the following line. After that, when we run the sniffing program, we can only capture the packets in or out of our own computer, but we are not able to capture the packets among other computers that are on the same network. What is the cause of this problem?

```
  handle = pcap_open_live("eth3", BUFSIZ, 0, 1000, errbuf);
```

12.3. Which line in Listing 12.3 requires the root privilege? If you do not run the program with the root privilege, what is going to happen?

12.4. The `pcap_setfilter()` call in Listing 12.3 sets the filter inside the kernel, so it seems that the root privilege is required for this call to be successful. Please design an experiment to either prove or disapprove this hypothesis.

12.5. There are two typical approaches for a sniffer program to filter out unwanted packets. The first approach gets all the packets from the system, and then filters out unwanted ones, before presenting the results to users (or save to files). The second approach uses `pcap_setfilter` to set the filter (see Listing 12.3). Please describe the differences of these two approaches.

12.6. ★★

As we have learned that for a program to turn on the promiscuous mode or to simply be able to sniff packets on the local machine, the program needs to have a special privilege; normal users do not have that privilege. With this in mind, we checked the privilege of the `wireshark` process like the following:

```
$ pgrep wireshark
7598
```

```
$ ps  -fp 7598
UID   PID   PPID  C STIME TTY  TIME     CMD
seed  7598     1  0 10:01 ?    00:00:01 /usr/bin/wireshark
```

From the result, we can see that the UID (effective user ID) of the `wireshark` process is `seed`. We also know that `wireshark` can capture all the packets on the local network, regardless of whether a packet is from/to where the program runs. This does not seem possible based on our knowledge about how sniffers work. Please conduct an investigation to solve this mystery.

Hint: start packet capturing in `wireshark`, and then show what child process is launched by `wireshark`. You can use the "`pstree -p 7598`" command to get the IDs of process `7598`'s child processes. Focus on the program executed by the child processes.

12.7. The `tcpdump` program is also a sniffer program, which predates `wireshark`. This program only produces text-only outputs or saves the captured packets into files. The program needs to run with the root privilege. However, it comes with an `-Z` option, with which, the `tcpdump` program will drop privileges (if root) and changes the user ID to whatever is specified in the option. For example, if we use "`-Z seed`", the program's effective user ID will be dropped to `seed`. Please explain why this option is provided and how can the program still work after dropping the privilege.

12.8. Based on the investigation conducted in Problem 12.6., please describe how you can apply the *Principle of Least Privilege* to reduce the attack surface for the sniffer program shown in Listing 12.3.

12.9. An integer `0xAABBCCDD` is stored in a memory address starting from `0x1000`. If the machine is a Big-Endian machine, what is the value stored in addresses `0x1000`, `0x1001`, `0x1002`, and `0x1003`, respectively? If the machine is a Little-Endian machine, how is this integer stored?

12.10. A network protocol contains a four-byte integer, specifying the length of the payload in the packet. The implementation of this protocol has a mistake in it. When a packet is received, the protocol implementation needs to copy the payload to a buffer. It first copies the length field from the packet header to a variable, but the program forgets to convert the number into the host order. Assume the value of this variable is X. The program then allocates X bytes of memory to hold a copy of the payload. On a Little-Endian machine, if the payload of a received packet is 255 bytes, how much memory will be allocated? What is a likely consequence of this mistake?

12.11. Please describe the printing result of the following program on (1) a Little-Endian machine, and (2) a Big-Endian machine.

```
void main()
{
  int a = 255;
  printf("%u\n", htonl(a));
  printf("%u\n", ntohl(a));
}
```

12.12. Assume that machines A and B are on the same network `10.3.2.0/24`. Machine A sends out spoofed packets, and Machine B tries to sniff on the network. When Machine A spoofs packets with a destination `1.2.3.4`, B can always observe the spoofed packets. However, when Machine A tries to spoof packets with a destination IP address `10.3.2.30`, B cannot see the spoofed packets. There is nothing wrong with the spoofing or sniffing program. Apparently, the spoofed packet has never been sent out. What could be the reason?

12.13. A news report says that company XYZ's network was attacked by outsiders, who apparently sent a lot of spoofed ARP requests/responses from remote machines to the company's network, trying to launch ARP cache poisoning attacks. Please comment on whether this is fake news or not.

12.14. Is it possible to spoof a packet with a size larger than `65535`, which is the up limit of the IP packet size (the length field in the IP header has only 16 bits)?

12.15. In the past, one can send a broadcast packet to all the machines on a subnet. This is called *Directed Broadcast*. If the subnet is `10.0.2.0/24`, the directed broadcast address is `10.0.2.255`. How can we use this feature to launch a denial-of-service attack on a victim? Basically, we would like to send a lot of packets to the target machine, but we cannot afford to do it ourselves, because the target has a larger bandwidth than us. We need to find a way to turn one packet into many.

12.16. When an UDP server provides a response to a request, the size of the response packet is significantly large than the size of the request packet. Please leverage this UDP server to magnify your power in a denial-of-service attack against a victim machine. Hint: search the term "UDP Amplification Attacks" to learn more about this type of attack.