Packet Sniffing Writeup

Ethical Hacking 2021/22, University of Padua

Eleonora Losiouk, Alessandro Brighente, Denis Donadel, Gabriele Orazi

1 Task 1.1A

Simply run the program and sniff traffic. You have to generate some network traffic to see something. For instance, you can ping two hosts. Running the script as not-sudo raise an exception (no permissions).

2 Task 1.1B

- Capture only ICMP: `icmp`
- Capture any TCP packet that comes from a particular IP and with a destination port number 23: `tcp and dst port 23`.

Hint: telnet works on port 23, you can use it to test your filter.

Hint: Berkeley Packet Filter is used in Wireshark as well, so you can easily try filters on a network capture in Wireshark.

Hint: if the filter is not working, you may need to install `tcpdump` packet (on Debian-based distro: `sudo apt install tcpdump`).

3 Task 1.2

You can use a simple modification to the code provided to send spoofed ICMP requests. You can also spoof the `a.src` value to modify the origin IP and so redirect the response to someone else.

4 Task 1.3

You have to write a tool that starts from TTL=1 (which will be your router) and increase its value until the ping is successful.

Warning: eduroam blocks your ICMP requests.

5 Task 1.4

You can observe that:

- `ping 1.2.3.4`: simply spoof the response message.
- `ping 10.9.0.99`: nothing happens. This is because the IP is in the same subnet, but the ARP request do not receive a valid response. You can see this behavior using, for instance, Wireshark. If you want, you can also spoof the ARP response, but it is not required.

- `ping 8.8.8.8`: in this case, you must see the spoofed response, but you will get another response from the right IP too.

Hint: try to craft the ICMP response to be as similar as possible to the original one. You have, for instance, to add the right Raw data and to set the correct ICMP parameters.

---

6 Task 2.1A

Hint: you will need to install libpcap (`sudo apt install libpcap-dev`)

Question 1:

1. `pcap_open_live`: generate the session, set the device to monitor and other options (e.g., promisc mode)
2. `pcap_compile`: if needed, compile a filter
3. `pcap_setfilter`: set a previously compiled filter
4. `pcap_loop`: monitor the traffic and perform an action when receiving not filtered packets

Question 2:

Without root access, you cannot access the traffic directly. The program fails without it since it cannot access the packets and so `pcap_open_live` raises an exception.

Question 3:

Hint: you can use `sudo ip link set [interface] promisc [on|off]` to activate/deactivate promisc mode on the interface. To check the promisc state use `ip -d link show dev [interface] | grep promisc`.

7 Task 2.1B

- `icmp and (host 10.9.0.5 and host 10.9.0.6)`
- The syntax can be slightly different between Wireshark, Scapy and C libraries: `tcp and tcp.port > 10 and tcp.port < 100` (Wireshark) or `tcp port > 10 and tcp port < 100`.

8 Task 2.1C

You can simply print all the payloads, or you can create a script to start printing after `Password:` and up to `\r`, which is the closing char of the password.

9 References

- Berkeley Packets Filter (BPF) guide
- Scapy Docs
- pcap.h lib guide