Simple Packet Analysis

This lab aims to practice simple packet analysis using tcpdump (or wireshark). You have to interact with the server and get a secret flag sent from the server. Once you obtain the secret flag from the server, you can then ask the server to verify the correctness of the flag.

Warning: Most UNIX operating systems have packages for tcpdump and wireshark. Nevertheless, wireshark can be downloaded from its official site.

Description

1. Run tcpdump (or wireshark) in your preferred environment.
2. Connect to the challenge server by running nc -u inp.zoolab.org 10495 from your preferred environment. This step can be done either inside or outside a docker.
3. Request a **challenge id** from the server by sending the command hello <id>, where <id> is your user id. Suppose a given <id> is chuang. You should receive an OK response from the server along with a **challenge id** composed of your id and a random string. chuang_d59acd9fb5577b76f0a82990c815eed8
4. Request a challenge using the obtained **challenge id**. You have to send a command chals <chals_id>, where the <chals_id> is the challenge id you received from the previous command.
5. The server sends many UDP packets, and each payload is in the form of SEQ/NNNNN:message. The NNNNN is a sequence number (starting from zero) for you to determine if there is any packet loss. The message can be one of the following:
    - An arbitrary length of character A's.
    - The BEGIN FLAG message
    - The END FLAG message

    You can then get the flag of this challenge by following the steps:
    a. Recognize the packet having the payload BEGIN FLAG. Once you have recognized the packet, perform flag extraction using the next few steps.
    b. For each packet, obtain its **x-header-length** by summing up the **length** of the IP options and UDP payloads. Suppose 10 packets were received after the BEGIN FLAG packet. You should get 10 **x-header-length**s.
    c. Convert each **x-header-length** into its corresponding ASCII character.
    d. The secret flag can be obtained by concatenating each ASCII character obtained in the second step in its order.
    e. You can find the end of the flag by recognizing the END FLAG message in a packet's UDP payload.
6. Decode the flag and verify if your flag is correct by connecting to the challenge

server and using the verfy <flag> command to verify your decoded flag. The server responds GOOD JOB! if your flag is correct.

Demonstration

1. [50%] If you can implement a program to request the server to send challenges. Your program should be able to send different user id requested by the TAs.
2. [30%] If you can capture the challenge packets, save them in a pcap file, and decode a correct flag from the saved packets.
3. [20%] If your program can perform the above two steps in a single implementation (command) and display the decoded flag in the terminal.
4. For steps (2) and (3), the flag verification process can be performed manually.

Warning: A sample pcap file can be downloaded from here.

Info: Your program may handle the outputs from the command:

tcpdump -ni any -Xxnv udp and port 10495.

Info: Note that your script (or program) can use a saved pcap file created by tcpdump or wireshark as its input.

Network Issues

If you have network connection issue with our challenge server, you can download the server binary from here, and run it in your Linux docker using the command:

sudo ./udpflag_release 10495

Danger: You need root permission to run the server program.

Please note that you need to grant execution permission to the downloaded executable. If you are working with an Apple chip Mac, you have to install Rosetta (x86_64 emulation) to run this executable in your Linux Docker.

If everything is fine, the server should only display a single message server:

running on port 10495.

If you have successfully invoked the server, you should be able to connect to the server using the command: **nc -u 127.0.0.1 10495**

and the behavior should be the same as the remote challenge server.