Homework 2
COSI 141A-1: System and Network Security

In this homework assignment, you will learn the concepts and tools for passive and active network analysis. The required files for this assignment are contained in `hw2_files.zip`. Make sure you read the handout before starting.

1 Getting Started

You are required to have a POSIX-compatible environment for all homework assignments, and it is highly recommended that you set it up early. The primary programming language will be Python 3.9 or above. Windows users use WSL2 or PowerShell for full compatibility with networking tools.

2 Wireshark (36 points)

In this first part of the assignment, you will learn about Wireshark, and its packet filtering and capturing features. You can install the latest version from link. `hw2_files.zip` provided contains two packet traces which you will use for this part by loading them into Wireshark.

For questions 1-6, use `trace1.pcap`. For questions 7-9, use `trace2.pcap`.

1. What are the total number of packets sent by the host `198.105.254.25`?
2. What is the source MAC address of the machine generating the ARP packets?
3. What is the cookie `session-id` in the Host header in the first GET request of the trace?
4. There is one DNS query to a `.edu` domain. What is the domain name in the query?
5. The trace contains a search query made from source IP: `192.168.0.100` to destination IP: `207.171.162.173`. What is it? Provide your response as the *exact* keyword which the user typed.
6. Extract the contents of the JPEG image corresponding to the GET request for `/pic_sm/101.jpg` to the destination host `50.93.210.4`. Save the image as `6.jpg` and include it the final zip archive.

(Reminder: Use `trace2.pcap` for questions 7 to 9)

7. What is the POP username?
8. What is the POP password?
9. How many email messages are there in the email account?
10. In this task, you use Wireshark (or tcpdump) to capture live traffic and save it as a packet trace. Using `curl`, make a request to `http://example.com`. **You should explicitly define the http:// protocol in your request.** Capture and save all relevant HTTP and DNS traffic and save it to a file named `10.pcap`. You should filter out any packets not a part of the request.

3 TCPDUMP (16 points)

Similar to Wireshark, TCPDUMP is a command line tool for passive network analysis. You can download and read about TCPDUMP here. For this part of the assignment you will only use `trace1.pcap`. To load the file into TCPDUMP you can use `tcpdump -n -r trace1.pcap`.

11. All packets sent from the host `131.253.40.84`.
12. Traffic received at port `80` by all the hosts.
13. Command to display only the ARP packets in the trace.
14. Filter all TLS/HTTPS traffic to the host `74.125.225.81`. *Hint: Remember the port commonly used for HTTPS (443).*

4 Nmap (20 points)

Nmap (or Zenmap) is an open source tool widely used to discover open ports, and hosts on the Internet and to also create a network map. In this task of the assignment, you will use Nmap to scan certain hosts and evaluate what services they are running. You can read about and install the version of Nmap compatible with your OS from here. For this task you will use `scanme.nmap.org` to answer the following questions.

15. List all the open TCP ports on the host. Provide your responses as comma-separated values in a single line and include the service names/versions.
16. Provide the exact Nmap command you would run to discover open UDP ports.
17. Nmap also provides a guess of OS running on the host. From your scan, what OS did Nmap detect on the host?
18. Identify one vulnerability reported by Nmap on the scanned host and briefly explain what the vulnerability means.

5 Packet Spoofing (28 points)

For the task of this assignment, you will learn how to create your own packets and send them over the network. Essentially, you will use Python's scapy module to write a spoofer. Below is a list of requirements that your code must meet to receive full credit.

- All your code should be contained as part of a single function defined as `send_packet`
- Your function should take in 4 arguments: `src_ip, dst_ip, dst_port, payload`
- The `send_packet` function should create a spoofed UDP packet with the payload and send it to the destination IP and port specified
- You should ensure that the payload does not exceed 150 bytes. If it does, your function should gracefully exit.
- For this task, you are only allowed to use the Python scapy module
- Save the file as `spoofer.py` and include it in the zip archive.

6 Submission

In total, you will submit the following 4 files in a compressed zip archive named `hw2.zip`:

1. `hw2.txt`: This file contains all text-based responses.
   Each answer must follow the format: <`filter/command`>`;` <`answer`>, corresponding to the filter you used in each tool (Wireshark, `tcpdump`, or Nmap) to obtain the result.

   Example: `1.  tcp.port == 80 ; 5 packets`. If a question requires multiple filters or manual analysis, include all relevant steps.

2. `6.jpg`: The image extracted in question 6.

3. `10.pcap`: The captured pcap in question 10.

4. `spoofer.py`: The file containing the code for the packet spoofing task.