Lab6 - Packet Sniffing and Spoofing Lab

Owner	Hetu Virajkumar Patel
Verification	Verified
Tags	compsci tmu

General Questions

- **What is packet sniffing and why is it important to understand it?**
  - Packet sniffing is the process of capturing and analyzing network traffic. It is crucial for network security professionals to understand it to identify potential threats and vulnerabilities.
- **What is packet spoofing and how can it be used for malicious purposes?**
  - Packet spoofing is the technique of disguising the source IP address of a packet to deceive the recipient. It can be used for various malicious activities like man-in-the-middle attacks and denial-of-service attacks.

- **Why is root privilege required for packet sniffing and spoofing?**
  - Root privilege is needed because packet sniffing and spoofing involve low-level network operations that require direct access to network interfaces. These operations are typically restricted to root users to prevent unauthorized access.
- **What are the ethical implications of packet sniffing and spoofing?**
  - These techniques can be used for both legitimate and malicious purposes. It's important to use them responsibly and ethically, and to adhere to relevant laws and regulations.
- **How can network administrators defend against packet sniffing and spoofing attacks?**
  - Implement strong network security measures such as firewalls, intrusion detection systems, and encryption protocols. Additionally, educate users about potential threats and best practices for secure network usage.

Task 1.1: Sniffing Packets

- **Why is it important to use a filter when sniffing packets?**
  - Filters help reduce the amount of captured traffic and focus on specific types of packets, making analysis more efficient.
- **What is the significance of using promiscuous mode?**
  - Promiscuous mode allows a network interface to capture all packets transmitted on the network, not just those destined for the interface itself. This is useful for capturing all traffic on a network segment.

- **How does Scapy simplify packet sniffing compared to using raw sockets?**
  - Scapy provides a high-level interface for packet manipulation and analysis, making it easier to write packet-sniffing programs without dealing with low-level socket programming details.
- **Why did you choose to use the `icmp` filter in the initial sniffing example?**
  - ICMP is a simple protocol that is frequently used for diagnostic purposes, making it a good choice for a basic sniffing example.
- **What are the advantages and disadvantages of using Scapy for packet sniffing?**
  - **Advantages:** Scapy provides a high-level interface, making it easy to write complex packet manipulation scripts. It also offers a rich set of features for packet analysis and modification.
  - **Disadvantages:** Scapy can be less efficient than lower-level libraries like pcap for high-performance sniffing scenarios.

Task 1.2: Spoofing ICMP Packets

- **How does Scapy allow you to spoof packets?**
  - Scapy provides functions to create and modify packet headers, including the source IP address.
- **What are the potential security implications of packet spoofing?**
  - Packet spoofing can be used to impersonate other devices on the network, leading to various attacks like man-in-the-middle attacks and denial-of-service attacks.

- **How does Scapy handle checksum calculations when spoofing packets?**
  - Scapy automatically calculates and sets the appropriate checksum values for the IP and ICMP headers, ensuring that the spoofed packets are valid.
- **What is the significance of setting the `ttl` field in the IP header?**
  - The `ttl` field specifies the maximum number of hops a packet can traverse before being discarded. By setting a low `ttl` value, we can control the number of routers the packet passes through, potentially limiting the impact of the spoofed packet.

Task 1.3: Traceroute

- **How does traceroute work using ICMP packets?**
  - Traceroute sends ICMP echo request packets with increasing TTL values to determine the path to a destination host. By analyzing the received ICMP Time Exceeded messages, the traceroute tool can identify the routers along the path.
- **Why is ICMP used for traceroute?**
  - ICMP is a reliable protocol that can be used to send error messages, such as Time Exceeded messages, which are essential for determining the path to a destination host.
- **How can the `ttl` field be used to determine the number of hops to a destination?**
  - By incrementing the `ttl` value in each packet sent, we can identify the routers along the path by observing the Time Exceeded messages received.

Task 1.4: Sniffing and Spoofing

- **How can sniffing and spoofing be combined to launch attacks?**
  - By sniffing network traffic, an attacker can identify potential targets and then use spoofing techniques to impersonate other devices to launch attacks.
- **What is the role of the ARP protocol in packet spoofing attacks?**
  - ARP spoofing involves sending forged ARP packets to associate a malicious IP address with a legitimate MAC address, allowing the attacker to intercept traffic intended for the legitimate device.
- **How does ARP spoofing work, and how can it be used to facilitate packet spoofing attacks?**
  - ARP spoofing involves sending forged ARP packets to associate a malicious IP address with a legitimate MAC address. This allows the attacker to intercept and manipulate traffic intended for the legitimate device.
- **What are some countermeasures to prevent ARP spoofing attacks?**
  - Implement static ARP entries, use ARP inspection features in network devices, and deploy intrusion detection systems to detect and mitigate ARP spoofing attempts.

Task 2.1: Writing Packet Sniffing Program

- **Why is the pcap library used for packet sniffing?**
  - The pcap library provides a low-level interface for capturing packets, allowing for fine-grained control over the sniffing process.
- **What is the role of the `pcap_open_live` function?**

    - This function opens a live capture session on a specified network interface.
- **How does the BPF filter work?**
    - BPF filters are used to specify the types of packets to capture, based on various criteria like IP addresses, port numbers, and protocol types.
- **Why is it necessary to compile the C program before running it in the container?**
    - C programs need to be compiled into machine code before they can be executed. This compilation process converts the source code into instructions that can be understood by the processor.
- **How does the `pcap_open_live` function handle errors?**
    - If an error occurs during the opening of the live capture session, the function returns NULL and sets an error message in the `errbuf` parameter.

Task 2.2: Writing Packet Spoofing Program

- **How are raw sockets used for packet spoofing?**
    - Raw sockets allow direct access to the network layer, enabling the construction and transmission of custom packets.
- **What are the challenges of using raw sockets for packet manipulation?**
    - Raw sockets require careful handling of packet headers, checksum calculations, and network byte order.
- **What are the challenges of manually calculating checksums for IP and TCP headers?**

    - Manually calculating checksums can be error-prone and time-consuming. It's important to ensure that the checksums are calculated correctly to avoid packet loss or corruption.

- **Why is it important to set the appropriate source IP address and port number in the spoofed packet?**
    - The source IP address and port number determine the identity of the sender of the packet. By spoofing these values, the attacker can deceive the recipient into believing that the packet originated from a legitimate source.

Task 2.3: Writing a Combined Sniffing and Spoofing Program

- **How does combining sniffing and spoofing techniques enhance the effectiveness of network attacks?**
    - By combining sniffing and spoofing, attackers can gain valuable information about network traffic patterns and then exploit these vulnerabilities by manipulating network packets. For example, an attacker can sniff network traffic to identify potential targets and then use spoofing techniques to impersonate legitimate devices, intercept sensitive information, or launch denial-of-service attacks.
- **Why is it important to handle multiple packets concurrently in the combined program?**
    - Network traffic can be bursty, and multiple packets may arrive simultaneously. The program needs to handle these packets efficiently to avoid missing important information or delaying responses.
- **How can you ensure that the spoofed responses are sent to the correct destination IP addresses?**

    - The program should extract the destination IP address from the sniffed ICMP echo request packet and use it as the destination IP address for the spoofed ICMP echo reply packet.

- **What are the potential performance implications of combining sniffing and spoofing in a single program?**
    - The program may consume significant CPU and network resources, especially when handling high traffic volumes. It's important to optimize the code to minimize performance overhead.

Task 2 Questions from the Doc:

Question 1 Please use your own words to describe the sequence of the library calls that are essential for sniffer programs

To create a simple packet sniffer, two libraries are utilized:

- stdio.h: This is the standard library for input and output operations in C.
- pcap.h: This library is specifically designed for packet capture.

The pcap library facilitates the handling of sockets required for network communication. It simplifies the process of sniffing packets, eliminating the need to manually code packet structures. By using pcap, we can open a raw socket in promiscuous mode, allowing it to capture all traffic on the network. The raw socket is opened with the pcap_open_live() function. Additionally, we can apply filters to focus on specific types of traffic using the pcap_compile() and pcap_setfilter() functions. Packet capture is performed using the pcap_loop() function.

Question 2 Why do you need the root privilege to run a sniffer program?

The sniffer program must operate the network interface card in promiscuous mode, a capability that is restricted to the superuser (root). Attempting to run the program without root privileges results in an error, typically indicating a lack of access rights. Specifically, the program will encounter a failure when invoking the pcap_open_live function to set up a socket with the NIC enp0s3 in promiscuous mode, as this operation is not permitted for standard user programs.

Question 3. Please turn on and turn off the promiscuous mode in your sniffer program. The value 1 of the third parameter in pcap open live() turns on the promiscuous mode (use 0 to turn it off). Can you demonstrate the difference when this mode is on and off? Please describe how you can demonstrate this. You can use the following command to check whether an interface's promiscuous mode is on or off (look at the promiscuity's value).

In Task 2.1A, we enabled promiscuous mode by setting the third parameter of the pcap_open_live function to 1. This allowed us to capture network traffic and observe packets transmitted by other users. Now, we will change the value of the third parameter to 0 to disable promiscuous mode and repeat the same activity as before.

Question 4. Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?

The IP packet length field can be set to any value greater than or equal to 20. Values below 20 cause the sendto() function to throw an error due to invalid arguments, as 20 bytes is the minimum size for an IP packet header without payload. While values above 20 allow the packet to be sent, the system

overwrites the total length to the actual packet size during transmission.

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

Regarding checksums for the IP header in raw socket programming, manual calculation is unnecessary. By default, setting ip_check = 0 allows the kernel to handle checksum calculation. However, if a different value is set, a custom checksum method must be implemented.

Question 6. 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?

Root privileges are required for programs using raw sockets because they need to set the Network Interface Card (NIC) to promiscuous mode. This mode allows capture of all network packets. Without root privileges, the program fails at the socket creation stage, producing an error indicating that the operation is not permitted. This occurs because non-privileged users cannot enable promiscuous mode on the NIC.