Sniffing and Spoofing Lab

Table of Contents:

Lab Environment Setup **2**

Lab Task Set-1: Using Tools to Sniff and Spoof Packets using Scapy **3**

Task 1.1 : Sniffing Packets 3

Task 1.1 A : Sniff IP packets using Scapy 3

Task 1.1 B : Capturing ICMP, TCP packet and Subnet 4

Task 1.2 : Spoofing 6

Task 1.3 : Traceroute 7

Task 1.4 : Sniffing and-then Spoofing 8

Submission 8

Lab Environment Setup

Please download the **Labsetup.zip** file from the link given below :
https://seedsecuritylabs.org/Labs_20.04/Networking/Sniffing_Spoofing/

Follow the instructions in the **lab setup document** to set up the lab environment.

In this lab, we will use three machines that are connected to the same LAN. We can either use three VMs or three containers. Figure 1 depicts the lab environment setup using containers. We will do all the attacks on the attacker container, while using the other containers as the user machines.

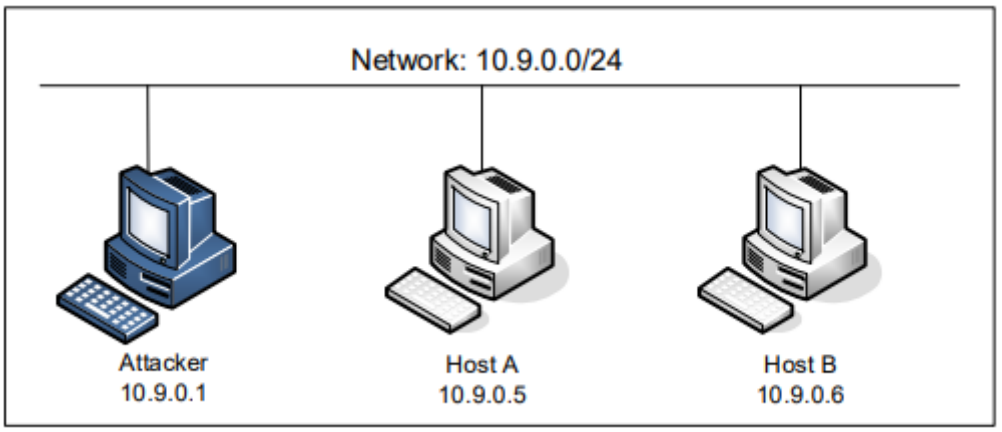

Figure 1 : Lab environment setup

Task 1.1 : Sniffing Packets

The objective of this task is to learn how to use Scapy to do packet sniffing in Python programs.

Task 1.1 A : Sniff IP packets using Scapy

- Open the file named **Task1.1A.py**, Check the **Lab setup instructions document** for detailed instructions on how to find out the network interface of your attacker machine.
- **Replace the network interface in the code wherever required**. (it will usually start with br-############).
- Open Wireshark and select the same network interface as that of attackers.
- By default, in these Network Security labs, the attacker container has the root privilege. This setting is made in the docker set up.

Step 1 :

- Run **Task1.1A.py** from the **attacker container terminal** and demonstrate that you can indeed capture the packets.

  # python3 Task1.1A.py

- From the host A machine's terminal ping a random IP address(let's say for example : 8.8.8.8)

  # ping 8.8.8.8

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Step 2 :

- Run the same program - **Task1.1A.py** from the attacker terminal, without root privilege. This can be done by executing the command **su seed**.

  # su seed

  $ python3 Task1.1A.py

- Do you find any issues? If so, why?
- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Step 3 :

Return to root user on the attacker container:

 # su root

Task 1.1 B : Applying Packet Filters to Capture certain types of packets for example : ICMP, TCP packet and Subnet

Usually, when we sniff packets, we are only interested in certain types of packets. We can do that by setting filters in sniffing. Scapy's filter uses the BPF (Berkeley Packet Filter) syntax as discussed in the class. The task involves capturing packets of certain kind by setting the below mentioned filters. Each filter should be set separately.

Step 1 : Capture only the ICMP packet

- Open file named **Task1.1B-ICMP.py**
- Fill in the network interface of the attacker machine in the given program.
- Open Wireshark and select the same network interface as that of attackers.
- Observe the filter expression in the code – Only ICMP packets are captured by the Scapy sniffer program. Hence, when some machine on the same network sends ping requests, the packets gets captured by the sniffer.
- On the Attacker terminal run the command:

  # python3 Task1.1B-ICMP.py

- From the host A machine's terminal ping a random IP address(8.8.8.8)

  # ping 8.8.8.8

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Step 2 : Capture any TCP packet that comes from a particular IP and with a destination port number 23

- Open file named **Task1.1B-TCP.py**
- Fill in the network interface of the attacker machine in the given program.
- Ensure Wireshark is up and running. Select the same interface as the one in the program.
- The program must capture the TCP packets being sent from the specified IP address on the port 23.
- On the Attacker terminal run the command:

  # python3 Task1.1B-TCP.py

- From the host A machine's terminal telnet to a random IP address.

  # telnet 10.9.0.1

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Step 3 : Capture packets that come from or go to a particular subnet

- Open file named **Task1.1B-Subnet.py**
- Ensure Wireshark is running.
- Fill in the network interface of the attacker machine in the given program.
- In the filter expression, you can pick any subnet. Let us say for example we monitor the subnet 192.168.254.0/24; you should not pick the subnet that your VM is attached to.
- On the Attacker terminal run the command:

  # python3 Task1.1B-Subnet.py

- From the host A machine's terminal, ping a random IP address on the chosen subnet.

  # ping 192.168.254.1

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Task 1.2 : Packet Spoofing

The objective of this task is to spoof IP packets with an arbitrary source IP address. We will spoof **ICMP echo request packets** and send them to another VM on the same network. We will use Wireshark to observe whether our request will be accepted by the receiver. If it is accepted, an echo reply packet will be sent to the spoofed IP address. Below shows the code to create the ICMP packet. The spoofed request is formed by creating our own packet with the header specifications.

```
#!/usr/bin/python3
from scapy.all import *
print ("SENDING SPOOFED ICMP PACKET...")
IPLayer = IP()
IPLayer.src="10.9.0.1"
IPLayer.dst="10.9.0.5"
ICMPpkt = ICMP()
pkt = IPLayer/ICMPpkt
pkt. show ()
send(pkt,verbose=0)
```

Step 1 :

- Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machines interface.
- Here we will spoof ICMP echo request packet from any machine within the local network and destination IP address of any remote machine on the internet which is alive.
- On the Attacker terminal run the command:

 # python Task1.2A.py

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Step 2 :

- Here we will spoof an ICMP echo request packet with an arbitrary source IP address.
- On the Attacker terminal run the command:

 # python Task1.2B.py

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Task 1.3 : Traceroute

- The objective of this task is to implement a simple traceroute tool using Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination.
- The code in **Task1.3.py** is a simple traceroute implementation using Scapy. It takes hostname or IP address as the input from the command line. We construct an ICMP packet. We send the packet to the input host using function sr1(). This function waits for the reply from the destination. If the ICMP reply type is 0, we receive an echo response from the destination, else we increase the TTL value and resend the packet.
- Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machine's interface.
- On the Attacker terminal run the command:

 # python3 Task1.3.py <Enter any IP address here>

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Task 1.4 : Sniffing and-then Spoofing

- In this task, the victim machine(Host A) pings a non-existing IP address “1.2.3.4”.
- As the attacker machine is on the same network, it sniffs the request packet, creates a new echo reply packet with IP and ICMP header and sends it to the victim machine.
- Hence, the victim machine(Host A) will always receive an echo reply from a non-existing IP address indicating that the machine is alive.
- The code in **Task1.4.py** sniffs ICMP packets sent out by the victim machine. Using the callback function, we can use the packets to send the spoofed packets. We retrieve source IP and destination IP from the sniffed packet and create a new IP packet. The new source IP of the spoofed packet is the sniffed packet’s destination IP address and vice versa. We also generate ICMP packets with id and sequence number. In the new packet, ICMP type should be 0 (ICMP reply). To avoid truncated packets, we also add the data to the new packet.
- Fill in the interface of the attacker machine in the given program and run the code.
- Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machine’s interface.
- On the Attacker terminal run the command:

  # python3 Task1.4.py

- From the host A machine’s terminal (victim machine) ping 1.2.3.4

  # ping 1.2.3.4

- Provide a screenshot of the output on the terminal and from the Wireshark and explain your observation.

Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to provide explanations to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.