Programming assignment 5

Due date

May 2, 2023

Academic Integrity

The following actions are strictly prohibited and violate the honor code. The minimum penalty for plagiarism is a grade of zero and a report to the Aggie honor system office.

- Uploading assignments to external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from external websites or tutoring websites such as Chegg, Coursehero, Stackoverflow, Bartleby, etc.
- Copying code from a classmate or unauthorized sources and submitting it as your work.

Keywords

TCP/IP, sockets, packet capture, network security, sniffing, spoofing, social engineering

Introduction

In this programming assignment, you will attack a server using spoofing and sniffing to get valuable information back from the server. You need PCAP for this packet manipulation. Here is the PCP documentation.

Download and run the lab environment

1. Clone the assignment, cd into the assignment folder
2. Go to the `environment` directory
3. Switch to the seed user by typing `sudo su seed`
4. Run `dcbuild`
5. Run `dcup` to start the dockers
6. Get another shell and again switch to seed user
7. Run `dockps` to list the dockers running
8. Run `docksh <first two digits of id>` to ssh into the corresponding docker
9. Inside the docker environment, cd to volumes, and you should see your PA code in there (you can still modify things as a normal user, no need to login every time)
10. Now use the victim container to host the server container attacker for the spoofer and sniffer. You need to open 2-3 terminals for this assignment, one `docksh` into the attacker and one or two `docksh` into the victim (read the implementation note for more info). Since we have the shared volumes folder, compiling code in one container directly affects the other.

Docker commands cheat sheet

These aliases are already defined for you in the .bash_aliases file of the seed user

Command	Alias
docker-compose build	dcbuild
docker-compose up	dcup
docker-compose down	dcdown
docker ps --format "{{.ID}} {{.Names}}"	dockps
docker exec -it <id> /bin/bash	docksh <id>

Requirements

In this assignment, you will write a sniffer and a spoofer program. The goal is to spoof packets with a fake identity accepted by the server. The server then parses the data from your packet to its reply message destination address along with the requested data.

Packet sniffing is the practice of intercepting and inspecting packets while they are flowing through the network. This can be for monitoring or malicious purposes.

Packet / IP spoofing can create Internet Protocol (IP) packets with a source IP address to conceal the sender's identity or impersonate another user.

Social engineering is a deception technique to trick individual(s) into divulging confidential or personal information that may be used for fraudulent purposes. Figure 1 shows an IP spoofing attack in which the attacker gets confidential data from the server.

From some leaked documents, you learn that the server only processes packets from a specific address, `154.123.122.111`, and drops everything else it receives.

The security maintainer is an Aggie who did not take CSCE 313 seriously and forgot to put in any protection, so the server does not do any other checks besides looking at the sender's IP address before processing the data. This means that by sending a request with the matching IP address, you can tell the server to execute various tasks, including sending back sensitive information.

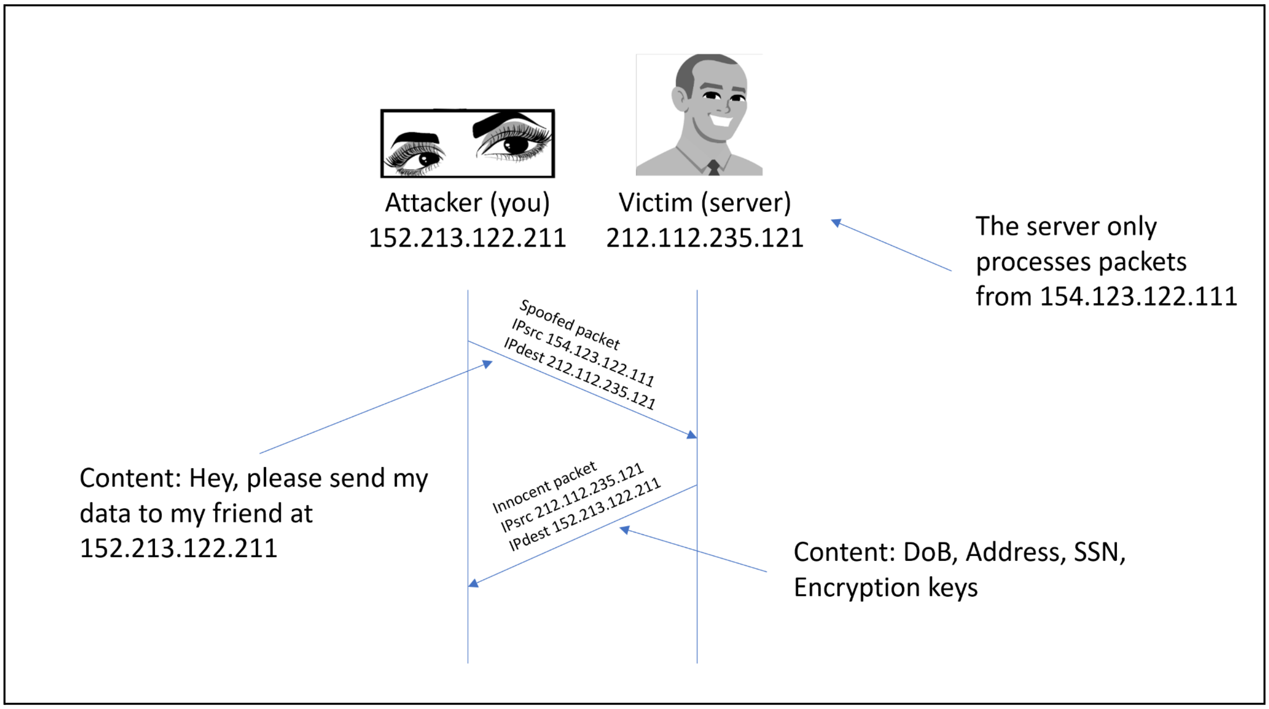


Figure 1: Basic IP spoofing attack. (All IP addresses above are for demonstration purposes. Read the document for the correct addresses)

Your goal is to implement the attack:

1. Write a spoofer that sends spoofed packets with the IP address `192.168.0.1` to the server with the IP address `10.9.0.5`.
2. Write a sniffer with IP address `10.9.0.1` that monitors the network for confidential packets sent back from the victim.

Implementation Note

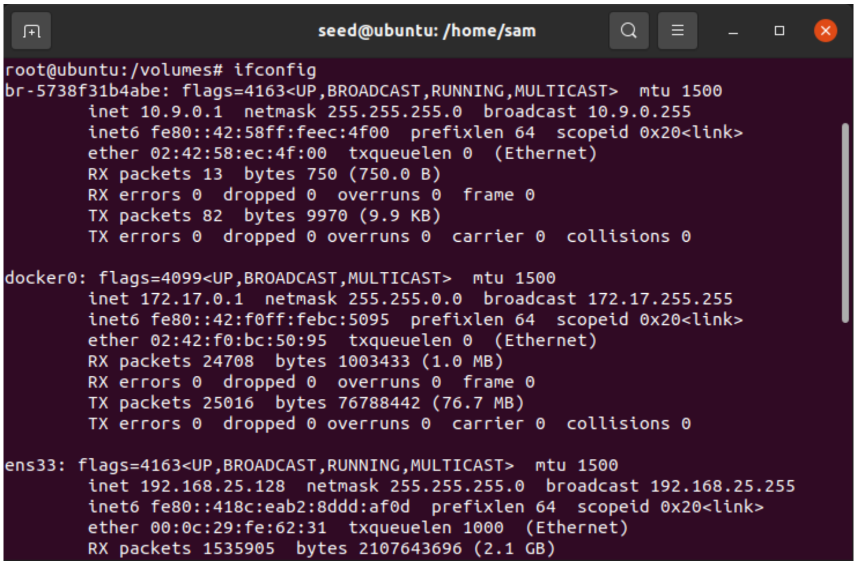
1. The server is already implemented (besides send_raw_ip_packet, which the server relies on). The only item that should be running on the victim container is the server; running it on the attacker will not work since the IP address is hard coded. All the correct IP addresses are provided to you in common.h, no need to modify those.

2. You will need to modify: sniff.cpp, spoof.cpp, and send_raw_ip_packet in common.h (no need to modify anything outside the TO DO comment)

3. Recommended completing steps:

   a. sniff.cpp -> spoof.cpp -> send_raw_ip_packet

   b. for sniffer's pcap_open_live, to get the correct interface, do ifconfig and select the one on top (for the pic below, it is br-5738f31b4abe as the correct interface)



```
root@ubuntu:/volumes# ifconfig
br-5738f31b4abe: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:58ff:feec:4f00  prefixlen 64  scopeid 0x20<link>
        ether 02:42:58:ec:4f:00  txqueuelen 0  (Ethernet)
        RX packets 13  bytes 750 (750.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82  bytes 9970 (9.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:f0ff:febc:5095  prefixlen 64  scopeid 0x20<link>
        ether 02:42:f0:bc:50:95  txqueuelen 0  (Ethernet)
        RX packets 24708  bytes 1003433 (1.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25016  bytes 76788442 (76.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.25.128  netmask 255.255.255.0  broadcast 192.168.25.255
        inet6 fe80::418c:eab2:8ddd:af0d  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:fe:62:31  txqueuelen 1000  (Ethernet)
        RX packets 1535905  bytes 2107643696 (2.1 GB)
```

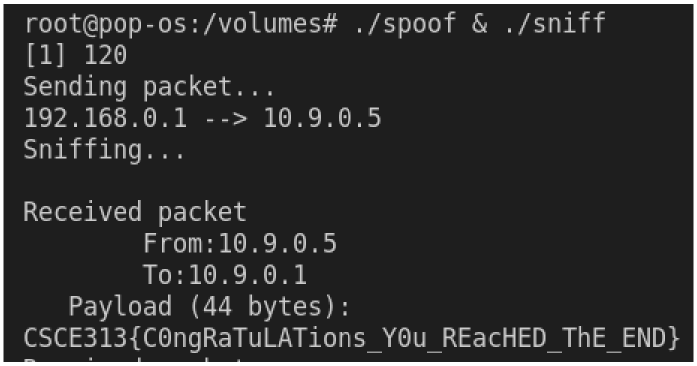
   c. you will need 3 shells to test: 1 for the victim, 2 for the attackers

   d. boot up the server in victim

   e. run sniffer in one shell then spoofer on the other shell

   f. you can also run spoofer in the background and then sniffer if you want to put them in the same shell

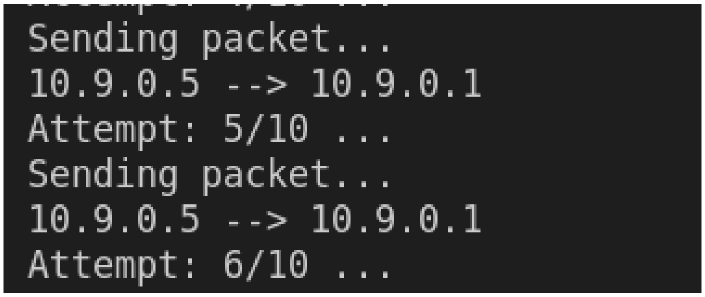
   spoofer and sniffer running correctly - note that we have multiple flags, so this might not be the one you will get, but all flags start with CSCE313



```
root@pop-os:/volumes# ./spoof & ./sniff
[1] 120
Sending packet...
192.168.0.1 --> 10.9.0.5
Sniffing...

Received packet
        From:10.9.0.5
        To:10.9.0.1
   Payload (44 bytes):
CSCE313{C0ngRaTuLATions_Y0u_REacHED_ThE_END}
```

server running correctly



```
Sending packet...
10.9.0.5 --> 10.9.0.1
Attempt: 5/10 ...
Sending packet...
10.9.0.5 --> 10.9.0.1
Attempt: 6/10 ...
```

4. A working spoofer should be able to sniff any packet, and you can test using ping.

5. Look at the given PCAP lectures for additional hints

Getting started

1. Go to the assignment's GitHub classroom: https://classroom.github.com/a/Elya2CmS
2. Create a repository for your project.
3. Once completed and committed, you must submit the assignment on Gradescope:
   a. The sniffer program sniff.cpp
   b. The spoofer program spoof.cpp
   c. common.h for the send_raw_ip_packet
   d. A screenshot of your output showing the captured flag

How to run PA5 tutorial: https://www.youtube.com/watch?v=hrjlRgPmrqo

Rubric Instructions:

Code Requirements [100 points]:

- [30 points] A screenshot of the flag being captured
- [35 points] Write a spoofer program
  a. If no correct flag is captured but attempted, you get no point for the first 30 points
- [35 points] Write a sniffer program
  a. If no correct flag is captured but attempted, you get no points for the first 30 points