Welcome again, today I’ll be talking about **NETWORK SNIFFING ATTACK.** First before we begin, Let’s know the meaning of **NETWORK SNIFFING AND SPOOFING.**

**NETWORK SNIFFING & SPOOFING. =>** Is a network traffic for information (e.g. where it’s coming from , which device, the protocol use.)

………………………………………………………………………………………………………………………………………………………………….

**EXERCISE 2: ENVIRONMENT SETUP USING CONTAINER**

**2.1 CONTAINER SETUP AND COMMANDS.**

To solve this, We’d have to download the folder from [SEED Project (seedsecuritylabs.org)](https://seedsecuritylabs.org/Labs_20.04/Networking/TCP_Attacks/)

Now that you’ve installed your package, Go on and opened it. You should see

Graphical user interface, text, application

Description automatically generated

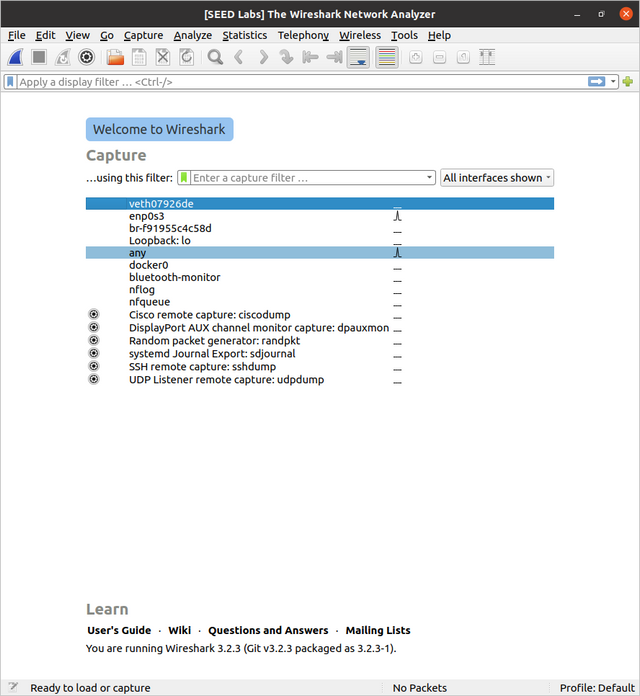
something like this ^^^^^^^^^

Looks like you’re there. WAIT!!! Did your forget to check if you have “WIRESHARK” installed in your machine?

To check this, just type the following command

|  |
| --- |
| [11/24/22][seed@VM:-/.../Labsetup$](mailto:seed@VM:-/.../Labsetup$) wireshark |

Your response should be like this.



Good! This means your have it installed.

Let’s build our labs now.

………………………………………………………………………………………………………………………………………………………………….

**dcbuild**

|  |  |
| --- | --- |
| Build your project.  Using this command will help build your project and make your ready to solve the seed lab  [11/24/22][seed@VM:-/.../Labsetup$](mailto:seed@VM:-/.../Labsetup$) dcbuild  build |  |

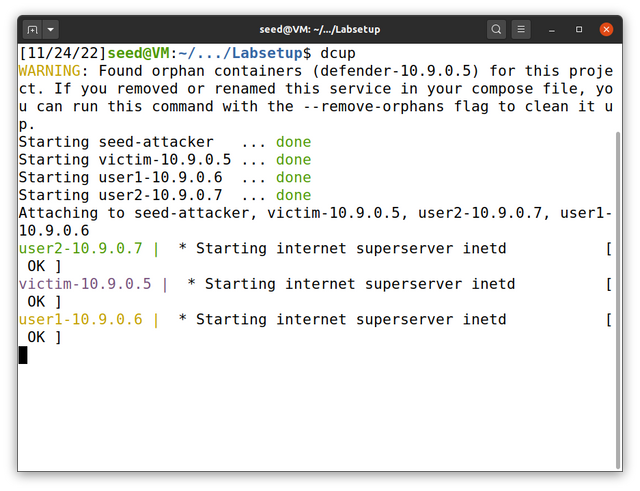
………………………………………………………………………………………………………………………………………………………………….

dcup

RUN your project.

Using this command will help build your project and make your ready to solve the seed lab

[11/24/22][seed@VM:-/.../Labsetup$](mailto:seed@VM:-/.../Labsetup$) dcup



The next command is to help your check for your project id. Example is being showed in the image below.

**dockps**

[11/24/22]seed@VM:~/.../Labsetup$ dockps

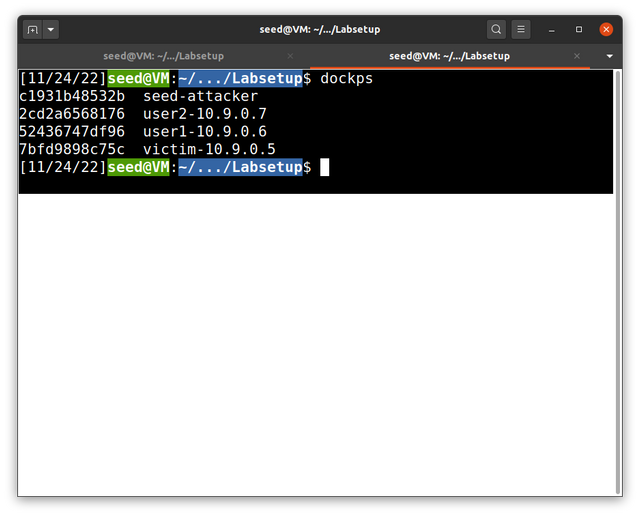
c1931b48532b seed-attacker

2cd2a6568176 user2-10.9.0.7

52436747df96 user1-10.9.0.6

7bfd9898c75c victim-10.9.0.5

[11/24/22]seed@VM:~/.../Labsetup$



………………………………………………………………………………………………………………………………………………………………….

To see you protocols type “ifconfig” This will help show you’re the flags.

Text

Description automatically generated

Text

Description automatically generated

Make sure to check your network id, name, driver, scope

Text

Description automatically generated

………………………………………………………………………………………………………………………………………………………………….

We’re done and ready. To start.

**Lab Task Set 1: Using Scapy to Sniff and Spoof Packets**

**SOLUTION:**

………………………………………………………………………………………………………………………………………………………………….

**Let’s check if scapy is installed to do this, Type into your terminal**

**SCAPY.**

Text

Description automatically generated

**Let’s check our IP address and other sources by using the following commands**

|  |
| --- |
| >> a = IP()  >> a.show()  result  ###[ IP ]###  version= 4  ihl= None  tos= 0x0  len= None  id= 1  flags=  frag= 0  ttl= 64  proto= hopopt  chksum= None  src= 127.0.0.1  dst= 127.0.0.1  \options\  >>> |

Text

Description automatically generated

………………………………………………………………………………………………………………………………………………………………….

**To save time, I’ll recommend you write all your code in a filename.py file so when ever you need something, you’ll just type python3 filename.py to run your code. They have the same result.**

**Example below**

Text

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**This task is completed…**

………………………………………………………………………………………………………………………………………………………………….

**Lab Task 1.1: Sniffing Packets**

**Solution.**

I solved this problem by writing the following code step by step

**Task 1.1 & 1.1A.**

**Task 1.1**

Code => {

#!/usr/bin/env python3

from scapy.all import \*

def print\_pkt(pkt):

pkt.show()

pkt = sniff(iface='br-f91955c4c58d', filter='icmp', prn=print\_pkt)

}

Text

Description automatically generated

Graphical user interface, application, PowerPoint

Description automatically generated

**Task 1.1A**

Graphical user interface, application

Description automatically generated

**Let’s then run this Task1.py file we’ve created and let’s run it in our terminal.**

**Example: chmod a+x task1.py Click on enter and then type python3 task1.py**

Text

Description automatically generated



**Oh!!! Error :( no worries you forgot to use sudo / root privilege**

**Example: sudo python3 task1.py Your response for this will be empty. Which means your execution worked.**

Graphical user interface, text

Description automatically generated

………………………………………………………………………………………………………………………………………………………………….

**TASK 1.1B**

………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………………………………….

# Solution

Graphical user interface, application

Description automatically generated# CAPTURE ONLY THE ICMP PACKET.

View On wiresharkText

Description automatically generated

View on ping.

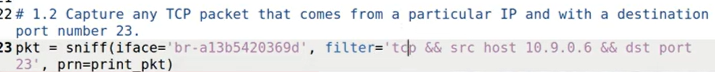
Graphical user interface, text

Description automatically generated



# CAPTURING THE TCP PACKET THAT COMES FROM A PARTICULAR IP AND WITH A DESTINATION PORT NUMBER 23. I wrote some code below

Graphical user interface, application

Description automatically generated

**TASK 1.2**: **Spoofing ICMP Packets**

SOLUTION…

………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………………………………….

This task to solve, You can even use scapy on your terminal or write your own python code so you don’t have to rewrite everything again after you’ve ran your code.

………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………………………………….

Example code needed.



from scapy.all import \*

a = IP()

a.dst = '1.2.3.4'

b = ICMP()

p = a/b

ls(a)

send(p, iface='br-f91955c4c58d')

Graphical user interface, text, application

Description automatically generated

Now that we’ve written our code, Save and head on to your terminal and type this command. root@VM:/volumes# python3 start.py

Simple response for this ^ Program.



Graphical user interface

Description automatically generated with medium confidence

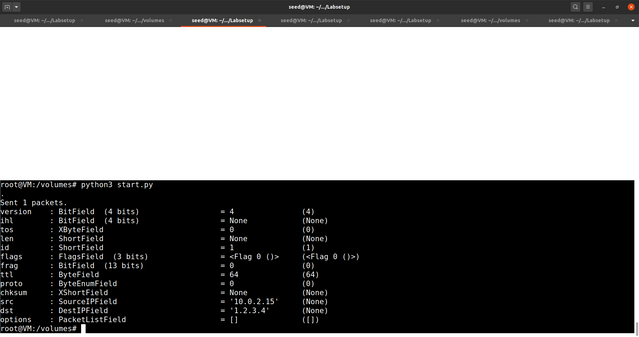
# Viewing result on wireshark

Text

Description automatically generated



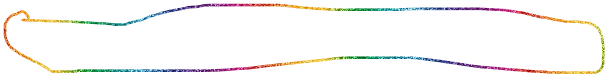
>> ls(a)



**Task 1.3: Traceroute**

**Solution.**

Let’s write some python code. This code however sends a ICMP Time-to-live and echos ping also. Your response after sending this should look like this.

Graphical user interface, text

Description automatically generated

………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………

SOURCE CODE.

Graphical user interface, text, application

Description automatically generated

Needed Result.

Graphical user interface, text

Description automatically generated

**TASK 1.4 SNIFFING AND THEN SPOOFING..**

Here, We’re gonna write some code to send, check for the destination, spoofed and source Ip.

With this type of attack, You can send multiple requests and sniffings / spoofing to a server. For this one, We’re gonna be using **1.2.3.4** server Below is the code for this work…

Read the code and get a better understanding.

#!/usr/bin/env python3

from scapy.all import \*

def spoof\_pkt(pkt):

# sniff and print out icmp echo request packet

if ICMP in pkt and pkt[ICMP].type == 8:

print("Original.......")

print("Source IP : ", pkt[IP].src)

print("Destination IP :", pkt[IP].dst)

# spoof an icmp echo reply packet

# swap srcip and dstip

ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)

icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)

data = pkt[Raw].load

newpkt = ip/icmp/data

print("Spoofed Packet....")

print("Source IP : ", newpkt[IP].src)

print("Destination IP :", newpkt[IP].dst)

send(newpkt, verbose=0)

filter = 'icmp and host 1.2.3.4'

#print("filter: {}\n".format(filter))

pkt = sniff(filter=filter, prn=spoof\_pkt)

Graphical user interface, text

Description automatically generated

………………………………………………………………………………………………………………………………………………………………….

This code Print’s out the sniffed, spoffed, and destination.

The above work contains cource code.

First we check if the ICMP in pkt and pkt array of ICMP with the type equal to 0

Then we print the sniffing responses and echo out the response.

Result:

Graphical user interface, text, application

Description automatically generated

Result: from wireshark.

Graphical user interface, text

Description automatically generated

……………………………………………………………………………………………………………………………………………………

To solve this work. I had to learn and understand ART Which was kinda difficult a little but I got a simple understanding not all but simple. Hope my work is well held. Thanks For Reading

……………………………………………………………………………………………………………………………………………………

THE END…