Challenge new hiring - Mercado Libre

Desarrollar una aplicación para análisis de tráfico con la responsabilidad de capturar paquetes desde

una interfaz de red y mostrar estadísticas básicas sobre el mismo.

para realizar dicho trabajo tenemos varios procesos como instalaciones de aplicativos, ejecuciones y pantallazos del mismo; a continuación envió el paso a paso de cómo se realizó este aplicativo de análisis de tráfico.

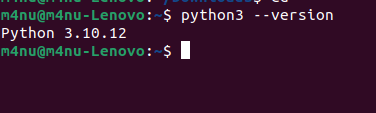
1- Sistema Operativo es linux (distribución Ubuntu 22.04.4 LTS) y debemos instalar 2 aplicativos para su gestión.

- python 3.10.12

- Scapy 2.4.4

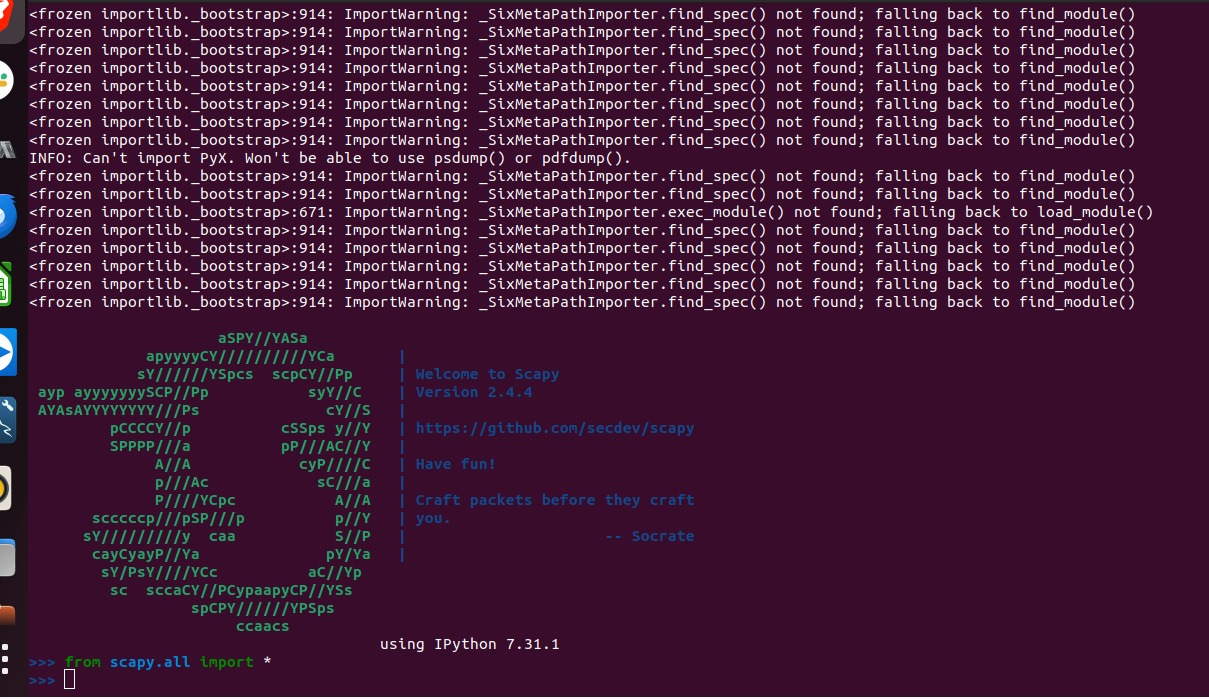
**para instalar la versión de python 3.10.12 se realiza los siguientes pasos:**

1. abrimos terminal y escribimos los siguientes comandos
2. sudo apt update
3. sudo apt install python3.10.12
4. python3 --version
5. wget https://www.python.org/ftp/python/3.11/Python-3.11.0.tar.gz
6. tar -xf Python-3.11.0.tar.gz
7. cd Python-3.11.0
8. ./configure --prefix=/usr/local
9. make install
10. python3 --version
11. sudo apt install python3-pip
12. python3 --version



**para instalar la versión de scapy 2.4.4 se realiza los siguientes pasos:**

1. abrimos terminal y escribimos los siguientes comandos
2. sudo apt install python3-pip
3. pip3 install scapy
4. sudo apt install git
5. git clone <https://github.com/secdev/scapy.git>
6. cd scapy
7. sudo python3 setup.py install
8. sudo scapy



2- Ya instalado los dos aplicativos, vamos a generar el script en Python3 con el siguiente código

(adjuntamos script para su ejecución)

#Librerías importadas para utilizar en el código

import argparse

from datetime import datetime

from scapy.all import sniff, IP, TCP, UDP

#Iniciación de variables

tcp\_count = 0

udp\_count = 0

source\_count = {} #Diccionario key:value

destination\_count = {}

#Función de análisis de paquetes

def packet\_handler(packet, output\_file):

global tcp\_count, udp\_count

#Se extrae día, mes, hora y minuto actual para el nombre del archivo

timestamp = datetime.now().strftime("%d\_%m\_%H\_%M")

#Loop para revisar todo el tráfico y llenar el archivo base de datos

with open(output\_file, "a") as file:

if IP in packet:

src\_ip = packet[IP].src

dest\_ip = packet[IP].dst

file.write(f"{timestamp} - Source IP: {src\_ip}, Destination IP: {dest\_ip}\n")

#Comparamos las IP de origen y destino para contar cada que se repitan (top 5)

source\_count[src\_ip] = source\_count.get(src\_ip, 0) + 1

destination\_count[dest\_ip] = destination\_count.get(dest\_ip, 0) + 1

#Contamos los paquetes TCP y UDP

if TCP in packet:

tcp\_count += 1

elif UDP in packet:

udp\_count += 1

protocol = "Unknown Protocol"

#Tamaño del paquete

packet\_size = len(packet)

if IP in packet:

src\_ip = packet[IP].src

dst\_ip = packet[IP].dst

if TCP in packet:

protocol = "TCP"

elif UDP in packet:

protocol = "UDP"

#Impresión de pantalla del tráfico capturado

print(f"Source IP: {src\_ip}, Destination IP: {dst\_ip}, Protocol: {protocol}, Packet Size: {packet\_size}")

#Función para imprimir el top 5 de IP origen y destino con mayor tráfico

def print\_top\_5(count\_dict, title):

# tomando los 5 valores mayores

sorted\_dict = dict(sorted(count\_dict.items(), key=lambda x: x[1], reverse=True)[:5])

print(f"\nTop 5 {title} IP addresses:")

for ip, count in sorted\_dict.items():

print(f"{ip}: {count} packets")

#Función para capturar el tráfico

def capture\_packets(interface, timeout):

filename = f"traffic\_capture\_{datetime.now().strftime('%d\_%m\_%H\_%M')}.txt"

sniff(iface=interface, prn=lambda pkt: packet\_handler(pkt, filename), timeout=timeout)

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser()

parser.add\_argument("-time", type=int, help="Total time value in seconds for packet capture")

parser.add\_argument("-interface", help="Network interface to capture packets from")

args = parser.parse\_args()

if args.time and args.interface:

capture\_packets(args.interface, args.time)

print(f"Total TCP packets captured: {tcp\_count}")

print(f"Total UDP packets captured: {udp\_count}")

print(f"Total packets captured: {tcp\_count + udp\_count}")

print\_top\_5(source\_count, "Source")

print\_top\_5(destination\_count, "Destination")

# imprime los datos que están en la aplicación en Scapy

else:

print("Please specify both arguments -time and -interface values.")

**Lista de palabras claves.**

Argparse — Analizador sintáctico (Parser) para las opciones, argumentos y sub-comandos de la línea de comandos

Sniffer - También denominado rastreador de red, es un software o hardware que se utiliza para monitorizar, capturar y analizar en tiempo real los paquetes de datos que pasan por una red.

TCP - El Protocolo de Control de Transmisión (Transmission Control Protocol en inglés o TCP) es el método de comunicación de datos por defecto entre distintos dispositivos, a través de una red

(subida - descarga)

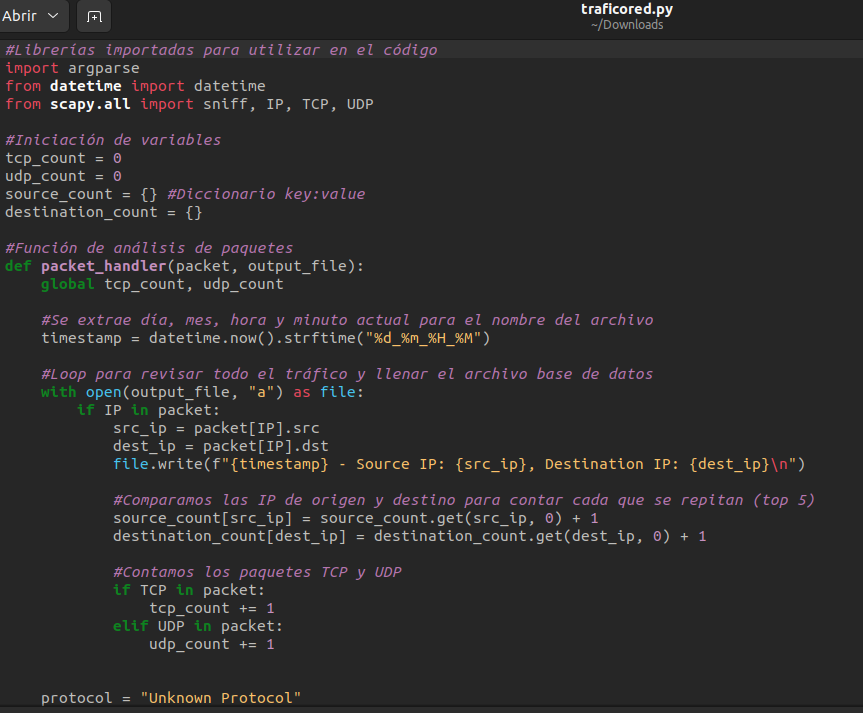
UDP - es un protocolo ligero de transporte de datos que funciona sobre protocolo de internet (IP). Dicho de otra manera, permite el envío rápido de datagramas en redes IP, sin necesidad de establecer una conexión previa. (streaming)

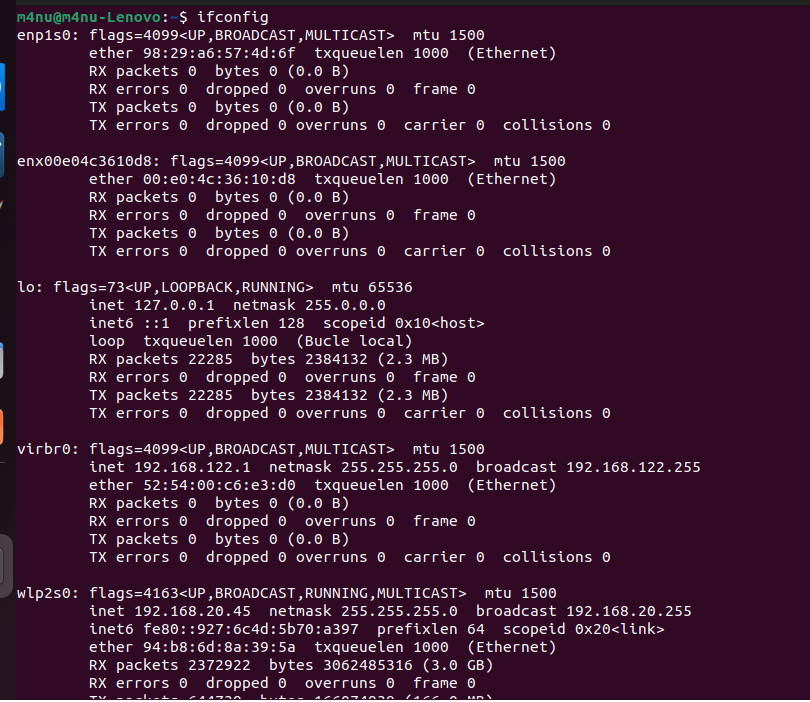
Packet\_handler - función que se ejecuta en el servidor web cuando se hace una petición a cierta url (endpoint) de dicho servidor web.

Lambda - forma corta de declarar funciones pequeñas y anónimas (no es necesario proporcionar un nombre para las funciones lambda)

Parser - proporciona una interfaz para el analizador sintáctico interno de Python y para el compilador de código de bytes

3- Se realizan las capturas desde python local con su código



4- para ejecutar el script en terminal con Scapy, verificamos la interfaz de red con la que deseamos mirar las salidas y entradas del direccionamiento, para ello revisamos desde terminal con el comando ifconfig en ubuntu. El cual implementamos la interfaz “wlp2s0” 

si desean ejecutar el script con interfaz desde otro sistema operativo :

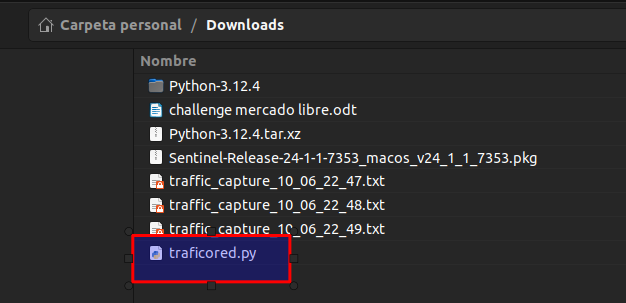
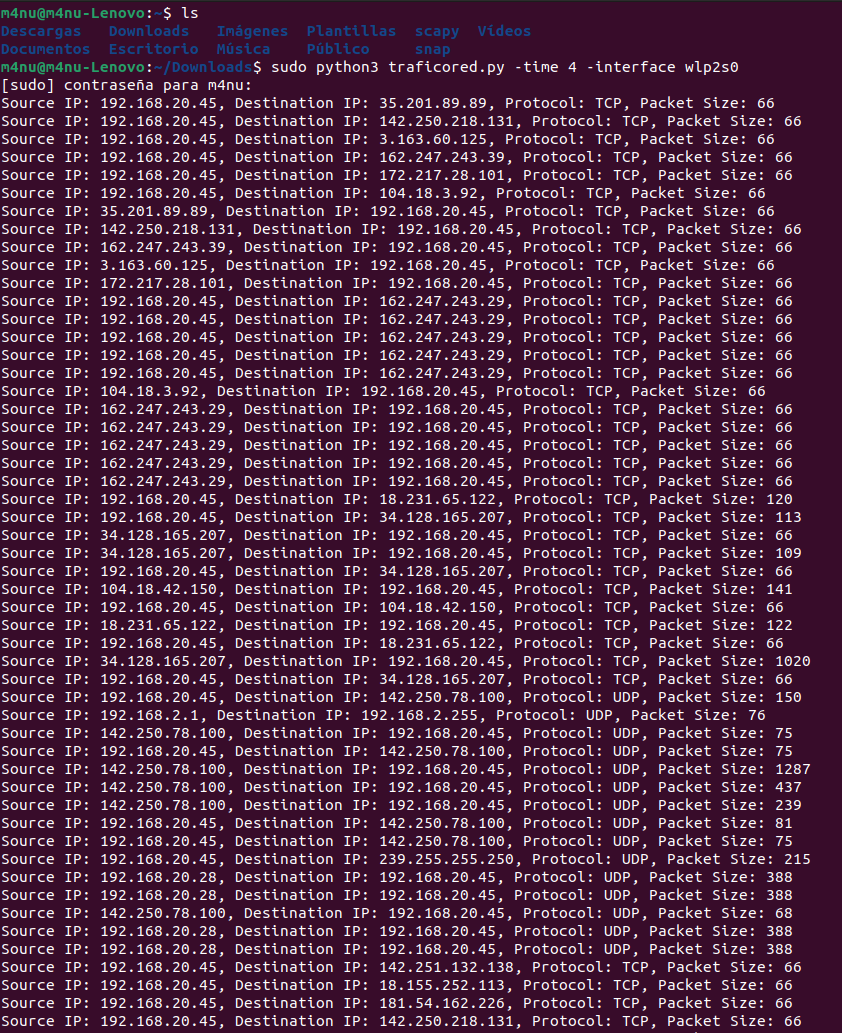
***macOS de MacBook es con el comando ifconfig***

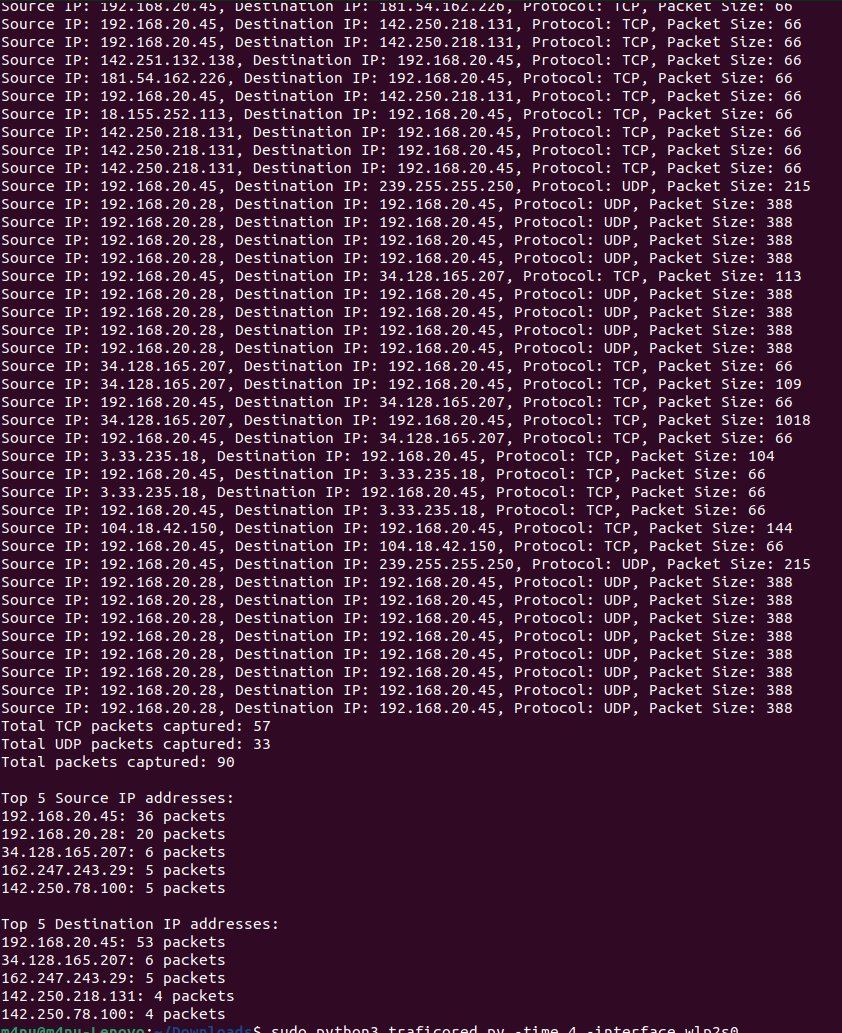
***Windows de Microsoft es con el comando ipconfig***

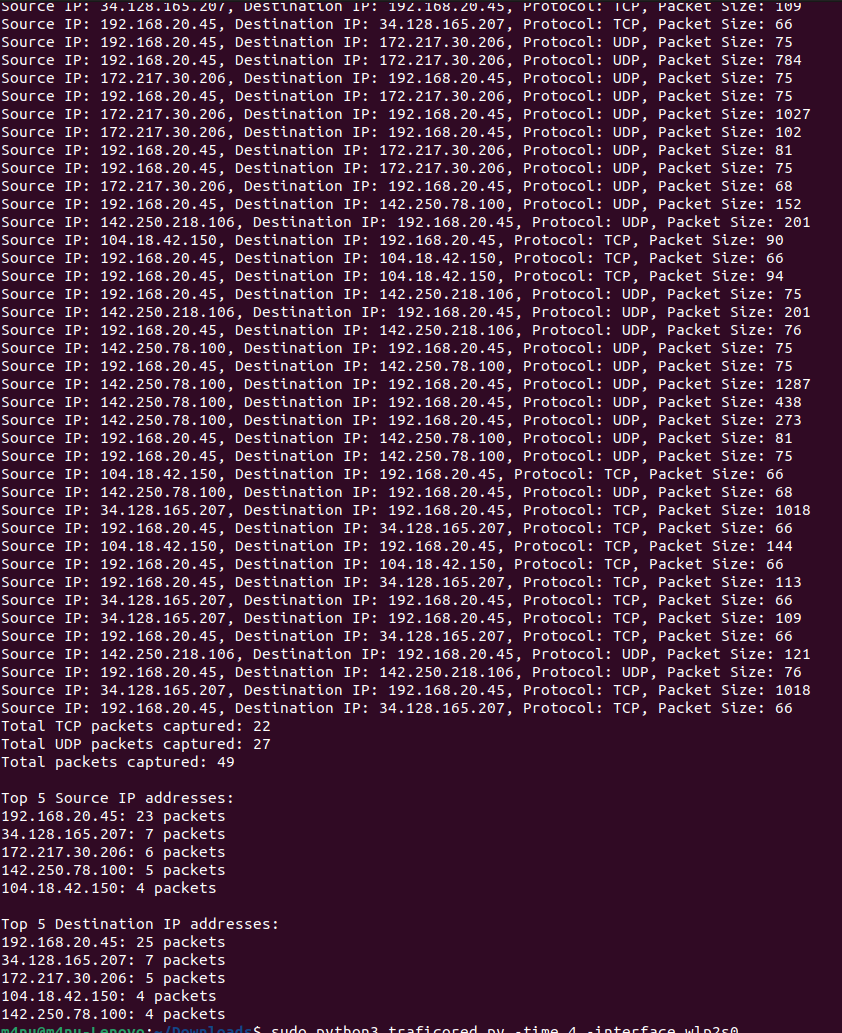
5- para ejecutar el script y se visualice el análisis y proceso de tráfico hacemos lo siguiente.

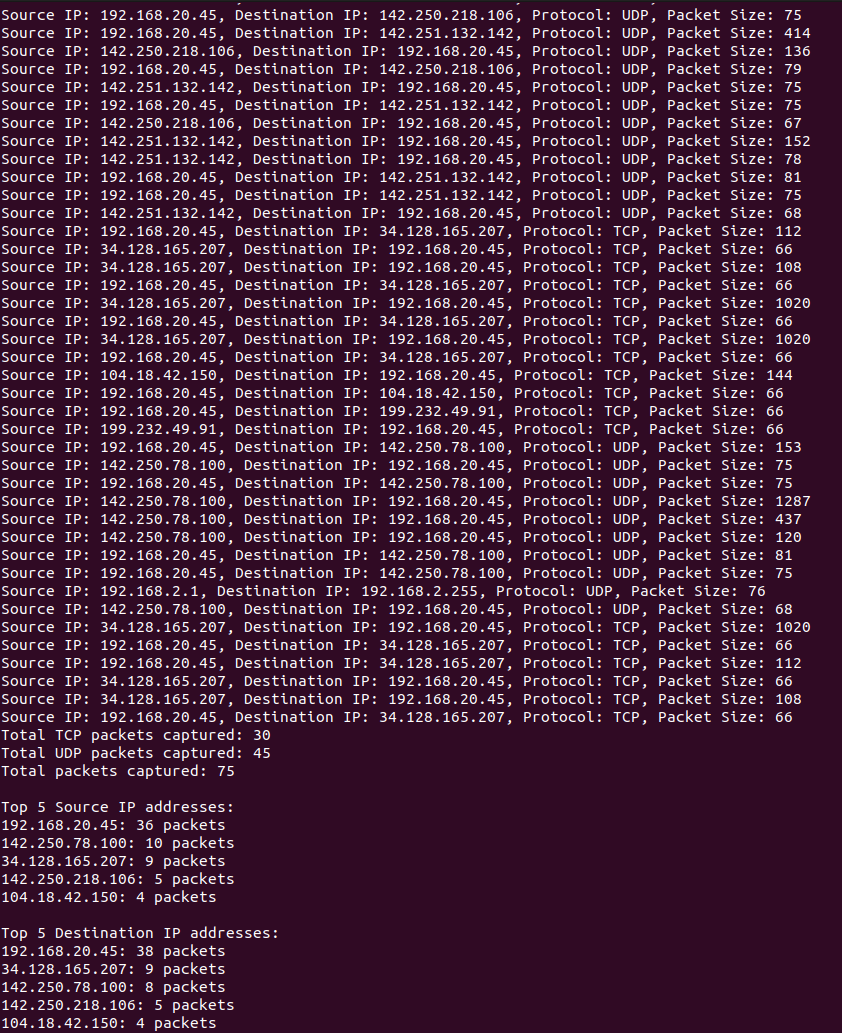
* guardamos el script en una ruta local, puede ser en *descargas*
* vamos a la terminal y dentro de python3 ejecutamos la siguiente linea   
  sudo python3 traficored.py -time 4 -interface wlp2s0 dentro del código sería así *print("Please specify both arguments -time and -interface values.")}*
* luego nos muestra todo el tráfico Origen y Destino con IPs y los protocolos TCP y UDP
* tambíen nos visualiza total de paquetes capturados con los dos protocolos TCP y UDP
* al finalizar el monitoreo nos enseña las 5 IPs de origen y las 5 IPs de destino

adjunto pantallazos del proceso





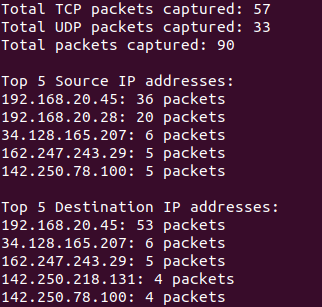


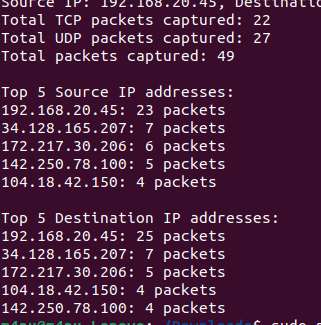
6- Número de paquetes por protocolo (por ejemplo, TCP, UDP).

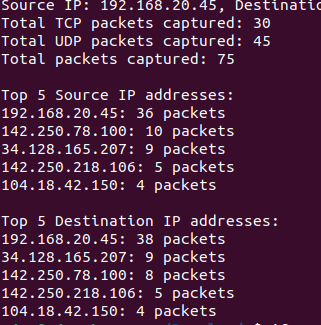
Las 5 principales direcciones IP de origen con mayor tráfico.

Las 5 principales direcciones IP de destino con mayor tráfico.

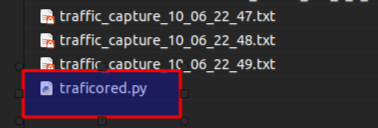
nos enseña que la IP 192.168.20.45 es de mi red local







7- Se crean 3 archivos en formato txt con una base de datos que muestra los paquetes de origen y destino. (se adjunta archivos en el repositorio)



7- Se crean 3 archivos en formato txt con una base de datos que muestra los paquetes de origen y destino. (se adjunta archivos en el repositorio)