

COMPUTER NETWORKS MINI PROJECT

**Project Title: Design a tool to identify the type of devices used
in communication (Mobile/Desktop)**

Submitted by

JAHNAVI KOLAKALURI (190905093)

RADHIKA TANEJA (190905344)

In partial fulfillment for the award of degree of

B.TECH

IN

COMPUTER SCIENCE ENGINEERING



**MANIPAL INSTITUTE
OF TECHNOLOGY**

MANIPAL

A Constituent Institution of Manipal University

Department of Computer Science & Engineering

NOVEMBER 2021

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

Certified that this project report is the bonafide work of “JAHNAVI KOLAKALURI (190905093) & RADHIKA TANEJA (190905344)” who carried out the mini project work under my supervision.

**Dr Ashalatha Nayak
Professor and HOD**

**Ms Roopalakshmi R
Assistant Professor**

Submitted to the Viva Voce Examination held on

_____.

EXAMINER 1

EXAMINER 2

ABSTRACT

Our work involves developing a tool in python for the windows operating system that can be used to identify the types of devices used in communication such as mobile and desktop.

In essence, this tool has been built by us by capturing raw traffic files from Wireshark; then scrapping it—extracting information such as user agent and MAC addresses of the communicating devices from the pcap files. This information was used to identify the device type and device vendors.

ACKNOWLEDGEMENT

We express our deep sense of respect and our gratitude to our supervisor Ms Roopalakshmi R, who created a healthy learning environment, enlightened us with great ideas, and patiently guided us. It was really a great experience for us to work with her, and we would not be able to finish our work without her guidance, support, and direction. We are indebted to her for her time and patience.

We extend our thanks to other faculty members and non-teaching staff of Manipal Institute of Technology for providing needed support for this project.

Place: Manipal

Date: 30/11/2021

TABLE OF CONTENTS

1. INTRODUCTION	
1.1 Background Information	5
1.2 Methodology	7
1.3 Learning Outcomes	10
2. IMPLEMENTATION	11
3. FURTHER PROSPECTS	18
4. REFERENCES	20

INTRODUCTION

BACKGROUND INFORMATION

- Regardless of whether one is working in a wired or a wireless environment, the commonality remains that both network hardware and software are put to use to transfer data from one device to another. Both of these entities have addresses attached to them—the network’s software uses an IP address, whereas the hardware relevant to the network is a key connection device in a computer called the **Network Interface Card**, or **NIC**. The NIC is essentially a computer circuit card that makes it possible for the computer to connect to a network. A NIC turns data into an electrical signal that can be transmitted over the network.
- Every NIC has a hardware address that’s known as a **MAC**, for **Media Access Control**. Where IP addresses are associated with TCP/IP (networking software), MAC addresses are linked to the hardware of network adapters.
- A MAC address is given to a network adapter when it is manufactured. It is hardwired onto your computer’s NIC and is unique to it. Something called the **ARP (Address Resolution Protocol)** translates an IP address into a MAC address. The ARP is like a passport that takes data from an IP address through an actual piece of computer hardware. MAC addresses and IP addresses go hand-in-hand here. All devices on the same network subnet have different MAC addresses. MAC addresses are very useful in diagnosing network issues, such as problems with IP addresses. MAC addresses are useful for network diagnosis because they never change, as opposed to a dynamic IP address, which can change from time to time. For a network administrator, that makes a MAC address a more reliable way to identify senders and receivers of data on the network.

- The **HTTP headers User-Agent** is a request header that allows a characteristic string that allows network protocol peers to identify the Operating System and Browser of the web-server. Your browser sends the user agent to every website you connect to. When your browser is connected to a website, a User-Agent field is included in the HTTP header. The data of the header field varies from browser to browser. This information is used to serve different websites to different web browsers and different operating systems.
- **Packet Capture or PCAP** (also known as libpcap) is an application programming interface (API) that captures live network packet data from **OSI model** Layers 2-7. Network analyzers like Wireshark create .pcap files to collect and record packet data from a network. Wireshark uses .pcap files to record packet data that has been pulled from a network scan. Packet data is recorded in files with the .pcap file extension and can be used to find performance problems and cyberattacks on the network. In other words, the PCAP file creates a record of network data that you can view through Wireshark. You can then assess the status of the network and identify if there are any service issues that you need to respond to.

METHODOLOGY

Programming language used: Python 3

Operating system: Windows PC

Libraries used: os, sys, pyshark

API used: <https://macvendors.co/>

Website to capture HTTP requests: <http://webscantest.com/csrf/csrfpost.php>

Tool used to capture pcap files for desktop: Wireshark

Tool used to capture pcap files on android phone: PCAPdroid

Using the background information and relevant research, we have come up with the following methodology to tackle the problem statement.

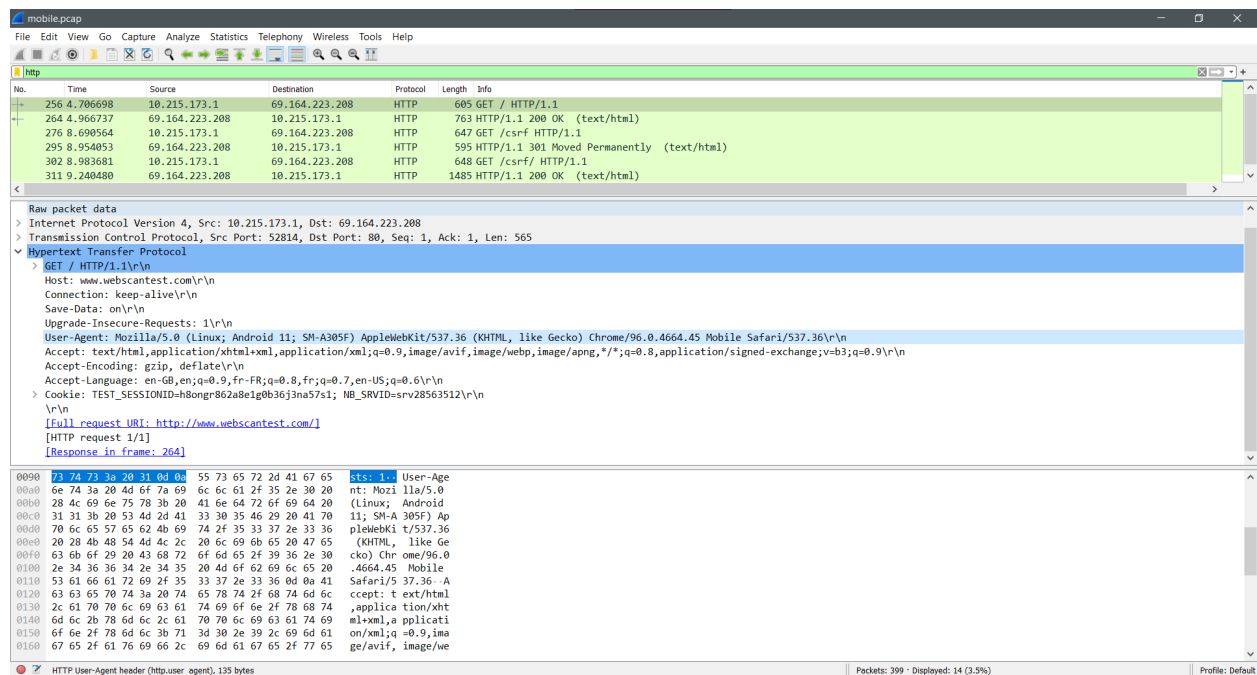
We implement two codes, one which parses user agent string in HTTP header to obtain the device type (called device.py). The other extracts the MAC address from the ARP and maps it to the device vendor (called vendor.py). The aggregate output of both the codes not only gives us the device type but also shows the operation system, browser and device vendor information.

Device.py implementation:

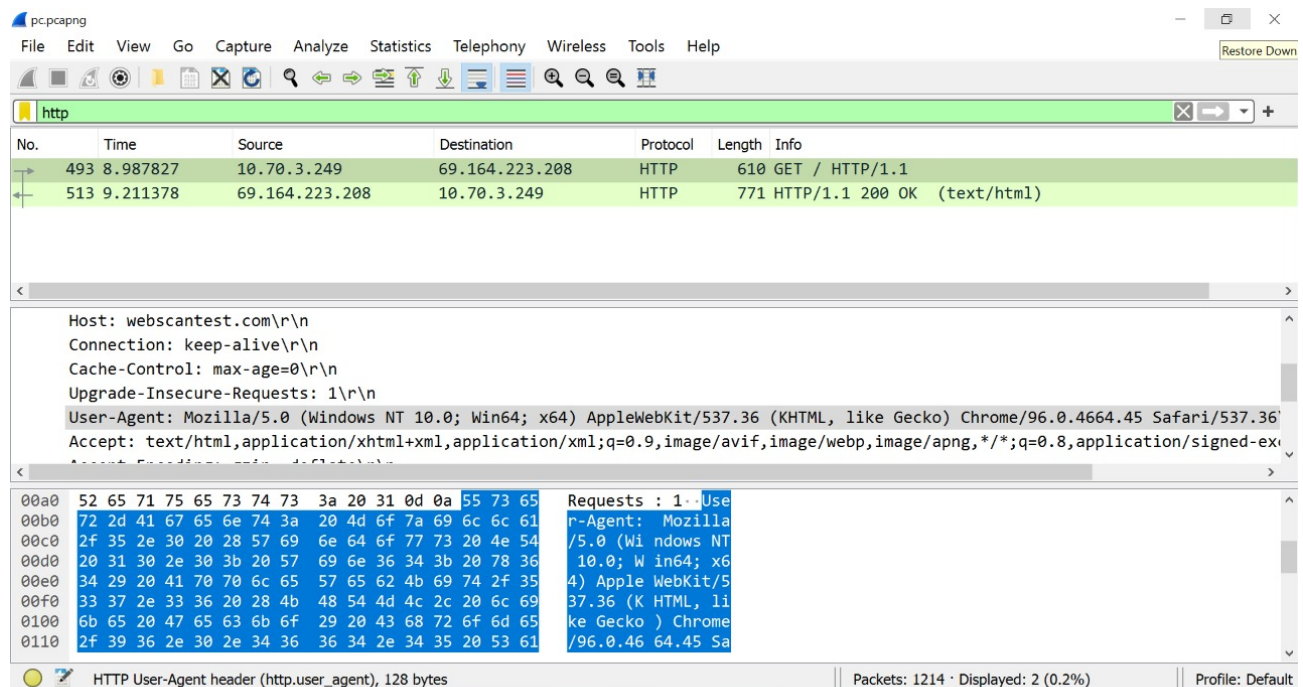
We start off by capturing packets for both laptop and mobile devices. The pcap files for the laptop were taken from Wireshark. An application called PCAPdroid was installed on an android mobile phone to capture pcap files from the mobile device.

From this pcap file, we use the HTTP header information to extract the user agent string.

Mobile.pcap file:



From this mobile.pcap file we can view the HTTP GET request. Under which we can see a user agent which is a characteristic string that allows network protocol peers to identify the Operating System and Browser of the web server. The same can be viewed in the pc.pcap file as shown below.

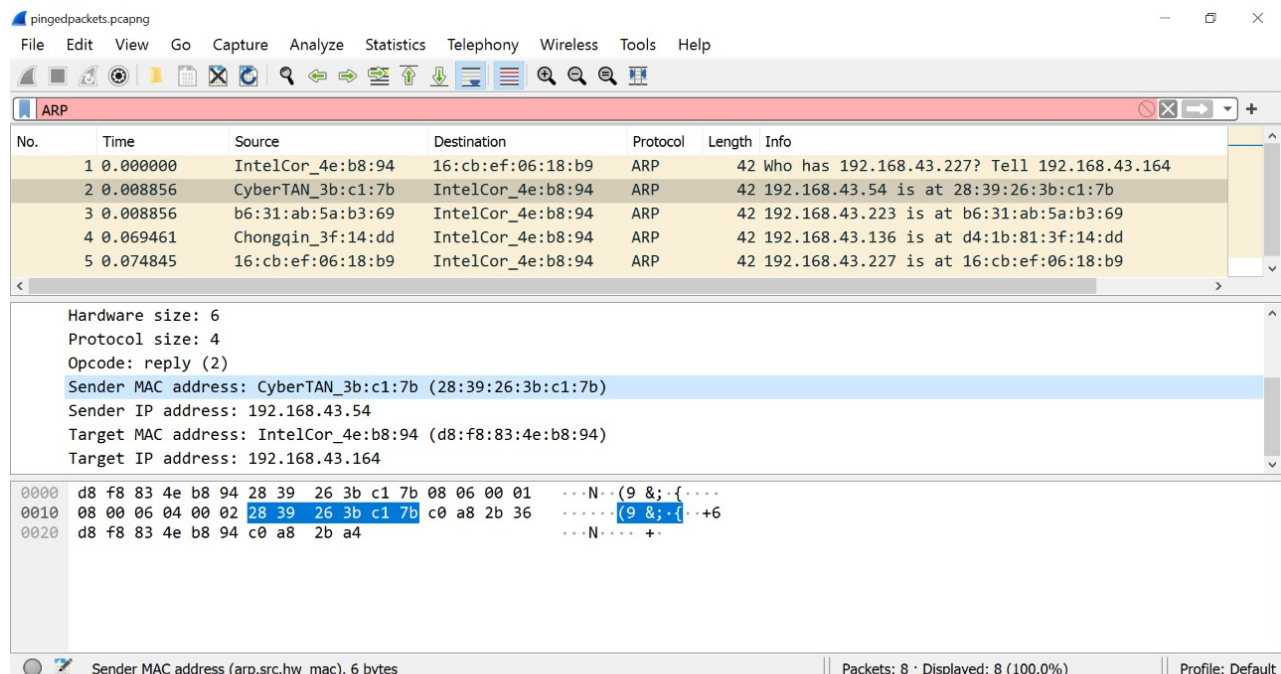


Finally, we use the `user_agents` python library. This library provides a way to identify/detect devices like mobile phones, tablets and their capabilities by parsing (browser/HTTP) user-agent strings.

Vendors.py implementation:

In order to obtain the required pcap file for this program, we connected various devices, including mobiles and laptops from various vendors to the same network via hotspot. All the devices pinged to our devices via the IP address of our system. These pings from the devices were captured in Wireshark into a pcap file.

For this program, we analyse another aspect of the raw pcap files - the ARP packets. The ARP packets were parsed to extract the sender and target MAC address. These addresses were then given to the `macvendors` API which mapped it to the corresponding device vendors and displays it on the terminal.



LEARNING OUTCOMES

1. In-depth knowledge of user agent string provided in the HTTP header in the pcap files.
2. Execution of python libraries in the context of packet analysis and scraping pcap files.
3. Appropriate usage of APIs and applications for acquiring relevant information.

IMPLEMENTATION

#device.py

```
#libraries to run files and operating systems
import os
import sys
#library used to capture packets from pcap
import pyshark
#library used to get device type from the user agent
from user_agents import parse
#library for building GUI
from tkinter import *
from tkinter import filedialog
import tkinter.font as tkFont

#making the GUI window and setting font family and size
root = Tk()
fontStyle = tkFont.Font(family="Lucida Grande", size=15)
# Please paste this in the terminal: files/pc.pcapng

root.geometry("1500x500")

#function to determine if device is mobile
def isMobileDevice(useragent):
    #parse user agent string and store in user_agent
    user_agent = parse(useragent)
    #using inbuilt .is_mobile method
    if user_agent.is_mobile:
        return True
    else:
        return False
```

```

#function to determine if device is table
def isTabletDevice(useragent):
    user_agent = parse(useragent)
    if user_agent.is_tablet:
        return True
    else:
        return False

#function to determine if device is table
def isPC(useragent):
    user_agent = parse(useragent)
    if user_agent.is_pc:
        return True
    else:
        return False

#function to extract user agent from pcap file
def getUserAgent():
    #taking pcap input from GUI interface
    root.filename = filedialog.askopenfilename(
        initialdir="/", title="Select file")

    if root.filename == '':
        print("No file selected")
        sys.exit()
    else:
        #array to store user agents
        useragents = []
        #using pyshark to extract HTTP GET request
        cap = pyshark.FileCapture(
            root.filename, display_filter='frame contains
"GET"')
        #parsing packets
        for packet in cap:
            print(packet['http'].user_agent)
            #parsing useragent string in HTTP header
            useragents.append(packet['http'].user_agent)

```

```

i = 0
for useragent in useragents:
    i = i+1
    #passing useragent through isMobileDevice,
isTabletDevice and isPC
    if isMobileDevice(useragent):
        myLabel = Label(root, text="Mobile
Device", font=fontStyle)
        myLabel.pack()
    elif isTabletDevice(useragent):
        myLabel = Label(root, text="Tablet
Device", font=fontStyle)
        myLabel.pack()
    elif isPC(useragent):
        myLabel = Label(root, text="PC Device",
font=fontStyle)
        myLabel.pack()
    else:
        myLabel = Label(root, text="Unknown
Device", font=fontStyle)
        myLabel.pack()
    #printing packet number, device type and user
agent string
    myLabel = Label(root, text="Packet"+str(i) +
        ": " + useragent,
font=fontStyle)
    print("\n")
    myLabel.pack()

cap.close()

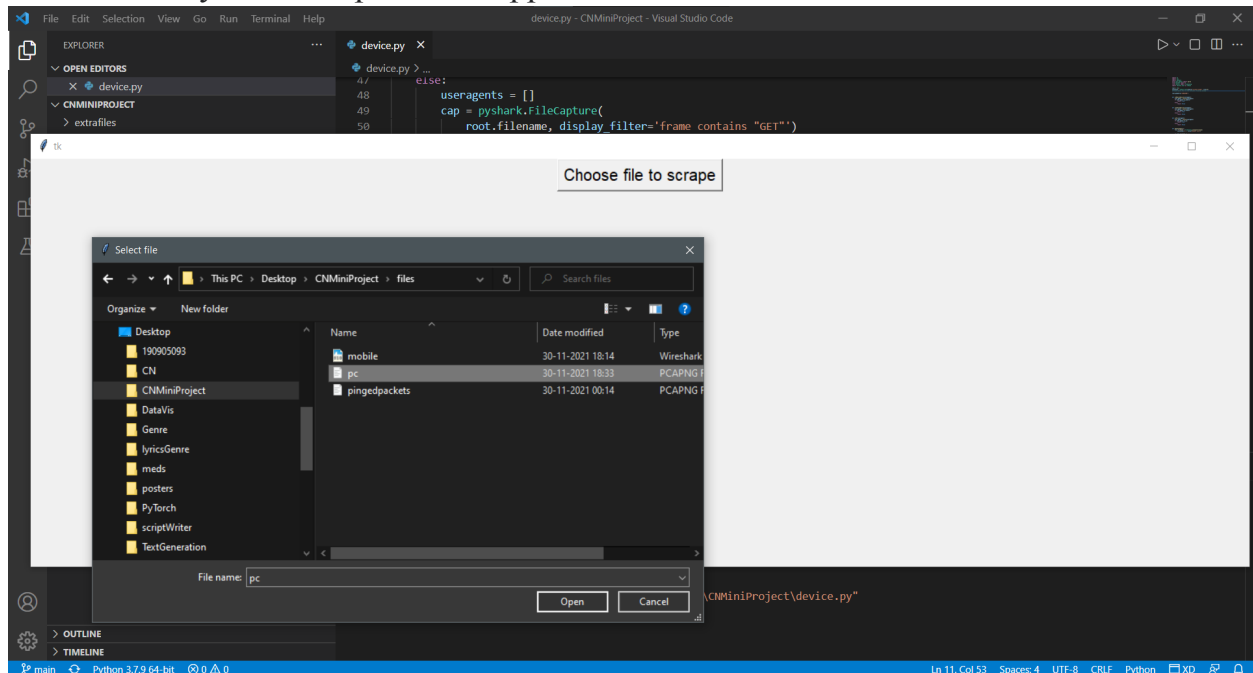
#adding button
fileInputBtn = Button(root, text="Choose file to scrape",
font=fontStyle,
                        command=getUserAgent)

fileInputBtn.pack()

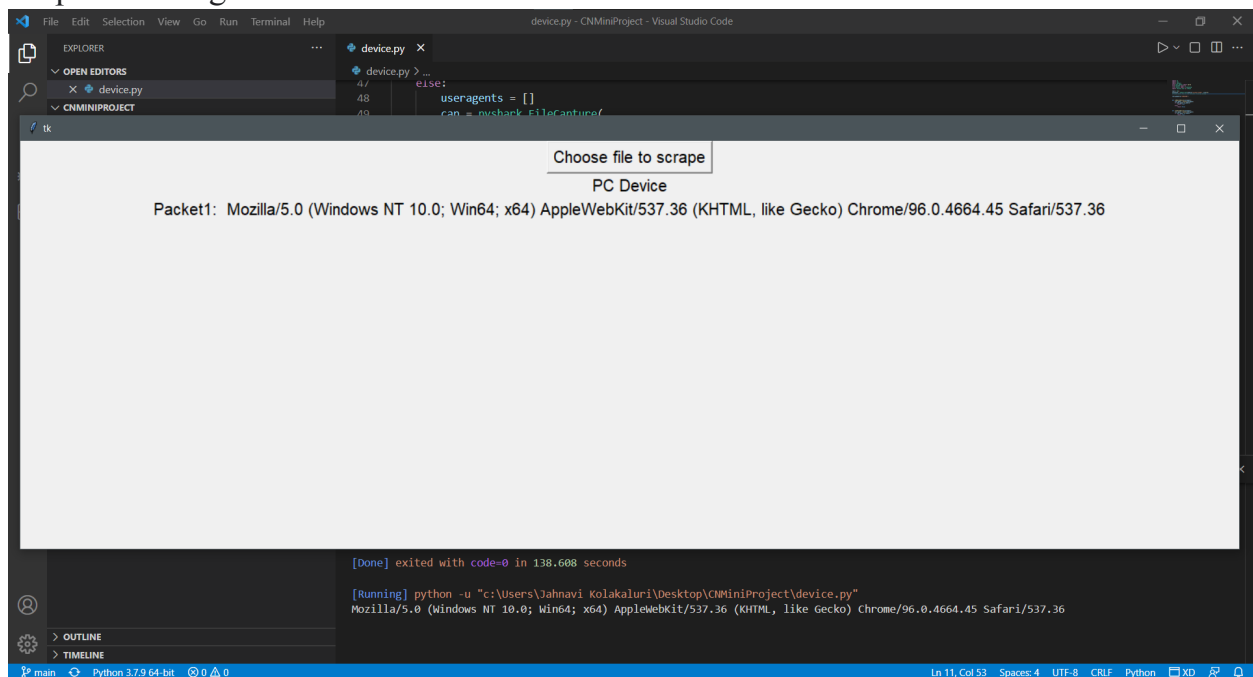
root.mainloop()

```

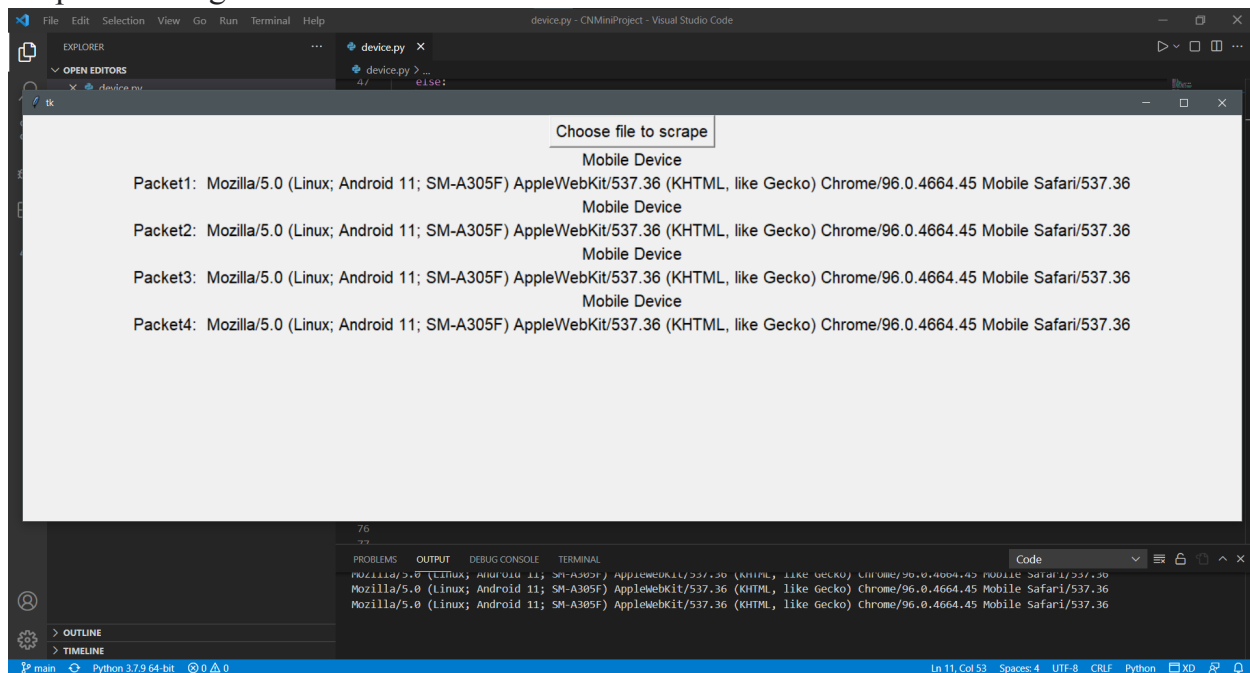
Running the **devices.py** program executes a GUI. Through the GUI we can select the file from the system as input to the application.



Output showing PC device as the client device.



Output showing Android mobile device as the client device.



#vendor.py

```
import os
import sys
import pyshark
import requests
import time
try:
    # Please paste this in the terminal:
    files/pingedpackets.pcapng
    filename = input("Enter path of file to read:
").strip()
    cap = pyshark.FileCapture(filename,
display_filter='arp')
    i = 0
    #inserting API
    url = "https://api.macvendors.com/"
    for packet in cap:
        i = i+1
        try:
            print("\n----- Packet ", i, "
-----")
```



```

        #parsing ARP packet and storing source and
destination MAC addresses
        sourceMac = packet['arp'].src_hw_mac
        destMac = packet['arp'].dst_hw_mac

        print("\nDevice 1")
        #calling API to get response
        response1 = requests.get(url+sourceMac)
        time.sleep(1)
        print("src mac: ", sourceMac)
        #decoding response to character string
        print(response1.content.decode())

        print("\nDevice 2")
        print("dest mac: ", destMac)
        response2 = requests.get(url+destMac)
        time.sleep(1)
        print(response1.content.decode())

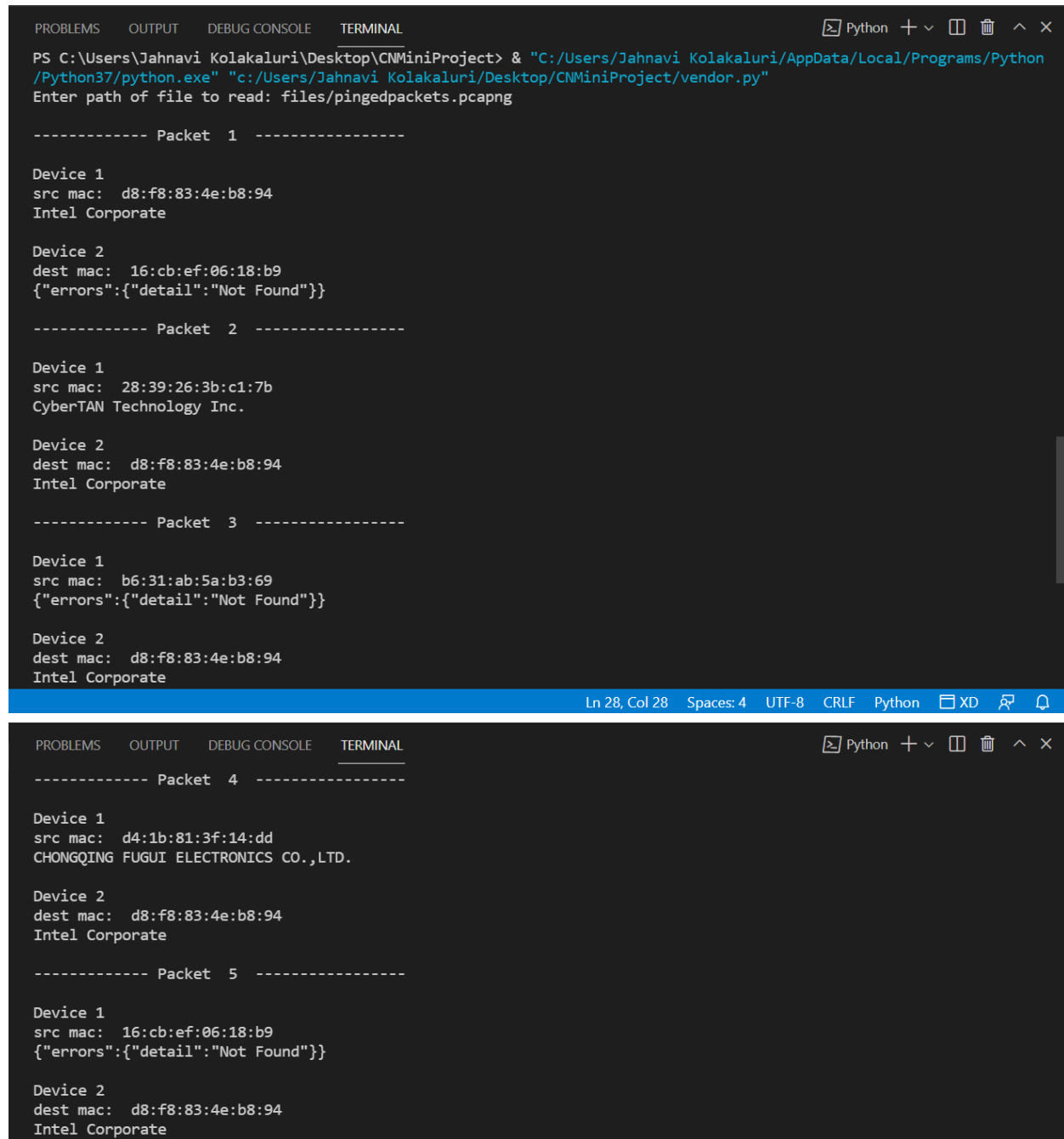
    except Exception as err:
        pass
        print('\nAn error has occurred: ')
        print(err)
        print()
cap.close()

except Exception as err:
    pass
    print('\nAn error has occurred: ')
    print(err)
    print()

```

All the devices were connected to the same network via hotspot. Various devices pinged to our PC's IP address. Therefore, the destination MAC address remains constant as d8:f8:83:4e:b8:94 and the vendor as obtained from the API is Intel Corporate.

The source MAC address along with the vendor is displayed. If the API is not able to find the vendor it prints an error in the output.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [ ] ^ x
PS C:\Users\Jahnavi Kolakaluri\Desktop\CNMiniProject> & "C:/Users/Jahnavi Kolakaluri/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/Jahnavi Kolakaluri/Desktop/CNMiniProject/vendor.py"
Enter path of file to read: files/pingedpackets.pcapng

----- Packet 1 -----

Device 1
src mac: d8:f8:83:4e:b8:94
Intel Corporate

Device 2
dest mac: 16:cb:ef:06:18:b9
{"errors":{"detail":"Not Found"}}

----- Packet 2 -----

Device 1
src mac: 28:39:26:3b:c1:7b
CyberTAN Technology Inc.

Device 2
dest mac: d8:f8:83:4e:b8:94
Intel Corporate

----- Packet 3 -----

Device 1
src mac: b6:31:ab:5a:b3:69
{"errors":{"detail":"Not Found"}}

Device 2
dest mac: d8:f8:83:4e:b8:94
Intel Corporate

Ln 28, Col 28 Spaces: 4 UTF-8 CRLF Python [ ] XD [ ] [ ]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [ ] ^ x

----- Packet 4 -----

Device 1
src mac: d4:1b:81:3f:14:dd
CHONGQING FUGUI ELECTRONICS CO.,LTD.

Device 2
dest mac: d8:f8:83:4e:b8:94
Intel Corporate

----- Packet 5 -----

Device 1
src mac: 16:cb:ef:06:18:b9
{"errors":{"detail":"Not Found"}}

Device 2
dest mac: d8:f8:83:4e:b8:94
Intel Corporate
```

FURTHER PROSPECTS

While the scope of this project of ours was limited to identification of communication devices that could be mapped using MAC addresses and IP addresses, it unravels a plethora of opportunities and applications for an enthusiast to be motivated from.

- For example, to our immediate attention, this project can be used as a basis to enable and secure content delivery networks. A **content delivery network (CDN)** is a distributed server network that delivers temporarily stored, or cached, copies of website content to users based on the user's geographic location. A CDN stores this content to protect against traffic surges, reduce latency, decrease bandwidth consumption, accelerate load times, and lessen the impact of hacks and attacks by introducing a layer between the end-user and your website infrastructure. The implementation of our code can further be expanded in scope to deliver a CDN prototype by identifying the kind of devices that are being used to stream certain services and helping the user secure their accounts.
- Computer network security protects the integrity of the information contained by a network and controls who access that information. Network security policies balance the need to provide service to users with the need to control access to information. There are many entry points to a network that includes the hardware and software that comprise the network itself as well as the devices used to access the network. Because of these entry points, network security requires using several defence methods. Similar to our justification above, a prototype can be delivered to identify the kind of users and devices trying to access a network. Further, several security measures can be employed to safeguard the network.
- One of the applications of this code could be to identify the kind of devices that are being used to stream a particular service and subsequently deliver the appropriate interface to run that particular software.

REFERENCES

- Jasraj. “HTTP Headers: User-Agent.” *GeeksforGeeks*, 11 Oct. 2019, <https://www.geeksforgeeks.org/http-headers-user-agent/>
- “User-Agents.” *PyPI*, <https://pypi.org/project/user-agents/>.
- Sieling, Gary. “How to Filter out a MAC Address in Wireshark.” *Gary Sieling*, 11 Mar. 2016, <https://www.garysieling.com/blog/filter-mac-address-wireshark/>.
- Hoffman, Chris. “How to Use Wireshark to Capture, Filter and Inspect Packets.” *How, How-To Geek*, 14 June 2017, <https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/>.
- “User-Agent - Http: MDN.” *HTTP | MDN*, <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent>.
- Oberheide, Jon. “Dpkt Tutorial #2: Parsing a PCAP File.” *Dpkt Tutorial #2: Parsing a PCAP File | Jon Oberheide*, <https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/>.
- IBM Cloud Education. “Networking-a-Complete-Guide.” *IBM*, <https://www.ibm.com/cloud/learn/networking-a-complete-guide>.