MAC ADDRESS ANALYSER

REPORT

A MINI PROJECT

REPORT

Submitted by

Gagan S Shenoy

Arun Adiga

In partial fulfilment for the award of the degree of

Bachelor of Technology

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 "MAC Address Analyser" is the bonafide work of "Arun Adiga and Gagan S Shenoy" who carried out the mini project work under my supervision.

_____ _____

(Signature of the HOD with date) (Signature of the Supervisor with date)

_____ _____

(Name of the HOD) (Name of the Supervisor)

Professor and Head

Submitted to the Viva voce Examination held on _____.

EXAMINER 1 EXAMINER 2

Abstract

Every Bluetooth/Ethernet device has a mac address associated with it, which can determine various pieces of information, such as the vendor that manufactured the device, the type of device, date range of manufacture, etc. Given a raw traffic file (such as a pcap file) of a particular network (gathered via software such as Wireshark), it is possible to analyze and gather various sorts of information of the devices in the network from the MAC addresses of the packets of the traffic file. One such information that we can collect is the vendor information of the network devices and the fake MAC addresses in the network. In our project, we attempt to build a tool that achieves the same with the help of a python script that detects real and fake MAC addresses in the network.

ACKNOWLEDGEMENT

1) Mrs. Roopalakshmi R [MAHE-MIT] for giving us an opportunity to do the project and constantly guiding us throughout the duration of the project.
2) "Practical packet analysis- Using Wireshark to solve real-world network problems" by Chris Sanders.
3) Wikipedia
4) IEEE website for the mac address vendor list.

Table of Contents

Introduction

What is a MAC Address?

A Media Access Control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used for numerous network technologies and most IEEE 802 network technologies, including Ethernet. Logically, MAC addresses are used in the Media Access Control protocol sub-layer of the OSI reference model.

MAC addresses are most often assigned by the manufacturer of a network interface card (NIC) and are stored in its hardware, the card's read-only memory, or some other firmware mechanism. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number and may be referred to as the burned-in address. It may also be known as an Ethernet hardware address (EHA), hardware address or physical address. A network node may have multiple NICs and will then have one unique MAC address per NIC.

MAC addresses are formed according to the rules of one of three numbering name spaces managed by the Institute of Electrical and Electronics Engineers (IEEE): MAC-48, EUI-48, and EUI-64. The IEEE claims trademarks on the names EUI-48 and EUI-64, in which EUI is an acronym for Extended Unique Identifier.

Notational Conventions

The standard (IEEE 802) format for printing MAC-48 addresses in human-friendly form is six groups of two hexadecimal digits, separated by hyphens (-) or colons (:), in transmission order (e.g., 01:23:45:67:89:ab). This form is also commonly used for EUI-64. Another convention used by networking equipment uses three groups of four hexadecimal digits separated by dots (.) (e.g., 0123.4567.89ab), again in transmission order.

Organizationally Unique Identifier(OUI)

An organizationally unique identifier (OUI) is a 24-bit number that uniquely identifies a vendor, manufacturer, or other organization.

OUIs are purchased from the Institute of Electrical and Electronics Engineers (IEEE) Registration Authority by the assignee (IEEE term for

the vendor, manufacturer, or other organization). Only assignment from MA-L registry assigns new OUI. They are used to uniquely identify a particular piece of equipment through derived identifiers such as MAC addresses, Subnetwork Access Protocol protocol identifiers, World Wide Names for Fibre Channel devices, or vendor blocks in EDID.

In MAC addresses, the OUI is combined with a 24-bit number (assigned by the assignee of the OUI) to form the address. The first three octets of the address are the OUI.

Applications

Due to the 48-bit address space, there exist 2^{48} (over 281 trillion) possible MAC addresses. This means that these could be used by one for identification.

The IEEE 802 network technology uses the EUI-48 identifier format for the same purpose. Some sample IEEE 802 networks are Ethernet, Wi-FI, Bluetooth. We can capture the network traffic by using an application like Wireshark, as done in this project.

The network traffic, stored as a pcap file, can be analyzed by using a programming language (such as python). We can then extract the mac address, thereby identifying the vendor name and the genuineness of the mac address.

Methodology

1. Creation of Database :

Firstly, we create a database that contains all the information regarding the vendors of the given MAC address(OUI specifically). In order to do so, gather the official mac address vendor list from IEEE (via the website http://standards-oui.ieee.org/oui/oui.csv which provides all the information in a CSV format).

The CSV file saved as **"oui_csv.csv"** is then used to create a sqlite3 database. We do this by executing the script **db_create.py**. The python script uses the CSV and sqlite3 modules and creates a table "mac(Registry text, Assignment text, Organisation text, Address text)" stored in a database **mac.db** . This mac.db file is used further by the main program to query the details of the vendors pertaining to a specific OUI.

2. Reading and Analysing the raw network traffic file :

The python program **mac.py** constitutes mainly of user-defined functions (APIs) such as format_mac(), get_vendors(), and sort_mac_vendors() used to analyze the raw network traffic file, such as a pcap file. It also includes a user-defined class, "Vendor," which stores the vendor information corresponding to a given OUI/Assignment. The vendor class constitutes Address, Registry, Assignment, Organisation, (Physical) Address as its members, similar to the entries of mac.db.

When the main driver code app.py is executed, the user gets a popup window to select the pcap file containing the captured ethernet/Bluetooth packets. The file contents are then parsed using scapy.rdcap() function, which takes the file name of the chosen file as a parameter and returns a reader(stored in the variable pckts) used to read the file's contents. We iterate through the contents (packets) of the file using a for loop, storing the information of a packet in a loop/iteration variable called pckt. The mac address of each entry in pckts is extracted by accessing the second layer(Ethernet layer) of each packet(i.e., since pckt is the loop variable that corresponds to a single, we can access mac address by reading the Ethernet layer and getting the source mac address, which is done as follows pckt['Ethernet'].src). The extracted mac address is then formatted

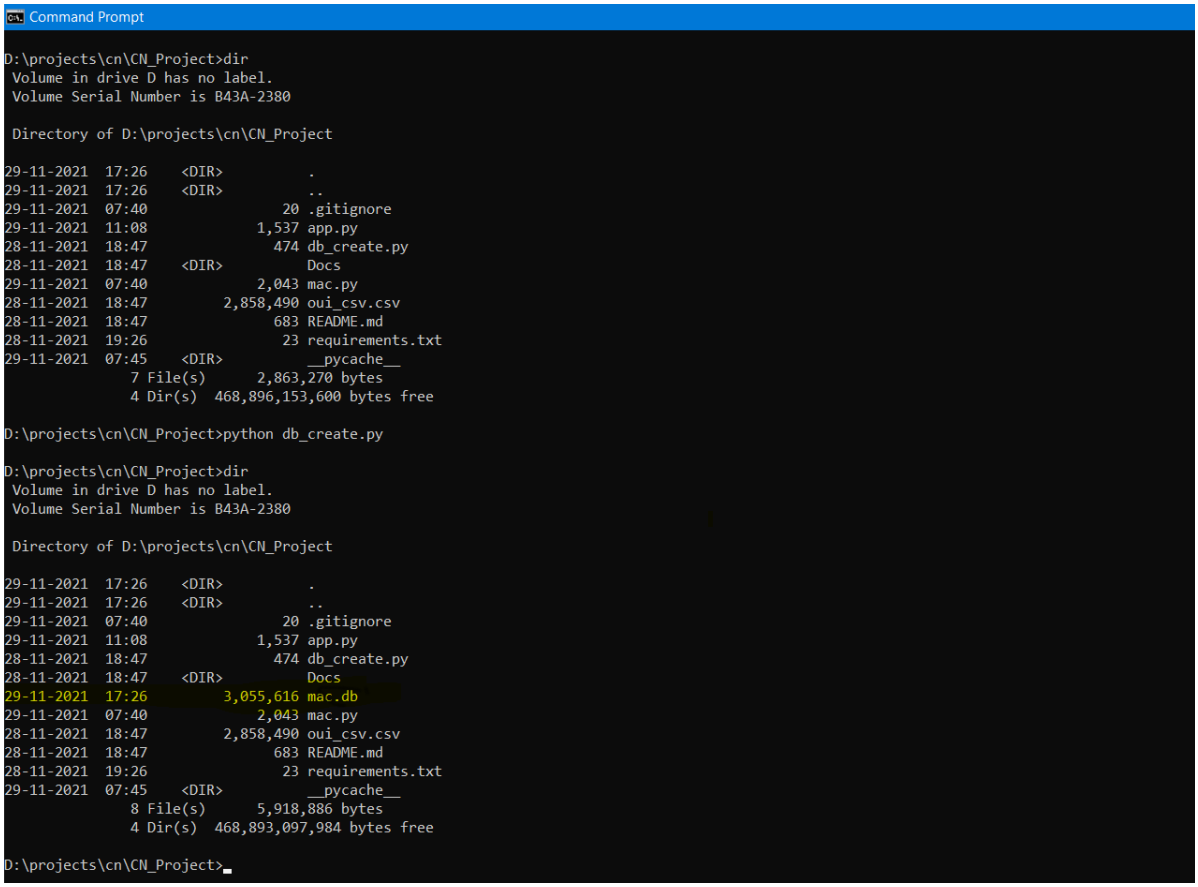
properly by passing it to the function format_mac()(in mac.py), which then returns the OUI of the mac address. The OUI obtained is then used to query the database to find an entry with the same OUI/Assignment field value in the database(achieved by the function get_vendors() in mac.py). While querying, one of the following two conditions might occur :

1) The OUI matches with a vendor's OUI in the database: In this case, we will create a Vendor class object, fill it with all the vendor details, and append it to a set "vendor_set".
2) There is no OUI in the database corresponding to the given OUI: In this case, we append the mac address to a set "fake_addrs", which is a set of all the fake mac addresses in the file.

The above functionality is handled by the sort_mac_vendors() function in mac.py. Both lists are then displayed using a simple GUI window where vendor_set denotes the details of the mac address and the vendors, whereas fake_addrs contains all the fake mac addresses. The GUI is implemented by using the python module Tkinter.

Program Screenshots

Creation of mac.db using db_create.py



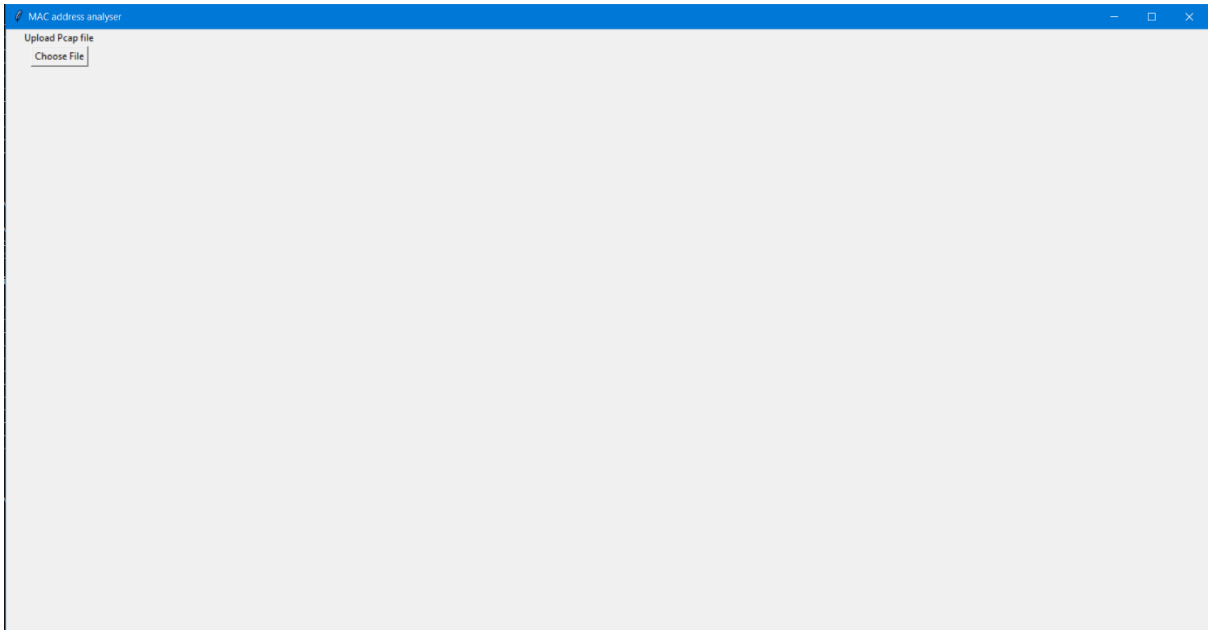
```
Command Prompt

D:\projects\cn\CN_Project>dir
 Volume in drive D has no label.
 Volume Serial Number is B43A-2380

 Directory of D:\projects\cn\CN_Project

29-11-2021  17:26    <DIR>          .
29-11-2021  17:26    <DIR>          ..
29-11-2021  07:40                20 .gitignore
29-11-2021  11:08             1,537 app.py
28-11-2021  18:47               474 db_create.py
28-11-2021  18:47    <DIR>          Docs
29-11-2021  07:40             2,043 mac.py
28-11-2021  18:47         2,858,490 oui_csv.csv
28-11-2021  18:47               683 README.md
28-11-2021  19:26                23 requirements.txt
29-11-2021  07:45    <DIR>          __pycache__
               7 File(s)      2,863,270 bytes
               4 Dir(s)  468,896,153,600 bytes free

D:\projects\cn\CN_Project>python db_create.py

D:\projects\cn\CN_Project>dir
 Volume in drive D has no label.
 Volume Serial Number is B43A-2380

 Directory of D:\projects\cn\CN_Project

29-11-2021  17:26    <DIR>          .
29-11-2021  17:26    <DIR>          ..
29-11-2021  07:40                20 .gitignore
29-11-2021  11:08             1,537 app.py
28-11-2021  18:47               474 db_create.py
28-11-2021  18:47    <DIR>          Docs
29-11-2021  17:26         3,055,616 mac.db
29-11-2021  07:40             2,043 mac.py
28-11-2021  18:47         2,858,490 oui_csv.csv
28-11-2021  18:47               683 README.md
28-11-2021  19:26                23 requirements.txt
29-11-2021  07:45    <DIR>          __pycache__
               8 File(s)      5,918,886 bytes
               4 Dir(s)  468,893,097,984 bytes free

D:\projects\cn\CN_Project>
```

```
(venv) PS C:\Users\hp\Desktop\CN_Project> sqlite3 mac.db
SQLite version 3.36.0 2021-06-18 18:36:39
Enter ".help" for usage hints.
sqlite> select * from mac limit 50;
MA-L|002272|American Micro-Fuel Device Corp.|2181 Buchanan Loop Ferndale WA US 98248
MA-L|00D0EF|IGT|9295 PROTOTYPE DRIVE RENO NV US 89511
MA-L|086195|Rockwell Automation|1 Allen-Bradley Dr. Mayfield Heights OH US 44124-6118
MA-L|F4BD9E|Cisco Systems, Inc|80 West Tasman Drive San Jose CA US 94568
MA-L|588SE9|Realme Chongqing MobileTelecommunications Corp Ltd|No.24 Nichang Boulevard, Huixing Block, Yubei District, Chongqing. Chongqing China CN 401120
MA-L|BC2392|BYD Precision Manufacture Company Ltd.|No.3001, Bao He Road, Baolong Industrial, Longgang Street,Longgang Zone, Shenzhen shenzhen  CN 518116
MA-L|405582|Nokia|600 March Road Kanata Ontario CA K2K 2E6
MA-L|A4E31B|Nokia|600 March Road Kanata Ontario CA K2K 2E6
MA-L|D89790|Commonwealth Scientific and Industrial Research Organisation|GPO Box 1700 Canberra ACT AU 2601
MA-L|883A30|Aruba, a Hewlett Packard Enterprise Company|3333 Scott Blvd Santa Clara CA US 95054
MA-L|B8A58D|Axe Group Holdings Limited|Road Town tortola  VG VG1110
MA-L|50CEE3|Gigafirm.co.LTD|3-21-8,kisonishi machida-city tokyo JP 1940037
MA-L|98E743|Dell Inc.|One Dell Way Round Rock TX US 78682
MA-L|C419D1|Telink Semiconductor (Shanghai) Co., Ltd.|No. 1500 Zuchongzhi Rd, Building #3 Shanghai  CN 201203
MA-L|887E25|Extreme Networks, Inc.|6480 Via Del Oro San Jose CA US 95119
MA-L|086083|zte corporation|12/F.,zte R&D building ,kejinan Road,Shenzhen,P.R.China shenzhen guangdong CN 518057
MA-L|E01954|zte corporation|12/F.,zte R&D building ,kejinan Road,Shenzhen,P.R.China shenzhen guangdong CN 518057
MA-L|10327E|Huawei Device Co., Ltd.|No.2 of Xincheng Road, Songshan Lake Zone Dongguan Guangdong CN 523808
MA-L|F8084F|Sagemcom Broadband SAS|250, route de l'Empereur Rueil Malmaison Cedex hauts de seine FR 92848
MA-L|30FBB8|HUAWEI TECHNOLOGIES CO.,LTD|No.2 Xin Cheng Road, Room R6,Songshan Lake Technology Park Dongguan  CN 523808
MA-L|F497C2|Nebulon Inc|3089 Skyway Court Fremont CA US 94539
MA-L|A44519|Xiaomi Communications Co Ltd|The Rainbow City of China Resources NO.68, Qinghe Middle Street Haidian District, Beijing CN 100085
MA-L|68DBF5|Amazon Technologies Inc.|P.O Box 8102 Reno NV US 89507
MA-L|2446C8|Motorola Mobility LLC, a Lenovo Company|222 West Merchandise Mart Plaza Chicago IL US 60654
MA-L|1802AE|vivo Mobile Communication Co., Ltd.|#283,BBK Road Wusha,Chang'An DongGuan City,Guangdong, CN 523860
MA-L|0C20D3|vivo Mobile Communication Co., Ltd.|#283,BBK Road Wusha,Chang'An DongGuan City,Guangdong, CN 523860
MA-L|44D791|HUAWEI TECHNOLOGIES CO.,LTD|No.2 Xin Cheng Road, Room R6,Songshan Lake Technology Park Dongguan  CN 523808
MA-L|8446FE|HUAWEI TECHNOLOGIES CO.,LTD|No.2 Xin Cheng Road, Room R6,Songshan Lake Technology Park Dongguan  CN 523808
MA-L|D82918|HUAWEI TECHNOLOGIES CO.,LTD|No.2 Xin Cheng Road, Room R6,Songshan Lake Technology Park Dongguan  CN 523808
```

Executing the command

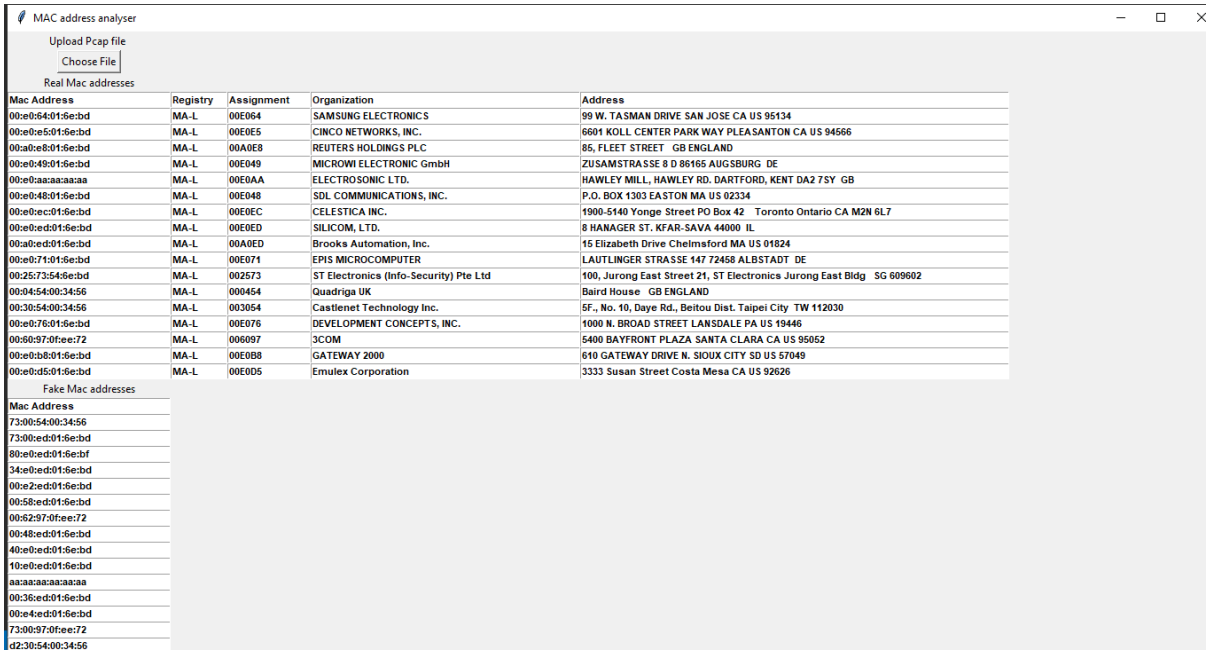
```
D:\projects\cn\CN_Project>python app.py
```

Will open up a popup window



Here we must select the required pcap file using the choose file button.

Now the script runs in the background, and upon completion, the results are updated in the same window as shown below.



As shown above, the tool lists all the vendor details for real mac addresses and explicitly displays the fake mac addresses.

Program Code

Note: The entire code for this program is uploaded to https://github.com/Gagan-Shenoy/CN_Project *for ease of execution.*

db_create.py

```
import csv

import sqlite3



con = sqlite3.connect("mac.db")

cur = con.cursor()

cur.execute("DROP TABLE IF EXISTS mac")

cur.execute("CREATE TABLE mac (Registry text, Assignment text,
Organization text, Address text);")

with open("oui_csv.csv", encoding="cp437") as f:

    reader = csv.DictReader(f)

    #field_names = reader.fieldnames

    for row in reader:

        cur.execute("INSERT INTO mac VALUES (?, ?, ?, ?)",
list(row.values()))

con.commit()

con.close()
```

mac.py

```
#necessary code to execute the main program app.py

import sqlite3
```

```
import scapy.all as scapy


class Vendor:

    def __init__(self, vendor_tuple, mac_addr):

        self.Registry, self.Assignment, self.Organization, self.Address =
vendor_tuple

        self.mac_addr = mac_addr



    def __str__(self):

        return f"Mac Address - {self.mac_addr}, Registry - {self.Registry},
Assignment - {self.Assignment}, Organization - {self.Organization},
Address - {self.Address}"



    def __eq__(self, other):

        return self.Assignment == other.Assignment



    def __hash__(self):

        return int(self.Assignment, 16)



    def to_list(self):

        return [self.mac_addr, self.Registry, self.Assignment,
self.Organization, self.Address]



def get_vendors(mac_addr):

    '''Input - Mac Address
```

```
        Output - List of Vendor Objects for the given Mac Address
    '''
    con = sqlite3.connect("mac.db")
    cur = con.cursor()
    result = cur.execute("SELECT * FROM mac WHERE Assignment = ?",
[format_mac(mac_addr)]).fetchall()
    vendors = []
    for i in result:
        vendors.append(Vendor(i, mac_addr))
    con.close()
    return vendors


def format_mac(mac_addr):
    t = mac_addr.split(sep = ":", maxsplit = 4)
    fmac_addr = "".join(t[:3])
    fmac_addr = fmac_addr.upper()
    return fmac_addr


def sort_mac_vendors(file):
    pckts = scapy.rdpcap(file)
    vendor_set = set()
    fake_addrs = set()
    for pckt in pckts:
```

```
        try :
            mac_addr = pckt['Ethernet'].src
            vendors = get_vendors(mac_addr)
            for vendor in vendors:
                vendor_set.add(vendor)
            if not vendors:
                fake_addrs.add(mac_addr)
        except:
            continue
    return vendor_set, fake_addrs

# file = input("Enter file(pcap) path : ")
# vendor_set, fake_addrs = sort_mac_vendors(file)
# print("Real Mac Addresses with Vendors : ")
# for vendor in vendor_set:
#     print(vendor)

# print("\nFake Mac Addresses : ")
# for mac_addr in fake_addrs:
#     print(mac_addr)
```

app.py

```
from tkinter import *
from tkinter import ttk
from tkinter.filedialog import askopenfile
from tkinter.filedialog import askopenfilename
from mac import *


def open_file():
    filetypes = [('Pcap', '*.pcap')]
    file = askopenfilename(title = 'Open a file', filetypes = filetypes)
    vendor_set, fake_addrs = sort_mac_vendors(file)
    mac = Label(ws, text = 'Real Mac addresses')
    mac.grid(row = 5, column = 0, padx = 10)
    vendor_list = [['Mac Address', 'Registry', 'Assignment', 'Organization',
'Address']]
    for i in vendor_set:
        vendor_list.append(i.to_list())
    widths = [30, 10, 15, 50, 80]
    for i in range(len(vendor_list)):
        for j in range(5):
            e = Entry(ws, width = widths[j], font = ('Arial', 8,'bold'))
            e.grid(row = 6 + i, column = j)
            e.insert(END, vendor_list[i][j])
    fmac = Label(ws, text = 'Fake Mac addresses')
```

```
    fmac.grid(row = 7 + len(vendor_list), column = 0, padx = 10)
    fakeaddr_list = ['Mac Address'] + list(fake_addrs)
    for i in range(len(fakeaddr_list)):
        e = Entry(ws, width = 30, font = ('Arial', 8,'bold'))
        e.grid(row = 8 + len(vendor_list) + i, column = 0)
        e.insert(END, fakeaddr_list[i])


ws = Tk()
ws.title('MAC address analyser')
ws.geometry('1500x750')
file = Label(ws, text = 'Upload Pcap file ')
file.grid(row = 0, column = 0, padx = 20)
filebtn = Button(ws, text = 'Choose File', command = open_file)
filebtn.grid(row = 1, column = 0, padx = 20)
ws.mainloop()
```