Tarea 2: Consultar Fabricantes de tarjetas de red a través de una API

Eduardo Amigo Araya, eduardo.amigo@alumnos.uv.cl

1. Introducción

La tarea presente tiene como objetivo la implementación de una herramienta que permita consultar el fabricante de una tarjeta de red a través de su dirección MAC, utilizando una API REST de consulta de direcciones MAC para obtener información según las direcciones

2. Descripción del problema y diseño de la solución

Para solucionar el problema dado, se usará la API REST de https://maclookup.app/ con el fin de consultar las direcciones MAC. Además, se realizará un programa en Python que realice las consultas ingresadas en línea de comandos.

Para realizar las consultas, se utilizará la librería *requests* dentro del programa.

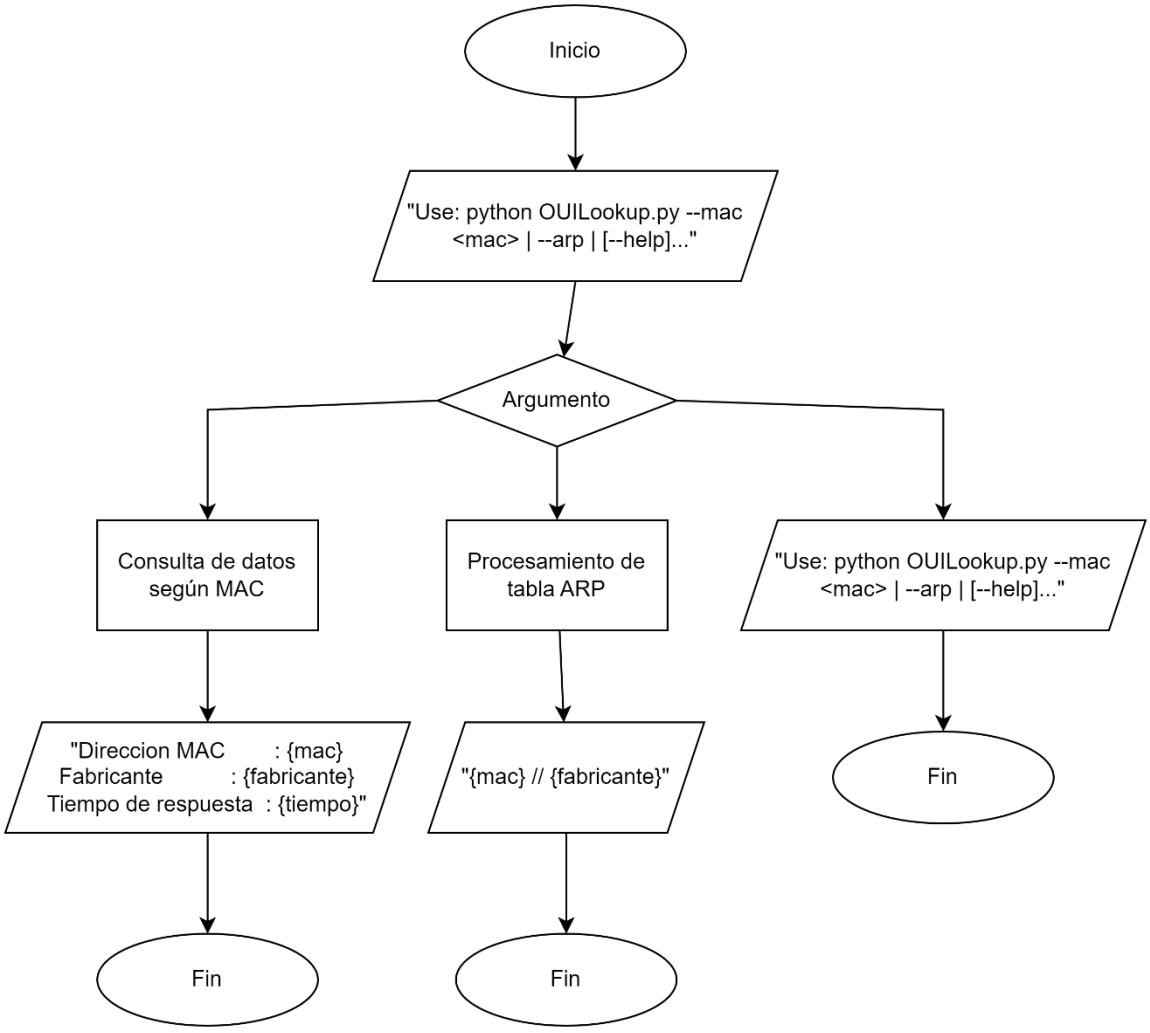

Figura 1: Diagrama de flujo del programa

3. Implementación

Algunas funciones claves para la resolución del problema son:

get_fabricante(string)

Esta función recibe como argumento una dirección MAC, realiza la consulta al servidor, y en caso de funcionar correctamente, retorna el fabricante y el tiempo de ejecución.

```python
def get_fabricante(direccion_mac):
    url = f"https://api.maclookup.app/v2/macs/{direccion_mac}"
    try:
        tiempo1 = time.time()
        respuesta = requests.get(url)
        tiempo2 = time.time()
        tiempo_respuesta = (tiempo2 - tiempo1) * 1000

        if respuesta.status_code == 200:
            fabricante = respuesta.json().get('company', 'Fabricante no encontrado')
        else:
            fabricante = None #En caso de error
    except requests.RequestException as e:
        print(f"Error en la consulta: {e}")
        return None, None

    return fabricante, tiempo_respuesta
```

Figura 2: Función get_fabricante()

obtener_tabla_arp()

Esta función ejecute el comando arp -a en línea de comandos, y luego consulta a la API por cada dirección MAC encontrada en la tabla.

```python
def obtener_tabla_arp():
    #ejecuta arp -a sin mostrarlo por pantalla
    resultado = subprocess.run(['arp', '-a'], capture_output=True, text=True)
    lineas = resultado.stdout.splitlines()[3:]  #ignorar las primeras 2 lineas
    #lista de direcciones mac en la tabla arp
    direcciones_mac = []
    for linea in lineas:
        partes = linea.split()
        if len(partes) > 1:
            mac = partes[1]
            direcciones_mac.append(mac)
    return direcciones_mac
```

Figura 3: Función obtener_tabla_arp()

main(argv)

La función main del programa recibe el argumento dado por línea de comandos, además, define el funcionamiento para cada posible argumento

```python
def main(argv):
    mac = None
    arp = False

    try:
        #definicion de parametros
        opts, args = getopt.getopt(argv, "hm:a", ["help", "mac=", "arp"])
    except getopt.GetoptError:
        mostrar_ayuda()
        sys.exit(2)

    for opt, arg in opts:
        if opt in ("--help"):
            mostrar_ayuda()
            sys.exit()
        elif opt in ("--mac"):
            mac = arg
        elif opt in ("--arp"):
            arp = True
```

Figura 4: Función main

Opción --mac

```python
71      if mac:
72          fabricante, tiempo_respuesta = get_fabricante(mac)
73          if fabricante: #en caso de fabricante valido
74              print(f"""
75    Direccion MAC        : {mac}
76    Fabricante           : {fabricante}
77    Tiempo de respuesta  : {tiempo_respuesta:.0f} ms""")
78          else: #en caso de fabricante not found
79              print(f"""
80    Direccion MAC        : {mac}
81    Fabricante           : Not found
82    Tiempo de respuesta  : {tiempo_respuesta:.0f} ms""")
```

Figura 5: Argumento --mac

Opción --arp

```
83      elif arp:
84          #bbtener la tabla ARP y procesar cada direccion MAC
85          direcciones_mac = obtener_tabla_arp()
86          print("MAC // Fabricante")
87          for mac in direcciones_mac:
88              fabricante, tiempo_respuesta = get_fabricante(mac)
89              if fabricante:
90                  print(f"""{mac} // {fabricante}""")
91              else:
92                  print(f"{mac} // Fabricante no encontrado")
```

Figura 6: Argumento --arp

Otros argumentos

```
93      else:
94          mostrar_ayuda()
95          sys.exit(2)
```

Figura 7: Otros argumentos

4. Pruebas

Para las pruebas, se ingresaron mediante línea de comandos los ejemplos dados en la tarea, además, se hicieron pruebas adicionales con la tabla ARP de la red local.

```
Windows PowerShell
PS C:\Users\eduam\OneDrive\Documents\GitHub\Tarea02-Amigo-Araya-Eduardo> python3 OUILookup.py --mac 98:06:3c:92:ff:c5

Direccion MAC        : 98:06:3c:92:ff:c5
Fabricante           : Samsung Electronics Co.,Ltd
Tiempo de respuesta  : 513 ms
PS C:\Users\eduam\OneDrive\Documents\GitHub\Tarea02-Amigo-Araya-Eduardo> python3 OUILookup.py --mac 9c:a5:13

Direccion MAC        : 9c:a5:13
Fabricante           : Samsung Electronics Co.,Ltd
Tiempo de respuesta  : 517 ms
PS C:\Users\eduam\OneDrive\Documents\GitHub\Tarea02-Amigo-Araya-Eduardo> python3 OUILookup.py --mac 48-E7-DA

Direccion MAC        : 48-E7-DA
Fabricante           : AzureWave Technology Inc.
Tiempo de respuesta  : 502 ms
PS C:\Users\eduam\OneDrive\Documents\GitHub\Tarea02-Amigo-Araya-Eduardo> python3 OUILookup.py --mac 98:06:3f:92:ff:c5

Direccion MAC        : 98:06:3f:92:ff:c5
Fabricante           : Not found
Tiempo de respuesta  : 534 ms
PS C:\Users\eduam\OneDrive\Documents\GitHub\Tarea02-Amigo-Araya-Eduardo>  python3 OUILookup.py --arp
MAC // Fabricante
9c-0b-05-38-b4-72 // eero inc.
90-f8-2e-25-33-38 // Amazon Technologies Inc.
48-43-dd-4b-61-83 // Amazon Technologies Inc.
3e-44-1b-04-3f-f5 // Fabricante no encontrado
d4-12-43-71-b8-24 // AMPAK Technology, Inc.
ff-ff-ff-ff-ff-ff // Fabricante no encontrado
01-00-5e-00-00-02 // Fabricante no encontrado
01-00-5e-00-00-07 // Fabricante no encontrado
01-00-5e-00-00-16 // Fabricante no encontrado
01-00-5e-00-00-fb // Fabricante no encontrado
01-00-5e-00-00-fc // Fabricante no encontrado
01-00-5e-7f-ff-fa // Fabricante no encontrado
ff-ff-ff-ff-ff-ff // Fabricante no encontrado
```

Figura 8: Pruebas en línea de comandos

5. Discusión y conclusiones

5.1. Direcciones MAC aleatorias

Las direcciones MAC aleatorias, son direcciones temporales generados por dispositivos de red al conectarse a una red Wi-Fi. El uso de direcciones MAC aleatorias es considerado una herramienta importante para mejorar la privacidad de los usuarios. [1]

La aleatorización de MAC aumenta la privacidad del usuario, ya que de lo contrario, las direcciones MAC puedes capturarse y usarse para rastrear la ubicación de un usuario. [2]

5.2. Conclusión

Finalmente, como se puede ver en las imágenes, se logró el funcionamiento esperado del programa, logrando hacer la consulta a la API y la respuesta correspondiente dependiendo del caso. En lo personal, además, fue un desafío la implementación de un programa que funcione a través de línea de comandos, pero finalmente fue logrado sin mayores problemas.

6. Referencias

[1] Gomez, C.A., Guerrero, L.J. y Pedraza, L.F. (2022) "Evolution of the use of random MAC addresses in public WI-fi networks", *Journal of Engineering Science and Technology Review*, 15(3), pp. 147–152. Disponible en: https://doi.org/10.25103/jestr.153.15.

[2] Comportamiento de aleatorización MAC (sin fecha) Android Open Source Project. Disponible en: https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior?hl=es