Consultar Fabricantes de tarjetas de red a través de una API.

Martin Mendez Herrera, martin.mendez@alumnos.uv.cl

Benjamín Acevedo Jorquera, Benjamin.acevedo@alumnos.uv.cl

1. Introducción

En la era moderna las direcciones MAC (Media Access Control) son identificadores únicos asignadas a las redes de un dispositivo, como tarjetas de red, routers, o cualquier equipo conectado a internet. Todas las MAC incluyen un identificador llamado OUI (Organizationally Unique Identifier) que será un bloque de direcciones asignadas a un fabricante especifico a su vez estas serán "OUI is taken from the first 6 hexadecimal digits out of 12 hexadecimal digits [5], or 24 bits out of 48 bits of the device's MAC address." (Riadi, Fadlil, & Prabowo, 2024, p. 3). OUI le permite definir el propósito de su construcción, incluida la gestión de la red y la seguridad. Conociendo al fabricante, podrá conectarse a dispositivos no particionados en la red y aprender a administrar los servicios de red.

Por otra parte, para obtener los fabricantes de las tarjetas de red se utilizará una API la cual será "una nueva opción o estilo de uso de los Servicios Web (Web Services, WS), para la creación de servicios" (Arsaute et al., 2018, p. 629). Lo cual nos permitirá desarrollar una codificación sin tener que implementar toda la lógica desde cero.

En un entorno donde la privacidad y la seguridad son clave e importantes, el uso del API es para identificar los fabricantes a través de la dirección MAC esto permitirá que los administradores puedan controlar la red y a la vez protegerlos. Esto reducirá el riesgo de recibir instrucciones no autorizadas y podrá identificar los dispositivos conectados.

Respecto a lo anterior, se utilizará una API de consulta de dirección MAC para devolver el nombre del fabricante de la dirección en cuestión. Para ello se generará una herramienta de línea de comandos "OUILookup" que será desarrollada en Python.

Posterior a esto luego de realizar diferentes pruebas con la implementación en el lenguaje planteado se realizarán diferentes discusiones tanto como conclusiones sobre el funcionamiento de las API REST para las consultas en funcionamiento a si como de sus diferentes fabricantes.

2. Descripción del problema y diseño de la solución

El objetivo de esta tarea es desarrollar una herramienta en Python que permita consultar el fabricante de una tarjeta de red mediante su dirección MAC. Para lograr esto, se utilizará una API REST que proporciona información sobre los fabricantes de tarjetas de red basándose en direcciones MAC. El desarrollo de esta herramienta, denominada "OUILookup", se realizará en línea de comandos y deberá ser capaz de funcionar tanto en sistemas operativos Linux como Windows. Además, deberá manejar la entrada de parámetros de forma eficiente a través de la librería getopt, siguiendo los principios de la programación funcional.

2.1 Requerimientos y especificaciones

2.1.1 Nombre del programa: OUILookup.
2.1.2 Lenguaje de programación: Python.

2.1.3 Parámetros de entrada:

- Parametro1: --mac
- Parametro2. –arp
- Parametro3. –help

2.1.4 **Fuente de datos**: El programa deberá hacer uso de una API REST pública, como la disponible en maclookup.app, para obtener la información del fabricante asociado a una dirección MAC.

2.1.5 Respuesta del programa:

- Si la dirección MAC consultada está registrada en la base de datos, deberá devolver el nombre del fabricante y el tiempo de respuesta.
- Si la dirección MAC no está registrada, deberá indicar que no se encontró.
- En el caso de usar la opción --arp, deberá listar las direcciones MAC presentes en la tabla ARP junto con sus fabricantes.

2.2 Diseño de la solución

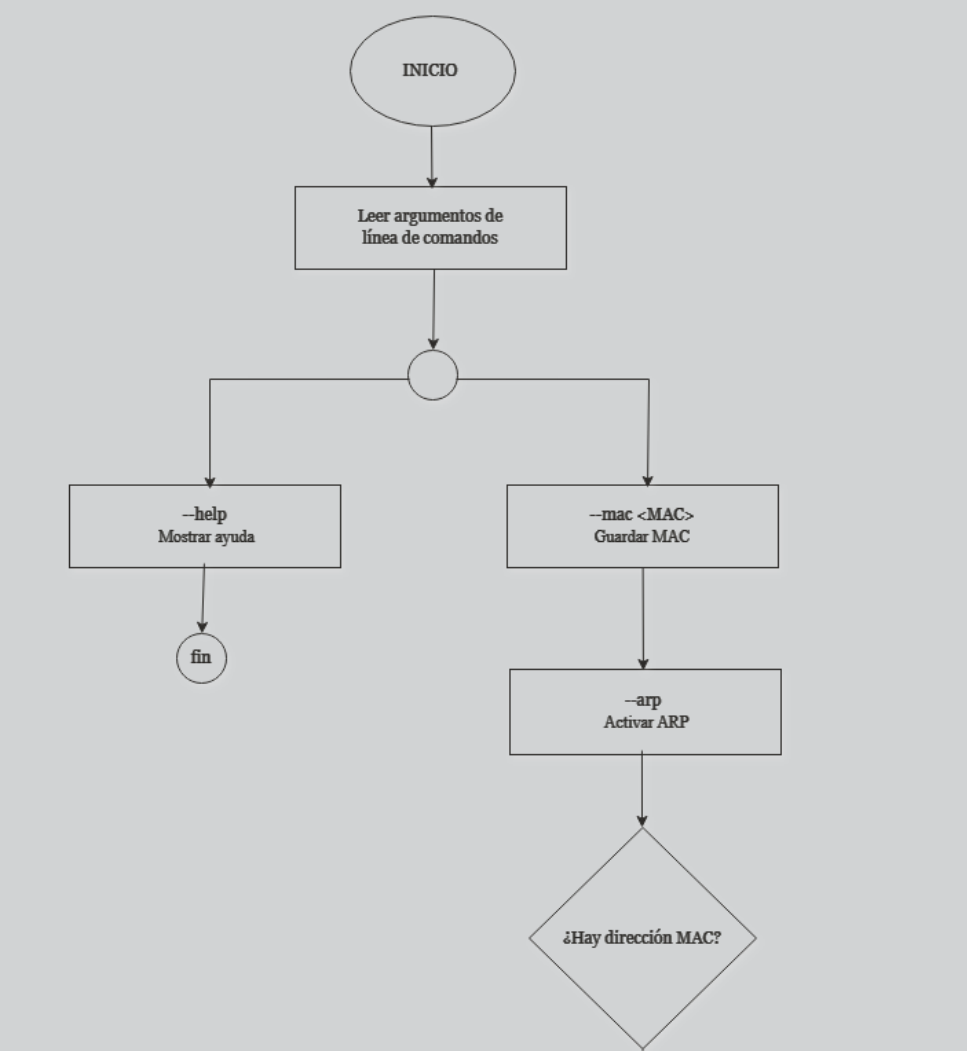

Ilustración 1. Diseño de la solución basado en la problemática

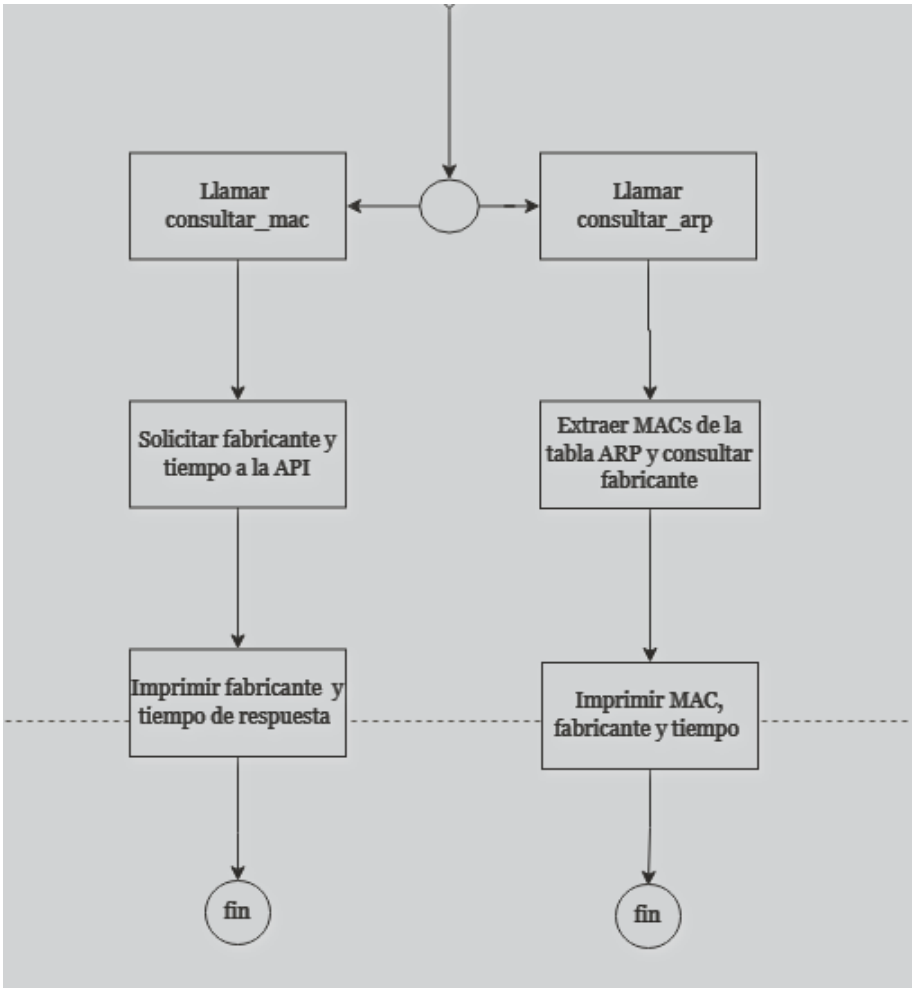

Ilustración 2. Diseño de la solución basado en la problemática

3. Implementación

3.1Librerias utilizadas.

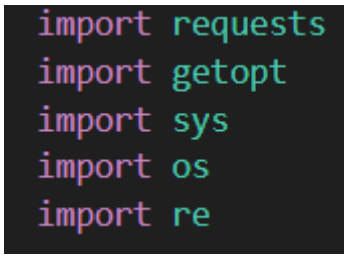

Ilustración 3. Imagen de las librerías del programa

3.1.1 Librería y función

- Requests: Para hacer solicitudes HTTP
- Getopt: Para manejar argumentos de línea de comandos
- Sys: Para interactuar con el sistema
- Os: Para ejecutar comandos en la terminal
- Re: Para buscar patrones de texto (en este caso, direcciones MAC)

3.2 Función consultar_mac

```python
def consultar_mac(mac_address):



    API_ENDPOINT = 'https://api.maclookup.app/v2/macs/'

    try:


        respuesta = requests.get(API_ENDPOINT + mac_address)



        if respuesta.status_code == 200:
            datos = respuesta.json()
            fabricante = datos.get("company", "Fabricante no encontrado")
            tiempo_respuesta = respuesta.elapsed.total_seconds() * 1000
            return fabricante, tiempo_respuesta
        else:
            return "Error en la consulta", None
    except Exception as error:
        return f"Error al hacer la consulta: {error}", None
```

La función consultar_mac consulta el fabricante de una Mac utilizando la API luego de esto si la solicitud es correcta nos devolverá el nombre del fabricante y el tiempo de respuesta en ms de la solicitud por otro lado en caso de ser incorrecta devolver error.

3.3 Función consultar_arp.

```
def consultar_arp():



    print("Consultando la tabla ARP.")
    command = "arp -a"
    try:
        arp_table = os.popen(command).read()
        print("MAC/Vendor:")



        mac_addresses = re.findall(r"([0-9a-fA-F]{2}(?:[:-][0-9a-fA-F]{2}){5})", arp_table)


        for mac in mac_addresses:
            fabricante, tiempo_respuesta = consultar_mac(mac)
            if fabricante:
                print(f"{mac} / {fabricante} - Tiempo de respuesta: {tiempo_respuesta:.2f} ms")
            else:
                print(f"{mac} / No encontrado")

    except Exception as e:
        print(f"Error al consultar la tabla ARP: {e}")
```

Ilustración 5.imagen de función consultar_arp

La función consultar_arp consulta la tabla arp del sistema y obtiene el fabricante de cada Mac a su vez esta misma se encargará de mostrar la tabla y consultar el fabricante de cada dirección Mac en la tabla y en caso de que esta consulta tenga un fallo esta nos devolverá “error al consultar la tabla arp”.

3.4 Función mostrar_ayuda.

```
def mostrar_ayuda():



    help_text = (
        "Uso: OUILookup.py --mac <direccion_mac> | --arp | [--help]\n"
        " -m, --mac  : Dirección MAC a consultar. Ejemplo: aa:bb:cc:00:00:00.\n"
        " -a, --arp  : Muestra los fabricantes de los hosts en la tabla ARP.\n"
        "      --help : Muestra este mensaje de ayuda.\n"
    )
    print(help_text)
```

Ilustración 6. Imagen de la función mostrar_ayuda.

La función mostrar_ayuda se encargará de mostrarnos como utilizar el programa a través del comando help que nos devolverá los principales comandos a utilizar en la consola, con lo anterior se ejecutaran según el comando a realizar se llame la función adecuada referente a esa llamada.

3.5 Función main.

```python
def main(argv):


    mac_address = None
    mostrar_arp = False


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


    for opt, arg in opts:
        if opt in ("-h", "--help"):
            mostrar_ayuda()
            sys.exit()
        elif opt in ("-m", "--mac"):
            mac_address = arg
        elif opt in ("-a", "--arp"):
            mostrar_arp = True


    if mac_address:
        fabricante, tiempo_respuesta = consultar_mac(mac_address)
        print(f"Dirección MAC : {mac_address}")
        print(f"Fabricante    : {fabricante}")
        if tiempo_respuesta is not None:
            print(f"Tiempo de respuesta: {tiempo_respuesta:.2f} ms")
    elif mostrar_arp:
        consultar_arp()
    else:
        mostrar_ayuda()
```

Ilustración 7. Imagen de función main

La función main se encargará de procesar los argumentos de las líneas de comandos que realizaran distintas operaciones relacionadas con las direcciones Mac y las tablas arp.

3.6 Condición.

```python
if __name__ == "__main__":
    main(sys.argv[1:])
```

Ilustración 8. Imagen de condición para función main

Este bloque de condición verifica que la función main solo se ejecute si el archivo se está ejecutando a su vez pasa los argumentos de la línea de comandos para que el programa pueda trabajar con ellos.

4. Pruebas

Para realizar las pruebas de la implementación y su correcto funcionamiento se ingresarán comandos de ejemplo dados con anterioridad por el docente a cargo.

4.1 ejemplo de uso sin parámetros.

```
Uso: OUILookup.py --mac <direccion_mac> | --arp | [--help]
 -m, --mac  : Dirección MAC a consultar. Ejemplo: aa:bb:cc:00:00:00.
 -a, --arp  : Muestra los fabricantes de los hosts en la tabla ARP.
     --help : Muestra este mensaje de ayuda.
```

Ilustración 9. Imagen de consola a la hora de ejecutar sin parámetros

La imagen anterior nos mostrara lo que nos mostrara el comando help por haber ejecutado sin parámetros el programa mostrando así ejemplos de como llevar a cabo una localización con una Mac a consultar.

4.2 Ejemplo de uso con parámetros.

4.2.1 Caso de Mac que esta en la base de datos.

```
C:\Users\marti\Desktop\ouilook.py>python OUILookup.py --mac 98:06:3c:92:ff:c5
Dirección MAC : 98:06:3c:92:ff:c5
Fabricante    : Samsung Electronics Co.,Ltd
Tiempo de respuesta: 821.01 ms
```

Ilustración 10. Imagen de consola al ingresar ejemplo con parámetros

En el caso de la Mac que se encuentra en la base de datos nos devolverá el fabricante y su tiempo de respuesta respectivo en milisegundos.

4.2.2 Caso de Mac que no esta en la base de datos.

```
C:\Users\marti\Desktop\ouilook.py>python OUILookup.py --mac 98:06:3f:92:ff:c5
Dirección MAC : 98:06:3f:92:ff:c5
Fabricante    :
Tiempo de respuesta: 790.65 ms
```

Ilustración 11. Imagen de consola de ejemplo con parámetros.

En el caso de la Mac que no está en la base de datos esta no mostrará el fabricante, pero si devolverá el tiempo de respuesta en milisegundos.

4.2.3 Caso fabricantes de las MAC disponibles en la tabla arp

```
C:\Users\marti\Desktop\ouilook.py>python OUILookup.py --arp
Consultando la tabla ARP.
MAC/Vendor:
2c-96-82-db-ea-80 / MitraStar Technology Corp. - Tiempo de respuesta: 998.16 ms
a0-d7-f3-97-6f-1c / Samsung Electronics Co.,Ltd - Tiempo de respuesta: 667.25 ms
50-da-d6-64-b5-ca / Xiaomi Communications Co Ltd - Tiempo de respuesta: 627.38 ms
28-f5-d1-d5-f4-45 / ARRIS Group, Inc. - Tiempo de respuesta: 754.51 ms
```

Ilustración 12. Imagen de consola de ejemplo con parámetros.

En el caso de los fabricantes de las Mac disponibles en la tabla arp nos mostrara la Mac perteneciente a ellos, el fabricador y su tiempo de respuesta en milisegundos.

4.3 Casos de pruebas definitivos.

4.3.1 Caso 1

```
C:\Users\marti\Desktop\ouilook.py>python OUILookup.py --mac 98:06:3c:92:ff:c5
Dirección MAC : 98:06:3c:92:ff:c5
Fabricante    : Samsung Electronics Co.,Ltd
Tiempo de respuesta: 668.85 ms
```

Ilustración 13. Imagen de caso de prueba definitiva

4.3.2 Caso 2

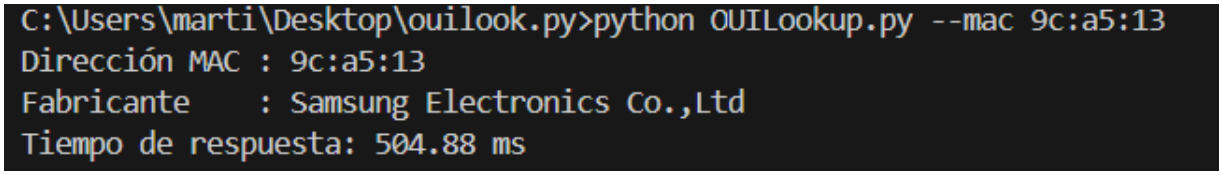

Ilustración 14. Imagen de caso de prueba definitiva.

4.3.3 Caso 3

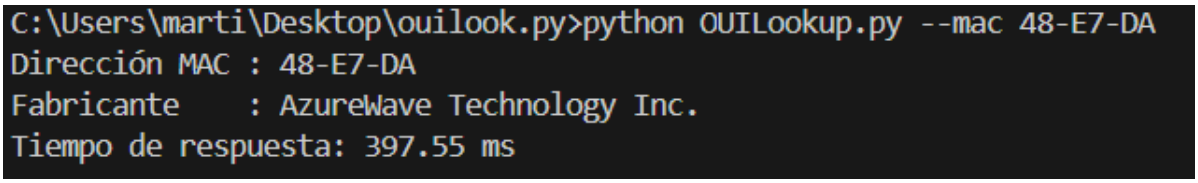

Ilustración 15. Imagen de caso de prueba definitiva.

5. Discusión

Durante el desarrollo de la herramienta "OUILookup", se abordaron diversos aspectos relacionados con el manejo de direcciones MAC y el uso de APIs para consultar los fabricantes de tarjetas de red. En este contexto, fue necesario realizar una investigación profunda sobre cómo funcionan las direcciones MAC, especialmente el componente OUI (Organizationally Unique Identifier), que permite identificar el fabricante de un dispositivo a partir de su dirección MAC. Esto resultó esencial para entender cómo funcionan las API REST en el contexto de la red y la seguridad.

El uso de la API pública, como la de **maclookup.app**, facilitó el acceso a bases de datos globales de fabricantes sin necesidad de implementar desde cero un sistema de consulta. La API REST demostró ser eficiente al proporcionar información precisa y crucial en tiempo real, lo que confirma su importancia como herramienta para la creación de aplicaciones robustas y escalables.

Al probar la herramienta "OUILookup", se validó su funcionalidad en diferentes sistemas operativos, tanto Linux como Windows, lo que asegura su portabilidad. Los resultados de las pruebas mostraron que:

- La herramienta pudo identificar correctamente fabricantes de direcciones MAC registradas en la base de datos, devolviendo también el tiempo de respuesta de la consulta, lo que permite un seguimiento adicional de la eficiencia de las solicitudes.
- En casos donde las direcciones MAC no se encontraban en la base de datos, la herramienta manejó la situación de manera adecuada, indicando que no se encontró información sin generar errores que interrumpieran el funcionamiento del programa.
- La funcionalidad para consultar las direcciones presentes en la tabla ARP fue exitosa, proporcionando detalles sobre los dispositivos conectados a la red local y los fabricantes de sus tarjetas de red.

5.1 Conclusión

El uso de direcciones MAC y su identificación mediante APIs REST es una práctica eficiente y necesaria en la administración de redes.
Las APIs REST no solo facilitan el desarrollo de aplicaciones, sino que también permiten ahorrar tiempo y recursos al no tener que implementar bases de datos y lógica complejas desde cero.
El programa "OUILookup" demuestra cómo la automatización y la programación pueden ser utilizadas para resolver problemas comunes de red de manera eficaz y escalable.

Es fundamental tener en cuenta las consideraciones de seguridad y privacidad al manipular información de red, ya que las direcciones MAC pueden ser utilizadas para rastrear dispositivos si no se toman las precauciones adecuadas.

5. Referencias

[1] Riadi, I., Fadlil, A., & Adhi Prabowo, B. (2024). Quick Random MAC Address Detection Based on Organizationally Unique Identifier in Captive Portal. *International Journal of Computing and Digital Systems*, *16*(1), 1-11.

[2] Arsaute, A., Zorzán, F. A., Daniele, M., González, A., & Frutos, M. (2018). Generación automática de API REST a partir de API Java, basada en transformación de Modelos (MDD). In *XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste)*..