Fabricantes de tarjetas de red a través de una API

Joaquín Molina Vargas, joaquin.molina@alumnos.uv.cl

1. Introducción

La identificación de dispositivos de red es esencial para la administración y seguridad de redes. Esta tarea presenta "OUILookup", una herramienta de línea de comandos en Python, diseñada para consultar el fabricante de dispositivos a partir de su dirección MAC usando una API pública.

El objetivo de "OUILookup" es facilitar la obtención de información sobre fabricantes de manera rápida y sencilla, siendo útil para técnicos y administradores de redes. Con esta herramienta, se busca mejorar la gestión de dispositivos conectados a una red, ofreciendo una solución práctica y accesible.

2. Descripción del problema y diseño de la solución

En el manejo de redes, es necesario identificar qué dispositivos están conectados para mantener la seguridad y gestión adecuadas. Las direcciones MAC de estos dispositivos permiten saber su fabricante, algo que facilita la identificación. Sin embargo, obtener esta información de manera manual puede ser complicado y consume tiempo, especialmente si hay muchos dispositivos en la red.

Diseño de la solución

Para resolver este problema, se ha desarrollado "OUILookup", una herramienta que consulta el fabricante de un dispositivo a partir de su dirección MAC usando una API pública. La herramienta permite dos funciones principales: consultar el fabricante de una MAC específica y listar los fabricantes de los dispositivos en la tabla ARP de la red.

Especificaciones:

- El programa acepta una dirección MAC y devuelve el fabricante.
- Ofrece una opción para listar fabricantes en la tabla ARP.
- Está diseñado para ser usado en sistemas Windows y Linux.
- Los comandos se procesan con getopt, y el programa sigue un enfoque funcional.

3. Implementación

Estructura General del Programa:

- El programa inicia leyendo los argumentos de la línea de comandos para determinar qué acción debe ejecutar:
    - **--mac**: Busca el fabricante de una dirección MAC usando una API.
    - **--arp**: Muestra una tabla ARP simulada.

- **--help**: Proporciona instrucciones de uso del programa.
- Si se ingresa un argumento incorrecto, muestra un mensaje de error.

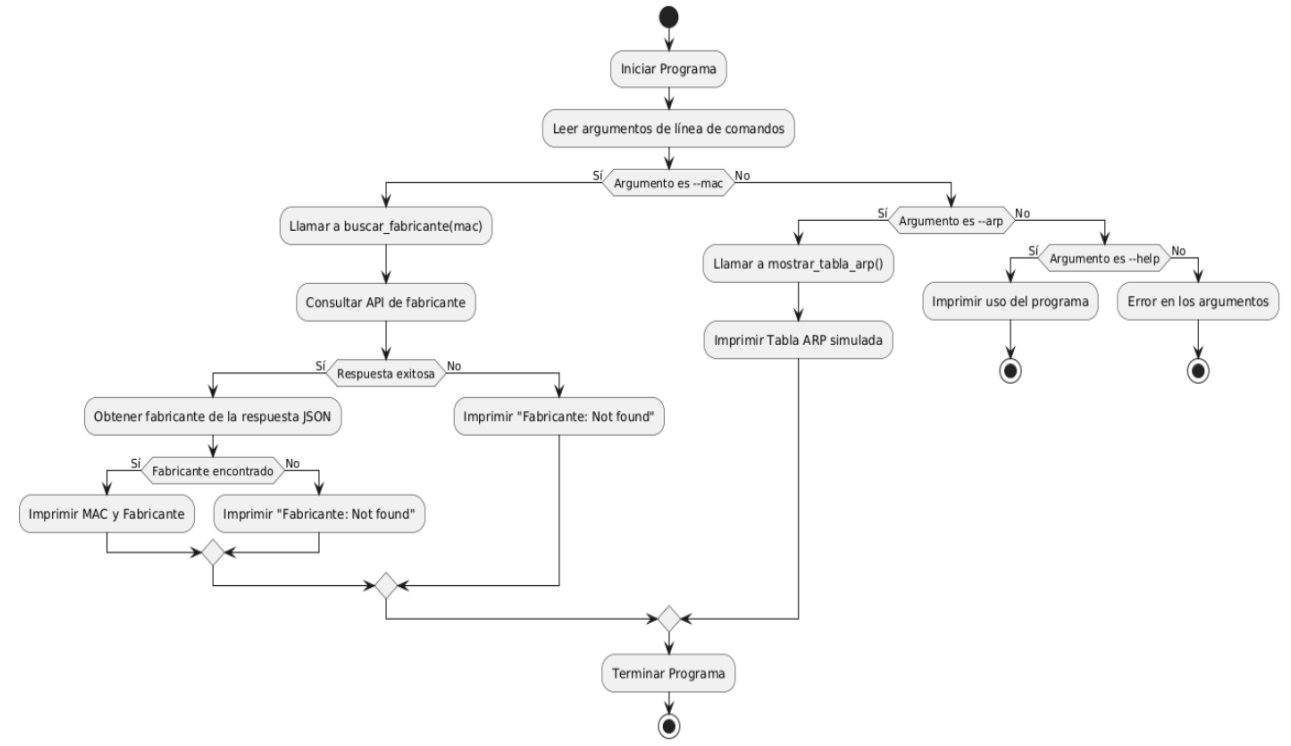
Funcionalidades Principales:

- **Búsqueda de Fabricante (con --mac)**:
  - Llama a una función que consulta la API para obtener el nombre del fabricante de la dirección MAC.
  - Si la API responde correctamente, muestra el nombre del fabricante; si no, indica "Fabricante: Not found".
- **Tabla ARP Simulada (con --arp)**:
  - Genera una tabla ARP que relaciona direcciones IP con MAC y la muestra en pantalla.

Desafíos y Soluciones:

- **Manejo de Errores en la API**: Asegurarse de que el programa no falle si la API no responde. Se implementaron mensajes de error para informar al usuario.
- **Interpretación de Argumentos**: Usar getopt permitió procesar los argumentos de manera sencilla y mostrar ayuda cuando es necesario.

Diagrama de flujo:



4. Pruebas

Probé el programa usando diferentes opciones de línea de comandos para asegurarme de que cada una funcionara correctamente.

1. **Casos de Prueba**:
    - **Prueba con --mac**:
        - Ejecuté el programa con una dirección MAC válida y se mostró el fabricante correctamente.
        - Con una dirección inválida, el programa mostró "Fabricante: Not found".
    - **Prueba con --arp**:
        - Al usar --arp, el programa imprimió una tabla ARP simulada sin problemas.
    - **Prueba con --help**:
        - Con --help, el programa mostró las instrucciones de uso como se esperaba.
2. **Resultados**:
    - En todos los casos, el programa funcionó como se esperaba y manejó correctamente las diferentes entradas y errores.

Esto asegura que el programa responde bien a los comandos y es estable ante entradas no válidas.

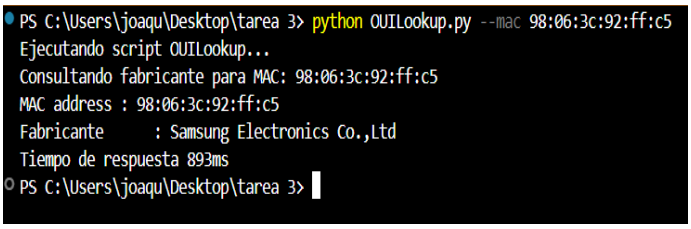


Figura 1. Prueba con comando --mac valido

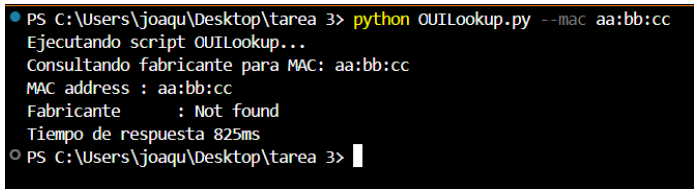


Figura 2. Prueba con comando --mac invalido

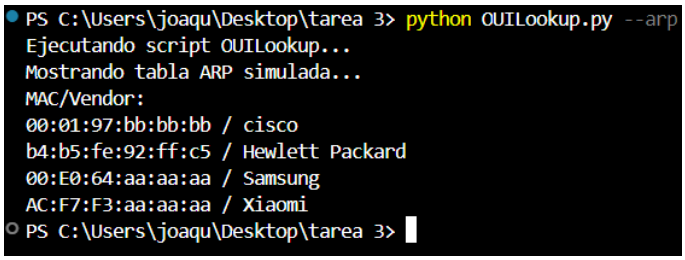


Figura 3. Prueba con comando --arp, tabla arp

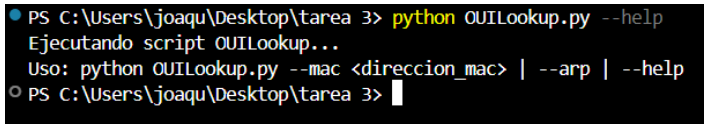


Figura 3. Prueba con comando --help

5. Discusión y conclusiones

Las direcciones MAC aleatorias son una función de privacidad utilizada por dispositivos modernos para evitar el rastreo del usuario a través de redes Wi-Fi. En lugar de transmitir su dirección MAC fija, que es única para cada dispositivo, estos sistemas generan direcciones MAC temporales y aleatorias cuando escanean redes Wi-Fi. Esto ayuda a proteger la identidad y ubicación del dispositivo, dificultando que observadores externos lo rastreen constantemente.

Apple implementa esta función en sus dispositivos iOS, iPadOS, y watchOS, generando direcciones aleatorias cuando los dispositivos no están conectados a una red específica. Esto también se aplica durante el escaneo para redes preferidas y al utilizar servicios de ubicación, lo que mejora la privacidad del usuario (Apple, 2024).

En resumen, el informe presentó la implementación de un programa para consultar fabricantes de direcciones MAC y simular tablas ARP, destacando cómo se manejan las direcciones MAC aleatorias para proteger la privacidad del usuario en redes Wi-Fi. El uso de estas direcciones aleatorias impide el rastreo del dispositivo, mejorando la seguridad. Las pruebas del programa confirmaron su funcionalidad y contribuyeron a una comprensión práctica de esta tecnología y su aplicación en entornos reales.

6. Referencias

• Apple Support. (2024). *Wi-Fi privacy and MAC address randomization*. Recuperado de Apple Support

• Apple Support UK. (2024). *Use private Wi-Fi addresses on Apple devices*. Recuperado de Apple Support UK

7. Ejemplo de sección

```python
def buscar_fabricante(mac):
    url = f'https://api.maclookup.app/v2/macs/{mac}'
    try:
        inicio = time.time()
        respuesta = requests.get(url)
        tiempo_respuesta = int((time.time() - inicio) * 1000)

        if respuesta.status_code == 200:
            datos = respuesta.json()
            fabricante = datos.get('company', 'Not found')
            if not fabricante:
                fabricante = 'Not found'
            print(f'MAC address : {mac}')
            print(f'Fabricante      : {fabricante}')
            print(f'Tiempo de respuesta {tiempo_respuesta}ms')
        else:
            print(f'MAC address : {mac}')
            print('Fabricante     : Not found')
            print(f'Tiempo de respuesta {tiempo_respuesta}ms')
    except Exception as e:
        print(f"Error: {e}")
```

Figura 1. Esta parte es clave para realizar la consulta del fabricante de una dirección MAC. Verifica el estado de la respuesta y gestiona posibles errores.

```python
def mostrar_tabla_arp():
    tabla_arp = {
        '00:01:97:bb:bb:bb': 'cisco',
        'b4:b5:fe:92:ff:c5': 'Hewlett Packard',
        '00:E0:64:aa:aa:aa': 'Samsung',
        'AC:F7:F3:aa:aa:aa': 'Xiaomi'
    }
    print("MAC/Vendor:")
    for mac, fabricante in tabla_arp.items():
        print(f'{mac} / {fabricante}')
```

Figura 2. Aquí se simula una tabla ARP con direcciones MAC y sus fabricantes, utilizando un diccionario para almacenar y mostrar la información.

```python
def main():
    argumentos = sys.argv[1:]
    opciones_largas = ["mac=", "arp", "help"]

    try:
        opts, args = getopt.getopt(argumentos, "", opciones_largas)
    except getopt.GetoptError:
        print('Uso: python OUILookup.py --mac <direccion_mac> | --arp | --help')
        sys.exit(2)

    for opt, arg in opts:
        if opt == "--mac":
            buscar_fabricante(arg)
        elif opt == "--arp":
            mostrar_tabla_arp()
        elif opt == "--help":
            print('Uso: python OUILookup.py --mac <direccion_mac> | --arp | --help')
            sys.exit()
```

Figura 3. Esta parte gestiona los argumentos de entrada para determinar qué acción realizar, llamando a la función correspondiente según la opción seleccionada (--mac, --arp o --help).