Tarea 2

Nikolas Lagos Moreno, nikolas.lagos@alumnos.uv.cl

1. Introducción

Los dispositivos de red y la información asociada son cruciales en un mundo tecnológico actual. Este informe presenta el desarrollo del proyecto OUILookup, una herramienta diseñada para consultar información sobre fabricantes a partir de direcciones MAC. La implementación se centra en el uso de una API pública, permitiendo a los usuarios obtener datos relevantes de manera eficiente. Se detallan los procesos de diseño, implementación y la documentación de las funciones utilizadas, junto con un diagrama de flujo que ilustra la estructura del programa.

2. Descripción del problema y diseño de la solución

Hoy en día, tener acceso a la información sobre los dispositivos de red es clave. Las direcciones MAC son importantes para identificar dispositivos, pero a menudo es difícil encontrar datos sobre sus fabricantes de manera rápida y sencilla. Para resolver este problema, se creó OUILookup, una herramienta que permite consultar fabricantes usando direcciones MAC a través de una API pública. La aplicación está diseñada para ser fácil de usar y eficiente, permitiendo que los usuarios obtengan información relevante con solo ingresar una dirección MAC.

3. Desarrollo

Para realizar este trabajo, se utilizó el sistema operativo Windows, junto con Python como lenguaje de programación. Se emplearon las bibliotecas requests para interactuar con una API pública, subprocess para ejecutar comandos del sistema y obtener la tabla ARP, y getopt para procesar los argumentos de línea de comandos.

3.1. Diagrama de flujo de la solución

Preparación para el main y resultado total. **Figura1**

Flujo total del código incluyendo todas las funciones utilizadas dependiendo del valor ingresado**. Figura2**

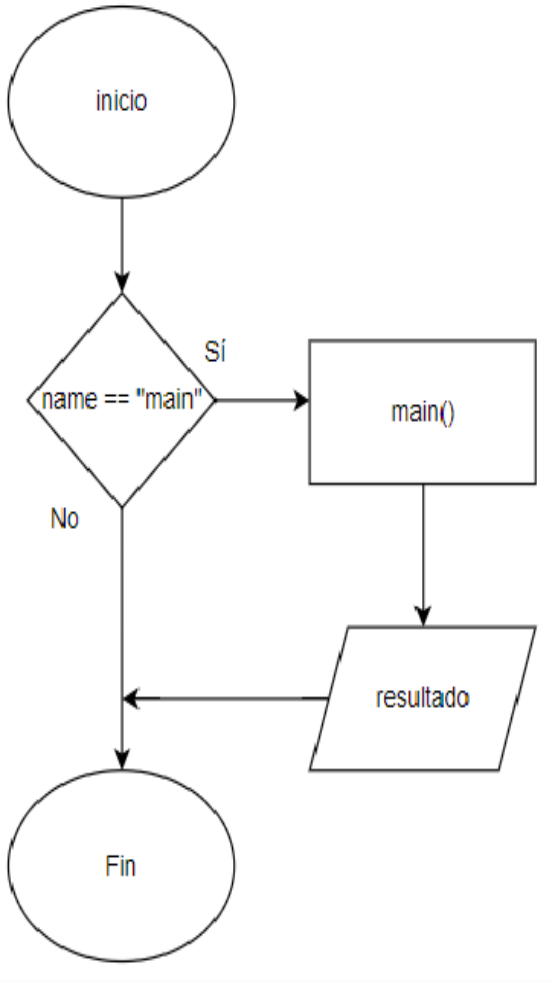

Figura1

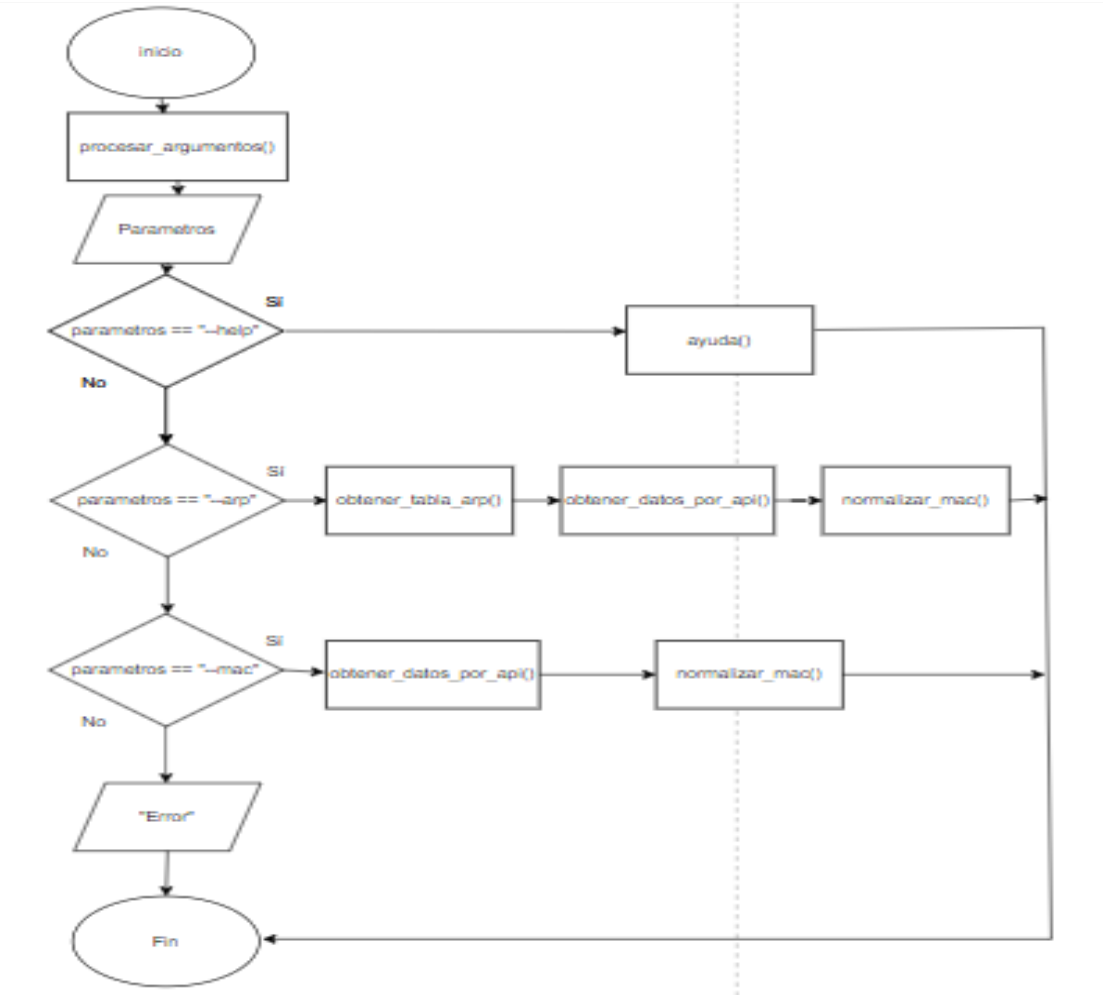

Figura2

3.2. Documentación de funciones

Descripción clara del propósito y funcionamiento de cada función.

3.2.1 procesar argumentos ()

Esta función procesa los argumentos ingresados en la línea de comandos. Utiliza la biblioteca getopt para identificar las opciones proporcionadas por el usuario y devuelve una lista de estos argumentos. **Figura3**

```
# Función para procesar la lógica de acuerdo a los argumentos pasados
def procesar_argumentos(argv):
    try:
        opts, args = getopt.getopt(argv, "m:ah", ["mac=", "arp", "help"])
        return opts
    except getopt.GetoptError:
        return None
```

Figura3

3.2.2 main ()

Es la función principal del programa. Dependiendo de los argumentos procesados, llama a las funciones adecuadas (ayuda, obtener_datos_por_api, obtener_tabla_arp) y muestra el resultado en la terminal. **Figura4**

```python
# Función principal que organiza la lógica
def main(argv):
    parametros = procesar_argumentos(argv)

    if parametros is None or len(parametros) == 0:
        return ayuda()

    resultados = []
    for opt, arg in parametros:
        if opt in ('--help', "-h"):
            resultados.append(ayuda())
        elif opt in ("-m", "--mac"):
            if not arg:
                resultados.append("La opción --mac no puede estar vacía.")
            else:
                vendor, tiempo = obtener_datos_por_api(arg)
                resultados.append(f"MAC Address : {arg}\nFabricante : {vendor}\nTiempo de respuesta: {tiempo}ms")
        elif opt in ("-a", "--arp"):
            diccionario_arp = obtener_tabla_arp()
            if diccionario_arp:
                resultados.append("IP\t/\tMAC\t/\t/Vendor")
                for ip, mac in diccionario_arp.items():
                    vendor, tiempo = obtener_datos_por_api(mac)  # Consulta la API por cada MAC
                    resultados.append(f"{ip}\t {mac}, {vendor}")  # Muestra la IP, la MAC y el fabricante
            else:
                resultados.append("Error al obtener la tabla ARP.")

    return "\n".join(resultados)
```

Figura4

3.2.3 obtener_datos_por_api ()

Recibe una dirección MAC, la normaliza usando normalizar_mac, y luego consulta la API pública para obtener el fabricante asociado a la MAC. Devuelve el nombre del fabricante y el tiempo que tomó la consulta. **Figura5**

```python
# Función pura que consulta la API por dirección MAC y devuelve el fabricante y el tiempo de respuesta
def obtener_datos_por_api(mac):
    mac = normalizar_mac(mac)  # Normalizamos la MAC aquí
    url = f"https://api.maclookup.app/v2/macs/{mac}/company/name?apiKey=01j9cmzmy3702pjpwyvdr54vcz01j9cnb7ta4aj7cvfzmqcm93ajaw63m3hknjtl"
    inicio = time.time()
    response = requests.get(url)
    fin = time.time()

    if response.status_code == 200:
        if response.text in ["*NO COMPANY*", "*PRIVATE*"]:
            return "Not found", round((fin - inicio) * 1000)  # Simula el "Not found" de la tarea
        else:
            return response.text.strip(), round((fin - inicio) * 1000)  # Simula el nombre del fabricante
    else:
        return None, round((fin - inicio) * 1000)
```

Figura5

3.2.4 normalizar_mac ()

Normaliza la dirección MAC ingresada, reemplazando guiones por dos puntos y añadiendo ceros a las partes faltantes, si es necesario, para asegurar el formato estándar. **Figura6**

```python
# Función que normaliza la MAC, añadiendo ceros si es necesario y cambiando guiones a dos puntos
def normalizar_mac(mac):
    # Convertir guiones a dos puntos
    mac = mac.replace("-", ":").lower()

    # Si la MAC es corta (como 9c:a5:13), añadimos ceros en las partes faltantes
    partes = mac.split(":")
    if len(partes) < 6:
        while len(partes) < 6:
            partes.append("00")
        mac = ":".join(partes)

    return mac
```

Figura6

3.2.5 obtener_tabla_arp ()

Obtiene la tabla ARP del sistema mediante el comando arp -a. Extrae las direcciones IP y MAC de los dispositivos conectados y devuelve un diccionario con los resultados. **Figura7**

```python
# Función pura que obtiene la tabla ARP y devuelve un diccionario con las IPs y sus MACs
def obtener_tabla_arp():
    dicc = {}
    try:
        tabla_arp = subprocess.check_output(['arp', '-a'], universal_newlines=True)
        arp_itt = tabla_arp.split('\n')
        for linea in arp_itt:
            if 'Internet Address' in linea or 'Interfaz' in linea or 'Dirección' in linea or 'Direccion' in linea or 'Interface' in linea:
                continue
            if linea.strip():
                parts = linea.split()
                if len(parts) >= 2:
                    cortado = parts[1].replace("-", ":").upper()  # Normaliza el formato de la MAC
                    dicc[parts[0]] = cortado  # IP: MAC
        return dicc
    except subprocess.CalledProcessError as e:
        return None  # Devuelve None si hay un error
    except Exception as e:
        return None
```

Figura7

3.2.6 Ayuda ()

Proporciona un mensaje de ayuda con las opciones disponibles para el usuario. Esta función se ejecuta si el usuario utiliza la opción --help o no proporciona los argumentos correctos. **Figura8**

```python
# Función pura para mostrar la ayuda
def ayuda():
    return (
        "Use: python OUILookup.py --mac <MAC> | --arp | [--help]\n"
        "--mac: MAC a consultar. P.e. aa:bb:cc:00:00:00.\n"
        "--arp: muestra los fabricantes de los hosts disponibles en la tabla ARP.\n"
        "--help: muestra este mensaje y termina.\n"
    )
```

Figura8

3.2.7 Código fuera de funciones

El bloque final se encarga de ejecutar el programa solo si se corre directamente desde la terminal. Se verifica con if __name__ == "__main__":, lo que asegura que el programa procese los argumentos y muestre los resultados cuando se ejecute el archivo. **Figura9**

```
# Esta parte solo se ejecuta si el archivo es ejecutado directamente
if __name__ == "__main__":
    resultado = main(sys.argv[1:])  # Pasamos solo los argumentos relevantes, sin el nombre del script
    print(resultado)
```

Figura9

4. Mac aleatorias

Las direcciones MAC aleatorias son un mecanismo utilizado por dispositivos electrónicos para mejorar la privacidad y la seguridad en redes. A diferencia de las direcciones MAC fijas, que son únicas para cada dispositivo y se utilizan para la identificación en la red, las direcciones aleatorias permiten que un dispositivo cambie su identificador de red en cada conexión, dificultando así el rastreo del usuario**.[1]**

Este enfoque es especialmente común en dispositivos móviles y portátiles, donde las redes Wi-Fi pueden almacenar y rastrear las direcciones MAC de los dispositivos que se conectan a ellas. Al usar direcciones aleatorias, los usuarios pueden evitar ser identificados de manera persistente, lo que protege su privacidad**.[2]**

Por ejemplo, en dispositivos que funcionan con sistemas operativos como Android y iOS, se generan direcciones MAC aleatorias al escanear redes Wi-Fi, lo que significa que cada vez que el dispositivo intenta conectarse, presenta una dirección diferente**.[3]**

5. Discusión y conclusiones

En conclusión, el proyecto OUILookup cumplió con el objetivo de facilitar la consulta de fabricantes a partir de direcciones MAC. La herramienta, construida sobre una API pública, permite obtener información de manera rápida y sencilla. Además, se documentaron las funciones y se generó un diagrama de flujo que refleja su funcionamiento. La exploración de las direcciones MAC aleatorias también destacó su relevancia en la privacidad de los usuarios. Estos elementos demuestran la importancia de contar con soluciones efectivas en el ámbito tecnológico actual.

6. Referencias

[1] https://www.rinconperdicion.com/mac-aleatoria-en-android-10/

[2] https://www.redeszone.net/noticias/wifi/direccion-mac-aleatoria-wifi/

[3] https://es.moyens.net/como/que-es-la-aleatorizacion-de-mac-y-como-usarla-en-sus-dispositivos/