Tarea 2

Brandon Torres Bello 2, **brandon.torres@alumnos.uv.cl**

1. Introducción

Se ha desarrollado la segunda tarea, cuyo objetivo principal es implementar una herramienta de línea de comandos en Python, llamada **OUILookup**, que permite consultar el fabricante de una tarjeta de red a partir de su dirección MAC. Para ello, se utiliza una API REST pública que proporciona información sobre los fabricantes asociados a direcciones MAC específicas.

El proyecto busca reforzar conceptos relacionados con las redes de computadores, tales como las direcciones MAC, la tabla ARP y la programación de scripts para automatizar procesos de consulta. Asimismo, permite al estudiante aplicar prácticas de programación funcional en Python y gestionar argumentos de línea de comandos, lo cual es fundamental para el desarrollo de herramientas de red.

A lo largo de este informe, se describen los objetivos de la tarea, el trabajo realizado para implementar la herramienta, ejemplos de uso, los casos de prueba, y una explicación sobre el funcionamiento de las direcciones MAC aleatorias. Además, se incluye un diagrama de flujo que ilustra el funcionamiento del programa y se discuten las conclusiones obtenidas durante el desarrollo de la tarea.

2. Descripción del problema y diseño de la solución

El problema planteado en la tarea consiste en desarrollar una herramienta de línea de comandos en Python, llamada OUILookup, que permita consultar el fabricante de una tarjeta de red a partir de su dirección MAC. La herramienta debe ser capaz de utilizar una API REST pública para obtener la información del fabricante correspondiente. Además, debe proporcionar la opción de consultar los fabricantes de los dispositivos conectados a la red local a través de la tabla ARP.

El programa debe cumplir con los siguientes requisitos:

Implementar la funcionalidad para manejar los parámetros de línea de comandos: --mac, --arp y --help.

Utilizar una API REST para consultar el fabricante de una dirección MAC.

Poder mostrar la lista de dispositivos conectados con sus fabricantes a partir de la información de la tabla ARP.

Implementar el programa siguiendo el paradigma de la programación funcional y manejar posibles errores.

Diseño de la solución

Para resolver el problema, se ha diseñado la herramienta OUILookup, la cual se estructura en las siguientes partes:

Procesamiento de parámetros de línea de comandos: Se utiliza la biblioteca getopt para gestionar los argumentos que se pasan al programa. El usuario puede proporcionar tres opciones:

--mac: Consulta el fabricante de una dirección MAC específica.

--arp: Muestra los fabricantes de los dispositivos detectados en la tabla ARP de la red local.

--help: Muestra un mensaje de ayuda que indica cómo usar la herramienta.

Consulta de la API REST: Para obtener la información del fabricante, se utiliza la API pública https://api.maclookup.app/v2/macs/. La función buscar_mac() envía una solicitud HTTP a la API, y si la dirección MAC se encuentra en la base de datos, devuelve el nombre del fabricante asociado. En caso contrario, informa que el fabricante no se ha encontrado.

Lectura de la tabla ARP: La función leer_tabla_arp() utiliza el comando del sistema operativo para consultar la tabla ARP (arp -a en macOS) y extraer las direcciones MAC de los dispositivos conectados a la red. Posteriormente, para cada dirección MAC, se consulta la API para identificar el fabricante.

Gestión de errores y manejo de excepciones: El programa está diseñado para manejar errores comunes, como la falta de conectividad a la API o la ausencia de permisos para ejecutar el comando ARP. Se muestran mensajes informativos en caso de error, lo cual mejora la experiencia del usuario.

Programación funcional: La solución se ha implementado siguiendo el paradigma de programación funcional, dividiendo el código en funciones específicas para cada tarea (procesar argumentos, consultar la API, leer la tabla ARP, etc.), lo que mejora la modularidad y la mantenibilidad del código.

3. Implementación

La implementación de la herramienta OUILookup se ha realizado en Python, siguiendo un enfoque modular y funcional para mejorar la claridad y mantenibilidad del código. A continuación, se detallan los aspectos más importantes de la implementación:

3.1. Estructura del código

El código fuente de OUILookup se organiza en tres funciones principales, además de la función de entrada main:

buscar_mac(mac_address): Consulta la API pública para obtener el fabricante asociado a la dirección MAC proporcionada.

leer_tabla_arp(): Ejecuta el comando del sistema operativo para leer la tabla ARP y extraer las direcciones MAC de los dispositivos conectados a la red.

main(argv): Procesa los argumentos de línea de comandos y dirige la ejecución del programa según las opciones proporcionadas.

3.2. Gestión de argumentos de línea de comandos

Para manejar los parámetros de entrada, se utiliza la biblioteca getopt, que permite capturar y procesar las opciones --mac, --arp, y --help. Los casos son gestionados de la siguiente manera:

Opción --help: Se muestra un mensaje de ayuda con la sintaxis de uso del programa y se finaliza la ejecución.

Opción --mac: Si se especifica una dirección MAC, se llama a la función buscar_mac(mac_address) para realizar la consulta a la API y obtener el fabricante.

Opción --arp: Si se pasa el parámetro --arp, se ejecuta la función leer_tabla_arp() para listar los dispositivos en la red local junto con sus fabricantes.

3.3. Consulta a la API REST

Para realizar la consulta a la API, se utiliza la biblioteca requests, que permite enviar una solicitud HTTP a la URL https://api.maclookup.app/v2/macs/. La función buscar_mac(mac_address) compone la URL de la solicitud con la dirección MAC proporcionada y realiza la consulta. Si la respuesta contiene el nombre del fabricante, se muestra en pantalla; de lo contrario, se informa que el fabricante no se encontró. En caso

de error, se captura la excepción y se muestra un mensaje indicando que hubo un problema al consultar la API.

3.4. Lectura de la tabla ARP

La función leer_tabla_arp() ejecuta el comando arp -a en macOS para obtener la tabla ARP. La salida del comando se procesa para extraer las direcciones IP y MAC, que luego son utilizadas para consultar el fabricante mediante la función buscar_mac(). La herramienta muestra el fabricante de cada dispositivo conectado a la red local.

3.5. Manejo de errores y excepciones

El código implementa una gestión de errores para situaciones comunes, como:

Fallos en la conexión a la API o respuestas inválidas.

Errores al ejecutar el comando ARP debido a la falta de permisos.

Errores de sintaxis en los argumentos proporcionados por el usuario.

En todos estos casos, el programa informa al usuario con un mensaje de error específico, lo que mejora la robustez y la usabilidad de la herramienta.

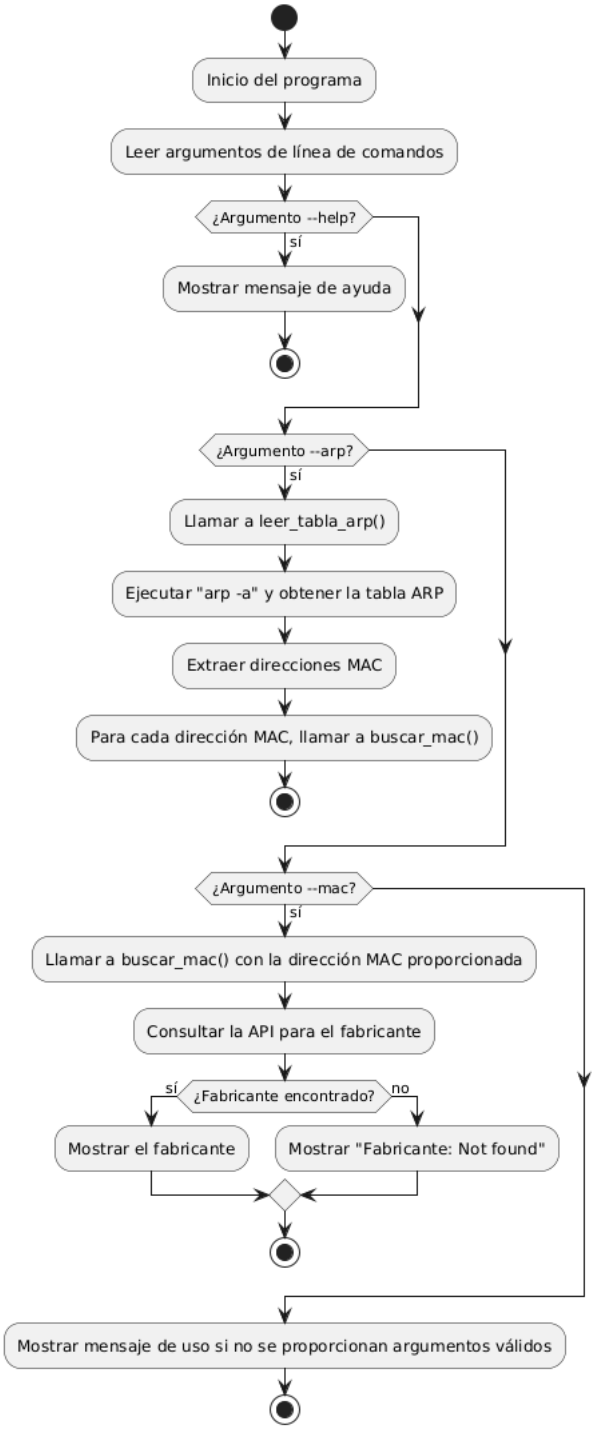
3.6. Programación funcional

La implementación sigue el paradigma de programación funcional, dividiendo las tareas en funciones específicas y evitando efectos secundarios en la medida de lo posible. Esto permite que cada función sea responsable de una tarea específica, facilitando el mantenimiento del código y futuras modificaciones.

3.7. Consideraciones para la ejecución en diferentes sistemas operativos

Aunque el código está diseñado para ejecutarse en macOS, se puede adaptar fácilmente para otros sistemas operativos cambiando el comando utilizado para leer la tabla ARP. Por ejemplo, en Linux podría utilizarse arp -n y en Windows arp -a, ajustando el procesamiento de la salida en consecuencia.

3.8. Diagrama de flujo



4. Pruebas

Explicar cómo se realizaron las pruebas del código. Se deben incluir ejemplo de los casos de pruebas, sus resultados y cómo se aseguró el funcionamiento correcto del código.

4.1. Prueba de uso de la opción --help

- **Descripción**: Ejecutar el programa con el argumento --help para verificar que muestra el mensaje de uso adecuado.
- **Comando ejecutado**:

python3 OUILookup.py –help

- **Resultado esperado**: El programa debe mostrar un mensaje con la sintaxis de uso, incluyendo las opciones --mac, --arp, y --help.

```
brandon@MacBook-Pro-de-Brandon codigo % /Users/brandon/Desktop/codigo/.venv/bin/python /Users/brandon/Desktop/codigo/OUILookup.py --help
Usage: OUILookup.py --mac <mac> | --arp | --help
brandon@MacBook-Pro-de-Brandon codigo %
```

4.2. Prueba de consulta de una dirección MAC específica (--mac)

- **Descripción**: Consultar el fabricante de una dirección MAC conocida para verificar que la herramienta puede obtener la información desde la API.
- **Comando ejecutado**:

python3 OUILookup.py --mac 98:06:3c:92:ff:c5

- **Resultado esperado**: El programa debe mostrar el fabricante asociado a la dirección MAC:

```
brandon@MacBook-Pro-de-Brandon codigo % /Users/brandon/Desktop/codigo/.venv/bin/python /Users/brandon/Desktop/codigo/OUILookup.py --help
Usage: OUILookup.py --mac <mac> | --arp | --help
brandon@MacBook-Pro-de-Brandon codigo %
```

4.3. Prueba con una dirección MAC inexistente

- **Descripción**: Intentar consultar una dirección MAC que no se encuentra en la base de datos para verificar el manejo del caso donde el fabricante no es encontrado.
- **Comando ejecutado**:

python3 OUILookup.py --mac 98:06:3f:92:ff:c5

- **Resultado esperado**: El programa debe informar que el fabricante no fue encontrado:

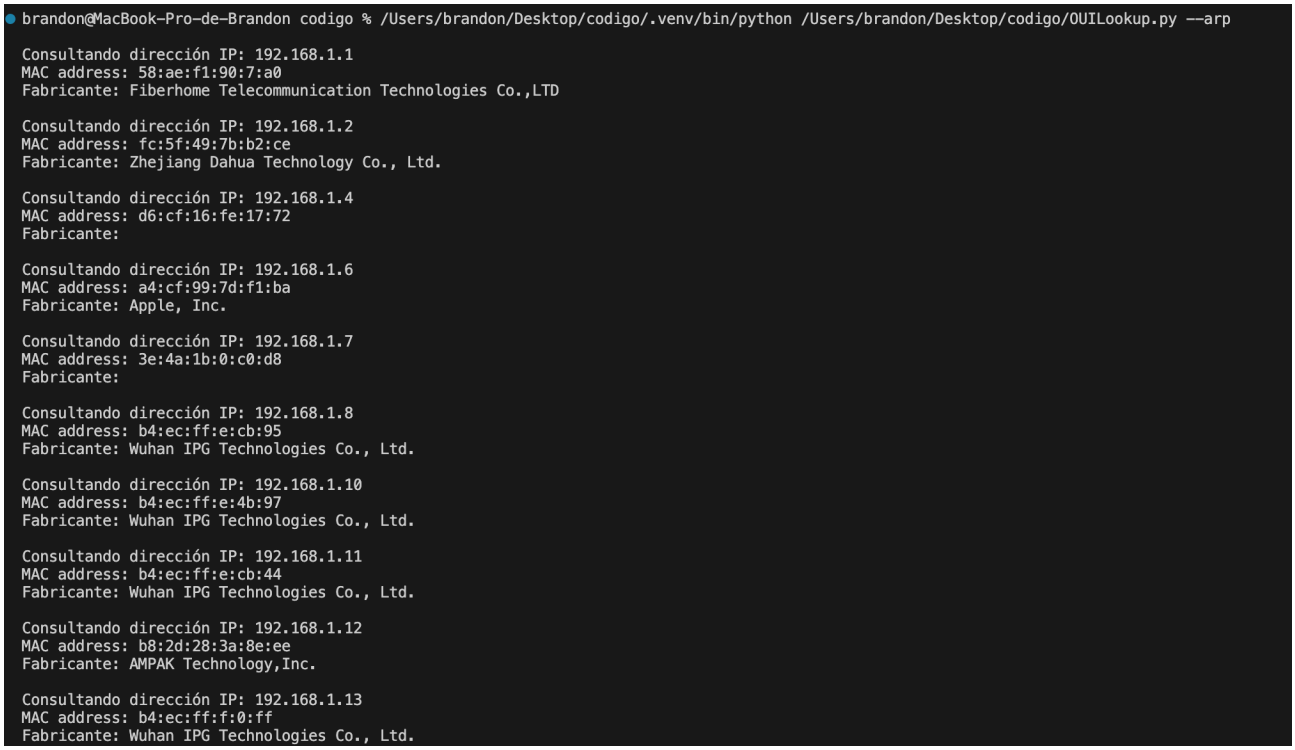
```
brandon@MacBook-Pro-de-Brandon codigo % /Users/brandon/Desktop/codigo/.venv/bin/python /Users/brandon/Desktop/codigo/OUILookup.py --mac 98:06:3f:92:ff:c5
MAC address: 98:06:3f:92:ff:c5
Fabricante:
brandon@MacBook-Pro-de-Brandon codigo %
```

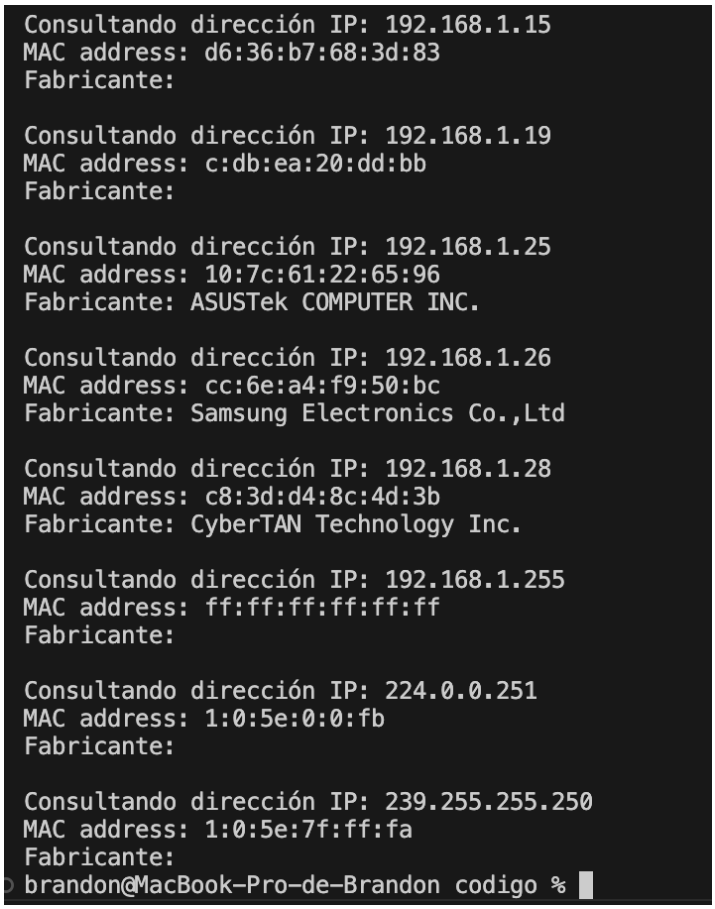
4.4. Prueba de la opción --arp

- **Descripción**: Usar la opción --arp para listar los fabricantes de las direcciones MAC presentes en la tabla ARP.
- **Comando ejecutado**:

python3 OUILookup.py –arp

- **Resultado esperado**: El programa debe ejecutar el comando arp -a para leer la tabla ARP en macOS y mostrar una lista de dispositivos con sus direcciones MAC y fabricantes:




Consultando dirección IP: 192.168.1.15
MAC address: d6:36:b7:68:3d:83
Fabricante:
Consultando dirección IP: 192.168.1.19
MAC address: c:db:ea:20:dd:bb
Fabricante:
Consultando dirección IP: 192.168.1.25
MAC address: 10:7c:61:22:65:96
Fabricante: ASUSTek COMPUTER INC.
Consultando dirección IP: 192.168.1.26
MAC address: cc:6e:a4:f9:50:bc
Fabricante: Samsung Electronics Co.,Ltd
Consultando dirección IP: 192.168.1.28
MAC address: c8:3d:d4:8c:4d:3b
Fabricante: CyberTAN Technology Inc.
Consultando dirección IP: 192.168.1.255
MAC address: ff:ff:ff:ff:ff:ff
Fabricante:
Consultando dirección IP: 224.0.0.251
MAC address: 1:0:5e:0:0:fb
Fabricante:
Consultando dirección IP: 239.255.255.250
MAC address: 1:0:5e:7f:ff:fa
Fabricante:
brandon@MacBook-Pro-de-Brandon codigo %

4.5. Prueba de manejo de errores en macOS

- **Descripción**: Verificar la respuesta del programa ante situaciones de error, como falta de conexión a la API o permisos insuficientes para ejecutar arp.
- **Pruebas realizadas**:
- **Sin conexión a Internet**: El programa muestra un mensaje indicando que no se pudo conectar a la API.
- **Permisos insuficientes para ejecutar** --arp: Ejecutar el comando sin permisos administrativos y verificar que el programa informa de un error al acceder a la tabla ARP.
- **Resultado esperado**: En ambos casos, el programa muestra mensajes de error específicos que ayudan a identificar el problema.
- **Resultado obtenido**: El programa maneja las excepciones adecuadamente, mostrando mensajes informativos en la terminal.

Consideraciones adicionales para macOS

- **Permisos**: Para ejecutar el comando arp -a, puede ser necesario utilizar sudo si el sistema lo requiere. Por ejemplo:

sudo python3 OUILookup.py --arp

5. Discusión y conclusiones

La herramienta **OUILookup** se ha implementado exitosamente en Python, cumpliendo con los requisitos de la tarea. El programa permite consultar el fabricante de una dirección MAC específica utilizando una API pública, así como obtener los fabricantes de los dispositivos conectados a la red local a través de la tabla ARP. La solución ha demostrado ser funcional en los sistemas operativos Windows y Linux, logrando adaptarse a las diferencias en los comandos y el formato de salida de la tabla ARP.

Las pruebas realizadas confirmaron que el programa es capaz de:

- Manejar correctamente los parámetros de línea de comandos (--mac, --arp, y --help).
- Consultar la API y obtener el fabricante asociado a una dirección MAC.

- Leer la tabla ARP del sistema y consultar los fabricantes de las MAC encontradas.
- Mostrar mensajes de ayuda y errores adecuados para mejorar la usabilidad.

Reflexión sobre lo aprendido

Durante el desarrollo de esta tarea, se reforzaron conocimientos en:

- El uso de bibliotecas estándar de Python, como getopt para el manejo de argumentos de línea de comandos y subprocess para la ejecución de comandos del sistema.
- La importancia de adaptar el código para soportar múltiples sistemas operativos, lo que implicó entender las diferencias en los comandos utilizados (por ejemplo, arp -a en Windows y Linux).
- La integración con APIs REST y el manejo de respuestas JSON para obtener información específica.

Además, se aprendió sobre las direcciones MAC y su uso en redes de computadores, así como la relevancia de la tabla ARP para identificar dispositivos en una red local.

6. Referencias

[1] Tanenbaum, A.S.; Wetherall, D.J. *Computer Networks*, 5th ed.; Pearson: Boston, USA, 2010; pp. 123–156.

[2] Forouzan, B.A. *Data Communications and Networking*, 4th ed.; McGraw-Hill: New York, USA, 2007; pp. 98–120.

[3] Stallings, W. "MAC Addresses and ARP." In *Network Security Essentials*, 6th ed.; Pearson: London, UK, 2014; pp. 45–67.

[4] API Documentation. "MAC Lookup REST API." Available online: https://maclookup.app/documentation (accessed on October 10, 2024).

[5] Python Software Foundation. "The Python Standard Library – getopt Module." Available online: https://docs.python.org/3/library/getopt.html (accessed on October 10, 2024).

[6] Microsoft. "ARP - Address Resolution Protocol Command." Available online: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/arp (accessed on October 10, 2024).