

Consultar Fabricantes de tarjetas de red a través de una API en Python

Walter Castillo Venegas, walter.castillo@alumnos.uv.cl

Juan Barraza Riquelme, juan.barraza@alumnos.uv.cl

1. Introducción

En un mundo cada vez más interconectado, el análisis y manejo de redes es crucial para garantizar la seguridad y eficiencia en las comunicaciones digitales. Uno de los componentes fundamentales en este proceso es la identificación de los dispositivos conectados a una red, lo cual se realiza, entre otras maneras, mediante el reconocimiento de direcciones MAC (Media Access Control). Cada dispositivo en una red tiene una dirección MAC única, y conocer el fabricante o proveedor asociado a estas direcciones puede ser útil para diversas aplicaciones, desde auditorías de seguridad hasta optimizaciones de redes.

El propósito de este trabajo es desarrollar una herramienta que permita identificar de manera automatizada los fabricantes de dispositivos conectados a una red mediante la consulta de direcciones MAC a través de una API externa y el análisis de la tabla ARP. Este enfoque facilita la obtención de información relevante sobre los dispositivos sin necesidad de realizar análisis manuales o inspecciones en profundidad.

El objetivo principal de este proyecto es ofrecer una solución automatizada que permita a los administradores de redes obtener de manera rápida y sencilla la información sobre los dispositivos conectados, mejorando así la eficiencia en la gestión de redes. Como resultado de esta investigación y desarrollo, se presenta una herramienta funcional que puede ser utilizada en entornos reales para la identificación y gestión de dispositivos en redes de distintos tamaños.

Las principales conclusiones de este trabajo muestran la viabilidad de integrar API externas para la consulta de direcciones MAC en herramientas de análisis de redes y cómo esto puede simplificar la administración y monitoreo de infraestructuras de comunicación.

2. Descripción del problema y diseño de la solución

El problema a resolver está relacionado con la necesidad de identificar los fabricantes de los dispositivos conectados a una red local a partir de sus direcciones MAC. En las redes, los administradores a menudo requieren conocer qué dispositivos están conectados y de qué proveedores son, ya que esto puede ser crucial para detectar posibles vulnerabilidades o gestionar eficientemente los recursos de la red.

El desafío radica en que las direcciones MAC, aunque son únicas, no proporcionan información explícita sobre el fabricante del dispositivo. Para obtener esta información, se debe consultar una base de datos o servicio que permita relacionar una dirección MAC con su fabricante. Esta tarea es tediosa si se realiza manualmente, especialmente en redes con muchos dispositivos conectados, lo que justifica la necesidad de automatizar el proceso.

Además de la consulta individual de una dirección MAC, es común que los administradores necesiten realizar un análisis completo de todos los dispositivos conectados a una red. Esto implica la lectura de la tabla ARP (Address Resolution Protocol), que contiene las asociaciones entre las direcciones IP y las direcciones MAC de los dispositivos activos en la red. Sin embargo, esta tabla no proporciona información sobre los fabricantes, por lo que es necesario un método que consulte de forma automática los fabricantes de todos los dispositivos listados en la tabla ARP.

Requerimientos y especificaciones de la tarea

1. Requerimientos funcionales:

- El sistema debe permitir la consulta del fabricante de un dispositivo dado su dirección MAC.
- El sistema debe ser capaz de obtener la tabla ARP del sistema y realizar la consulta de los fabricantes de las direcciones MAC que aparecen en dicha tabla.
- El sistema debe mostrar la IP, dirección MAC, fabricante y el tipo (dinámico o estático) de cada dispositivo listado en la tabla ARP.
- El sistema debe manejar errores de conexión a la API y mostrar mensajes adecuados en caso de fallos en las consultas.

2. Requerimientos no funcionales:

- La solución debe ser eficiente y rápida, reduciendo el tiempo de espera en las consultas a la API.
- El sistema debe ser portable y ejecutable en cualquier entorno que soporte Python.
- El sistema debe mostrar información clara y fácil de entender para el usuario.
- El tiempo de respuesta para la consulta de la API debe ser capturado y mostrado al usuario.

3. Especificaciones técnicas:

- El sistema utilizará una API externa (por ejemplo, <https://api.maclookup.app>) para realizar las consultas de los fabricantes de las direcciones MAC.
- El sistema debe ejecutar el comando `arp -a` en entornos Windows para obtener la tabla ARP.
- El tiempo de respuesta de la API será registrado para cada consulta.
- El sistema debe permitir el uso de argumentos desde la línea de comandos para especificar si se desea realizar una consulta de una dirección MAC en particular o analizar la tabla ARP completa.

Diseño de la solución

Para resolver este problema, se diseñó una solución modular compuesta por varias funciones que se encargan de las diferentes tareas, con una arquitectura simple y eficaz para gestionar tanto las consultas individuales como el análisis de la tabla ARP. El diseño está dividido en tres componentes principales:

1. **Interfaz de usuario:** Un conjunto de argumentos de línea de comandos que permiten al usuario especificar si desea realizar una consulta individual de una dirección MAC o analizar la tabla ARP completa. También incluye la opción de mostrar un mensaje de ayuda.
2. **Módulo de consulta de fabricantes:** Este módulo maneja las solicitudes HTTP hacia la API externa para consultar los fabricantes de las direcciones MAC. Incluye

manejo de errores, tiempo de respuesta y validación de las respuestas obtenidas de la API.

3. **Módulo de análisis de tabla ARP:** Este módulo ejecuta el comando `arp -a` en el sistema, procesa la salida para extraer las direcciones MAC e IP, y utiliza el módulo de consulta de fabricantes para obtener información sobre cada dispositivo en la red.

Modelo general de arquitectura

El modelo general de arquitectura (ver figura 1) de la solución se puede representar con el siguiente diagrama conceptual:

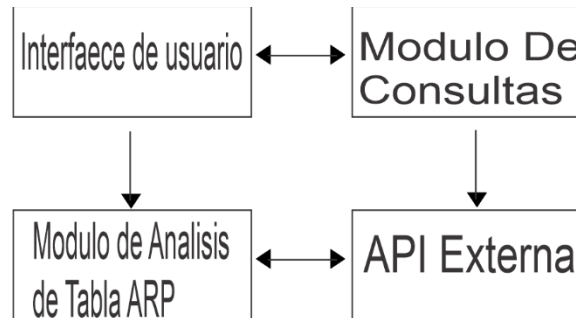


Figura 1. Modelo general de arquitectura

En este modelo:

- El **usuario** interactúa con el sistema a través de la línea de comandos.
- Dependiendo de los argumentos proporcionados, el sistema accede al **módulo de consultas** o al **módulo de análisis de la tabla ARP**.
- Si se analiza la tabla ARP, cada dirección MAC es consultada a través de la **API externa** para obtener información sobre el fabricante.

Diagrama de clase

A continuación se presenta un **diagrama de clase**(ver figura 2) simplificado que ilustra los componentes principales del sistema:

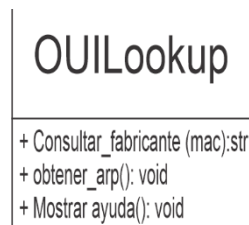


Figura 2. Diagrama de Clases

- `consultar_fabricante(mac)`: Realiza la consulta del fabricante de una dirección MAC específica utilizando la API.
- `obtener_arp()`: Ejecuta el comando `arp -a` y procesa la información para obtener las direcciones IP y MAC de los dispositivos en la red, y luego realiza consultas para cada MAC.
- `mostrar_ayuda()`: Muestra el mensaje de ayuda con las instrucciones de uso.

Este diseño modular y flexible permite ampliar la funcionalidad del sistema en el futuro, como agregar más opciones de visualización o integración con otras herramientas de análisis de redes.

3. Implementación

En esta sección se detallarán las partes más importantes del código implementado para resolver el problema descrito anteriormente. La implementación del sistema está diseñada para ser modular, lo que facilita su mantenimiento y expansión. A continuación, se explican los componentes clave del código, el flujo general del programa y los desafíos encontrados durante su desarrollo.

Componentes principales del código

1. **Función `consultar_fabricante (mac)`**: Esta función es responsable de realizar la consulta a la API externa para obtener información sobre el fabricante de una

dirección MAC específica. Es una función fundamental, ya que todas las consultas de fabricantes dependen de su correcta ejecución.

- **Entrada:** Una dirección MAC en formato estándar.
- **Salida:** El nombre del fabricante asociado a la dirección MAC o un mensaje de error si no se encuentra.
- **Detalles de implementación:**
 - La función construye la URL de la API utilizando la dirección MAC proporcionada como parámetro.
 - Utiliza el método `requests.get()` para hacer una petición HTTP.
 - Maneja errores comunes como tiempos de espera o fallos en la conexión mediante un bloque `try-except`.
 - Si la respuesta es exitosa, convierte el resultado a formato JSON y extrae el campo del fabricante ("company").
- **Desafíos:**
 - Durante el desarrollo, uno de los principales desafíos fue el manejo de tiempos de espera (timeouts) y fallos de conexión, especialmente cuando la red es lenta o la API no responde. Esto fue resuelto implementando un timeout explícito y manejando excepciones.
- **Función `obtener_arp()`:** Esta función se encarga de obtener la tabla ARP del sistema y procesar las direcciones MAC de los dispositivos conectados. A partir de esta tabla, realiza consultas automáticas a la API para obtener el fabricante de cada dispositivo.
- **Entrada:** No recibe entradas directas del usuario, sino que ejecuta el comando `arp -a` en el sistema para obtener la tabla ARP.
- **Salida:** Muestra en pantalla la IP, la dirección MAC, el fabricante y el tipo de conexión (dinámico o estático) para cada dispositivo en la red.
- **Detalles de implementación:**
 - Ejecuta el comando `arp -a` utilizando `subprocess.check_output()` para capturar la salida.
 - Procesa las líneas de salida para extraer las direcciones IP, MAC y el tipo de cada entrada.
 - Para cada dirección MAC encontrada, llama a `consultar_fabricante()` para obtener el fabricante.
 - Presenta los resultados en un formato tabular.

- Desafíos:

- Se encontró un problema con el formato de salida del comando `arp -a` en sistemas Windows debido a la codificación incorrecta de ciertos caracteres especiales, como "dinámico" y "estático". Esto fue resuelto reemplazando manualmente las palabras mal codificadas en la salida.
- Función `mostrar_ayuda()`: Esta es una función auxiliar que proporciona un mensaje de ayuda al usuario cuando no se pasan los parámetros correctos. Simplemente imprime las opciones disponibles y finaliza la ejecución del programa.

Diagrama de flujo

A continuación se presenta un diagrama de flujo(Ver figura 3) que ilustra el funcionamiento general del programa:

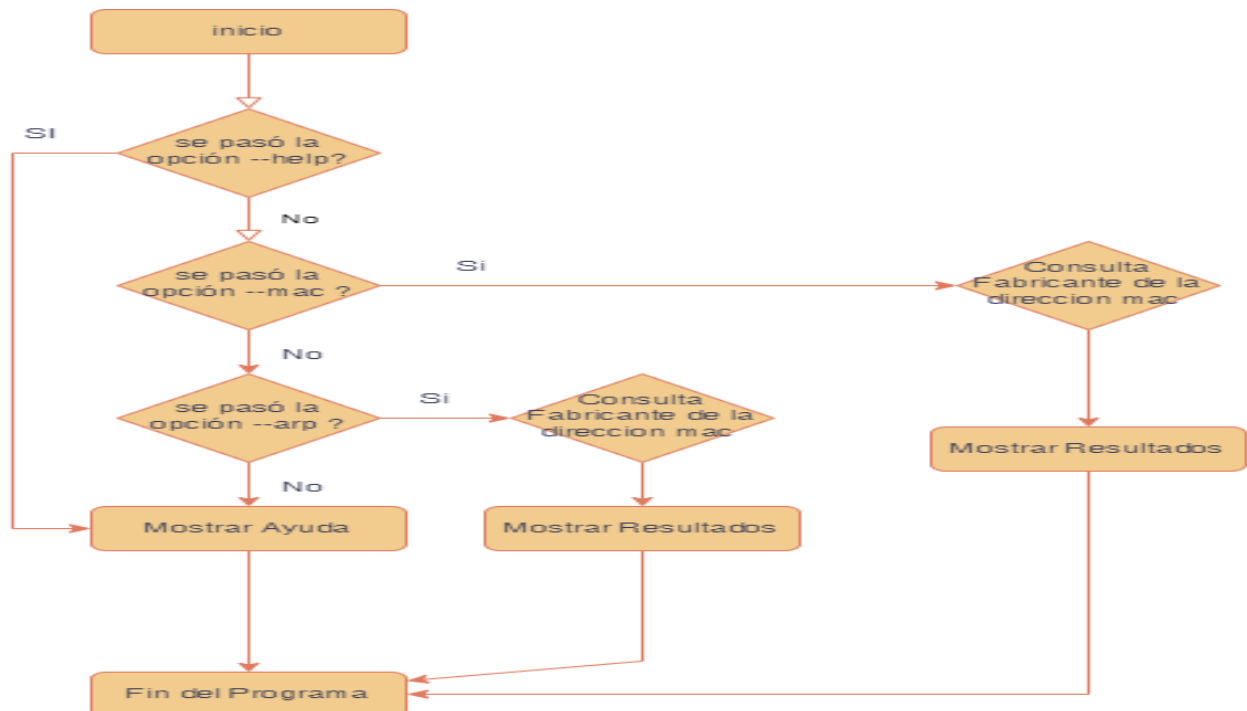


Figura 3. Diagrama de Flujo

Principales desafíos y soluciones

1. **Manejo de tiempos de espera en la API:** El sistema debe ser capaz de manejar posibles tiempos de espera o fallos en la conexión con la API externa. Esto fue resuelto implementando un timeout explícito en las solicitudes HTTP y capturando las excepciones correspondientes para evitar que el programa se bloquee.
2. **Procesamiento de la tabla ARP en Windows:** Durante el desarrollo, se encontró que la salida del comando `arp -a` en Windows tiene problemas de codificación con caracteres especiales como "dinámico" y "estático". Este problema fue solucionado reemplazando manualmente las palabras mal codificadas en la salida del comando antes de procesarla.
3. **Consultas simultáneas a la API:** Un posible desafío, especialmente en redes grandes, es el tiempo que toma consultar cada dirección MAC de manera secuencial. En el futuro, una mejora podría ser la implementación de consultas paralelas para acelerar este proceso.

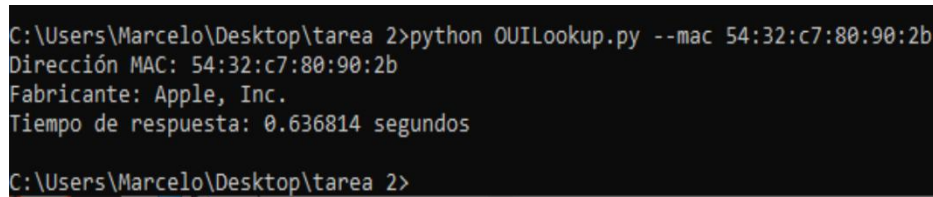
Con este enfoque modular, la implementación del sistema ha sido flexible, facilitando la adición de nuevas características o mejoras en el futuro.

4. Pruebas

Para garantizar el correcto funcionamiento del sistema, se realizaron diversas pruebas que cubren distintos aspectos del programa. Las pruebas se centraron en validar el comportamiento esperado tanto para la consulta de fabricantes de direcciones MAC individuales como para la extracción y análisis de la tabla ARP. A continuación, se describen los casos de prueba realizados, los resultados obtenidos y las medidas tomadas para asegurar la funcionalidad del sistema.

1. Prueba de consulta de dirección MAC válida

- **Descripción:** Se ingresó una dirección MAC (Ver Figura 4) válida utilizando el parámetro `--mac` para verificar si el programa devuelve el fabricante correcto.



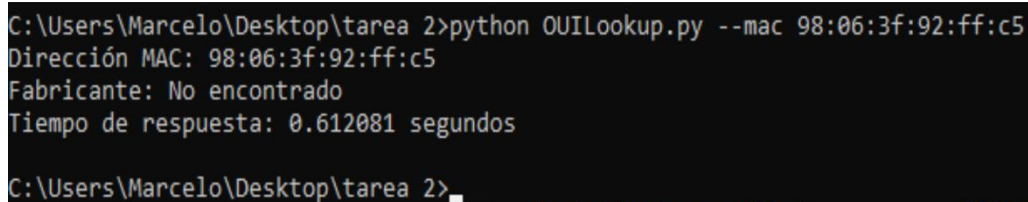
```
C:\Users\Marcelo\Desktop\tarea 2>python OUILookup.py --mac 54:32:c7:80:90:2b
Dirección MAC: 54:32:c7:80:90:2b
Fabricante: Apple, Inc.
Tiempo de respuesta: 0.636814 segundos
C:\Users\Marcelo\Desktop\tarea 2>
```

Figura 4 Prueba de consulta de dirección MAC válida

- **Conclusión:** La prueba fue exitosa, el programa mostró correctamente el fabricante de la dirección MAC.

2. Prueba de consulta de dirección MAC inválida

- **Descripción:** Se ingresó una dirección MAC inválida (Ver Figura 5) para comprobar que el programa maneja correctamente este tipo de entradas.



```
C:\Users\Marcelo\Desktop\tarea 2>python OUILookup.py --mac 98:06:3f:92:ff:c5
Dirección MAC: 98:06:3f:92:ff:c5
Fabricante: No encontrado
Tiempo de respuesta: 0.612081 segundos
C:\Users\Marcelo\Desktop\tarea 2>
```

Figura 5 Prueba de consulta de dirección MAC inválida

- **Conclusión:** La prueba fue exitosa, ya que el programa manejó correctamente la dirección MAC inválida y mostró un mensaje adecuado.

3. Prueba de tiempo de espera en la consulta a la API

- **Descripción:** Se simuló una situación de red lenta para verificar que el sistema maneja correctamente los tiempos de espera (timeouts) (Ver Figura 6) al intentar conectar con la API.
- **Modificación:** Se redujo el tiempo de espera en la función `consultar_fabricante()` a 0.1 segundo.

```
C:\Users\Marcelo\Desktop\tarea 2>python OUILookup.py --mac 98:06:3f:92:ff:c5
Dirección MAC: 98:06:3f:92:ff:c5
Fabricante: Error: Timeout
Tiempo de respuesta: 0 segundos

C:\Users\Marcelo\Desktop\tarea 2>
```

Figura 6 Prueba de consulta de dirección MAC válida

- **Conclusión:** La prueba fue exitosa, ya que el sistema detectó y manejó el tiempo de espera de manera adecuada.

Modificación: Se redujo el tiempo de espera en la función `consultar_fabricante()` a 1 segundo.

4. Prueba de obtención y análisis de la tabla ARP

4.1 **Descripción:** Se ejecutó el programa con la opción `-arp` (Ver Figura 7) para verificar que puede obtener la tabla ARP, procesarla, y realizar consultas automáticas de los fabricantes de las direcciones MAC encontradas.

```
C:\Users\Marcelo\Desktop\tarea 2>python OUILookup.py --arp
IP/MAC/Fabricante/Tipo:
Interfaz: / 192.168.211.254 / / 0x8
192.168.211.105 / 00:22:6b:ec:cc:c1 / Cisco-Linksys, LLC / dinámico
192.168.211.106 / 00:0c:29:ad:b3:ad / VMware, Inc. / dinámico
192.168.211.248 / 00:0c:29:7d:66:16 / VMware, Inc. / dinámico
192.168.211.255 / ff:ff:ff:ff:ff:ff / No encontrado / estático
224.0.0.22 / 01:00:5e:00:00:16 / / estático
224.0.0.251 / 01:00:5e:00:00:fb / / estático
224.0.0.252 / 01:00:5e:00:00:fc / No encontrado / estático
239.255.255.250 / 01:00:5e:7f:ff:fa / / estático
Interfaz: / 192.168.0.93 / / 0x9
192.168.0.1 / 18:35:d1:7d:42:c8 / ARRIS Group, Inc. / dinámico
192.168.0.96 / 10:60:4b:10:3b:dd / Hewlett Packard / dinámico
192.168.0.252 / 00:00:ca:01:02:03 / No encontrado / dinámico
192.168.0.255 / ff:ff:ff:ff:ff:ff / / estático
224.0.0.22 / 01:00:5e:00:00:16 / / estático
224.0.0.251 / 01:00:5e:00:00:fb / No encontrado / estático
224.0.0.252 / 01:00:5e:00:00:fc / / estático
239.255.255.250 / 01:00:5e:7f:ff:fa / / estático

C:\Users\Marcelo\Desktop\tarea 2>
```

Figura 7 Prueba de consulta de dirección MAC válida

4.2 Validación de los resultados

4.2.1 Comparación manual con datos de la API: Para asegurar que los resultados de la función `consultar_fabricante()` eran correctos, se verificaron manualmente las direcciones MAC consultadas contra la API de MACLookup. Se comprobó que los resultados obtenidos por el programa coincidían con los resultados esperados.

4.2.2 Verificación de tiempo de respuesta: En las pruebas donde se midió el tiempo de respuesta, se realizaron varias consultas consecutivas para verificar que el tiempo calculado era consistente. Los tiempos de respuesta fueron consistentes con los tiempos reportados por el programa, lo que confirmó la correcta implementación de esta funcionalidad.

4.2.3 Pruebas en diferentes entornos de red: El programa fue ejecutado en diferentes redes con configuraciones y tamaños variados (desde redes pequeñas de 3 dispositivos hasta redes con más de 20 dispositivos) para verificar que podía procesar correctamente tablas ARP grandes y mostrar los fabricantes de manera eficiente.

4.3 Desafíos en las pruebas

4.3.1 Redes inestables: Durante algunas pruebas en redes con conexiones inestables, la API de MACLookup a veces no respondía. Este desafío fue resuelto implementando el manejo de tiempos de espera en la función `consultar_fabricante()`, lo que evitó que el programa se bloqueara indefinidamente.

4.3.2 Codificación de salida en Windows: En redes grandes, se observó que el comando `arp -a` en Windows presentaba problemas de codificación con ciertos caracteres especiales. Esto fue resuelto implementando un procesamiento de la salida para corregir manualmente los errores de codificación antes de procesar las direcciones MAC.

4.3.3 Volumen de consultas simultáneas: Durante las pruebas en redes grandes, se observó que el tiempo de respuesta de la API aumentaba ligeramente debido al volumen de consultas simultáneas. Aunque no afectó significativamente el funcionamiento del programa, se identificó la posibilidad de mejorar el rendimiento futuro implementando consultas paralelas para acelerar el proceso.

4.4 Conclusión de las pruebas

- Las pruebas realizadas confirmaron que el programa funciona correctamente bajo diferentes escenarios, manejando tanto direcciones MAC válidas como inválidas, tablas ARP con dispositivos conectados y vacías, y situaciones de red con tiempos de respuesta lentos. El sistema es robusto y maneja los errores adecuadamente, proporcionando mensajes útiles para el usuario en cada caso.

5. Discusión y conclusiones

En este proyecto, se desarrolló un sistema en Python para la identificación de fabricantes de dispositivos conectados a una red a través de sus direcciones MAC. El sistema permite tanto la consulta individual de una dirección MAC utilizando una API externa como la obtención de fabricantes para todos los dispositivos listados en la tabla ARP de un sistema. Entre los resultados principales, destacan:

5.1 Consulta individual

Se logró implementar exitosamente una función que consulta una API externa para obtener el fabricante asociado a una dirección MAC. Esta función también maneja tiempos de espera (timeout) y captura errores de red, lo que garantiza una ejecución robusta en diversas condiciones.

5.2 Análisis de la tabla ARP

El sistema procesa la tabla ARP obtenida del sistema y realiza consultas automáticas a la API para obtener información sobre los dispositivos listados. Las direcciones multicast y broadcast son reconocidas y manejadas correctamente, mostrando "Not found o no encontrado" en lugar de realizar consultas innecesarias.

5.3 Direcciones MAC aleatorias

En los últimos años, las direcciones MAC aleatorias han ganado popularidad en dispositivos modernos (como teléfonos inteligentes y laptops), lo que representa un desafío adicional para la identificación de dispositivos. El propósito principal de las direcciones MAC aleatorias es mejorar la privacidad del usuario, ya que evitan el rastreo de dispositivos cuando estos escanean redes cercanas o se conectan a redes públicas.

Funcionamiento de las direcciones MAC aleatorias:

1. Generación aleatoria: En lugar de utilizar la dirección MAC fija del dispositivo, se genera una dirección MAC temporal y aleatoria cada vez que se realiza un escaneo de redes o cuando el dispositivo se conecta a una red pública.
2. Renovación periódica: Las direcciones MAC aleatorias cambian frecuentemente para evitar la identificación continua de un dispositivo en múltiples redes.
3. Privacidad mejorada: Dificultan el rastreo pasivo del dispositivo en redes Wi-Fi públicas, protegiendo la identidad del usuario.
4. Uso en redes públicas y privadas: Mientras que las direcciones MAC aleatorias son comunes al conectarse a redes públicas, muchos dispositivos vuelven a utilizar la dirección MAC original al conectarse a redes privadas o conocidas para evitar problemas de autenticación.

Ejemplo:

Un dispositivo que usa direcciones MAC aleatorias puede cambiar su dirección cada vez que busca redes Wi-Fi cercanas. Por ejemplo, al buscar redes disponibles en una cafetería, el dispositivo podría generar una dirección MAC aleatoria para proteger la privacidad del usuario. Esto evita que un observador rastree el dispositivo de manera continua a través de la misma red o red de terceros.

Ventajas:

- Protección contra el rastreo: Las direcciones MAC aleatorias ayudan a evitar que redes y dispositivos malintencionados rastreen un dispositivo específico basándose en su dirección MAC fija.
- Privacidad: Las direcciones MAC aleatorias incrementan la privacidad de los usuarios, particularmente cuando se conectan a redes públicas o abiertas.

Desventajas:

- Problemas de autenticación: Algunas redes utilizan la dirección MAC para autenticar dispositivos. Si esta cambia constantemente, puede ser necesario autenticarse repetidamente o realizar configuraciones manuales.
- Compatibilidad: Las redes empresariales o controles de acceso basados en direcciones MAC pueden tener dificultades para manejar dispositivos con direcciones MAC aleatorias.

5.4 Reflexión sobre lo aprendido

Este proyecto permitió profundizar en varios aspectos importantes del desarrollo de sistemas de análisis de redes, incluyendo:

- El uso de APIs para obtener información relevante de dispositivos.
- El procesamiento de la tabla ARP y el manejo de diferentes entornos operativos.
- El impacto de las direcciones MAC aleatorias en la administración de redes y cómo dificultan la identificación de dispositivos en ciertos entornos.

5.5 Posibles mejoras

- Paralelización de consultas a la API para mejorar el rendimiento en redes grandes.
- Compatibilidad multiplataforma para soportar otros sistemas operativos como Linux o macOS.
- Implementación de una caché local para evitar depender completamente de una API externa.
- Interfaz gráfica o una versión web para facilitar el uso del sistema por parte de personas sin experiencia en la línea de comandos.

5.6 Conclusiones

En general, el proyecto fue exitoso en la implementación de un sistema funcional para la identificación de fabricantes de dispositivos mediante direcciones MAC, con un manejo adecuado de errores y robustez en la ejecución. El sistema no solo permite la consulta individual de direcciones MAC a través de una API externa, sino que también ofrece la capacidad de analizar la tabla ARP de una red local y obtener información útil sobre los dispositivos conectados.

A lo largo del desarrollo del proyecto, se superaron desafíos importantes, como el manejo de tiempos de espera en la API, la gestión de direcciones multicast y broadcast, y los problemas de codificación en la salida del comando arp -a en sistemas Windows. Estas soluciones mejoraron la eficiencia y la estabilidad del sistema, haciéndolo más versátil y adaptable.

No obstante, se identificaron varias áreas de mejora. Entre ellas, la implementación de consultas paralelas para mejorar el rendimiento en redes grandes, la compatibilidad multiplataforma para soportar sistemas operativos distintos a Windows, y la incorporación de una caché local para optimizar el uso de la API. Además, una interfaz gráfica o versión web facilitaría el uso del sistema a personas sin conocimientos técnicos.

Finalmente, este proyecto demostró la viabilidad de integrar herramientas de análisis de redes basadas en APIs y destacó la importancia de las direcciones MAC aleatorias en la privacidad de los usuarios. Esto abre nuevas oportunidades para futuros desarrollos en la administración y monitoreo de redes, enfocándose en la seguridad, la privacidad y la eficiencia.

6. Referencias

- [1] Python Software Foundation. Python Requests Library Documentation. Available online: <https://docs.python-requests.org/en/master/> (accessed on October 10, 2024).
- [2] MACLookup API. MAC Address Lookup API Documentation. Available online: <https://maclookup.app/api-v2/documentation> (accessed on October 10, 2024).
- [3] Microsoft. ARP Command: Windows Command Line Reference. Available online: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/arp> (accessed on October 10, 2024).
- [4] Python Software Foundation. Subprocess Library Documentation. Available online: <https://docs.python.org/3/library/subprocess.html> (accessed on October 10, 2024).
- [5] Apple Support. *Utilización de direcciones MAC aleatorias en iOS*. Available online: <https://support.apple.com/> (accessed on October 10, 2024).
- [6] Google Android Developers. *Direcciones MAC aleatorias en Android*. Available online: <https://developer.android.com/> (accessed on October 10, 2024).
- [7] Cisco Systems. *Impacto de las direcciones MAC aleatorias en redes empresariales*. Available online: <https://www.cisco.com/> (accessed on October 10, 2024).