Tarea 2 :Consultar Fabricantes de tarjetas de red a través de una API

Joyce Vivar Guzman, joyce.vivar@alumnos.uv.cl

1. Objetivo

Implementar una herramienta basada en línea de comandos llamada "OUILookup" en Python.
Desarrollar un programa que permita consultar el fabricante de una tarjeta de red a través de su dirección MAC.
Utilizar una API REST de consulta de direcciones MAC para acceder a la base de datos de fabricantes.

2. Descripción del trabajo a realizar

Se debe implementar una herramienta basada en línea de comandos para consultar el fabricante de una tarjeta de red dada su dirección MAC. La aplicación se llamará OUILookup y se desarrollará en Python. La base de datos a utilizar será cualquier API REST pública que permita obtener los fabricantes de tarjetas de red a partir de una MAC determinada. Para fines de revisión, se puede utilizar la API REST pública de consulta de MACs disponible en https://maclookup.app.

3. Descripción del problema y diseño de la solución

Este código resuelve el requisito de identificar los fabricantes de dispositivos de red basándose en sus direcciones MAC (Media Access Control). Esto es útil para administradores de red y profesionales de seguridad que necesitan obtener información sobre los dispositivos conectados a una red

Este diseño permite una clara separación de responsabilidades, distribuyendo las tareas del programa en funciones,tales como:

- La función main() maneja la lógica principal y el flujo del programa,es decir Esta función procesa los parámetros de la línea de comandos, como --mac y --arp, utilizando getopt para manejar los argumentos
- mac_lookup() se encarga de consultar la API externa y procesar la respuesta,como lo hace ,pues la función lookup_mac tiene como objetivo realizar una consulta a la API pública de maclookup.app para obtener información sobre el fabricante asociado con una dirección MAC específica. Además, mide el tiempo que tarda la respuesta de la API.
- get_arp_table() funcion que obtiene la tabla ARP del sistema y la complementa con entradas de prueba, usando el comando arp -e,através de subprocess.check_output(), de la misma manera el uso de subprocess, cómo se

procesa la salida de la tabla ARP, y cómo se realiza la consulta a la API para cada dirección MAC.

- usage() proporciona instrucciones de uso al usuario, indicando las opciones disponibles como - - mac , - - arp, y - - help

Para asegurar que el script OUILookup funcione correctamente y gestionar sus dependencias de manera aislada, se configuró un entorno virtual específico para este proyecto. Aquí están los pasos que se siguieron:

- Se creó el entorno virtual :

  python3 -m venv /home/joey/OUILookup_task/ouilookup_env

- Activación del entorno virtual:

  source /home/joey/OUILookup_task/ouilookup_env/bin/activate

- Instalación de dependencias :cuando sea activado el entorno virtual, se instalaron las dependencias necesarias:

  pip install requests

- Configuración : En la primera línea del script, se especificó la ruta al intérprete de Python dentro del entorno virtual:

  #!/home/joey/OUILookup_task/ouilookup_env/bin/python

  Esto asegura que el script se ejecute usando el Python del entorno virtual, con todas las dependencias correctamente instaladas

- y por último la ejecución del script:

  ./OUILookup [con las opciones pertinentes]

  Esta configuración permite aislar las dependencias del proyecto y evitar conflictos con otros paquetes instalados en el sistema. Además, facilita la reproducibilidad del entorno de ejecución en diferentes máquinas.

4. Casos con Linux

Uso sin parámetros o con la opción `--help`.

```
joey@Joy:~/OUILookup_task$ ./OUILookup.py --help
usage: OUILookup.py [-h] (--mac MAC | --arp)

OUILookup - Consulta fabricantes de tarjetas de red

options:
  -h, --help  show this help message and exit
  --mac MAC   MAC a consultar. P.e. aa:bb:cc:00:00:00
  --arp       Muestra los fabricantes de los hosts disponibles en la tabla ARP
```

Uso con parámetros:

- Caso MAC que esté en la base de datos

```
joey@Joy:~/OUILookup_task$ ./OUILookup.py --mac 98:06:3c:92:ff:c5
MAC address: 98:06:3c:92:ff:c5
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de respuesta: 700.36ms
```

- Caso MAC que no esté en la base de datos

```
joey@Joy:~/OUILookup_task$ ./OUILookup.py --mac 98:06:3f:92:ff:c5
MAC address: 98:06:3f:92:ff:c5
Fabricante: Not found
Tiempo de respuesta: 622.43ms
```

Caso fabricantes de las MAC disponibles en la tabla arp:

```
joey@Joy:~/OUILookup_task$ ./OUILookup.py --arp
MAC/Vendor:
00:15:5d:57:13:df / Microsoft Corporation
00:01:97:bb:bb:bb / Cisco Systems, Inc
b4:b5:fe:92:ff:c5 / Not found
00:E0:64:aa:aa:aa / SAMSUNG ELECTRONICS
AC:F7:F3:aa:aa:aa / Xiaomi Communications Co Ltd
```

5. Casos de Pruebas

```
joey@Joy:~/OUILookup_task$ ./OUILookup.py --mac 98:06:3c:92:ff:c5
MAC address: 98:06:3c:92:ff:c5
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de respuesta: 734.04ms
joey@Joy:~/OUILookup_task$ ./OUILookup.py --mac 9c:a5:13
MAC address: 9c:a5:13
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de respuesta: 1740.62ms
joey@Joy:~/OUILookup_task$ ./OUILookup.py --mac 48-E7-DA
MAC address: 48-E7-DA
Fabricante: AzureWave Technology Inc.
Tiempo de respuesta: 597.71ms
```

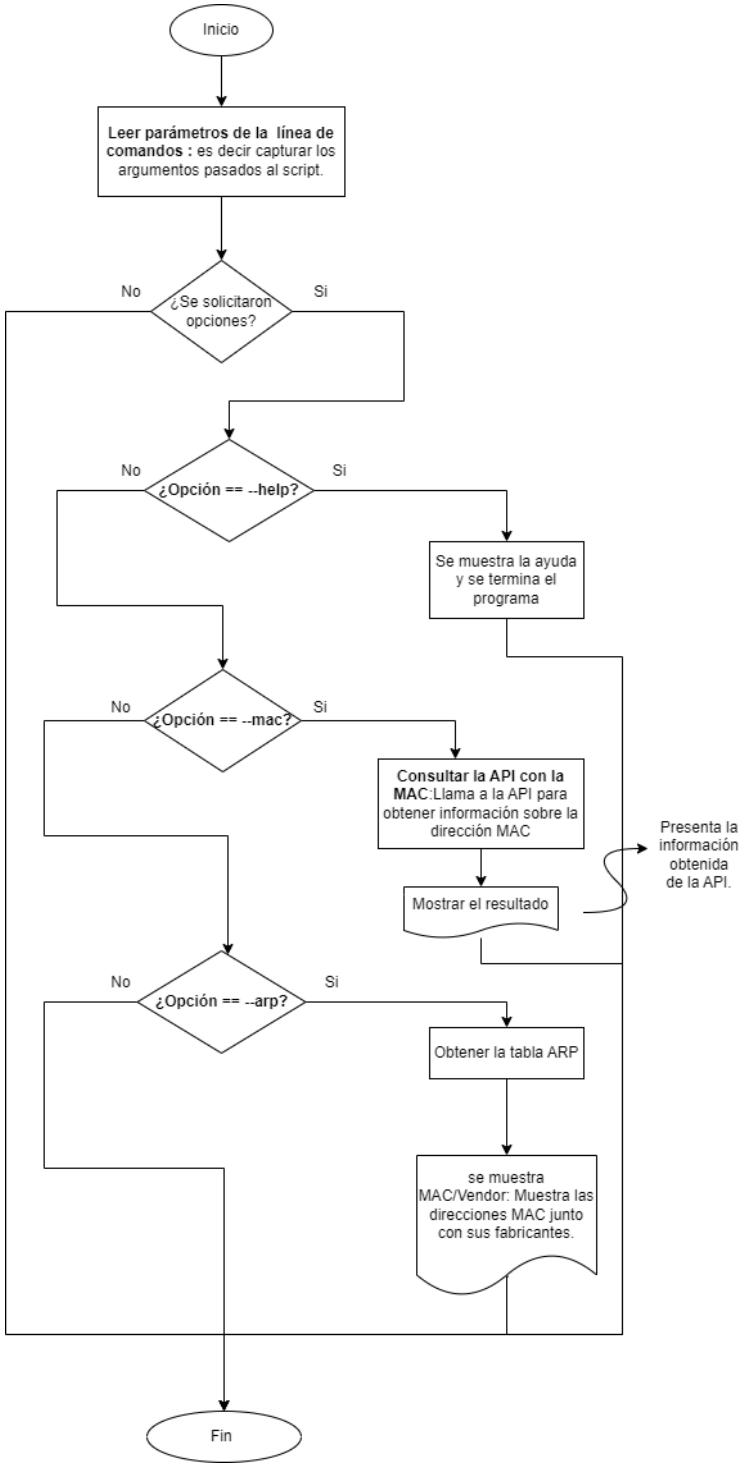
6. Mac Aleatorias

Las direcciones MAC (Media Access Control) aleatorias son una técnica de privacidad implementada en dispositivos móviles y otros dispositivos electrónicos para proteger la identidad del usuario al conectarse a redes Wi-Fi. En lugar de usar la dirección MAC única y permanente del dispositivo, se genera una dirección temporal y aleatoria para cada conexión. Esto dificulta el seguimiento del dispositivo por parte de terceros y mejora la privacidad del usuario.

1. Aleatoriedad de direcciones MAC y privacidad:Para abordar este problema de privacidad, algunos dispositivos móviles modernos utilizan direcciones MAC temporales y aleatorias que son diferentes de su dirección global real. Cuando se envían solicitudes de sondeo, usan una dirección MAC seudónima aleatoria que se cambia periódicamente.
2. Evasión de ataques mediante direcciones MAC cambiantes: "Si el dispositivo malicioso cambia su dirección MAC en cada solicitud, simula la existencia de múltiples clientes, lo que complica mucho más la detección y bloqueo del atacante"
3. Implementaciones en iOS y Android: "Apple añadió la aleatorización de direcciones MAC a sus dispositivos a partir de iOS 8. En iOS 9, esta funcionalidad

se extendió para incluir la aleatorización en lo que Apple llama escaneos de localización y auto-join. Android 6.0 utiliza la aleatorización para escaneos en segundo plano, si el controlador y el hardware lo soporta.

7. Diagrama de Flujo



8. Referencias

[1] Martin, J., Mayberry, T., Donahue, C., Foppe, L., Brown, L., Riggins, C., Rye, E. C., & Brown, D. (2017). A Study of MAC Address Randomization in Mobile Devices and When it Fails. Proceedings on Privacy Enhancing Technologies, 2017(4), 365-383.

[2] Xu, Q., Zheng, R., Saad, W., & Han, Z. (2016). Device Fingerprinting in Wireless Networks: Challenges and Opportunities. IEEE Communications Surveys & Tutorials, 18(1), 94-104.

[3] Vanhoef et al. (2016) - "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms"