Informe tarea 2

Basthian Hernandes Gallardo, basthian.hernandez@alumnos.uv.cl

Carlos González Pérez,carlos.gonzalezp@alumnos.uv.cl

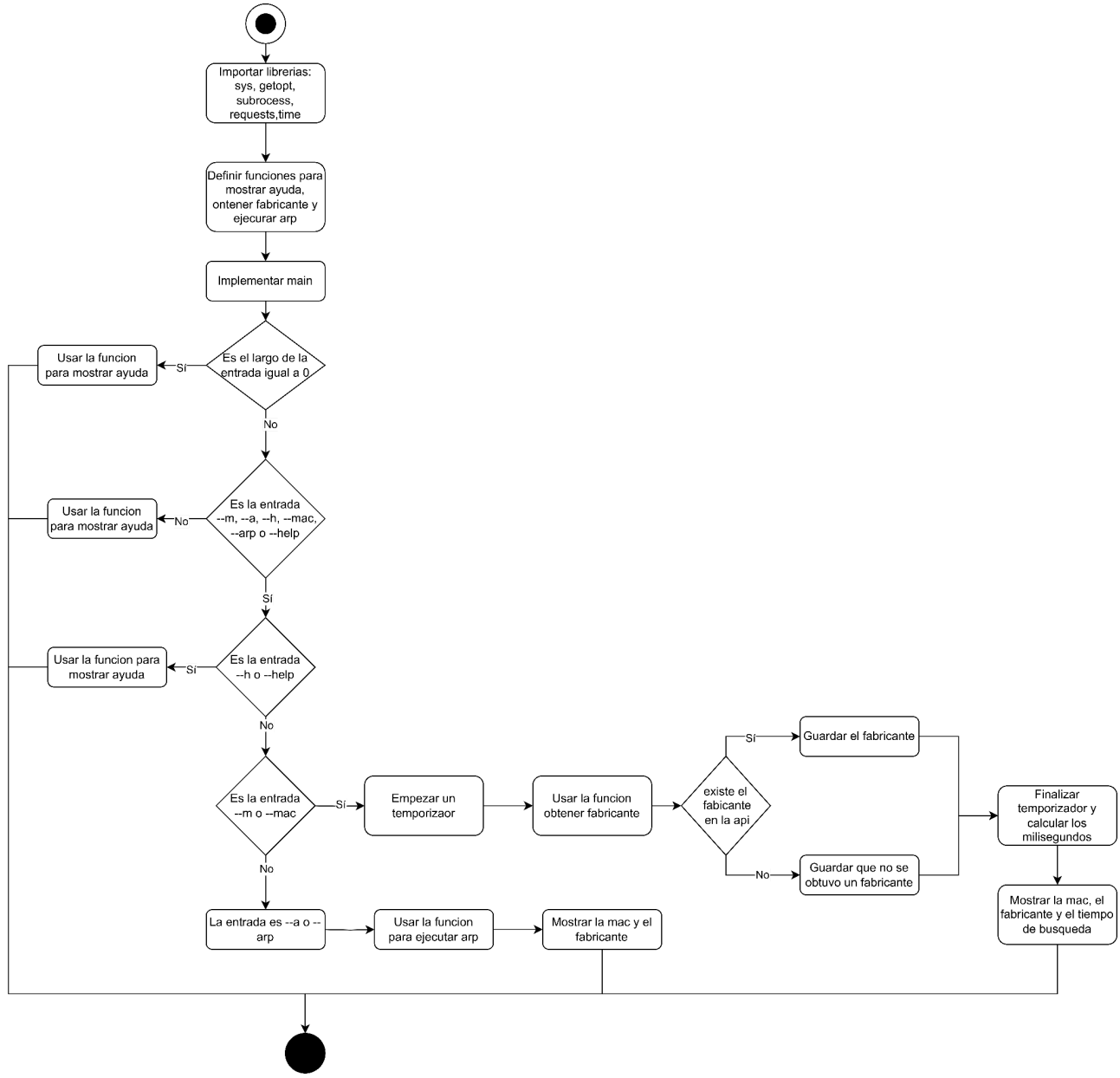
1. Introducción

Este informe presenta la implementación de OUILookup, una herramienta de línea de comandos desarrollada en Python que permite consultar el fabricante de una tarjeta de red a partir de su dirección MAC. Utiliza una API REST para acceder a una base de datos de fabricantes, facilitando la obtención de información sin necesidad de procesos manuales o herramientas complejas. En este se describe la estructura del programa y su integración con la API, mostrando cómo se cumplen los objetivos de manera eficiente y escalable.

2. Descripción del problema y diseño de la solución

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 esto se utilizará la API REST pública de consulta de MACs disponible en https://maclookup.app.

2.1. diagrama de flujo



3. Implementación

Antes de ejecutar el código, se debe tener en cuenta las bibliotecas implementadas al principio para saber su uso durante su ejecución:

```
import sys
import getopt
import subprocess
import requests
import time
```

- Import sys: Biblioteca que asegura que el programa se cierre de manera controlada una vez ya ejecutado
- Import getopt: Biblioteca utilizada para el ingreso de parámetros (como –mac o –arp en este caso).
- Import subprocess: Biblioteca utilizada para ejecutar nuevos procesos (comando arp -a) y capturar su salida
- Import request: Biblioteca utilizada para realizar una solicitud GET a una API (maclookup.app)
- Import time: Biblioteca utilizada para intervalos de tiempo.

- Función mostrar_ayuda():

Esta función solo funciona para mostrar los parámetros disponibles para la ejecución del código. Solo funciona en caso de que no se ingrese un parámetro o algún parámetro incorrecto en la ejecución del código.

```
def mostrar_ayuda():
    print("""
    Opciones:
    -m, --mac: Consulta el fabricante de una dirección MAC específica. Ejemplo: -m aa:bb:cc:00:00:00
    -a, --arp: Muestra los fabricantes de los host disponibles en la tabla arp.
    -h, --help: Muestra este mensaje.
    """)
```

- Función obtener_fabricante(mac_solicitado)

Sirve para obtener el fabricante de una mac solicitado a la API de maclookup.app. Se encarga de obtener la variable “company” dentro de la API y mostrarlo en pantalla. En el caso de que no se presente una variable “company” dará un mensaje “Fabricado no encontrado” y en el caso de que ocurra un error durante la solicitud con la API, dará un mensaje “Error”.

```python
def obtener_fabricante(mac_solicitado):
    """
    Consulta la API maclookup.app para obtener el fabricante de la MAC solicitada.
    """
    url = f"https://api.maclookup.app/v2/macs/{mac_solicitado.replace(':', '')}"

    try:
        # Realiza la solicitud
        response = requests.get(url)

        if response.status_code == 200:
            data = response.json()
            if data["found"]:
                # Devuelve el fabricante
                return data["company"]
            else:
                return "Fabricante no encontrado."
        else:
            return "Error: No se pudo obtener la información."
    except requests.RequestException as e:
        return f"Error: {e}"
```

- Función ejecutar_arp():

La función obtiene las direcciones MAC visibles en la red local a partir del comando arp -a presentado en el comando subprocess, consulta el fabricante correspondiente para cada una y muestra los resultados en la terminal.

```python
def ejecutar_arp():
    try:
        # Ejecuta el comando 'arp -a' y captura la salida
        result = subprocess.run(["arp", "-a"], capture_output=True, text=True, check=True)
        lines = result.stdout.splitlines()

        print("Mac/Fabricante:")
        for line in lines:
            # Extrae la MAC y reemplaza guiones por dos puntos
            parts = line.split()
            if len(parts) >= 2:
                mac_address = parts[1].replace('-', ':')
                fabricante = obtener_fabricante(mac_address)  # Llama a la función para obtener el fabricante
                print(f"{mac_address} / {fabricante}")
    except subprocess.CalledProcessError as e:
        print("Error al ejecutar arp -a:", e)
```

- Función main:

Se encarga de ejecutar las funciones antes mencionadas gracias a la implementación de la biblioteca Getopt. A partir de estos parámetros definirá cuáles funciones hará funcionar el programa.

Los siguientes parámetros son:
-h, –help: Muestra los parámetros disponibles en el programa.

-m, –mac: Muestra el fabricante de una dirección mac solicitada.

-a, –arp: Muestra una tabla con todas las direcciones mac y sus fabricantes (si estas existen) en la red local.

```python
def main(argv):
    # En el caso de que no haya argumentos, se muestra los parametros disponibles
    if len(argv) == 0:
        mostrar_ayuda()
        sys.exit(2)

    try:
        opts, args = getopt.getopt(argv, "m:ah", ["mac=", "arp", "help"])
    except getopt.GetoptError:
        mostrar_ayuda()
        sys.exit(2)

    for opt, arg in opts:
        if opt in ("-h", "--help"):
            mostrar_ayuda()
            sys.exit()
        elif opt in ("-m", "--mac"):
            start_time = time.time()  # Inicio del temporizador
            fabricante = obtener_fabricante(arg)
            end_time = time.time()  # Fin del temporizador
            elapsed_time_ms = (end_time - start_time) * 1000  # Tiempo en milisegundos
            print(f"Dirección MAC: {arg}\nFabricante: {fabricante}\nTiempo de búsqueda: {elapsed_time_ms:.2f} ms")
        elif opt in ("-a", "--arp"):
            ejecutar_arp()  # Llama a la función que muestra la tabla ARP

if __name__ == "__main__":
    main(sys.argv[1:])
```

4. Pruebas

- Prueba 1: mac 98:06:3c:92:ff:c5

```
(mi_entorno) PS C:\Users\ezze_\py> python OUILookup.py --mac 98:06:3c:92:ff:c5
Dirección MAC: 98:06:3c:92:ff:c5
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de búsqueda: 405.37 ms
```

- prueba 2: mac 9c:a5:13

```
(mi_entorno) PS C:\Users\ezze_\py> python OUILookup.py --mac 9c:a5:13
Dirección MAC: 9c:a5:13
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de búsqueda: 642.84 ms
```

- prueba 3: mac 48-E7-DA

```
(mi_entorno) PS C:\Users\ezze_\py> python OUILookup.py --mac 48-E7-DA
Dirección MAC: 48-E7-DA
Fabricante: AzureWave Technology Inc.
Tiempo de búsqueda: 810.92 ms
```

4.1. ¿Qué es y para qué funciona una mac aleatoria?

Tradicionalmente, los dispositivos emplean la dirección MAC de fábrica, que es única y estática, lo que facilita su identificación. Como los dispositivos utilizan direcciones MAC para conectarse a redes Wi-Fi y estas cómo se transmiten sin cifrado, pueden ser capturadas y usadas para rastrear la ubicación de un usuario. Para mejorar la privacidad, la función de aleatorización de MAC genera direcciones aleatorias al conectarse a redes, modificando ciertos bits para crear una dirección única y evitar el rastreo. Las direcciones MAC tienen 48 bits y se representan en 12 dígitos hexadecimales. [1]

5. Discusión y conclusiones

En conclusión la herramienta OUILookup, desarrollada en Python, cumple con los objetivos al permitir consultar de manera rápida y precisa el fabricante de una tarjeta de red a través de su dirección MAC, utilizando una API REST para acceder a la base de datos de fabricantes.

6. Referencias

[1]Android. Comportamiento de aleatorización MAC. 2024-03-18,párrafos de 2 a 4, https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior?hl=es