Tarea 2: Implementacion de un OUI Lookup Tool.

Alejandro Dinamarca Cáceres, alejandro.dinamarca@alumnos.uv.cl

Thean Orlandi Guzman, thean.orlandi@alumnos.uv.cl

1. Introducción

En el presente trabajo se busca implementar una herramienta de línea de comandos en Python que consulta el fabricante de una tarjeta de red a partir de su dirección MAC o dirección IP. Utilizando una base de datos parametrizada basada en el repositorio de Wireshark para identificar fabricantes de dispositivos de red de manera eficiente.

1.1. Materiales y Métodos

1.2. Materiales

Se utilizó un sistema basado en MAC OS, que a su vez está basado en Unix.

1.3. Herramientas

1.3.1. Python

Para la implementación de la herramienta OUI Lookup Tool se utilizó el paradigma de la programación funcional utilizando el esqueleto proporcionado en GitHub (https://github.com/Stry12/Tarea02-Redes-De-Computadoras).

1.3.2. iTerm2

El programa iTerm2 es un emulador del terminal, específico para MacOS, con el cual se ejecutará la aplicación desarrollada.

Repositorio en el siguiente enlace.

2. Resultados

2.1. OUILookup

De acuerdo al intérprete de comandos tipo BASH:

2.1.1. Ejemplo de uso con parámetro --ip

Para comprobar una IP específica, es posible utilizar el siguiente comando, el cuál dependiendo de si pertenece o no al área local (emulada por el diccionario parametrizado), proporcionará como salida los datos correspondientes o un mensaje de que la IP está fuera del área local:

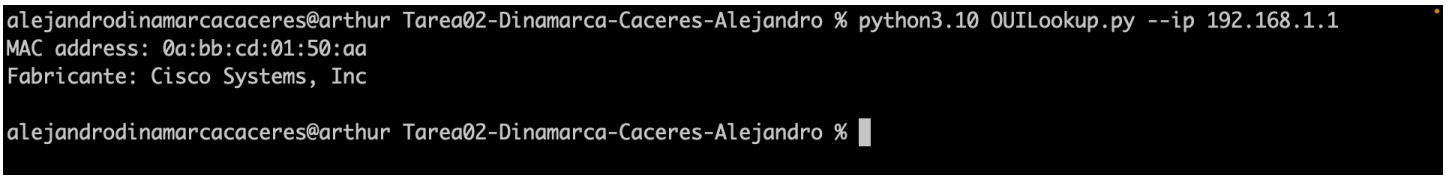

Figura 1. Terminal con parámetro --ip y argumento de IP conocida.

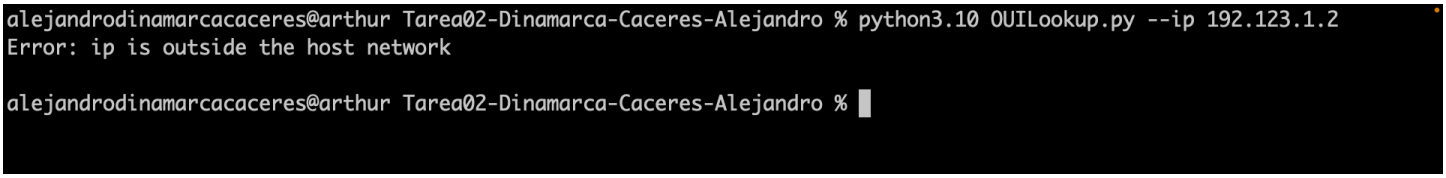

Figura 2. Terminal con parámetro --ip y argumento de IP desconocida.

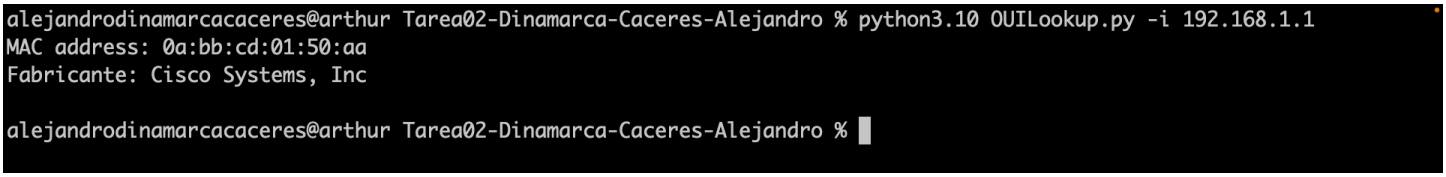

Figura 3. Terminal con atajo -i.

2.1.2. Ejemplo de uso con parámetro --mac

Para comprobar una MAC, es posible utilizar el siguiente comando:

```
alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro % python3.10 OUILookup.py --mac 00:00:00:00:00:01
MAC address: 00:00:00:00:00:01
Fabricante: Apple, Inc

alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro %
```

Figura 4. Terminal con parámetro --mac y argumento de MAC conocida.

```
alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro % python3.10 OUILookup.py --mac 00:00:00:00:05:01
MAC address: 00:00:00:00:05:01
Fabricante: Not found

alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro %
```

Figura 5. Terminal con parámetro --mac y argumento de MAC desconocida.

2.1.3. Ejemplo de uso con parámetro --arp

Para obtener la tabla ARP:

```
alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro % python3.10 OUILookup.py --arp
IP              /       MAC                     /       VENDOR
192.168.1.0     /       ss:ss:ss:ss:ss:ss       /       Unknown
192.168.1.30    /       00:00:00:00:00:01       /       Apple, Inc
192.168.1.1     /       0a:bb:cd:01:50:aa       /       Cisco Systems, Inc
192.168.1.2     /       aa:bb:cc:dd:ee:ff       /       Netgear
192.168.1.4     /       dd:ee:ff:00:00:00       /       Huawei
192.168.1.5     /       ab:cd:ef:12:34:56       /       TP-Link

alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro %
```

Figura 6. Terminal con parámetro --arp.

2.1.4. Ejemplo de uso con parámetro --help

Para obtener ayuda:

```
alejandrodinamarcacaceres@arthur Tarea02-Dinamarca-Caceres-Alejandro % python3.10 OUILookup.py --help
Use: ./OUILookup --ip <IP> | --mac <IP> | --arp | [--help]
--ip : IP del host a consultar.
--mac: MAC a consultar. P.e. aa:bb:cc:00:00:00.
--arp: muestra los fabricantes de los host
disponibles en la tabla arp.
--help: muestra este mensaje y termina.
```

Figura 7. Terminal con parámetro --arp.

3. Implementación

A continuación, se expondrá el código de fuente de la implementación, el cuál será dividido en secciones para ser explicado. Cabe destacar que todas las secciones, en conjunto, conforman el programa.
Al final de esta misma sección se encontrará todas las secciones unificadas.

3.1. Variables globales

A modo de facilitar la modificación del "boilerplate code", se colocaron al principio del código. Además, se implementó el cálculo automático del NET-ID mediante la separación de las cadenas en octetos válidos, y el respectivo cálculo mediante la operación AND.

```
import subprocess
import getopt
import sys


# Redes
RED_HOST = "192.168.1.30"
MASK = "255.255.255.0"
# Operación entre RED_HOST y MASK (AND por cada octeto)
NET_ID = ".".join([str(i & j) for i, j in zip(map(int, RED_HOST.split(".")), map(int, MASK.split(".")))])
# Strings de respuesta
MISMA_RED = "MAC address: {mac}\nFabricante: {manufacturer}\n"
OTRA_RED = "Error: ip is outside the host network\n"
MAC_EN_BASE_DE_DATOS = "MAC address: {mac}\nFabricante: {manufacturer}\n"
AYUDA = """Use: ./OUILookup --ip <IP> | --mac <IP> | --arp | [--help]
--ip : IP del host a consultar.
--mac: MAC a consultar. P.e. aa:bb:cc:00:00:00.
--arp: muestra los fabricantes de los host
disponibles en la tabla arp.
--help: muestra este mensaje y termina."""

# Base de datos parametrizada
db: dict = {
    NET_ID: {
        "mac": "ss:ss:ss:ss:ss:ss",
        "manufacturer": "Unknown"
    },
```

Figura 8. Declaración de variables globales.

3.2. Funciones para obtener IP, MAC, y datos en general

A continuación, se expondrán las funciones que emulan la obtención de una IP, MAC o fabricante de una tarjeta de red mediante un diccionario parametrizado, por lo que, las funciones hacen uso de la obtención de datos mediante las llaves “mac” y “manufacturer”:

```python
# Función para obtener los datos de fabricación de una tarjeta de red por IP
def obtener_datos_por_ip(ip: str) -> str:
    """
    Obtiene los datos de fabricación de una tarjeta de red por IP.

    Parámetros:
        ip : IP del host a consultar.

    Retorna:
        Datos de fabricación de la tarjeta de red o "Not found" si no se encuentra.
    """
    if ip in db:
        if "manufacturer" in db[ip]:
            return db[ip]["manufacturer"]
        else:
            return "Not found"
    else:
        return ""
```

Figura 9. Función para obtener datos por IP mediante uso de par llave-valor en diccionario.

De forma análoga y similar, las funciones “`obtener_datos_por_mac(mac: str)`” y “`obtener_mac_por_ip(ip: str)`” hacen uso del diccionario parametrizado, por lo que se obviará su descripción. Cabe destacar que su documentación está definida en el código.

3.3. Función para obtener tabla ARP

La función para obtener la tabla ARP hace uso de las funciones previamente definidas, como también, del formatear correctamente la cadena para que tenga estructura de tabla:

```
# Función para obtener la tabla ARP
def obtener_tabla_arp():
    """
    Obtiene la tabla ARP.

    Retorna:
        Texto con la tabla ARP.
    """
    tabla_arp = f"IP\t\t/\tMAC\t\t\t/\tVENDOR\n"
    for ip, datos in db.items():
        # Si la MAC no está vacía, se agrega a la tabla (ya que si está vacía, no se conoce la MAC y
por ende aún no se ha hecho ARP)
        if datos["mac"] != "":
            tabla_arp += f"{ip}\t/\t{obtener_mac_por_ip(ip)}\t/\t{obtener_datos_por_ip(ip)}\n"
    return tabla_arp
```

Figura 10. Función para obtener tabla ARP. Destacar cómo se utilizan funciones previamente definidas.

3.4. Función principal main

Finalmente, en la función main se ejecutan de acuerdo a los parámetros pasados por terminal las funciones correspondientes, por lo que es un condicional que, además, se encarga del manejo de excepciones para brindar ayuda al usuario.

```
def main(argv):
    ip = None

    try:
        opts, args = getopt.getopt(argv, "i:", ["ip=", "mac=", "arp", "help"])

    except getopt.GetoptError:
        #Modificar para coincidir con tarea
        print("Use: python OUILookup.py --ip <IP> | --Arg2 <Arg2> | --Arg3 | [--help] \n --ip : IP del
host a consultar. \n --Arg2:  \n --Atg3: \n --help:")
        sys.exit(2)

    for opt, arg in opts:
        if opt in ("-i", "--ip"):
            if arg in db:
                print(MISMA_RED.format(mac=obtener_mac_por_ip(arg),
manufacturer=obtener_datos_por_ip(arg)))
            else:
                print(OTRA_RED)
        elif opt in ("--mac"):
            print(MAC_EN_BASE_DE_DATOS.format(mac=arg, manufacturer=obtener_datos_por_mac(arg)))
        elif opt in ("--arp"):
            print(obtener_tabla_arp())
        elif opt in ("--help"):
            print(AYUDA)
        else:
            print("Debe proporcionar una opción válida (-i, -m o -a).")
            sys.exit(2)
        sys.exit(0)
```

Figura 11. Función principal.

3.5. Esquema general de resolución (diagrama de flujo)

El siguiente diagrama de flujo proporciona una visión macro del flujo del programa:

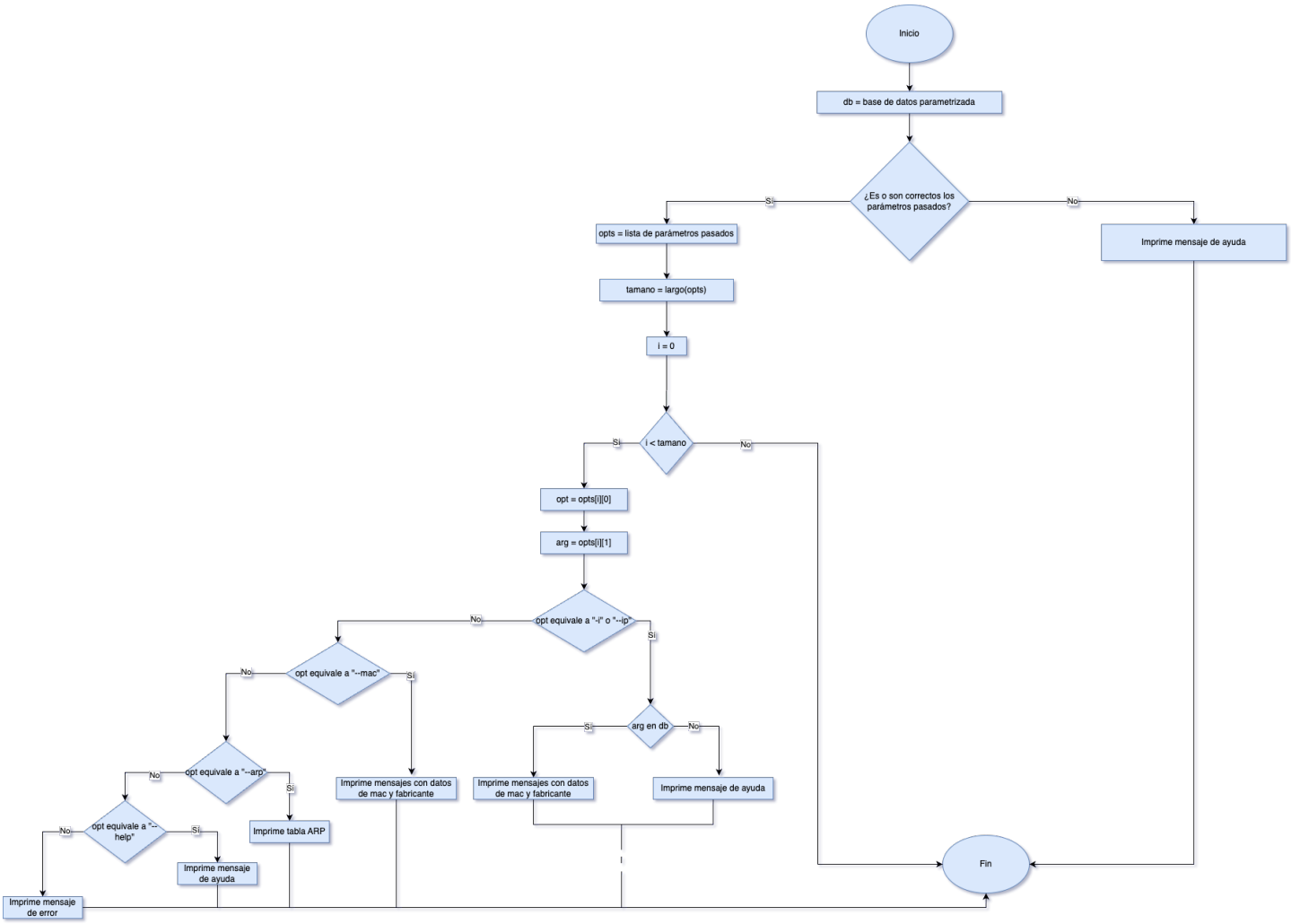

Figura 12. Diagrama de flujo.

3.6. Enlace al repositorio en Github

El repositorio está en el siguiente enlace: https://github.com/adinamarca/tarea2-OUILookup.

4. Discusión y conclusiones

En la resolución del trabajo, se logró implementar un programa capaz de capturar parámetros desde el terminal, lo que en un principio pareciera desafiante, pero gracias a la estructura y cuerpo proporcionados se logró comprender que, se logró entender el cómo funciona el módulo opt, como también recibir correctamente los argumentos proporcionados por el usuario.