Tarea 2: Implementación de un OUI Lookup tool

Vicente Miralles vicente.miralles@alumnos.uv.cl

Matias Rivas matias.rivas@alumnos.uv.cl

Nataniel Palacios Nataniel.palacios@alumnos.uv.cl

1. Introducción

Determinar el fabricante desde las direcciones MAC o IP puede proporcionarnos información valiosa en diversos escenarios. Por un lado, resulta crucial para resolver problemas de redes al tratar con dispositivos específicos de cada fabricante. Además, desempeña un papel fundamental en la seguridad de redes, permitiendo rastrear y reconocer los dispositivos conectados. Por ejemplo, si nos topamos con una dirección MAC desconocida, el fabricante puede ofrecernos pistas sobre la naturaleza del dispositivo en cuestión.

En este contexto, hemos desarrollado un programa en Python diseñado para consultar el fabricante de una tarjeta de red desde una dirección MAC o IP. Esta herramienta facilita la identificación y gestión de dispositivos en una red Sacando la información de la base de datos

2. Materiales y Métodos

Para esta tarea se necesita un computador con un editor de texto cualquiera, aunque en esta ocasión se utilizo un IDE llamado Visual Studio Code para realizar el código.

3. Resultados

El código inicializa con la función *with open()* para abrir el archivo llamado *manuf* (archivo que contiene los fabricantes de las tarjetas de red dada una MAC) abrirlo con la opción 'r' o sea en modo lectura y especifica que la codificación utilizada es la UTF-8. Todo esto se referenciará al llamar la variable archivo, por lo que con el método *readlines()* leerá todo el archivo y lo devolverá como una cadena de String en una lista, donde cada palabra será un elemento de esta. Además, anterior a todo esto se declara una variable con un numero IP pero tipo String y estas variables se declaran como variables globales (figura 1).

```
global contenido,computadorIP
computadorIP="192.168.1.30"
with open('manuf','r',encoding='utf-8') as archivo:
    contenido=archivo.readlines()
```

Figura 1

Luego se define una funcion llamada *obtener_datos_por_ip()* en donde a esta funcion se le dará una ip y esta muestra la direccion MAC y su fabricante.

Se declara la variable *aux* con la direccion IP antes mencionada con el metodo *.split('.')* en donde este metodo lo que hace es crear una lista y cada elemento de esta lista, es el numero delimintado por los '.' (['192', '168', '1', '30']).

La variable *compa* lo que hace es unir los primeros 3 elementos de la lista utilizando el punto como un separador para decir que se le aplica la mascara a la direccion.

Luego se aplica esta mascara a una ip para comprobar si esta pertenece a la red.

La funcion *try:* lo que hace es ejecutar el comando arp con la biblioteca *subprocess* para obtener la direccion MAC

Luego decodifica la salida con *ISO-8859-1.*

El resultado lo devolverá como una lista con el metodo *split()* y lo almacenará en la variable partes.

Como la variable **partes** es una lista, el indice 11 es la direccion mac, en donde se almacenará en la variable llamada *mac*

En el caso en que si existe un error en *try:* se ejecutará *except* donde nos devolveria un mensaje de error.

En el caso contrario, si la comparacion es falsa, eso quiere decir que la ip no pertenece a la red (ver Figura 2).

```python
# Función para obtener los datos de fabricación de una tarjeta de red por IP
def obtener_datos_por_ip(ip):
    #"aplica la mascara" a la direcicon ip del computador
    aux=computadorIP.split('.')
    compa='.'.join(aux[:3])
    #"aplica la mascara" a la direccion ip que se quiere buscar para comprobar si pertenecen a la misma red
    prueba=ip.split('.')
    comprobacion='.'.join(prueba[:3])
    if compa==comprobacion:
        try:
            # Ejecutar el comando ARP para obtener la dirección MAC
            resultado = subprocess.check_output(["arp", "-a", ip])
            # Decodificar la salida en una cadena
            resultado = resultado.decode("ISO-8859-1")
            # Buscar la dirección MAC en la salida
            partes = resultado.split()
            mac = partes[11]

            return mac
        except Exception as e:
            print("Error al obtener la dirección MAC:", str(e))
    else:
        print("Error: ip is outside the host network")
        sys.exit(2)
    return None
```

Figura 2

Se define una funcion para ingresar una direccion mac y devolver el fabricante de aquella tarjeta de red.

Se recorre *contenido* (contenido del archivo) linea por linea.

Luego a *linea* se le aplica el metodo *.strip()* para eliminar los espacios en blanco al principio y al final de una linea, luego seguido al metodo antes mencionado, se le aplica el *.split('\t')*, dandole un argumento para que use como delimitador las tabulaciones y finalmente todo lo almacena en la variable partes.

Luego el indice 0 representa la mac y el indice 1 al fabricante que se guardan en un diccionario llamado *fabricantes* en donde si la mac está en este diccionario, imprime la mac y su fabricantes, y si es falsa la condicion entrega un mensaje diciendo que no se encuentra la mac en el diccionario (not found)(ver Figura 3).

```
# Función para obtener los datos de fabricación de una tarjeta de red por MAC
def obtener_datos_por_mac(mac):
    # Implementa la lógica para obtener los datos por MAC aquí
    fabricantes={}
    for linea in contenido:
        partes = linea.strip().split('\t')
        if len(partes) >= 2:
            macs = partes[0]
            fabricante = partes[1]
            fabricantes[macs] = fabricante
    if mac in fabricantes:
        print(f"MAC Adress: {mac} \nFabricante   : {fabricantes[mac]}")
    else:
        print(f"MAC Adress: {mac} \nFabricante   : Not found")
```

Figura 3

Se define la funcion *obtener_tabla_arp()* que permite mostrar los fabricantes de las direcciones mac presentes en la tabla ARP del dispositivo.

busca y muestra la dirección IP, la dirección MAC y el posible fabricante asociado a cada dispositivo en la red local.

Para hacer esto, el código ejecuta el comando arp -apara obtener la tabla ARP y luego procesa la salida. Para cada entrada en la tabla ARP, intenta identificar el fabricante asociado a la dirección MAC. Finalmente, imprime la información de cada dispositivo encontrado en la red. Si se producen errores durante este proceso, se manejarán con un bloque de excepción.

Cabe mencionar que para que el código funcione correctamente, se supone que hay una variable contenidodefinida con información de fabricantes asociados a direcciones MAC y que el sistema en el que se ejecuta tiene acceso a la funcionalidad de ARP (Ver Figura 4).

```
# Función para obtener la tabla ARP
def obtener_tabla_arp():
    try:
        arp_output = subprocess.getoutput("arp -a").split('\n')[3:]
        arp_entries = [line.split() for line in arp_output if line.strip()]

        print("IP/MAC/Fabricante:")
        for entry in arp_entries:
            ip = entry[0]
            mac = entry[1].upper()
            mac_prefix = mac.replace(':', '').upper()[:6]

            fabricante = "Not found"

            for line in contenido:
                if line.startswith(mac_prefix):
                        parts = line.strip().split('\t')
                        if len(parts) >= 2:
                            fabricante = parts[1].strip()
                            break

            print(f"{ip} / {mac} / {fabricante}")

    except Exception as e:
        print(f"Error: {e}")
```

Figura 4

Si lo ingresado por consola es un IP ejecuta la funcion obtener datos por IP, la funcion devuelve un MAC que se utiliza como parametro para llamar a la funcion que entrega datos con un MAC especificado.

Si el parametro ingresado es una direccion MAC, el programa corre la funcion obtener_datos_por_mac (ver Figura 5).

```
if opt in ("--ip"):
    resultado=obtener_datos_por_ip(arg)
    resultado=resultado.upper()
    obtener_datos_por_mac(resultado)
    sys.exit()
```

Figura5

Si se ingresa *--arp*, se ejecuta la funcion *obtener_tabla_arp*. que entrega los fabricantes de las direcciones MAC asociadas a dicha tabla (figura 6).

```
elif opt in ("--arp"):
    obtener_tabla_arp()
```

Figura 6

Si se ingresa el parametro --help o se inicializa el archivo sin parametros se mostrara por pantalla el apartado –help (figura 7).

```
    elif opt in ("--help"):
        print("Use: python OUILookup.py --ip <IP> | --mac <IP> | --arp | [--help] \n --ip : IP del host a c
        sys.exit()
print("Use: python OUILookup.py --ip <IP> | --mac <IP> | --arp | [--help] \n --ip : IP del host a consultar
sys.exit()
```

Figura 7

4. Diagrama de Flujo

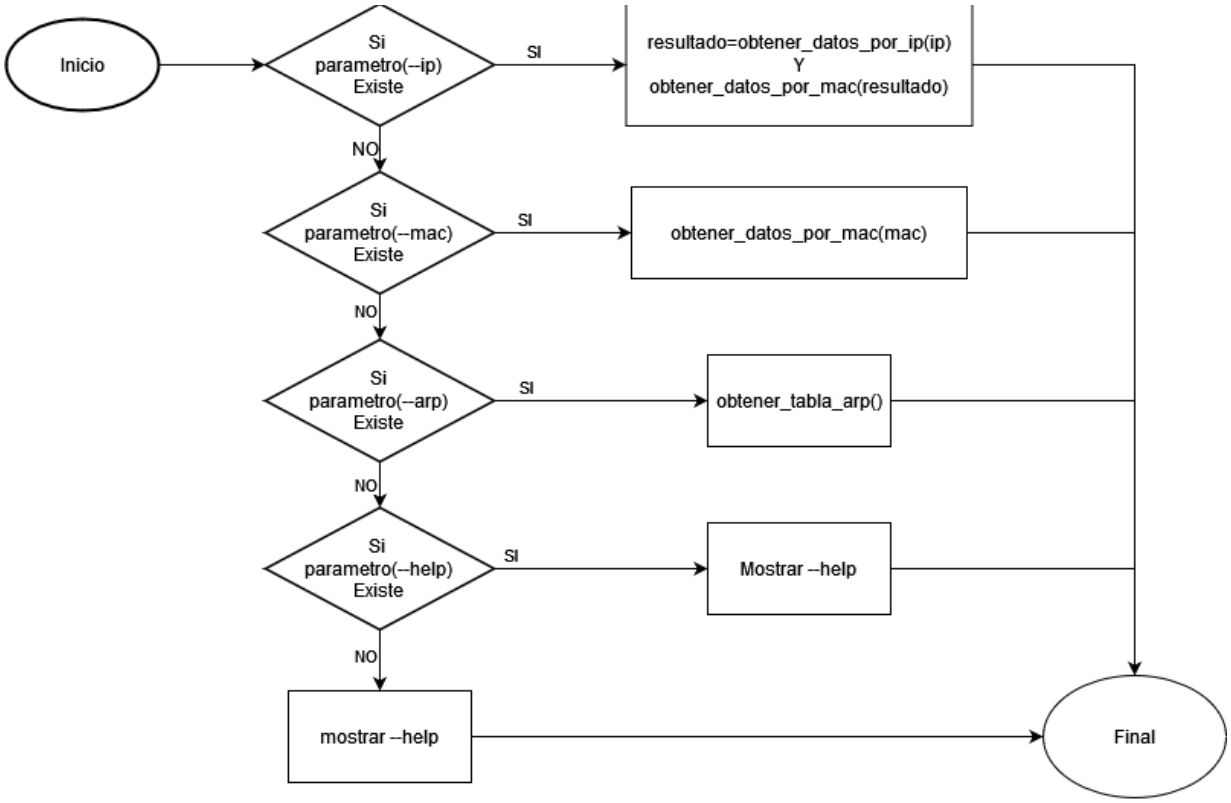

Figura 8

5. Discusión y conclusiones

Como se pudo Analizar, El código se enfoca en obtener y analizar información sobre direcciones MAC y fabricantes asociados a ellas. Se hace uso de comandos del sistema operativo y manipulación de archivos para llevar a cabo esta tarea a traves de funciones o sea que se creó una herramienta util para obtener informnacion sobre las direcciones MAC y sus fabricantes que puede tener multiples razones para saber aquella informacion.