Tarea 03 Redes de Computadores

Matias Rivas Caceres matias.rivas@alumnos.uv.cl

Nataniel Palacios Toro nataniel.palacios@alumnos.uv.cl

Vicente Miralles Olivares vicente.miralles@alumnos.uv.cl

1. Introducción

El presente informe tiene como objetivo proporcionar una visión general y una explicación detallada de un programa desarrollado en Python para realizar diagnósticos en una red local y poder obtener información consultando una API. Este programa, diseñado para ser ejecutado desde la línea de comandos, se centra en la obtención de información crucial sobre dispositivos en una red, específicamente, la dirección MAC y el fabricante asociado.

2. Materiales y Métodos

Para esta tarea se necesita un computador con un editor de texto cualquiera, aunque en esta ocasión utilizó un IDE llamado Visual Studio Code para realizar el código. Además es necesario una API que obtenga información al pasarle una dirección MAC. El método *subprocess* para obtener información a través de la línea de comandos.

3. Resultados

El código comienza definiendo una variable global *computadorIP* (Figura 1).

```
global computadorIP
computadorIP = "192.168.1.30"
```

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).

```
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
            (variable) resultado: Any k_output(["arp", "-a", ip])
                                      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 la *funcion datos_mac(mac_address)* con el propósito de entregar el fabricante de la tarjeta de red y la latencia entre la API REST y nuestro computador, para lograr esto nos conectamos a través de una API que nos brindará información sobre los fabricantes de cada MAC que exista en esta base de datos, adicionalmente como se mencionó, se implementa un algoritmo para medir la latencia que existe entre la API REST y nuestro computador, a continuación se implementa una condición para que se ejecute el programa siempre y cuando el código de estado de la respuesta sea 200, en el caso contrario, devuelve un error de conexión, si se logra la conexión nuevamente consultamos a que si la MAC ingresada como parámetro se encuentra en la API, entregamos como respuesta el fabricante asociado con esa dirección y la misma dirección MAC ingresada, en el caso contrario de esto, nos devolverá como resultado de que no existe aquella dirección dada (Ver figura 3).

```python
def datos_mac(mac_address):
    try:
        url = f'https://api.maclookup.app/v2/macs/{mac_address}'

        start_time = time.time()

        response = requests.get(url)

        end_time = time.time()
        response_time = end_time - start_time


        print(f'Status Code: {response.status_code}')

        if response.status_code == 200:
            if response.json()['found'] == True:
                empresa = response.json()['company']
                print(f'MAC address: {mac_address}')
                print('Fabricante:', empresa)
                print(f'Tiempo de respuesta: {response_time:.4f} segundos')
            else:
                print(f'MAC address: {mac_address}')
                print('Fabricante: no existe')
                print(f'Tiempo de respuesta: {response_time:.4f} segundos')
    except Exception as e:
        print(f'Error: {e}')
```

Figura 3

Se define la función *tabla_arp()*, esta busca muestra la dirección IP, MAC y el posible fabricante asociado a cada dispositivo en la red local.

Para poder lograr esto se obtiene la tabla ARP de un sistema utilizando el módulo *subprocess.* Luego se consulta la API para obtener la información sobre el fabricante de cada dirección MAC en la tabla. Finalmente, muestra las direcciones IP, MAC y fabricantes correspondientes en un formato "IP/MAC/Fabricante". Si no se encuentra información sobre el fabricante, muestra "Not found" (Ver Figura 4).

```python
def 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]

            try:
                url = f'https://api.maclookup.app/v2/macs/{mac}'
                response = requests.get(url)

                if response.status_code == 200 and response.json()['found'] == True:
                    response_json = response.json()
                    fabricante = response_json['company']
                else:
                    fabricante = "Not found"
            except Exception as e:
                fabricante = str(e)

            print(f"{ip} / {mac} / {fabricante}")
    except Exception as e:
        print(f"Error: {e}")
```

Figura4

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

Si el parámetro ingresado es una dirección MAC, el programa corre la función *obtener_datos_por_mac()* (ver Figura 5).

```python
for opt, arg in opts:
    if opt in ("--mac"):
        datos_mac(arg)
        sys.exit()
    elif opt in ("--ip"):
        resultado=obtener_datos_por_ip(arg)
        datos_mac(resultado)
        sys.exit()
```

Figura 5

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

```python
elif opt in ("--arp"):
    tabla_arp()
    sys.exit()
```

Figura 6

Si se ingresa el parámetro *--help* o se inicializa el archivo sin parámetros se mostrará por pantalla el apartado *–help* (figura 7).

```
elif opt in ("--help"):
    print("Use: python OUILookup.py --mac <MAC> | --ip <IP> | --arp | --help \n --mac: MAC a consultar. P.e. aa:bb:cc:00:00:00.  \n --
    sys.exit()
```

Figura 7

4. Mediciones

Matias Rivas:

```
PS C:\Users\Matia\OneDrive\Escritorio\OUILookup> python3 prueba2.py --ip 192.168.1.1
Status Code: 200
MAC address: fc-12-63-11-71-40
Fabricante: ASKEY COMPUTER CORP
Tiempo de respuesta: 0.9314 segundos
PS C:\Users\Matia\OneDrive\Escritorio\OUILookup> python3 prueba2.py --mac fc-12-63-11-71-40
Status Code: 200
MAC address: fc-12-63-11-71-40
Fabricante: ASKEY COMPUTER CORP
Tiempo de respuesta: 1.0890 segundos
PS C:\Users\Matia\OneDrive\Escritorio\OUILookup>
```

Nataniel Palacios:

```
C:\Users\eddie\OneDrive\Documentos\Universidad\Redes_de_computadores\Tarea3> python OUILookup.py --ip 192.168.100.1
Error: ip is outside the host network

C:\Users\eddie\OneDrive\Documentos\Universidad\Redes_de_computadores\Tarea3> python OUILookup.py --mac 98-44-ce-e7-ca-ed
Status Code: 200
MAC address: 98-44-ce-e7-ca-ed
Fabricante: HUAWEI TECHNOLOGIES CO.,LTD
Tiempo de respuesta: 1.7111 segundos
```

Vicente Miralles:

```
PS C:\Users\vigam\OneDrive\Escritorio\U\U 2023\Segundo Semestre\Redes\Tarea03> ./Tarea03.py --ip 192.168.1.1
Status Code: 200
MAC address: 7c-db-98-4e-b1-1a
Fabricante: ASKEY COMPUTER CORP
Tiempo de respuesta: 0.9310 segundos
PS C:\Users\vigam\OneDrive\Escritorio\U\U 2023\Segundo Semestre\Redes\Tarea03> ./Tarea03.py --mac 7c-db-98-4e-b1-1a
Status Code: 200
MAC address: 7c-db-98-4e-b1-1a
Fabricante: ASKEY COMPUTER CORP
Tiempo de respuesta: 0.9343 segundos
```

5. Preguntas a responder

a) ¿Qué es REST? ¿Qué es una API?

REST (Representational State Transfer) es cualquier interfaz entre sistemas que usen el protocolo HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

una API (application programming interface) es un conjunto de reglas que describen como una aplicación puede interactuar con otra.

b) ¿Cómo se relaciona el protocolo HTTP con las API REST y cuál es su función en la comunicación entre clientes y servidores?

el protocolo HTTP con las API REST se relacionan entre si debido a que dentro de las API se utilizan los métodos del protocolo tales como GET, POST, PUT y DELETE, y su función:

GET: Solicita datos de un recurso especifico a un servidor

POST: envía datos en el cuerpo de la solicitud del protocolo.

DELETE: Solicita la eliminación de un recurso en específico en un servidor

PUT: Solicita actualización o creación de un recurso especifico en un servidor.

c) ¿Qué papel juega la dirección IP en el acceso a recursos a través de una API REST?

la dirección IP es importante para la parte del cliente que acceden a los recursos, a esto nos referimos para identificar el cliente (identificar el origen de una solicitud al servidor), controlar el acceso (permitir y bloquear solicitudes según la dirección IP del cliente) y entre muchas otras funcionalidades al respecto de seguridad para saber quiénes realizan las solicitudes al servidor a través de las direcciones IP.

d) ¿Por qué es importante considerar la latencia de red y el ancho de banda? ¿Cómo afectan estos factores al rendimiento de la API?

Una latencia de red alta puede afectar en los tiempos de respuesta de la API, esto puede provocar retardo en la comunicación de servicios que hagan uso de esta y afectar negativamente la experiencia del usuario de estos servicios.

Un ancho de banda que sea limitado puede provocar congestión en la red y esto a su vez retrasos en la transferencia de datos, si una API genera una gran cantidad de datos o tiene un uso intensivo, necesita un ancho de banda que de abasto para lograr esta transferencia de manera efectiva, sino sus respuestas serán lentas o puede que no haya respuestas en absoluto.

e) ¿Por qué el programa desarrollado utilizando API REST es más lenta su ejecución?

El programa desarrollado es más lenta su ejecución al utilizar API REST ya que al funcionar a traves del protocolo HTTP existe una latencia por la comunicación que existe entre el cliente y servidor y por los datos que se transfieren desde la API que ralentiza la ejecución del software.

f) ¿Cuál es la diferencia entre la dirección MAC (Media Access Control) y la dirección IP, y en qué capa de la red se utilizan cada una de ellas?

La dirección MAC sirve para identificar de manera única un dispositivo en una red local y se utiliza para la conección de dispositivos en esta misma red local, mientras que la dirección ip sirve para identificar de manera única cada dispositivo en una red más grande, sirve para que los datos se enruten a través de redes locales interconectadas y direccionar el tráfico.

La dirección MAC opera en la capa de enlace de datos (L2) mientras que la dirección ip opera en la capa de red (L3).

g) ¿Cómo pueden las redes LAN (Local Area Networks) y WAN (Wide Area Networks) afectar la accesibilidad y la velocidad de respuesta de una API REST?

Las redes LAN y WAN pueden afectar en la accesibilidad y respuesta de una API REST. Algunas de las maneras en que puede afectar es en la distancia física de forma en que las redes LAN como son locales, la latencia es baja mientras que en una red WAN la latencia puede ser más alta debido a la distancia física que hay entre los dispositivos. Otra manera en que puede afectar la accesibilidad es la disponibilidad, en las redes LAN suelen ser más estables ya que la probabilidad de la perdida de paquetes y problemas de conectividad es muy baja, mientras que en las WAN como son redes amplias, son más propensas a tener fallos de conectividad, perdida de paquetes, entre otros, hablando en entornos de internet.

h) ¿Qué es un enrutador y cómo se utiliza para dirigir el tráfico de datos? ¿Qué relación tiene esto con el enrutamiento de solicitudes en una API REST?

un enrutador es un dispositivo de red que conecta redes y dirige el tráfico de datos entre ellas. Esto lo hace teniendo una tabla de enrutamiento que contiene información sobre las rutas disponibles y como llegar a ellas en donde además toma decisiones del reenvío de datos, examinando la dirección IP de destino del paquete para determinar el mejor camino hacia el destino.

Esto tiene relación con el enrutamiento de solicitudes de una API REST, ya que, el enrutamiento de solicitudes se refiere a dirigir solicitudes HTTP a los recursos de un servidor mapeando rutas URL a controladores o servicios específicos, así mismo, además también se pueden redireccionar las solicitudes a diferentes partes de una aplicación a través de la URL, como por ejemplo http://aplicacion.com/api/clientes.

Diciendo todo esto, ambas "definiciones" comparten la función de dirigir el tráfico de manera eficiente hacia su destino.

i) ¿Cómo se asocian los puertos de red con servicios y aplicaciones específicas?

Los puertos de red con servicios y aplicaciones específicas se asocian mediante la cantidad de puertos asignados de manera estándar, a través de estos puertos definidos se pueden identificar los servicios y aplicaciones dentro de una red permitiendo que múltiples servicios se ejecuten simultáneamente en un dispositivo.

Existen los puertos conocidos los cuales están reservados para servicios y aplicaciones estandarizadas, están enumerados del 0 al 1024, como por ejemplo en el puerto 53 está DNS.

Por otro lado, también existen los puertos registrados, que están asignados a servicios y aplicaciones de internet por la IANA y están enumerados del 1024 al 49151, como por ejemplo en el puerto 3306 está asociado con el servicio de base de datos MYSQL.

6. Discusión y conclusiones

Este programa es una herramienta versátil para administradores de redes y profesionales de TI que deseen diagnosticar y administrar dispositivos en una red local. Proporciona detalles valiosos de dispositivos, verifica la pertenencia a la misma red y ofrece información sobre el fabricante asociado. Su facilidad de uso desde la línea de comandos lo convierte en una herramienta práctica y efectiva.

7. Referencias

https://maclookup.app/api-v2/documentation