Redes de Computadores, 2020

Tarea 1: Implementación de un OUI Lookup Tool

Escuela de Ingeniería Civil Informática

Universidad de Valparaíso

Integrantes: - Eliezer Zuñiga
-Aracelly Balboa
-Maximiliano Espíndola

Esta herramienta está implementada para el sistema operativo Windows y Linux.

Para la programación se utilizó el intérprete Python.

Recursos utilizados

Las siguientes librerías y módulos serán necesariamente utilizadas e implementadas para la ejecución del código (si no se encuentra instalada alguna de las siguientes librerías o módulos, el código no funcionará):

- **getopt**: se usa para analizar las líneas de comando y ocuparlas como secuencias de argumentos.

- **sys**: es un módulo utilizado para funciones que interactúan con el intérprete.

- **requests**: es una librería para HTTP, y se utiliza para realizar una solicitud al URL donde se encuentra la base de datos de las direcciones MAC.

- **getmac**: se utiliza para obtener la dirección MAC de los hosts de la red local.

Forma de instalar las librerías
Si no se tienen las librerías anteriormente mencionadas se pueden instalar con los siguientes comandos:

OUI VENDORS

Para obtener las direcciones MAC de los proveedores de OUI (Organizationally Unique Identifier) o la misma empresa se utiliza el siguiente link:

https://gitlab.com/wireshark/wireshark/-/raw/master/manuf

En caso de no obtener conexión a internet o la URL no se encuentra se cubrirá con un archivo con la misma información para analizar las direcciones MAC y la empresa o fabricante.

Explicación del funcionamiento

El código está estructurado mediante cuatro funciones:

1. ***MAIN***

- ***Descripción General:***

La función principal se encarga de recibir los parámetros ingresados en la ejecución del programa, tales como *--mac*, *--ip* y *--help*. Aquellos son los indicadores esenciales que la herramienta necesita y las órdenes que genera la función, ya sea obtener el empresa o fabricante dada una dirección mac, el registro de una dirección mac perteneciente a la red local mediante una ip, y un minúsculo manual de indicaciones para el uso de la herramienta *OUILookup*.

- ***Descripción estructural y específica:***

```
10  # CUERPO PRINCIPAL :
11  def main():
12      try:
13          if (len(sys.argv)==1):
14              help()
15              return(0)
16          options, args = getopt.getopt(sys.argv[1:],"ip,mac",['ip=','mac=','help'])
17
18      except:
19          print("\n ¡Error!: Parametros incorrectos.")
20          help()
21          return(1)
```

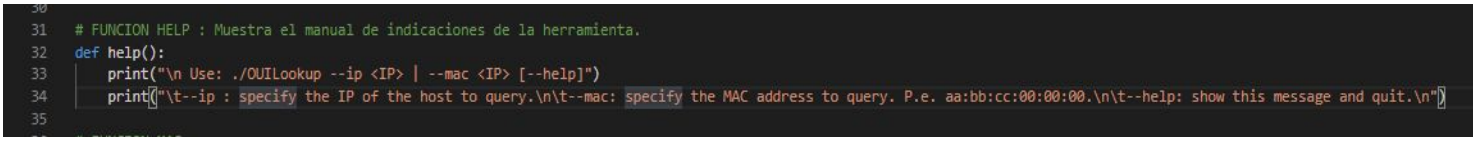
Figura 1

Los parámetros pasados en la ejecución son almacenados en un vector de arreglos de caracteres. Aquel vector llamado *argv* se obtiene del módulo *sys* que provee accesos a funciones y objetos mantenidos por el intérprete, donde *sys* es importado al inicio del fichero py.

Las opciones macros que se establecen al ejecutar el programa se puede optar a ejecutar con un solo parámetro (el mismo nombre del fichero py) o más los parámetros (*--mac*, *--ip* y *--help*). Sin embargo habrá casos posibles en que aquellos comandos de tipo BASH no sean correctos o no haya reconocimiento en la función para las acciones. La solución es utilizar la excepción mostrada en la ***Figura 1***. Inicialmente se explicará el tratamiento de los casos de ingreso de parámetros en el bloque de *ln.12* en ***Figura 1***:

- **Try**: El bloque en cual pertenece al tratamiento de los parámetros, primero analiza si el programa arranca sin los parámetros adicionales (*--mac, --ip*). Esto lo garantiza la función *len()* para la cantidad de elementos que contiene el vector. Dado que el nombre del fichero es el primer elemento en cual contiene *argv*, la sentencia de la *ln.13* verifica si el vector contiene *solo* aquel argumento. En aquel caso la instrucción es llamar a la función *help()* en

Figura 2 e indicar los comandos para las utilidades en cual acciona la herramienta. Luego de exponerlo, se detiene el programa con la acción *return(0)* indicando que todo se ha ejecutado de forma exitosa.



```
30
31  # FUNCION HELP : Muestra el manual de indicaciones de la herramienta.
32  def help():
33      print("\n Use: ./OUILookup --ip <IP> | --mac <IP> [--help]")
34      print("\t--ip : specify the IP of the host to query.\n\t--mac: specify the MAC address to query. P.e. aa:bb:cc:00:00:00.\n\t--help: show this message and quit.\n")
35
```

Figura 2

Si surge el otro caso donde se den los comandos o parámetros de tipo BASH, aquellos son analizados y organizados por el módulo *getopt.* De aquel se obtiene una función **getopt()* en cual recibe 3 parámetros para el análisis de la secuencia y la función devuelve una secuencia de tuplas que contiene pares(opción, argumento). El primer argumento proviene del vector *sys.argv* donde contiene los parámetros BASH pasados en la línea de comandos en la ejecución (ignorando el nombre del programa *sys.argv[0:]*). El segundo y tercer argumento de la función * se definen las opciones o los parámetros que van después del nombre del ejecutable. La diferencia existe en que el segundo argumento son opciones de un carácter y el tercero pueden existir opciones en cadenas de caracteres como *--mac o --ip*. Para las opciones de cadena debe integrar un '=' si el parámetro debe llevar argumento adicional. En *ln.13* de ***Figura 1*** el tercer argumento es una lista de los parámetros en cadenas, donde *ip* y *mac* se les deben pasar el argumento de direcciones ip y mac. *help* interesa en mostrar el manual de comandos.

- **Except**: En el caso que el usuario ingrese comandos no identificados en el análisis de secuencias de parámetros de la función *getopt()*, se lanzará la excepción mostrando un mensaje de *error de parámetros* y luego llamando a la función help() indicando los comandos que la herramienta reconozca.

Suponiendo que el ingreso de parámetros es correcto, la función principal hará el llamado a otras funciones que conforman al programa de acuerdo al paso de parámetro o estos con sus argumentos (*ip, mac*) conforme a lo que el usuario desea obtener con la herramienta. La ***Figura 3*** muestra las sentencias que debe implementar el programa.

```
23      for opt, arg in options:
24          if opt in ('--help'):
25              help()
26          elif opt in ('--ip'):
27              if(verificarip(arg)):
28                  ip(arg)
29              else:
30                  print("ERROR EN ARGUMENTOS")
31
32          elif opt in ('--mac'):
33              if(verificarmac(arg)):
34                  mac(arg)
35              else:
36                  print("ERROR EN ARGUMENTOS")
37
```

Figura 3

En aquel ciclo *for* se utilizan las variables *opt* y *arg* como el par opcion-argumento en cual se adentran a la secuencia de tuplas que almacena la variable *options.* Las tres sentencias indican si el usuario dio el comando de ayuda (*--help*) para la muestra de manual o al igual que *--ip* o *--mac* indicando que quiere encontrar la dirección mac y fabricante de área local de acuerdo a la ip como argumento o el fabricante con la dirección mac respectivamente. En *ln.27* y *ln.33* de la ***Figura 3*** se llaman a las funciones de verificación de argumento, ya sea para la mac e ip. Aquellas devuelven *cierto* si la dirección mac tienen la estructura y datos propios de la notación hexadecimal al igual que la dirección ip, verdadera tanto el número de dígitos que deba contener y el rango de valores de cada símbolo ip.

2. *MAC*

Dada la una MAC ingresada, se pasará como parámetro a esta función la cual se buscará en la URL del la base de datos de direcciones o se buscará en el archivo dependiendo si se tiene conexión a internet o no, esto funciona de la siguiente manera (se utiliza la librería requests para las funciones de URL):

Se intentará hacer un “request” a la página de la base de datos de direcciones MAC con la función:

```
respuesta = requests.get(url)
```

Y se obtendrá la respuesta al sitio web solo si el código de estatus es de 200. Este código se encontrará con la función:

```
respuesta.status_code
```

Al tener éxito en la conexión se abrirá un archivo y en él se escribirá el contenido de las direcciones. De otro modo se obtendrá una excepción que será dependiendo de

varias condiciones para la conexión (ej: Página no encontrada, Servidor caído de la página, problemas con la conexión a internet, etc.). Esto se visualiza mejor en la **Figura 4.**

```
def mac(macAdress):
    try:
        url = "https://gitlab.com/wireshark/wireshark/-/raw/master/manuf"       # url de base de datos de direcciones MAC
        respuesta = requests.get(url)                                           # se obtiene respuesta de sitio web
        if (respuesta.status_code==200):                                        # si hubo respuesta
            contenido = respuesta.content                                       # se guarda el contenido
            file = open("direcciones.txt", "wb")                                # se genera unsdsd archivo txt
            file.write(contenido)                                               # se escribe el contenido en el archivo
            file.close()
        pass

    #En caso de que no exista conexion
    #Para ejecutar sin conexion se requiere que el archivo de direcciones se encuentre en la misma carpeta que el archivo.py
    except:
        print("\n\t404 Not Found.")
```

Figura 4

En caso de no tener conexión a la página será necesario tener explícitamente el archivo con las direcciones el cual se encontrará en la misma carpeta donde está el código. Si se tiene una conexión a internet en el dispositivo se guardará línea por línea los datos de la página en un archivo de tipo byte y este se podrá utilizar de forma off-line cuando se escriba. Después se abrirá el archivo y se obtendrán los datos linea por linea, los cuales se encuentran separados por una tabulación. Se compara la dirección MAC ingresada por parámetros con cada MAC de los fabricantes en el archivo, si encuentra la que se busca entonces se mostrará por pantalla la MAC(MAC Address) buscada y el fabricante(Vendor):

```
print("\n\tMAC address\t:",macAdress)
print("\tVendor    \t:",vendor[1])
```

Si no se encuentra se mostrará por pantalla que no se encontró la MAC del fabricante(`print("\tVendor   \t: Not found\n")`). Esto se puede visualizar mejor en la **Figura 5.**

```
#Se abre el archivo para analizar linea por linea las respectivas direcciones mac.
finally:
    file = open("direcciones.txt", encoding="utf8")        #Se abre el archivo, UTF-8 es una codificación de caracteres de ancho variabl
    encontrada=False

    while(True):
        linea = file.readline()
        datos = linea.split("\t")          #Se obtienen los datos de la linea, los cuales se encuentran separados por una tabulacion
        dirMac = linea[0:8]
        macAdress = macAdress.upper()

        if (dirMac==macAdress[0:8]):
                vendor = datos[1:]                    #Datos del fabricante se encuentran desde la posicion 1 en adelante
                print("\n\tMAC address\t:",macAdress)
                print("\tVendor   \t:",vendor[1])
                encontrada=True
                break

        if not linea:           #Si no hay mas lineas
            break               #Se rompe el ciclo

    file.close()

    if not encontrada:
        print("\n\tMAC address\t:",macAdress)
        print("\tVendor  \t: Not found\n")
```

Figura 5

3. IP

Para la función IP se ocupa el módulo getmac y la función principal utilizada es:

```
dirMac = get_mac_address(ip = dirIp)
```

get_mac_address() funciona utilizando el protocolo ARP el cual tiene por objetivo permitir a un dispositivo conectado a una red LAN obtener la dirección MAC de otro dispositivo conectado a la misma red LAN cuya dirección IP es conocida enviando un paquete (*ARP request*) a la dirección de la red . Al encontrar esta dirección MAC en la red la devuelve y se utiliza en la función mac() mencionada anteriormente para buscarla en el archivo o de forma online.

Si la IP buscada no pertenece a la misma red del dispositivo se muestra en pantalla que la IP está fuera de la red del host. Esto se puede visualizar mejor en la **Figura 6.**

```
# Funcion IP
# Se utilizo la libreria getmac para encontrar la MAC de la IP ingresada, si existe la IP dentro de la red local,
# se registra dicha direccion MAC para luego utilizar la función MAC para encontrar a su fabricante.
def ip(dirIp):
    try:

        dirMac = get_mac_address(ip = dirIp)
        mac(dirMac.upper())
    except:
        print("\n\t¡Error! ip is outside the host network.\n")
```

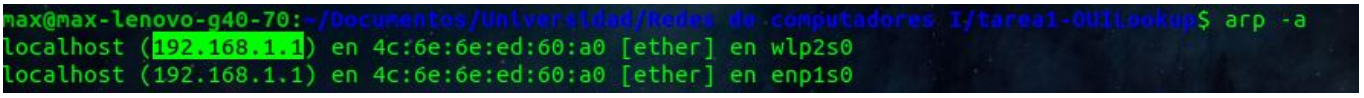
Figura 6

Testeo

Como ejemplo para verificar la ejecución exitosa del programa se utilizaron las siguientes direcciones MAC y IP:

- **Caso de IP que pertenezca a su misma red**

Para esto necesitamos buscar una IP en la tabla de direcciones de la red, esto se hace ejecutando:



Como se puede observar, se copia una dirección y se utiliza para la ejecución del código.

Entonces se puede encontrar la dirección MAC y el fabricante el cual está asociado.

- **Caso de IP que NO pertenezca a su misma red**

Para este caso se utiliza una IP cualquiera que no sea perteneciente a la red. por ejemplo:

ya que en la tabla ARP no se encuentra esta IP.

- **Caso de MAC que esté en la base de datos**

En este caso la MAC tiene que ser específicamente de 6 dígitos ya que si fueran de 12 dígitos se muestra un error en pantalla.

- **Caso de MAC que no esté en la base de datos**

```
max@max-lenovo-g40-70:~/Documentos/Universidad/Redes de computadores I/tarea1-OUILookup$ python3 OUILookup.py --mac b4:b5:fe

        MAC address     : B4:B5:FE
        Vendor          : Not found
```

En este caso la MAC buscada no se encuentra en el archivo, entonces no se encuentra su fabricante.

Diagrama De Flujo