Proyecto en Python: Escaneo de redes.

Javier Ponferrada López, Miguel Ángel Gavilán Merino,
Guillermo Boquizo Sánchez

Lenguaje de Marcas y Sistemas de Gestión de Información
1º de Desarrollo de Aplicaciones Web
I.E.S Gran Capitán, Junio 2017

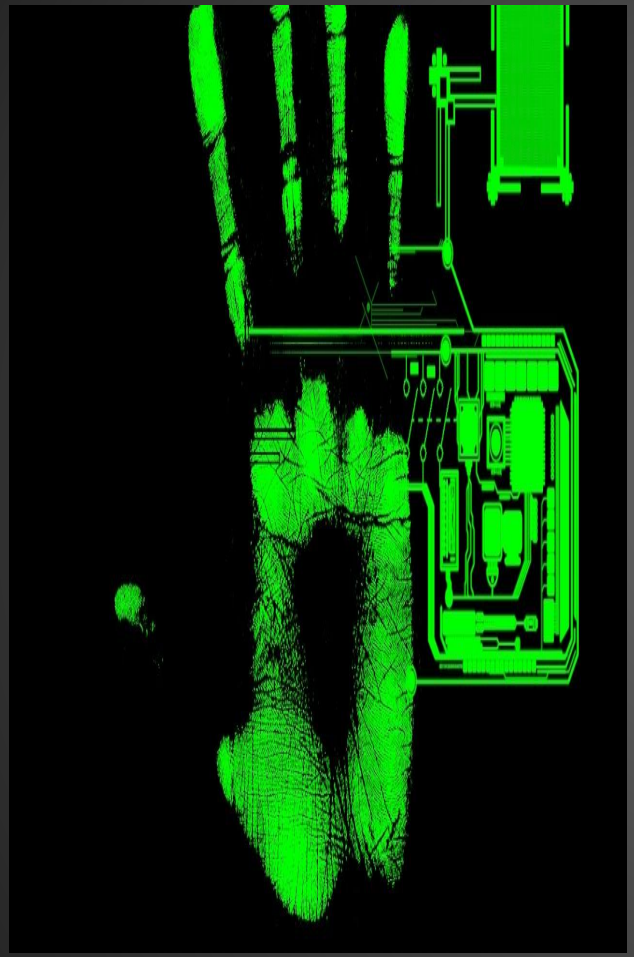
ÍNDICE

1. Presentación de motivos:

La presente documentación se encarga de dar cuenta del propósito del proyecto, de los objetivos que se han perseguido, así como trata de explicar la funcionalidad de la aplicación resultante tras la elaboración de la misma en Python.

Este proyecto surge como resultado del trabajo colaborativo de las personas citadas, con fines pedagógicos.

2. ¿Qué es el escaneo de redes?



Nmap: presentación

El escaneo de redes que vamos a realizar es similar al que se realiza empleando el comando nmap

Nmap es muy usado por todo aquel interesado por las tareas de seguridad y hacking en general.

Las técnicas de escaneo que usa Nmap han sido ya implementadas en sistemas de detección de intrusos y cortafuegos, ya que los desarrolladores de sistemas de seguridad también usan Nmap en su trabajo y toman medidas.

No obstante, pese a estar ampliamente documentado su funcionamiento, hay formas de escaneo que lo hacen difícil de detectar cuando se trata de obtener información.

Nmap: características

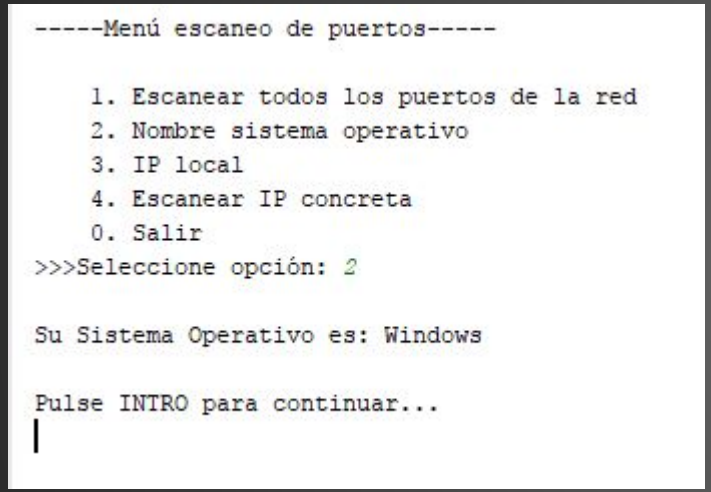
Nmap es una aplicación multiplataforma usada para explorar redes y obtener información acerca de los servicios, sistemas operativos y vulnerabilidades derivadas de la conjunción de éstos.

Algunas de sus características principales son:

- Descubrimiento de servidores: Identifica computadoras en una red, por ejemplo listando aquellas que responden ping
- Identifica puertos abiertos en una computadora objetivo.
- Determina qué servicios está ejecutando la misma.
- Determinar qué sistema operativo y versión utiliza dicha computadora, (esta técnica es también conocida como fingerprinting).
- Obtiene algunas características del hardware de red de la máquina objeto de la prueba.

3. Y desde nmap, nuestro propio escaneador de redes

Aplicación



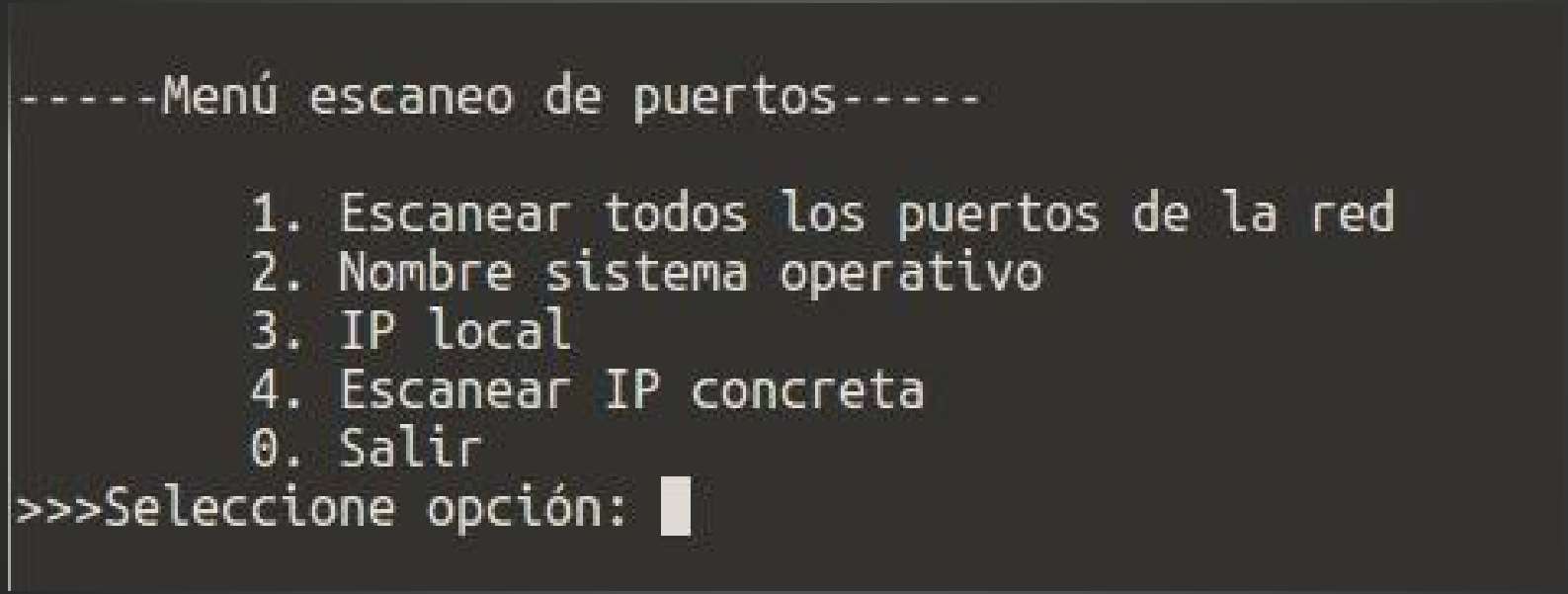
Así, partiendo del uso de nmap, nuestro proyecto se basa en la creación de sendos scripts en Python que permitan recoger los resultados que este comando genera, almacenarlos en un fichero, y una vez obtenida la información de la red escaneada, volcarlos en una base de datos MySQL.

Para ello, hemos implementado las siguientes opciones, recogidas en un menú dentro de uno de los scripts:

1. Escanear todos los puertos de la red
2. Nombre sistema operativo
3. IP local
4. Escanear IP concreta
0. Salir

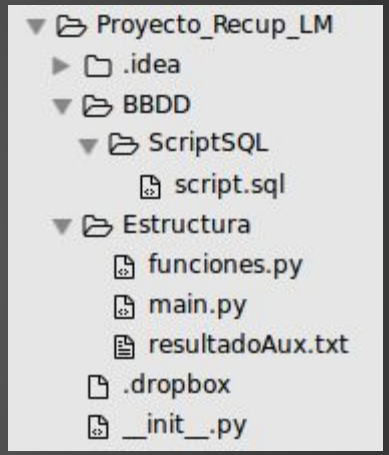
Menú escaneo de puertos

Ésta es la visualización de las citadas opciones al ejecutar el script:



Estructura general proyecto

Esta imagen visualiza la estructura básica del proyecto, donde se identifica qué archivos se generan y con qué propósito



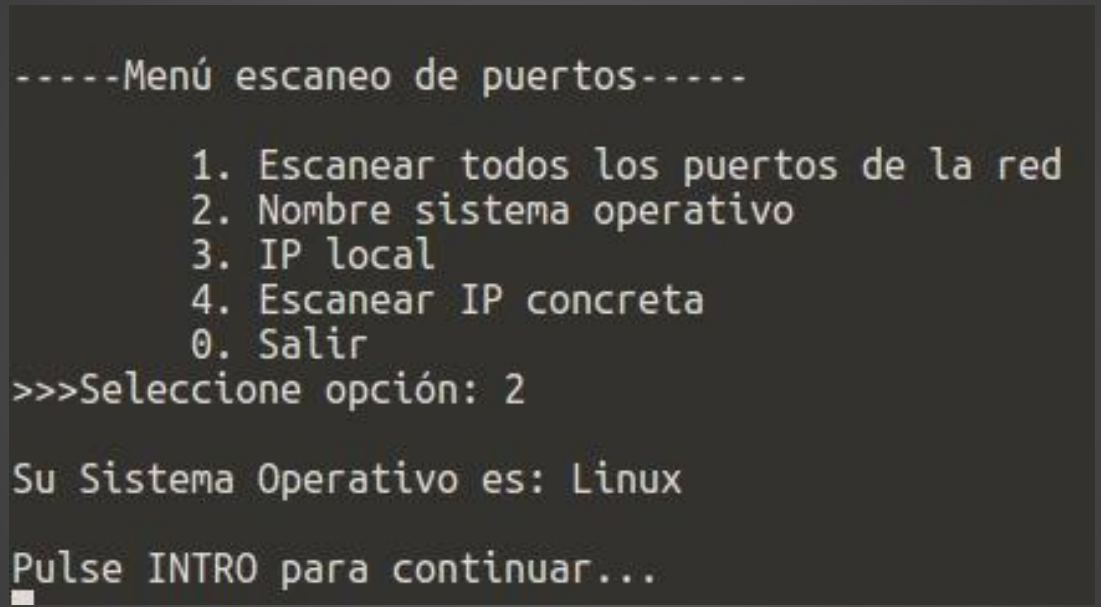
Opción 1: Escanear todos los puertos de la red

A continuación, podremos observar la ejecución de un barrido de cualquier red disponible y su resultado:

```
-----Menú escaneo de puertos-----

        1. Escanear todos los puertos de la red
        2. Nombre sistema operativo
        3. IP local
        4. Escanear IP concreta
        0. Salir
>>>Seleccione opción: 1

Escaneando red: 192.168.3.0/24 ...
Los hosts a escanear son:  ['192.168.3.1', '192.168.3.100', '192.168.3.101']

Escaneando IP: 192.168.3.1 ...

HOST            PUERTO          PROTOCOLO               SERVICIO
192.168.3.1     22              ssh                     Dropbear sshd
192.168.3.1     80              http                    TP-LINK WR841N WAP http
config
192.168.3.1     1900            upnp                    ipOS upnpd
192.168.3.1     49152           http                    Huawei HG8245T modem htt
p config
None

Escaneando IP: 192.168.3.100 ...

HOST            PUERTO          PROTOCOLO               SERVICIO
None

Escaneando IP: 192.168.3.101 ...

HOST            PUERTO          PROTOCOLO               SERVICIO
192.168.3.101   80              http                    Apache httpd
192.168.3.101   443             http                    Apache httpd
192.168.3.101   3306            mysql                   MariaDB
192.168.3.101   17500           db-lsp
None

Pulse INTRO para continuar...
```

Opción 2: Nombre del sistema operativo

La siguiente opción nos devuelve el sistema operativo de la máquina desde la que se opera el script



Opción 3: IP Local

La siguiente opción nos devuelve la dirección IP de la máquina desde la que se opera el script

```
-----Menú escaneo de puertos-----

        1. Escanear todos los puertos de la red
        2. Nombre sistema operativo
        3. IP local
        4. Escanear IP concreta
        0. Salir
>>>Seleccione opción: 3

Su IP local es:  192.168.3.101

Pulse INTRO para continuar...
```

Opción 4: Escanear IP concreta

La siguiente opción nos permite escanear la dirección IP que el usuario desee introducir

```
-----Menú escaneo de puertos-----

        1. Escanear todos los puertos de la red
        2. Nombre sistema operativo
        3. IP local
        4. Escanear IP concreta
        0. Salir
>>>Seleccione opción: 4

IP:(xx.xx.xx.xx) ---> 192.168.3.101

Escaneando IP: 192.168.3.101 ...

HOST            PUERTO          PROTOCOLO               SERVICIO
192.168.3.101   80              http                    Apache httpd
192.168.3.101   443             http                    Apache httpd
192.168.3.101   3306            mysql                   MariaDB
192.168.3.101   17500           db-lsp
None

Pulse INTRO para continuar...
```

Opción 4b: Escanear IP concreta 2: Actuando el patrón regex

Para controlar que la IP siempre sea una IP válida, empleamos un patrón regex. Aquí lo vemos actuar:

```
-----Menú escaneo de puertos-----

        1. Escanear todos los puertos de la red
        2. Nombre sistema operativo
        3. IP local
        4. Escanear IP concreta
        0. Salir
>>>Seleccione opción: 4

IP:(xx.xx.xx.xx) ---> 1111111
IP incorrecta.

Pulse INTRO para continuar...
```

4. Fantasma en la máquina: Destripando el código

main.py

Las siguientes diapositivas muestran el contenido del script Main.py, script principal que contiene el menú, desde el cual se ejecutan las funciones contenidas en otro script que se mostrará a continuación.

```
1  # -*- coding: utf-8 -*-
2
3  from funciones import *
4
5
6  def mostrarOpciones():
7      print '\n-----Menú escaneo de puertos-----\n'
8      print '\t1. Escanear todos los puertos de la red'
9      print '\t2. Nombre sistema operativo'
10     print '\t3. IP local'
11     print '\t4. Escanear IP concreta'
12     print '\t0. Salir'
13
14
15 def limpiarPantalla():
16     if os.name == "nt":
17         os.system("cls")
18     elif os.name == "posix":
19         os.system("clear")
20
21 limpiarPantalla()
22 try:
23     mostrarOpciones()
24     opcion = int(raw_input(">>>Seleccione opción: "))
25 except Exception as e:
26     opcion=9
```

```
while opcion != 0:
    if opcion == 1:
        scanTotal()
        print '\nPulse INTRO para continuar...'
        raw_input()
        limpiarPantalla()
    elif opcion == 2:
        print '\nSu Sistema Operativo es: ' + obtenerOS()
        print '\nPulse INTRO para continuar...'
        raw_input()
        limpiarPantalla()
    elif opcion == 3:
        print '\nSu IP local es: ', obtenerIpEquipo()
        print '\nPulse INTRO para continuar...'
        raw_input()
        limpiarPantalla()
    elif opcion == 4:
        print scanIpConcreta(raw_input("\nIP:(xx.xx.xx.xx) ---> "))
        print '\nPulse INTRO para continuar...'
        raw_input()
        limpiarPantalla()
    elif opcion == 0:
        print '\nSALIENDO...\n'
    else:
        print '\n¡¡¡Opción erronea!!!'
        limpiarPantalla()

    mostrarOpciones()
    try:
        opcion = int(raw_input(">>>Seleccione opción: "))
    except Exception as e:
        opcion = 9

```

funciones.py

Las siguientes diapositivas muestran el contenido del script funciones.py, script que contiene en sí la lógica del proyecto, puesto que en él podemos encontrar definidas las múltiples funciones referenciadas en el menú

```
1  # -*- coding: utf-8 -*-
2
3  import socket
4  import sys
5  import os
6  import re
7  import nmap
8  import time
9  import MySQLdb
10
11 #Patrón que controla que la IP sea válida
12 PATRON_IP = re.compile("^([0-9]\.|[1-9][0-9]\.|[1][0-9][0-9]\.|[2][0-5][0-5]\.){3}([0-9]|[1-9][0-9]|[1][0-9][0-9]|[2][0-5][0-5])$")
13
14
15 #Obtiene la ip local del equipo actual
16 def obtenerIpEquipo():
17     s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18     s.connect(("gmail.com",80))
19     cadena = str(s.getsockname()).split('(')
20     cadena = str(cadena[1]).split(',')
21     return str(cadena[0]).replace("'","")
22
```

funciones.py

```python
#Obtiene la ip local del equipo actual
def obtenerIpEquipo():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.connect(("gmail.com",80))
    cadena = str(s.getsockname()).split('(')
    cadena = str(cadena[1]).split(',')
    return str(cadena[0]).replace("'","")


#Obtiene la plataforma del Sistema Operativo
def obtenerOS():
    nombre = sys.platform
    if nombre == 'linux2':
        return "Linux"
    elif nombre == 'win32' or nombre == 'win64':
        return "Windows"
    else:
        return "OSX"

```

funciones.py

```
34  #Guarda los resultados del escaneo de nmap en un fichero auxiliar, que lee posteriormente
35  #y vuelca a una BBDD de MySQL
36  def obtenerDatosNmapScan(resultadoNmap):
37      fichero = open("resultadoAux.txt",'w')
38      fichero.write(resultadoNmap)
39      fichero.close()
40
41      fechaYHora = ""
42      host = ""
43      puerto = ""
44      protocolo = ""
45      servicio = ""
46      fichero = open("resultadoAux.txt",'r')
47
48      lectura = str(fichero.readline()).split(";")
49
50      conn = MySQLdb.connect("127.0.0.1","root","","BBDD_LOGS_RED")
51      cursor = conn.cursor()
52
53      print '\nHOST\t\tPUERTO\t\tPROTOCOLO\t\tSERVICIO'
54
```

funciones.py

```
55 ▼    while True:
56          lectura = str(fichero.readline()).split(";")
57          if lectura == ['']:
58              break
59          host = lectura[0]
60          puerto = int(lectura[4])
61          protocolo = lectura[5]
62          servicio = lectura[7]
63          fechaYHora = str(time.strftime("%d/%m/%y ") + " - " + str(time.strftime("%H:%M")))
64
65          print host, "\t" + str(puerto), "\t\t" + protocolo, "\t\t\t" + servicio
66          cursor.execute(("INSERT INTO REGISTROS(IP,PUERTO,PROTOCOLO,SERVICIO,FECHA) VALUES('%s','%d','%s','%s','%s')") %
            (host,puerto,protocolo,servicio,fechaYHora))
67          conn.commit()
68      conn.close()
69      fichero.close()
70
```

funciones.py

```
71  #Realiza un escaneo total
72  def scanTotal():
73      ip = str(obtenerIpEquipo())
74      ipTrozos = ip.split(".")
75      ipFinal = ""
76      if ipTrozos[0] >= '0' and ipTrozos[0] <= '127':
77          ipFinal = ipTrozos[0]+"." + ipTrozos[1] + "." + ipTrozos[2]+ "." + "0/8"
78      elif ipTrozos[0] >= '128' and ipTrozos[0] <= '191':
79          ipFinal = ipTrozos[0]+"." + ipTrozos[1] + "." + ipTrozos[2]+ "." + "0/16"
80      elif ipTrozos[0] >= '192' and ipTrozos[0] <= '223':
81          ipFinal = ipTrozos[0] + "." + ipTrozos[1] + "." + ipTrozos[2]+ "." + "0/24"
82      else:
83          return "Rango de ip no válido."
84
85      nmapScan = nmap.PortScanner()
86      print "\nEscaneando red: " + ipFinal + " ..."
87
88      nmapScan.scan(ipFinal, '0')
89
90      print 'Los hosts a escanear son: ',
91      print nmapScan.all_hosts()
92
93      for host in nmapScan.all_hosts():
94          print scanIpConcreta(host)
```

funciones.py

```
96  #Realiza un escaneo a partir de una IP concreta, controlada por el patrón regex definido previamente
97  def scanIpConcreta(ip):
98      try:
99          if PATRON_IP.match(ip):
100             print "\nEscaneando IP: " + ip + " ..."
101             nmapScan = nmap.PortScanner()
102             nmapScan.scan(ip, '1-60000')
103             resultado = str(nmapScan.csv())
104             obtenerDatosNmapScan(resultado)
105         else:
106             return "IP incorrecta."
107     except Exception as e:
108         return 'Debe tener instalado NMAP'
109
```

script.sql

Para poder realizar el volcado a una base de datos MySQL, hemos empleado Xampp, y el siguiente script de creación de la tabla Registros

```sql
CREATE TABLE REGISTROS(
  ID INT NOT NULL AUTO_INCREMENT,
  IP VARCHAR(300),
  PUERTO INT NOT NULL,
  PROTOCOLO VARCHAR(300),
  SERVICIO VARCHAR(300),
  FECHA VARCHAR(300),
  CONSTRAINT PK_REGISTROS PRIMARY KEY(ID)
);
```

resultadoAux.txt

De manera previa al volcado en la base de datos, hemos necesitado almacenar la información obtenida en el escaneo en un fichero resultadoAux, que posteriormente leemos para obtener los valores a insertar en la misma

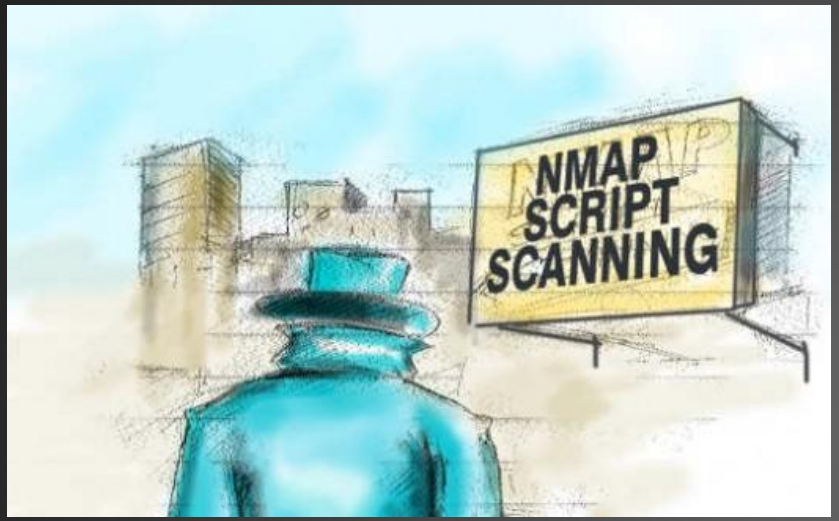
```
1  host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;version;conf;cpe
2  192.168.3.101;;;tcp;80;http;open;Apache httpd;(Unix) OpenSSL/1.0.2j PHP/5.6.30 mod_perl/2.0.8-dev Perl/v5.16.3;syn-
   ack;2.4.25;10;cpe:/a:apache:http_server:2.4.25
3  192.168.3.101;;;tcp;443;http;open;Apache httpd;(Unix) OpenSSL/1.0.2j PHP/5.6.30 mod_perl/2.0.8-dev Perl/v5.16.3;syn-
   ack;2.4.25;10;cpe:/a:apache:http_server:2.4.25
4  192.168.3.101;;;tcp;3306;mysql;open;MariaDB;unauthorized;syn-ack;;10;cpe:/a:mariadb:mariadb
5  192.168.3.101;;;tcp;17500;db-lsp;open;;;syn-ack;;3;
6
```

Inserción en la BBDD de MySQL

La siguiente diapositiva muestra un ejemplo de inserción de datos en una BBDD generada por el script, accedida mediante Xampp y phpMyAdmin

ID	IP	PUERTO	PROTOCOLO	SERVICIO	FECHA
1	192.168.114.173	21	ftp	ProFTPD	30/05/17 - 10:29
2	192.168.114.173	80	http	Apache httpd	30/05/17 - 10:29
3	192.168.114.173	443	http	Apache httpd	30/05/17 - 10:29
4	192.168.114.173	3306	mysql	MariaDB	30/05/17 - 10:29
5	192.168.114.173	17500	db-lsp		30/05/17 - 10:29
6	192.168.3.1	22	ssh	Dropbear sshd	30/05/17 - 20:40
7	192.168.3.1	80	http	TP-LINK WR841N WAP http config	30/05/17 - 20:40
8	192.168.3.1	1900	upnp	ipOS upnpd	30/05/17 - 20:40
9	192.168.3.1	49152	http	Huawei HG8245T modem http config	30/05/17 - 20:40
10	192.168.3.101	80	http	Apache httpd	30/05/17 - 20:41
11	192.168.3.101	443	http	Apache httpd	30/05/17 - 20:41
12	192.168.3.101	3306	mysql	MariaDB	30/05/17 - 20:41
13	192.168.3.101	17500	db-lsp		30/05/17 - 20:41
14	192.168.3.101	80	http	Apache httpd	30/05/17 - 20:46
15	192.168.3.101	443	http	Apache httpd	30/05/17 - 20:46
16	192.168.3.101	3306	mysql	MariaDB	30/05/17 - 20:46
17	192.168.3.101	17500	db-lsp		30/05/17 - 20:46

5. Conclusiones

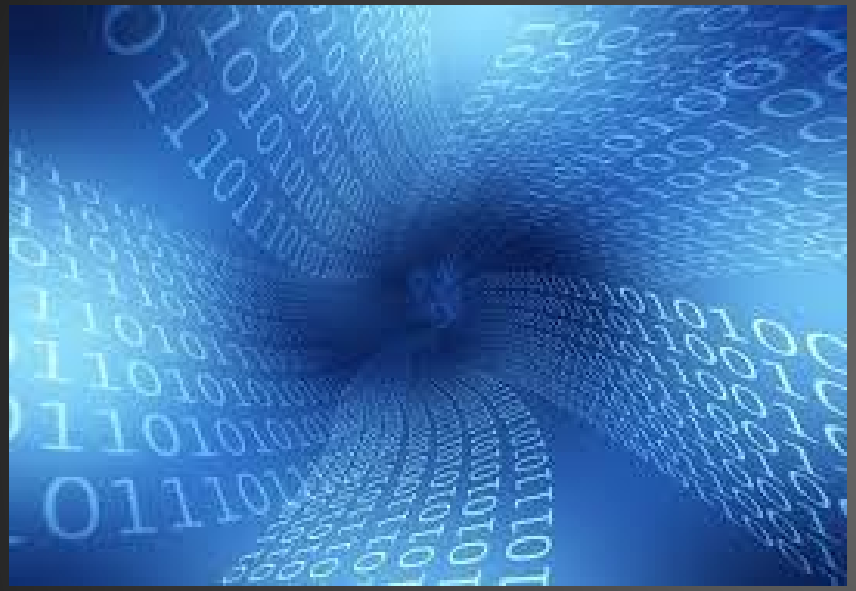


Mediante el presente proyecto hemos podido realizar sendos scripts en Python, con su conveniente apoyo en SQL

Hemos desarrollado en el mismo, pues, la manera de acceder mediante los citados scripts a la información que nos brinda nmap, personalizando las funciones que hemos definido para ello tanto en main.py como en funciones.py.

Los scripts han sido testeados y son plenamente funcionales en sistemas Linux y Windows que tengan implementados adecuadamente los correspondientes módulos nmap y mysqldb, así como sus correspondientes variables de entorno.

6. Bibliografía



Algunos elementos de bibliografía y webgrafía empleados:

- https://conocimientolibre.wordpress.com/2007/06/30/nmap-a-fondo-escaneo-de-redes-y-hosts/
- https://es.wikipedia.org/wiki/Nmap
- Violent Python, a CookBook for Hackers, Forensic Analyst, Penetration Testers and Search Engineers, O'Connor, T.J, Elsevier, USA, 2013,