



# Reporte semanal - Semana 3

16/02/2024

—

Ismael Montoro Peñasco  
Fundación Goodlob

<b>Acciones llevadas a cabo.....</b>	<b>2</b>
<b>Avances realizados.....</b>	<b>2</b>
Labor 1: Estudio de la librería Scapy.....	2
Labor 1.1: Instalar Scapy.....	2
Labor 1.2: Manejo de terminal.....	3
Labor 1.3: Scripting básico con Scapy.....	3
Labor 2: Masterclass Virtualización.....	4
Labor 3: Ejercicios de la Semana.....	4
Labor 3.1: Ejercicios de la función map.....	4
Labor 3.2: Ejercicios de algoritmos.....	5
<b>Planes para la próxima semana.....</b>	<b>5</b>

## Acciones llevadas a cabo

Seguiré trabajando con la librería Scapy. Iré descubriendo nuevas librerías de Python, trabajaré en la API de Python para seguir descubriendo nuevas funcionalidades, e investigar aún más Excel.

Si da tiempo, investigaré temas de ciberseguridad.

## Avances realizados

### Labor 1: Estudio de la librería Scapy

He realizado algunas tareas básicas con Scapy gracias al uso de algunas fuentes

Fuentes:

- <https://scapy.net/>
- <https://github.com/secdev/scapy/blob/master/doc/notebooks/Scapy%20in%2015%20minutes.ipynb>
- <https://codingninjablogs.tech/tryhackme-python-for-pentesters-47b7ce525b90>

#### Labor 1.1: Instalar Scapy

He estado realizando una exploración de Scapy gracias al [Webinar Gratuito: Construir Paquetes con Scapy](#). Como muestra de ello también realice apuntes sobre Scapy.

```
Prueba_Flask.py  LM_Studio_chat.py  prueba_scapy.py  pedir_informacion.py  Primera recopilacion de datos.txt X
Librerías > SRC > test_scapy > Primera recopilacion de datos.txt
28 ##### Instalacion #####
29
30 Instalas scapy con sus dependencias
31
32     Dependencias del sistema para Scapy:
33     Windows - https://npcap.com/#download
34
35     Scapy y las dependencias de Python
36     pip install ipython PyX scapy
37
38 ##### Uso #####
39
40 Para usar scapy desde la terminal escribiendo
41 el comando scapy. Se te abra una shell de scapy
42 en la que podemos ejecutar comandos
43
```

## Labor 1.2: Manejo de terminal

He estado aprendiendo a manejar la shell de Scapy, y viendo qué recursos ofrece. Y he ido apuntando los comandos y recursos aprendidos de Scapy en un archivo de texto.

```

Scapy 2.5.0
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\ismae> cd .\Downloads\EstudiosPython\
PS C:\Users\ismae\Downloads\EstudiosPython> cd .\Librerias\
PS C:\Users\ismae\Downloads\EstudiosPython\Librerias> .\Scripts\Activate.ps1
(Librerias) PS C:\Users\ismae\Downloads\EstudiosPython\Librerias> scapy
INFO: PyX dependencies are not installed ! Please install TexLive or MikTeX.

      aSPY//YASa
    ayyyyyCY/////////YCa
  sY////////YSpCs  scpCY//Pp
ayp ayyyyyySCP//Pp  syY//C
AYAsAYYYYYYYY//Ps  cY//S
  pCCCCY//p  cSSps y//Y
  SPPPP//a  pP//AC//Y
    A//A  cyP///C
    p//Ac  sC//a
    P///YCpc  A//A
  sccccp///pSP///p  p//Y
  sY/////////y  caa  S//P
  cayCyayP//Ya  pY/Ya
  sY/PSY///YCc  aC//Yp
  sc  sccaCY//PCypaapyCP//YSs
    spCPY////////YPSps
      ccaacs

Welcome to Scapy
Version 2.5.0
https://github.com/secdev/scapy
Have fun!
What is dead may never die!
-- Python 2

using IPython 8.21.0
>>> packet=IP(dst="172.67.27.10")/TCP(dport=443)/"Hola"
>>> packet
<IP frag=0 proto=tcp dst=172.67.27.10 |<TCP dport=https |<Raw load='Hola' |>>>
>>>
  
```

## Labor 1.3: Scripting básico con Scapy

He realizado un script con Scapy para pedir mi IP, y mi MAC local.

```

Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  ...  <-  ->
Prueba_Flask.py  LM_Studio_chat.py  prueba_scapy.py 9+  pedir_informacion.py X

Librerias > SRC > test_scapy > pedir_informacion.py > ...
1  import scapy.all as scapy
2
3  p = scapy.Ether()/scapy.IP()/scapy.TCP()
4  print(p.sprintf("%Ether.src%"))
5  print(p.sprintf("%IP.src%"))

PROBLEMAS 12  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
• PS C:\Users\ismae\Downloads\EstudiosPython> & c:/Users/ismae/Downloads/EstudiosPython/Librerias/
  pedir_informacion.py
  48:e7:da:54:a9:cf
• PS C:\Users\ismae\Downloads\EstudiosPython> & c:/Users/ismae/Downloads/EstudiosPython/Librerias/
  pedir_informacion.py
  48:e7:da:54:a9:cf
  192.168.1.34
• PS C:\Users\ismae\Downloads\EstudiosPython>
  
```

## Labor 2: Masterclass Virtualización

Fui a la masterclass de la fundación GoodJob sobre virtualización, y con ello entendí la diferencia entre emulador y contenedor, y repasé el concepto de virtualización.

Emulación es recrear un entorno virtual concreto para ejecutar ciertas aplicaciones, consume muchos recursos debido a que acapara ciertos recursos, y no puede crear más de un entorno virtual a la vez.

Virtualización es un entorno virtual en el cual se acaparan ciertos recursos hardware para la virtualización de cierto software. Consume demasiado, pero al contrario que la emulación se pueden crear varios entornos virtuales, donde se pueden ejecutar varios tipos de software.

## Labor 3: Ejercicios de la Semana

He realizado los ejercicios de la semana, planteándome 4 ejercicios de la función map, función usada para procesar estructuras de datos con una función, y 6 ejercicios de algoritmos de datos.

### Labor 3.1: Ejercicios de la función map

Los ejercicios de la función map constaban de 3 ejercicios básicos para comprender la estructura de la función, y uno avanzado para ver que se puede hacer con la función map.

```
▼ Funcion map
```

```
1º Elevar al cuadrado: Dada una lista de números, crea una nueva lista que contenga cada número elevado al cuadrado. Por ejemplo:  
Entrada: [2, 4, 6, 8]  
Salida esperada: [4, 16, 36, 64]
```

```
[4]: lista = [2, 4, 6, 8]  
  
def potencia(iterable):  
    return iterable ** 2  
  
resultado = list(map(potencia, lista))  
print(resultado)  
  
[4, 16, 36, 64]
```

```
2º Convertir a mayúsculas: Dada una lista de palabras, crea una nueva lista que contenga las mismas palabras en mayúsculas. Por ejemplo:  
Entrada: ['hola', 'mundo', 'python']  
Salida esperada: ['HOLA', 'MUNDO', 'PYTHON']
```

```
[6]: diccionario = ['hola', 'mundo', 'python']  
  
def mayusculas(i):  
    return i.upper()  
  
resultado = list(map(mayusculas, diccionario))
```

## Labor 3.2: Ejercicios de algoritmos

Realice unos 6 ejercicios de algoritmos de datos para procesar distintos tipos de datos, entre ellos está el algoritmo de Edsger Dijkstra.

```
def dijkstra(graph, start_node) -> dict():
    """
    Está es la funcion que usara el algorimo de Edsger Dijkstra,
    con el fin de realizar un analisis de las rutas de todo el
    grafo hasta llegar al nodo final indicado en el diccionario
    """
    # Primero crearemos un diccionario en el que todas la claves tengan el valor de infinito (inf)
    distances = {node: float('inf') for node in graph}
    # Segundo indicaremos que a la clave con el nombre del nodo inicial se le asigne 0
    # porque no hay que pasar por ninguna arista, dado que es el nodo inicial
    distances[start_node] = 0
    # Crearemos un set y un diccionario para los nodos visitados.
    visited = set()
    previous_nodes = {}

    # Este bucle while recorrera todo el grafo, llegando a todos los nodos
    while len(visited) < len(graph):
        # Buscaremos el nodo mas cercano al nodo que se esta analizando
        current_node = min((node for node in graph if node not in visited), key=distances.get)
        visited.add(current_node) # Una vez analizado el nodo se registra como visitado

        # Actualizamos las distancias para los nodos vecinos al nodo mas cercano al nodo analizado
        for neighbor, weight in graph[current_node].items(): # Del nodo actual
            new_distance = distances[current_node] + weight
            if new_distance < distances[neighbor]:
                distances[neighbor] = new_distance # Guardamos el peso entre el nodo anterior y nodo actual
                previous_nodes[neighbor] = current_node # Guardamos el nodo visitado

    return distances, previous_nodes
```

## Planes para la próxima semana

Realizaré todo aquello que no me dio tiempo a realizar esta semana, aprender un poco de scripting python con la librería Scapy, me dedicaré a mi proyecto personal de Python a lo largo del mes, el proyecto “Espadas”.