PIA

Instrucciones del PIA

DARIEN ALEJANDRO LEZA BARCENAS

1- Verificar un puerto con TCP

```
if opc == 1 :
    escaner = nmap.PortScanner()
    escaner.scan('127.0.0.1', '1-1024')
    print(escaner.scan('127.0.0.1', '1-1024', '-v -sV'))
    print(escaner.command_line())
    print(escaner.scaninfo())
    print(escaner.all_hosts())
    print(escaner['127.0.0.1'].hostname())
    print(escaner['127.0.0.1'].state())
    print(escaner['127.0.0.1'].all_protocols())
    print(escaner['127.0.0.1']['tcp'].keys())
    n = int(input("que puerto quieres verificar: \n "))
    print(escaner['127.0.0.1'].has_tcp(n))
    print(escaner['127.0.0.1']['tcp'][n])
    print(escaner['127.0.0.1']['tcp'][n]['product'])
```

Aquí primero se importo el modulo nmap, que requiere una instalación ademad de la del PIP INSTALL.

Primero creamos un scanner de nmap, después establecemos al local host para el rango de puertos, el tercer parámetro es para que sea TCP, con escaner.command_line() se realiza el esaneo, con escaner.scaninfo() muestra un resumen, escaner.all_hosts() escanea la IP dada, después muestra el nombre del host, su estado y los protocolos, se muestran los keys para que lo introduzcamos y sequemos el el estado del puerto

2- Cifrado Cesar

```
if opc == 2 :
    sim = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890'
    r = " "
    men = input("introduze un mensaje ")
    key = int(input("introduze una clave "))
    ci = int(input("¿deseas cifrar(1) o desifrar(2)? "))
    for simon in men:
        if simon in sim:
            isimo = sim.find(simon)
            if ci == 1:
                indin = isimo + key
                if indin >= len(sim):
                    indin = indin - len(sim)
                elif indin < 0:
                    indin = indin + len(sim)
                r = r + sim[indin]
            elif ci == 2:
                indin = isimo - key
                r = r + sim[indin]
            else:
                print("te equivocaste")
                exit()
    print(r)
    with open('reporte_cesar.txt', 'w') as esc:
        esc.write(r)
```

Se tiene el abecedario en la una variable, se piden los datos, realiza un ciclo en el abecedario y otro en el mensaje, se hace el procedimiento en caso de que quiera cifrar o descifrar.

Se imprime el resultado y crea un archivo txt con el mismo

3- Scrapting

```
elif opc == 3 :
    url = input("introduce la url \n ")
    print("descargando imagenes de:"+ url)
    try:
        res = requests.get(url)
        pana = html.fromstring(res.text)
        ima = pana.xpath('//img/@src')
        print ('Imagenes %s encontradas' % len(ima))
        os.system("mkdir images")
        for j in ima:
            if j.startswith("http") == False:
                desca = url + image
            else:
                desca = j
                print(desca)
                ro = requests.get(desca)
                f5 = open('images/%s' % desca.split('/')[-1], 'wb')
                f5.write(ro.content)
                f5.close()
        print("se a creado una carpeta con imagenes")

    except Exception as e:
        print(e)
        print ("Error conexion con " + url)
        pass
```

Se pide la url de una imagen, la except es en caso de un error de conexión y con el try se buscan las imágenes de la url y además con el mkdir se crea el directorio donde se guardaran las imágenes finalmente se descargan y se agregan a la carpeta

4- Servidor UDP

```
elif opc == 4 :
#este viene con otro programa de cliente udp para que funcione
#se tiene que abrir primero el servidor
    ipl = "127.0.0.1"
    pl = 2000
    buffer = 1024
    udpserver = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    udpserver.bind((ipl, pl))
    print("Servidor UDP listo para recibir preguntas")
    while(True):
        reci = udpserver.recvfrom(buffer)
        men = reci[0]
        ip = reci[1]
        print("pregunta: " + men.decode())
        re = input("respuesta: ")
        udpserver.sendto(str.encode(re), ip)
```

Para este es necesario tener el archivo del cliente en la misma carpeta, el servidor se tiene que abrir primero.

Con socket se establece la conexión de internet además que sea da la dirección local para que trabaje en el mismo equipo, el programa tiene un ciclo while para hacer preguntas hasta que el cliente diga que pare.

```
import socket
import sys
server = ("127.0.0.1", 2000)
buffer = 1024
udpsocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while(True):
    pregunta = input("introduze una pregunta, no olvide los signos ¿? y si quieres salir escribe bye ")
    if pregunta == "bye":
        udpsocket.close()
        sys.exit()
        exit()
    udpsocket.sendto(str.encode(pregunta), server)
    men = udpsocket.recvfrom(buffer)
    re = "respuesta" + men[0].decode()
    print(re)
```

El cliente hace lo mismo de establecer conexión y el while de las preguntas con la diferencia de que puede salir si quiere

5- Hash 512

```
elif opc == 5:
#para este se necesita un archivo txt con 100 paswords
    lista = open("100passwords.txt","r",errors='ignore').readlines()
    arr = []
    for i in lista:
        sha512 = hashlib.sha512(i.encode())
        r = sha512.hexdigest()
        arr.append(r)
        with open('pasword_hash512.txt', 'w') as tabla:
           for j in arr:
            tabla.write(j)
    print("se ha creado un documento txt con los hashes")
else :
    print("te equivocaste opcion no valida")
```

Para este se necesita el archivo de las 100 password en la misma carpeta

Primero lee el txt, los cuenta y los convierte en hashes 512 y los escribe en un nuevo documento txt