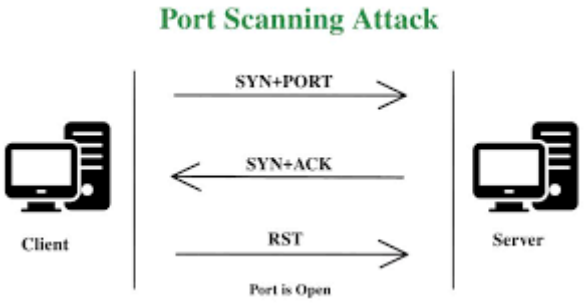
Escaneo de Red




```
root →/app (main X) $ python app.py -r 192.168.95.176/30 -p 22,80 -t 2
Escaneo de red


IP                    Sistema               Mac
192.168.95.179        Linux           02:42:87:30:89:dd
--------              ------

Escaneo de puertos

IP              Puerto          Servicio
192.168.95.179  22                      ssh
192.168.95.179  80                      http
root →/app (main X) $
```

Escaneo de Red

Índice

Objetivo **2**

Solución **2**

Puesta en marcha **2**

- Normal 2
- Con env 2
- Con dev containers 2
- Repositorio 2

Estructura del proyecto **3**

Algoritmo **4**

- Tratado de errores con ValueError: 4
- Manejo de los hilos: 4
- Manejo de expresiones regulares 5
- Funciones estáticas 5
- Programación orientada a objetos 5

Pruebas **5**

Escaneo de Red

Objetivo

Nuestro objetivo es crear una herramienta que sea capaz de escanear la red y todos los puertos para conocer mejor nuestra red.

Solución

Hemos optado por usar scapy ya que es una librería que trabaja con socket (bajo nivel de python) con lo cual no dependemos nada más que python para hacer el escaneo de red.

Puesta en marcha

Normal

pip install -r requirements.txt
python app.py -h

Con env

python -m venv environment
pip install -r requirements.txt

Con dev containers

Si te fijas en el proyecto se encuentra una carpeta llamada .dev containers con lo cual si abres el proyecto con el vscode con la extensión dev containers le notificará que abra la carpeta en el contenedor y cuando se le habrá tendra un contenedor python con las dependencias instaladas

Repositorio

https://github.com/Luisalberto2020/escaneoRedHacking

Ejemplos de uso

python app.py -r 192.168.95.176/30 -p 22,80 -t 2 -j salida.json
python app.py -r 192.168.95.176/30 -p all -j salida.json
python app.py -r 192.168.95.176/30 -p 22,80 -t 2 -v -j salida.json
python app.py -i 192.168.95.178 -p 22,80 -pn

Escaneo de Red

Estructura del proyecto

- Archivo principal es app.py
- Archivo de pruebas testFuncionamiento.py
- **Utils**
    - **menu.py**: Se encarga de los argumentos y de imprimir por pantalla el resultado.
    - **network.py** Se encarga de comprobar los argumentos en tema de red aparte hace operaciones básicas de red como sacar los host de la red etc.
    - **ping.py**: Es la clase que cuando se construye hace un ping a una máquina y luego para obtener la información si está activa la máquina da un resumen de la información del ping en un diccionario etc.
    - **portScanner.py:**Es la clase que cuando se construye hace un escaneo de un puerto y luego tiene diversos métodos para obtener la información resultante en formato diccionario.
- **Servicios**
    - **AppService.py**: Es el servicio principal de la app llevando la lógica de la aplicación en especial con los métodos main_ping y main_scan con la gestión de los hilos y los bucles de generación de los objetos ping y port scanner.
    - **almacenamientoService**:Es el que se encarga de almacenar los datos en formato json.

Escaneo de Red

Algoritmo

Tratado de errores con ValueError:

```python
def __check_args(self):
    '''Comprueba que los arguemntos sean correctos'''
    if self.argumentos['verbose']:
        print(f'argumentos: {self.argumentos}')
    if self.argumentos['red'] is not None:
        if   not Network.check_network(self.argumentos['red']):
            raise ValueError('La red no es valida')

    if self.argumentos['ip'] is not None:
        if not Network.check_ip(self.argumentos['ip']):
            raise ValueError('La ip no es valida')
    if self.argumentos['threads'] < 1:
        raise ValueError('Los hilos no pueden ser menor que 1')

    if not self.__check_port():
        raise ValueError('El puerto no es valido')
```

Como se ve en la foto si hay error de parseo mandara un value error que en el main lo capturara y mandará el mensaje

```python
except ValueError  as error:
    print(f'{Fore.RED}Error: {error}{Style.RESET_ALL}')
    exit(1)
except PermissionError as error:
    print(f'{Fore.RED} Por favor ejecuta el programa como administrador{Style.RESET_ALL}')
    exit(1)
```

Manejo de los hilos:

```python
with ThreadPoolExecutor(max_workers=threads) as executor:
    for result_port in executor.map(AppService.__make_port, [host] * len(ports), ports, [verbose] * len(ports),
        if result_port is not None:
            result[result_port['port']] = {'servicio': result_port['servicio']}
```

Aquí trabajamos con una cola de hilos donde le especificamos el número máximo de hilos y python gestiona que no pase nunca ese número máximo de hilos.

Escaneo de Red

Manejo de expresiones regulares

los argumentos se chequean con expresiones regulares como la siguiente con re.mach comprobamos que la ip sea correcta

```python
@staticmethod
def check_ip(ip: str) -> bool:
    return re.match(
        r'^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$',
        ip
    ) is not None
```

Funciones estáticas

Son los objetos que no hace falta construirlos por ejemplo network y usamos funciones estáticas co @staticmetod

```python
@staticmethod
def get_ip(network:str) -> str:
    ip =  network.split('/')[0]
    if not  Network.check_ip(ip):
        raise ValueError('La ip no es valida')
    return ip
```

Programación orientada a objetos

Como se puede observar las clases ping y scanPort son dos clases orientadas a objetos donde cuando se construye realizamos las operaciones y después consultamos y modificamos sus atributos

```python
class Ping():
    def __init__(self, target: str):
        '''Hace un ping a una maquina y obtiene su sistema operativo'''
        self.target = target
        self.packet = IP(dst=self.target) / ICMP()
        self.response = sr1(self.packet, timeout=0.5,verbose=0)
```

Pruebas

Para las pruebas hemos comprobados los errores al introducir los datos como por ejemplo

Escaneo de Red

```
root ➜/app (main X) $ python app.py -r 192.168.95.176 -p 22,80 -t 2
Error: La red no es valida
root ➜/app (main X) $
```

y en el funcionamiento hemos hecho un programa de pruebas que se llama de testdefuncionamiento para probar el funcionamiento del programa