

SCANNER DE PUERTOS INTERACTIVO

Juan C. Palacio

Abstract— Using the Python programming language, a script was developed with the purpose of using libraries such as Socket, Os, among others, and software such as nmap to identify active hosts in the segment (LAN) and additionally the ports they have open, what services they have open and if there are known vulnerabilities in them. The purpose of this tool is that students, enthusiastic staff, security auditors, Ethical Hackers or Pentesters can make a process of recognition of ports in a comfortable and automated way, so that you can find the information if you run any particular command.

Resumen – Mediante el lenguaje de programación Python se desarrollo un script que tiene como propósito hacer unos de librerías como Socket, Os entre otras, y software como nmap para identificar Host activos en el segmento (red LAN) y adicionalmente los puertos que tienen abiertos, que servicios tienen abiertos y si existen vulnerabilidades conocidas en estos. El propósito de esta herramienta es que los estudiantes, personal entusiasta, auditores de seguridad, Etical Hackers o Pentesters puedan hacer un proceso de reconocimiento de puertos de manera cómoda y automatizada, de manera que se pueda encontrar la información si ejecutar algún comando en particular.

I. INTRODUCCIÓN

Este documento tiene como propósito documentar y explicar el funcionamiento de una herramienta informática programa en el lenguaje de programación Python la cual es llamada como scanner de puertos interactivos.

De acuerdo a lo anterior se abordará primeramente en el presente documento un manual de usuario en el cual se especifican todas las funcionalidades del programa y que tipo de información podemos adquirir de esta. Seguidamente se especificará un manual para el desarrollador, el cual tiene como objetivo principal especificar el código paso a paso con su respectiva explicación técnica, con el fin de que al momento de ser interpretado se logre comprender que hace particularmente cada porción de código.

De acuerdo a lo anterior es importante mencionar que finalmente este desarrollo tiene como finalidad aportar al desarrollo de conocimiento en materia de ciberseguridad y a los profesionales en seguridad informática también conocidos como Etical Hackers o Pentesters, que realizan múltiples

auditorias en entornos corporativos en un ambiente de red interno (Redes LAN pequeñas), con el fin de automatizar este tipo de procesos que pertenecen a una fase de reconocimiento según el ciclo de Pentesting.

II. MANUAL DE USUARIO

Inicialmente para ejecutar el programa 5c4n3r se debe instalar una serie de paquetes de Python para su correcto funcionamiento.

Para instalar los paquetes se debe seguir las siguientes instrucciones:

Una vez instalado los paquetes necesarios se debe ejecutar el script de la siguiente manera en un sistema operativo Linux.

Python3 5c4n3r.py



Imagen propia

Después de ejecutar el comando anterior se despliega la interfaz por consola del programa.

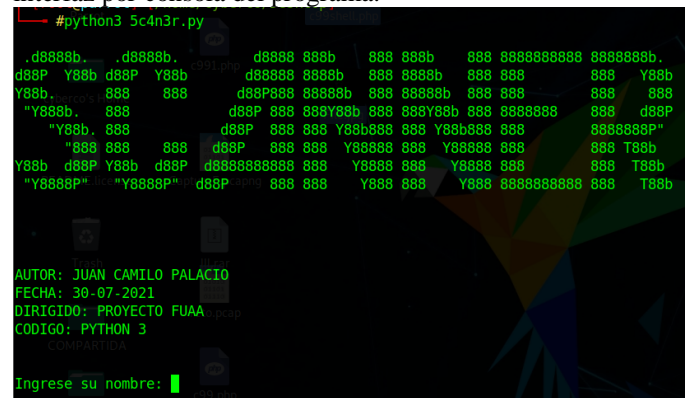


Imagen propia

Aquí inicialmente se debe introducir un nombre que puede ser numero, o alfanumerico.

Después de introducir el nombre se despliega el siguiente menu:

```

Ingrese su nombre: juan
Hola juan, Bienvenido al programa de Scanner FUAA
Tu Equipo es: parrot y tu IP es: 192.168.40.148
1 : Buscar IPs Activas en el segmento
2 : Escanear Puertos de Ips Activas
3 : Conectarse a FTP o SSH de Ips Activas
4 : Salir
Seleccione una Opción: █

```

Imagen propia

Aquí inicialmente se indica el nombre introducido, el nombre del equipo en el que se ejecuto el programa y la IP privada del equipo.

Seguidamente se encuentran 4 opciones de las cuales la 2 y 3 su funcionamiento depende de los datos previos recolectados en la primer opcion de lo anterior no se habilitara su funcionamiento.

```

Ingrese su nombre: juan
Hola juan, Bienvenido al programa de Scanner FUAA
Tu Equipo es: parrot y tu IP es: 192.168.40.148
1 : Buscar IPs Activas en el segmento
2 : Escanear Puertos de Ips Activas
3 : Buscar Información de Puertos (Version, Vulns)
4 : Salir

```

Imagen propia

En primer opcion llamada Buscar Ips Activas en el segmento se despliega el siguiente menu:

```

El segmento en el que estas es: 192.168.40.0/24
1 : Buscar IPs entre un rango especifico del segmento
2 : Buscar en todo el segmento
3 : Ver IPs ya Identificadas
4 : Volver Menu Anterior
5 : Salir del programa
Seleccione una Opción: █

```

Imagen propia

Aquí inicialmente se indica en que segmento se puede buscar hosts activos y seguidamente tenemos 5 opciones.

En la primer opcion podemos buscar un rango especifico de IP's dentro de la red LAN.

Como ejemplo buscaremos la IP 192.168.40.149 la cual corresponde a una maquina virtual linux.

```

Seleccione una Opción: 1
Desde 192.168.40.148
Hasta 192.168.40.149
Buscando IPs Activas ...
192.168.40.148 Host Activo. SO Linux
192.168.40.149 Host Activo. SO Linux
Fin del escaneo

```

Imagen propia

Como respuesta se observa que hay 2 hosts activos entre estos la IP que estamos buscando donde se indica que es un sistema operativo Linux.

Despues de finalizar el escaneo se obtiene los resultados y un menu con una serie de opciones.

```

Las IPs Activas son:
Parrot
c991.php
192.168.40.148
192.168.40.149
1 : Volver a escanear otro rango
2 : Ver Ips Activas Identificadas
3 : Volver al menu principal
4 : Salir del programa

```

Imagen propia

En la primer opcion podemos volver a escanear otro rango especifico para tratar de identificar nuevas IP's activas.

En la segunda opcion se puede verificar el consolidado de las IP's activas que se han identificado a traves de las multiples busquedas que se realicen.

En la tercera opcion se puede volvr al menu inicial.

```

Seleccione una Opción: 3
1 : Buscar IPs Activas en el segmento
2 : Escanear Puertos de Ips Activas
3 : Conectarse a FTP o SSH de Ips Activas
4 : Salir

```

Imagen propia

En la ultima opcion sencillamente es para salir del programa.

```

4 : Salir del programa
Seleccione una Opción: 4
Hasta Pronto

```

Imagen propia

Sin embargo continuando con la opcion 3 y regresando al menu inicial ahora se puede utilizar la segunda opción.

Importante que en la segunda opcion si no se ha hecho una identificacion previa de IP's no es posible utilizarla, se obtendria el siguiente mensaje:

```

Seleccione una Opción: 2
No hay Hosts para escanear

```

Imagen propia

De lo contrario aparecera lo siguiente:

```

Los Host Disponibles para escanear son:
1 : 192.168.40.148
2 : 192.168.40.149

1 : Escanear todos los Hosts
2 : Escanear Ip en particular
3 : Salir al menu anterior

```

Imagen propia

Aquí podemos observar los hosts disponibles para escanear y en que modalidad lo podemos hacer (Todos los Hosts, o Hosts en particular).

Si se escoge la primer opción se escanearan todas las IP's ya identificadas.

```

Seleccione una Opción: 1
1: Escanear todos los puertos
2: Escanear Puertos en Particular
3: Regresar al Inicio

```

Imagen propia

Seguidamente se despliega un menu en el cual nos indica como queremos hacer el escaneo a puerto de las IP's si de manera general (1-65536) o de manera particular en la cual se especifica un(os) puerto(s) en particular.

Como ejemplo practico realizaremos un escaneo particular.

```

Ingrese Una Opción: 2
Introduce los puertos separados por "," Ejemplo: 80,445,3389
Introduce los puertos a escanear: 21,80

```

Imagen propia

Aquí encontramos unas indicaciones de uso.

Una vez se indique los puertos a escanear se obtiene una serie de resultados similares a los siguientes:

```

Escanear 192.168.40.148
Comenzando el escaneo a puertos...
Puerto Cerrado 21
Puerto Cerrado 222
Puerto Cerrado 80
No hay Puertos Abiertos

Escanear Finalizado

Escanear 192.168.40.149
Comenzando el escaneo a puertos...
}Puerto abierto 21
Puerto Cerrado 222
Puerto abierto 80
Los puertos abiertos de 192.168.40.149 son: 21,80,

Escanear Finalizado

```

Imagen propia

Como solo hemos identificado 2 IP's activas solo obtendremos los resultados de estas.

Como se puede observar la IP 192.168.40.148 no tiene puertos abiertos de acuerdo a los parametros que se establecieron, sin embargo la IP 192.168.40.149 tiene los puertos 21 y 80 abiertos y el 222 cerrado.

Despues de finalizar el escaneo nos vuelve a salir el menu inicial.

```

1 : Buscar IPs Activas en el segmento
2 : Escanear Puertos de Ips Activas
3 : Conectarse a FTP o SSH de Ips Activas
4 : Salir

```

Imagen propia

Aquí podemos realizar mas busquedas de IP's o volver a escanear puertos.

Para el caso de escaneo a puertos tambien se tiene como opcion hacer un escaneo a una opcion en particular:

```

1 : Escanear todos los Hosts
2 : Escanear Ip en particular
3 : Salir al menu anterior

```

Imagen propia

Ejemplo:

```

Seleccione una Opción: 2
Los Host Disponibles para escanear son:
1: 192.168.40.148
2: 192.168.40.149

```

Imagen propia

Aquí se indica los hosts que se encuentran disponibles para escanear con su respectiva numeración.

Para hacer el escaneo debemos digitar exactamente la IP que se desea escanear de lo contrario saldra error y nos devolvera al menu inicial.

```

Por favor digita la IP disponible que quieres escanear: 192.168.40.149
1: Escanear todos los puertos
2: Escanear Puertos en Particular
3: Regresar al Inicio

```

Imagen propia

Despues de digitar la IP de manera correcta, nuevamente se indicara en que modalidad se quiere escanear los puertos, de manera practica esta vez se escanearan todos los puertos posibles.

```

Escaneando 192.168.40.149
Comenzando el escaneo a puertos...
Puerto abierto 21
Puerto abierto 22
Puerto abierto 23
Puerto abierto 25
Puerto abierto 53
Puerto abierto 80
Puerto abierto 111
Puerto abierto 139
Puerto abierto 445
Puerto abierto 512
Puerto abierto 513
Puerto abierto 514
Puerto abierto 1099
Puerto abierto 1524
Puerto abierto 2049
Puerto abierto 2121
Puerto abierto 3306
Puerto abierto 3632
Puerto abierto 5432
Puerto abierto 5900
Puerto abierto 6000
Puerto abierto 6667
Puerto abierto 6697
Puerto abierto 8009
Puerto abierto 8180
Puerto abierto 8787
Puerto abierto 39988
Puerto abierto 41093
Puerto abierto 45254
Puerto abierto 50762

```

Imagen propia

```

Se han encontrado 4 hosts: 21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514, 1099, 1524, 2049, 2121, 3306, 3632, 5432, 5900, 6000, 6667, 6697, 8009, 8180, 8787, 39988, 41093, 45254, 50762
Escaneo Finalizado

```

Imagen propia

Como se puede observar se obtiene los resultados.

En la tercera opcion del menu principal podemos buscar vulnerabilidades de las IPs identificadas con sus respectivos puertos, sin embargo si previamente no se ha hecho una identificacion de IPs activas, aparece lo siguiente:

```

Seleccione una Opción: 3
IPs disponibles para escanear:
Todavía no se han identificado Hosts activos

```

Imagen propia

y seguira continuamente en el menu inicial.

De igualmanera si ya hay IPs identificadas pero no puertos sale el siguiente mensaje:

```

Todavía no se han identificado IPs con puertos abiertos

1 : Buscar IPs Activas en el segmento
2 : Escanear Puertos de Ips Activas
3 : Buscar Información de Puertos (Version, Vulns)
4 : Salir

```

Imagen propia

En caso de que existan datos IP-Puertos se habilitara el siguiente menu.

```

IPs disponibles para escanear:
192.168.40.149

1 : Escanear Todas las IPs
2 : Seleccione una IP especifica
3 : Regresar al menu anterior
4 : Salir del programa

```

Imagen propia

Aquí podemos inicialmente observar las IPs que estan activas y contienen algun puerto abierto.

Por otra parte esta la serie de opciones que permiten buscar informacion de puertos de las IPs de manera general o particular.

Como ejemplo practico seleccionare la segunda opcion.

```

1: Escanear todos los puertos abiertos de todas las IPs
2: Escanear Puertos en Particular de IPs
3: Regresar al Inicio

```

Imagen propia

Aquí sale otro submenu que permite escoger si buscar en todos los puertos activos de todas las ips o escoger puertos en particular, para el caso escogeremos la segunda opcion

```

IPs identificadas con puertos:
1 : 192.168.40.149

Digite la IP que desea escanear: 192.168.40.149

```

Imagen propia

Aquí deberemos copiar textualmente alguna de las IPs activas Seguidamente se despliega informacion nueva

```

Digite la IP que desea escanear: 192.168.40.149
la IP 192.168.40.149 tiene estos puertos abiertos:
80

```

Imagen propia

Aquí como se puede observar de ejemplo se puede ver que hay una indicacion previa que dice que solo el puerto 80 esta abierto.

Una vez se seleccione el puerto abierto o varios puertos comenzara a buscarse la version del servicio desplegado en el puerto indicado y tambien se buscara posibles vulnerabilidades mediante nmap.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-07-29 21:10 PDT
Stats: 0:02:09 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
Imagen propia
```

Como se observa el escaneo comienza y solo es esperar resultados.

De esta manera podemos utilizar todas las modalidades del script para buscar ips activas en la Red LAN, saber su sistema operativo y seguidamente buscar si tiene puertos abiertos de manera genera o de manera particular.

III. MANUAL DEL DESARROLLADOR

En este apartado se explicara de manera tecnica los diferentes fragmentos del codigo que conforman el script desarrollado en python llamado 5c4n3r el cual tiene como proposito buscar un IP's activas en el segmento y sus puertos abiertos.

Inicialmente se define unas variables que seran utilizadas durante el desarrollo del script.

```
ips_activas = set()
list_ips_activas = []
variable_control = []
relacionip_ports = {}
```

Imagen propia

En el comienzo de la elaboracion del programa y con el fin de generarle una identidad al script se creo el siguiente banner.

```
print("""
.d8888b. .d8888b. .d8888 888b 888 8888888888 8888888b.
d88P Y88b d88P Y88b .d88888 88888b 888 888 888 Y88b
Y88b. 888 888 .d88P888 88888b 888 88888b 888 888 888
"Y888b. 888 .d88P 888 888Y88b 888 888Y88b 888 8888888 888 d88P
"Y88b. 888 .d88P 888 888 Y88b888 888 Y88b888 888 8888888P"
"888 888 888 d88P 888 888 Y88888 888 Y88888 888 888 T88b
Y88b d88P Y88b d88P d8888888888 888 Y8888 888 Y8888 888 888 T88b
"Y8888P" "Y8888P" d88P 888 888 Y888 888 Y888 8888888888 888 T88b

AUTOR: JUAN CAMILO PALACIO
FECHA: 30-07-2021
DIRIGIDO: PROYECTO FUAA
CODIGO: PYTHON 3
""")
```

Imagen propia

Este banner esta hecho con ascii art, y es mostrado por pantalla mediante la funcion print.

Mediante el siguiente codigo se definen las primeras instrucciones del script.

```
while True:
    nombre_usuario = input('Ingrese su nombre: ')
    if nombre_usuario != '':
        print(f'Hola {nombre_usuario}, Bienvenido al programa de Scanner FUAA')
        myhost = socket.gethostname()
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("8.8.8.8", 80))
        ip = (s.getsockname()[0])
        s.close()

        print(f'Tu Equipo es: {myhost} y tu IP es: {ip} ')
        ipfrag = ip.split(".")
        segmento = ipfrag
        del (segmento[-1])
        segmento_red = '.'.join(segmento)

        menu = [
            'Buscar IPs Activas en el segmento',
            'Escanear Puertos de Ips Activas',
            'Conectarse a FTP o SSH de Ips Activas',
            'Salir'
        ]
```

Imagen propia

Inicialmente se crea un ciclo infinito para su ejecucion en caso de que se genere algun errorpiden unos valores por consola para la identificacion del usuario que utiliza el script, seguidamete se guarda un valor string equivalente a la IP del equipo en el cual se esta ejecutando el script. Este valor es dividido por comas y guardado en una nueva variable, y finalmente a esta se le elimina el ultimo valor y es guardado en otra variable con el fin de evitar el ultimo octeto de la ip. Seguidamente se crea un menu el cual se imprime por pantalla para indicar una serie de instrucciones al usuario.

Seguidamente se crearon una serie de instrucciones que estan alineadas de acuerdo a lo que se muestra en el menu anterior

```
while True:
    try:
        for i, j in enumerate(menu, 1):
            print(i, ': ', j)

        opcion_usuario = int(input('Seleccione una Opción: '))
        if opcion_usuario not in [1, 2, 3, 4]:
            print('Opción fuera del rango')
            continue
        elif opcion_usuario == 4:
            print('Hasta Pronto')
            exit()
        elif opcion_usuario == 1:
            try:
                limpiar_pantalla()

                print(f'El segmento en el que estas es: {segmento_red}.0/24')
                menu2 = [
                    'Buscar IPs entre un rango especifico del segmento',
                    'Buscar en todo el segmento',
                    'Ver IPs ya Identificadas',
                    'Volver Menu Anterior',
                    'Salir del programa'
                ]
```

Imagen propia

Aquí se crea otro ciclo infinito y un try except con el fin de manejar errores.

Inicialmente mediante funcion enumerate y un ciclo for se imprime el anterior menu de forma ordenada y enumerada de

acuerdo a la cantidad de opciones (04), se establecen unas serie de instrucciones en las cuales se indican su operación.

Para el caso de que el usuario seleccione la opción uno dentro de las instrucciones esta inicialmente un manejo de excepciones en caso de errores seguidamente la invocación de una función de la cual hablaremos mas adelante.

Después se muestra por pantalla información del segmento de red y se crea otro menú el cual esta conformado por 5 opciones.

Seguidamente se programa la instrucción mediante un ciclo for y la función enumerate de imprimir de manera ordenada el contenido del menú.

```
for a, b in enumerate(menu2, 1):
    print(a, ': ', b)
seleccion_opcion = int(input('Seleccione una Opción: '))
if seleccion_opcion not in [1, 2, 3, 4, 5]:
    print('Opción fuera del rango')
```

Imagen propia

Después se compara la opción ingresada por el usuario y si no esta dentro de las opciones posibles se imprimira un mensaje indicando la mala acción.

Si en el input el usuario ingresa el valor 1 se ejecutara lo siguiente:

```
elif seleccion_opcion == 1:
    ping_rangos()
    sub_menu()
```

Imagen propia

Se invocara 2 funciones, las cuales se especificaran en este documento mas adelante.

Si la opción de input es igual a 2:

```
elif seleccion_opcion == 2:
    ping_General()
```

Imagen propia

Se ejecutara otra función en particular. esta sera especificada mas adelante.

```
elif seleccion_opcion == 3:
    if list_ips_activas == []:
        print('No hay Host Activos')
    else:
        print('Los hosts activos son:')
        print('\n')
        for ips in list_ips_activas:
            print(ips)

elif seleccion_opcion == 4:
    continue
else:
    print('Hasta Pronto')
    exit()
except ValueError:
    print('Error de Opción')
```

Imagen propia

Aquí en caso de que el input reciba un valor igual a 3 se comparara el contenido de la variable global list_ips_activas, en caso de que no tenga ningún tipo de valor se imprimira un mensaje por consola, de lo contrario mediante un ciclo for se imprimira el contenido de esta 1 a 1.

Por otra parte si el valor recibido por el input es igual a 4 el programa regresara al menú anterior

Finalmente si la opción es 5 el programa mediante la función exit saldra del programa.

Si dentro de todo el código se recibe un valor diferente al aceptado en los inputs hay un manejo de errores sobre el error ValueError el cual tendra como fin imprimir un mensaje al usuario.

Aquí termina la serie de instrucciones de acuerdo al segundo menú.

Regresando al código de instrucciones para el menú uno en caso de que la opción sea igual a 2 se llama una función.

```
elif opcion_usuario == 2:
    scanner()
```

Imagen propia

Esta función esta hecha con el fin de buscar puertos abiertos sobre las IP's que logren ser enumeradas, de esta función se hablara mas adelante.

en la opción 3 se tiene como proposito buscar información mas detallada de los puertos abiertos de las Ips activas en el segmento

```

elif opcion_usuario == 3:
    try:
        print('IPs disponibles para escanear: ')
        if list_ips_activas == []:
            print('Todavía no se han identificado Hosts activos \n')
            continue
        else:
            if relacionip_ports == {}:
                print('Todavía no se han identificado IPs con puertos abiertos \n')
                continue
            else:
                for i in relacionip_ports:
                    print(i)

                print('\n')
                menu6 = [
                    'Escanear Todas las IPs',
                    'Selecciona una IP especifica',
                    'Regresar al menu anterior',
                    'Salir del programa'
                ]
    
```

Imagen propia

Inicialmente se crea un menu y una serie de parametros que estan alineados al contenido de las variables `list_ips_activas` y `relacionip_ports` con el fin de parametrizar las respuestas en caso de que halla o no hallan dantros previos.

```

        for i, j in enumerate(menu6, 1):
            print(f'{i} : {j}')
        seleccionoeop = int(input('Selecciona una opción: '))

        if seleccionoeop not in [1, 2, 3, 4]:
            print('Error de Opción')
            continue
        else:
            if seleccionoeop == 1 or 2:
                versiones()
            elif seleccionoeop == 3:
                continue
            else:
                seleccionoeop == 4
                exit()

    except ValueError:
        print('Error')
        continue
    
```

Imagen propia

Seguidamente ese menu es impreso por pantalla con el fin de que el usuario lo visualice, a partir de sus opciones se invoca la funcion `versiones()` para la opcion 1 o 2 que relacionan tipos de escaneos.

Finalmente se cierra el programa con el `execet ValueError`

```

    except ValueError:
        print('Error de Opción')

    else:
        limpiar_pantalla()
        print('Por favor Introduce un nombre ')
    
```

Imagen propia

Donde arroja un mensaje para el usuario en caso de error. El else siguiente invoca la funcion `limpiar_pantalla()` y en caso de que al inicio del codigo donde se pide el nombre al usuario

el valor sea igual a una string vacio, se muetsre un mensaje en pantalla indicando que ingrese algun valor para el nombre el cual sera guardado en la variable `nombre_usuario`.

A. Funciones definidas en el script.

Funcion `limpiar_pantalla()`:

```

def limpiar_pantalla():
    time.sleep(1)
    if platform.system() == "Windows":
        os.system('cls')
    else:
        os.system('clear')
    
```

Imagen propia

Tiene como proposito limpiar la pantalla de la consola de acuerdo a su sistema operativo.

Funcion `ping_rangos()`:

Para crar la funcionalidad de buscar IP's activas en una red LAN dentro de un rango especifico, se creo la funcion `ping_rangos`.

```

def ping_rangos():
    global list_ips_activas
    rango_inicial = segmento_red
    rango_final = segmento_red
    primer_valor = int(input(f'Desde {rango_inicial}.'))
    if primer_valor > 255:
        print('Error de coordenada')
        return
    listprimer_valor = [primer_valor]
    segundo_valor = int(input(f'Hasta {rango_final}.'))
    if segundo_valor > 255:
        print('Error de coordenada')
        return
    listsegundivalor = [segundo_valor]
    rangos = [listprimer_valor, listsegundivalor]
    if rangos[0] > rangos[1]:
        print('Error de Rango')
        return
    print('Buscando IPs Activas ...')
    time.sleep(2)
    
```

Imagen propia

En esta primera parte se creo una variable global llamada `list_ips_activas`, con el fin de poder cosultar su contenido en otras partes del codigo como funciones.

Seguidamente a 2 variables se les asigna el valor de `segmento_red` donde anteriormente se evidencio que es el valor de la ip sin el ultimo octeto.

A raiz de esto se le pide al usuario por pantalla que ingrese el

ultimo octeto de la ip para hacer un escaneo desde una IP inicial hasta una IP final es importante recalcar que estos valores debe estar en un valor un rango de 0 a 255 ya que ese es el parametro validos de una ip.

Seguidamente se especifican una serie de parametros donde se indica que el primer valor ingresado debe ser menor al segundo valor con el fin de crear un rango de escaneo logico, si los valores ingresados son correctos se imprimira un mensaje que indica que la busqueda de ips comenzo, y una vez finalizado el proceso se invoca la funcion time para que espere 2 segundos.

Seguidamente se especifica la siguiente porcion de codigo dentro de la misma funcion

```
if (platform.system() == "Linux"):
    consola_ping = 'ping -c 1 ' + segmento_red + '.'
    for i in range(primer_valor, segundo_valor + 1):
        comando = consola_ping + str(i)
        ip = segmento_red + '.' + str(i)
        respuesta = os.popen(comando)
        for line in respuesta.readlines():
            if ('ttl' in line.lower()):
                ips_activas.add(ip)
                print(f'{ip} Host Activo. ', end='')
            if ('ttl=128' in line.lower()):
                print('SO Windows')
            elif ('ttl=64' in line.lower()):
                print('SO Linux')
            else:
                print('Desconocido')
                break
    print('Fin del escaneo')
    time.sleep(2)
    limpiar_pantalla()
```

Imagen propia

aquí se indica un medio de respuesta según el sistema operativo en caso de que el SO sea Linux en la variable consola_ping se guardara una instrucción que traduce la ejecución del comando ping -c 1 + el contenido de la variable de segmento de red, una vez ejecutado la instrucción se ejecuta un ciclo for con los parametros guardados en las dos variables que definen los rangos de escaneo.

Lo anterior sera para crear una iteracion entre el rango definido y de esta manera ejecutar el comando ping en cada una de las posibles ips dentro del rango.

Finalmente en cada una de estas se verificara la respuesta del comando ping y en caso de que en la respuesta en las diferentes iteraciones se encuentra el ttl=128 se respondera con un print indicando el mensaje SO Windows, si la respuesta es ttl=64 indicara SO Linux, de lo contrario se respondera con el mensaje desconocido.

Una vez finalizado el primer ciclo se terminara el escaneo y se mostrara con un print el mensaje Fin del escaneo y luego se invocara la funcion time y se esperara 2 segundos.

```
limpiar_pantalla()
if ips_activas != set():
    list_ips_activas = list(set(ips_activas))
    list_ips_activas.sort()
    print(f'Las IPs Activas son: ')
    print('\n')
    for ips in list_ips_activas:
        print(ips)
```

Imagen propia

Finalmente se invoca la funcion limpiar pantalla y si la variable ips_activas tiene algun valor estos se mostraran de manera ordenada.

En caso de que el sistema operativo no sea Linux se ejecutara lo siguiente:

```
else:
    consola_ping = 'ping -n 1 ' + segmento_red + '.'
    for i in range(primer_valor, segundo_valor + 1):
        comando = consola_ping + str(i)
        ip = segmento_red + '.' + str(i)
        respuesta = os.popen(comando)
        for line in respuesta.readlines():
            if ('ttl' in line.lower()):
                ips_activas.add(ip)
                print(f'{ip} Host Activo. ', end='')
            if ('ttl=128' in line.lower()):
                print('SO Windows')
            elif ('ttl=64' in line.lower()):
                print('SO Linux')
            else:
                print('Desconocido')
                break
    print('Fin del escaneo')
    time.sleep(2)
    limpiar_pantalla()
if ips_activas != set():
    list_ips_activas = list(set(ips_activas))
    list_ips_activas.sort()
    print(f'Las IPs Activas son: ')

    for ips in list_ips_activas:
        print(ips)

else:
    print('No hay Host Activos')
```

Imagen propia

Se establecen los mismos parametros con la diferencia que en el comando ping no se ejecute con la opcion -c si no con -n para que sea interpretado por el sistema operativo Windows.

Función Ping_genera()


```
def ping_General():
    global list_ips_activas
    print('Buscando IPs Activas ...')
    time.sleep(2)
    if (platform.system() == "Linux"):
        consola_ping = 'ping -c 1 ' + segmento_red + '.'
        for i in range(1,256):
            comando = consola_ping + str(i)
            ip = segmento_red + '.' + str(i)
            respuesta = os.popen(comando)
            for line in respuesta.readlines():
                if ('ttl' in line.lower()):...
        print('Fin del escaneo')
        time.sleep(2)
        limpiar_pantalla()
        if ips_activas != set():
            list_ips_activas = list(set(ips_activas))
            list_ips_activas.sort()
            print(f'Las IPs Activas son: ')
            print('\n')
            for ips in list_ips_activas:
                print(ips)
        else:
            print('No hay Host Activos')
```

Imagen propia

```
else:
    consola_ping = 'ping -n 1 ' + segmento_red + '.'
    for i in range(1,256):
        comando = consola_ping + str(i)
        ip = segmento_red + '.' + str(i)
        respuesta = os.popen(comando)
        for line in respuesta.readlines():
            if ('ttl' in line.lower()):...
    print('Fin del escaneo')
    time.sleep(2)
    limpiar_pantalla()
    if ips_activas != set():
        list_ips_activas = list(set(ips_activas))
        list_ips_activas.sort()
        print(f'Las IPs Activas son: ')
        print('\n')
        for ips in list_ips_activas:
            print(ips)
    else:
        print('No hay Host Activos')
```

Imagen propia

A direnecia de la función ping_rangos() esta esta diseñada para hacer un escaneo general dentro de una red LAN para descubrir hosts activos con sistema operativo windows, linux u otros en el rango posible de 0 a 255.

Función sub_menu()

Esta funcion esta diseñada para generar un menu luego de que finalice la busqueda de hosts activos en la red LAN

```
def sub_menu():
    menu3 = [
        'Volver a escanear otro rango',
        'Ver Ips Activas Identificadas',
        'Volver al menu principal',
        'Salir del programa'
    ]
    for c, d in enumerate(menu3, 1):
        print(c, ': ', d)
    seleccion_opcion = int(input('Seleccione una Opción: '))
    if seleccion_opcion not in [1, 2, 3, 4]:
        print('Opción fuera del rango')
    elif seleccion_opcion == 1:
        ping_rangos()
        sub_menu()
    elif seleccion_opcion == 2:
        print('Los Hosts Identificados como activos hasta el momento son: ')
        ips_identificadas = list_ips_activas
        for num_posicion in enumerate(ips_identificadas, 1):
            print(num, ': ', posicion)
            print('\n')
            return
    elif seleccion_opcion == 3:
        return
    else:
        print('Hasta Pronto')
        exit()
```

Imagen propia

Aquí se imprime por pantalla un menu que esta conformado por 4 opciones y tiene como proposito principal poder volver a realizar nuevos escaneos enntre rangos especificos, visualizar las IPs identificadas, regresar al menu principal o salir del programa.

Función opcion_ports()

Esta funcion esta creada para realizar el escaneo a puertos mediante la funcion socket, donde consiste enviar paquetes y/o peticiones de red mediante el protocolo TCP para obtener una respuesta 0 o diferente de 0 que finalmente definira si el puerto esta abierto o cerrado.

```
def opcion_ports():
    global relacionip_ports
    menu_rangos_puertos = [
        'Escanear todos los puertos',
        'Escanear Puertos en Particular',
        'Regresar al Inicio'
    ]
    for pos, opcion in enumerate(menu_rangos_puertos, 1):
        print(f'{pos}: {opcion}')
    opcion_rango_port = int(input('Ingrese Una Opción: '))
    if opcion_rango_port not in [1, 2, 3]:
        print('Error de Opción')
        return
    if opcion_rango_port == 3:
        return
```

Imagen propia

Inicialmente en esta funcion se crea un menu con el fin de mostrar una serie de opciones al usuario que definen si

escanear todos los puertos de una o varias IPs, puetos en particular o salir del programa, en caso de que el usuario ingrese una opcion diferente a las permitidas el programa mostrara un mensaje de error de opcion y regresara al menu principal.

En caso de que el usuario escoja la opcion 1 se ejecutara el siguiente codigo de la funcion.

```
else:
    if opcion_rango_port == 1:
        abirtos = []
        if len(list_ips_activas) > 1:
            for i in list_ips_activas:
                print(f'Escaneando {i}')
                ipequipo = socket.gethostname(i)
                print('Comenzando el escaneo a puertos...')
                for puertos in range(1, 65536):
                    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                    respuesta = client.connect_ex((ipequipo, puertos))
                    if (respuesta == 0):
                        client.close()
                        abirtos.append(puertos)
                        print(f'Puerto abierto {puertos}')
                if abirtos != []:
                    print(f'Los puertos abiertos de {i} son: ', end='')
                    for n in abirtos:
                        print(f'{n}', sep=',', end='')
                    relacionip_ports[list_ips_activas] = list(abirtos)
                    abirtos.clear()
                else:
                    print('No hay Puertos Abiertos')
            print('\n')
        print('Escaneo Finalizado')
```

Imagen propia

Aquí si la variable global list_ips_activas tiene una longitud mayor a 1 (Mas de 1 IP) se escanearan los 65535 puertos de cada una de las IPs.

Si hay puertos abiertos estos se mostraran por pantalla y en caso de que no hayan puertos abiertos se mostrara un mensaje por pantalla indicandolo.

En caso de que la variable global list_ips_activas tenga mas de un valor se ejecutara lo siguiente:

```
else:
    for i in list_ips_activas:
        print(f'Escaneando {i}')
        ipequipo = socket.gethostname(i)
        print('Comenzando el escaneo a puertos...')
        for puertos in range(1, 65536):
            client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            respuesta = client.connect_ex((ipequipo, puertos))
            if (respuesta == 0):
                client.close()
                abirtos.append(puertos)
                print(f'Puerto abierto {puertos}')
            if abirtos != []:
                print(f'Los puertos abiertos de {i} son: ', end='')
                for n in abirtos:
                    print(f'{n}', sep=',', end='')
                relacionip_ports[i] = list(abirtos)
                abirtos.clear()
            else:
                print('No hay Puertos Abiertos')
        print('\n')
    print('Escaneo Finalizado')
```

Imagen propia

Aquí se tiene la misma intencion de escanear todos los puertos pero solo de una IP.

En caso de que el usuario escoja la opcion 2 del menu principal de la funcion se tendra como fin hacer un escaneo especifico de puertos y ejecutara lo siguiente:

```
elif opcion_rango_port == 2:
    separador = ','
    abirtos = []
    print('Introduce los puertos separados por "," Ejemplo: 80,445,3389')
    opcion = (input('Introduce los puertos a escanear: '))
    lista_p = opcion
    v_enteros = []
    separados = lista_p.split(separador)
    lista_p = separados
    portconjunto = set(lista_p)
    ordenados = list(portconjunto)
    ordenados.sort()
```

Se crean una serie de variables locales en la funcion y se pide por consola valores en tipo string que seran equivalente a los puertos que se quieren escanear.

De igual forma en caso de que la variable global list_ips_activas contenga mas de una IP se ejecuta lo siguiente:

```
if len(list_ips_activas) < 2:
    try:
        print('\n')
        for i in list_ips_activas:
            print(f'Escaneando {i}')
            ipequipo = socket.gethostname(i)
            print('Comenzando el escaneo a puertos...')
            for puertos in range(len(ordenados)):
                v_enteros = int(ordenados[puertos])
                client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                respuesta = client.connect_ex((ipequipo, v_enteros))
                if (respuesta == 0):
                    client.close()
                    abirtos.append(v_enteros)
                    print(f'Puerto abierto {v_enteros}')
                else:
                    print(f'Puerto Cerrado {v_enteros}')
            if abirtos != []:
                print(f'Los puertos abiertos de {i} son: ', end='')
                for n in abirtos:
                    print(f'{n}', sep=',', end='')
                relacionip_ports[i] = list(abirtos)
                abirtos.clear()
            else:
                print('No hay Puertos Abiertos')
        print('\n')
        print('Escaneo Finalizado')
    except ValueError:
        print('Error de Opción')
```

Imagen propia

Donde escaneara los puertos indicados por cada una de las ips activas, y finalmente los puertos abiertos seran mostrados por consola de acuerdo a su IP de igual manera si los puertos estan cerrados tambien sera indicado al usuario mediante la funcion print.

De lo contrario si la variable global list_ips_activas solo tiene un valor se ejecuta lo siguiente:

```

else:
    for i in list_ips_activas:
        try:
            print('\n')
            print(f'Escaneando {i}')
            ipequipo = socket.gethostbyname(i)
            print('Comenzando el escaneo a puertos...')
            for puertos in range(len(ordenados)):
                v_enteros = int(ordenados[puertos])
                client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                respuesta = client.connect_ex((ipequipo, v_enteros))
                if (respuesta == 0):
                    client.close()
                    abiertos.append(v_enteros)
                    print(f'Puerto abierto {v_enteros}')
                else:
                    print(f'Puerto Cerrado {v_enteros}')

            if abiertos != []:
                print(f'Los puertos abiertos de {i} son: ', end='')
                for n in abiertos:
                    print(f'{n}, ', end='')
                relacionip_ports[i] = list(abiertos)
                abiertos.clear()
            else:
                print('No hay Puertos Abiertos')

            print('\n')
            print('Escaneo Finalizado')
        except ValueError:

```

Imagen propia

Aquí se busca escanear los puertos pero de solamente una IP.

Funcion option_ports_select()

Esta funcion al igual que la anterior tiene como proposito escanear puertos, con la diferencia de que si en el menu de la funcion scanner se elige la opcion la segunda opcion (IP en particular) se llamara a esta funcion

```

abiertos = []
if opcion_rango_port == 1:
    print(f'Escaneando {ip}')
    ipequipo = socket.gethostbyname(ip)
    print('Comenzando el escaneo a puertos...')
    for puertos in range(1, 65536):
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        respuesta = client.connect_ex((ipequipo, puertos))
        if (respuesta == 0):
            client.close()
            abiertos.append(puertos)
            print(f'Puerto abierto {puertos}')
    if abiertos != []:
        print(f'Los puertos abiertos de {ip} son: ', end='')
        for n in abiertos:
            print(f'{n}, ', end='')
        relacionip_ports[ip] = list(abiertos)
        abiertos.clear()
    else:
        print('No hay Puertos Abiertos')

    print('\n')
    print('Escaneo Finalizado')

```

Imagen propia

Aquí el valor de la IP sera guardado en una nueva variable llamada ip, de acuerdo al valor de esta se ejecutara toda la funcion que practicamente es igual a la de opcion_ports().

Funcion scanner()

Esta funcion esta diseñada para enlazar las funciones opcion_ports() y option_ports_select de acuerdo a las opciones que tiene en el siguiente menu.

```

def scanner():
    global ip
    if list_ips_activas == variable_control:
        print('No hay Hosts para escanear')
        print('\n')
    else:
        limpiar_pantalla()
        print('Los Host Disponibles para escanear son: ')
        for listaa, num in enumerate(list_ips_activas, 1):
            print(listaa, ': ', num)
        print('\n')

        menu_scanner = [
            'Escanear todos los Hosts',
            'Escanear Ip en particular',
            'Salir al menu anterior'
        ]
        print('\n')
        for num, posicion in enumerate(menu_scanner, 1):
            print(num, ': ', posicion)

        seleccion_opcion = int(input('Seleccione una Opción: '))

```

Imagen propia

Aquí se establece varias opciones de escaneo si es a todas las ips ya identificadas o a una sola IP.

```

if seleccion_opcion not in [1,2,3]:
    print('Error de Opción ')
    print('\n')
elif seleccion_opcion == 1:
    opcion_ports()

elif seleccion_opcion == 2:
    print('Los Host Disponibles para escanear son: ')
    for num, disponibles in enumerate(list_ips_activas, 1):
        print(f'{num}: {disponibles}')

    ip = input('Por favor digita la IP disponible que quieres escanear: ')
    if ip not in list_ips_activas:
        print('La IP que digistaste no estan en las disponibles')
        return
    else:
        opcion_ports_selec()
else:
    return

```

Imagen propia

Entonces de acuerdo a la opcion seleccionada se llama a la funcion correspondiente.

Función Versiones

Esta es la última función del programa y tiene como propósito integrar nmap con el proyecto, con el fin de poder identificar mediante estas posibles vulnerabilidades en las versiones de los servicios que se encuentran desplegados a través de los

puertos abiertos que contiene una IP activa en un segmento de red abierto.

```
def versiones():
    comando = "nmap -sV --script=vuln "
    lista = []
    seleccion = []
    #def menu():
    menu_rangos_puertos = [
        'Escanear todos los puertos abiertos de todas las IPs',
        'Escanear Puertos en Particular de IPs',
        'Regresar al Inicio'
    ]
    for pos, opcion in enumerate(menu_rangos_puertos, 1):
        print(f'{pos}: {opcion}')
    opcion_rango_port = int(input('Ingrese Una Opción: '))
    if opcion_rango_port not in [1, 2, 3]:
        print('Error de Opción')
        return
    if opcion_rango_port == 3:
        return
```

Imagen propia

Como se observa en la imagen inicialmente esta función crea una serie de variables entre estas la mas importante es comando que contiene una instrucción que mas adelante mediante la librería os se le dará la instrucción al SO de ejecutarla.

Siguiendo las instrucciones del menú en caso de que se seleccione la opción 1 se ejecutara lo siguiente:

```
if opcion_rango_port == 1:
    for b in relacionip_ports.values():
        for c in b:
            lista.append(str(c))
            lista.sort()
        for i in relacionip_ports:
            ejecucion = comando + i + ' -p' + ' ' + ','.join(lista)
            os.system(ejecucion)
            lista.clear()
```

Imagen propia

Aquí sencillamente se escanearan con nmap todos los puertos abiertos que contengan cada una de los host identificados como activos.

En caso de que se seleccione la segunda opción se ejecutara lo siguiente:

```
opcion_rango_port == 2:
    print('IPs identificadas con puertos: ')
    for i, j in enumerate(relacionip_ports.keys(), 1):
        print(f'{i}: {j}')
    print('\n')
    selec = input('Elija la IP que desea escanear: ')
    if selec not in relacionip_ports.keys():
        print('IP errónea')
    elif selec == '':
        print('Introduce algun valor')
        return
    else:
        print(f'La IP {selec} tiene estos puertos abiertos: ')
        ports = relacionip_ports.get(selec)
        for i in ports:
            print(i)
        selec = input('selecciona uno o varios puertos de los que desea escanear: No si son varios puertos deben estar separados por "," ejemplo: 22,80')
        if len(selec) > 0 and len(selec) <= 2:
            seleccion.append(selec)
            print('Comando a ejecutar de seleccion:')
            v = comando + selec + ' -p' + ' ' + ','.join(seleccion)
            os.system(v)
        else:
            separados = selec.split(',')
            print('Comando a ejecutar con los puertos:')
            portul = ','.join(separados)
            v = comando + selec + ' -p' + ' ' + portul
            os.system(v)
```

Imagen propia

aquí sencillamente se busca escanear con nmap la versión de servicios y búsqueda de vulnerabilidades de los puertos particulares de IPs específicas.

IV. CONCLUSIÓN

Es importante que en el momento en que estemos desarrollando una herramienta o script en un lenguaje de programación, primero pensemos en el problema y hacer un esquema grafico de como lo vamos a diseñar con el fin de definir una metodología que nos permita comenzar a desarrollar el código de una forma ordenada, para así evitar posibles impases o dificultades en el código y buscar así un desarrollo más optimo y compacto.

Por otra parte, desde la perspectiva del desarrollador es vale la pena tener en cuenta lenguajes de programación como Python, lenguaje que gracias a su facilidad nos permite desarrollar herramientas útiles para las labores que necesitamos en entornos empresariales, y esto es gracias a las grandes comunidades que existen de Python que podemos consultar para documentarnos y poder resolver problemas que nos surjan durante el proceso de desarrollo.

Dentro del desarrollo de este proyecto se aprendió que Python es un lenguaje de programación realmente fundamental para el entorno de la ciberseguridad ya que por medio de este podemos automatizar tareas de una manera fácil, tareas que nos permiten hacer más fáciles actividades puntuales de auditorías como ejemplo el escaneo a puertos que es una tarea que se encuentra en la primera fase del Pentesting.

Finalmente es importante decir que mediante la programación independientemente del lenguaje de programación que se este utilizando, podemos llegar a hacer grandes proyectos que contribuyan día a día en el bienestar humano, haciendo cada día más fácil las tareas industriales y hasta algunas personales.

V. REFERENCIAS

ConectaBell. 2021. Usando NMAP con Python - ConectaBell. [online] Available at: <<https://conectabell.com/usando-nmap-con-python/>> [Accessed 29 July 2021].

Uso de las funciones split y join en Python - El Camino del Sysadmin. (s. f.-b). El Camino del Sysadmin. < <https://blog.carreralinux.com.ar/2017/07/uso-split-y-join-python/#:~:text=La%20funci3n%20join%20convierte%20una,la%20lista%20separados%20por%20comas.&text=Por%20defecto,%20los%20elementos%20de,las%20palabras%20de%20la%20cadena.> [Accessed 29 July 2021]. >

"3.11. Tipo diccionarios — Materiales del entrenamiento de programación en Python - Nivel básico". Programación en Python - Nivel básico — Materiales del entrenamiento de programación en Python - Nivel básico. < https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo_diccionarios.html > [Accessed 29 July 2021].