**Guía para Construir un Escáner de Red en Python (Estilo Nmap)**

**Objetivo del Proyecto**

Crear un script en Python que permita escanear una red local, identificando hosts activos y puertos abiertos, trabajando en equipos de hasta 3 personas utilizando Git para el control de versiones.

**Requisitos del Proyecto**

1. Trabajo en equipo (máximo 3 personas)

2. Uso de Git para control de versiones

3. Script en Python funcional

4. Documentación básica del todo el proyecto. (incluya capturas y explicaciones)

**Fases del Proyecto**

**1. Organización del Equipo**

- Definir roles:

- Líder del proyecto (coordina tareas)

- Desarrollador principal (código principal)

- Documentador (README y comentarios)

- Crear repositorio Git (GitHub)

- Configurar .gitignore para Python (investigue que es y para que se utiliza)

- Crear estructura inicial del proyecto:

/network\_scanner

├── scanner.py Código principal

├── requirements.txt Dependencias

├── README.md Documentación

└── /tests Pruebas unitarias

**2. Diseño del Escáner**

Funcionalidades básicas:

- Escaneo de hosts activos (ping sweep: técnica para identificar hosts activos en una red enviando solicitudes ICMP (ping) a múltiples direcciones IP)

- Escaneo de puertos en un host específico

- Detección de servicios básicos

Pseudocódigo inicial (sugerido):

python

def ping\_sweep(subred, timeout=1):

Enviar ping a todos los hosts en la subred

Retornar lista de hosts que responden

def scan\_ports(host, puertos, timeout=1):

Intentar conexión a cada puerto en la lista

Retornar lista de puertos abiertos

def main():

Parsear argumentos de línea de comandos

Ejecutar funciones según parámetros

Mostrar resultados

**3. Implementación Básica**

Código inicial para scanner.py:

python

def ping(host, timeout=1):

Realiza un ping a un host y retorna True si responde

def ping\_sweep(network, timeout=1):

Escanea una red y retorna hosts activos

def scan\_port(host, port, timeout=1):

Intenta conectar a un puerto específico

def port\_scan(host, ports, timeout=1):

Escanea puertos en un host específico

Mostrar Ejemplo de uso (básico) de cada una.

**4. Mejoras y Funcionalidades Adicionales**

1. Añadir argumentos por línea de comandos con argparse (Permite definir argumentos posicionales, opciones, ayuda automática y validación.)

2. Implementar diferentes tipos de escaneo (TCP Connect, SYN, etc.)

3. Añadir detección de servicios (banner grabbing: Consiste en conectarse a un puerto (como el 22 para SSH o el 80 para HTTP) y leer el mensaje inicial (banner) que envía el servidor.).

Ejemplo:

| **Puerto** | **Servicio** | **Ejemplo de Banner** |
| --- | --- | --- |
| 22 | SSH | SSH-2.0-OpenSSH\_8.2p1 Ubuntu-4ubuntu0.3 |

4. Mejorar la presentación de resultados

5. Añadir opción para guardar resultados en un archivo

**5. Pruebas y Documentación**

Tareas:

1. Probar el script en diferentes redes

2. Documentar el código con comentarios

3. Crear un README.md con:

- Descripción del proyecto

- Requisitos e instalación (investigue como crear un ejecutable en Python compatible para windows)

- Ejemplos de uso

- Integrantes del equipo

4. Asegurar que todos los commits estén en el repositorio Git

**Consejos para el Trabajo en Equipo con Git**

1. Usar ramas (branches) para características nuevas

2. Hacer commits pequeños y descriptivos

3. Resolver conflictos temprano

4. Usar .gitignore para archivos innecesarios

5. Hacer pull requests para revisión de código

**Evaluación**

Se considerará:

- Funcionalidad del script

- Trabajo colaborativo (commits en Git)

- Calidad del código

- Documentación

**Extras para impresionar**

- Interfaz gráfica simple con Tkinter (para crear interfaces gráficas de usuario (GUIs))

- Detección de sistema operativo remoto

- Comparación con Nmap real

- Escaneo UDP

¡Éxito con su proyecto de escáner de red, versión Ingenieros en Telecomunicaciones INACAP!