

# Escáner de Red con Flask

Diego

25 de julio de 2025

## Resumen del Proyecto

Este proyecto implementa un sistema de escaneo de red local que identifica dispositivos conectados, detecta si son nuevos o recurrentes y analiza los puertos abiertos junto con su tipo (TCP o UDP). La herramienta utiliza **Python** y **Flask** para ofrecer una interfaz web amigable y en tiempo real.

Es útil en contextos de administración de redes, seguridad informática y entornos domésticos o académicos.

## 1. Dependencias y Librerías

- **Flask**: Microframework web para servir la interfaz.
- **Nmap** / **python-nmap**: Escaneo de red y puertos.
- **manuf**: Identifica el fabricante del dispositivo a partir de la MAC.
- **json**, **os**: Almacenamiento local y manejo de archivos.
- **HTML + Bootstrap**: Mejora visual de la interfaz web.

## Instalación de Librerías

Ejecuta en consola:

```
pip install flask python-nmap manuf  
sudo apt install nmap
```

## 2. Estructura del Proyecto

- **run.py** – Ejecuta el servidor Flask.
- **web/app.py** – Controlador principal y ruteo de vistas.
- **web/scanner.py** – Escaneo, comparación y lógica de red.
- **web/templates/index.html** – Interfaz gráfica (HTML + Bootstrap).
- **dispositivos.json** – Registro histórico de dispositivos conectados.

### 3. ¿Cómo ejecutar el proyecto?

1. Clona el repositorio.
2. Instala las dependencias.
3. Asegúrate de estar conectado a la red que quieres escanear.
4. Ejecuta el servidor:  

```
python run.py
```
5. Accede desde el navegador a: `http://localhost:5000`

## 4. Funcionamiento General

### 1. Escaneo de red

Se usa Nmap para identificar todos los dispositivos activos en la red local. Se extrae la IP, dirección MAC y el fabricante.

### 2. Escaneo de puertos

Por cada dispositivo detectado, se escanean los 10 puertos más comunes en TCP y UDP. Se registra el número de puerto y su protocolo.

### 3. Detección de nuevos dispositivos

Almacena los dispositivos detectados en un archivo JSON. En nuevos escaneos, compara las MAC para marcar si el dispositivo es nuevo.

## 5. Interfaz Web

- Se muestran:
  - IP
  - Dirección MAC
  - Fabricante
  - Puertos abiertos
  - Tipo de protocolo (TCP o UDP)
  - Estado (nuevo o ya conocido)
- Está basada en **Bootstrap**, por lo que es responsiva y visualmente ordenada.



Figura 1: Captura de la interfaz web del escáner

## 6. Fragmentos de Código Clave

### Escaneo de Puertos

```
def scan_ports(ip):  
    nm = nmap.PortScanner()  
    nm.scan(hosts=ip, arguments='-sS -sU -T4 -O --top-ports 10')  
    ports = []  
    for proto in nm[ip].all_protocols():  
        for port in nm[ip][proto].keys():  
            ports.append({"port": port, "protocol": proto.upper()})  
    return ports
```

### Comparación de Dispositivos

```
def compare_devices(new_list , old_list):  
    old_macs = {dev['mac'] for dev in old_list}  
    for dev in new_list:  
        dev['is_new'] = dev['mac'] not in old_macs  
    return new_list
```

## 7. Consideraciones

- Debes tener permisos para ejecutar Nmap (posiblemente sudo).
- La precisión depende del estado de la red (firewalls pueden bloquear escaneos).
- El proyecto puede extenderse fácilmente para notificaciones, reportes PDF o monitoreo periódico.

## 8. Criterios Cubiertos

- Interfaz visual clara y funcional.
- Código modular, limpio y comentado.
- Uso real de herramientas y librerías relevantes.
- Aplicación práctica de conceptos de redes.
- Presentación estructurada y entendible.

## 9. Trabajo Futuro

- Agregar autenticación de usuario.
- Reportes exportables en PDF o Excel.
- Sistema de alertas por correo o Telegram ante nuevos dispositivos.
- Automatización por intervalo de tiempo o eventos.

## Contacto

Proyecto realizado por Diego  
Estudiante de Ingeniería Electrónica  
Correo: diarboledac@unal.edu.co