

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

PROGRAMA ESPECIALIZACIÓN EN SEGURIDAD INFORMATICA



SEGURIDAD EN REDES

**Aplicación para detectar servidores con el servicio DNS expuesto con la ayuda de IA
(ChatGPT) y Shodan.**

Presentado por:

Juan Camilo Ospina Ospina

Código:

22501385

Docente:

Andres Felipe Gonzalez Coronado

CALI

ABRIL 2025

Construcción de la aplicación con las funciones básicas

Para iniciar con el proyecto necesitaremos una herramienta de IA para realizar los scripts en el lenguaje Python, para ello usaremos ChatGPT y una. Todo el proyecto será montado en una maquina Kali Linux en un entorno virtual. Es importante comentar también que debemos tener una cuenta en SHODAN, el cual es la pagina donde buscaremos las IP's públicas que tengan el puerto 53 abierto y vulnerable.

Empezamos diciéndole a ChatGPT lo que queremos realizar con el siguiente prompt:

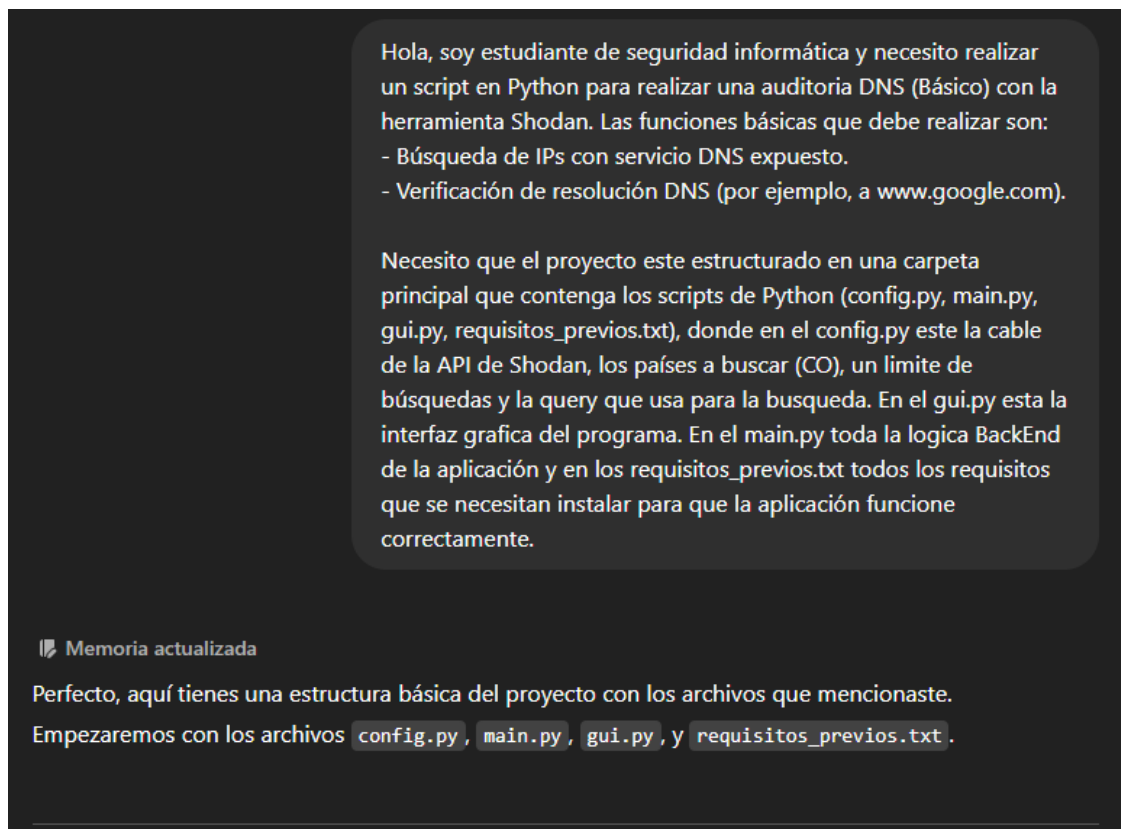


Imagen #1. Prompt inicial.

En este caso el prompt contiene las funciones que queremos en la aplicación, también como queremos que nos estructure el proyecto. La respuesta fue la siguiente:

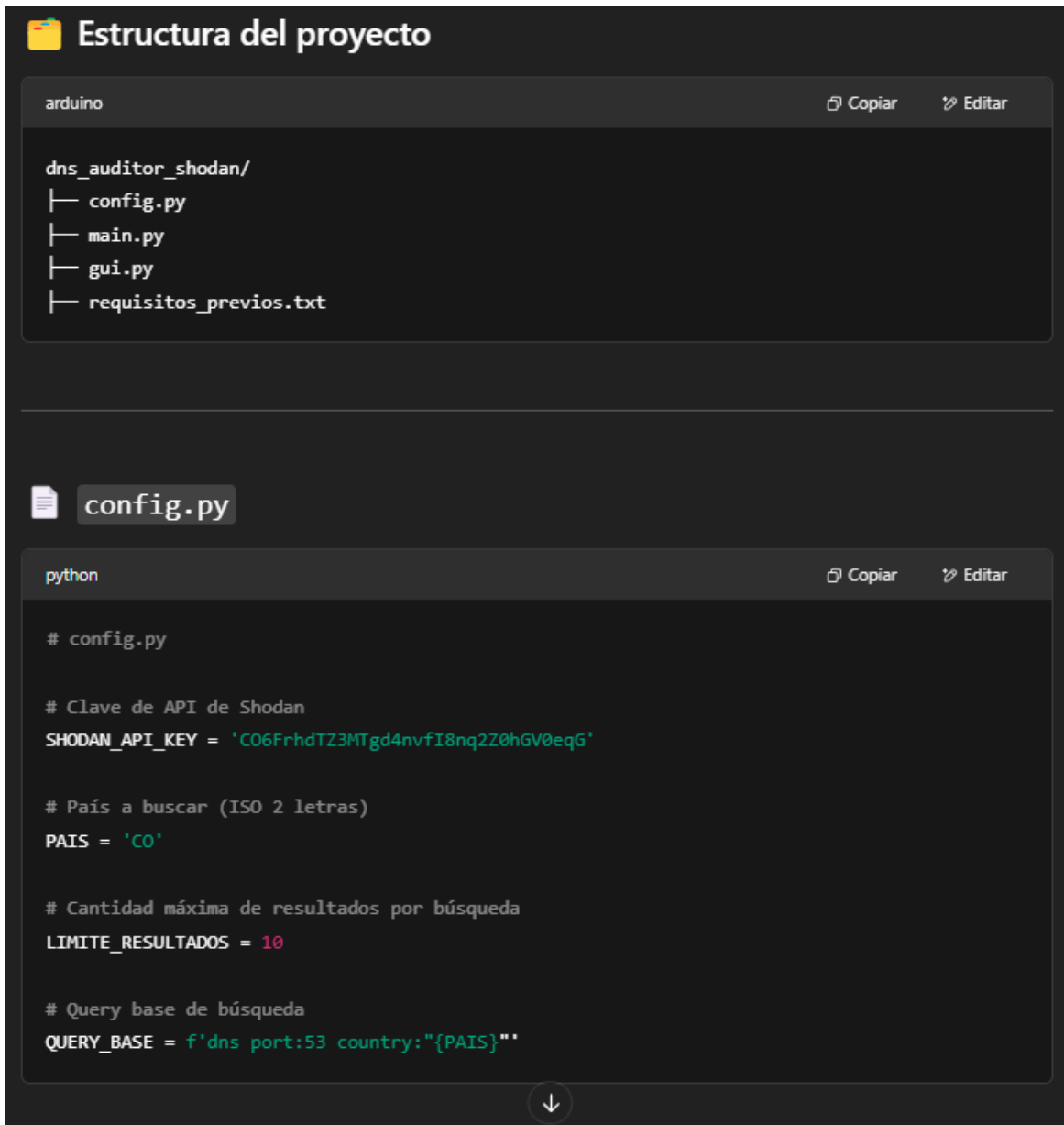


Imagen #2. Respuesta al prompt inicial (Estructura del proyecto, config.py).

✓ Actualización para main.py

```
python ⌕ Copiar ✎ Editar

# main.py

import shodan
import socket
from config import SHODAN_API_KEY, QUERY_BASE, LIMITE_RESULTADOS

def buscar_dns_expuestos():
    api = shodan.Shodan(SHODAN_API_KEY)
    resultados_formateados = []

    try:
        resultados = api.search(QUERY_BASE, limit=LIMITE_RESULTADOS)
        print(f"[+] {resultados['total']} resultados encontrados.")

        for result in resultados['matches']:
            ip = result.get('ip_str', 'N/A')
            org = result.get('org', 'Desconocida')
            pais = result.get('location', {}).get('country_name', 'Desconocido')

            resultados_formateados.append((ip, org, pais))

        return resultados_formateados

    except Exception as e:
        print(f"[!] Error al buscar en Shodan: {e}")
        return []

def verificar_resolucion_dns(ip, dominio='www.google.com'):
    try:
        respuesta = socket.gethostbyname_ex(dominio)
        print(f"[+] {ip} resolvió {dominio} a {respuesta[2]}")
        return True, respuesta[2]
    except Exception as e:
        print(f"[!] {ip} no pudo resolver {dominio}: {e}")
        return False, []

if __name__ == "__main__":
    resultados = buscar_dns_expuestos()
    for ip, org, pais in resultados:
        print(f"IP: {ip} | Organización: {org} | País: {pais}")
        verificar_resolucion_dns(ip)
```

Imagen #3. Respuesta al prompt inicial (Main.py).

```
gui.py (versión simplificada)

python
Copy
Edit

import tkinter as tk
from tkinter import ttk, messagebox
from main import buscar_dns_expuestos, verificar_resolucion_dns

class DNSAuditorGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Auditoria DNS con Shodan")

        # Entrada para la IP a verificar
        frame_input = tk.Frame(root)
        frame_input.pack(pady=5)

        tk.Label(frame_input, text="IP a verificar:").grid(row=0, column=0, padx=5)
        self.entry_ip = tk.Entry(frame_input)
        self.entry_ip.grid(row=0, column=1, padx=5)
        tk.Button(frame_input, text="Verificar IP", command=self.verificar_ip).grid(row=0, column=2, padx=5)

        # Dominio (solo uno)
        tk.Label(root, text="Dominio a resolver (por ejemplo: www.google.com):").pack()
        self.entry_dominio = tk.Entry(root, width=40)
        self.entry_dominio.insert(0, "www.google.com")
        self.entry_dominio.pack(pady=5)

        # Botón para buscar DNS expuestos
        tk.Button(root, text="Buscar DNS Expuestos", command=self.buscar_ips).pack(pady=10)

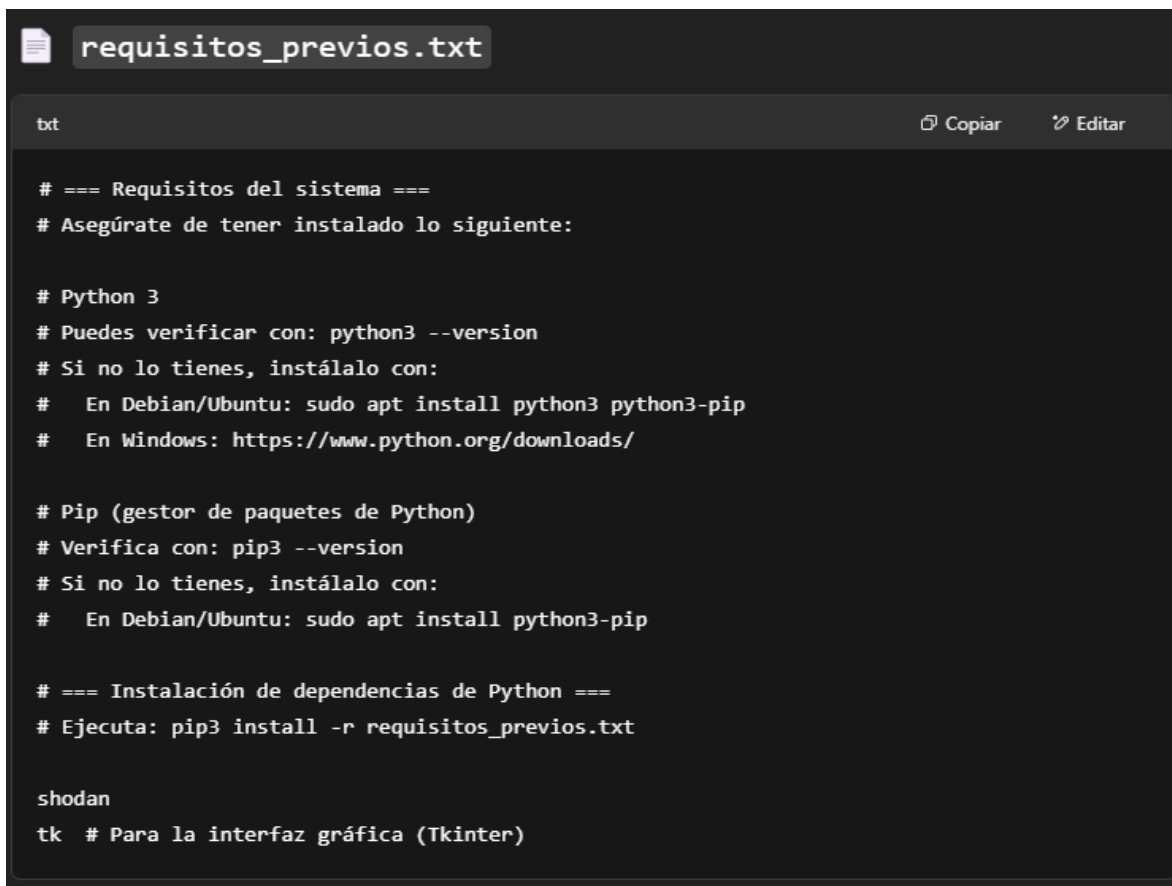
        # Tabla de resultados
        columns = ("IP", "Organización", "País")
        self.tree = ttk.Treeview(root, columns=columns, show="headings")
        for col in columns:
            self.tree.heading(col, text=col)
            self.tree.column(col, width=150)
        self.tree.pack(expand=True, fill="both", pady=10)

    def buscar_ips(self):
        self.tree.delete(*self.tree.get_children())
        resultados = buscar_dns_expuestos()
        for ip in resultados:
            self.tree.insert("", "end", values=(ip, "N/A", "CO")) # Aquí podrías enriquecer

    def verificar_ip(self):
        ip = self.entry_ip.get()
        dominio = self.entry_dominio.get()
        if not ip or not dominio:
            messagebox.showwarning("Advertencia", "Debes ingresar una IP y un dominio.")
            return
        success, resultado = verificar_resolucion_dns(ip, dominio)
        if success:
            messagebox.showinfo("Éxito", f"{ip} resolvió {dominio} a: {resultado}")
        else:
            messagebox.showerror("Error", f"{ip} no pudo resolver {dominio}.")

if __name__ == "__main__":
    root = tk.Tk()
    app = DNSAuditorGUI(root)
    root.mainloop()
```

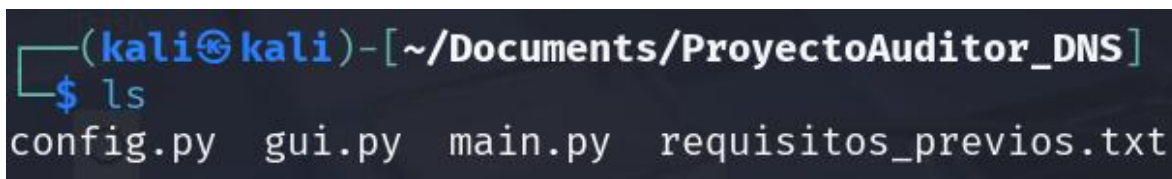
Imagen #4. Respuesta al prompt inicial (gui.py).

A screenshot of a text editor window titled 'requisitos_previos.txt'. The editor has a dark background and shows the following text:

```
# === Requisitos del sistema ===  
# Asegúrate de tener instalado lo siguiente:  
  
# Python 3  
# Puedes verificar con: python3 --version  
# Si no lo tienes, instálalo con:  
#   En Debian/Ubuntu: sudo apt install python3 python3-pip  
#   En Windows: https://www.python.org/downloads/  
  
# Pip (gestor de paquetes de Python)  
# Verifica con: pip3 --version  
# Si no lo tienes, instálalo con:  
#   En Debian/Ubuntu: sudo apt install python3-pip  
  
# === Instalación de dependencias de Python ===  
# Ejecuta: pip3 install -r requisitos_previos.txt  
  
shodan  
tk # Para la interfaz gráfica (Tkinter)
```

Imagen #5. Respuesta al prompt inicial (requisitos_previos.txt).

Procedo en la maquina Kali Linux crear la carpeta y archivos que me menciona ChatGPT en la imagen#2.

A screenshot of a terminal window in Kali Linux. The prompt is '(kali㉿kali)-[~/Documents/ProyectoAuditor_DNS]'. The user has entered 'ls' and the output shows the following files:

```
config.py  gui.py  main.py  requisitos_previos.txt
```

Imagen #6. Carpeta y archivos creados en el Kali Linux.

Seguidamente le pregunto a ChatGPT cuál es el paso por seguir ahora que tengo todos los archivos ya creados.

Ahora que ya tengo la carpeta y los archivos creados en mi maquina Kali Linux cual es el siguiente paso?

Imagen #7. Prompt Nro 2.

✓ 1. Abre una terminal y navega a la carpeta del proyecto

Ejemplo (ajusta la ruta a donde tengas tu carpeta):

```
bash
```

[Copiar](#)[Editar](#)

```
cd ~/Escritorio/dns_auditor_shodan
```

✓ 2. (Opcional) Crea un entorno virtual para mantener el proyecto limpio

```
bash
```

[Copiar](#)[Editar](#)

```
python3 -m venv venv
```

```
source venv/bin/activate # En Windows: venv\Scripts\activate
```

✓ 3. Instala los requisitos

```
bash
```

[Copiar](#)[Editar](#)

```
pip install -r requisitos_previos.txt
```

Esto instalará `shodan` y `tkinter` si es necesario.

Imagen #8. Respuesta al prompt Nro2.

```

(kali@kali) [~/Documents/ProyectoAuditor_DNS]
$ python3 -m venv venv

(kali@kali) [~/Documents/ProyectoAuditor_DNS]
$ source venv/bin/activate

(venv) (kali@kali) [~/Documents/ProyectoAuditor_DNS]
$ pip install -r requisitos_previos.txt
Collecting shodan (from -r requisitos_previos.txt (line 18))
  Using cached shodan-1.31.0-py3-none-any.whl
Collecting tk (from -r requisitos_previos.txt (line 19))
  Downloading tk-0.1.0-py3-none-any.whl.metadata (693 bytes)
Collecting click (from shodan->-r requisitos_previos.txt (line 18))
  Using cached click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting click-plugins (from shodan->-r requisitos_previos.txt (line 18))
  Using cached click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
Collecting colorama (from shodan->-r requisitos_previos.txt (line 18))
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting requests>=2.2.1 (from shodan->-r requisitos_previos.txt (line 18))
  Using cached requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting XlsxWriter (from shodan->-r requisitos_previos.txt (line 18))
  Using cached XlsxWriter-3.2.2-py3-none-any.whl.metadata (2.8 kB)
Collecting tldextract (from shodan->-r requisitos_previos.txt (line 18))
  Downloading tldextract-5.2.0-py3-none-any.whl.metadata (11 kB)
Collecting charset-normalizer<4, >=2 (from requests>=2.2.1->shodan->-r requisitos_previos.txt (line 18))
  Using cached charset-normalizer-3.4.1-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (35 kB)
Collecting idna<4, >=2.5 (from requests>=2.2.1->shodan->-r requisitos_previos.txt (line 18))
  Using cached idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3, >=1.21.1 (from requests>=2.2.1->shodan->-r requisitos_previos.txt (line 18))
  Using cached urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests>=2.2.1->shodan->-r requisitos_previos.txt (line 18))
  Using cached certifi-2025.1.31-py3-none-any.whl.metadata (2.5 kB)
Collecting requests-file>=1.4 (from tldextract->shodan->-r requisitos_previos.txt (line 18))

```

Imagen #9. Creación del entorno virtual y requisitos_previos.txt.

Siguiendo la recomendación de ChatGPT hay que ingresar al archivo config.py y reemplazar SHODAN_API_KEY = 'tu_api_key_real' por la key personal de Shodan.

```

GNU nano 8.3 config.py
# Clave de API de Shodan
SHODAN_API_KEY = 'COLXXXXXXXXXXXXXXXXXXXXXG'

# Pais a buscar (ISO 2 letras)
PAIS = 'CO'

# Cantidad máxima de resultados por búsqueda
LIMITE_RESULTADOS = 10

# Query base de búsqueda
QUERY_BASE = f'dns port:53 country:{PAIS}'

```


Imagen #10. Configuración archivo config.py.

Por último ejecutamos la aplicación con el comando python3 gui.py.

```

(kali@kali) [~/Documents/ProyectoAuditor_DNS]
$ python3 gui.py
[+] 233 resultados encontrados.

```



IP	Organización	País
200.31.70.18	TIVIT COLOMBIA TERCERIZACION DE PROCESOS S	Colombia
45.169.99.155	NET ISP S.A.S	Colombia
158.247.98.47	Oracle Corporation	Colombia
200.10.29.162	SUPER REDES S.A.S	Colombia
138.186.141.245	IPYXON Networks Colombia	Colombia
45.239.88.75	ESCUELA COLOMBIANA DE INGENIERIA	Colombia
179.51.121.44	HV TELEVISION S.A.S	Colombia
201.182.249.140	SP SISTEMAS PALACIOS LTDA	Colombia
186.180.19.202	Colombia Móvil	Colombia
190.7.110.80	ZONA FRANCA DE BOGOTA S.A.	Colombia

Imagen #11. Aplicación en funcionamiento.

Como podemos observar la aplicación muestra la IP, la Organización a la cual pertenece y el país, así como el dominio al cual se puede verificar la IP encontrada. En la imagen #12 vemos como al ingresar la IP 200.31.70.18 y dar click en “Verificar IP” el aplicativo nos dice que pudo resolver el dominio www.google.com.

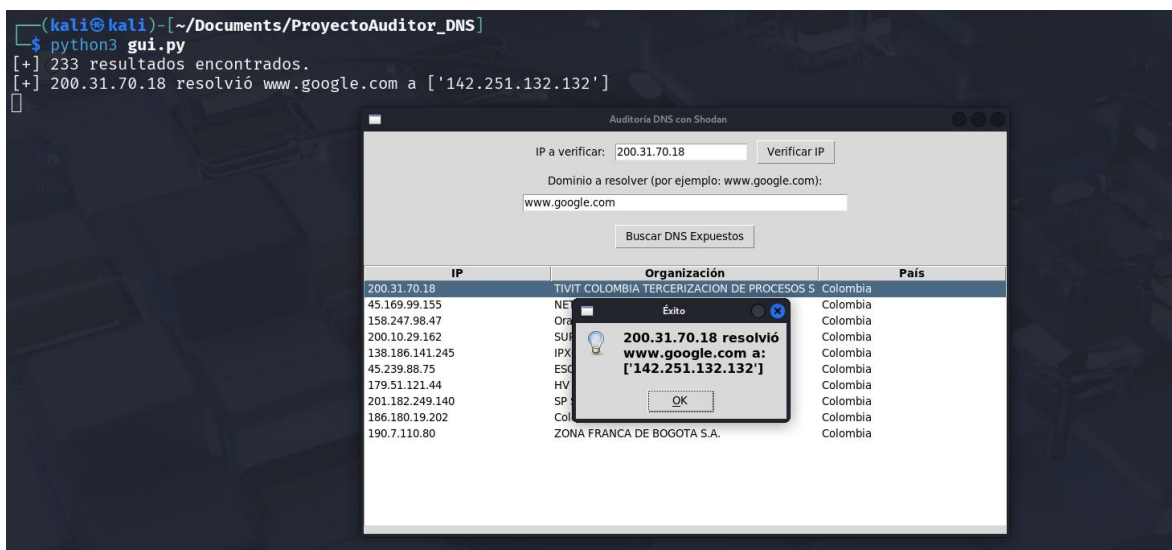


Imagen #12. Resolución de dominio.

Construcción de la aplicación con las funciones avanzadas

Ya que tenemos la versión básica de la aplicación vamos a anexarle 3 funciones avanzadas, la paginación en Shodan, ya que si podemos observar en la imagen #12 nos muestra que encontró 233 IPs pero solo nos muestra 10. La segunda función avanzada es la de añadir más de un dominio de prueba, ya que en la versión básica solo podemos colocar un dominio. La tercera sería verificar si la IP seleccionada esta reportada en alguna blacklist

Con el siguiente prompt le pedimos que añadamos las 3 funcionalidades.

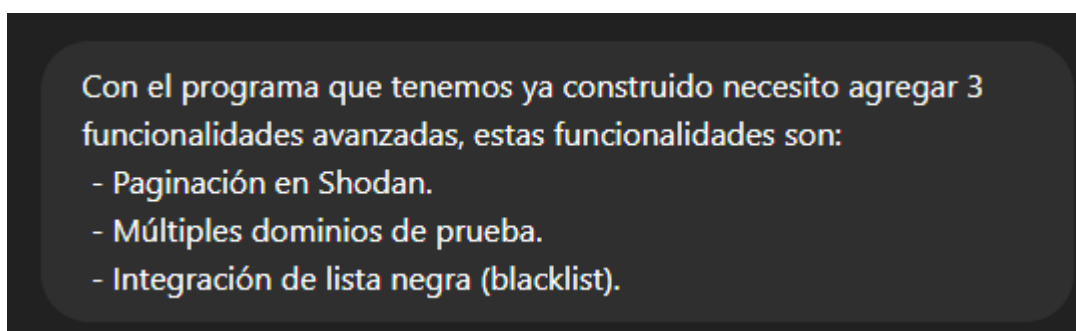


Imagen #13. Prompt funcionalidades avanzadas.

La IA nos da un resumen de lo que se va a realizar, así:



Imagen #14. Respuesta al prompt funcionalidades avanzadas.

Luego nos pregunta una serie de consideraciones para que le digamos como queremos que quede el aplicativo.

Siguientes pasos

¿Querés que integre todo esto directamente en `main.py` y `gui.py` con ejemplos listos para probar? También puedo crear el archivo `blacklist.txt` como parte del proyecto.

¿Te gustaría que la paginación tenga un límite máximo o que sea infinita (mientras haya resultados)?

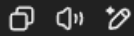


Imagen #15. Consideraciones.

Le respondemos de la siguiente manera:

Si, necesito que integres las funciones avanzadas en los 2 scripts, quiero que en la interfaz en el textbox del dominio permita agregar mas dominios separados por coma. Necesito que crees 2 nuevos scripts (`mainAvanzado.py` y `guiAvanzado.py`) con los cambios, esto para diferenciar la versión básica de la versión avanzada

¡Perfecto! Vamos a crear dos nuevas versiones de los scripts:

- `mainAvanzado.py` : lógica con paginación, múltiples dominios, y lista negra.
- `guiAvanzado.py` : interfaz gráfica que permita ingresar varios dominios y controlar la paginación.

También vamos a incluir:

- Un archivo `blacklist.txt` donde podrás poner IPs que no querés que aparezcan.
- Controles de página (siguiente/anterior) en la GUI.

Imagen #16. Respuesta consideraciones.



```
mainAvanzado.py

python Copiar Editar

import shodan
import dns.resolver
from config import SHODAN_API_KEY, QUERY_BASE

# Iniciar cliente de Shodan
api = shodan.Shodan(SHODAN_API_KEY)

# Cargar blacklist desde archivo
def cargar_blacklist(ruta="blacklist.txt"):
    try:
        with open(ruta, "r") as f:
            return set(line.strip() for line in f if line.strip())
    except FileNotFoundError:
        return set()

BLACKLIST = cargar_blacklist()


# Lista de IPs detectadas en blacklist durante ejecución
ips_en_blacklist = []

def buscar_dns_expuestos(query=QUERY_BASE, pagina=1):
    """
    Realiza búsqueda en Shodan por página (paginación).
    Filtra IPs que estén en blacklist y guarda coincidencias.
    """
    resultados_formateados = []
    try:
        resultados = api.search(query, page=pagina)
        for servicio in resultados['matches']:
            ip = servicio['ip_str']
            org = servicio.get('org', 'Desconocida')
            pais = servicio.get('location', {}).get('country_name', 'Desconocido')
            if ip in BLACKLIST:
                ips_en_blacklist.append(ip)
                continue
            resultados_formateados.append((ip, org, pais))
    except shodan.APIError as e:
        resultados_formateados.append(("Error de API: {e}", "", ""))

    guardar_ips_en_blacklist()
    return resultados_formateados

def guardar_ips_en_blacklist(ruta="coincidencias_blacklist.txt"):
```

Imagen #17. mainAvanzado.py (Fragmento).



```
guiAvanzado.py

python Copiar Editar

import tkinter as tk
from tkinter import messagebox, ttk
from config import QUERY_BASE
from mainAvanzado import buscar_dns_expuestos, verificar_ip, resolver_con_ip
import os

# Página global para paginación
pagina_actual = 1
RESULTADOS_POR_PAGINA = 15

def iniciar_gui():
    global pagina_actual

    ventana = tk.Tk()
    ventana.title("Auditoría DNS con Shodan")
    ventana.geometry("800x540")

    frame = tk.Frame(ventana)
    frame.pack(pady=10, fill="x")

    # Entrada IP
    tk.Label(frame, text="IP a verificar:").grid(row=0, column=0, sticky="w", padx=5)
    entry_ip = tk.Entry(frame, width=25)
    entry_ip.grid(row=0, column=1, padx=5)

    # Entrada dominios
    tk.Label(frame, text="Dominios (separados por coma):").grid(row=1, column=0, sticky="w",
    entry_dominios = tk.Entry(frame, width=50)
    entry_dominios.insert(0, "www.google.com, www.microsoft.com")
    entry_dominios.grid(row=1, column=1, columnspan=2, pady=5, padx=5, sticky="we")

    # Función Verificar IP
    def verificar_ip_input():
        ip = entry_ip.get()
        dominios = entry_dominios.get()
        if not ip:
            messagebox.showwarning("Campo vacío", "Por favor ingresa una IP.")
            return
        resultado_shodan = verificar_ip(ip)
        resultado_dns = resolver_con_ip(ip, dominios)
        resultado_total = f"""
        Verificación de IP: {ip}
```

Imagen #18. guiAvanzado.py (Fragmento).

Ya con estos cambios veremos lo siguiente:

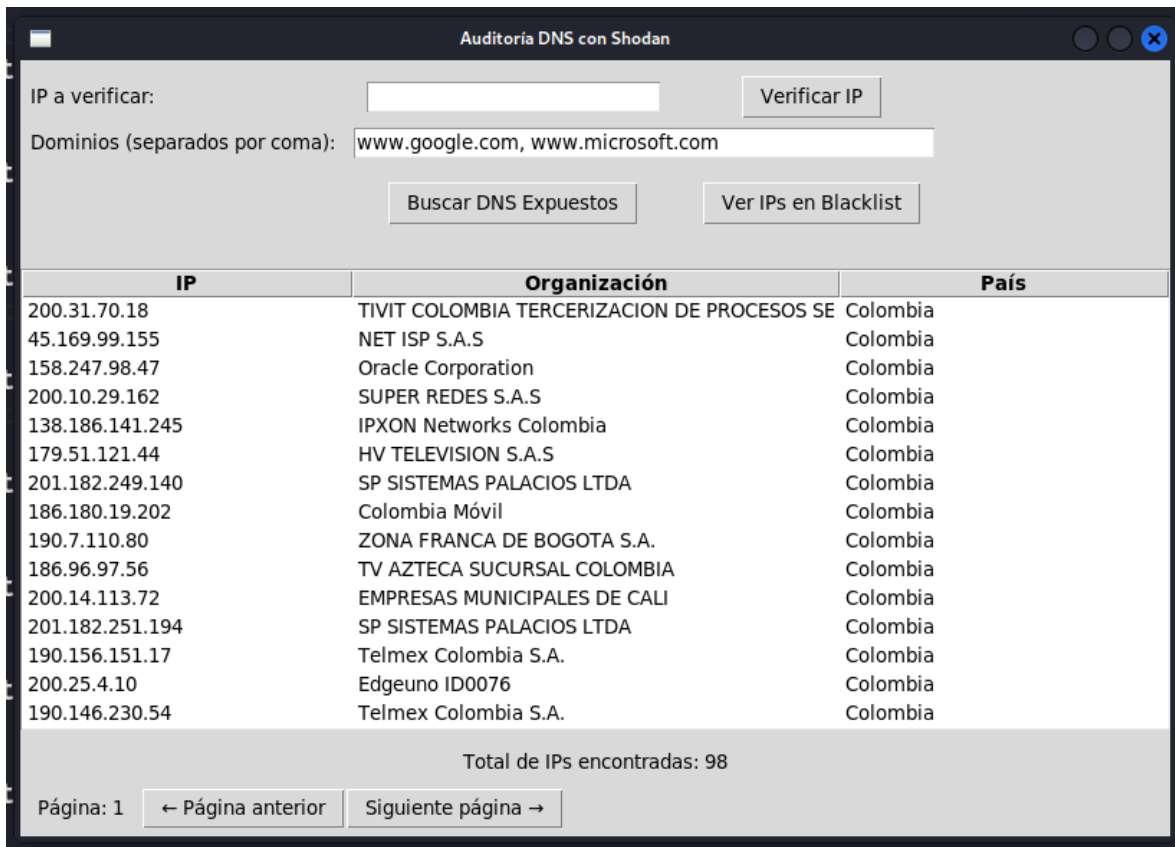


Imagen #19. Versión final del aplicativo.

Como podemos observar tenemos un apartado para verificar la IP que queramos y observar si resuelve los dominios que añadamos, también vemos un botón para corroborar que IPs están en las blacklist y por último tenemos una paginación y un texto para observar cuantas IPs fueron encontradas.

Conclusiones

1. Papel fundamental del DNS en la seguridad informática

Los servidores DNS son esenciales para el funcionamiento de Internet, ya que convierten los nombres de dominio en direcciones IP comprensibles por las máquinas. Cuando estos servidores están mal configurados o expuestos, pueden convertirse en vectores de ataque utilizados para redirección maliciosa, secuestro de tráfico o campañas de denegación de servicio.

2. Beneficios de automatizar auditorías DNS

Automatizar la recolección y verificación de información DNS agiliza enormemente el proceso de análisis de seguridad. Al usar APIs como la de Shodan junto con consultas DNS personalizadas, se pueden identificar rápidamente configuraciones inseguras o servidores potencialmente peligrosos sin depender de herramientas manuales.

3. Importancia de la disponibilidad DNS

Verificar que los servidores DNS estén activos y accesibles es clave para la estabilidad de servicios web. Un servidor DNS inaccesible puede causar caídas, errores de navegación o interrupciones en plataformas críticas. La validación de su respuesta permite anticipar problemas de disponibilidad.

4. Evaluación funcional mediante resolución de dominios

Probar si un servidor DNS resuelve correctamente nombres de dominio conocidos, como `www.google.com`, permite determinar si funciona adecuadamente o si presenta anomalías. Un fallo en la resolución podría indicar errores de configuración, saturación del servicio o manipulación externa.

5. Control de resolución recursiva

El comportamiento recursivo de un servidor DNS puede convertirse en una vulnerabilidad si está mal gestionado. Servidores que permiten resolución recursiva sin restricción pueden ser explotados para amplificar ataques DDoS. Por eso, es vital auditar si los DNS analizados presentan este riesgo.

6. Detección de IPs peligrosas mediante blacklist

Comparar las IPs recolectadas con una lista negra permite filtrar aquellos servidores con antecedentes maliciosos o comportamiento sospechoso. Esta funcionalidad es clave para prevenir que estos elementos formen parte de una infraestructura confiable o sean utilizados en redes empresariales.