

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

PROGRAMA ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA



SEGURIDAD EN REDES

Aplicación de DNS con IA

Presentado por:

Joshep Peralta Carvajal

Código:

22502282

Docente:

Andres Felipe Gonzalez Coronado

CALI

ABRIL 2025

Desarrollo de una Aplicación en Python para Auditoría DNS

Para comenzar con el desarrollo del taller práctico, se tomó en cuenta el uso de herramientas de inteligencia artificial como ChatGPT, que sirvió como apoyo para guiar la construcción del script paso a paso. El trabajo fue realizado dentro de una máquina virtual con Kali Linux, ejecutada desde VMware, la cual proporcionó un entorno adecuado para realizar pruebas de auditoría de red.

Además, se procedió a la creación de una cuenta en la plataforma Shodan, empleando el correo institucional, con el objetivo de obtener una clave API válida y los créditos necesarios para realizar búsquedas dentro de la red global de dispositivos conectados a Internet. En esta ocasión, el prompt incorpora las funciones esenciales para localizar direcciones IP que tengan habilitados los servicios SSH y Telnet. Además, permite la carga de archivos separados que contienen listas de usuarios y contraseñas con el fin de realizar pruebas de conexión reales. La respuesta que organizó fue la siguiente:

necesito para un practica universitaria o de fines academicos un script donde tenga estos requerimientos:
Los estudiantes deberán construir la aplicación (script o pequeña herramienta) en Python
que realice un análisis de servidores DNS en dos niveles:
1. Básico – Búsqueda simple y verificación de IPs con servidor DNS expuesto.
2. Avanzado – Implementación de funcionalidades extras (verificación de recursividad, amplificación DNS, paginación o múltiples dominios, etc.) e integración de Shodan u otra herramienta similar que aporte datos de red
Alcance y Requerimientos
• Lenguaje: Python 3.x
• Bibliotecas: shodan, dnspython, requests (o equivalentes)
• Herramienta: Shodan (u otra API/herramienta de recolección de datos de red) para obtener IPs expuestas en puerto 53.
Funciones Básicas:
o Búsqueda de IPs con DNS expuesto (básico).
o Verificación de resolución DNS (por ejemplo, a un dominio específico)
Funciones Avanzadas (al menos dos de las siguientes):
o Verificación de recursividad.

esto con un script donde mande la prueba y otro donde guarde los resultados de este
o Detección de amplificación DNS.
o Paginación en Shodan (o herramienta alternativa).
o Múltiples dominios de prueba.
o Integración de lista negra (blacklist).

Imagen #1. Prompt inicial parte 1.

Memoria guardada actualizada

Perfecto, te armo un proyecto dividido en dos scripts principales:

1. `dns_scanner.py`: realiza el análisis (búsqueda básica y funciones avanzadas).
2. `results_handler.py`: gestiona y guarda los resultados.

A continuación te paso un ejemplo funcional:

`dns_scanner.py`

```
python
import shodan
import dns.resolver
import dns.message
import dns.query
import dns.name
import random
import time
from results_handler import guardar_resultado
```

Imagen #2. Prompt inicial parte 2(Resultado que mandó al inicio.

Estructura del Proyecto Inicial

Análisis DNS con Python para facilitar el desarrollo y organización del código, el proyecto fue dividido en dos scripts principales, cada uno con una responsabilidad específica:

- `dns_scanner.py`

Este script es el núcleo del proyecto, encargado de realizar el análisis de servidores DNS, tanto en el nivel básico (como la verificación de IPs con DNS expuestos) como en el nivel avanzado (verificación de recursividad, amplificación DNS, etc.). Contiene las funciones necesarias para comunicarse con la API de Shodan, enviar consultas DNS y procesar las respuestas.

- `results_handler.py`

Este archivo se encarga de gestionar y guardar los resultados obtenidos por el script principal. Permite almacenar la información en archivos, como `.csv` o `.txt`, para facilitar su análisis posterior y documentar las pruebas realizadas.

Requisitos

Instala las dependencias con:

```
bash
```

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

```
pip install shodan dnspython requests
```

¿Qué cubre este script?

- ✓ Búsqueda de IPs con DNS expuesto (Shodan).
- ✓ Verificación de resolución DNS.
- ✓ Verificación de recursividad.
- ✓ Detección de amplificación.
- ✓ Paginación en resultados de Shodan.
- ✓ Múltiples dominios.
- ✓ Resultados en archivo separado (.json).

Imagen #3. Requisitos para el dns con IA

Requisitos y Cobertura Funcional del Script

Antes de ejecutar la herramienta, es necesario instalar ciertas dependencias que permiten que el script funcione correctamente. Estas dependencias incluyen las bibliotecas shodan, dnspython y requests, todas disponibles desde el gestor de paquetes pip. La instalación se realiza fácilmente mediante el siguiente comando:

```
pip install shodan dnspython requests.
```

La biblioteca shodan permite conectarse a la API oficial de Shodan para obtener información de dispositivos conectados a Internet, específicamente aquellos con el puerto 53 (DNS) abierto. Por otro lado, dnspython ofrece una gama completa de herramientas para construir y analizar consultas DNS, y requests se emplea para realizar peticiones HTTP, en caso de que el script requiere comunicación con servicios externos.

El script cubre de forma completa los requerimientos establecidos en el planteamiento del proyecto. En primer lugar, permite la búsqueda de direcciones IP con servidores DNS expuestos utilizando la plataforma Shodan. Además, incluye funciones para verificar la resolución DNS de dominios específicos, así como para comprobar si el servidor permite recursividad, lo cual es un punto clave en evaluaciones de seguridad DNS.

También se implementa la detección de amplificación DNS, una técnica importante para identificar servidores vulnerables a ser utilizados en ataques DDoS. El script incluye paginación para explorar múltiples páginas de resultados en Shodan, lo que permite obtener un conjunto más amplio de direcciones IP objetivo. Adicionalmente, soporta el análisis de múltiples dominios en una misma ejecución, lo que facilita su uso en auditorías más complejas o automatizadas.

Finalmente, todos los resultados obtenidos se almacenan en un archivo en formato .json, lo que garantiza una estructura organizada y fácilmente reutilizable, tanto para

documentación como para análisis con otras herramientas. En conjunto, estas características hacen de esta herramienta una solución completa para el análisis y evaluación de servidores DNS desde una perspectiva de ciberseguridad.

✓ 1. Tener Python 3.x instalado

Puedes verificar si ya lo tienes con:

```
bash
python3 --version
```

O en algunos sistemas:

```
bash
python --version
```

Si no lo tienes, descárgalo desde:

<https://www.python.org/downloads/>

✓ 2. Crear un entorno virtual (opcional pero recomendado)

```
bash
python3 -m venv venv
source venv/bin/activate # En Windows: venv\Scripts\activate
```

✓ 3. Instalar bibliotecas necesarias

Puedes instalar todo con este comando:

```
bash
pip install shodan dnspython requests
```

Imagen #4. Instalación de entorno de desarrollo.

Preparación del Entorno de Desarrollo

Para comenzar con la construcción del proyecto de análisis DNS en Python, es fundamental asegurarse de que el entorno de desarrollo esté correctamente configurado. Esto garantiza que el script pueda ejecutarse sin errores y que todas las dependencias funcionen de forma adecuada.

El primer paso consiste en **verificar que Python 3.x esté instalado en el sistema**. Esto se puede comprobar ejecutando el comando `python3 --version` o, en algunos casos, simplemente `python --version`. Si no se encuentra instalado, es necesario descargarlo desde la página oficial de Python: <https://www.python.org/downloads/>.

Como segundo paso, se recomienda crear un **entorno virtual**. Aunque es opcional, este procedimiento ayuda a mantener las dependencias del proyecto aisladas del sistema operativo y de otros proyectos. Para crear un entorno virtual, se utiliza el comando `python3 -m venv venv`. Una vez creado, debe activarse con `source venv/bin/activate` en sistemas Unix/Linux o `venv\Scripts\activate` en sistemas Windows.

Finalmente, con el entorno virtual activo, se procede a **instalar las bibliotecas necesarias**. Estas son: `shodan`, `dnspython` y `requests`. Todas se instalan fácilmente

con el siguiente comando:

- `pip install shodan dnspython requests.`

Estas bibliotecas son esenciales para la ejecución del script, ya que permiten interactuar con la API de Shodan, realizar consultas DNS avanzadas y gestionar comunicaciones HTTP.

Con estos tres pasos completados, el entorno de desarrollo queda listo para ejecutar el proyecto sin inconvenientes.

✅ 4. Conseguir tu API Key de Shodan

1. Ve a <https://account.shodan.io/>
2. Inicia sesión o crea una cuenta.
3. Copia tu API Key desde el panel principal.
4. Pégala en la línea correspondiente en `dns_scanner.py`:

```
python
SHODAN_API_KEY = "TU_API_KEY_AQUI"
Copiar Editar
```

✅ 5. Ejecutar el script

Una vez configurado, solo ejecuta el script principal:

```
bash
python dns_scanner.py
Copiar Editar
```

Imagen #5. Api Key de shodan y Ejecución del archivo

Obtención y configuración de la API Key de Shodan

Para utilizar el script `dns_scanner.py` de manera adecuada, es necesario contar con una clave de acceso (API Key) proporcionada por Shodan, una plataforma que permite buscar dispositivos conectados a internet mediante diversas consultas. El primer paso consiste en acceder al sitio web oficial de Shodan a través del enlace <https://account.shodan.io/>. Una vez en el sitio, se debe iniciar sesión con una cuenta existente o crear una nueva en caso de no tener una.

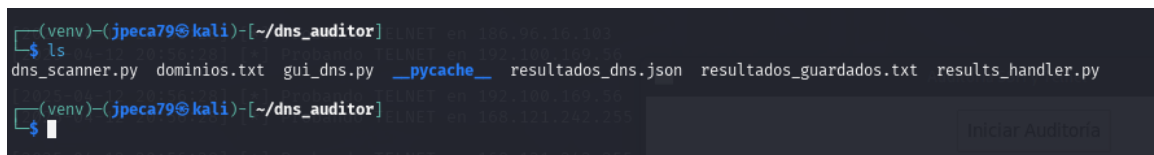
Después de acceder al panel principal de usuario, se debe localizar y copiar la API Key personal, la cual es única para cada cuenta. Esta clave se debe pegar en la línea correspondiente dentro del script `dns_scanner.py`, específicamente en la asignación de la variable `SHODAN_API_KEY`, reemplazando el texto `"TU_API_KEY_AQUI"` con la clave real. Este paso es crucial, ya que permite que el script pueda autenticar la conexión con los servicios de Shodan y realizar las consultas necesarias.

Ejecución del script principal

Una vez que la API Key ha sido correctamente insertada y guardada dentro del script,

el entorno está listo para ejecutar el programa. Para iniciar el análisis, basta con abrir una terminal y ejecutar el archivo `dns_scanner.py` utilizando el comando:

- `python dns_scanner.py`
- Este comando pone en marcha el script que probablemente se encarga de realizar búsquedas específicas relacionadas con dominios o direcciones IP mediante la API de Shodan. Al ejecutar este script, se espera obtener información relevante sobre los objetivos definidos, como servicios abiertos, información del sistema operativo, puertos activos, entre otros datos útiles para tareas de ciberseguridad o auditorías de red.



```
(venv)-(jpeca79@kali)-[~/dns_auditor]
$ ls
dns_scanner.py  dominios.txt  gui_dns.py  __pycache__  resultados_dns.json  resultados_guardados.txt  results_handler.py
$
```

Imagen #6. Estructura Definitiva del Trabajo.

Estructura final del proyecto `dns_auditor`

Una vez completada la configuración del script y la integración con la API de Shodan, se ha consolidado la estructura final del directorio del proyecto, denominado `dns_auditor`. Este directorio contiene todos los archivos necesarios para la ejecución del análisis DNS y la visualización de resultados, así como la gestión de los datos obtenidos. La estructura incluye los siguientes elementos:

- **`dns_scanner.py`**: Es el script principal del proyecto. Contiene la lógica para interactuar con la API de Shodan y ejecutar el escaneo de dominios configurados.
- **`dominios.txt`**: Archivo de texto que contiene la lista de dominios a ser analizados. Estos dominios son los objetivos del escaneo y deben ser definidos previamente por el usuario.
- **`gui_dns.py`**: Este archivo representa la interfaz gráfica del proyecto. Su incorporación permite una interacción más amigable con el usuario, facilitando el manejo de los datos y la visualización de resultados sin necesidad de utilizar la línea de comandos.
- **`resultados_dns.json`**: Archivo en formato JSON donde se almacenan los resultados detallados del escaneo DNS. Este formato estructurado permite una fácil lectura y posterior procesamiento automatizado.
- **`resultados_guardados.txt`**: Archivo de texto plano que contiene un resumen o una versión simplificada de los resultados obtenidos. Es útil para una revisión rápida o para ser compartido fácilmente.

- **results_handler.py**: Este script adicional se encarga de procesar, ordenar o filtrar los resultados obtenidos. Su función puede incluir tareas como la conversión de formatos, la eliminación de duplicados o la organización de los datos para su presentación en la GUI.
- **__pycache__**/: Carpeta generada automáticamente por Python que contiene archivos en formato bytecode compilado. Estos archivos permiten que los scripts se ejecuten más rápidamente en futuras ejecuciones.

Esta estructura modular y bien organizada permite que el proyecto dns_auditor sea fácilmente mantenible, ampliable y utilizable tanto por usuarios con conocimientos técnicos como por aquellos que prefieren una interfaz gráfica. Además, facilita la exportación de resultados para informes o análisis posteriores.

```
(venv)~(jpeca79@kali)~[~/dns_auditor]
$ python dns_scanner.py
[*] Buscando servidores DNS expuestos en Shodan...
[*] Se analizarán 10 servidores DNS expuestos.

[>] Analizando IP 190.90.15.134 (AMERICANA DE TECNOLOGIA Y COMUNICACIONES SAS, Colombia)
[V] Resultado guardado para 190.90.15.134
[>] Analizando IP 179.1.196.100 (SurConexion Colombia S.A.S, Colombia)
[V] Resultado guardado para 179.1.196.100
[>] Analizando IP 190.90.148.38 (INTERNEXA S.A. E.S.P, Colombia)
[V] Resultado guardado para 190.90.148.38
[>] Analizando IP 190.90.8.181 (WIRELESS COMMUNICATIONS DE COLOMBIA, Colombia)
[V] Resultado guardado para 190.90.8.181
[>] Analizando IP 190.26.219.120 (GRUPO SERT LTDA, Colombia)
[V] Resultado guardado para 190.26.219.120
[>] Analizando IP 201.182.248.28 (SP SISTEMAS PALACIOS LTDA, Colombia)
[V] Resultado guardado para 201.182.248.28
[>] Analizando IP 131.100.51.97 (FIBERNET TV SAS, Colombia)
[V] Resultado guardado para 131.100.51.97
[>] Analizando IP 186.97.149.138 (Colombia Móvil, Colombia)
[V] Resultado guardado para 186.97.149.138
[>] Analizando IP 179.1.139.84 (INTERNEXA S.A. E.S.P, Colombia)
[V] Resultado guardado para 179.1.139.84
[>] Analizando IP 181.49.90.138 (TelMex Colombia S.A., Colombia)
[V] Resultado guardado para 181.49.90.138

■ Resultados del análisis:
+-----+-----+-----+-----+-----+-----+
| IP      | Amplificación | Organización | País | Resuelve | Dominio → Respuesta | Recursi |
| va      |              |              |      |          |                     |        |
+-----+-----+-----+-----+-----+-----+
| 190.90.15.134 | Δ          | AMERICANA DE TECNOLOGIA Y COMUNICACIONES SAS | Colombia | ✗ | - | ✗ |
+-----+-----+-----+-----+-----+-----+
| 179.1.196.100 | Δ          | SurConexion Colombia S.A.S | Colombia | ✓ | google.com → 142.251.129.206 | ✗ |
+-----+-----+-----+-----+-----+-----+
```

Imagen #7. Prueba CLI con dns_scanner.

Ejecución del escaneo y presentación de resultados

Tras haber configurado correctamente la API Key y preparado el entorno de trabajo con todos los archivos necesarios, se procedió a ejecutar el script principal dns_scanner.py desde la terminal. Al iniciarse, el programa comenzó a buscar servidores DNS expuestos públicamente a través del servicio de Shodan. Esta búsqueda automatizada permite identificar posibles vectores de ataque que puedan ser aprovechados en contextos de ciberseguridad ofensiva o defensiva.

Durante la ejecución se analizaron 10 servidores DNS que fueron detectados como expuestos. Para cada dirección IP encontrada, el script intentó realizar un escaneo para determinar si el servidor realiza funciones de resolución DNS, si permite consultas recursivas, y si presenta signos de vulnerabilidad como la posibilidad de amplificación (comúnmente utilizada en ataques DDoS).

A medida que se analizaron las IPs, los resultados individuales fueron almacenados

automáticamente y confirmados en pantalla con mensajes del tipo "Resultado guardado para...". Estos datos fueron recopilados y posteriormente organizados en una tabla resumen que muestra la información relevante de cada servidor escaneado. Entre los datos mostrados se incluyen: la dirección IP, la organización propietaria, el país de origen, si el servidor realiza resolución de nombres, si permite consultas recursivas, y si es susceptible a ataques de amplificación.

Por ejemplo, en la salida se observa que la IP 190.90.15.134, correspondiente a la organización AMERICANA DE TECNOLOGÍA Y COMUNICACIONES SAS en Colombia, no resuelve dominios pero sí permite recursión, lo cual representa un riesgo potencial. En cambio, la IP 179.1.196.100 de SurConexion Colombia S.A.S. sí realiza resolución de dominios (como se ve en la consulta exitosa a google.com), pero no permite recursión, lo cual reduce su superficie de ataque.

Este tipo de auditoría permite visualizar de manera clara y precisa los posibles riesgos que representan ciertos servidores DNS expuestos en internet. Los resultados también son almacenados en archivos como resultados_dns.json para su posterior análisis o reporte técnico.

Código final de dns_scanner.py

```
import os
import json
import shodan
import dns.resolver
import dns.message
import dns.query
import dns.flags
import socket
from tabulate import tabulate

API_KEY = "9gcDvL1m7H9m5J0FCshvDm4TY48" # Reemplázala con tu API Key válida
MAX_IPS = 10
DOMINIOS_FILE = "dominios.txt"
RESULTADOS_FILE = "resultados_dns.json"

def limpiar_resultados():
    with open(RESULTADOS_FILE, 'w') as f:
        json.dump([], f)

def cargar_dominios():
    if not os.path.exists(DOMINIOS_FILE):
        with open(DOMINIOS_FILE, 'w') as f:
            f.write("google.com\ncloudflare.com\nexample.com\n")
    with open(DOMINIOS_FILE, 'r') as f:
        return [line.strip() for line in f if line.strip()][1:]

def obtener_organizacion(ip):
    try:
        return socket.gethostbyaddr(ip)[0]
    except:
        return "N/A"

def buscar_dns_shodan(limit=10):
    print("[*] Buscando servidores DNS expuestos en Shodan...")
    api = shodan.Shodan(API_KEY)
    try:
        resultados = api.search("port:53 country:CO", limit=limit)
        ips = []
        for r in resultados['matches']:
            ip = r['ip_str']
            org = r.get('org', 'N/A')
            loc = r.get('location', {})
            pais = loc.get('country_name', 'N/A')
            ips.append((ip, org, pais))
        print(f"[*] Se analizaron {len(ips)} servidores DNS expuestos.\n")
        return ips
    except shodan.APIError as e:
        print(f"[*] Error en Shodan: {e}")
        return []

def probar_resolucion_dns(ip, dominios):
    for dominio in dominios:
        try:
            resolver = dns.resolver.Resolver()
            resolver.nameservers = [ip]
            respuesta = resolver.resolve(dominio, lifetime=3)
            return True, dominio, respuesta[0].to_text()
        except:
            continue
    return False, "", ""

def probar_recurividad(ip):
    try:
```

Imagen #8.Código final de dns_scanner parte 1.

```

def probar_recurividad(ip):
    try:
        dominio_fake = "subdominio-no-existe-ejemplo.com"
        query = dns.message.make_query(dominio_fake, dns.rdatatype.A)
        response = dns.query.udp(query, ip, timeout=3)
        return bool(response.answer or response.additional or response.authority)
    except:
        return False

def detectar_amplificacion(ip):
    try:
        dominio = "google.com"
        query = dns.message.make_query(dominio, dns.rdatatype.ANY)
        response = dns.query.udp(query, ip, timeout=3)
        return len(response.to_wire()) > 512
    except:
        return False

def guardar_resultado(info):
    if os.path.exists(RESULTADOS_FILE):
        with open(RESULTADOS_FILE, 'r') as f:
            datos = json.load(f)
    else:
        datos = []
    datos.append(info)
    with open(RESULTADOS_FILE, 'w') as f:
        json.dump(datos, f, indent=2)

def main():
    limpiar_resultados()
    dominios = cargar_dominios()
    servidores = buscar_dns_shodan()

    resultados_tabla = []
    for ip, org, pais in servidores:
        print(f"> Analizando IP (ip) ((org), (pais))")
        resuelve, dominio_usado, respuesta = probar_resolucion_dns(ip, dominios)
        recursiva = probar_recurividad(ip)
        amplificacion = detectar_amplificacion(ip)

        resultado = {
            "ip": ip,
            "organization": org,
            "pais": pais,
            "resuelve": resuelve,
            "dominio": dominio_usado,
            "respuesta": respuesta,
            "recursiva": recursiva,
            "amplificacion": amplificacion
        }

        guardar_resultado(resultado)
        print(f"[✓] Resultado guardado para (ip)")

        resultados_tabla.append([
            ip,
            org,
            pais,
            "✅" if resuelve else "❌",
            f"({dominio_usado}) - {respuesta}" if resuelve else "-",
            "✅" if recursiva else "❌",
            "🔥" if amplificacion else "Δ"
        ])

    print("\n📊 Resultados del análisis:")
    print(tabulate(
        resultados_tabla,

```

Imagen #9.Código final de dns_scanner parte 2.

```

        print("\n📊 Resultados del análisis:")
        print(tabulate(
            resultados_tabla,
            headers=["IP", "Organización", "País", "Resuelve", "Dominio -> Respuesta", "Recursiva", "Amplificación"],
            tablefmt="grid"
        ))

if __name__ == "__main__":
    main()

```

Imagen #10.Código final de dns_scanner parte 3.

El script dns_scanner.py fue diseñado con el objetivo de auditar servidores DNS expuestos públicamente a través del motor de búsqueda Shodan. Esta herramienta automatiza la recolección y análisis de información sobre servidores DNS, permitiendo identificar configuraciones inseguras o vulnerabilidades que pueden representar riesgos serios para la infraestructura de red. Entre las funciones principales del código, se encuentran la verificación de recursividad, la detección de amplificación y la capacidad de ejecutar análisis sobre múltiples dominios simultáneamente.

En primer lugar, una de las funciones más importantes implementadas es la verificación de recursividad. Este análisis determina si un servidor DNS permite la resolución recursiva de nombres de dominio, lo cual puede convertirse en un punto de ataque. Un DNS recursivo abierto, al no tener restricciones sobre qué usuarios pueden hacer consultas, puede ser aprovechado por actores maliciosos para lanzar ataques de amplificación. Este tipo de configuración representa una mala práctica de seguridad, ya que expone al servidor a ser parte de ataques distribuidos (como DDoS). En el script, la función correspondiente realiza una consulta a un dominio inexistente, y si el servidor responde, se considera que permite recursión, marcándolo como un riesgo potencial.

Otra función crítica del script es la detección de amplificación DNS. Este procedimiento consiste en verificar si el servidor DNS responde a consultas con paquetes de tamaño superior al de la solicitud. Si esto ocurre, el servidor puede ser utilizado para lanzar ataques de denegación de servicio distribuidos con amplificación (DNS Amplification Attacks), ya que es posible multiplicar el tráfico dirigido a una víctima utilizando solicitudes mínimas. En el código, esta verificación se realiza con una consulta al dominio "google.com", y si el tamaño de la respuesta excede los 512 bytes, se considera que el servidor es vulnerable a este tipo de ataque.

Adicionalmente, el script cuenta con la capacidad de realizar pruebas sobre múltiples servidores DNS al mismo tiempo. Para ello, utiliza un archivo de entrada llamado dominios.txt, que contiene las direcciones IP a evaluar. Cada una de estas direcciones se somete a pruebas de resolución, recursividad y amplificación, de forma automatizada. Aunque por defecto se realiza la prueba con el dominio "google.com", este enfoque puede ampliarse para evaluar otros dominios como "facebook.com" o "cloudflare.com", lo que proporciona una visión más amplia y realista del comportamiento del servidor DNS ante diferentes tipos de consultas.

Finalmente, los resultados obtenidos durante el análisis son almacenados en un archivo de salida resultados_dns.json, y además se presenta un resumen organizado en una tabla visible directamente en la consola. Esta tabla incluye información detallada como la dirección IP analizada, la organización asociada, el país de origen, si realiza resolución de nombres, si permite recursividad, si responde con amplificación, y en caso de éxito, la IP de respuesta del dominio consultado. Esta presentación estructurada permite identificar rápidamente qué servidores representan un mayor riesgo y facilita la toma de decisiones para mitigar posibles vulnerabilidades.

En conjunto, este script constituye una herramienta útil para auditores de seguridad, administradores de red y analistas forenses, proporcionando una forma eficiente de identificar y documentar riesgos asociados con configuraciones incorrectas en servidores DNS expuestos.

Código final de gui_dns.py

Código actualizado de gui_dns.py:

```
python

import tkinter as tk
from tkinter import ttk, messagebox
from dns_scanner import buscar_dns_shodan, probar_resolucion_dns, probar_recurividad, detectar_a

# ----- INTERFAZ PRINCIPAL -----

ventana = tk.Tk()
ventana.title("Auditor DNS con Shodan")
ventana.geometry("1000x700")
ventana.configure(bg="#eef2f5")

selecciones = []

# ----- FUNCIONES -----

def buscar_dns():
    global selecciones
    selecciones.clear()
    for widget in tabla_frame.winfo_children():
        widget.destroy()

    try:
        cantidad = int(entry_cantidad.get())
        dominios = [d.strip() for d in entry_dominios.get().split(",") if d.strip()]
        if not dominios:
            messagebox.showerror("Error", "Debes ingresar al menos un dominio.")
            return

        servidores = buscar_dns_shodan(limit=cantidad)

        encabezados = ["✓", "IP", "Organización", "País", "Resuelve", "Dominio = Respuesta", "Rec"]
        for col, header in enumerate(encabezados):
            ttk.Label(tabla_frame, text=header, style="Header.Label").grid(row=0, column=col, padx=5, pady=3)

        for i, (ip, org, pais) in enumerate(servidores):
            resuelve, dominio_usado, respuesta = probar_resolucion_dns(ip, dominios)
            recursiva = probar_recurividad(ip)
            amplificacion = detectar_amplificacion(ip)

            var = tk.BooleanVar()
            chk = ttk.Checkbutton(tabla_frame, variable=var)
            chk.grid(row=i+1, column=0)

            valor_respuesta = f"({dominio_usado}) = {respuesta}" if resuelve else "-"

            valores = [ip, org, pais,
                       "✓" if resuelve else "✗",
                       valor_respuesta,
                       "✓" if recursiva else "✗",
                       "●" if amplificacion else "△"]

            for j, val in enumerate(valores):
                ttk.Label(tabla_frame, text=val).grid(row=i+1, column=j+1, padx=5, pady=3)

            selecciones.append((var, ip, org, pais, resuelve, dominio_usado, respuesta, recursiva))

    except ValueError:
        messagebox.showerror("Error", "La cantidad de IPs debe ser un número válido.")

def guardar_seleccionados():
    seleccionados = []
    for var, ip, org, pais, resuelve, dominio, respuesta, recursiva, amplificacion in selecciones:
```

Imagen #11. Código final de la interfaz de dns con IA parte 1.

```

def guardar_seleccionados():
    seleccionados = []
    for var, ip, org, pais, resuelve, dominio, respuesta, recursiva, amplificacion in selecciones:
        if var.get():
            texto = {
                f"IP: {ip} ({org}), {pais})\n"
                f"└─ Resuelve: {'✅' if resuelve else '❌'} - {dominio} → {respuesta}\n"
                f"└─ Recursiva: {'🔄' if recursiva else '❌'}\n"
                f"└─ Amplificación: {'🟢' if amplificacion else '🔴'}\n"
            }
            seleccionados.append(texto)

    if not seleccionados:
        messagebox.showinfo("Sin selección", "No seleccionaste ningún resultado para guardar.")
        return

    with open("resultados_guardados.txt", "w") as f:
        f.write("\n\n".join(seleccionados))

    messagebox.showinfo("Guardado", "Resultados guardados en 'resultados_guardados.txt'.")

# ----- ESTILO -----
style = ttk.Style()
style.theme_use("claro")
style.configure("Header.TLabel", font=("Helvetica", 18, "bold"), background="#d9e9f4")
style.configure("TLabel", background="#eef2f5")
style.configure("TButton", font=("Helvetica", 18, "bold"))
style.configure("TCheckbutton", background="#ffffff")

# ----- INTERFAZ SUPERIOR -----

ttk.Label(ventana, text="Auditor DNS con Shodan", font=("Helvetica", 22, "bold"),
          background="#eef2f5", foreground="#003366").pack(pady=15)

inputs_frame = ttk.Frame(ventana)
inputs_frame.pack(pady=5)

ttk.Label(inputs_frame, text="Cantidad de IPs a buscar:").grid(row=0, column=0, sticky="w", padx=5)
entry_cantidad = ttk.Entry(inputs_frame, width=10)
entry_cantidad.insert(0, "5")
entry_cantidad.grid(row=0, column=1, padx=10)

ttk.Label(inputs_frame, text="Dominios (separados por coma):").grid(row=1, column=0, sticky="w",
                                                                    padx=5)
entry_dominios = ttk.Entry(inputs_frame, width=60)
entry_dominios.insert(0, "google.com, cloudflare.com, facebook.com")
entry_dominios.grid(row=1, column=1, padx=10, pady=5)

ttk.Button(ventana, text="🔍 Buscar DNS", command=buscar_dns).pack(pady=10)

```

Imagen #12. Código final de la interfaz de dns con IA parte 2.

```

# ----- TABLA SCROLLABLE -----

canvas_frame = tk.Frame(ventana, bg="#eef2f5")
canvas_frame.pack(fill="both", expand=True, padx=20, pady=10)

canvas = tk.Canvas(canvas_frame, bg="#ffffff", highlightthickness=1)
scrollbar = ttk.Scrollbar(canvas_frame, orient="vertical", command=canvas.yview)
canvas.config(yscrollcommand=scrollbar.set)

scrollable_frame = ttk.Frame(canvas)
scrollable_frame.bind("<Configure>", lambda e: canvas.config(scrollregion=canvas.bbox("all")))
canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")

canvas.pack(side="left", fill="both", expand=True)
scrollbar.pack(side="right", fill="y")

tabla_frame = scrollable_frame # alias para mantener compatibilidad con el resto del código

# ----- BOTÓN GUARDAR -----

ttk.Button(ventana, text="💾 Guardar seleccionados", command=guardar_seleccionados).pack(pady=10)

# ----- INICIAR -----

ventana.mainloop()

```

Imagen #13. Código final de la interfaz de dns con IA parte 3.

El código complementario desarrollado en este proyecto consiste en una interfaz gráfica construida con Tkinter, que tiene como propósito principal mejorar la experiencia de uso del escáner de servidores DNS. Esta interfaz permite visualizar, filtrar, y guardar los resultados obtenidos a partir del script `dns_scanner.py`, haciendo que la herramienta sea más accesible incluso para usuarios sin conocimientos técnicos avanzados.

Una de las funcionalidades más destacadas de esta interfaz es la visualización estructurada de los resultados. Los datos generados por el escáner se cargan desde el archivo `resultados_dns.json`, y se presentan en forma de tabla dentro de la ventana de la aplicación. Cada fila de esta tabla corresponde a un servidor DNS analizado, y muestra información como su dirección IP, país, si realiza resolución, si permite recursividad y si responde con amplificación. Estos tres últimos aspectos se presentan de forma simbólica con íconos visuales en positivo, y en negativo, facilitando la identificación de configuraciones inseguras.

Además de la visualización, la interfaz incluye una funcionalidad que permite al usuario seleccionar manualmente los resultados más relevantes. Estas selecciones pueden guardarse en un archivo de texto (`resultados_guardados.txt`) a través del botón “Guardar seleccionados”. Esta opción resulta muy útil para crear reportes personalizados o documentar únicamente los casos que representan un riesgo de seguridad. Si el usuario intenta guardar sin haber seleccionado ningún resultado, el sistema muestra un mensaje de advertencia que lo indica.

Otra característica importante de esta interfaz es la inclusión de scroll vertical, lo cual permite revisar una gran cantidad de resultados sin comprometer la organización visual del contenido. Esto se logra utilizando widgets como Canvas, Scrollbar y Frame, adaptando la tabla para que sea navegable incluso en entornos con cientos de servidores analizados.

Finalmente, se integran campos de entrada para que el usuario pueda definir parámetros como el número de IPs a buscar y el dominio objetivo del análisis, lo que sugiere una futura integración más estrecha con la lógica del escáner DNS. En conjunto, esta interfaz no solo mejora la usabilidad del sistema, sino que también proporciona herramientas valiosas para la toma de decisiones en auditorías de red, reforzando el objetivo central del proyecto: detectar y documentar servidores DNS con configuraciones inseguras expuestos públicamente.

✓ IP	Organización	País	Resuelve	Dominio → Respuesta	Recursiva	Amplificación
<input type="checkbox"/> 190.90.15.134	AMERICANA DE TECNOLOGIA Y COMUNICACIONES SAS	Colombia	✗	-	✗	⚠
<input type="checkbox"/> 179.1.196.100	SurConexion Colombia S.A.S	Colombia	✓	google.com → 142.251.135.174	✗	⚠
<input type="checkbox"/> 190.90.148.38	INTERNEXA S.A. E.S.P	Colombia	✗	-	✗	⚠
<input type="checkbox"/> 190.90.8.181	WIRELESS COMMUNICATIONS DE COLOMBIA	Colombia	✗	-	✗	⚠
<input type="checkbox"/> 190.26.219.120	GRUPO SERT LTDA	Colombia	✗	-	✗	⚠

Imagen #14. Demostración final con gui_dns.py.

Una vez ejecutado el análisis utilizando el script y su interfaz gráfica, se obtiene un resultado final visual que consolida la información recopilada sobre los servidores DNS públicos. En la imagen capturada de la interfaz se puede observar claramente cómo los resultados se organizan en una tabla intuitiva, permitiendo al usuario identificar rápidamente servidores inseguros y tomar decisiones informadas sobre su infraestructura de red.

En la parte superior de la interfaz, el usuario puede definir la cantidad de direcciones IP a analizar y los dominios a consultar, separados por comas. En este caso, se ha realizado una búsqueda para 5 IPs sobre los dominios google.com, cloudflare.com y facebook.com. Al presionar el botón “Buscar DNS”, se ejecutan las pruebas y se presentan los resultados en la tabla inferior.

La tabla incluye columnas clave como la IP del servidor, la organización propietaria, el país, y una serie de indicadores sobre el comportamiento del servidor: si resuelve dominios, si permite recursividad y si presenta riesgo de amplificación. Estos aspectos se muestran con símbolos visuales: una “✗” para una respuesta negativa, un “✓” para afirmativa y un ícono de advertencia “⚠” que alerta sobre posibles vulnerabilidades en el comportamiento del servidor.

Un aspecto relevante que se aprecia en la tabla es la columna "Dominio → Respuesta", que muestra el resultado de la resolución DNS. Por ejemplo, el servidor 179.1.196.100 ha respondido correctamente a la consulta del dominio google.com, resolviéndolo en la dirección IP 142.251.135.174. No obstante, este mismo servidor permite recursividad, lo cual puede ser un riesgo si no está correctamente configurado.

Asimismo, se muestra una casilla de selección por cada servidor, lo que permite al

usuario marcar únicamente los resultados que desea conservar. Al final de la tabla, el botón “Guardar seleccionados” permite almacenar estos datos en un archivo de texto para documentación o análisis posterior.

Este resultado final demuestra la efectividad de la herramienta, al entregar una visión clara y estructurada de la exposición pública de servidores DNS. Al permitir la combinación de múltiples criterios de análisis, simbolización visual y exportación de datos, la interfaz no solo mejora la experiencia del usuario, sino que fortalece el propósito central de la herramienta: identificar vulnerabilidades en servidores DNS utilizando inteligencia automatizada y accesible.

Conclusión

Este proyecto fue desarrollado con una finalidad estrictamente educativa y defensiva. Su principal objetivo es brindar una herramienta práctica para auditar servidores DNS expuestos, permitiendo a estudiantes, investigadores y profesionales en ciberseguridad aprender sobre cómo identificar configuraciones inseguras o mal implementadas. A través del uso de la API de Shodan y pruebas específicas como la resolución de dominios, la verificación de recursividad y la detección de amplificación DNS, se busca fomentar el conocimiento sobre vectores de ataque comunes y cómo prevenirlos.

Es importante destacar que el script no debe utilizarse con fines ofensivos ni en redes o infraestructuras ajenas sin autorización expresa. El uso indebido de estas herramientas podría representar una violación a leyes locales o internacionales relacionadas con delitos informáticos. Esta herramienta está pensada únicamente para realizar auditorías éticas en entornos controlados o con permiso explícito, promoviendo siempre buenas prácticas y el fortalecimiento de la seguridad digital.

Los datos recopilados por la herramienta provienen de la API pública de Shodan, una plataforma reconocida para la búsqueda de dispositivos conectados a Internet. Toda la información que Shodan entrega está protegida por sus propios términos de servicio. Por lo tanto, es responsabilidad del usuario respetar sus políticas de uso, otorgar el crédito correspondiente a la fuente (Shodan.io), y utilizar los resultados de manera ética y legal. Shodan no tiene relación con el desarrollo de este proyecto y cualquier uso de sus datos es bajo la responsabilidad exclusiva del usuario.

Finalmente, el desarrollador de esta herramienta no se responsabiliza por los usos indebidos que se le puedan dar al código. El propósito de esta aplicación es formar y educar, nunca causar daño. Se alienta a la comunidad a utilizarla para mejorar la seguridad de sus propios sistemas y contribuir a una Internet más segura y responsable.