Subdomain Enumeration & DNS Lookup Tool

CODE :

```
import requests
import threading
import os
import dns.resolver
import tkinter as tk
from tkinter import scrolledtext, messagebox, filedialog, ttk

class SubdomainAndDNSLookupTool:
    def __init__(self, master):
        self.master = master
        master.title("Advanced Subdomain & DNS Tool")
        master.geometry("800x700")
        master.configure(bg='#f0f0f0')

        # Set window icon
        try:
            master.iconbitmap('icon.ico')
        except:
            pass
```

```
        # Custom style
        self.style = ttk.Style()
        self.style.configure('TFrame', background='#f0f0f0')
        self.style.configure('TButton', font=('Helvetica', 10), padding=6)
        self.style.configure('TLabel', background='#f0f0f0',
font=('Helvetica', 10))
        self.style.configure('Header.TLabel', font=('Helvetica', 12, 'bold'),
foreground='#2c3e50')

        # Main frame
        self.main_frame = ttk.Frame(master, padding="10")
        self.main_frame.pack(fill=tk.BOTH, expand=True)

        # Header
        self.header = ttk.Label(self.main_frame, text="SUBDOMAIN &
DNS ENUMERATOR", style='Header.TLabel')
        self.header.pack(pady=(0, 10))

        # Domain input frame
        self.input_frame = ttk.Frame(self.main_frame)
        self.input_frame.pack(fill=tk.X, pady=5)

        self.domain_label = ttk.Label(self.input_frame, text="Domain to
analyze:")
        self.domain_label.pack(side=tk.LEFT, padx=(0, 5))
```

```
        self.domain_entry = ttk.Entry(self.input_frame, width=40)
        self.domain_entry.pack(side=tk.LEFT, expand=True, fill=tk.X)

        # File selection frame
        self.file_frame = ttk.Frame(self.main_frame)
        self.file_frame.pack(fill=tk.X, pady=5)

        self.file_button = ttk.Button(self.file_frame, text="Select
Subdomain File", command=self.load_file)
        self.file_button.pack(side=tk.LEFT)

        self.file_label = ttk.Label(self.file_frame, text="No file selected",
foreground='#7f8c8d')
        self.file_label.pack(side=tk.LEFT, padx=10)

        # Button frame
        self.button_frame = ttk.Frame(self.main_frame)
        self.button_frame.pack(fill=tk.X, pady=10)

        self.enumerate_button = ttk.Button(self.button_frame, text="
Enumerate Subdomains",
                                           command=self.enumerate_subdomains)
        self.enumerate_button.pack(side=tk.LEFT, expand=True, padx=5)
```

```
        self.ns_button = ttk.Button(self.button_frame, text="🔍 DNS Records Lookup",
                                    command=self.dns_lookup)
        self.ns_button.pack(side=tk.LEFT, expand=True, padx=5)

        # Status bar
        self.status_var = tk.StringVar()
        self.status_var.set("Ready")
        self.status_bar = ttk.Label(self.main_frame, textvariable=self.status_var,
                                    relief=tk.SUNKEN, anchor=tk.W, foreground='#34495e')
        self.status_bar.pack(side=tk.BOTTOM, fill=tk.X, pady=(10, 0))

        # Results area
        self.result_frame = ttk.Frame(self.main_frame)
        self.result_frame.pack(fill=tk.BOTH, expand=True, pady=(10, 0))

        self.result_text = scrolledtext.ScrolledText(self.result_frame, width=90, height=30,
                                                     wrap=tk.WORD, font=('Consolas', 10),
                                                     padx=10, pady=10)
        self.result_text.pack(fill=tk.BOTH, expand=True)

        # Initialize variables
```

```
        self.filename = None
        self.discovered_subdomains = []
        self.lock = threading.Lock()

        # Set focus to domain entry
        self.domain_entry.focus()

    def load_file(self):
        self.filename = filedialog.askopenfilename(title="Select
Subdomain File",
                                                   filetypes=[("Text files", "*.txt")])
        if self.filename:
            self.file_label.config(text=os.path.basename(self.filename))
            self.status_var.set(f"Loaded:
{os.path.basename(self.filename)}")

    def check_subdomain(self, subdomain, domain):
        url = f'http://{subdomain}.{domain}'
        try:
            requests.get(url, timeout=5)
        except (requests.ConnectionError, requests.Timeout):
            pass
        else:
            with self.lock:
```

```
            self.discovered_subdomains.append(url)
            self.result_text.insert(tk.END, f"[+] Discovered: {url}\n")
            self.result_text.see(tk.END)
            self.status_var.set(f"Discovered: {url}")

    def enumerate_subdomains(self):
        domain = self.domain_entry.get().strip()
        if not domain:
            messagebox.showerror("Input Error", "Please enter a domain name.")
            return
        if not self.filename:
            messagebox.showerror("Input Error", "Please select a subdomain file.")
            return

        self.discovered_subdomains.clear()
        self.result_text.delete(1.0, tk.END)
        self.result_text.insert(tk.END, f"= Starting subdomain enumeration for {domain} =\n\n")
        try:
            with open(self.filename) as file:
                subdomains = file.read().splitlines()
        except Exception as e:
```

```
            messagebox.showerror("File Error", f"Failed to read
subdomain file:\n{str(e)}")
            return


        self.status_var.set("Enumerating subdomains...")
        self.enumerate_button.config(state=tk.DISABLED)


        threads = []
        for subdomain in subdomains:
            thread = threading.Thread(target=self.check_subdomain,
args=(subdomain, domain))
            thread.start()
            threads.append(thread)
# Wait for threads to complete in a separate thread to keep GUI
responsive
        def check_threads():
            alive = sum(1 for t in threads if t.is_alive())
            if alive > 0:
                self.status_var.set(f"Scanning... ({alive} threads remaining)")
                self.master.after(100, check_threads)
            else:
                count = len(self.discovered_subdomains)
                self.result_text.insert(tk.END, f"\n= Scan complete. Found
{count} subdomains =\n")
```

```
            self.result_text.insert(tk.END, "Results saved to
'discovered_subdomains.txt'\n")
            with open("discovered_subdomains.txt", 'w') as f:
                f.write('\n'.join(self.discovered_subdomains))
            self.status_var.set(f"Ready | Found {count} subdomains")
            self.enumerate_button.config(state=tk.NORMAL)

        self.master.after(100, check_threads)

    def dns_lookup(self):
        domain = self.domain_entry.get().strip()
        if not domain:
            messagebox.showerror("Input Error", "Please enter a domain
name.")
            return

        records_type = ['A', 'AAAA', 'NS', 'MX', 'TXT', 'CNAME', 'SOA']
        self.result_text.delete(1.0, tk.END)
        self.result_text.insert(tk.END, f"= DNS records for {domain}
=\n\n")
        self.status_var.set(f"Looking up DNS records for {domain}...")

        resolver = dns.resolver.Resolver()
        found_records = False
        for record_type in records_type:
```

```
            try:
                answer = resolver.resolve(domain, record_type)
                self.result_text.insert(tk.END, f"=== {record_type} Records ===\n")
                for data in answer:
                    self.result_text.insert(tk.END, f"{data}\n")
                self.result_text.insert(tk.END, "\n")
                found_records = True
            except (dns.resolver.NoAnswer, dns.resolver.NXDOMAIN):
                continue
            except Exception as e:
                self.result_text.insert(tk.END, f"Error retrieving {record_type} records: {str(e)}\n")

        if not found_records:
            self.result_text.insert(tk.END, "No DNS records found for the domain\n")

        self.result_text.insert(tk.END, "\n= DNS lookup complete =\n")
        self.status_var.set("DNS lookup completed")

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

Code Breakdown :

1. **Imports**:

Code :

```
import requests
import threading
import os
import dns.resolver
import tkinter as tk
from tkinter import scrolledtext, messagebox, filedialog, ttk
```

- **requests**: Used for making HTTP requests to check the existence of subdomains.
- **threading**: Allows concurrent execution of subdomain checks to improve performance.
- **os**: Provides a way to interact with the operating system, such as file handling.
- **dns.resolver**: Used for DNS record lookups.
- **tkinter**: The standard GUI toolkit for Python, used to create the application interface.
- **ttk**: Themed Tkinter widgets for a more modern look.

2. **Class Definition**:

Code : class SubdomainAndDNSLookupTool:

- This class encapsulates the entire functionality of the tool, including the GUI and the logic for subdomain enumeration and DNS lookups.

3. **Initialization**:

Code : def __init__(self, master):

- The constructor initializes the main window and its components.

4. **Window Configuration**:

Code :
```
master.title("Advanced Subdomain & DNS Tool")
master.geometry("800x700")
master.configure(bg='#f0f0f0')
```

- Sets the title, size, and background color of the main window.

5. **Icon Setup**:

code :
```
try:
    master.iconbitmap('icon.ico')
except:
    pass
```

- Attempts to set a custom icon for the window, handling any exceptions if the icon file is not found.

6. **Custom Styles**:

Code : self.style = ttk.Style()

self.style.configure('TFrame', background='#f0f0f0')

- Configures the appearance of various widgets using the **ttk.Style()** class.

7. **Main Frame and Header**:

Code : self.main_frame = ttk.Frame(master, padding="10")

self.main_frame.pack(fill=tk.BOTH, expand=True)

self.header = ttk.Label(self.main_frame, text="SUBDOMAIN & DNS ENUMERATOR", style='Header.TLabel')

self.header.pack(pady=(0, 10))

- Creates a main frame to hold all components and adds a header label.

8. **Domain Input**:

Code : self.domain_entry = ttk.Entry(self.input_frame, width=40)

- A text entry field for the user to input the domain they want to analyze.

9. **File Selection**:

Code : self.file_button = ttk.Button(self.file_frame, text="Select Subdomain File", command=self.load_file)

- A button that opens a file dialog to select a text file containing subdomains.

10. **Enumerate Subdomains Button**:

Code : self.enumerate_button = ttk.Button(self.button_frame, text="🔍 Enumerate Subdomains", command=self.enumerate_subdomains)

- A button that triggers the subdomain enumeration process.

11. **Status Bar**:

Code : self.status_var = tk.StringVar()

self.status_bar = ttk.Label(self.main_frame, textvariable=self.status_var, relief=tk.SUNKEN, anchor=tk.W, foreground='#34495e')

- Displays the current status of the application, such as "Ready" or "Enumerating subdomains...".

12. **Results Area**:

Code : self.result_text = scrolledtext.ScrolledText(self.result_frame, width=90, height=30, wrap=tk.WORD, font=('Consolas', 10))

- A scrollable text area to display the results of the subdomain enumeration and DNS lookups.

13. **Load File Method**:

Code : def load_file(self):

self.filename = filedialog.askopenfilename(title="Select Subdomain File", filetypes=[("Text files", "*.txt")])

- Opens a file dialog to select a subdomain file and updates the status bar.

14. **Check Subdomain Method**:

Code : def check_subdomain(self, subdomain, domain):

url = f'http://{subdomain}.{domain}'

requests.get(url, timeout=5)

- Constructs the full URL for each subdomain and checks if it is reachable. If reachable, it adds the URL to the results.

15. **Enumerate Subdomains Method**:

Code : def enumerate_subdomains(self):

domain = self.domain_entry.get().strip()

- Retrieves the domain from the input field, reads subdomains from the selected file, and starts checking each subdomain in a separate thread.

16. **DNS Lookup Method**:

Code : def dns_lookup(self):

domain = self.domain_entry.get().strip()
resolver = dns.resolver.Resolver()

- Retrieves DNS records for the specified domain using the **dns.resolver** library and displays the results.

17. **Main Loop**:

Code :

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

- Initializes the Tkinter main loop, creating an instance of the application.

Features

- **Subdomain Enumeration**: The tool allows users to input a domain and a file containing potential subdomains. It checks each subdomain concurrently, displaying discovered subdomains in real-time.
- **DNS Lookup**: Users can perform DNS lookups for various record types (A, AAAA, NS, MX, TXT, CNAME, SOA) and view the results in the application.
- **User -Friendly GUI**: The application features a clean and modern interface with input fields, buttons, and a scrollable text area for results.
- **File Handling**: Users can easily select a file containing subdomains, and the application handles file reading and error checking.
- **Threading**: The use of threading allows the application to remain responsive while performing network operations.
- **Status Updates**: The status bar provides real-time feedback on the application's current state, enhancing user experience.

1. Overview

- The tool performs two core functions:
  - **Subdomain Enumeration:** Discovers live subdomains by testing combinations from a wordlist.
  - **DNS Lookup:** Retrieves DNS records (A, AAAA, MX, TXT, etc.) for domain analysis.

➢ Built with **tkinter** for GUI and **dnspython/requests** for network operations, it features:

✓ Modern responsive interface
✓ Threaded scanning (faster results)
✓ Real-time progress tracking
✓ Auto-saving results
✓ Error handling & user feedback

2. Core Components Breakdown

2.1 Graphical User Interface (GUI)

The interface uses **ttk** (Themed Tkinter) for a polished look:

Main Sections:

Component	Function
Domain Entry	Input field for target domain (e.g., example.com)
File Selector	Button to load subdomain wordlist (.txt file)
Action Buttons	Start subdomain scan or DNS lookup
Results Box	Large scrollable output area (width=90, height=30)
Status Bar	Live updates during operations (e.g., "Scanning...")

2.2 Key Features

I. Subdomain Enumeration

1. **Process:**
    - Reads subdomains from a wordlist (e.g., **subdomains.txt**).
    - Tests each via HTTP requests (**http://subdomain.example.com**).
    - Threaded for parallel scanning (faster results).
2. **Output Example:**

= Starting subdomain enumeration for example.com =

[+] Discovered: http://www.example.com

[+] Discovered: http://mail.example.com

= Scan complete. Found 2 subdomains =

II. DNS Lookup

1. **Records Checked:**
   A, **AAAA**, **NS**, **MX**, **TXT**, **CNAME**, **SOA**.
2. **Output Example:**

= DNS records for example.com =

=== A Records ===

192.0.2.1

=== MX Records ===

10 mail.example.com

3. Technical Implementation

3.1 Critical Functions

Function	Description
load_file()	Opens file dialog to select subdomain wordlist
check_subdomain()	Tests if a subdomain exists (threaded)
enumerate_subdomains()	Manages scanning process and UI updates
dns_lookup()	Queries DNS records and displays results

3.2 Threading Mechanism

- Prevents GUI freezing during scans.
- Uses Python's **threading** module:

for subdomain in subdomains:

thread = threading.Thread(target=self.check_subdomain, args=(subdomain, domain))

4. Error Handling

- The tool includes robust checks:

1. **Input Validation:**
    - Ensures domain is entered before scanning.
    - Validates wordlist file selection.
2. **Network Errors:**
    - Catches connection timeouts/DNS failures gracefully.
    - Shows user-friendly messages via **messagebox.showerror()**.

5. Output & Reporting

- **Auto-Save:** Results saved to **discovered_subdomains.txt**.
- **Real-Time Updates:** Status bar shows progress (e.g., *"Discovered: http://test.example.com"*).

Conclusion

This tool provides an **elegant, functional solution** for:

- **Security researchers** performing reconnaissance.
- **Network admins** troubleshooting DNS issues.
- **Developers** testing domain configurations.

The combination of **automated scanning**, **clean GUI**, and **detailed reporting** makes it ideal for both technical and non-technical users.

Appendix A: Sample Usage

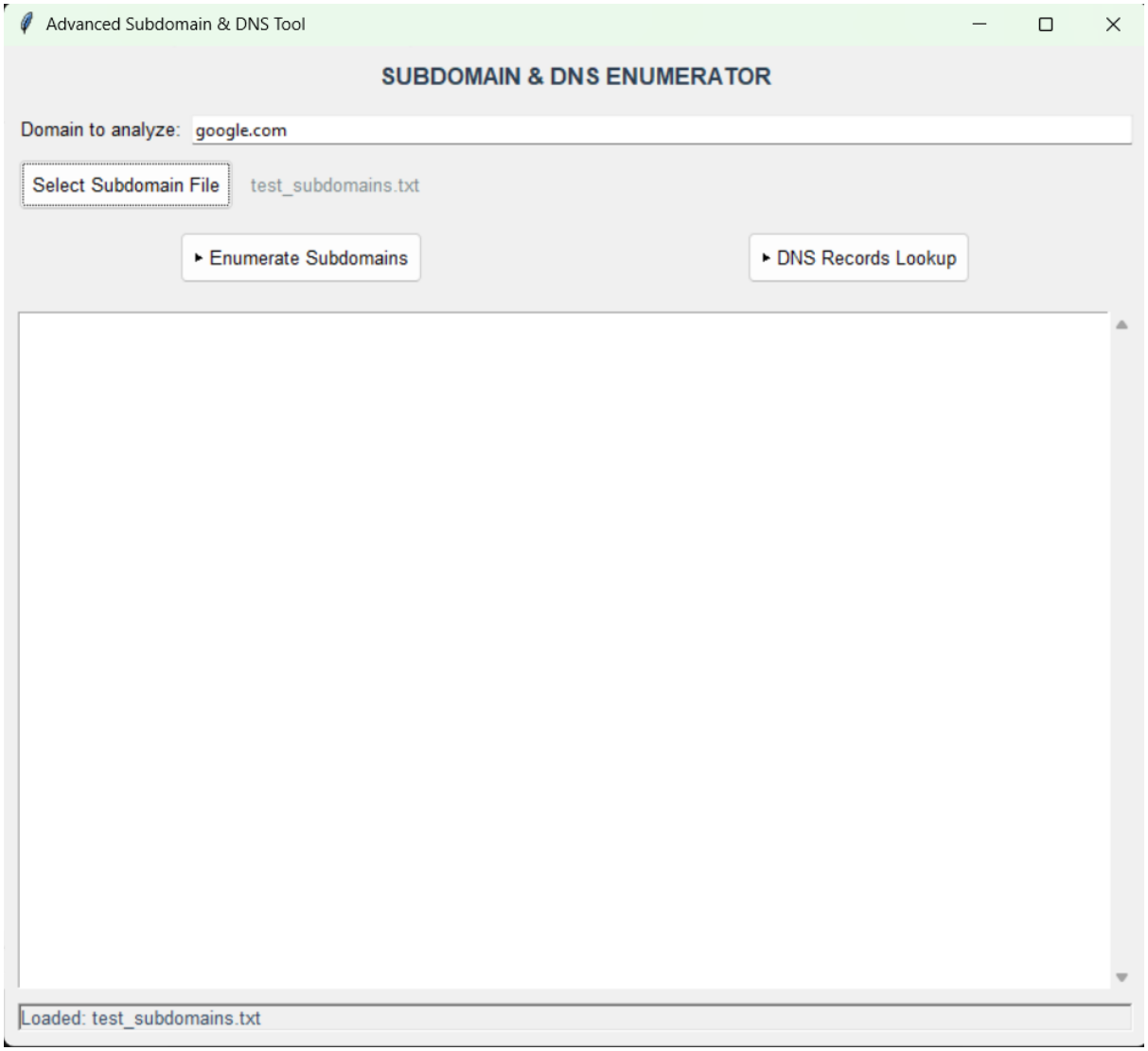
1. Enter domain (e.g., **example.com**).
2. Click **"Select Subdomain File"** to load **subdomains.txt**.
3. Click **"Enumerate Subdomains"** or **"DNS Lookup"**.
4. View results in the output box and saved files.

Appendix B: Dependencies

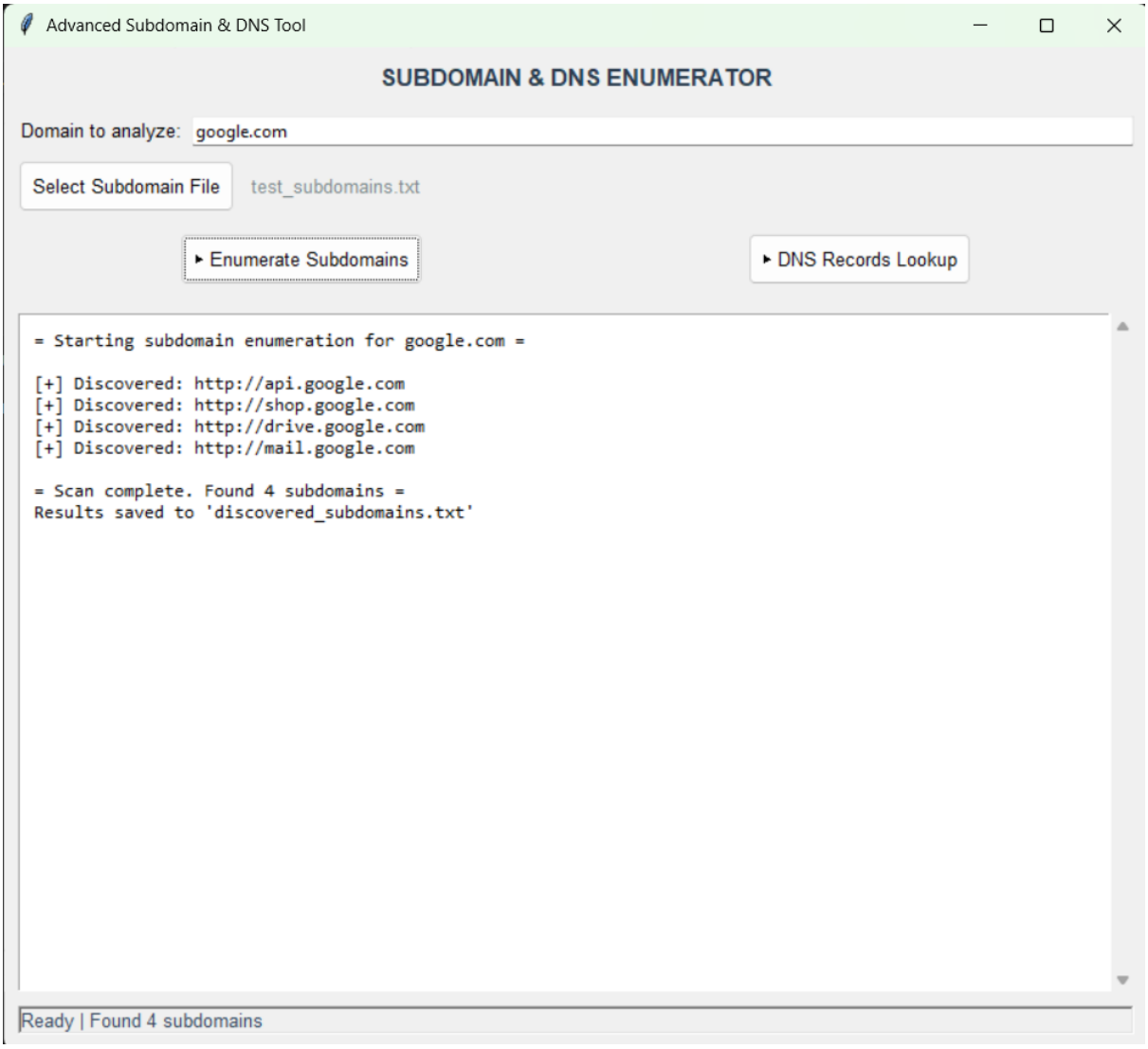
- pip install dnspython requests
- pip install threads

Screenshots :

1. entered google.com as domain to analyze and list of subdomain file



2. this image shows output of subdomain enumeration



3. this image shows output of DNS records enumeration