Subdomain Enumeration Using Python and Threading

This project focuses on building a simple and efficient subdomain enumeration tool using Python. It uses multithreading to scan multiple subdomains simultaneously, helping to identify active hosts within a target domain. The tool automates an important step of cybersecurity reconnaissance, demonstrating Python's capability in automation and networking tasks.

Objective

To develop a Python script that automatically detects active subdomains for a given domain using multithreading and HTTP requests, improving speed and efficiency compared to manual enumeration.

Introduction

Subdomain enumeration is the process of identifying valid subdomains associated with a main domain. This helps ethical hackers and security professionals uncover hidden or forgotten services that may pose vulnerabilities.
In this project, Python's requests, threading, and queue

modules are used to send concurrent HTTP requests and record responsive subdomains.

Working Process

1. The script loads a list of subdomains from subdomains.txt.
2. A thread-safe queue stores subdomains to be checked.
3. Multiple threads run simultaneously to check subdomains.
4. Each subdomain is combined with the main domain and tested using an HTTP request.
5. If a valid response is received, the subdomain is marked as active and saved.
6. Thread locks prevent multiple threads from writing to the output file at the same time.
7. Once all subdomains are processed, the tool displays total active subdomains and time taken.

Tools and Libraries

Tool	Purpose
Python 3	Programming language
requests	Send HTTP requests
threading	Manage concurrent execution
queue	Handle tasks between threads
time	Measure execution duration

Code Explanation

- **load_subdomains() → Reads subdomains and loads them into a queue.**
- **check_subdomain() → Sends HTTP request to test if subdomain is reachable.**
- **save_discovered_subdomain() → Writes active subdomains to file safely using a thread lock.**
- **worker() → Thread function that processes subdomains concurrently.**
- **main() → Handles user input, creates threads, and prints final results. main() → Handles user input, creates threads, and prints final results.**

Sample Input & Output

Input:

```
PS D:\inlingx\subdomain_enum> py subdomain_enum.py
============================================================
Subdomain Enumeration Tool
============================================================
Enter the target domain (e.g., example.com): hi.com
```

Output:

```
Threads: 10
Timeout: 3s
============================================================
[+] Loaded 257 subdomains to check

[*] Starting enumeration with 10 threads...

[+] Found: email.hi.com [Status: 404]
[+] Found: static.hi.com [Status: 403]
[+] Found: web.hi.com [Status: 200]
[+] Found: secure.hi.com [Status: 404]
[+] Found: api.hi.com [Status: 404]
[+] Found: www.hi.com [Status: 200]
[+] Found: cms.hi.com [Status: 403]
[+] Found: resources.hi.com [Status: 403]
[+] Found: auth.hi.com [Status: 404]
[+] Found: shop.hi.com [Status: 200]
[+] Found: sandbox.hi.com [Status: 200]
[+] Found: help.hi.com [Status: 200]

============================================================
Enumeration Complete!
============================================================
Total subdomains checked: 257
Active subdomains found: 12
Time elapsed: 11.21 seconds
Results saved to: discovered_subdomains.txt
============================================================
```

Results

The script efficiently identified active subdomains using concurrent threads. The use of a thread-safe queue and locks ensured smooth file operations. Execution time was significantly reduced compared to single-threaded enumeration.

Advantages

- Fast and lightweight
- Simple and easy to modify
- Uses minimal resources
- Saves results automatically

Limitations

- May miss subdomains hosted on non-HTTP ports
- Can produce false positives on wildcard DNS
- Network latency affects results

Conclusion

- The project successfully demonstrates subdomain enumeration using Python and threading. It showcases how multithreading enhances speed in network reconnaissance, providing practical insight into automation in cybersecurity.