Subdomain Enumeration Using Python and Threading

Abstract

In the field of cybersecurity, subdomain enumeration plays a vital role in identifying potential points of vulnerability within a target domain. This project presents a Python-based tool designed to automate the process of subdomain discovery by reading a predefined list of common subdomain names, appending them to a user-defined target domain, and sending HTTP requests to check their availability.

To improve the efficiency and speed of enumeration, the tool leverages **multithreading**, allowing concurrent checks of multiple subdomains. Active (live) subdomains are identified based on HTTP response status codes and are logged into an output file for further analysis. The implementation is lightweight, platform-independent, and serves as a foundational tool in reconnaissance during penetration testing and ethical hacking practices.

This project demonstrates how scripting and automation can enhance information gathering tasks in cybersecurity, making it an ideal starting point for aspiring security professionals.

Introduction

Subdomain enumeration is a crucial technique in cybersecurity, primarily used during the reconnaissance phase of penetration testing and ethical hacking. It involves discovering valid and active subdomains linked to a main domain, which could potentially expose hidden or vulnerable services running on a network.

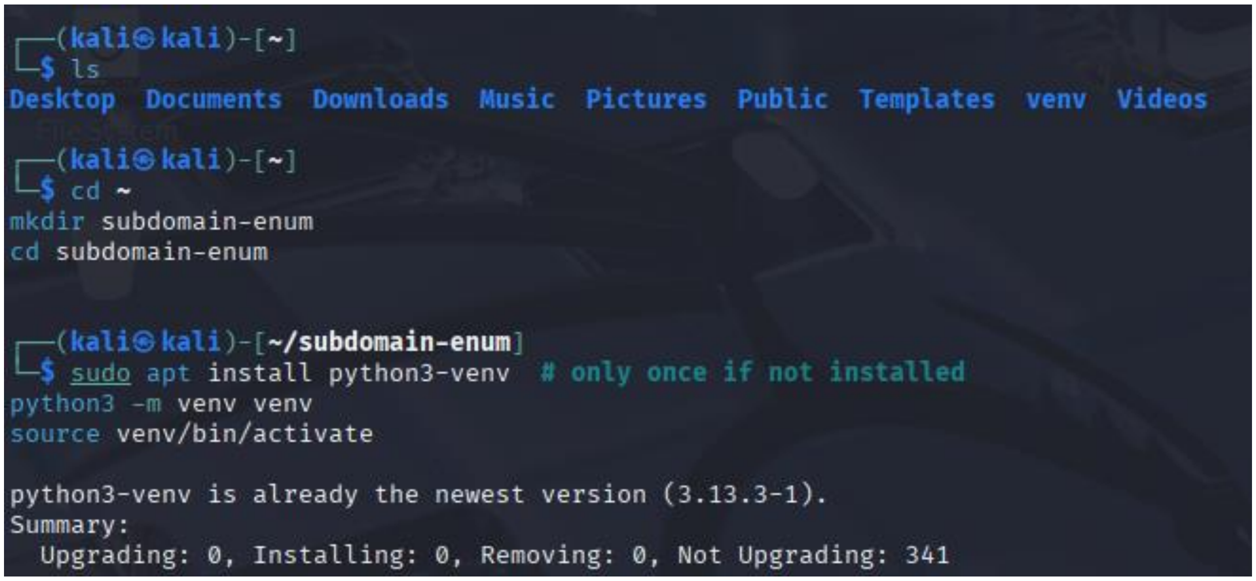
This project focuses on building a Python-based subdomain enumeration tool that automates this process. It uses a list of common subdomain prefixes, combines them with a target domain, and checks their availability by sending HTTP requests. To optimize performance and reduce scanning time, the tool implements **multithreading**, allowing multiple subdomains to be checked simultaneously.

The result is a lightweight and effective tool suitable for both beginners and professionals looking to enhance their reconnaissance capabilities in real-world security assessments.

Step –by- Step process:

Step 1: Setting Up the Environment

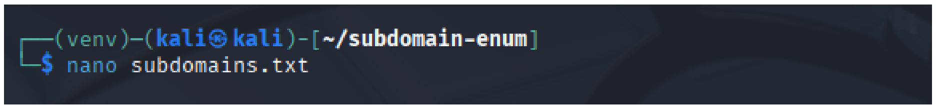
- A project folder named subdomain-enum is created to organize all files.
- A Python virtual environment is initialized.
- The required module requests are installed inside the virtual environment.



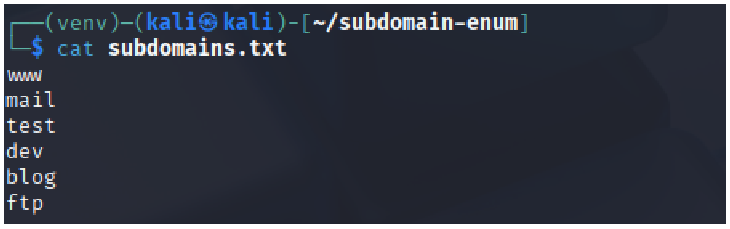
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  venv  Videos

┌──(kali㉿kali)-[~]
└─$ cd ~
mkdir subdomain-enum
cd subdomain-enum

┌──(kali㉿kali)-[~/subdomain-enum]
└─$ sudo apt install python3-venv  # only once if not installed
python3 -m venv venv
source venv/bin/activate

python3-venv is already the newest version (3.13.3-1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 341
```

Step 2: Preparing the Subdomain Wordlist

- A text file named subdomains.txt is created.
- It contains a list of commonly used subdomain names such as:



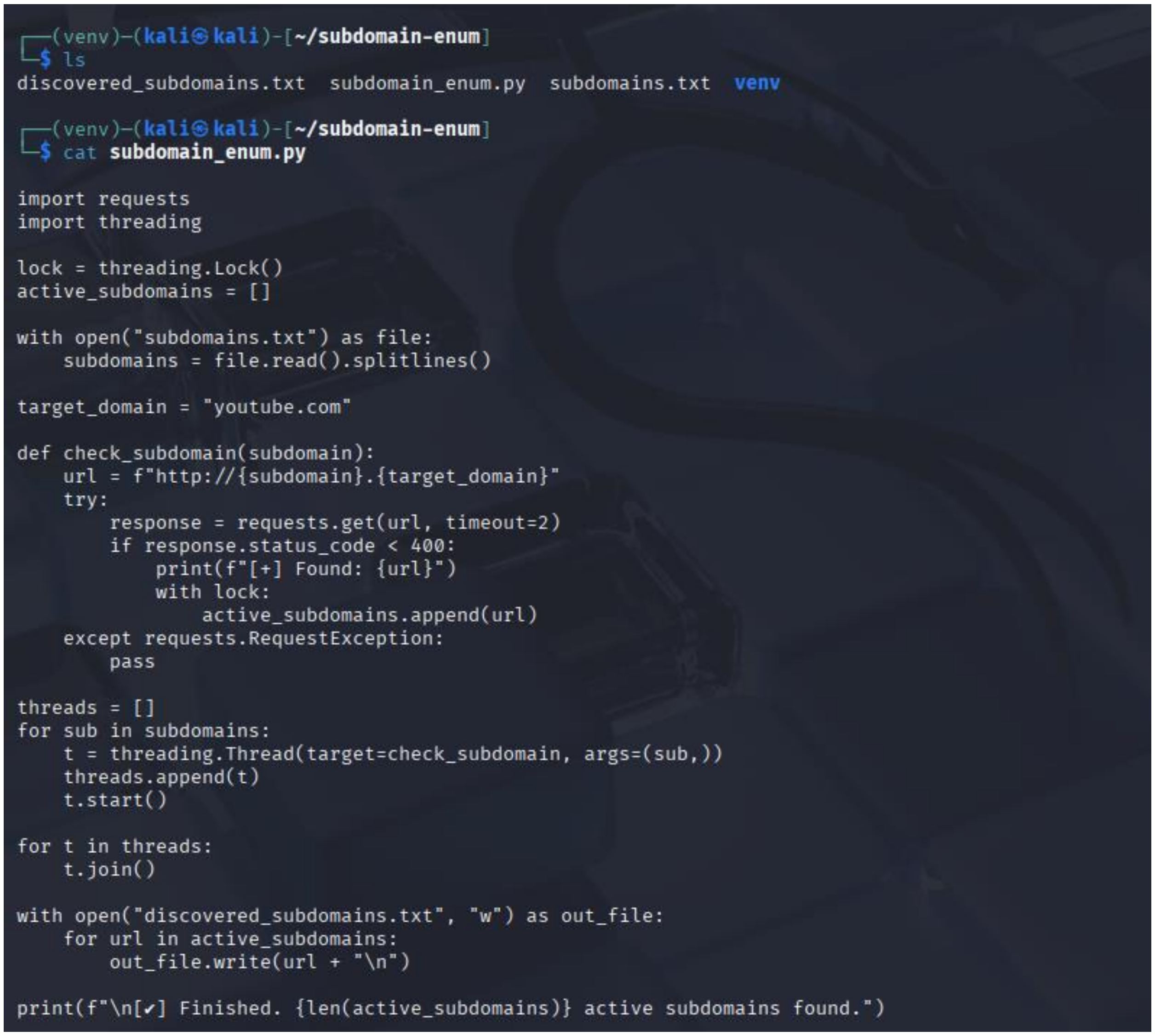
```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ nano subdomains.txt
```



```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ cat subdomains.txt
www
mail
test
dev
blog
ftp
```

Step 3: Writing the Python Script

- A Python script file named subdomain_enum.py is created.
- The script:
  - Reads subdomains from the wordlist
  - Appends each to a target domain
  - Sends HTTP requests to check if the subdomain is active
  - Uses multithreading to improve speed
  - Stores the live subdomains in discovered_subdomains.txt



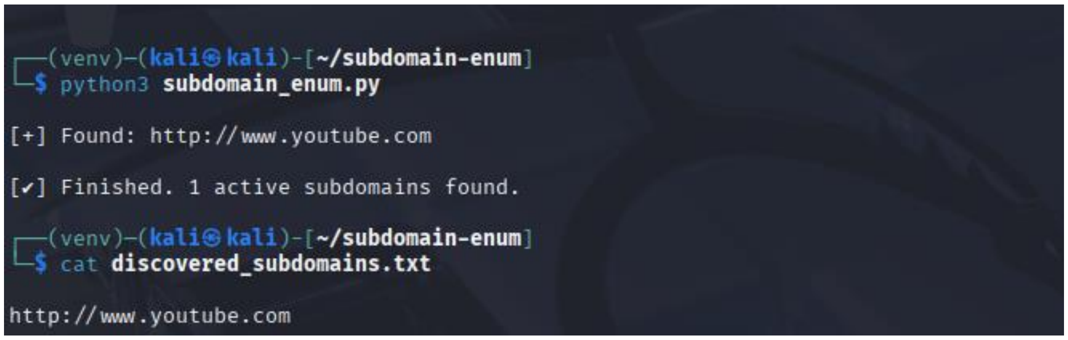
```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ ls
discovered_subdomains.txt  subdomain_enum.py  subdomains.txt  venv

┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ cat subdomain_enum.py

import requests
import threading

lock = threading.Lock()
active_subdomains = []

with open("subdomains.txt") as file:
    subdomains = file.read().splitlines()

target_domain = "youtube.com"

def check_subdomain(subdomain):
    url = f"http://{subdomain}.{target_domain}"
    try:
        response = requests.get(url, timeout=2)
        if response.status_code < 400:
            print(f"[+] Found: {url}")
            with lock:
                active_subdomains.append(url)
    except requests.RequestException:
        pass

threads = []
for sub in subdomains:
    t = threading.Thread(target=check_subdomain, args=(sub,))
    threads.append(t)
    t.start()

for t in threads:
    t.join()

with open("discovered_subdomains.txt", "w") as out_file:
    for url in active_subdomains:
        out_file.write(url + "\n")

print(f"\n[✓] Finished. {len(active_subdomains)} active subdomains found.")
```

If subdomains are found to be active (respond with status code < 400), they are:

- Printed on the terminal
- Written into an output file discovered_subdomains.txt

Step 4: Running the Script

- The script is executed using



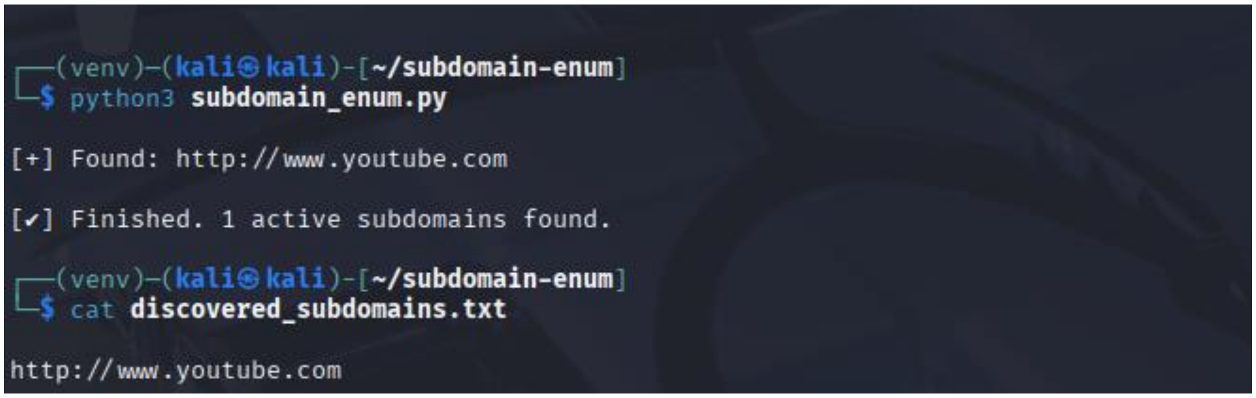
```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ python3 subdomain_enum.py

[+] Found: http://www.youtube.com

[✔] Finished. 1 active subdomains found.

┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ cat discovered_subdomains.txt

http://www.youtube.com
```

If subdomains are found to be active (respond with status code < 400), they are:

- Printed on the terminal
- Written into an output file discovered_subdomains.txt

Step 5: Output Verification

- The discovered subdomains can be viewed with

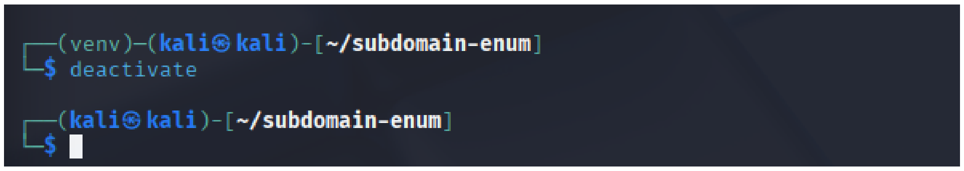


```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ python3 subdomain_enum.py

[+] Found: http://www.youtube.com

[✔] Finished. 1 active subdomains found.

┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ cat discovered_subdomains.txt

http://www.youtube.com
```

This file contains only the **active subdomains**, which are valuable for security analysis.

Step 6: Cleanup and Exit

- The virtual environment is deactivated using



```
┌──(venv)─(kali㉿kali)-[~/subdomain-enum]
└─$ deactivate

┌──(kali㉿kali)-[~/subdomain-enum]
└─$ 
```

Conclusion

The step-by-step implementation of the subdomain enumeration project provides a hands-on understanding of how automated reconnaissance tools are developed using Python. By setting up a virtual environment, creating a wordlist, writing a multithreaded Python script, and executing the enumeration process, this project demonstrates the practical workflow of discovering live subdomains associated with a target domain.

Each step—from environment setup to result generation—plays a critical role in ensuring the tool is efficient, modular, and suitable for real-world penetration testing. This structured process helps in building foundational skills in scripting, networking, and ethical hacking, making it a valuable learning experience for cybersecurity enthusiasts.