IT SOLERA

WEB RECON TOOL

Presented by: Alpha Team

Team Alpha

Research Team

- Muhammad Mughees
- Roshany R M
- Khizer Nadeem

Development Team

- Mohamed Ahm ed
- Saad Hassan
- Abdullah RIAZ
- Abdifatah Abdullahi Mohamed

Reporting Team

- Noshair Wan
- Safiullah shahid
- Muhammad Ahmad

INTRODUCTION

In today's complex digital landscape, efficient penetration testing is crucial. Our team has developed a Python scripting toolkit that empowers you to streamline this process. We'll delve into tools for directory and file brute-forcing, subdomain enumeration, web crawling, and active subdomain discovery. See how these tools can automate tasks, enhance efficiency, and uncover hidden vulnerabilities. Let's explore how this toolkit can empower your security assessments!

RESEARCH TEAM

Research team has conducted an extensive evaluation of various cybersecurity tools, focusing on their unique features, capabilities, and integration possibilities. This work aims to provide a comprehensive understanding of these tools to enhance our cybersecurit tools understanding while development of out tool.

Existing Tools

- WhatWeb
- Recon-ng
- theHarvester
- SpiderFoot
- Amass
- Sublist3r
- Photon
- Wfuzz
- dirbuster
- ZenBuster
- Dnsenum
- Aquatone

DEVELOPMENT TEAM

Development team has created multiple Python scripts designed to assist in Web Information Gathering during penetration testing (ethical hacking) activities. These scripts leverage the power and flexibility of Python to automate various reconnaissance and vulnerability assessment tasks.

Modules Developed

- Directory Brute-Forcing
- File Brute-Forcing
- Subdomain Enumeration Tool
- Basic Web Crawler
- Active Subdomains Enumeration Using DNS Resolvers

REPORTING TEAM

The reporting team has created a comprehensive set of documentation providing a thorough understanding of the Python scripting toolkit, guiding penetration testers through its capabilities and promoting its effective usage designed for penetration testing:

- Proposal Report
- Tools Explanation Report:
- Presentation

PROJECT OVERVIEW

Objective

The main goal of the Web Information Gathering Tool is to automate the process of collecting various types of information about a web application to support security assessments and identify potential vulnerabilities.

Scope

The scope of the Web Information Gathering Tool project is comprehensive, focusing on several key functionalities to support security assessments and identify potential vulnerabilities in web applications. Here is a refined scope section for the project:

FEATURES:

Subdomains Enumeration:
Utilizing both active DNS resolvers and passive methods such as Google search to identify subdomains.

Directories and Files Brute Forcing:
Implementing brute force techniques to discover hidden directories and files on the target web server.

Crawler:
Developing a web crawler to systematically browse and index the website's pages, identifying any potential security issues.

SCRIPTS

Crawler:

- The program initiates a web crawl from a specified starting URL.
- It traverses web pages up to a defined depth, collecting all visited URLs.
- It uses the requests library to fetch page content and BeautifulSoup to parse HTML and extract links.
- A ThreadPoolExecutor is employed to handle concurrent crawling of multiple pages.
- The crawler includes a delay between requests to avoid overloading the server.
- Finally, it saves the collected URLs to a JSON file.

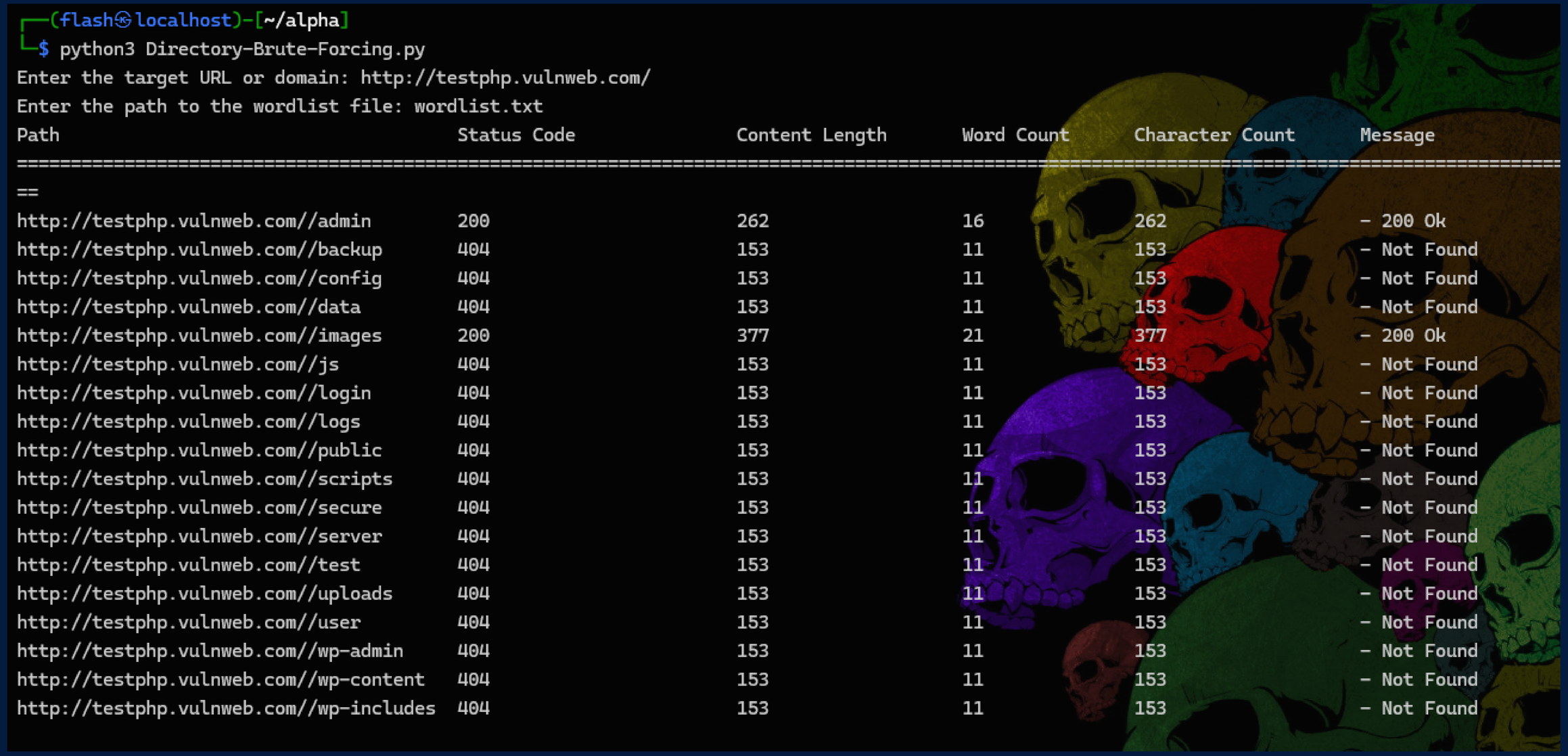
Output:

```
┌──(flash㉿localhost)-[~/alpha]
└─$ python3 crawler.py
Crawling: https://www.nu.edu.pk/
Crawling: https://www.nu.edu.pk/Admissions/Scholarship
Crawling: https://www.nu.edu.pk/Admissions/HowToApply
Crawling: https://www.nu.edu.pk/Degree-Programs
Crawling: https://www.nu.edu.pk/Admissions/Schedule
Crawling: https://www.nu.edu.pk/Admissions/TestPattern
Crawling: http://isb.nu.edu.pk
Crawling: http://isb.nu.edu.pk/Faculty/allfaculty
Crawling: http://isb.nu.edu.pk/Media/NewsList
Crawling: https://www.nu.edu.pk/Admissions/EligibilityCriteria
Crawling: http://cfd.nu.edu.pk/all-departments
Crawling: http://cfd.nu.edu.pk
Crawling: http://pwr.nu.edu.pk/cs-faculty/
Crawling: http://pwr.nu.edu.pk/latest-news
Crawling: https://www.nu.edu.pk/Admissions/FeeStructure
Crawling: http://pwr.nu.edu.pk
Crawling: http://www.sites.ieee.org/sb-nuces-karachi/
Crawling: http://khi.nu.edu.pk/events.php
Crawling: https://www.nu.edu.pk/Campus/Chiniot-Faisalabad/Events
Crawling: http://www.nuces-acm.org/
Crawling: https://www.nu.edu.pk/Campus/Peshawar/PhdSupervisors
Crawling: https://www.nu.edu.pk/campus/missionvision/Electrical Engineering
```

Directory Brute Forcing

- The script performs directory enumeration on a given target URL using paths from a wordlist file.
- It validates the provided URL, ensuring it is complete and accessible.
- It checks the existence of the wordlist file specified by the user.
- For each path in the wordlist, the script constructs the full URL and sends a GET request to it.
- It prints detailed information about each request, including status code, content length, word count, and character count.
- The results are saved into files categorized by status codes within a 'responses' directory.

Output:

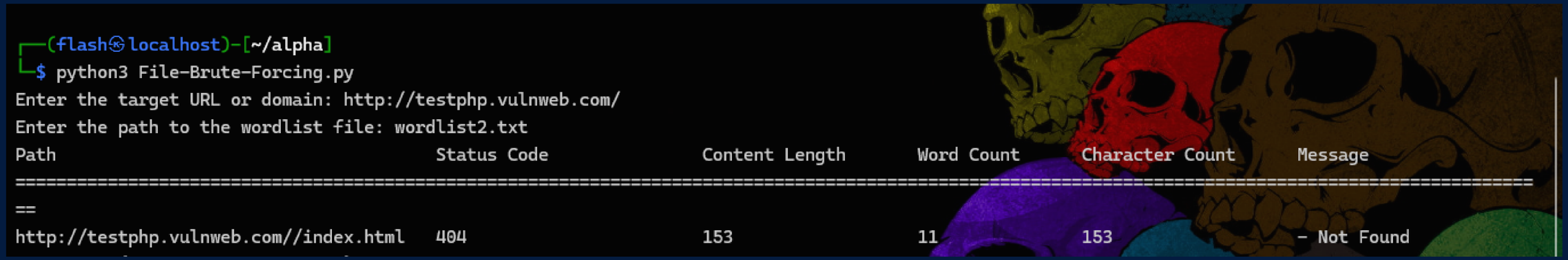


```
┌──(flash㉿localhost)-[~/alpha]
└─$ python3 Directory-Brute-Forcing.py
Enter the target URL or domain: http://testphp.vulnweb.com/
Enter the path to the wordlist file: wordlist.txt
Path                                    Status Code             Content Length      Word Count      Character Count     Message
==========================================================================================================================
==
http://testphp.vulnweb.com//admin       200                     262                 16              262                 - 200 Ok
http://testphp.vulnweb.com//backup      404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//config      404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//data        404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//images      200                     377                 21              377                 - 200 Ok
http://testphp.vulnweb.com//js          404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//login       404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//logs        404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//public      404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//scripts     404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//secure      404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//server      404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//test        404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//uploads     404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//user        404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//wp-admin    404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//wp-content  404                     153                 11              153                 - Not Found
http://testphp.vulnweb.com//wp-includes 404                     153                 11              153                 - Not Found
```

File Brute Forcing

- **Input Handling:** Asks the user for a target URL and a wordlist file path.
- **Validation:** Checks if the URL is valid and reachable. Also verifies if the wordlist file exists.
- **Enumeration:** Iterates through each path in the wordlist, constructs full URLs by combining them with the target URL, and sends HTTP GET requests to each URL.
- **Response Handling:** Prints details about each HTTP response including status code, content length, word count, and character count. Saves URLs to files categorized by their HTTP status codes.
- **Error Handling:** Manages exceptions during URL validation and HTTP requests, displaying error messages when issues occur.

Output:



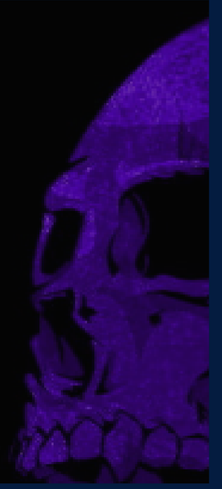
```
┌──(flash㉿localhost)-[~/alpha]
└─$ python3 File-Brute-Forcing.py
Enter the target URL or domain: http://testphp.vulnweb.com/
Enter the path to the wordlist file: wordlist2.txt
Path                                    Status Code             Content Length      Word Count      Character Count     Message
==========================================================================================================================
==
http://testphp.vulnweb.com//index.html  404                     153                 11              153                 - Not Found
```

Passive Subdomain Enumeration

- **Input Handling:** Takes a domain input from the user, sanitizes it for formatting by removing unnecessary prefixes (like "http://" or "https://") and trailing slashes.
- **Multi-threaded Subdomain Enumeration:** Uses crt.sh API to query subdomains associated with the domain. Distributes the task across 5 threads to concurrently fetch and process subdomain data.
- **Data Extraction:** Each thread retrieves 100 subdomain entries from crt.sh API JSON responses. Filters and collects subdomains ending with ".domain".
- **Thread Management:** Manages thread execution using Python's threading module, ensuring synchronization of thread completion with thread.join().
- **Output:** Prints discovered subdomains in alphabetical order if any are found. Alerts if no subdomains are identified for the specified domain.

Output:

```
┌──(flash㉿localhost)-[~/alpha]
└─$ python3 passive_subdomains_emumerator.py
Enter the root domain (e.g., example.com): myguestlist.com
Sending request to URL: https://crt.sh/?q=%25.myguestlist.com&output=json - Page 0
Sending request to URL: https://crt.sh/?q=%25.myguestlist.com&output=json - Page 100
Sending request to URL: https://crt.sh/?q=%25.myguestlist.com&output=json - Page 200
Sending request to URL: https://crt.sh/?q=%25.myguestlist.com&output=json - Page 300
Sending request to URL: https://crt.sh/?q=%25.myguestlist.com&output=json - Page 400
```

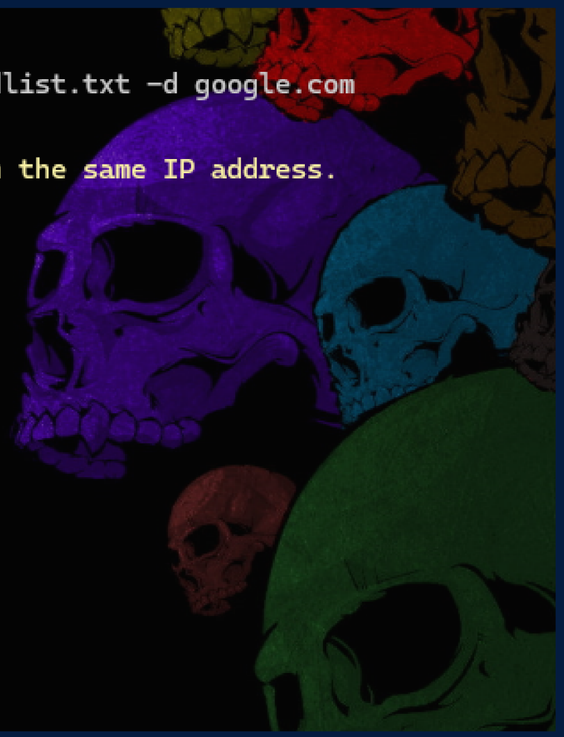


Active Subdomain Enumeration

- **Initialization**: Reads user-provided configurations like file paths, validates them, and sets up trusted DNS resolvers.
- **DNS Resolver Validation (Optional):** Checks if provided resolvers return the same IP as trusted resolvers for a baseline domain and optionally validates NXDOMAIN responses for known domains.
- **Subdomain Enumeration:** Reads domains/subdomains from a file, uses threads to perform DNS lookups with the chosen resolvers, and saves valid subdomains with resolved IP addresses.
- **Output:** Prints the number of valid subdomains and saves them to a user-specified file.

Output:

```
┌──(flash㉿localhost)-[~/alpha/ASEUDR]
└─$ python3 aseudr.py nameservers.txt --enable_dns_validator --mode brute-force -w wordlist.txt -d google.com

[*] Starting verification: checking if the DNS resolver and trusted DNS resolver return the same IP address.

[-] 199.255.137.34 raised Timeout.

[-] 82.146.26.2 raised Timeout.

[+] 8.8.8.8 is valid.

[+] 208.67.222.222 is valid.

[-] 103.129.221.226 raised Timeout.

[-] 46.105.55.84 raised Timeout.
```



Limitations

- The scope is restricted to information gathering and does not include vulnerability exploitation.
- Reliance on publicly available data and tools which may not always yield exhaustive results.
- The efficiency of brute forcing is dependent on the comprehensiveness of the word lists used.

Conclusion

In closing, this presentation has showcased the capabilities of our Python scripting toolkit designed to streamline penetration testing workflows. We've explored a range of modeules, including directory and file brute-forcing, subdomain enumeration, basic web crawling, and active subdomain discovery.

By leveraging these tools, penetration testers can:

- Automate Repetitive Tasks
- Enhance Efficiency
- Uncover Hidden Vulnerabilities

Thank You