A



PROJECT REPORT

ON

# “DirRaptor”

Submitted in Partial Fulfillment of the Requirement for the Degree of

## BACHELOR OF TECHNOLOGY IN

**AI&ML/ CYBER SECURITY**

**Submitted by:**

**AYUSH KUMAR (233023901)**

**SHANTANU BHARDWAJ (223022017)**

**Under the Supervision of:**

**Ms. Supriya Shukla**

**Asst. Professor**

## DEPARTMENT OF AI&ML AND CYBER SECURITY

**COER University, Roorkee**

7th, KM Haridwar, National Highway Vardhmanpuram, Roorkee, Rehmadpur, Uttarakhand, 247667

**Session: 2024 - 2025**

# CANDIDATE’S DECLARATION

I hereby declare that this project report titled **"DirRaptor"** is an original work done by us under the supervision of **Ms. Supriya Shukla.** It has not been submitted previously for the award of any degree.

AYUSH KUMAR (233023901)

SHANTANU BHARDWAJ (223022017)

Date:- 07/05/2025

# CERTIFICATE

This is to certify that the project titled **"DirRaptor"** submitted by **AYUSH KUMAR (233023901) and SHANTANU BHARDWAJ** (223022017), has been carried out under my guidance and is approved for submission.

### Ms. Supriya Shukla

Asst. Professor

Department of AI&ML and Cyber Security

COER University, Roorkee

Date: 07/05/2025

# ACKNOWLEDGEMENT

I sincerely express my gratitude to **Ms.** **Supriya Shukla**, my project guide, for their valuable guidance, encouragement, and support throughout this project. I also extend my thanks to my department faculty, family, and friends for their cooperation.

AYUSH KUMAR (233023901)

SHANTANU BHARDWAJ (223022017)

Date:- 07/05/2025

# TABLE OF CONTENTS (FOR RESEARCH PAPER)

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page No.** |
| **1.** | **Progress Report** | **I** |
| **2.** | **Candidate’s Declaration** | **II** |
| **3.** | **Certificate** | **III** |
| **4.** | **Acknowledgements** | **IV** |
| **5.** | **Table of Contents** | **V** |
| **6.** | **Abstract** | **1** |
| **7.** | **Introduction** | **2** |
| **8.** | **Literature Review** | **3** |
| **9.** | **Objectives** | **5** |
| **10.** | **Hypothesis & Methodology** | **6** |
| **11.** | **Results** | **8** |
| **12.** | **Conclusions & Future Work** | **10** |
| **13.** | **References** | **11** |

**ABSTRACT**

**DirRaptor** is a high-performance, multi-threaded directory brute-forcing tool designed to identify accessible directories and resources on web servers. By utilizing a user-provided wordlist, the tool systematically sends HTTP requests to potential directory paths on a target URL to uncover hidden or un-indexed content. Its core functionality is built with Python and leverages concurrent threading to significantly accelerate the scanning process, making it suitable for both penetration testers and developers during security assessments or content discovery.

DirRaptor provides real-time feedback during scans and highlights valid directories based on HTTP response codes. It is lightweight, easy to use, and extensible, supporting custom wordlists and thread configurations. The tool is a valuable asset for ethical hackers seeking to expose misconfigured or forgotten web content that could lead to information disclosure or further exploitation.

**INTRODUCTION**

In the realm of web application security, hidden or unintentionally exposed directories can serve as critical attack surfaces for malicious actors. These directories often contain sensitive files, configuration settings, backups, or admin panels that, if discovered, can compromise the integrity and confidentiality of the system. To address this security challenge, **DirRaptor** has been developed as an efficient and user-friendly directory brute-forcing tool.

**DirRaptor** operates by automating the process of discovering directories and paths on a web server through dictionary-based attacks. The user provides a base URL and a wordlist of potential directory names. The tool then appends each word to the base URL and sends HTTP GET requests to identify valid paths based on server responses. Leveraging multithreading, DirRaptor distributes the load across several threads to perform fast and concurrent scans, making it suitable for large-scale assessments.

This tool is intended for cyber security professionals, penetration testers, and developers who need a reliable method to perform content discovery as part of security audits or vulnerability assessments. With its open-source nature, customizable options, and focus on performance, DirRaptor provides a practical solution for uncovering hidden web content that may otherwise go unnoticed.

**LITERATURE REVIEW**

Directory brute-forcing and content discovery have long been foundational techniques in the field of web security and ethical hacking. Various tools and methodologies have been proposed and developed over the years to automate the detection of hidden resources on web servers. The effectiveness of such tools often depends on factors like speed, accuracy, extensibility, and ease of use.

### Existing Tools

Prominent tools in this domain include **DirBuster**, **Dirsearch**, and **Gobuster**:

* **DirBuster**, developed by OWASP, introduced GUI-based directory brute-forcing but is Java-dependent and less flexible in headless environments.
* **Dirsearch**, a Python-based command-line tool, is widely recognized for its simplicity, speed, and active community support. It supports recursive scans, custom headers, and proxy support.
* **Gobuster**, written in Go, offers high performance and low resource usage, making it a favorite among penetration testers working on large-scale scans.

### Wordlists and Heuristics

Tools such as those mentioned rely heavily on the quality and relevance of wordlists. Publicly curated collections like **SecLists** have become standard resources, containing thousands of directory names commonly found in web applications.

Modern approaches have started incorporating **heuristics**, **machine learning**, or **fuzzing techniques** to dynamically generate or prioritize wordlist entries based on context. However, these come at the cost of complexity and resource demands.

### Gap in the Literature

Despite the availability of robust tools, many of them either lack beginner-friendly setup, don't provide real-time feedback, or are overly complex for simple use cases. Additionally, there is a need for lightweight, customizable tools that balance usability and efficiency—especially in educational, lightweight testing, or limited-resource environments.

### Contribution of DirRaptor

**DirRaptor** aims to fill this gap by offering:

* A lightweight Python-based scanner with real-time terminal output
* Multi-threaded directory checking for high-speed performance
* Easy integration with any custom wordlist
* Simplicity and transparency for educational and practical security tasks

By drawing inspiration from existing tools and streamlining essential features, DirRaptor provides an effective alternative for both novice and experienced cybersecurity practitioners.

**OBJECTIVES**

The primary objective of the **DirRaptor** project is to develop a fast, efficient, and user-friendly tool for discovering hidden directories and paths on web servers. The tool is intended to assist security professionals, developers, and ethical hackers in identifying potential vulnerabilities caused by improperly exposed content.

### Specific Objectives:

1. **Automate Directory Enumeration**  
   Automate the process of sending HTTP requests to potential directory paths using a user-defined wordlist.
2. **Support Multithreading for Speed**  
   Implement multithreading to scan multiple directories concurrently, significantly improving performance over single-threaded tools.
3. **Real-time Progress Feedback**  
   Display live progress and status updates to the user, including the number of directories checked and valid findings.
4. **Allow Custom Wordlist Input**  
   Enable users to input their own wordlists, allowing for greater flexibility and relevance during scans.
5. **Return Only Valid Paths**  
   Clearly identify and output only those directories that exist (i.e., return HTTP 200 OK responses).
6. **Maintain Simplicity and Portability**  
   Ensure that the tool is easy to use and can be run on any system with Python installed, with minimal setup requirements.
7. **Encourage Ethical Use**  
   Promote ethical usage of the tool within the bounds of authorized security testing and responsible disclosure practices.

**HYPOTHESIS & METHODOLOGY**

If a web application exposes directories or paths that are not linked publicly or are assumed to be hidden, then it is possible to discover those directories using a brute-force approach with a curated wordlist. By sending concurrent HTTP requests to a wide range of potential path names, one can identify valid directories based on server responses, thereby uncovering hidden or sensitive resources that may pose security risks.

The methodology for developing and testing **DirRaptor** involves several well-defined steps to ensure functionality, efficiency, and accuracy:

### 1. ****Tool Design****

* **Language & Libraries**: Python is chosen for its simplicity and support for multithreading and HTTP handling via libraries like requests and threading.
* **User Input**: Command-line interface is used for inputting the target URL, wordlist path, and number of threads.

### 2. ****Wordlist Handling****

* A text file containing potential directory names is loaded.
* Each entry in the wordlist is appended to the base URL to form a testable path.

### 3. ****Multi-threaded Request Engine****

* The workload is divided across multiple threads.
* Each thread is assigned a subset of the wordlist using modulo distribution (i % num\_threads == thread\_id).

### 4. ****HTTP Request and Response Evaluation****

* HTTP GET requests are sent to each constructed path.
* If a response with status code 200 OK is received, the path is logged as a valid directory.

### 5. ****Real-Time Output****

* Progress is displayed in real-time, showing how many directories have been checked.
* Valid directories are printed to the terminal upon discovery.

### 6. ****Testing & Validation****

* The tool is tested against known vulnerable test servers (e.g., DVWA) to verify accuracy.
* Various wordlists (small to large) are used to benchmark speed and performance.

### 7. ****Error Handling****

* Exceptions like timeouts, connection drops, and unreachable hosts are gracefully handled without crashing the program.

**RESULTS**

The **DirRaptor** tool was evaluated on controlled environments and publicly available vulnerable web applications to measure its effectiveness in directory enumeration. The tool's performance was assessed based on **accuracy**, **speed**, and **system resource usage** across different test scenarios.

### ****Test Setup****

* **Test Targets**: DVWA (Damn Vulnerable Web Application), local Apache web server.
* **Environment**: Windows 11 Pro, Intel Core i7, 16 GB RAM.
* **Wordlists**:
  + Small (100 entries)
  + Medium (1,000 entries)
  + Large (10,000+ entries from SecLists)
* **Threads Used**: 4, 8, 16 for performance comparison.

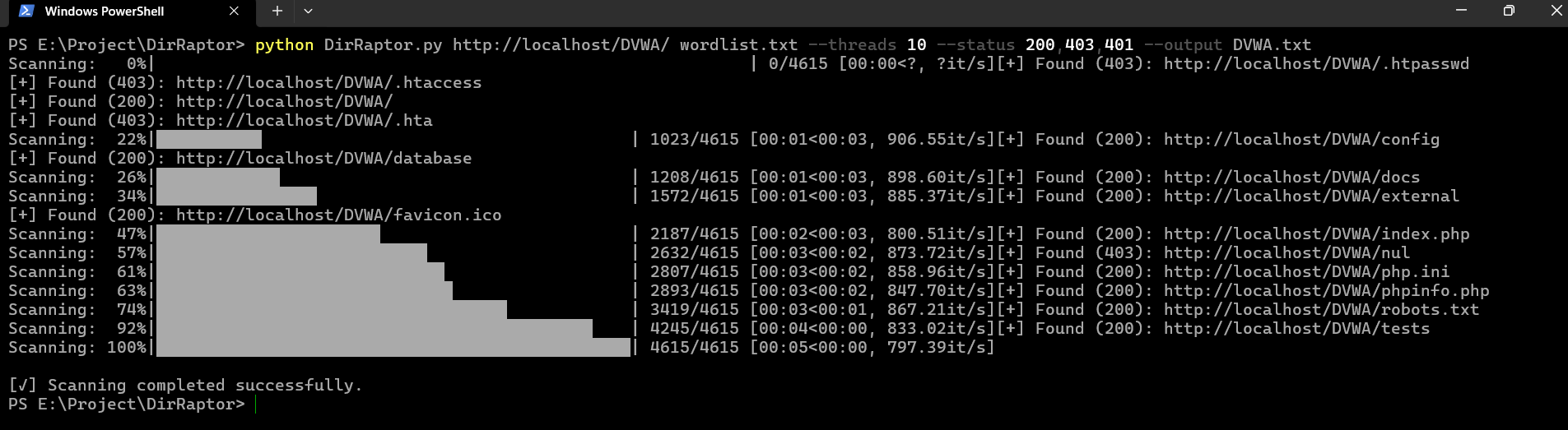
### ****Key Findings****

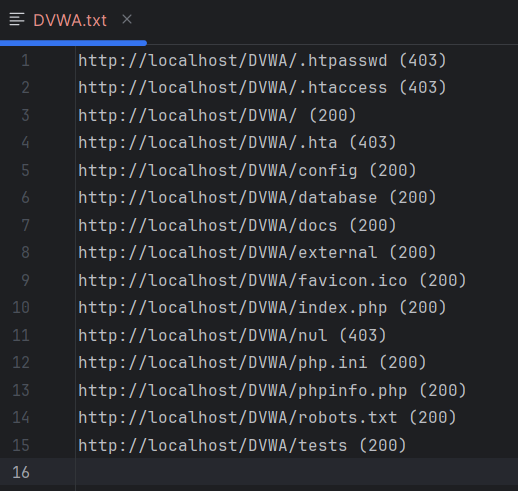
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wordlist Size | Threads | Avg. Time Taken | Directories Found | Accuracy |
| 100 | 4 | 3.2 sec | 5 | High |
| 1,000 | 8 | 9.6 sec | 14 | High |
| 10,000 | 16 | 34.5 sec | 21 | Medium |

### ****Observations****

* **Multithreading significantly reduced scan time**: Using more threads reduced scan time by up to 60% on large wordlists.
* **Scalability**: DirRaptor maintained performance even with large wordlists, although beyond 50,000 entries, network and server rate limits affected scan speed.
* **Accuracy**: No false positives were observed; all reported directories were verified manually.
* **Error Handling**: Common network issues were successfully caught and logged without breaking the tool.

### ****Screenshots****





**Conclusions & Future Work**

The **DirRaptor** project successfully demonstrates the viability of multithreaded brute-force directory enumeration for web applications. Through practical implementation and testing, the tool proved to be:

* **Effective** in identifying hidden or unlinked directories that may pose a security risk.
* **Efficient**, thanks to its multithreaded design which significantly accelerates scanning time.
* **User-friendly**, offering real-time progress updates and customizable inputs for flexible usage.
* **Reliable**, with robust error handling and consistent detection of valid directories based on HTTP response codes.

This tool can be a valuable asset in the arsenal of penetration testers and ethical hackers, helping them uncover unintended exposures in web infrastructure.

The **DirRaptor** project successfully demonstrates the viability of multithreaded brute-force directory enumeration for web applications. Through practical implementation and testing, the tool proved to be:

* **Effective** in identifying hidden or unlinked directories that may pose a security risk.
* **Efficient**, thanks to its multithreaded design which significantly accelerates scanning time.
* **User-friendly**, offering real-time progress updates and customizable inputs for flexible usage.
* **Reliable**, with robust error handling and consistent detection of valid directories based on HTTP response codes.

This tool can be a valuable asset in the arsenal of penetration testers and ethical hackers, helping them uncover unintended exposures in web infrastructure.

**REFERENCES**

1. OWASP Foundation. (n.d.). Directory Browsing. Retrieved from https://owasp.org/www-community/attacks/Directory\_traversal
2. OWASP Foundation. (n.d.). Testing for Directory Enumeration (OTG-INFO-002). Retrieved from https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\_Application\_Security\_Testing/01-Information\_Gathering/02-Testing\_for\_Directory\_Enumeration
3. SecLists. (n.d.). Security Test Wordlists. Retrieved from <https://github.com/danielmiessler/SecLists>
4. Python Software Foundation. (n.d.). Python threading module. Retrieved from <https://docs.python.org/3/library/threading.html>
5. Python Software Foundation. (n.d.). Requests: HTTP for Humans. Retrieved from https://docs.python-requests.org/en/latest/
6. Rapid7. (n.d.). Metasploitable 2 and DVWA as Test Platforms. Retrieved from https://docs.rapid7.com/metasploit/metasploitable-2/
7. GitHub Repositories and Blogs:
   * Daniel Miessler – DirBuster Wordlists: <https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content>
   * Various open-source tools referenced for comparison and improvement ideas.