

GUI-BASED CYBERSECURITY UTILITY TOOLS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology

by

SANJAIKANTH J (40120078)

SURIYA S (40120096)



DEPARTMENT OF INFORMATION TECHNOLOGY

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– 600119, TAMILNADU

APRIL- 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by
UGC | Approved by AICTE www.sathyabama.ac.in

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SANJAIKANTH.J (Reg. No. 40120078) & SURIYA S (Reg. No. 40120096)** who carried out the project as a "**GUI-BASED CYBERSECURITY UTILITY TOOLS**" under our supervision from November 2023 to April 2024.


Internal Guide

Dr. C. GEETHA, M.Tech., Ph.D.


Head of the Department

Dr. R. Subhashini, M.E., Ph.D.

Submitted for Viva-voce Examination held on

18 / 04 / 24


Internal Examiner


External Examiner

DECLARATION

I SURIYA S (Reg. No.40120096) hereby declare that the Project Report entitled "GUI-BASED CYBERSECURITY UTILITY TOOLS" is done by me under the guidance of Dr.C.GEETHA, M.Tech., Ph.D., and is submitted in partial fulfillment of the requirements for the award of **Bachelor of Technology degree in Information Technology.**

DATE: 12/04/24
PLACE: Chennai



SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **Sathyabama Institute of Science and Technology** 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. R Subhashini, M.E., Ph.D.**, Head of the Department of Information Technology for providing me with 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 Project Guide **Dr. C. GEETHA, M.Tech., Ph.D.** for her valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

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

ABSTRACT

Cybersecurity is the technology and process for safeguarding networks and devices against attacks, damage, or unauthorized access. Everyone who is in the field of cybersecurity has been at a phase where they have been confused about the endless list of tools that are used in this field. From Nmap to Burp suite. People who want to enter this field get confused at the endless list of tools and resources and thus end up using high-level tools for a long time before figuring them out or just giving up halfway. The challenge presented in this scenario is that for a given objective such as scanning a web application, the user needs to learn and use more than three or four tools, which are difficult to integrate. Thus, the need for a general-purpose tool that has everything built inside it comes up so that professionals and new entrants don't have to search through an endless list of tools to find the good and useful ones. The objective of this research is to build a fully integrated tool that can be used by users, without having to download and learn all the different tools available on the market. The research work proposed a similar idea in the form of a general-purpose tool based on Python that caters to everyone in this field. This tool includes custom scanners to crawlers and malware. The proposed work envisions an all-purpose tool that is both flexible and customizable so that it does not only cater to new users in the field but also includes tools for professionals for a quick lookout.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vi
	LIST OF ABBREVIATIONS	vii
1	INTRODUCTION	1
	1.1 AIM	1
	1.2 PROBLEM STATEMENT	1
	1.3 SCOPE	1
2	LITERATURE SURVEY	3
3	SYSTEM ANALYSIS	7
	3.1 EXISTING SYSTEM	7
	3.2 PROPOSED SYSTEM	8
	3.3 REQUIREMENTS SPECIALIZATIONS	9
	3.3.1 Hardware requirements	9
	3.3.2 Software Requirements	10
	3.3.3 Development Environment	10
	3.3.4 User Environment	11
4	SYSTEM ARCHITECTURE	12
	4.1 ARCHITECTURE	12
	4.2 GRAPHICAL USER INTERFACE	13
	4.2.1 Elements of GUI	13
	4.2.2 How Does GUI work	14
	4.2.3 Benefits of GUI	14
	4.3 CYBERSECURITY TOOLS	15
	4.4 TYPES OF CYBERSECURITY TOOLS	16
	4.4.1 Network Security Monitoring Tools	15
	4.4.2 Encryption Tools	16
	4.4.3 Web Vulnerability Scanning Tools	16
	4.4.4 Penetration Testing Tools	16
	4.4.5 Packet Sniffers	17
	4.5 KIVY FRAMEWORK	17
	4.5.1 Kivy Architecture	17

5		MODULE DESCRIPTION	21
	5.1	TOOL DESIGNED	23
6		CONCLUSION	26
		FUTURE WORKS	27
		REFERENCE	28
		APPENDIX 1	29
		APPENDIX 2	39

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	Architecture	12
4.2	GUI	14
4.3	Cybersecurity Tools	15
4.4	Kivy Architecture	16
5.1	Block Diagram	23
5.2	Encryption/Decryption Tool	24
5.3	GUI-Output	25

LIST OF ABBREVIATIONS

SET	Social Engineering Toolkit
GUI	Graphical User Interface
IDE	Integrated Development Environment
NUI	Natural User Interface
TUI	Tangible User Interface
MDA	Message Digest Algorithm
MSF	Microsoft Solutions Framework

CHAPTER 1

INTRODUCTION

In an era where our daily lives and critical infrastructure are increasingly reliant on digital systems, the significance of cybersecurity cannot be overstated. The continuous evolution of cyber threats demands a proactive approach to safeguarding our digital assets and information. To address this pressing need, our project introduces a Graphical User Interface (GUI)-based Cybersecurity Tools, a comprehensive resource designed to empower users with accessible and user-friendly tools for enhancing their cybersecurity posture. In this project report, we will delve into the development, functionality, and significance of our GUI-based Cybersecurity utility Tools. We will explore the critical role it plays in helping individuals and organizations combat a wide spectrum of cyber threats, ranging from malware and phishing attacks to network vulnerabilities and data breaches. Moreover, we will discuss the underlying technology stack, user interface design principles, and the key features that make our database a valuable asset in the field of cybersecurity. Cybersecurity is a major issue in the connected world of today, when individuals and businesses are constantly at risk from criminals. To effectively counter these evolving threats, cybersecurity experts employ a wide range of specialized hardware and software. But if you're not an expert or are new to the field, it can be challenging to navigate the complex world of cybersecurity technologies. This abstract introduces a Cybersecurity Tools Database with a Graphical User Interface (GUI). It aims to increase usability and accessibility while giving customers access to a wide range of cutting-edge cybersecurity solutions.

2.1. Aim

The aim of this project is to build a GUI which contains cybersecurity utility tools that are customizable and flexible for the users.

2.2. Problem Statement

The project aims to develop a user-friendly graphical user interface (GUI) for a suite of cybersecurity utility tools to address the growing need for simplified and accessible cybersecurity solutions. The current challenge is that existing cybersecurity tools often lack an intuitive interface, making them difficult for non-experts to use effectively. This project seeks to bridge that gap by creating a comprehensive and user-friendly platform that provides essential cybersecurity functionalities such as vulnerability scanning, malware analysis, network monitoring, and threat detection through an intuitive GUI. The goal is to empower users, including individuals and small businesses, to enhance their cybersecurity posture without requiring extensive technical expertise.

2.3. Scope

The main purpose of this project is to provide a user-friendly graphical interface for easy navigation and interaction. Include a variety of cybersecurity utilities such as malware scanning, network monitoring, vulnerability assessment, and more. Allow users to customize and configure the tool to suit their specific cybersecurity needs. Support integration with third-party cybersecurity APIs and tools.

LITERATURE SURVEY

A. Analysis of Kali Linux Penetration Tools: A Survey of Hacking Tools

This research work analyses eighteen distinct tools that are supported on Kali Linux. These tools are used by developers and security professionals to harden the security of their applications. These eighteen tools are grouped into six groups, namely creation of payload, anti-virus evasion tool, network mapping, analysis of the vulnerability, digital forensics tool, and information gathering. The researchers further explain the functionalities of each tool in detail. These tools include some well-known tools such as social engineering toolkit (SET) and Nmap to specific tools such as Masscan, MSFVenom, etc. After explaining the functionalities, the tools are analyzed further for their said functionalities and compared to other tools in the same group. All the tools analyzed in this research are free and thus acts as a double edge sword to businesses and organizations who use them.

B. A Survey: How Python Pitches in IT-World

This research work focuses on surveying the various aspects of Python as a programming language. The researchers start with a discussion on how Python was first proposed in the late 1980s, and how it was implemented in the year 1989. They continue by discussing the various data structures, and libraries, and how since the advent of big data, it has emerged as a language with multiple paradigms. The discussion continues to sentiment analysis using python, which is essentially a method to analyze the emotions of customers expressed through various sentiments. These feelings can take on positive, negative, or neutral forms. Python's standard library includes a massive collection of python syntax, tokens, and semantics, which number more than two hundred in just the core modules, these include libraries such

as matplotlib, numpy, pandas, etc. They have also compared python to other languages such as Matlab and R. they have argued about how Matlab focuses on linear operations and R is a programming language that is used for interactive analysis of data as well as statistical packages.

C. Vulnerabilities and Security of Web Applications

This research work focuses on reviewing trends and prevention methods Of unauthorized attacks on web applications. They discuss how to prevent vulnerabilities by providing appropriate data types for inputs and preventing users from accessing unnecessary files on the server. They have also discussed the security of web applications in specific fields such as academia and e-commerce. The discussion about overall security precautions for operating systems and mobile applications, as well as how to promote total application and data security, continues. The discussion with regards to academia states how the data in academic fields continue to grow in volume and variety and how those large databases in the web servers should have authorized access. The discussion on the e-commerce field provides insight into the many policies that are set to provide security to the organization. The researchers analyze and discuss the previously mentioned aspects of security and propose that the government should concentrate on developing rigorous laws and regulations to defend against bad actors. Large enterprises should be held accountable for ensuring web security by finding vulnerabilities before an event occurs. Another proposal is to use OOPS based language such as Java to develop web applications that deal with sensitive data such as passwords. Although complete web security cannot be achieved the policymakers can make sure that the web is secure enough to win the trust of the users. The research concluded on a note that web security is the prime requirement for digitization.

D. Practical web security testing: Evolution of web application modules and open-source testing tools

This research work here states that the aspect of testing for web application security is essential for preventing certain flaws in the security system with respect to the web application design. One of the most difficult challenges in web security testing is the constant change and evolution of web design tools and modules. As a result, most tools that are used for open source may be outdated in terms of keeping up with new technologies. In this paper, the author reported the effort and experience of testing their recently developed website, and the authors used and reported vulnerabilities from several open-source security testing tools. They have also reported debugging efforts in order to resolve these security issues that have arisen during the development process, as the authors have used already present tools to test their websites. The various tools used by them are sqlmap, Nmap, OWASP ZAP, Nikto, etc. This research work again talks about the various web application security aspects but does not cover topics such as network security or malware tools. This research gave the inspiration for using multiple tools under a single roof but the change that we have made is instead of using already available in-built tools we are opting for custom developed tools.

E. A Systematic Review of Encryption and Keylogging for Computer System Security

This research work discusses the AES encryption algorithm and keylogging and how both intervene in real life. Encryption is the process of converting plainly readable text or data to an unreadable form called cipher text. AES or Advanced Encryption Standard is one of the symmetric encryption algorithms that is widely used, i.e. it uses a single key for encryption and decryption. AES is one of the encryption algorithms used in Internet communication. Keylogging generally refers to the function of a program called a keylogger or keystroke logger. It is a program that records the keystrokes without the affected user knowing. This can be used for

various functions such as recording passwords, bank details, etc. Even this seemingly vengeful software has its own uses in the ethical space of technology. Keyloggers are generally used by organizations to monitor their employees or used by law enforcers to investigate cyber-crimes. They conclude with ways that AES and Keyloggers can be used to protect data and privacy. Although this research provides a very in-depth analysis of keylogging and AES encryption, it is limited to a few algorithms and doesn't discuss various other algorithms. This Research work was used as a base to build the keylogging tool that is integrated into the proposed tool.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the existing system, users primarily interact with cybersecurity tools through conventional methods such as command-line interfaces or standalone applications. These tools are often scattered across various directories on the system, requiring users to navigate through file systems manually to locate and execute them. This method can be time-consuming and prone to errors, especially for users who are not familiar with command-line interfaces or the specific directory structure of the tools.

Moreover, the lack of a centralized graphical user interface (GUI) means that users must rely on memorizing or documenting the locations of different tools, which can lead to inefficiencies and frustration. Without a clear organizational structure or categorization, finding the right tool for a specific task becomes a challenging and cumbersome process.

Additionally, the absence of a GUI limits the accessibility of cybersecurity tools, as some users may prefer visual interfaces or may not be comfortable using command-line interfaces. This can create barriers to adoption for users who are not proficient in using command-line tools or who prefer a more intuitive interface for interacting with software.

Overall, the existing system's reliance on command-line interfaces and scattered tool directories contributes to inefficiencies, usability challenges, and accessibility issues for users seeking to leverage cybersecurity tools for their needs.

3.2 PROPOSED SYSTEM

Existing System Shortcomings:

Lack of centralized access to cybersecurity tools. Dependency on command-line interfaces or standalone applications. Tedious navigation through directories to locate specific tools.

Proposed System Features:

GUI with Kivy: Introduce a graphical user interface using Kivy framework to provide a more user-friendly experience.

Cybersecurity Utility Tools Heading: Display a heading "Cybersecurity Utility Tools" at the top of the GUI to clearly indicate the purpose.

Categorized Tool Buttons: Organize tools into four categories (e.g., "Network Security," "Encryption," "Vulnerability Scanning," "Forensics").

Clickable Tool Buttons: Each tool is represented as a button that users can click to access the corresponding tool.

Store Tool Paths: Store the paths of the tools' directories within the Kivy application to enable seamless access and execution.

Advantages of the Proposed System:

Ease of Access: Users can quickly access cybersecurity tools through a centralized GUI without the need for command-line navigation.

Improved User Experience: The GUI provides a more intuitive and visually appealing interface compared to traditional command-line interfaces.

Efficient Navigation: Categorized tool buttons streamline the process of finding and launching specific tools.

Customization: The ability to add or remove tools and categories can be easily incorporated into the GUI, providing flexibility and scalability.

Implementation Plan:

Design GUI Layout: Create a layout using Kivy that includes the heading and categorized tool buttons.

Define Button Actions: Implement functionality for each button to execute the corresponding tool.

Store Tool Paths: Use Kivy's storage capabilities to store the paths of tool directories within the application.

Testing and Refinement: Conduct thorough testing to ensure the GUI functions correctly and refine as needed based on user feedback. By implementing the proposed system, users will have a more efficient and user-friendly way to access and utilize cybersecurity tools, ultimately enhancing their cybersecurity practices

3.3 .REQUIREMENT SPECIFICATION

3.3.1.Hardware Requirements:

Processor: Intel Core i5 or equivalent AMD processor (or higher).

RAM: 4 GB minimum (8 GB recommended).

Storage: At least 100 MB of free disk space for the application and tool directories.

Display: Minimum resolution of 1280x720 pixels (or higher) for optimal GUI display.

Input Devices: Keyboard and mouse (or touchpad) for interacting with the GUI.

3.3.2 Software Requirements:

Operating System: Windows 10, macOS, or Linux (Ubuntu, CentOS, etc.).

Python: Python 3.6 or later installed on the system.

Kivy Framework: Version 2.0.0 or later for developing the graphical user interface.

Python Packages:

kivy: Required for building the GUI components.

os: For interacting with the operating system and handling file paths.

Any additional packages required by the specific cybersecurity tools being integrated into the application.

IDE/Text Editor: Any preferred code editor such as Visual Studio Code, PyCharm, or Sublime Text for writing and editing Python scripts.

Version Control: Git for version control (optional but recommended for project management and collaboration).

Dependency Management: pip or Anaconda for managing Python packages and dependencies.

Documentation: Sphinx or similar tool for generating documentation (optional but recommended for project documentation).

Testing Framework: Pytest or unit test for writing and executing unit tests (optional but recommended for ensuring code reliability).

3.3.3 Development Environment:

Development Machine: A computer or laptop with the hardware and software specifications mentioned above.

Internet Connection: Required for downloading libraries, packages, and documentation during development.

Code Repository: Git repository (e.g., GitHub, GitLab) for version control and collaboration with team members (if applicable).

Project Management Tools: Trello, Asana, or similar tools for managing tasks, milestones, and deadlines (optional but recommended for project organization).

3.3.4 User Environment:

End-User Devices: Computers or laptops with compatible hardware and software specifications mentioned above for running the final application.

User Documentation: User manuals, guides, or online help resources to assist users in installing, configuring, and using the application effectively.

User Support: Helpdesk or support channels for addressing user queries, issues, and feedback during and after deployment.

By adhering to these hardware and software requirements, developers can ensure compatibility, performance, and usability of the cybersecurity utility application across different environments and user devices

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

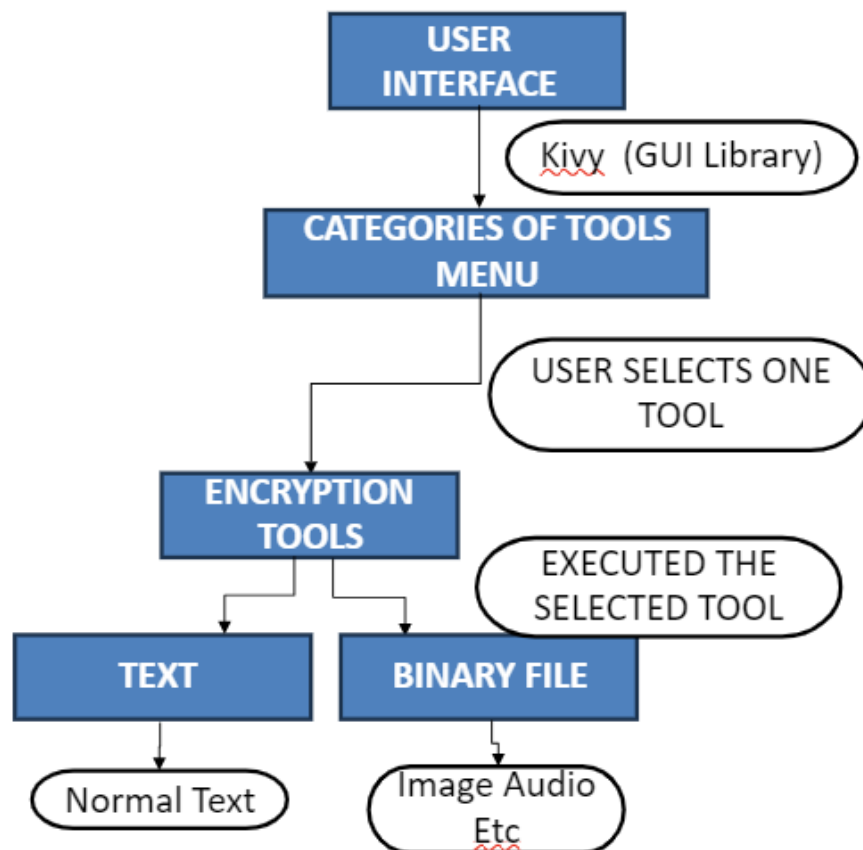


Figure 4.1 Architecture of the proposed system

4.2.GRAPHICAL USER INTERFACE

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information and represent actions that the user can take. The objects change color, size, or visibility when the user interacts with them.

The GUI was created at Xerox PARC (Palo Alto Research Center) by Alan Kay, Douglas Engelbart, and other researchers in 1981. Later, Apple introduced the Lisa computer with a GUI on January 19, 1983.

4.2.1Elements of GUI

To make a GUI as user-friendly as possible, there are different elements and objects that the user use to interact with the software. Below are these elements with a brief description.

- **Button** - A graphical representation of a button that acts as a program when pressed
- **Dialog box** - A type of window that displays additional information and asks a user for input.
- **Hamburger menu** - A button with three lines that expands and shows a hidden menu when pressed.
- **Icon** - Small graphical representation of a program, feature, or file.
- **Kebab menu** - A button with three dots that expands and shows a hidden menu when pressed.
- **Menu** - List choices offered to the user through the menu bar.
- **Menu bar** - Thin, horizontal bar containing the labels of menus.
- **Ribbon** - Replacement for the file menu and toolbar that groups program activities together.
- **Tab** - Clickable area at the top of a window that shows another page or area.
- **Toolbar** - A row of buttons, often near the top of an application window that controls software functions.

- **Window** - Rectangular section of the computer's display that shows the program currently being used.

4.2.2. *How does GUI work*

A GUI uses windows, icons, and menus to perform commands, like opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a mouse, a keyboard can also be used via keyboard shortcuts or arrow keys.

For example, if you want to open a program on a GUI system, move the mouse pointer to the program's icon and double-click it. With a command line interface, you need to know the commands to navigate to the directory containing the program, list the files, and then run the file.

4.2.3. *Benefits of GUI*

A GUI uses windows, icons, and menus to perform commands, like opening, deleting, and moving files. Although a GUI operating system is primarily navigated using a mouse, a keyboard can also be used via keyboard shortcuts or arrow keys.

For example, if you want to open a program on a GUI system, move the mouse pointer to the program's icon and double Click it with a command icon interface.



4.3. CYBERSECURITY TOOLS

Protecting our IT environment is very critical. Every organization needs to take cybersecurity very seriously. There are number of hacking attacks that affect businesses of all sizes. Hackers, malware, and viruses are some of the real security threats in the virtual world. It is essential that every company is aware of the dangerous security attacks, and it is necessary to keep themselves secure. There are many different aspects of cyber defense may need to be considered. Here are six essential tools and services that every organization needs to consider to ensure their cybersecurity is as strong as possible. They are described below:



Fig.4.3 Cybersecurity Tools

4.4. TYPES OF CYBERSECURITY TOOLS

4.4.1. Network Security Monitoring Tools:

These are software that is specifically developed to monitor the network for potential security issues. It is intended to collect metrics related to client-server communications, encrypted traffic sessions, and other network processes. It also allows you to do traffic analysis to determine traffic flow patterns. Some of the available tools include Argus, P0f, Splunk, etc.

4.4.2. Encryption Tools:

These are software that uses cryptography to encrypt information to prevent unauthorized access. This is accomplished by using some cryptographic algorithm to encrypt the data, i.e converting plain text to incomprehensible form, and decrypting the data at the destination, i.e converting the incomprehensible form of text to the original plain text again. Some examples include Tor, KeePass, and TrueCrypt.

4.4.3. Web Vulnerability Scanning Tools:

Vulnerability scanners are advanced tools that examine web applications for security flaws. They examine web applications for common security flaws like cross-site scripting, SQL injection, and cross-site request forgery. Examples include Burp Suite, Nikto, etc.

4.4.4 Penetration Testing Tools:

Penetration testing tools are used in penetration tests to automate specific procedures, accelerate testing, and find problems that would be difficult to detect using only human analytic techniques. Some of the common tools are Metasploit, Kali Linux, etc.

4.4.5. Packet Sniffers:

Packet sniffers also known as packet analyzers are software that is used to monitor network traffic. It works by examining the data packet streams

that flow in a network. Wireshark is a leading Example of a packet sniffer.

4.5.KIVY FRAMEWORK

- Kivy is an open-source and Graphical User Interface (GUI) development platform for Python.
 - It helps us to develop mobile applications and multi-touch application software with a NUI (Natural User Interface)
 - It allows developers to build an application once and use it across all devices.
 - It can also access mobile APIs to manipulate things such as the camera on a phone, GPS tracking, vibrator, and so on.
 - It contains various components for creating an application such as:
- A graphical library OpenGL ES 2.
 - An extensive range of widgets that support multi-touch.
 - An intermediate Kv language to design custom widgets.
 - Extensive support for input devices such as a mouse, keyboard, and TUIO, as well as OS-specific multi-touch events.

4.5.1 Kivy Architecture

The Kivy Architecture Consists of the following:

- ▯ Core Providers and Input Providers
- ▯ Graphics
- ▯ Core
- ▯ UIX

- ▢ Modules
- ▢ Input Events
- ▢ Widgets And Input Dispatching

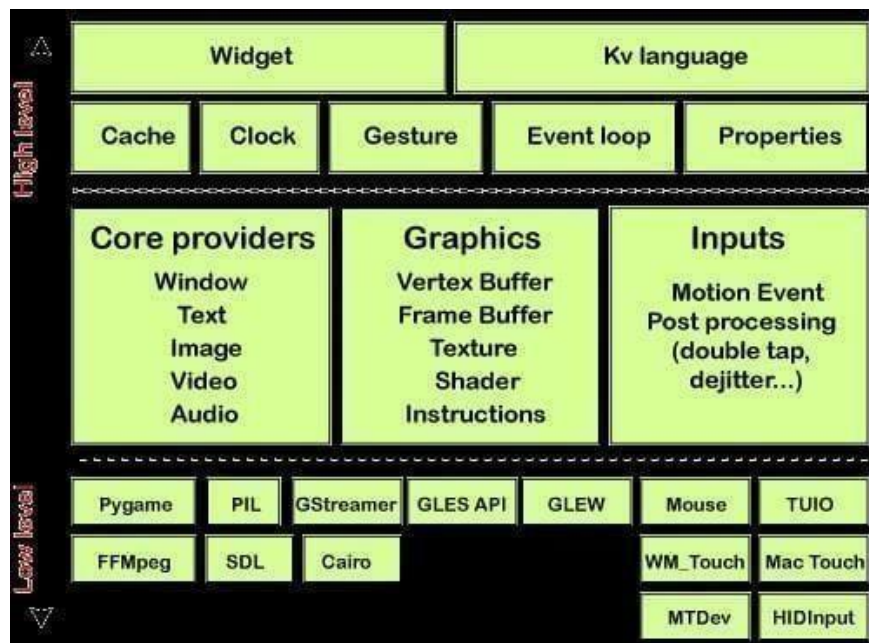


Fig.4.4 Kivy Architecture

- **Core providers and Input providers:** Modularity and Abstraction is the key to understanding Kivy's internals. We abstract basic tasks like opening a window, playing audio, displaying images and text, getting images from a camera, spelling correction etc. These are called core tasks and make API both easy to use and extend. A piece of code that uses one of the APIs to talk to the operating system on one side and Kivy on the other i.e., acting as an intermediate communication layer is called core provider.
- **Graphics:** Graphics APIs Kivy has the ability to automatically optimize the drawing commands.
- **Core:** The code in the core package provides commonly used features, such as:
 - **Clock:** The clock is used to schedule timer events. One-shot

timers and periodic timers are both supported.

- **Cache:** When you need to cache something that you often use, you can use the class for that instead of writing your own.
- **Gesture Detection:** Provides a simple gesture recognizer that you can use to detect different kinds of strokes, like circles or rectangles.
- **Kivy Language:** The kivy language is used to easily and efficiently describe user interfaces.
- **Properties:** They are not normal properties that you may know from Python. They are the property classes that link your widget code with the user interface description.

- **UIX:** This contains widgets and layouts that are commonly used.

Widgets are user interface elements that are added to the program to provide some kind of functionality. They may or may not be visible. Widgets receive Motion Events.

Ex: file browser, buttons, sliders, lists and so on.

Layouts are used to arrange widgets. It is possible to calculate your widgets' positions by yourself, but often it is more convenient to use one of our readymade layouts. You can also nest layouts.

Ex: Grid Layouts or Box Layouts.

- **Modules:** If you have ever used a modern web browser and customized it with some add-ons then you know the basic idea behind our module classes. They are used to inject functionality into Kivy programs, even if the original author did not include it.

Ex: module that always shows the FPS of the current application and some graph depicting the FPS over time.

- **Input Events:** In the Kivy architecture, the Kivy Input events play an important role. Kivy abstract contains different input events like touches, mice, TUIO, Mouse, MTDev, and HID Input. Input types are characterized by an instance of the touch () class. A touch () class instance can be one of the following three states.

- Down
- Move
- Up

- **Widgets and input dispatching:** Widgets are frequently used in GUI programming to describe that part of the program which interacts with a user. In kivy, widgets work as an object that receives input events. Widgets are arranged in the form of a widget tree. A widget may contain any number of child or none.

CHAPTER -5

MODULE DESCRIPTION

The overall modules and their description have been explained below following this module describes the entire workflow and process of our methodology has been explained Fig 5.1 represents the block diagram for the methodology that has been proposed.

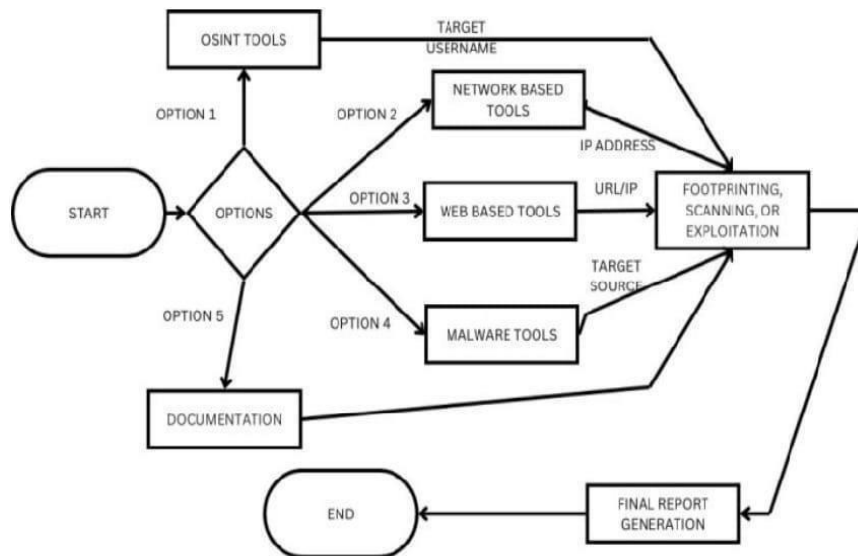


Fig.5.1 Block Diagram

The methodology is split as:

- **Options:**

This provides the list of all the available tools under different domains, and it requests the user or pentester to choose their desired tool domain to carry on with the process.

- **OSINT tools:**

This is the first option from the list provided. OSINT tools are a collection of various software and online resources that are used together to collect all the possible publicly available information about a particular target, they can be used for several functionalities such as background checks,

investigation, intelligence gathering, etc. The tools can be used in several domains such as social media scraping tools, data visualization tools, web archive tools, etc.

- **Network-based tools:**

This is the second option from the list provided. The network-based tools are used to perform a network penetration test to identify vulnerabilities in the system that an attacker could use to exploit and evaluate the

effectiveness of an organization's security measures. It includes a variety of services such as port scanners, Arp scanners, ARP spoofers, packet sniffers, etc.

- ***Web-based tools:***

This is the third option from the list provided. A web-based tool is used by the user/ pen tester to exploit and evaluate the underlying vulnerabilities present in the web application which in turn is used to evaluate the overall security measures of the target web application. This tool includes services such as a web directory crawler,

DNS spoofer, injection-based vulnerability scanner, etc.

- ***Malware tools:***

This is the fourth option from the list provided. Malware tools are used to spy, record, or disrupt the target's system. This is used so that the user/ pen tester can have awareness about potential entry points and warn potential victims to not download suspicious files and click on suspicious links and protect their system from not succumbing to phishing and other social engineering attacks. The services provided in this tool are key logger, backdoor, file interactor, packaging, etc.

- ***Documentation:***

This is the final option from the list provided. Documentation is used to train the user/tester about the tool by providing instructions on how to execute each service, it also provides insight into the attack and the reason behind the usage of the attack. It provides documentation for each tool and its respective service. The documentation part provides steps, instructions, and merits as well as demerits of using the particular tool or service, it also provides an overall policy that will be discussed and verified between the tester and the organization.

5.1 DESIGNED TOOLS:

• ***Password Generator Tool:***

This tool generates strong, randomized passwords according to specified criteria such as length, complexity, and character types (e.g., uppercase letters, lowercase letters, numbers, special characters).

Usages: Users can input their desired password length and select the types of characters they want to include (e.g., letters, numbers, special characters). The tool then generates a random password meeting the specified criteria, which users can then use for securing their accounts, Wi-Fi networks, or any other sensitive information.

• ***Url Mask Tool:***

This tool obscures or "masks" URLs by converting them into a shorter, more user-friendly format while preserving their functionality. It is often used for aesthetic purposes, to make URLs more shareable, or to track clicks on links without revealing the original URL.

Usages: Users input the long URL they want to mask, and the tool generates a shorter, custom alias that redirects to the original URL. Users can then share this masked URL with others, who can access the intended destination without seeing the full, lengthy URL.

• ***Integrity Checker Tool:***

This tool verifies the integrity and authenticity of files by calculating cryptographic hashes (e.g., MD5, SHA-256) and comparing them with precomputed hashes. It helps detect any unauthorized modifications or tampering with files.

Usages: Users select the file they want to verify, and the tool computes its hash value using a chosen hashing algorithm. They then compare this hash value with a known, trusted hash value provided by the file's creator or a reputable source. If the hash values match, it confirms that the file has not been altered since the hash was generated, ensuring its integrity and authenticity.

• ***Encryption/Decryption Tool:***

This tool encrypts plaintext data into ciphertext using cryptographic algorithms and keys, making it unreadable without the corresponding decryption key. Conversely, it decrypts ciphertext back into plaintext using the decryption key.

Usages: Users input the plaintext they want to encrypt along with a cryptographic key, and the tool applies a chosen encryption algorithm (e.g., AES, RSA) to generate the ciphertext. To decrypt the ciphertext, users input the ciphertext and the decryption key, allowing the tool to reverse the encryption process and recover the original plaintext. This tool is commonly used to protect sensitive data during storage or transmission, ensuring confidentiality and privacy.

The sample execution of one of the modules is demonstrated and shown using the screenshots attached below. Fig 5.2 shows the main menu and network menu of the proposed tool.

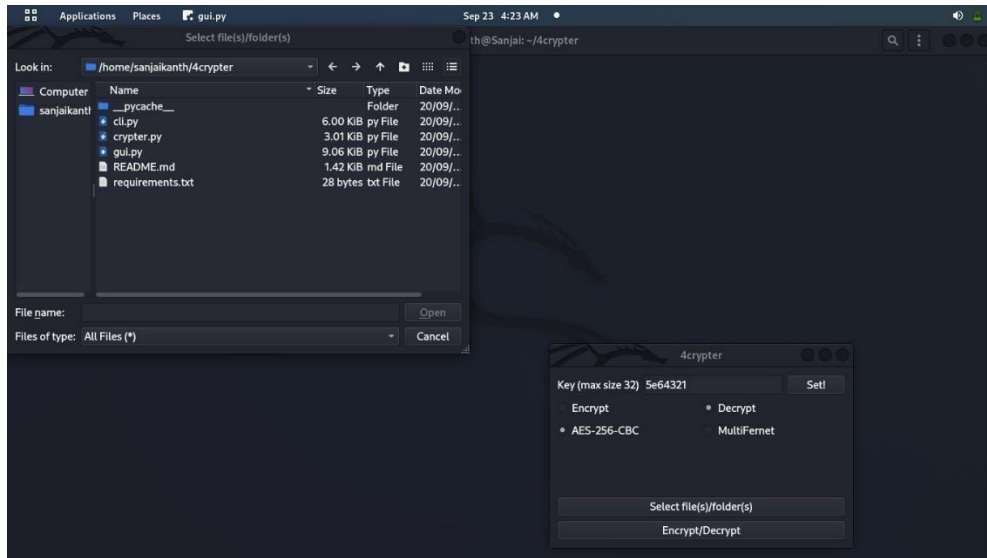


Fig.5.2 Encryption/Decryption Tool output

As shown in Fig 5.2 the user is prompted to select an option in order to determine the service that is to be used.

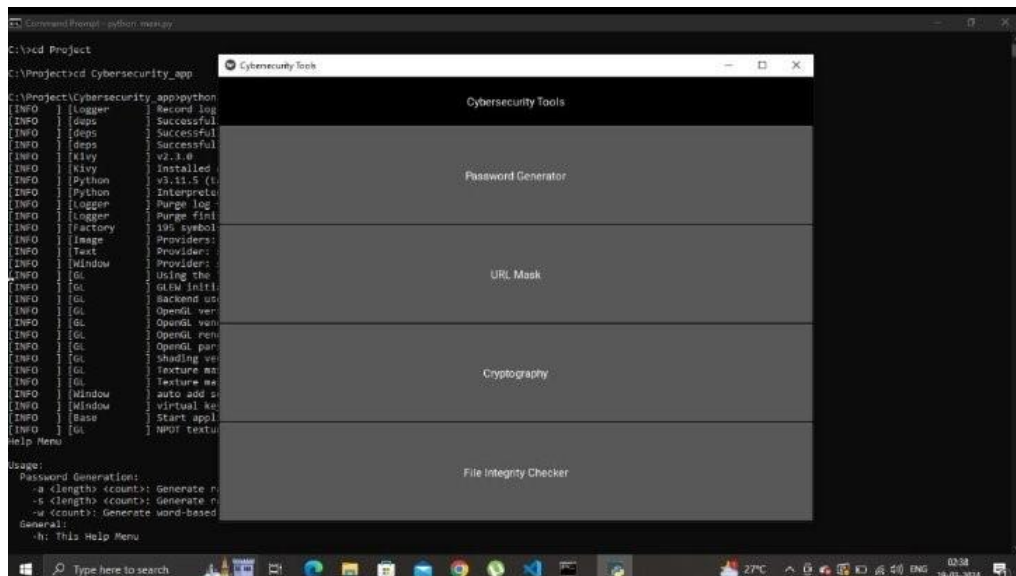


Fig.5.3. GUI-Output

Fig.5.3 is a Gui image that was built in the Kivy framework using codes.

CHAPTER 6

CONCLUSION

In conclusion, the development of a Graphical User Interface (GUI) using Kivy for accessing a comprehensive suite of cybersecurity tools represents a significant advancement in user experience and efficiency within the cybersecurity domain. By addressing the shortcomings of traditional command-line interfaces and standalone applications, the proposed system offers a streamlined and intuitive approach to accessing, managing, and utilizing cybersecurity tools. Through the implementation of categorized tool buttons and centralized storage of tool paths within the Kivy application, users benefit from a user-friendly interface that simplifies navigation and enhances accessibility. The inclusion of a prominently displayed heading, "Cybersecurity Utility Tools," serves to reinforce the purpose and functionality of the GUI, ensuring clarity and usability for users of varying experience levels. Furthermore, the proposed system not only improves the user experience but also promotes customization and scalability. The modular nature of the GUI allows for easy integration of additional tools and categories, facilitating adaptation to evolving cybersecurity needs and preferences. Additionally, the flexibility inherent in the design enables future enhancements and updates to be seamlessly incorporated, ensuring the longevity and relevance of the system over time. Overall, the development of this GUI utilizing Kivy represents a significant step forward in democratizing access to cybersecurity tools, empowering users to enhance their cybersecurity practices with greater efficiency, convenience, and effectiveness. Through its user-centric design and robust functionality, the proposed system sets a new standard for user interfaces within the cybersecurity landscape, ultimately contributing to the advancement of cybersecurity education, awareness, and resilience in an increasingly digital world.

FUTURE WORKS

For future iterations of this project, several avenues for enhancement and expansion can be explored to further optimize the user experience and

functionality of the GUI for cybersecurity utility tools. One potential area of focus is the integration of additional tools and categories, catering to a broader range of cybersecurity needs and preferences. This could involve collaborating with cybersecurity experts to identify emerging tools and technologies that can be seamlessly incorporated into the GUI, thereby ensuring its relevance and comprehensiveness in addressing evolving cybersecurity challenges. Additionally, enhancements to the GUI's design and layout could be explored to further improve usability and aesthetics, potentially incorporating features such as customizable themes or interactive elements to enhance engagement. Moreover, the implementation of advanced functionalities, such as user authentication and authorization mechanisms, could be considered to enhance security and access control within the GUI. Furthermore, the development of companion mobile applications or web interfaces could extend the accessibility of the GUI beyond traditional desktop environments, enabling users to conveniently access cybersecurity tools from a variety of devices and platforms. Overall, future works for this project should aim to continually innovate and adapt to the evolving landscape of cybersecurity, ultimately empowering users with the tools and resources they need to effectively safeguard their digital assets and information.

REFERENCES

- [1] Kumar, Sharath, and S. Pallavi. "An Effective Network Monitoring Tool for Distributed Networks." 2020 Fourth International Conference on ISMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC). IEEE, 2020.

- [2] Sharma, Ritik, Sarishma Dangi, and Preeti Mishra. "A comprehensive review on encryption based open source cyber security tools." 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC). IEEE, 2021.

- [3] Anand, Piyush, and Ajay Shankar Singh. "Penetration Testing Security Tools: A Comparison." 2021 10th International Conference on System Modeling and Advancement in Research Trends (SMART). IEEE, 2021.

- [4] Kunda, Mohammed Ali, and Izzat Alsmadi. "Practical web security testing: Evolution of web application modules and open source testing tools." 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA). IEEE, 2022.

- [5] Rai, Swarnima, Vaaruni Choubey, and Puneet Garg. "A Systematic Review of Encryption and Keylogging for Computer System Security." 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT). IEEE, 2022.

APPENDIX 1

SAMPLE CODE

TOOLS:

1. Encryption/Decryption Tool:

```
from cryptography.fernet import Fernet

def generate_key():

    return Fernet.generate_key()

def save_key(key, filename="secret.key"):

    with open(filename, "wb") as key_file:

        key_file.write(key)

def load_key(filename="secret.key"):

    return open(filename, "rb").read()

def encrypt_file(file_path, key):

    cipher = Fernet(key)

    with open(file_path, "rb") as file:

        data = file.read()

        encrypted_data = cipher.encrypt(data)

    with open(f"{file_path}.enc", "wb") as encrypted_file:

        encrypted_file.write(encrypted_data)

def decrypt_file(encrypted_file_path, key):

    cipher = Fernet(key)

    with open(encrypted_file_path, "rb") as encrypted_file:

        encrypted_data = encrypted_file.read()

        decrypted_data = cipher.decrypt(encrypted_data)

    with open(f"{encrypted_file_path[:-4]}_decrypted.txt", "wb") as
decrypted_file:
```

```

        decrypted_file.write(decrypted_data)

if __name__ == "__main__":
    key = generate_key()
    save_key(key)

    file_path = input("Enter the path to the file to encrypt: ")
    encrypt_file(file_path, key)

    encrypted_file_path = input("Enter the path to the encrypted file to
decrypt: ")
    decrypt_file(encrypted_file_path, key)

```

2.Url Mask Tool:

```

import base64
import random
import string
import sys

def mask_url(url, mask, shorten=False):
    encoded_url = base64.b64encode(url.encode()).decode()

    random_param = ".join(random.choices(string.ascii_letters +
string.digits, k=8))

    random_value = ".join(random.choices(string.ascii_letters + string.digits,
k=8))

    masked_url =
f"https://{mask}/{random_param}={random_value}&url={encoded_url}"

    if shorten:

        print("Shortening service not implemented. Masked URL:",
masked_url)

```

```

else:

    print("Masked URL:", masked_url)

def do_help():

    print("Help Menu")

    print("\nUsage:")

    print(" URL Masking:")

    print(" -u <original_url> <mask> [true|false]: Mask the original URL with
the provided mask.")

    print(" General:")

    print(" -h: This Help Menu")

if __name__ == "__main__":

    if len(sys.argv) < 2 or sys.argv[1] == '-h':

        do_help()

    elif sys.argv[1] == '-u':

        if len(sys.argv) < 4:

            print("\n./url_masking.py -u <original_url> <mask> [true|false]: For URL
Masking")

            else:

                shorten = sys.argv[4] == 'true' if len(sys.argv) > 4 else False

                mask_url(sys.argv[2], sys.argv[3], shorten)

            else:

                print("\n./url_masking.py -h: For Help Menu")

```

3.Password Generator Tool:

```
import random
```

```

import string

import sys

def genrandom_number(max_value):
    return random.randint(1, max_value)

def random_special_char():
    special_chars = ['!', '@', '#', '$', '%', '^', '&', '(', ')', '_', '-', '+']
    return random.choice(special_chars)

def genrandom_alphanumeric(max_length):
    return "".join(random.choices(string.ascii_letters + string.digits,
k=max_length))

def genrandom_alphanumeric_special(max_length):
    password = genrandom_alphanumeric(max_length - 2)
    return password + random_special_char() + random_special_char()

def choose_word(dictionary):
    dictionary_max = len(dictionary)

    if dictionary_max > 0:
        line1 = genrandom_number(dictionary_max)
        line2 = genrandom_number(dictionary_max)
        word1 = dictionary[line1 - 1].strip().capitalize()
        word2 = dictionary[line2 - 1].strip().capitalize()
        return f"{word1}{genrandom_number(100)}{word2}"
    else:
        return "\nDictionary is empty.\n"

```

```

def do_help():
    print("Help Menu")
    print("\nUsage:")
    print(" Password Generation:")
    print(" -a <length> <count>: Generate random alphanumeric
passwords.")
    print(" -s <length> <count>: Generate random alphanumeric
passwords with special characters.")
    print(" -w <count>: Generate word-based passwords using the
dictionary.")
    print(" General:")
    print(" -h: This Help Menu")
if __name__ == "__main__":
    if len(sys.argv) < 2 or sys.argv[1] == '-h':
        do_help()
    elif sys.argv[1] == '-a':
        password_length = int(sys.argv[2]) if len(sys.argv) > 2 else 20
        count = int(sys.argv[3]) if len(sys.argv) > 3 else 1
        for _ in range(count):
            print(genrandom_alphanumeric(password_length))
    elif sys.argv[1] == '-s':
        password_length = int(sys.argv[2]) if len(sys.argv) > 2 else 20
        count = int(sys.argv[3]) if len(sys.argv) > 3 else 1
        for _ in range(count):
            print(genrandom_alphanumeric_special(password_length))

```

```

elif sys.argv[1] == '-w':
    count = int(sys.argv[2]) if len(sys.argv) > 2 else 1
    dictionary = ["word1", "word2", "word3"] # Add your dictionary here
    for _ in range(count):
        print(choose_word(dictionary))
    else:
print("\n./password_generator.py -h: For Help Menu")

```

4. Integrity Checker Tool:

```

import socket
import concurrent.futures

def scan_ports(target_ip, start_port, end_port, num_threads=10,
verbose=False):
    open_ports = []
    def scan(port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)
        result = sock.connect_ex((target_ip, port))
        if result == 0:
            open_ports.append(port)
            if verbose:
                print(f"Port {port} is open on {target_ip}")
        sock.close()
    if verbose:

```

```

        print(f"Scanning ports {start_port}-{end_port} on {target_ip} using
{num_threads} threads...")

    with
concurrent.futures.ThreadPoolExecutor(max_workers=num_threads) as
executor:

        executor.map(scan, range(start_port, end_port + 1))

    if verbose:

        if open_ports:

            print(f"Open ports on {target_ip}: {open_ports}")

        else:

            print(f"No open ports found on {target_ip}")

    return open_ports

```

```
def display_help():
```

```
    help_message = """
```

```
    Multi-threaded Network Port Scanner
```

```
    Command-line arguments:
```

```

    -h, --help          Show this help message and exit.

    -t, --target        Specify the target IP address.

    -sp, --start_port   Specify the starting port.

    -ep, --end_port     Specify the ending port.

    -nt, --num_threads  Specify the number of threads for scanning.

    -v, --verbose       Enable verbose mode.

```

```
    Example usage:
```

```
python port_scanner.py -t 192.168.1.1 -sp 1 -ep 1024 -nt 20 -v
```

```

"""

print(help_message)

if __name__ == "__main__":

    import argparse

    parser = argparse.ArgumentParser(description="Multi-threaded Network
Port Scanner")

    parser.add_argument("-t", "--target", required=True, help="Specify the
target IP address.")

    parser.add_argument("-sp", "--start_port", type=int, required=True,
help="Specify the starting port.")

    parser.add_argument("-ep", "--end_port", type=int, required=True,
help="Specify the ending port.")

    parser.add_argument("-nt", "--num_threads", type=int, default=10,
help="Specify the number of threads for scanning.")

    parser.add_argument("-v", "--verbose", action="store_true",
help="Enable verbose mode.")

    parser.add_argument("-help", "--help_menu", action="store_true",
help="Display the help menu.")

    args = parser.parse_args()

    if args.help_menu:

        display_help()

    else:

        open_ports = scan_ports(args.target, args.start_port, args.end_port,
args.num_threads, args.verbose)

```

GUI:

```

from kivy.app import App

from kivy.uix.button import Button

from kivy.uix.boxlayout import BoxLayout

class CybersecurityToolsApp(App):

    def build(self):

        layout = BoxLayout(orientation='vertical')

        # Title label

        title_label = Button(text='Cybersecurity Tools', size_hint_y=None,
height=50)

        layout.add_widget(title_label)

        # List of categories

        categories = ['Category 1', 'Category 2', 'Category 3', 'Category 4',
'Category 5']

        for category in categories:

            btn = Button(text=category, size_hint_y=None, height=40)

            btn.bind(on_press=self.on_category_click)

            layout.add_widget(btn)

        return layout

    def on_category_click(self, instance):

        print(f'{instance.text} clicked!') # Replace print statement with desired
action

if __name__ == '__main__':

    CybersecurityToolsApp().run()

```

APPENDIX 2

Screenshots

```

1  import random
2  import string
3  import sys
4
5  def genrandom_number(max_value):
6      return random.randint(1, max_value)
7
8  def random_special_char():
9      special_chars = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '_', '=', '+']
10     return random.choice(special_chars)
11
12 def genrandom_alphanumeric(max_length):
13     return ''.join(random.choices(string.ascii_letters + string.digits, k=max_length))
14
15 def genrandom_alphanumeric_special(max_length):
16     password = genrandom_alphanumeric(max_length - 2)
17     return password + random_special_char() + random_special_char()
18
19 def choose_word(dictionary):
20     dictionary_max = len(dictionary)
21     if dictionary_max > 0:
22         line1 = genrandom_number(dictionary_max)
23         line2 = genrandom_number(dictionary_max)
24         word1 = dictionary[line1 - 1].strip().capitalize()
25         word2 = dictionary[line2 - 1].strip().capitalize()
26         return f'{word1}{genrandom_number(100)}{word2}'
27     else:
28         return "\nDictionary is empty.\n"
29
30 def do_help():
31     print("Help Menu")
32     print("\nUsage:")

```

```

44 if len(sys.argv) < 4 or sys.argv[4] == "-h":
45     do_help()
46
47 elif sys.argv[1] == "-a":
48     password_length = int(sys.argv[2]) if len(sys.argv) > 2 else 20
49     count = int(sys.argv[3]) if len(sys.argv) > 3 else 1
50     for _ in range(count):
51         print(genrandom_alphanumeric(password_length))
52
53 elif sys.argv[1] == "-s":
54     password_length = int(sys.argv[2]) if len(sys.argv) > 2 else 20
55     count = int(sys.argv[3]) if len(sys.argv) > 3 else 1
56     for _ in range(count):
57         print(genrandom_alphanumeric_special(password_length))
58
59 elif sys.argv[1] == "-w":
60     count = int(sys.argv[2]) if len(sys.argv) > 2 else 1
61     dictionary = ["word1", "word2", "word3"] # Add your dictionary here
62     for _ in range(count):
63         print(choose_word(dictionary))
64
65 else:
66     print("\n./password_generator.py -h: For Help Menu")

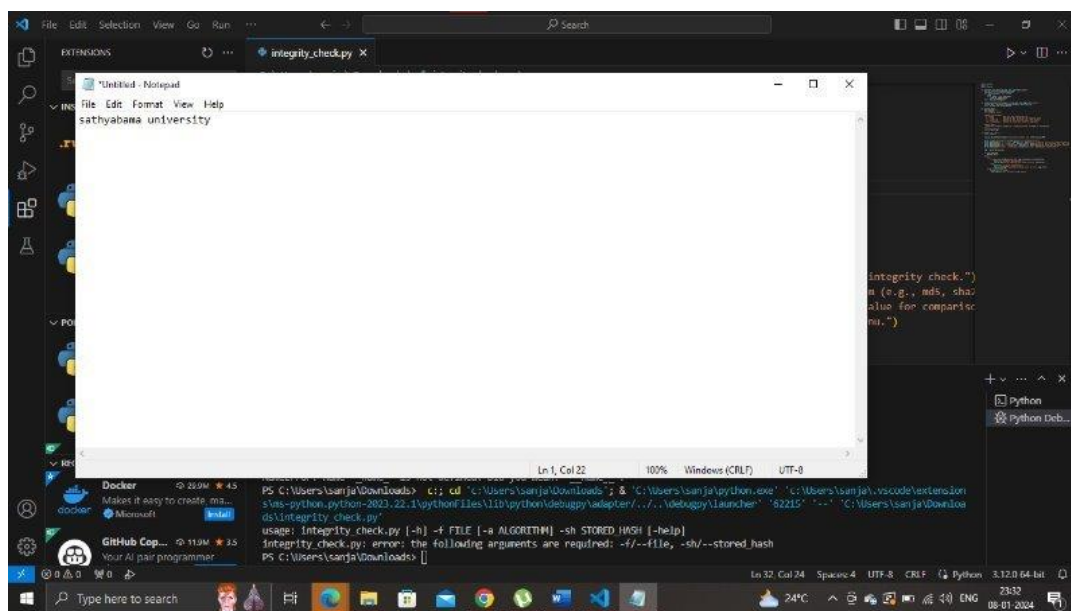
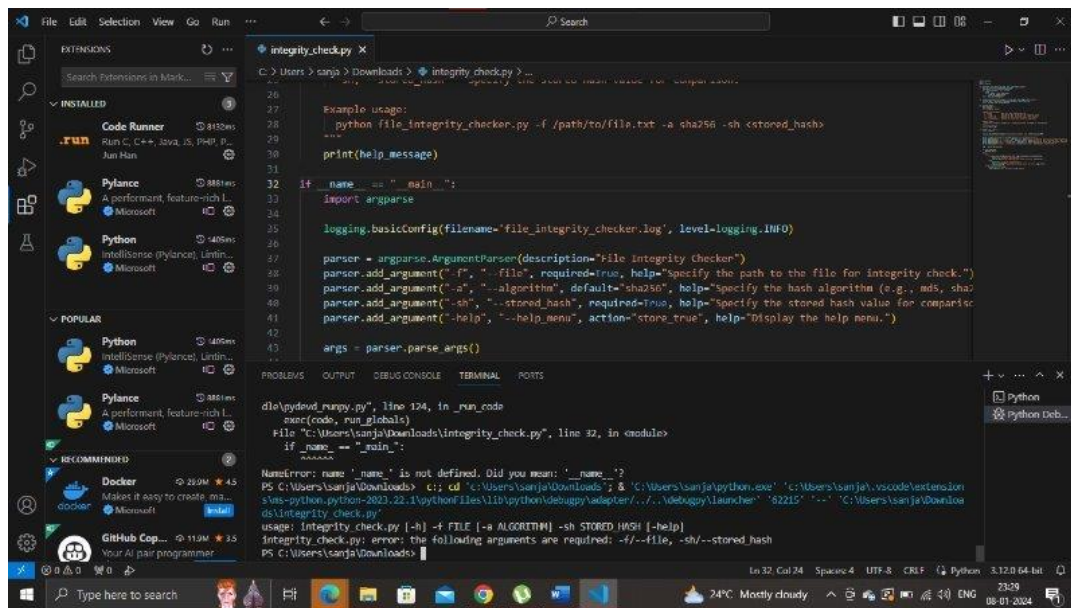
```

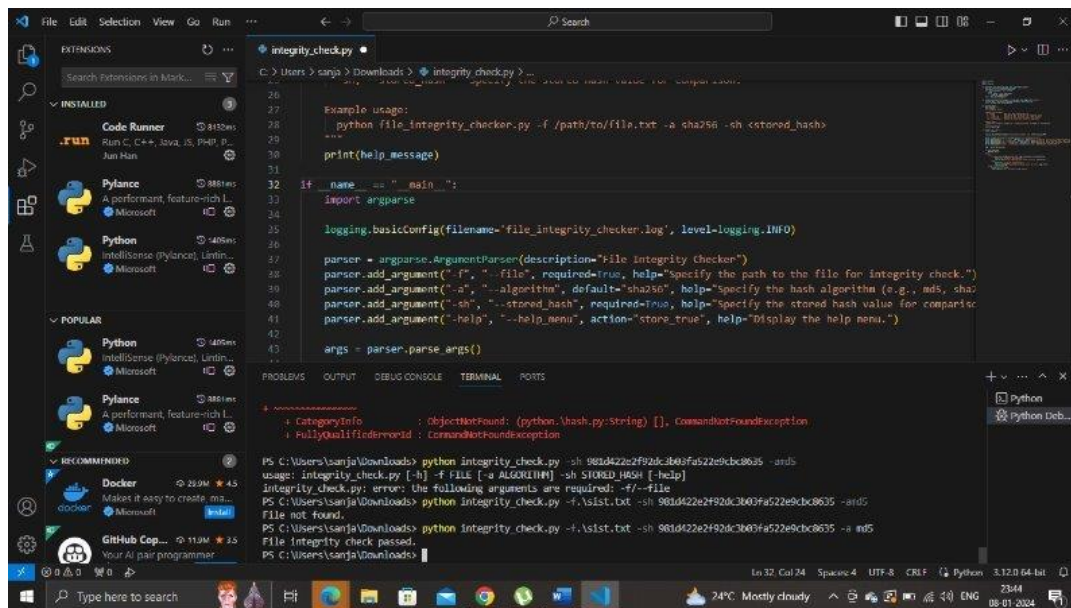
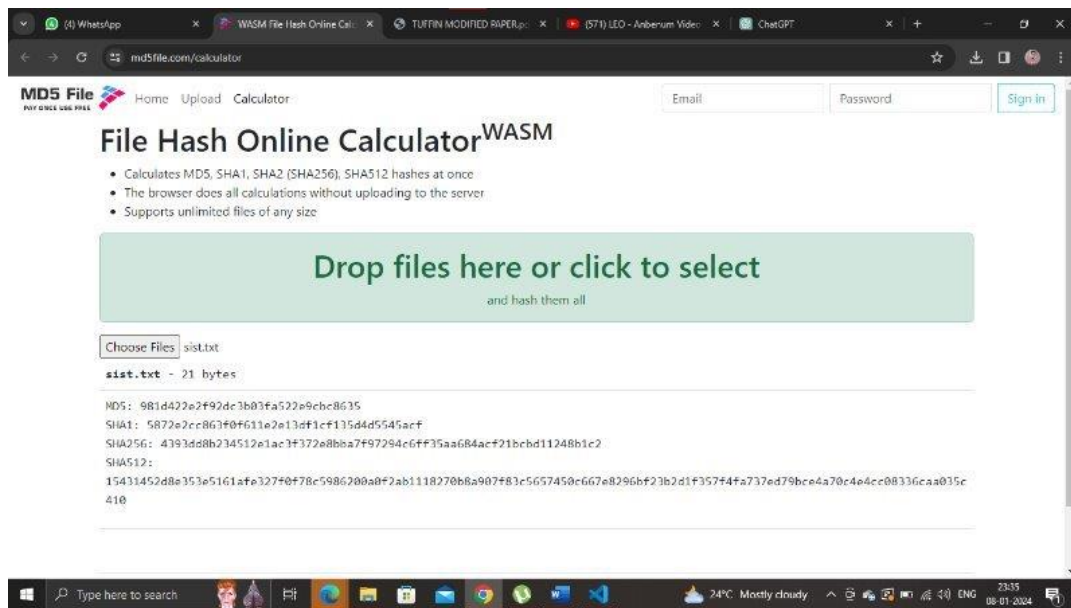
The screenshot shows the Visual Studio Code editor with the file `password_generator.py` open. The left sidebar displays the 'EXTENSIONS' view with a search bar and lists of installed, popular, and recommended extensions. The main editor area shows the Python code for a password generator. The code includes imports for `random`, `string`, and `sys`. It defines functions for generating random numbers, special characters, alphanumeric strings, and word-based passwords. A `choose_word` function is also present, which takes a dictionary as input. The bottom status bar shows the file is in UTF-8 encoding, using the Python interpreter, and is 3,320 bytes in size. The system tray at the bottom indicates a temperature of 24°C and a date of 08-01-2024.

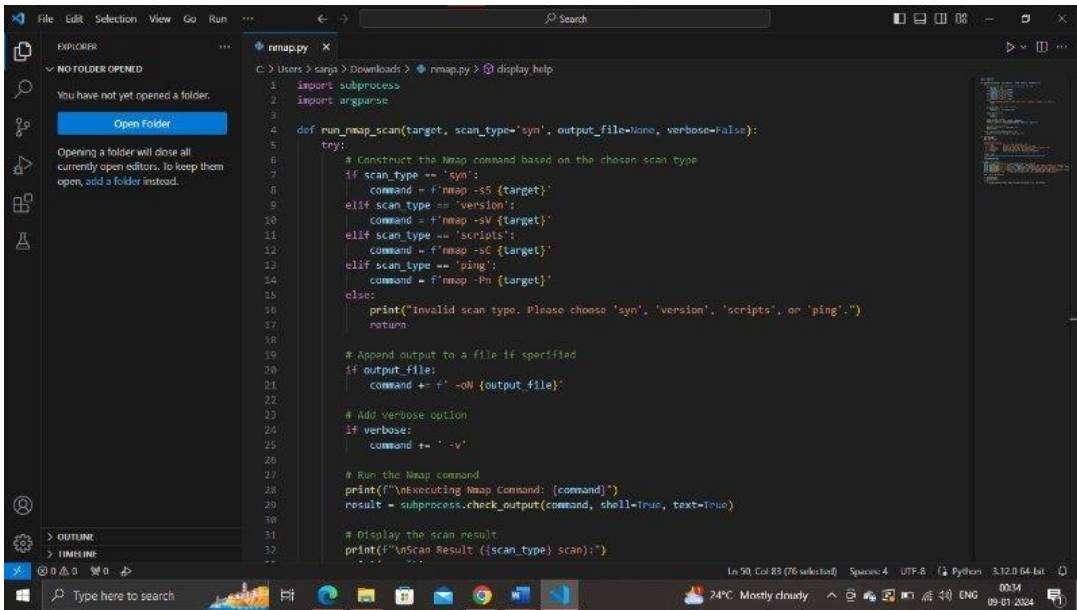
```
1 import random
2 import string
3 import sys
4
5 def generate_random_number(max_value):
6     return random.randint(1, max_value)
7
8 def generate_random_special_char():
9     special_chars = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '_', '+']
10    return random.choice(special_chars)
11
12 def generate_random_alphanumeric(max_length):
13    return ''.join(random.choices(string.ascii_letters + string.digits, k=max_length))
14
15 def generate_random_alphanumeric_special(max_length):
16    password = generate_random_alphanumeric(max_length - 2)
17    return password + generate_random_special_char() + generate_random_special_char()
18
19 def choose_word(dictionary):
```

The screenshot shows the Visual Studio Code editor with the file `integrity_checker.py` open. The left sidebar displays the 'EXTENSIONS' view with a search bar and lists of installed, popular, and recommended extensions. The main editor area shows the Python code for a file integrity checker. The code includes imports for `hashlib`, `os`, and `logging`. It defines functions for calculating file hashes using SHA256, checking file integrity by comparing current and stored hashes, and displaying help information. The code also includes a main block that checks if the script is being run directly. The bottom status bar shows the file is in UTF-8 encoding, using the Python interpreter, and is 3,320 bytes in size. The system tray at the bottom indicates a temperature of 24°C and a date of 08-01-2024.

```
1 import hashlib
2 import os
3 import logging
4
5 def calculate_file_hash(file_path, hash_algorithm="sha256"):
6     hash_obj = hashlib.new(hash_algorithm)
7     with open(file_path, "rb") as file:
8         chunk = 0
9         while chunk := file.read(8192):
10            hash_obj.update(chunk)
11    return hash_obj.hexdigest()
12
13 def check_file_integrity(file_path, stored_hash, hash_algorithm="sha256"):
14    current_hash = calculate_file_hash(file_path, hash_algorithm)
15    return current_hash == stored_hash
16
17 def display_help():
18    help_message = """
19    File Integrity Checker
20
21    Command-line arguments:
22    -h, --help          Show this help message and exit.
23    -f, --file           Specify the path to the file for integrity check.
24    -a, --algorithm      Specify the hash algorithm (e.g., md5, sha256).
25    -s, --stored_hash    Specify the stored hash value for comparison.
26
27    Example usage:
28    python file_integrity_checker.py -f /path/to/file.txt -a sha256 -s <stored_hash>
29    """
30    print(help_message)
31
32 if __name__ == "__main__":
```







```
cmd Command Prompt
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sanja>cd C:\
C:\>cd Project
C:\Project>cd Cybersecurity_app
C:\Project\Cybersecurity_app>python main.py
```

```
cmd Command Prompt
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sanja>cd C:\
C:\>cd Project
C:\Project>cd Cybersecurity_app
C:\Project\Cybersecurity_app>python main.py
```

