**ABSTRACT**

As clipboard attacks and unintended information leakage are more probable, copied data security is now most crucial in terms of cybersecurity. "An Intelligent Real-Time Clipboard Monitoring and Alerting System for Windows with Malicious URL Detection and Password Breach Analysis"—a Python utility that seeks to defend clipboard usage through the identification of sensitive data, filtering of suspected malicious content, and alerting users in real-time with actionable alerts—is this project's contribution.

The system monitors clipboard activity in real time on a continuous basis and automatically compares copied text to predefined detection patterns. It is sensitive to and detects a wide range of security threats including emails, credit card numbers, passwords, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials), and dangerous or suspicious URLs. For better threat detection functionality, the utility comes with external threat intelligence API integrations—Google Safe Browsing for detecting threats from URLs and Have I Been Pwned (HIBP) for password breach monitoring.

The project also addresses a crucial vulnerability found in OWASP Top 10 – A04:2021 (Insecure Design) by inhibiting accidental disclosure of data through clipboard data. The application also features features such as blocking of malicious commands, cloud clipboard syncing prevention, detection of clipboard intrusions, and reminder adjustable to delete sensitive content. A Tkinter-built GUI provides a friendly interface with light/dark schemes, hotkeys, and settings customization to large extent.

Designed particularly for the Windows environment, this intelligent light monitoring system helps users maintain data in a pure state and benefits from improved security posture, where sensitive data is being processed.

**Keywords**: Clipboard Security, Data Leakage Prevention, Real-Time Monitoring, Malicious URL Detection, Password Breach Analysis, Google Safe Browsing API, Have I Been Pwned (HIBP), OWASP A04:2021, Insecure Design, Python, Tkinter GUI, Sensitive Data Detection, Windows Security Tool, Cybersecurity

**Chapter 1: INTRODUCTION**

* 1. **Introduction and Problem Summary**

# With increasing reliance on copy-paste actions in modern computing systems, the clipboard has become a silent but important vector for cyber attacks. Users knowingly copy sensitive information such as emails, passwords, financial details, crypto wallets, API tokens, AWS credentials, and social security numbers. or confidential data without realizing that this data is momentarily present in the system clipboard. This creates a potential attack surface for malicious apps or untrusted users and initiates unintended data disclosure—a security flaw highlighted in OWASP Top 10 – A04:2021 (Insecure Design).

# Despite growing sophistication levels among cybersecurity tools, clipboard monitoring is a fairly under-explored area. Traditional security software typically does not include real-time clipboard monitoring, identification of sensitive information, or malicious URL filtering. Moreover, the growing popularity of malware that actively scans clipboard content—e.g., clipboard hijackers or crypto stealers—indicates a growing need for proactive clipboard protection.

This project introduces an intelligent real-time clipboard monitoring and alerting system specifically built for the Windows platform.

# 1.2 Aim and Objectives of the Project

# Aim The aim of this project is to develop a Windows-optimized clipboard monitoring tool that enhances security by detecting and notifying sensitive data, unsafe commands, and possibly harmful URLs copied onto the clipboard. The tool also features real-time notification, global hotkey support, and an easy-to-use GUI to enhance security awareness and proactive user activity.

# Objectives

1. **Clipboard Monitoring**: Enforce live, continuous system clipboard monitoring of copied data in a timely way to detect risky or sensitive content.
2. **Sensitive Data Detection**: Identify and highlight sensitive data such as passwords, credit card numbers, email addresses, phone numbers, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials),and private keys using regular expressions and heuristics.
3. **Malicious Command & URL Detection**: Identify suspect shell commands as potentially unsafe or risky or phishing URLs by combining pattern analysis and Google Safe Browsing API integration.
4. **Password Breach Check**: Integrate Have I Been Pwned (HIBP) API to identify whether passwords copied to clipboard have been part of known breaches.
5. **Security Reminder System**: Show reminder notices to users that can be customized regarding clearing clipboard after copying sensitive information to avoid accidental data exposure.
6. **Intrusion Detection & Cloud Sync Prevention**: Detect and attempt to prevent unauthorized access to the clipboard and sync sensitive data by cloud clipboard services.
7. **Global Hotkey Integration**: Offer quick access through global hotkeys for clipboard check, monitoring toggle, and clearing clipboard data.
8. **GUI Development**: Develop and deploy a graphical user interface with Tkinter that has light and dark themes for simple usage and accessibility.
9. **Configuration & Persistence**: Support long-term settings with config.json for API keys, whitelisted domains, size for clipboard history, and user options.
10. **System Integration**: Provide autostart capabilities on Windows using registry changes for unobtrusive background operation.

# 1.3 Scope of the Project

# The Clipboard Security Tool is a light, user-friendly, and real-time clipboard monitoring tool tuned to run optimally on Windows-based systems. Its focus is on raising user awareness of security and avoiding unintentional data leakage through the detection and response to possibly malicious or sensitive material copied to the clipboard.

# Security-Focused Clipboard Monitoring: The project provides clipboard data monitoring in real-time to identify patterns of sensitive information and potentially harmful content such as malicious shell commands or phishing links.

# Windows-Specific Optimization: The software is Windows-specifically optimized, employing APIs such as win32clipboard and registry editing (winreg) for access to the clipboard, system integration, and cloud syncing prevention.

# No External Data Dependency: Detection mechanisms are content analysis only (regex and heuristics), minimizing the use of DNS or WHOIS lookups and therefore latency and increasing performance.

# API Integration for Enhanced Security: Integrates the Google Safe Browsing API to detect malicious URLs and the Have I Been Pwned API for leaked password checks, supplying up-to-date threat intelligence.

# Real-Time Notifications and Reminders: Users are shown timely desktop notifications for copying any risky or sensitive information, as well as reminder notifications to clear clipboard data after a user-defined time interval.

# Hotkey Accessibility and GUI Interface: The inclusion of user-definable global hotkeys and multi-tab Tkinter GUI (with light/dark theme support) allows the software to be accessible and easy to use for novices as well as professionals.

# Security Awareness Alignment with OWASP: The utility adheres to OWASP Top 10 (A04:2021 – Insecure Design) by addressing risks of unintended data exposure ahead of time in a design-focused way.

# Extensible and Scalable Design: The design is scalable and extensible, making it possible to add potential future features such as Linux/Mac support, browser extensions, or cloud-based connections.

5

**Chapter 2: LITERATURE SURVEY**

Wieczorek et al. [1] analyzed clipboard attacks that spoof copied banking data and self-install via registry manipulation without administrative privileges. Their research demonstrated how clipboard data is hijacked by malware and used simple C++ libraries to manipulate it in real-time. A proof-of-concept system substituted bank account numbers copied into the clipboard with attacker-controlled values. Malware persistence techniques via Windows registry changes were also discussed in the paper. This research emphasizes the importance of clipboard monitoring and explains how normal user behavior is being exploited without visible system cues.

Mohamed et al. [2] researched remote clipboard data attacks through shared environments like RDP, VMware, and TeamViewer. They demonstrated that attackers can remotely hijack clipboard data—without installing malware on the victim's system—by injecting malicious DLLs into remote services like rdpclip.exe. The paper proposed a real-time detection approach to distinguish between benign and malicious clipboard sharing. The authors emphasized the importance of detecting API calls like SetClipboardData in an attempt to detect stealthy clipboard hijacks. Their findings expand the scope of clipboard threats beyond local attack vectors.

Chen et al. [3] proposed ClipboardScope, a static analysis method for analyzing clipboard usage of Android apps. They grouped clipboard behavior into four types: spot-on, grand-slam, selective, and cherry-pick—based on the validation of the data and its destination. The tool, analyzing over 2.2 million apps, found that many apps store or transmit clipboard contents without user authorization, which is privacy invasive. The study identified security loopholes in Android's clipboard access control and demanded stricter policies. This bulk analysis offers actionable knowledge of mobile clipboard exploitation on a systemic level.

Nedeljković et al. [4] presented web application vulnerabilities according to the OWASP Top 10 framework to raise awareness and guide developers to create more secure applications. The article provides real-world examples of attacks like SQL injection, XSS, and insecure deserialization. It explains each OWASP threat and recommends some particular countermeasures such as input validation, encryption, and improved logging. The authors highlight that there are various weaknesses that remain despite awareness due to improper development practices or lack of competence. The study identifies that application of OWASP guidelines and ethical hacking methods with consistency can cut security threats and stop critical breaches.

Gerbet et al. [5] scrutinized the security design of Google Safe Browsing and found them vulnerable to denial-of-service attacks. They illustrated how attackers may utilize digest collisions and false positives to overwhelm the service with requests. Their evidence demonstrated that attackers can manipulate 32-bit prefix hashes to yield false positives, leading to duplicate server requests. The study gives "false positive flooding" and "boomerang" attacks that undermine client-server efficiency. Recommendations include elongating digest size and hash randomization.

Pal et al. [6] proposed "Might I Get Pwned" (MIGP), a sophisticated credential checking service that is capable of detecting password variants vulnerable to tweaking attacks. Unlike current services like HIBP that only check for exact matches, MIGP offers similarity-based detection without ever compromising user privacy. The system uses generative models for password variant detection on client and server sides. Their use with Cloudflare validated its scalability and resistance to breach extraction attacks. This paper makes a significant contribution to credential reuse prevention.

Koli et al. [7] introduced CHPDA, Context-Aware Hybrid Pattern Detection Algorithm for sensitive data detection. The architecture merges regex with named entity recognition (NER) to provide improved accuracy in the detection of PII and PHI. Google's RE2 engine was the best in terms of speed and scale as found by their benchmarking. CHPDA provides a boost to exact and fuzzy matching with context filters and scoring logic. The hybrid model has less false positives and is suitable for large-scale classification and regulation adherence.

Rahman et al. [8] evaluated secret discovery in source code using Large Language Models (LLMs) and proposed a hybrid approach combining regex with LLM-based classification. Their approach curbed the runaway false-positive rate in conventional tools by using LLMs like LLaMA-3.1 to pre-filter false leads. With a high F1-score of 0.9852, the approach showed good precision and speed. The work is amenable to scalable implementation in secure development pipelines and data loss prevention platforms. This approach enhances the utility of AI in cybersecurity contexts.

Pathak et al. [9] proposed a cryptographic clipboard security method to prevent data theft while copying and pasting. Their solution encrypts clipboard data at the time of copying and decrypts it on paste, hence defeating clipboard hijacking attacks. Authors pointed out vulnerabilities in the Windows clipboard infrastructure and promoted client-side encryption as a proactive remedy. Their approach confirms the effectiveness of cryptographic measures in inter-process data sharing security. The study demonstrates how clipboard security can be integrated into more robust host-level protection mechanisms.

Zhang and Du [10] conducted a systematic assessment of attacks against the Android clipboard since it is world-readable with no permission barriers. They classified threats into two categories: data theft and data manipulation (like JavaScript injection). By conducting an analysis of over 20,000 apps, they found that innocent apps inadvertently leak clipboard content and malicious ones exploit it. Their findings showed how abuse of the clipboard can lead to phishing, command injection, and privacy invasion. Their findings underscore the necessity for OS-level boundaries and secure clipboard APIs in mobile platforms.

Ivanov and Yan [11] introduced EthClipper, a clipboard hijacking malware targeting Ethereum hardware wallet users. The malware exploits ClipperCloud, a distributed service offering duplicate crypto addresses mined for visual similarity, leveraging users' confirmation bias when manually checking addresses. EthClipper hijacks clipboard data with deceptive addresses to trick users into validating invalid transactions. The malware is not privileged and is optimized for low latency and high deception success. Experimentation on popular hardware wallets confirmed the attack's feasibility, causing serious alarms among manufacturers, who supported the threat as real and serious.

**2.1.1 Study of Current System**

The clipboard facility in contemporary operating systems allows data to be stored and transferred temporarily among applications by using copy and paste operations. As much as usability and productivity improve with the facility, no built-in security controls exist to protect sensitive information copied onto the clipboard. Currently available systems such as Windows Clipboard Manager and Cloud Clipboard support content synchronization and part history tracking but no monitoring, threat detection, or user warning for security breaches.

There are existing third-party clipboard tools that are typically intended for productivity, such as clipboard history or snippet handling, but not interested in actual-time security tracking, malicious content identification, or data leakage prevention. Moreover, system-level clipboard access within Windows is not protected by default, allowing any application running to access clipboard contents without user permission. This leaves an enormous attack surface, particularly for clipboard hijacking malware or keyloggers.

Some business-class solutions come with Data Loss Prevention (DLP) functionality that includes monitoring the clipboard but are resource-intensive, closed-source, and not often available for personal or academic use. As a result, individual users and small organizations lack lightweight, affordable, and intelligent clipboard protection tools.

**2.1.2 Problem & Weakness of Current System**

The clipboard systems available today have a number of drawbacks and weaknesses:

* No Real-Time Monitoring: Natural OS environments quietly monitor clipboard use for secure data or security issues, thus remaining unaware of real-time security issues.
* No Threat Detection Capability: Today's clipboard managers do not scan copied text for malicious URLs, leaked passwords, or dangerous shell commands.
* Unrestricted Access: Any foreground process has unlimited access to clipboard information without seeking approval, leaving personal data exposed to clipboard hijacking or spyware.
* No Warnings or Notices: No notice or warning is provided to users when sensitive data (e.g., passwords, credit card details) are disclosed or when spyware or other malicious content is copied.
* Absence of Integration with Threat Intelligence APIs: Utilities like Google Safe Browsing or Have I Been Pwned are not leveraged by existing clipboard systems to establish risk in real-time.

These limitations highlight the need for a dedicated, intelligent clipboard security utility that can identify, classify, and respond to suspicious or sensitive clipboard activity in a lightweight and user-friendly manner.

**2.2 Requirements of New System**

To overcome the limitations of traditional clipboard behavior and improve security, the new system must satisfy the following requirements:

1. Real-Time Monitoring: Continuous monitoring of clipboard data for modifications with low latency.
2. Sensitive Data Detection: Identify sensitive data such as passwords, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials),credit card numbers, email addresses, and private keys using regex patterns.
3. Malicious Content Detection: Detect dangerous commands and suspicious URLs, with support for Google Safe Browsing API integration and whitelist for the avoidance of false-positives.
4. Password Breach Check: Scan whether copied passwords were exposed in a known breach via the Have I Been Pwned API.
5. Security Reminders: Alert users to clear clipboard data after copying sensitive data with configurable timeouts.
6. Intrusion & Sync Prevention: Detect unauthorized clipboard activity and attempt to block cloud sync by Windows registry settings.
7. System Integration: Support global hotkeys and optional auto-run at Windows boot.
8. User Interface: Offer a Tkinter graphical user interface with tabbed UI, theme switching, and non-persistent user settings.
9. Customizability: Allow configuration of API keys, reminder settings, and detection parameters via a config.json file.

**2.3 Feasibility Study**

**2.3.1 Technical Feasibility**

The solution is technically feasible, coded in Python and nicely documented libraries such as Tkinter, pyperclip, winotify, and requests. Access to and observation of the clipboard are handled seamlessly using Windows APIs (pywin32), and compatibility with outside services like Google Safe Browsing and Have I Been Pwned can be achieved through public APIs. The project is optimized for Windows OS with no specialized hardware or external infrastructure required.

**2.3.2 Operational Feasibility**

The software is easy to use, with a clean GUI interface and minimal configuration level. It operates in the background, with hotkey-controlled interaction and real-time notification. It can be deployed by using Python and low-level requirements alone, making it easy to deploy in single-user environments or organizational environments. It is light in size, does not take noticeable resources, and enhances daily security without disrupting daily workflows.

**2.4 Tools/Technology Required**

The materials and inputs involved in the project are as follows:

1. **Python 3.8+** – Primary language for coding
2. **Tkinter** – GUI library for building application interface
3. **Development Environment** - VS Code / PyCharm – For coding
4. **APIs** –

* **Google Safe Browsing API** – To verify URL reputation
* **Have I Been Pwned API** – To detect leaked passwords (using k-anonymity)

**Hardware Specifications** - Average Windows laptop/computer

**Software Specifications**

**Libraries & Packages**:

* **pyperclip** – Clipboard functionality
* **pynput** – Global hotkey handling
* **winotify** – Windows toast notification
* **pywin32** – Windows clipboard and system access
* **requests** – API functionality (Google Safe Browsing, HIBP)
* **psutil** – Process monitoring
* **json, re, threading** – For config management, regex matching, and multithreading

**Operating System: Windows OS**

**CHAPTER 3: Design: Analysis and Design Methodology**

**3.1 Function of the System**

The Clipboard Security Tool is an intermediate security layer between the system clipboard and the user in real time. It continuously monitors for content copied and performs a series of security checks, reporting actionable alerts using a graphical user interface. The system process can be summarized in the following steps:

**1 Clipboard Monitoring Loop:**

* A background thread continually monitors clipboard changes using Windows API and pyperclip.
* On finding new content (and it is not similar to recent past), it is forwarded for analysis.

1. **Content Analysis Pipeline:**

The text copied is analyzed multiple times for the following

* Sensitive Data Detection: Compares content with regular expressions for passwords, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials), credit cards, emails, phone numbers, etc.
* Malicious Command Detection: Scans for well-known malicious shell or Windows commands (e.g., rm -rf /, format C:).
* URL Detection: Removes any URLs and scans against Google Safe Browsing API for potential threats.
* Password Breach Check: For likely passwords, checks whether they exist in breach databases using the HIBP API.

1. **Security Actions & Logging:**

If threats are found-

* Alerts are shown through Windows notifications (winotify).
* Optional clearing of clipboard (e.g., on encountering a malicious command).
* Logs event to a log file (clipboard\_security.log).

1. **Reminder System:**

Reminds the user to clear clipboard on finding sensitive data (delayed reminder e.g., 10 seconds after).

1. **Intrusion & Sync Control:**

* Tracks clipboard accesses by other processes (when running).
* Temporarily disables Windows clipboard cloud sync by registry changes.

1. **User Interaction via GUI:**

Users can -

* + - Start/stop monitoring
    - Show clipboard history
    - Review recent clipboard information and show risk analysis
    - Set settings (API keys, theme, reminders, whitelist domains)

1. **Global Hotkey Integration:**

Users can call main operations without opening GUI:

* + - Ctrl + Shift + M: Enable/Disable monitoring
    - Ctrl + Shift + C: Clear clipboard
    - Ctrl + Shift + I: Show clipboard content in a popup for inspection

**3.1.1 Use Case Diagram**

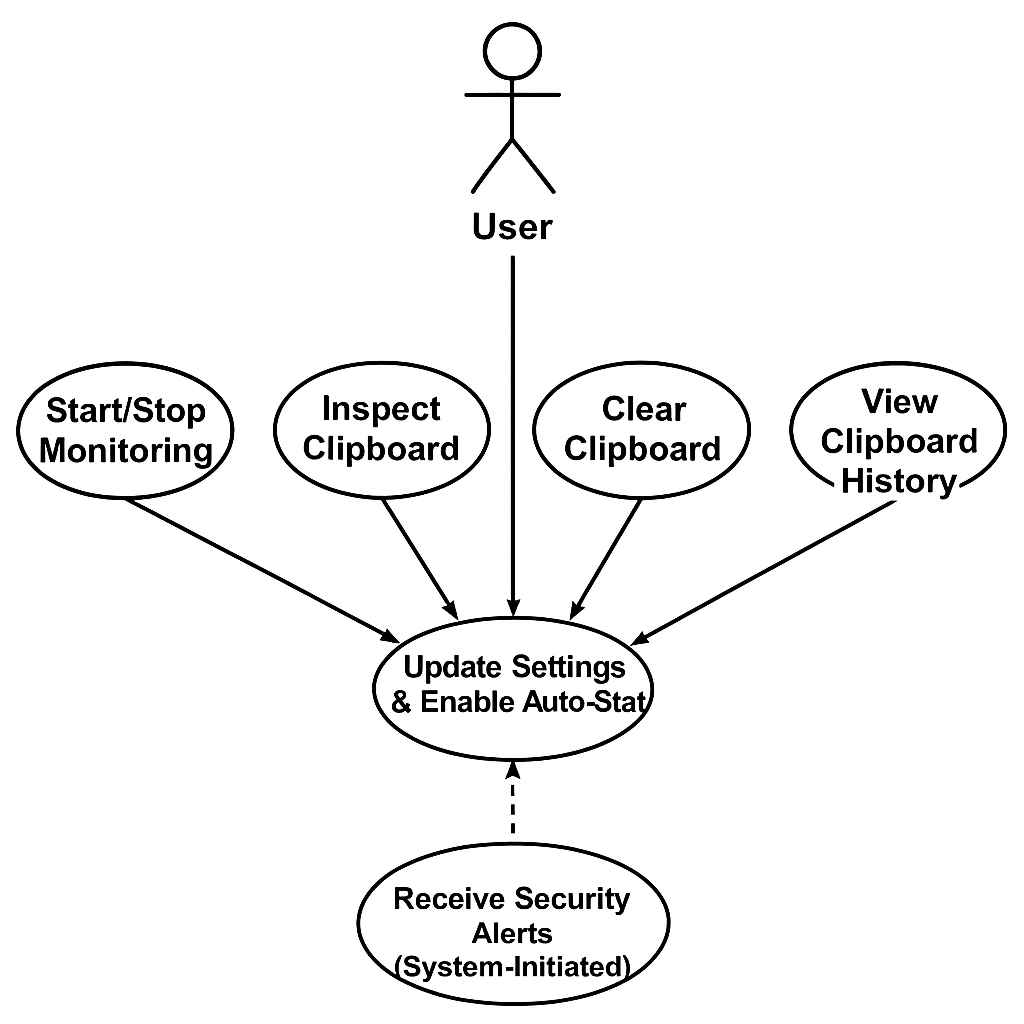
The use case diagram illustrates the interaction between the user and the Clipboard Security Tool. It illustrates the key functionalities offered through the GUI or hotkeys.

Actors:

User: The primary actor who interacts with the application.

Use Cases:

1. Start/Stop Monitoring: User initiates or terminates real-time clipboard monitoring.
2. View Clipboard History: User shows previously copied information in a scrollable history window.
3. Inspect Clipboard Content: User shows a detailed analysis window showing classification and alerts.
4. Clear Clipboard & History: User manually or via hotkey clears the clipboard contents and history log.
5. Update Settings: User updates API keys, reminder timeouts, theme, or whitelist domains via GUI.
6. Enable Auto-Start: User enables the tool to start on system boot automatically.
7. Receive Security Alerts: System automatically notifies user when sensitive or risky material is detected.



*Figure 3.1: Use case diagram*

**3.1.2 Activity Diagram**

The activity diagram illustrates the flow of clipboard monitoring and analysis from when a user copies data until when the system issues alerts or reminders.

Activity Flow:

1. Start Application
2. Initialize Clipboard Monitoring
3. Clipboard Data Changes? (Decision)

 If No → Continue monitoring

 If Yes → Proceed to next step

1. Analyze Clipboard Content

* Detect sensitive data
* Check for malicious commands
* Scan for URLs

1. Threat Detected? (Decision)

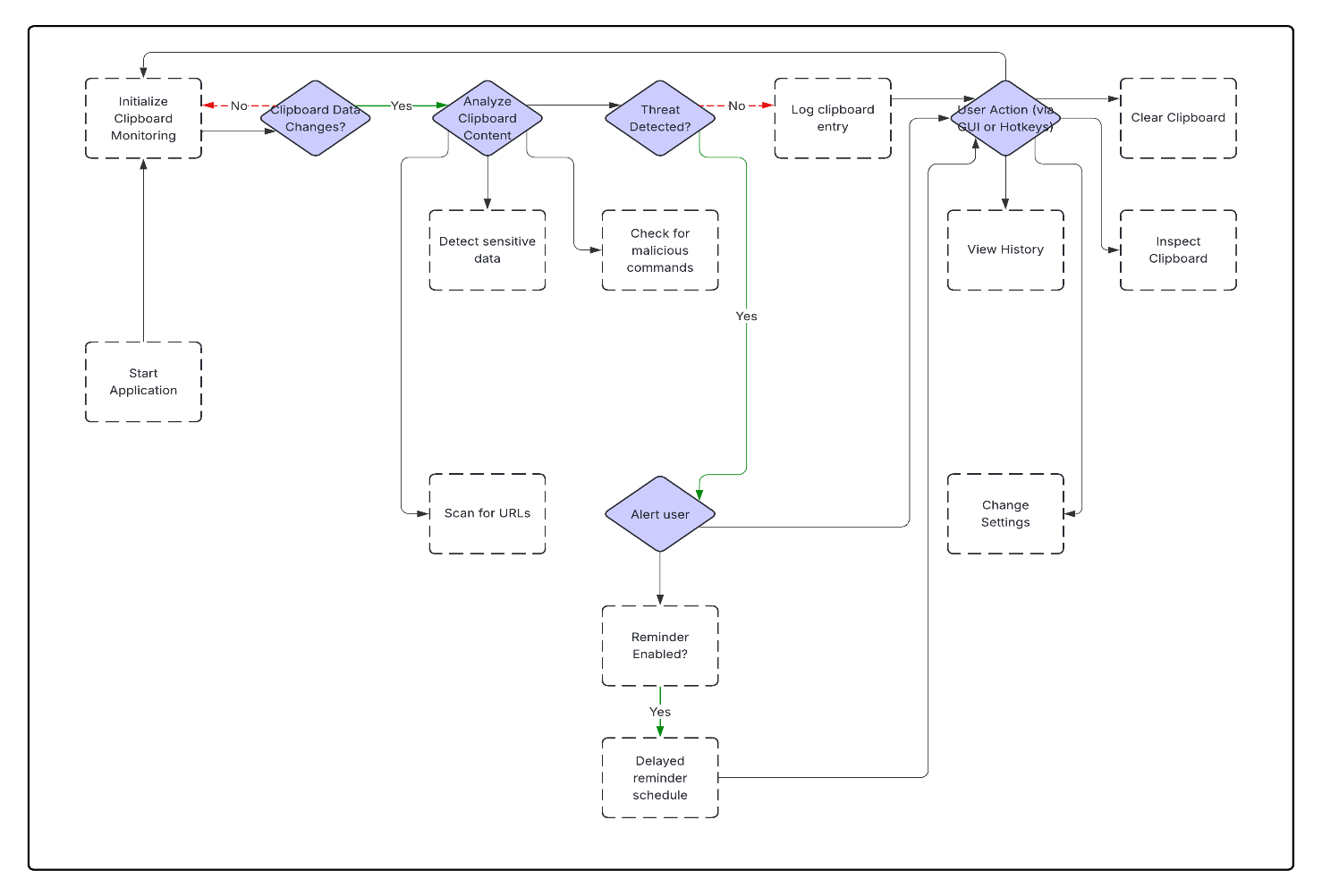
* If No → Log clipboard entry
* If Yes → Alert user

1. Reminder Enabled? (Decision)

* If Yes → Delayed reminder schedule

1. User Action (via GUI or Hotkeys)

* Clear Clipboard
* View History
* Inspect Clipboard
* Change Settings
* Continue Monitoring

****

*Figure 3.2: Activity diagram*

**3.1.3 Sequence Diagram**

Sequence diagram illustrates interaction between the user, system components, and background processes involved in clipboard monitoring and threat detection. The diagram shows how a clipboard event triggers a series of detection and alarm processes within the system.

Actors & Components:

* User
* GUI Interface
* Clipboard Monitor
* Detection Engine
* API Services (Google Safe Browsing, HIBP)

Sequence Flow:

1. User copies data
2. Clipboard Monitor detects modification
3. Detection Engine scans data

* Regex filter for sensitive data
* URL scan
* Command identification
* HIBP password verify

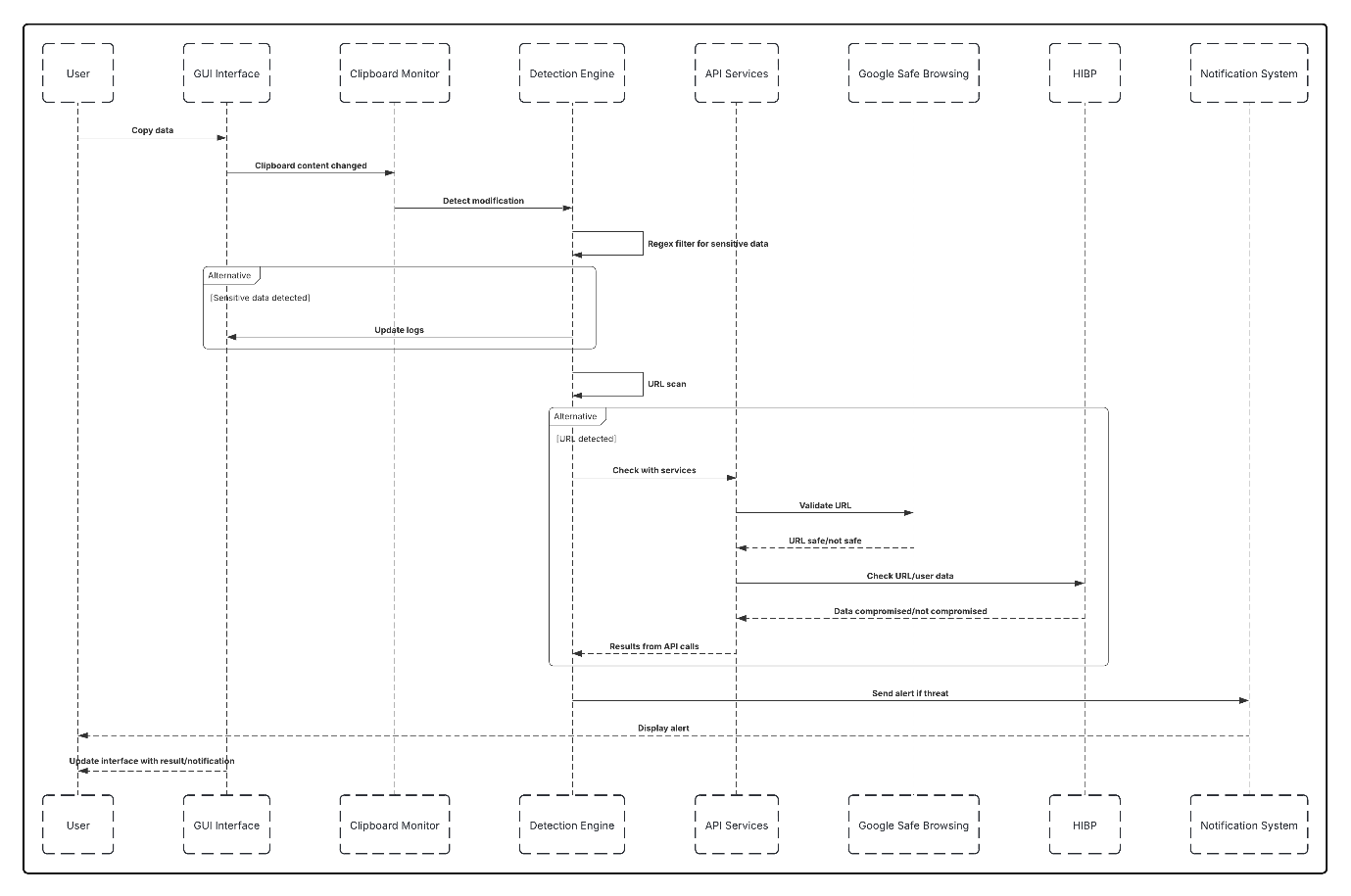
1. API calls (if necessary)

* Google Safe Browsing → URL status
* HIBP API → Password breach status

1. Detection Engine receives results
2. Notification System provides alerts

* Displays security alert
* Sends reminder optionally

1. GUI updates with new content or logs



*Figure 3.3: Sequence diagram*

**3.2 Data Modelling**

**3.2.1 Class diagram**

The class diagram depicts the underlying components of the Clipboard Security Tool, their attributes, methods, and relationships. The organization stresses modularity in order to ease the autonomy of detection logic, UI interaction, and configuration control.

**Main Classes:**

**Clipboard Monitor :**

Attributes:

* is\_monitoring: Boolean
* history: List

Methods:

* start\_monitoring()
* stop\_monitoring()
* log check\_clipboard\_change()
* \_history()

**Detection Engine**

Attributes:

* regex\_patterns: Dict
* whitelist\_domains: List

Methods:

* + detect\_sensitive\_data(text)
  + check\_malicious\_commands(text)
  + scan\_url\_with\_gsb(url)
  + check\_password\_hibp(password)

**ReminderSystem**

Attributes**:**

* + enabled: Boolean
  + delay: Int

Methods:

* + schedule\_reminder()
  + show\_reminder\_notification()

**Notifier**

Methods**:**

* + send\_toast\_notification(message, level)
  + log\_alert(message)

**SettingsManager**

Attributes:

* + config\_file: Path
  + settings: Dict

Methods:

* + load\_config()
  + save\_config()
  + update\_setting(key, value)

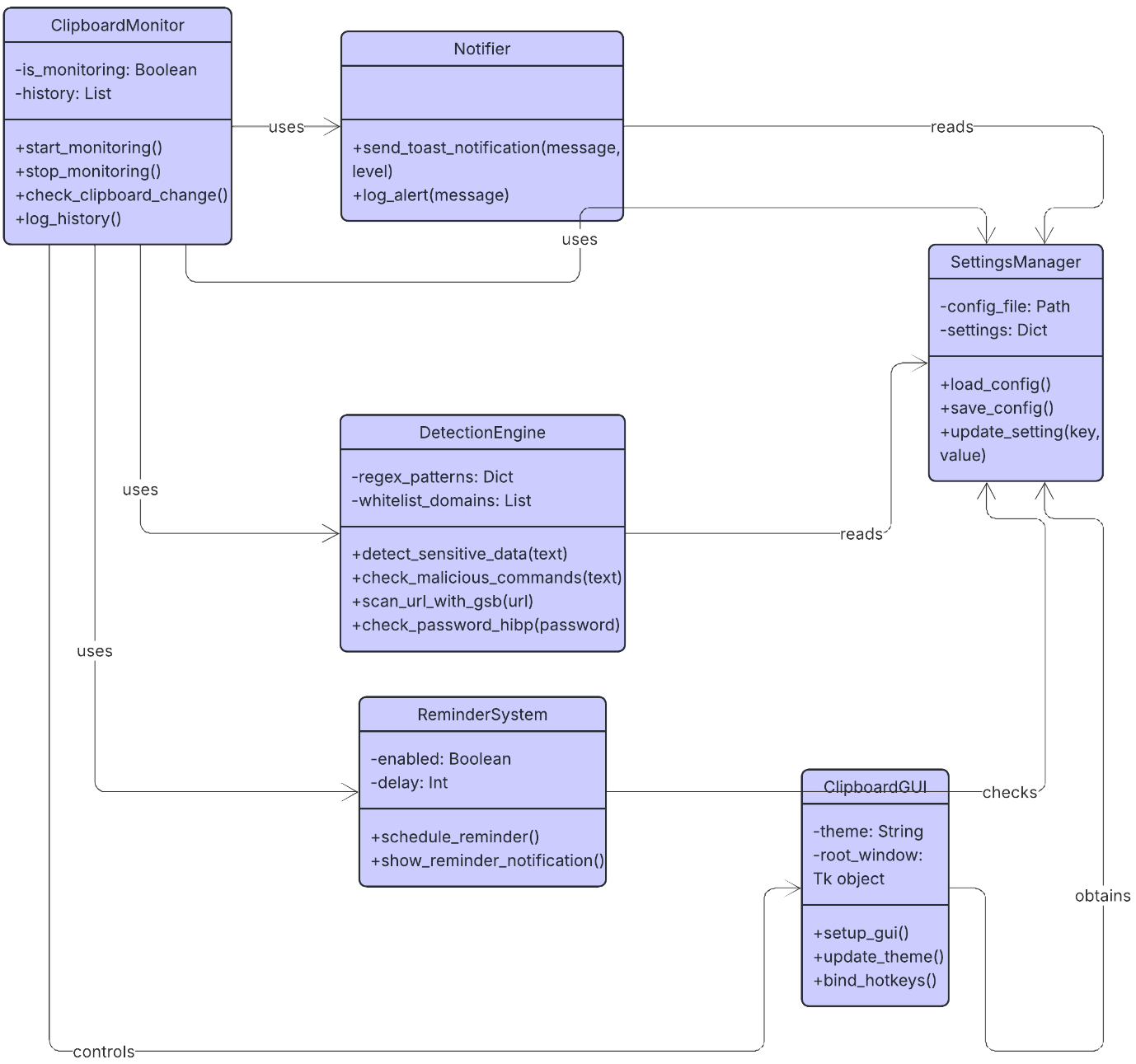
**Clipboard GUI**

Attributes:

* + theme: String
  + root\_window: Tk object

Methods:

* + setup\_gui()
  + update\_theme()
  + bind\_hotkeys()



*Figure 3.4: Class diagram*

**CHAPTER 4: IMPLEMENTATION**

This chapter explains the design and development of the Clipboard Security Tool, a live-monitoring application developed to enhance clipboard-related security on Windows systems. The overall aim is to detect and respond to sensitive data, malicious commands, and insecure URLs copied to the clipboard, besides sending actionable notifications and user engagement through a graphical interface.

Implementation is divided into two big components: the background monitor engine, responsible for content detection and API communication, and the graphical user interface (GUI), implemented with Tkinter, for allowing users to interact with the system, view alerts, set settings, and manage clipboard history. The program also has integration with external services like Google Safe Browsing and Have I Been Pwned APIs for further threat detection and operates as a light background service with optional global hotkey support and system integration features like autostart.

**4.1 Implementation Environment**

The Clipboard Security Tool was developed in Python 3.8+ under a Windows operating system. The coding and testing were carried out on a standard laptop with Windows 11 installed on it with the following environment specifications:

**Hardware Configuration:**

* **Processor:** AMD Ryzen 9
* **RAM:** 16 GB
* **Storage:** 1 TB SSD

**Software Configuration:**

* **Operating System:** Windows 10/11
* **Development Tools:**
  + Visual Studio Code (for code development)
  + Command Prompt / PowerShell (for execution and testing)
* **Python Libraries Used:**
  + pyperclip, pywin32, winotify, pynput, psutil, requests, json, threading, tkinter
* **External APIs:**
  + **Google Safe Browsing API** – for URL threat detection
  + **Have I Been Pwned (HIBP) API** – for password breach checking

**4.2 Coding Standards**

Development of the Clipboard Security Tool followed Python's PEP 8 guidelines for coding to ensure readability, consistency, and maintainability in the overall codebase. The tool was developed with modular design, and detection logic, GUI components, API interfaces, and configuration management were separated into separate classes and functions.

Principal Coding Practices Followed:

* Code Modularity: Functional separation was maintained between monitoring logic, detection engine, UI, notification system, and settings handler.
* Descriptive Naming Conventions: Variables, function, and class names were assigned clear names (e.g., start\_monitoring, detect\_sensitive\_data) in order to take full advantage of self-documentation.
* Comments and Documentation: Each notable function and class contains inline comments and docstrings explaining its usage and purpose.
* Error Handling: Try-except blocks were utilized to handle run-time exceptions within clipboard operation, API, and file I/O to avoid system crashes.
* Configuration Management: User configuration (e.g., API keys, looks, hotkey setup) is stored in an always-present config.json file and updated neatly by a dedicated config handler.
* Hotkey Binding and GUI Responsiveness: Event listeners (pynput) were integrated safely into the GUI without locking the main Tkinter event loop through the use of threading.
* Security Considerations: Clipboard data is not stored persistently to avoid accidental data exposure. Only non-sensitive logs (e.g., detection events) are stored in local files.

**4.3 Laboratory Setup**

This part explains the programming libraries, tools, external APIs, and technologies used in developing the Clipboard Security Tool, along with sample screenshots and key code components.

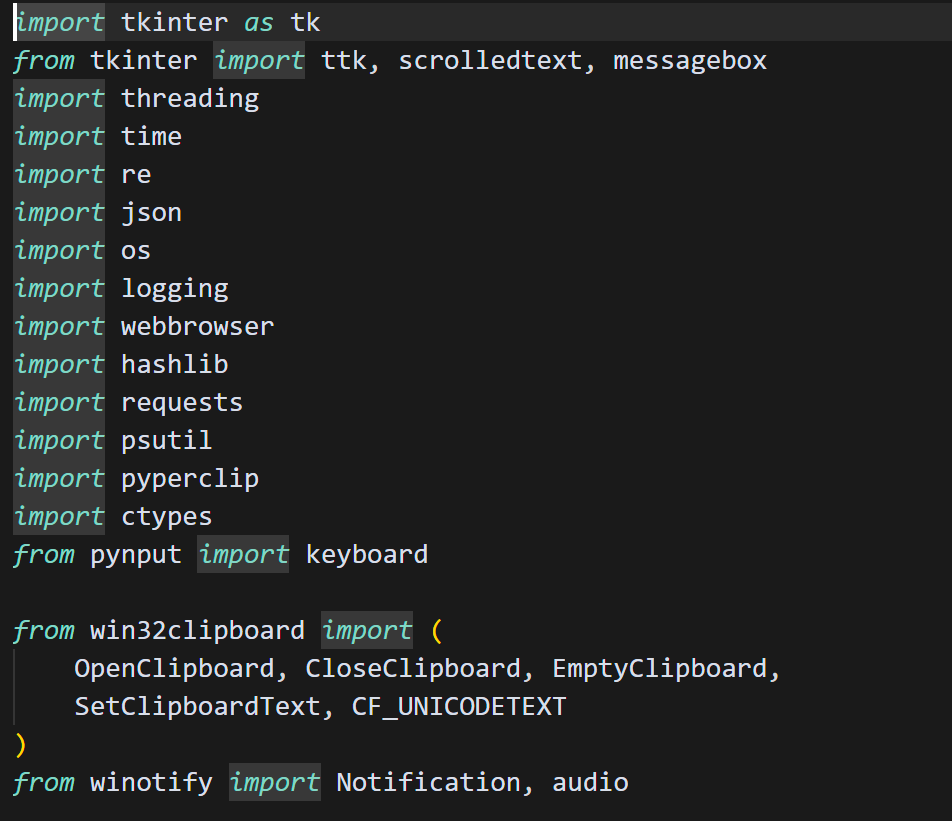
**Development Environment**

* Python 3.8+ – Primary language used for the backend function, GUI layout, and API requests.
* Visual Studio Code – IDE used for writing code and debugging.
* Windows 11 OS – Development and testing platform.

**Libraries and Packages**

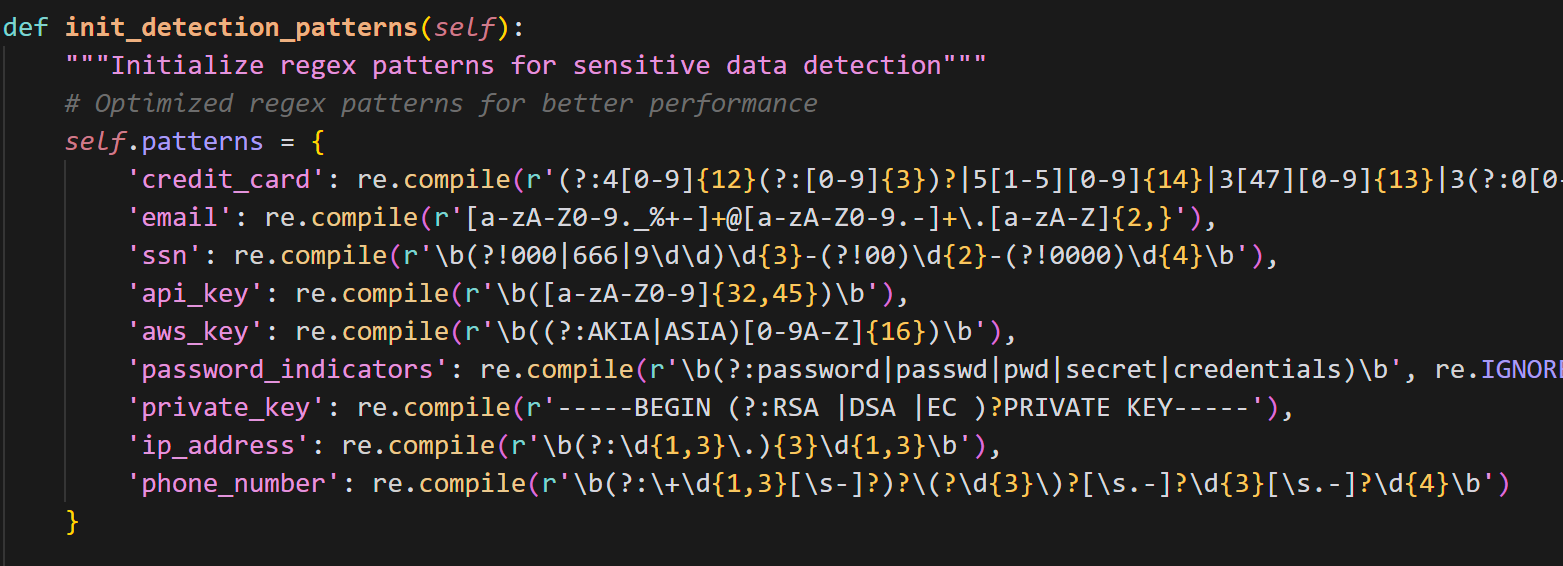
The utility relies on several Python libraries to handle clipboard access, user interface, detection, and notifications.

* pyperclip – Reading and writing the clipboard.
* pywin32 – Windows-specific features like clipboard access and registry modification.
* pynput – Used to implement global hotkeys for immediate action.
* winotify – Displays native Windows toast notifications.
* tkinter – Used to design the GUI with tabbed layout and theme switching.
* requests – For Google Safe Browsing and HIBP API communication.
* psutil, re, json, threading, logging – For detection logic, config management, logging, and multithreading.



*Screenshot 4.1: List of Requirments*

**Regex Patterns:** Regular Expressions (regex) are employed for matching sets of characters in strings. Regex within this project is employed to locate sensitive data within clipboard content through the search for specific formats like emails, passwords, phone numbers, and cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials).



*Screenshot 4.2: Regex patterns*

**External APIs**

Two outside APIs were implemented to enhance detection capabilities:

* Google Safe Browsing API - Used to check if a copied URL has been associated with known threats.
* Have I Been Pwned API (k-anonymity model)- Used to determine if copied passwords have appeared in known recorded data breaches.



*Screenshot 4.3: API Key*

**System-Level Features**

These features allow deeper integration with the operating system:

* Registry Access using pywin32

Allow "Start with Windows" option and turn off cloud clipboard sync.

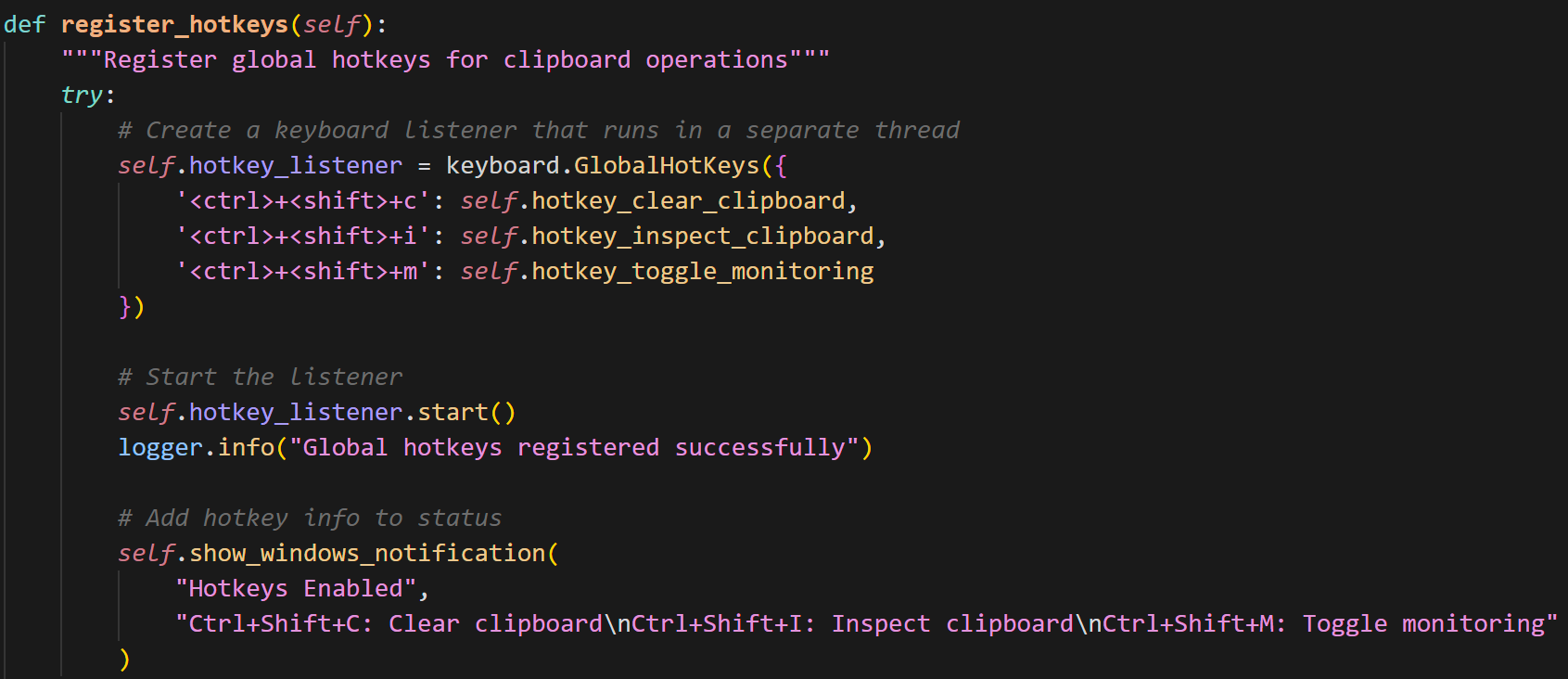
* Global Hotkeys using pynput

Allow quick access to:

Ctrl + Shift + C → Clear clipboard

Ctrl + Shift + I → Inspect clipboard

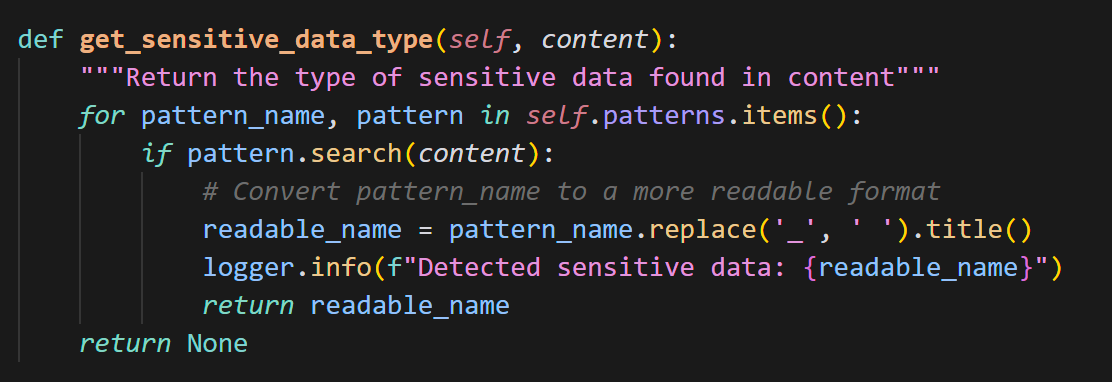
Ctrl + Shift + M → Toggle monitoring



*Screenshot 4.4: Hotkey implementation*

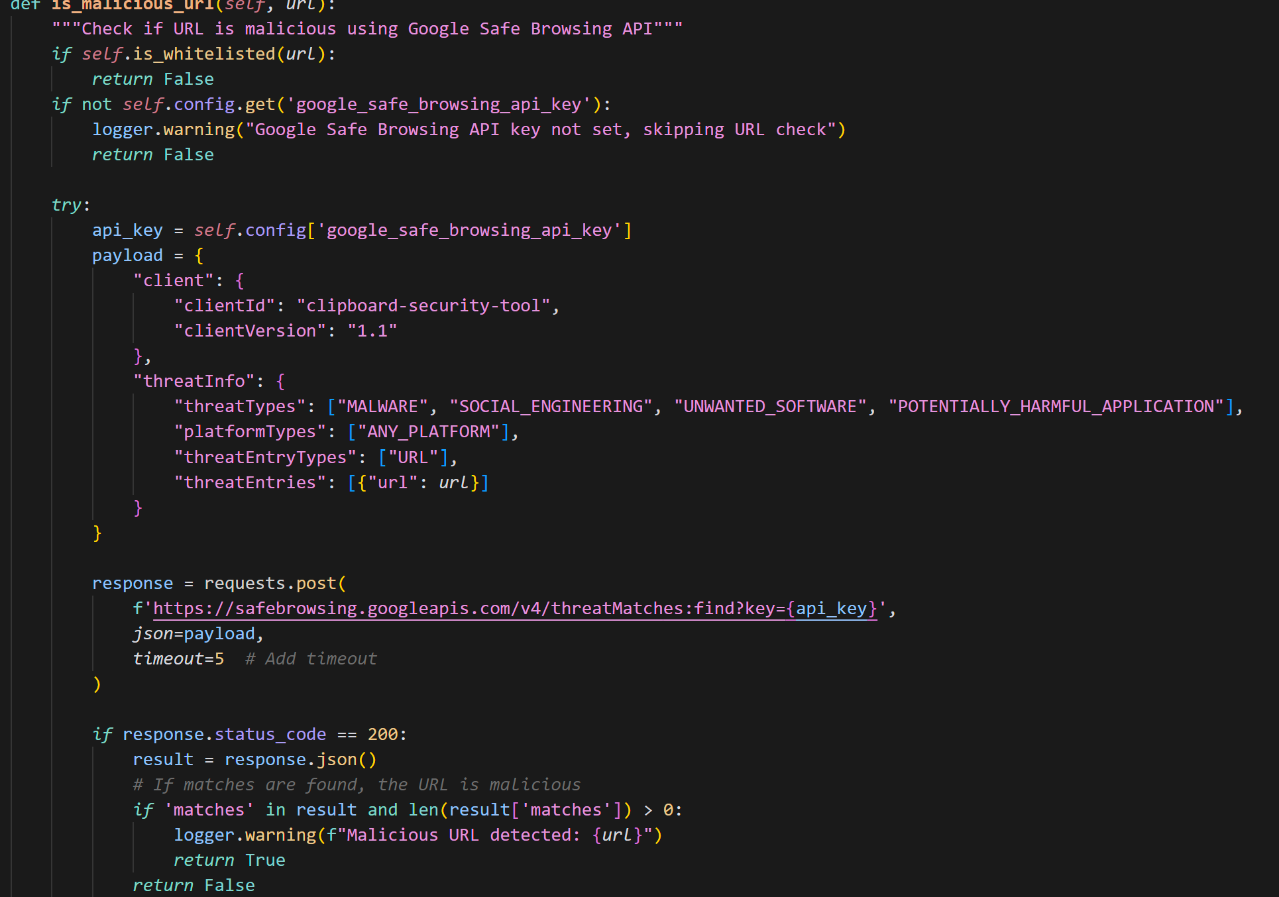
**Sensitive Data Detection**

The method get\_sensitive\_data\_type() (see Figure 4.5) determines the type of sensitive data contained in the clipboard contents. It invokes a dictionary of regular expressions (regexes) to search for things like credit cards, email addresses, passwords, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials), or private keys. When a match is detected, it logs the event and returns the human-readable string name of the sensitive data type. This process is crucial in informing users of potential data exposure and triggering subsequent operations like reminders or content clearing.



*Screenshot 4.5: Sensitive data detection*

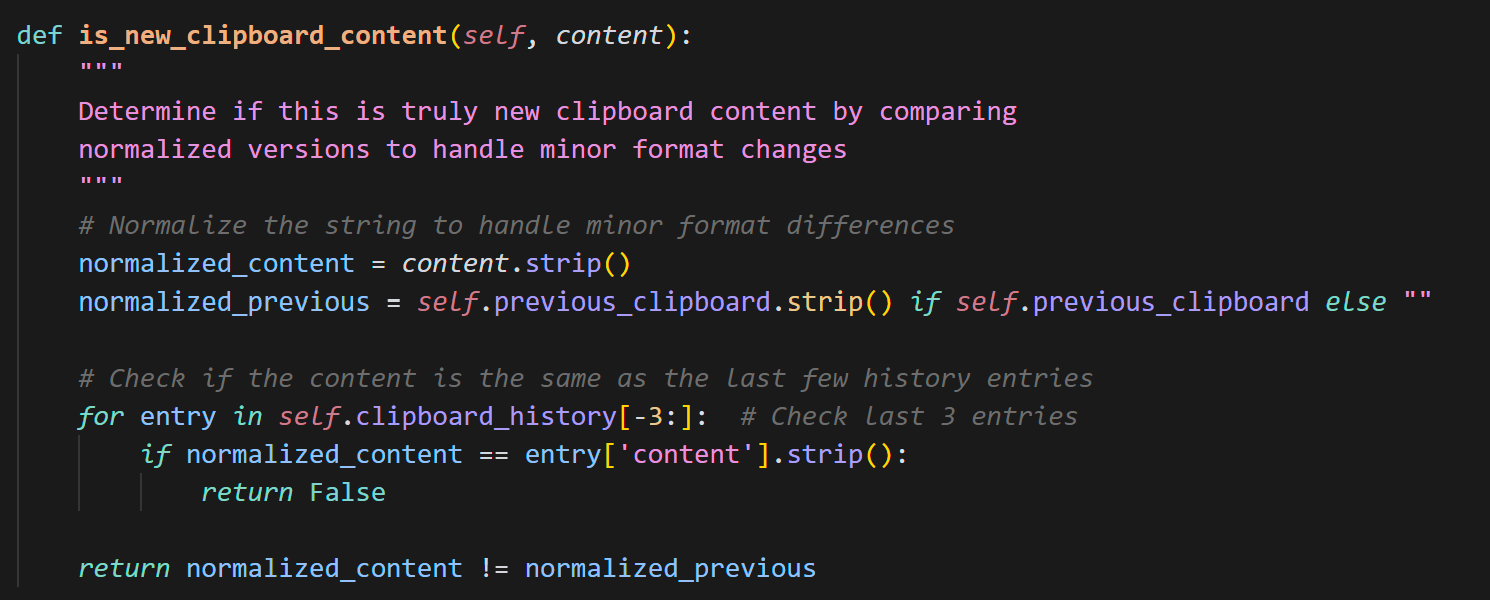
**Malicious URL Detection using Google Safe Browsing API**

The is\_malicious\_url() method (Figure 4.6) checks if any URL copied is harmful using the Google Safe Browsing API. It skips whitelisted domains, verifies the presence of the API key, and passes the URL to Google's threat analysis service. If flagged, the user is notified by the system immediately. This check outside the program offers another stringent security feature to block phishing and malware links to be copied onto the clipboard.

*Screenshot 4.6: URL check using google safe browsing API*

**New Clipboard Content Identification**

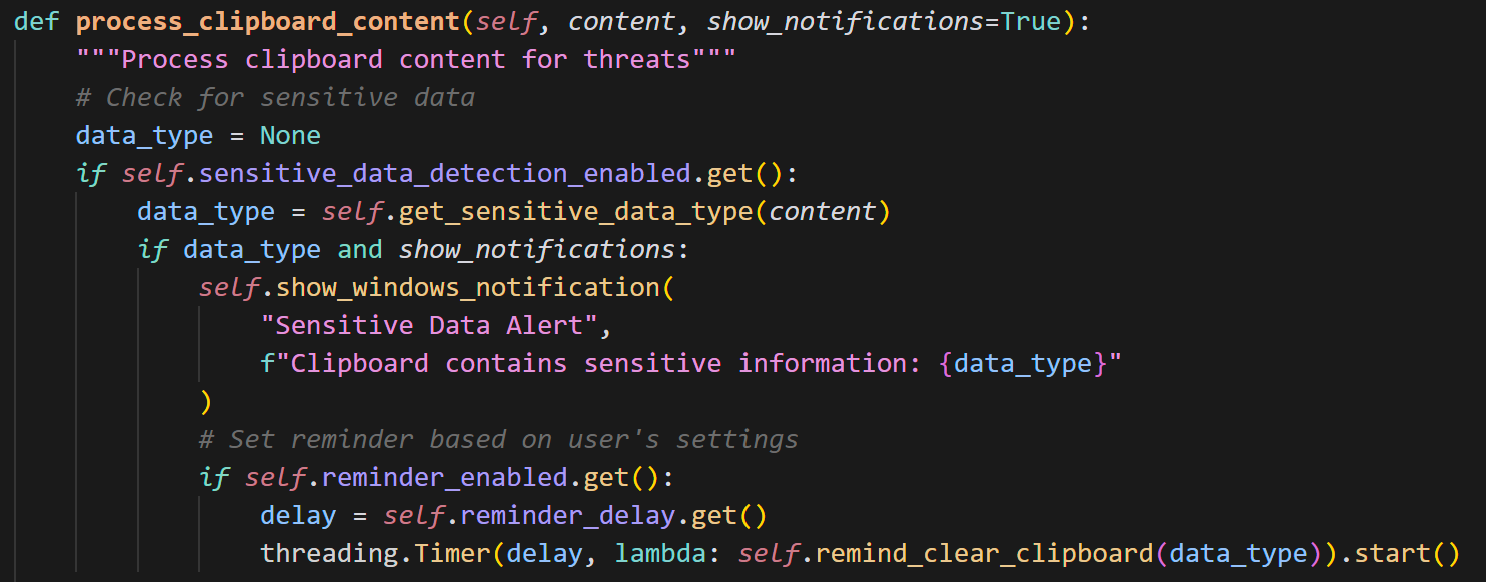
Figure 4.7 shows the is\_new\_clipboard\_content() function, which checks whether the present clipboard content is indeed new. It compares normalized text (cleaned and trimmed of insignificant formatting changes) with the last several clipboard entries. This avoids spurious alerts and only reports meaningful changes, thus optimizing performance and user experience.



*Figure 4.7: check clipboard content is new or not*

**Threat Processing and Notification Trigger**

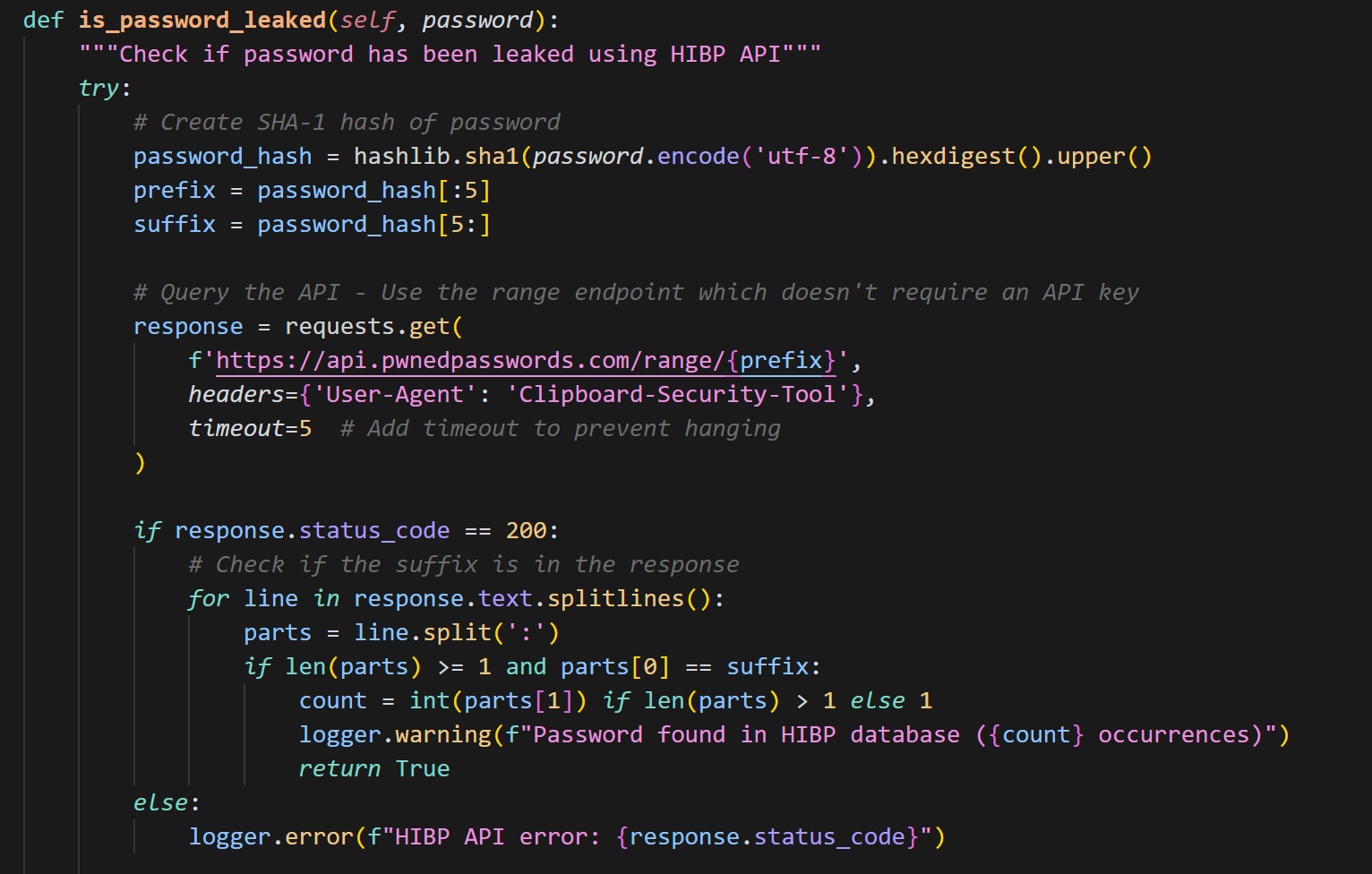
The process\_clipboard\_content() method (Figure 4.8) is the main manager that handles clipboard content for potential threats. It performs checks successively for exposed data, harmful commands, unsafe URLs, and revealed passwords. Based on user settings, it triggers Windows toast notifications to alert the user and initiates scheduled reminders. All the detection functions and user feedback are combined with this consolidated logic.



*Screenshot 4.8: Notification sender*

**Password Breach Detection using HIBP API**

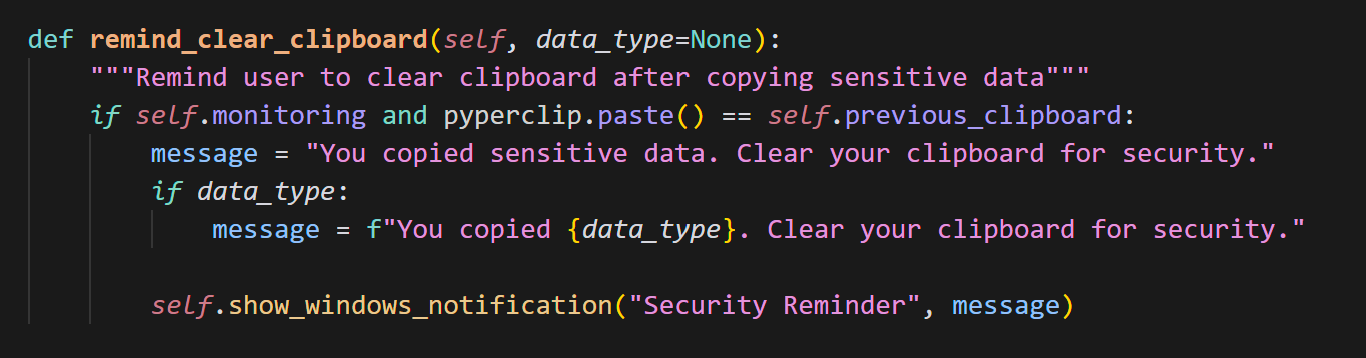
In Figure 4.9, the is\_password\_leaked() function checks if a copied password has emerged in reported data breaches through the Have I Been Pwned (HIBP) API. The function utilizes a privacy-sensitive SHA-1 hash approach to perform a partial match without disclosure of the original password. This feature allows the user to immediately ascertain compromised credentials.



*Screenshot 4.9: Password leak check*

**Clipboard Reminder Mechanism**

As shown in Figure 4.9.1, the remind\_clear\_clipboard() function sends a delayed reminder to the user to clear confidential data from the clipboard. In the event that the user fails to overwrite the clipboard contents within a specified period, a warning is shown. This is particularly useful in preventing passive data leakage when users fail to clear confidential copies upon usage.



*Screenshot 4.9.1: Reminder*

**4.4 Tools and Technology Used**

**Programming Language:**

* Python 3.8+ – Used for developing the monitoring engine, GUI, and API integrations.

**Libraries:**

* pyperclip, pywin32 – Clipboard access and Windows integration
* pynput – Global hotkey handling
* winotify – Windows toast notifications
* requests – API calls (Google Safe Browsing, HIBP)
* tkinter – GUI development
* psutil, json, re, threading, logging – Support utilities

**APIs Used:**

* Google Safe Browsing API – For detecting malicious URLs
* Have I Been Pwned API – For password breach checking

**Development Tools:**

* Visual Studio Code – Code writing and testing
* Windows Terminal / PowerShell – Script execution

**Platform:**

* Windows 10/11 – The tool is designed specifically for Windows systems

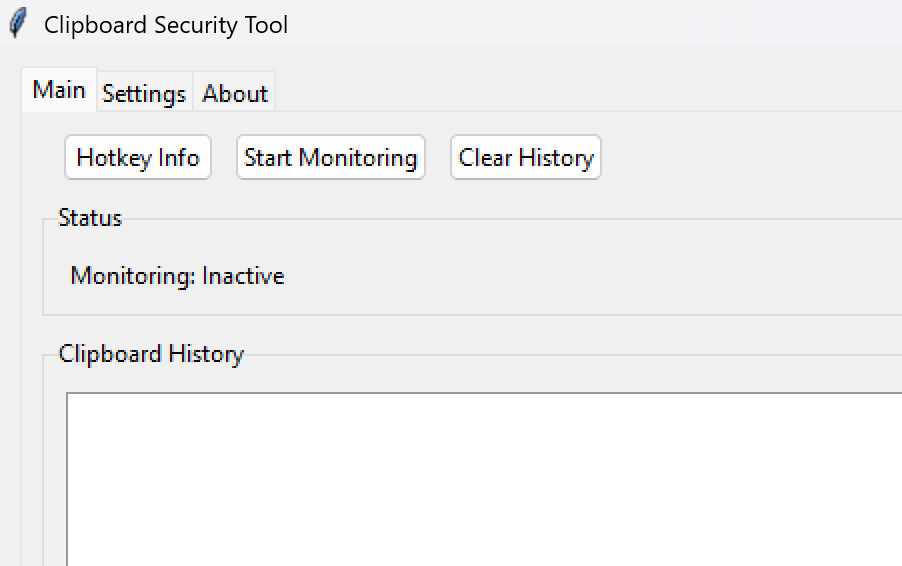
**Chapter 5: Summary of Results and Future Scope**

This section introduces the most critical results and interface elements of the developed application: "An Intelligent Real-Time Clipboard Monitoring and Alerting System for Windows with Malicious URL Detection and Password Breach Analysis."

The following screenshots demonstrate the functionality, user experience, and design of the clipboard security tool.

**1. Main Dashboard Interface**

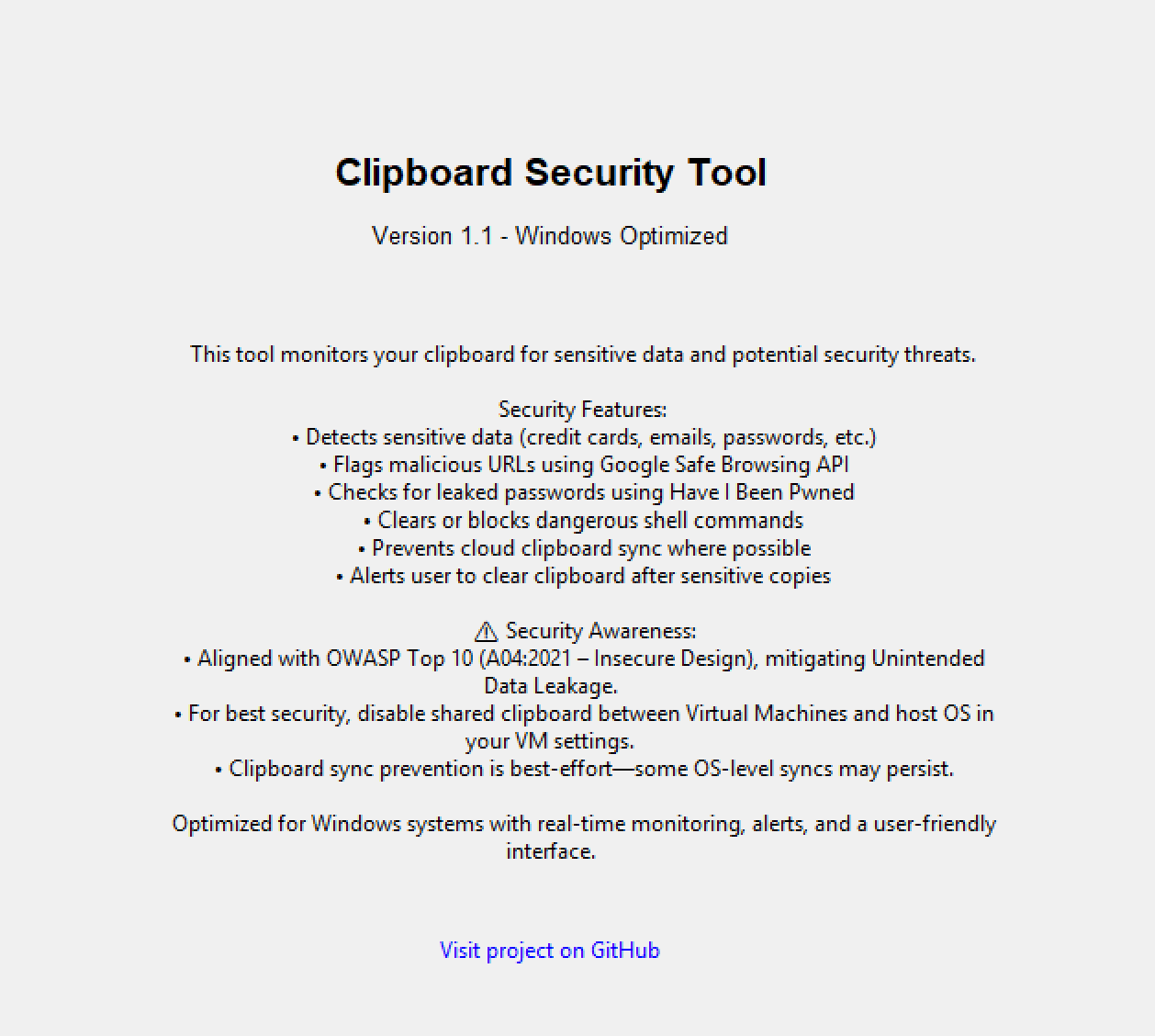
The Main tab is the operational dashboard. It contains start/stop clipboard monitoring, display clipboard history, and access to hotkey information. The screen is clean and minimalistic, with both light and dark color themes.

**

*Screenshot 5.1 Main tab in GUI*

**2. About Tab Overview**

The **About** tab summarizes the tool’s purpose, core features, and version information. It includes a list of security features such as real-time monitoring, password and URL detection, and clipboard hygiene reminders. A GitHub link allows users to access the source code and documentation.

.

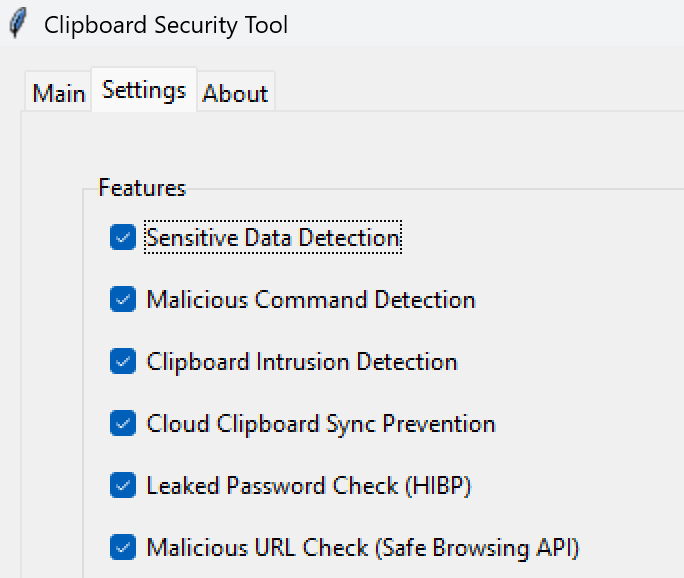
*Screenshot 5.2 – About tab in GUI*

**3. Feature Configuration Panel**

Users can enable or disable various features from the Settings tab. The tool offers modules for:

* Sensitive data detection
* Malicious command detection
* Clipboard intrusion alerts
* Cloud clipboard sync prevention
* Password leak checking via HIBP
* Malicious URL detection using Google Safe Browsing

This modularity enhances user control and customization.



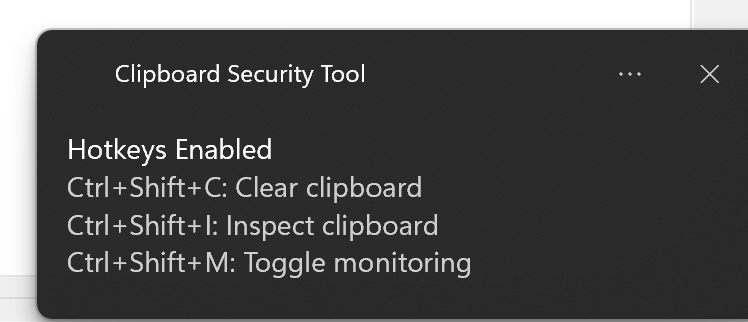
*Screenshot 5.3 Feature selection in settings tab*

**4. Hotkey Notification Popup**

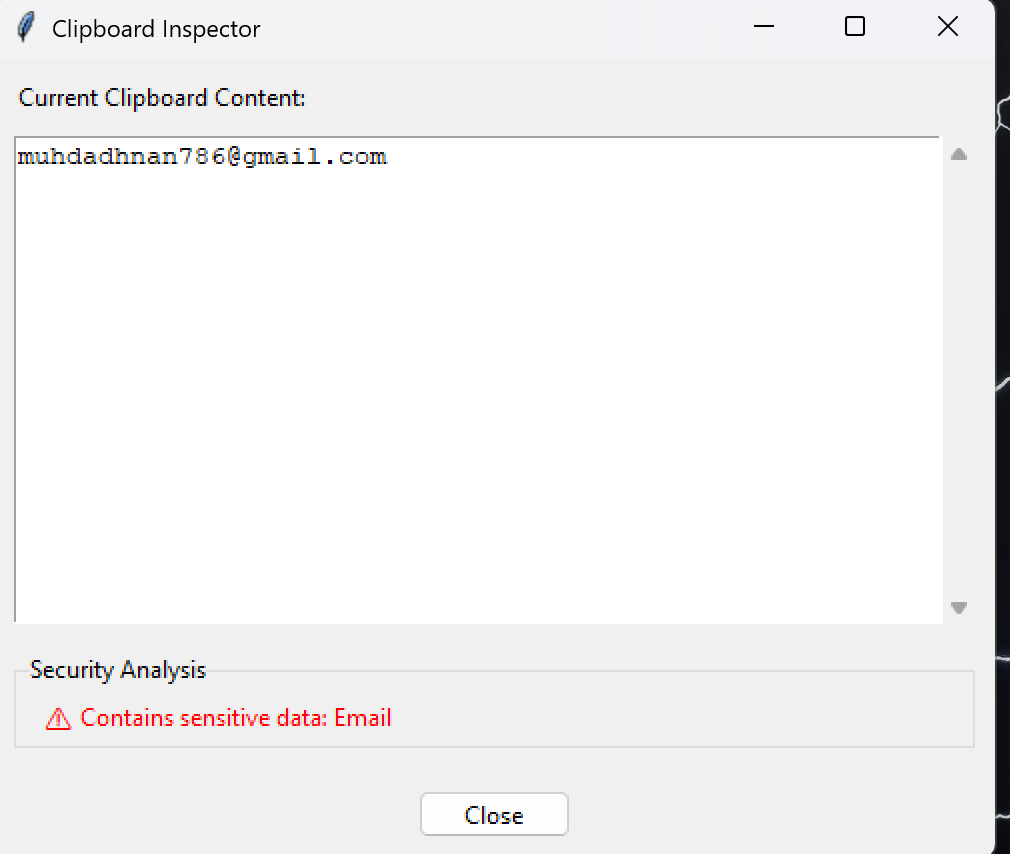
On startup, the tool gives a Windows toast notification listing the global hotkeys available:

* Ctrl + Shift + C: Empty the clipboard
* Ctrl + Shift + I: Examine clipboard data
* Ctrl + Shift + M: Switch monitoring on and off

These hotkeys improve convenience by facilitating rapid security operations.



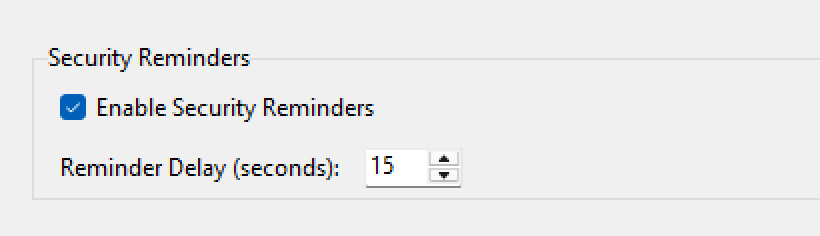
*Screenshot 5.4 Hotkey enabled notification*



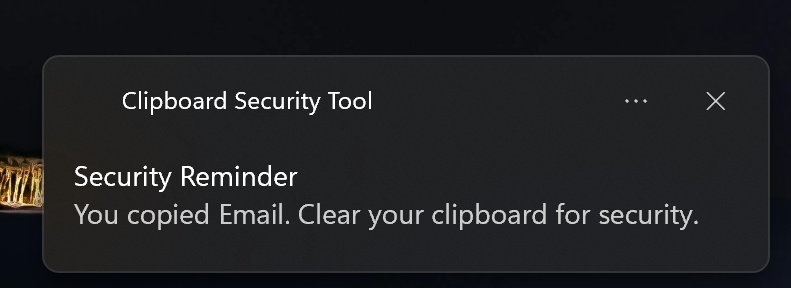
*Screenshot 5.4.1 Ctrl + Shift + I opens the Clipboard Inspector*

**5. Reminder Configuration**

Reminder System is an innovative feature that reminds users to remove confidential data from the clipboard after a specified delay time. The reminder time can be customized (for example, 15 seconds) and is designed to prevent accidental release of data.



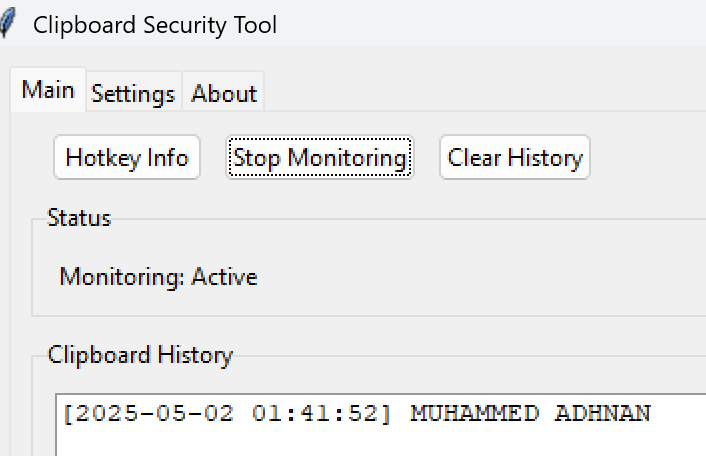
*Screenshot 5.5 Reminder settings*



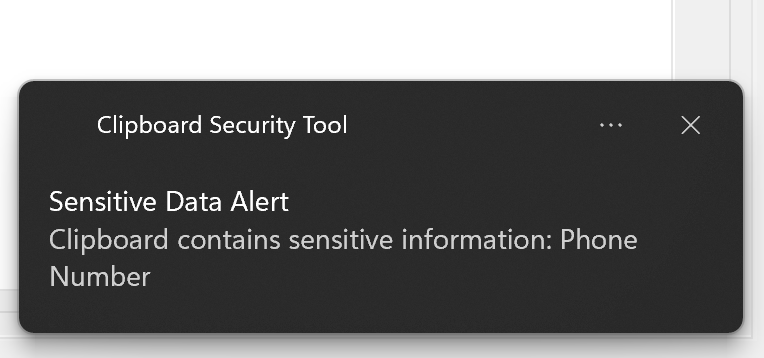
*Screenshot 5.5.1 Reminder pop-up*

**6. Real-Time Monitoring and Alerting in Action**

The utility actively monitors clipboard data in real time and logs security events as they occur. When monitoring is enabled, the status is displayed in the GUI (e.g., "Monitoring: Active"), and newly copied data is monitored in the clipboard history panel. Additionally, if sensitive data such as passwords, emails, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials), or credit card numbers are detected, the system pops up a Windows toast notification to the user. This real-time response system gives immediate alertness to potential data breaches or security threats.

****

*Screenshot 5.6 Monitoring started*

**

*Screenshot 5.6.1 Sensitive data alert*

*\*

**Chapter 6: CONCLUSION**

**6.1 Future Scope**

The Clipboard Security Tool has laid a good foundation for clipboard security. Future improvements will focus on additional cross-platform support, extended security, and scalability for business and enterprise use. The significant future scopes are:

1. **Cross-Platform and Browser Compatibility**: Support macOS and Linux operating systems, and create browser extensions to safeguard clipboard data in web applications.
2. **Cloud and Mobile Integration**: Enable encrypted clipboard sync between trusted devices and develop Android/iOS companion apps for real-time monitoring and alerting.
3. **Advanced Security Features**: Implement encrypted clipboard storage, content expiration timers, and zero-knowledge password verification to improve data confidentiality and meet security standards.
4. **AI-Powered Threat Detection**: Use machine learning and NLP to identify context-aware sensitive data, enabling proactive and intelligent security measures.
5. **Enterprise Integration**: Provide SIEM system, password manager, and threat intelligence feed integration support, and generate compliance reports for GDPR, HIPAA, and PCI DSS.
6. **Domain-Specific Data Handling**: Add domain-specific data detection for medical IDs, financial credentials, and legal document IDs to secure industry-specific information.
7. **Accessibility and Localization**: Add support for assistive technology and include multilingual UI options to expand the tool's accessibility and geographical reach.
8. **Research and Resilience Enhancement**: Examine clipboard hijacking defences, side-channel attacks, and zero-day exploits against clipboard APIs to future-proof the security stance of the tool.

**6.2 Conclusion**

The Clipboard Security Tool provides a major security solution in the form of fixing a highly critical vulnerability in modern computing: clipboard-based exposure and data leakage. By examining real-time clipboard monitoring, the tool has defense against some of the possible threats such as accidentally copying sensitive data, malicious commands, and phishing URLs. The addition of third-party threat intelligence APIs like Google Safe Browsing and Have I Been Pwned gives an added protection level against phishing links and stolen passwords.

By adopting regular expressions, the utility finds sensitive information such as passwords, cryptocurrency wallets (Bitcoin, Ethereum, Solana), API keys (including AWS access credentials), credit cards, and emails, as well as providing users with real-time warnings and notifications to empty their clipboard. The feature of the program to lock out cloud sync of clipboard data and detect malicious access to the clipboard also ensures users' information remains secure against external attacks.

The default Tkinter GUI and hotkey support for worldwide hotkeys deliver convenience and ease of use for both new and old users alike. Support for adjusting the behavior of the system from API keys all the way down to clipboard history size means that the experience can be adapted to meet the needs of the user. By complying with OWASP-defined security requirements, the Clipboard Security Tool also assists in the reduction of the risk of insecure design in modern systems.**References**

[1] M. Wieczorek, M. Troscianko, and A. Szmigiero, “Attacks Using the Clipboard—Methods of Operation and Risk Mitigation,” Sensors, vol. 23, no. 1, Jan. 2023, doi: 10.3390/s23010356.

[2] K. F. Mohamed, N. AbdelBaki, and A. Shosha, “Remote Clipboard Data In-Memory Attacks and Detection,” in CEUR Workshop Proceedings, vol. 3617, pp. 54–71, 2023. [Online]. Available: http://ceur-ws.org/Vol-3617/

[3] Y. Chen, R. Tang, C. Zuo, X. Zhang, L. Xue, X. Luo, and Q. Zhao, “Attention! Your Copied Data is Under Monitoring: A Systematic Study of Clipboard Usage in Android Apps,” in Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (ICSE '24), Lisbon, Portugal, pp. 1525–1537, Apr. 2024, doi: 10.1145/3597503.3623317.

[4] M. Nedeljković, D. Gajić, and D. Ćirović, “Web Application Security: OWASP Top 10 and Real-World Vulnerabilities,” International Journal of Engineering Research and Applications, vol. 11, no. 5, pp. 22–28, 2021.

[5] T. Gerbet, J. L. Lanet, and C. Bidan, “On the (In)security of Google Safe Browsing,” Université Joseph Fourier and INRIA, 2015. [Online]. Available: https://hal.science/hal-01130484

[6] B. Pal, J. Li, R. Zhang, D. Woodruff, and T. Ristenpart, “Might I Get Pwned: A Second Generation Compromised Credential Checking Service,” in 30th USENIX Security Symposium, Aug. 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/pal

[7] L. Koli, S. Jindal, and V. Saini, “CHPDA – Context-Aware Hybrid Pattern Detection Algorithm,” arXiv preprint arXiv:2502.07815, Feb. 2025. [Online]. Available: https://arxiv.org/abs/2502.07815

[8] M. N. Rahman, S. Malik, R. Sun, and P. Liu, “Secret Breach Detection in Source Code with Large Language Models,” arXiv preprint arXiv:2504.18784, Apr. 2025. [Online]. Available: https://arxiv.org/abs/2504.18784

[9] G. Pathak and G. K. Tak, “Implementation of Clipboard Security using Cryptographic Techniques,” International Journal of Computer Applications, vol. 86, no. 6, pp. 20–24, Jan. 2014, doi: 10.5120/15054-3366.

[10] X. Zhang and W. Du, “Attacks on Android Clipboard,” in Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), LNCS 8550, pp. 172–191, Springer, 2014, doi: 10.1007/978-3-319-08509-8\_9.

[11] S. Ivanov and T. Yan, “EthClipper: Hijacking Clipboard for Targeted Ethereum Address Substitution,” arXiv preprint arXiv:2304.12123, Apr. 2023. [Online]. Available: https://arxiv.org/abs/2304.12123