Keylogger and Analysis Tools

Charles albert Kemerley Jr

CYS 6960 Cybersecurity Capstone

6/1/2025

**Introduction**

This technical Document presents the design and implementation of a software-based keylogger intended to study how someone could record and analyze keystrokes and clicks to gather important information about the user. The objective is to understand the operational functions of the keyloggers to capture, store, and exfiltrate keystroke data within modern operating systems. The Keylogger was developed in an environment using Python. Findings illustrate the ease with which such tools can be developed and deployed, underscoring the critical need for robust endpoint protection and user awareness.

**Project Goal**

The goal of designing the Keylogger was to create a keylogger that not only records keystrokes and clicks on a computer or device but also provides a secure and reliable solution. But allowed for the information to be parsed in a way that would be useful in the future for data analysis. However, at the same time, it should not be too burdensome to cause problems with the system in which it is working. The overall design of the Keylogger is straightforward and effective using the Python pynput module. This Keylogger is designed to write the information to the as a way to hide the log somewhere out of the way. The point is to hide the file in plain sight so that it is not discovered. The next part of the Program is to use the Proeccss\_Data executable to parse and put the data in the correct way.

**Project description**

**Keylogger design and setup**

This project involves the development of a keylogger using Python, focusing on the collection of keyboard and mouse event monitoring with timestamped logging. This core functionality was designed to use of three essential Python modules for this project: pynput.keyboard, pynput.mouse, and datetime. The pynput.keyboard module enables the Program to record keyboard events, such as key presses and releases. Using pynput.mouse, the Keylogger is responsible for detecting mouse actions, including clicks and movement. However, the current Keylogger does not worry about mouse click locations. Together, these modules enable the accurate tracking of inputs using the Keyboard and mouse. The datetime module allows the Program to generate timestamps, ensuring that each input combination is recorded with precise context for when it happens. The combination of modules forms the major portion of the keylogging functionality, facilitating both the capture of inputs and event logging with dtaetime stamps.

The Keylogger's logic includes specialized handlers for specific input events. The "on\_tab", "on\_enter", and "on\_punctuation" functions are triggered when the Tab, Enter, or punctuation keys (such as period, comma, exclamation mark, and question mark) are pressed. Each of these functions generates a timestamp, formats it with a comma and newline for clarity, appends it to a global list called keys, and immediately writes it to a log file using the "write\_file" function. These functions are enclosed in try-except blocks to handle potential exceptions in a graceful manner.

The general on\_press function acts as the primary keyboard event handler. It logs all key presses and filters and processes special keys and delegates actions to their respective handlers. If the key is a standard character, it is added to the keys list and logged. If it is a special key, such as Tab or Enter, the corresponding handler is called to ensure that those inputs are timestamped. Punctuation characters such as "." or ";" are also detected and logged through the "on\_punctuation function", this makes the better chance at discovering when password is effective.

Mouse clicks are captured using the on\_click function. When the mouse button is pressed, the function logs the event with a timestamp, which is then written to the log file. Mouse clicks are more complex for the user that is interacting with the data to the logger, enabling more comprehensive monitoring.

The write\_file function is responsible for persisting captured data to a file named log.txt. It cleans up the data by removing quotation marks and handles special characters by converting them into spaces for better readability. After writing, the keys list is cleared to prepare for new input events. The Keylogger does some of the parsing of the information. This process excludes most inputs, such as Ctrl or Shift. This is because they do not provide the necessary information when searching for what is being entered into the system or clicked on. But some of these keys do help in recognizing when a sure thing is happening. For example, when you press the Tab key, it could be that you are indenting a paragraph. This is also used for moving to the input box in the website or application also use. Therefore, the keylogger tab creates a new line. One thing that is interesting to me is the click of the mouse. This may seem unimportant, but it allows you to follow the same line of thought as when someone presses enter or tab. These allow for the ability for you to connect with important information.

In the second phase of the project, a separate data processing script is used to load and parse the log file, extracting key information and exporting it to a CSV format. This enables further data analysis, such as tracking usage patterns over time, correlating input activity with timestamps, or conducting behavioral studies based on input frequency and timing. The combined logging and processing systems create a powerful tool for tracking input and analyzing data.

This Python program provides a graphical interface for analyzing and processing log files generated by the Keylogger. Built using the wxPython library for the GUI and pandas for data manipulation, the application enables users to load a log file, analyze its contents, and export meaningful results to a CSV file.

Upon launching, the application opens a window titled "Log File Analyzer" with a user-friendly layout. At the top, there is an icon, when clicked, which allows users to browse and select a .txt log file. Once a file is selected, clicking the Submit button triggers the "on\_submit" function. This function reads the log file using pandas, treating each line as a record with three columns: "Text" and "Dates and Time." It strips any whitespace from the text, removes empty or null entries, and converts the date strings into datetime objects. Then, it splits the text into individual words and "explodes" them into separate rows, effectively transforming each word into its own record. To focus on more meaningful entries, the Program filters out any words shorter than Three characters. The result is a refined data frame containing only the date, time, and relevant words.

The formatted results are then displayed in a read-only multi-line text control within the GUI, with aligned columns for easy readability. A header row labels each column—Entry, Date, and Time—and rows of filtered data are listed beneath it.

Finally, users can choose to take said text and file of the processed data by clicking the Save to CSV button. "The on\_save" function checks if any data is available to save, prompts the user to choose a location and filename, and then writes the filtered data to a CSV file using pandas. If successful, the user is notified with a confirmation message.

Overall, this tool transforms raw keylogger output into structured, human-readable, and analyzable data, supporting further investigation or reporting.

**Manual for Use and Setup**

This section will cover how to effectively deploy and utilize the Keylogger. Some of this is I will not be able to do this with the code at this time. However, this will go through the process of deploying the Keylogger. We will be doing this by step. First, I will show you the setup of the code and then demonstrate how to deploy it. Then, I will go through how to use the code in the best way possible to achieve the best outcome.

Section 1: Setup

First step: We need to rename the Keylogger from its current name to a more inconspicuous one. It can be anything from a system log to Windows. The name is not as important as if you are trying to hide it in the plan site, so it comes down to what best hides it. Because we will eventually place it in the startup folder inside %APPDATA%, so once you have the name, my name will be Systemdrive.py.

**Step Two:** This step compiles the Keylogger into an executable that can be run on Windows. This step is important because you want to be able to run without errors upon start. The command you use to

pyinstaller --noconsole --onefile Systemdrive.py

This allows for the Python script to be complied with and packaged up to be run as an .exe. Additionally, this enables the Program to run on Windows without requiring the host computer to have Python installed. Since most people do not have it installed.

Section: Deployment:

**First step:** Put Systemdrive.exe in the startup folder this can be done by a move script. Eventually this would be done will a Malicious Link. This will be enabled every time the PC restarts, allowing the program to start automatically. Caution: The Target computers antivirus may stop it from running, this can be gotten around by running it in admin. Also you have to off windows defender or whatever antivirus that you use. This also makes it easier to hide at startup and make it appear as if it is part of the system. Once you have completed this step, data collection will begin. The Keylogger will write any information into a text file. Inside the Keylogger, you can choose the location where you want the text file to be written. This location can be in the same directory as the Program. This is where all of the keystrokes are kept.

**Step two**: This is where the Data process part of the Program is used. Once you have the location of the txt file of the keystrokes, open up the program "process\_data." Once it opens, there are a couple of options. First option at the top of the application will be a browser button. Once you have selected the log.txt located current floder in the flie hit submit. Then, the information in the log file will open up in the box.

**Step Three:** The last step is to press save CSV file. Once you save a file. This can be located anywhere you prefer, allowing you to access the information easily. If you open up the CSV, you will see three columns Date, time, and Words. This is the final information for you to review. Process\_data can be running from anywhere as long as you can point to the location of the text file.

A computer screen shot of a computer

AI-generated content may be incorrect.

Problems during the project

Issue: Improper Key Capture with Pynput

One of the initial challenges encountered was getting the keys to behave as expected when using the Pynput module when some keys were not being recorded correctly or were producing unexpected output. While troubleshooting, especially when handling special keys or key combinations, I found that I was not getting the necessary information. Some of the issues that arose were incorrect keymapping, Inconsistent Output, Keyboard Layout Differences, and Modifier Keys. When it comes to incorrect mapping, key events might return as object types (like Key.space) instead of standard characters, requiring additional logic to convert them into readable text. An issue I did not come across but might become an issue is different keyboard layouts causing the issues. This took me some time to get over. One thing that I came across was the right and left Shift and ctrl keys Instead of generic keys like

pynput.keyboard.Key.shift\_r

pynput.keyboard.Key.shift

Issue: Processing Data

The primary part of the data processing issue was understanding the possible misinterpretation of information. When I was working with the on\_submit function, a few problems could arise, such as if the log file wasn't formatted exactly as the code expects. For example, the Program assumes every line in the file is perfectly structured with clean text and matching timestamps. However, in real-world use, logs might contain messy lines, missing entries, formatting issues, or blank spaces. If that happens, the date parser will fail quietly, and those lines will get dropped without any warning. That means you could lose valuable data without even realizing it. Also, there's a filter in place that removes any words shorter than 5 characters. While that helps clean up noise, it can also filter out important short terms like "fail," "admin," or "exit," which might actually be useful in analysis.

Another thing to watch out for is how the user interface handles things. If someone loads a huge log file, the app tries to format and display all of it at once, which can make the window freeze or lag because it's doing everything on the main thread. There's also no prompt when loading a second fil it just overwrites the first one, which might confuse users who think both files are being analyzed. And if a file is empty or corrupt, the Program doesn't provide a clear message—it just does nothing. Therefore, I will likely need to add checks for file content, make the word-length filter customizable, and possibly utilize threading to prevent large files from locking up the app. That'll make the tool a lot more reliable and user-friendly.

**Future improvements**

Adding the Website:

To increase the effectinvess of the data collected from the browser history, I could add a new column that isolates the base site or domain from full URLs. This process uses the Python urlparse module to extract just the scheme and netloc (e.g., https://www.google.com from https://www.google.com/search?q=wxpython). This function performs the operation by parsing each URL and formatting it into a clean, consistent domain string. This additional field would be crucial for identifying and categorizing where the user is visiting from the browser by site, rather than tracking individual pages, which can vary significantly in structure. This enhancement would allow for retrieving the browsing history with the entries for possible passwords. This feature could enhance the effectiveness of user analysis, such as tracking the frequency of a user's visits to a specific website. It could also gather data from known login portals or correlate domain visits with captured keylogging data for deeper behavioral insights.

Possibly integrating into a GUI-based program using wxPython, this site parsing could also included as part of the data processing function once a file is submitted for analysis. While the current program focuses on text-based logs and keypress content, the same concept can be extended to browser data. Adding websites to the Keylogger, allows users to cross-reference visited sites with specific keywords or behaviors. This will allow for a more informative and security-focused report. Saving this enriched dataset to a CSV ensures portability and future analysis opportunities using external tools or scripts.

Improvement: Autorun and move Program to the startup folder:

To enable a program to automatically move itself to the Windows Startup folder upon unzipping, a common method is to include a batch file (.bat) inside the ZIP archive. This batch file contains commands that, when run by the user, copy the desired Program (e.g., a .exe or .py file) into the system's Startup directory. The Startup folder path typically is:

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup. When the Program is copied there, it will automatically launch the next time the user logs into Windows.

The process works like this: after the user unzips the archive, they are instructed to run the batch file (often named something like runme.bat or install.bat). This file contains basic commands such as copy, xcopy, or move to transfer the main program file to the Startup folder. This command silently copies Systemdrive.exe into the Startup folder. Once placed there, Windows will automatically execute it during the next system start.

It's essential to note that this process requires the user to click to run the batch file. Windows does not allow scripts inside ZIPs to execute automatically upon extraction due to security restrictions. Therefore, the ZIP file should include clear instructions asking the user to double-click the batch file after unzipping. While this approach is simple and effective for self-deployment in administrative or utility scenarios, silently installing programs to autorun may be interpreted as malware behavior by antivirus software or system administrators. This would cause them to either delete or stop the function.

Improvement: Encryption or Data Masking

One important process that could be consider in the development of the keylogger system is the integration of encryption or data masking techniques. In its current form, the Keylogger writes the keystroke data as it is being recorded, including potentially sensitive information such as login credentials, personal messages, or confidential notes, directly to a plain-text file. If an unauthorized user gains access to the log file, they could easily view or exploit the recorded content. In the sense that you want to hide or maintain data integrity. By encrypting the data as it is written or immediately after logging, the information becomes significantly harder to interpret without the appropriate decryption key, adding a crucial layer of security to the overall system.

Data masking can also play a crucial role in enhancing the effectiveness of behavioral analysis. For example, while still storing exact typed content, specific categories of input like passwords or form entries could be added with masked symbols or tokens. This approach allows the system still to track input frequency, timing, and structure. Additionally, masking or selectively encrypting only specific parts of the data (such as email addresses or credit card numbers) ensures that only meaningful metadata is preserved for analysis, reducing the ethical concerns tied to storing raw input.

**Technologies Used**

**import pynput.keyboard**:

Imports the keyboard module from pynput, allowing you to monitor and control keyboard input in Python.

**Import pynput.mouse**:

Imports the mouse module from pynput, enabling detection and control of mouse events like clicks and movements.

**Import datetime**:

Imports the datetime class from Python’s datetime module to work with date and time objects (e.g., timestamps).

**import wx**:

Imports the wxPython GUI library, which is used to create graphical user interface applications in Python.

**import pandas as pd**:

Imports the pandas library, a powerful tool for data manipulation and analysis, with the alias pd for convenience.

**Conclusion**

This project successfully demonstrates the design, implementation, and potential applications of a Python-based keylogger for academic and educational purposes. By leveraging libraries such as pynput, datetime, and wxPython, the Keylogger captures both keyboard and mouse inputs with contextual timestamps, stores them discreetly, and enables further analysis through a dedicated processing tool. The secondary GUI application enhances the usability of collected data by transforming raw logs into a structured CSV format, making it easier to identify user behavior patterns and trends.

Throughout the development process, challenges such as special key handling, inconsistent log formatting, and UI responsiveness were identified and addressed, with additional improvements suggested for future versions. These include support for browser history parsing, enhanced user feedback, performance optimizations, and self-deployment features such as autorun scripts.

Overall, the project highlights how straightforward it can be to build and deploy a functional keylogger, emphasizing the importance of cybersecurity awareness, ethical boundaries, and the necessity for robust endpoint protection. While the tool provides valuable insights into input tracking and behavioral analysis, it must be used responsibly within legal and educational frameworks to avoid crossing into malicious intent.

Refernces

1.A. Moses. pynput.keyboard — Controlling and monitoring the keyboard. Retrieved from https://pynput.readthedocs.io/en/latest/keyboard.html

2. Python Software Foundation. datetime — Basic date and time types. Retrieved from https://docs.python.org/3/library/datetime.html

3. wxPython Project. Overview — wxPython. Retrieved from https://wxpython.org/pages/overview/

4. The pandas development team. pandas Documentation. Retrieved from https://pandas.pydata.org/docs/

5. Stack Overflow. Batch file to copy files from one folder to another folder. Retrieved from https://stackoverflow.com/questions/986447/batch-file-to-copy-files-from-one-folder-to-another-folder

6. Python Software Foundation. urllib.parse — Parse URLs into components. Retrieved from <https://docs.python.org/3/library/urllib.parse.html>  
  
  
Git hub repository  
<https://github.com/ckemerley/Final-Capstone-Charles-Kemerley.git>