NORTHWEST NAZARENE UNIVERSITY

Cyber Forensics Murder Case Creation

THESIS

Submitted to the Department of Mathematics and Computer Science

in partial fulfillment of the requirements

for the degree of

BACHELOR OF SCIENCE

Trent M. Ferguson

2024

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Trent M. Ferguson
2024

Cyber Forensics Murder Case Creation

Author: Trent Ferguson
Trent Ferguson

Approved: Kevin S McCarty
Kevin McCarty, Ph.D., Associate Professor of Computer Science, Department of Mathematics and Computer Science, Faculty Advisor

Approved: Catherine Becker
Catherine Becker, Ph.D., Associate Professor of English, Department of English, Second Reader

Approved: Dale A Hamilton
Dale Hamilton, Ph.D., Chair, Department of Mathematics and Computer Science

Abstract

Cyber Forensics Murder Case Creation.

FERGUSON, TRENT (Department of Mathematics and Computer Science), MCCARTY, KEVIN (Department of Mathematics and Computer Science)

Northwest Nazarene University (NNU) has a cybersecurity focus for the Computer Science major in the Department of Mathematics and Computer Science. The instructor of the courses for this major plans to write his own textbook and was in need of a unique, hands-on cyber forensics activity for the students. To resolve this problem, he requested a fake murder case to be created for students to work on solving throughout the course. This entails developing various applications and tools to perform various tasks, creating virtual machines to represent the hard drives for characters involved in the fake murder case, and modifying existing tools to better support this course. Applications were created using either Python's Tkinter library or Windows Forms in C#. Overall, the project is now in a much better position to keep moving forward in fulfilling the plans for NNU's cybersecurity courses and future students will soon be able to participate in solving this forensics case.

Acknowledgements

I want to say thank you to the Department of Math and Computer Science for providing funding for work on this project. I also would like to thank Dr. Kevin McCarty and Cody Lirazan for their assistance with the overall scope of the project. Finally, thank you to the rest of the faculty of the Math and Computer Science Department, my peers, my family, and my fiancé for continued support as I pursued my degree.

Table of Contents

Table of Figures

Overview

The research for the forensics case project comprises three main parts: the creation of email and social media accounts, setting up virtual machines (VMs), and developing multiple applications. Most of the project work took place over one hundred hours between May 15th and June 16th of 2023. About thirty hours were spent setting up the social media accounts and creating virtual machines. The remaining seventy hours were spent developing a couple of applications for future development of the cyber forensics case and for the Cyber Forensics course at Northwest Nazarene University (NNU). A few months later, between October 21st and December 13th, a few more applications were worked on for the same purpose.

Overall, the purpose of the forensics case project is to be a hands-on activity for the Cyber Forensics course at NNU. In the class, students will have the opportunity to test the methods that they learn in class on a semi-realistic forensics case. Email and social media accounts were created so that evidence and information can be posted on them for students to view. Similarly, the VMs were created to represent the character's hard drives. Digital evidence can be placed on the VMs to help students determine who they think the culprit is. Finally, the applications were developed for creating evidence as well as for being tools that will assist students in figuring out information about each character and keeping track of information about them.

Background

Dr. Kevin McCarty, the professor of NNU's cybersecurity courses intends to write his own textbook. To go alongside the textbook, he requested an interactive digital

forensics case for students to practice their skills on. In the context of the project, "forensics" is the use of examination to solve crimes. Therefore, a digital forensics case is a criminal case where technological devices are examined to search for digital evidence. To satisfy his request, two students were tasked with working on separate parts of the forensics case project. The forensics case is designed to be based on a semi-realistic murder scenario.

In a broader sense, the project was necessary because crime and technology are becoming strongly connected. Therefore, if a crime is committed, evidence of said crime may be found on a suspect's device, such as their computer or phone. Therefore, there is a growing need for expertise in searching for digital evidence, which leads to the primary goal of the project: students should be prepared for a semi-realistic forensics case by practicing the methods they learn in class in an interactive way.

Implementation

Social Media Account Creation

The first task for the forensics case project was to create social media accounts for each of the characters involved in the fake murder case. Creating the social media accounts would allow students to view their accounts and look for evidence and motives. First, Gmail and Outlook email accounts were created for each of the characters. Afterwards, it was fairly simple to create social media accounts using those email addresses. Almost every character received accounts on X (formerly Twitter), Facebook, and LinkedIn, each of which were created using the character's Gmail addresses. The Outlook addresses, on the other hand, were used for one of the next tasks, which was creating Windows VMs for each of the characters. All of the account login information,

including usernames, passwords, PINs, and birthdates is documented in a Google spreadsheet in case it is needed in the future.

Setting Up Virtual Machines

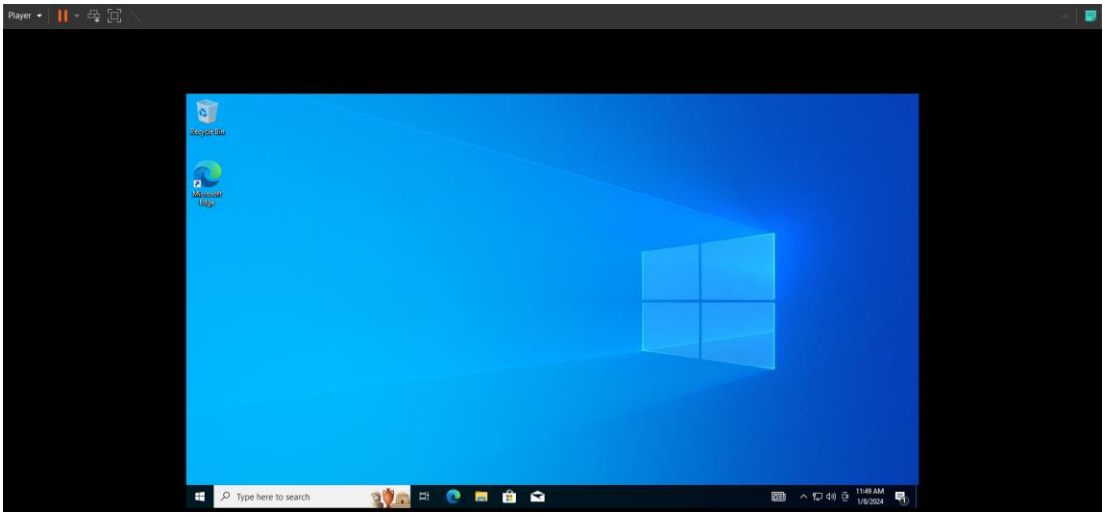

Figure 1. Windows VM. An image of one character's Windows 10 virtual machine running in VMware Workstation 17 Player.

The next part of the forensics case project was to create virtual machines for each character. Like the social media accounts, the VMs were made so that evidence could be placed on them for students to search through. Two VMs were created for each character; one on Windows 10 (shown in Figure 1) and one on an Android phone (shown in Figure 2). The VM's represent each character's computer and phone. The Windows VM was downloaded from the Microsoft website (*Download Windows 10*, 2024). The Android VM, on the other hand, was downloaded from the Fosshub website (Fosshub, 2024). Both of the virtual machines were set up and copied for each character. The VMs are all stored on an external hard drive to be duplicated for each student to access.

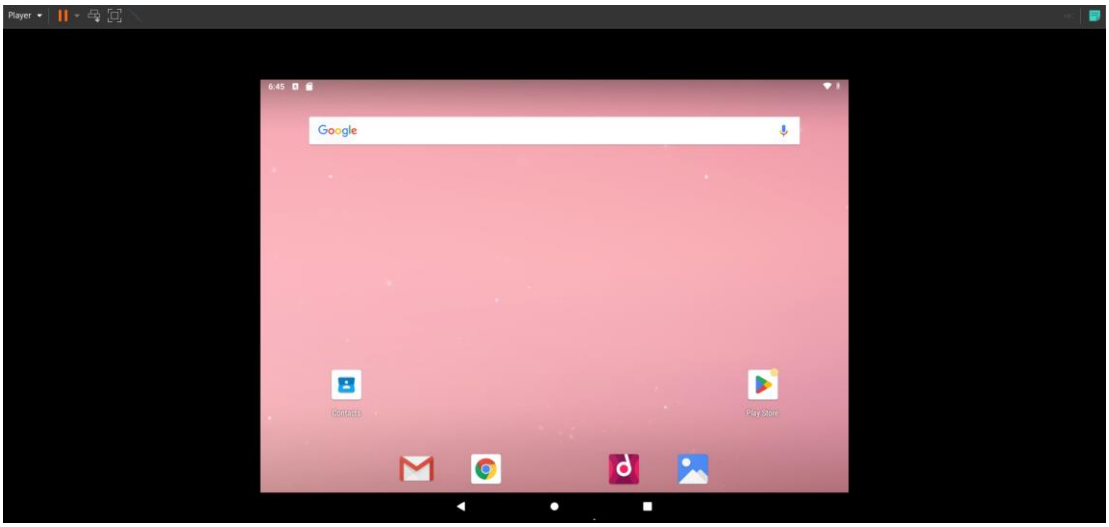

Figure 2. Android VM. An image of one character's Android virtual machine running in VMware Workstation 17 Player.

Application Development

Four application tools were developed, and one existing tool was modified during the project work. The purpose of each of these apps is to assist in the creation of evidence for the forensics case project or to help students in their evaluation of evidence. More information and sample code for each application can be found in Appendix C. The first two tools are complementary, as they were designed to generate email headers and display information from email headers. An email header is a collection of information that is essential for computers to send and receive emails (Bone, 2024). The "EmailHeaderGenerator.py" app (see Figure 3) creates an email header that mimics a Gmail header. It takes input from the user in the form of the sender, recipient, date, time, subject, body, and attachment. The email header generator app was created so that fake emails could be created for the characters in order for students to look through them.

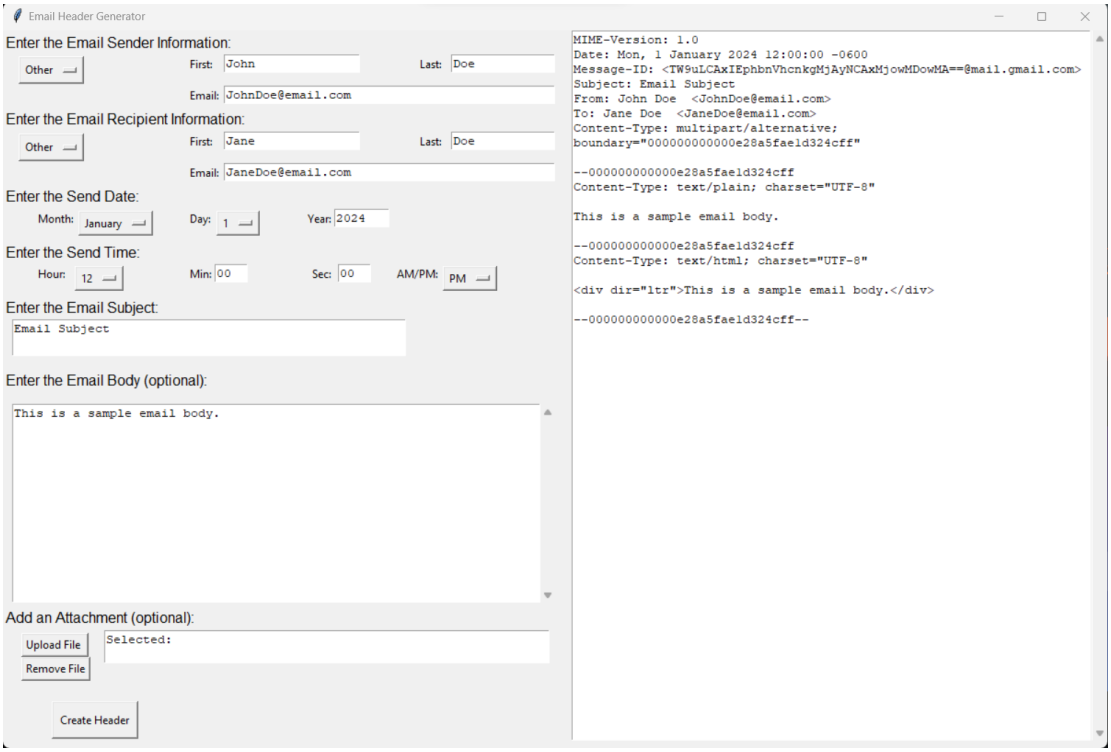

Figure 3. Generated Email Header. This application takes user input on the left and will generate an email header, displaying it in the textbox on the right. It can then be copied to be saved elsewhere.

The “EmailHeaderReader.py” app (shown in Figure 4) strips all of the unnecessary header information and displays the same information that is taken in by the generator app. The purpose of the email header reader is to be a helpful tool for students to easily obtain important information from an email header without having to read through each header.

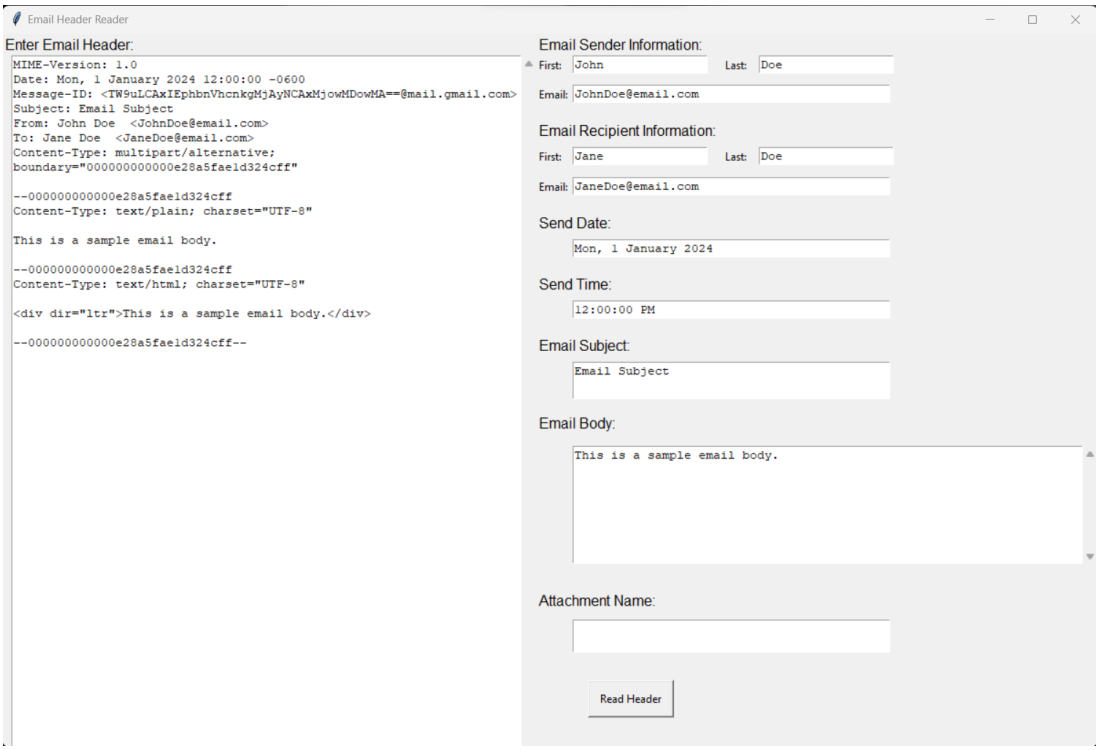

Figure 4. Deciphering an Email Header. The header that was generated in Figure 3 was used as input on the left; the important information was then extracted and is displayed on the right.

The application that was modified was a pre-existing steganography tool that would hide a message within an image file (see Figure 5). This steganography app was developed by Hamzeh Soboh (2014) and works by "replacing bits of useless or unused data in regular computer files with bits of different, invisible information" (para. 1). There were three main things that were added or adjusted. The first were two radio buttons at the top labeled "Hide Message" and "Extract Message." Before adding the

radio buttons, it was unclear to the user whether or not they were hiding or extracting a message. The next addition was the “Use Default Image” tab on the menu bar, which references a directory of images. The “Use Default Menu” tab is useful when there are multiple images that the user may want to add secret messages to. The user copies images to the directory, and the images will be available when the menu button is clicked. The last change was that the program will now clear the text box when switching between hiding or extracting a message as well as after a message is successfully hidden or extracted.

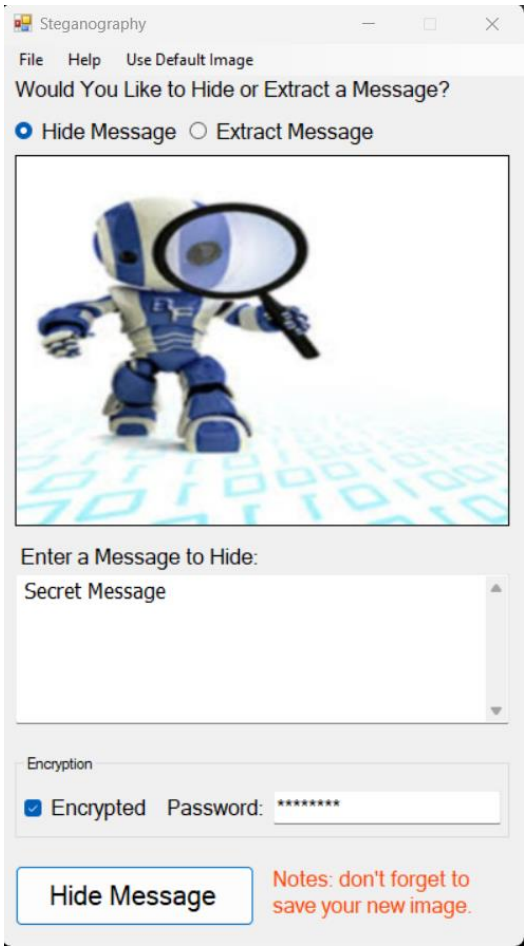

Figure 5. Secret Message in an Image. The steganography tool has taken the phrase “Secret Message” and embedded it in the robot image. It was also encrypted with the password “password” which is required to retrieve the message from the image.

Another steganography tool was also developed. This steganography tool, however, uses a program known as SNOW, or Steganographic Nature of Whitespace according to its index page, to encrypt a message within the whitespace of a text file (Kwan, 2013b). The author page of the SNOW website states that the program was created by Matthew Kwan, a graduate from the University of Melbourne and owner of Darkside Technologies (Kwan, 2013a). The application that was created uses the SNOW command and incorporates it into the graphical user interface of a Windows Forms app, allowing for easier use.

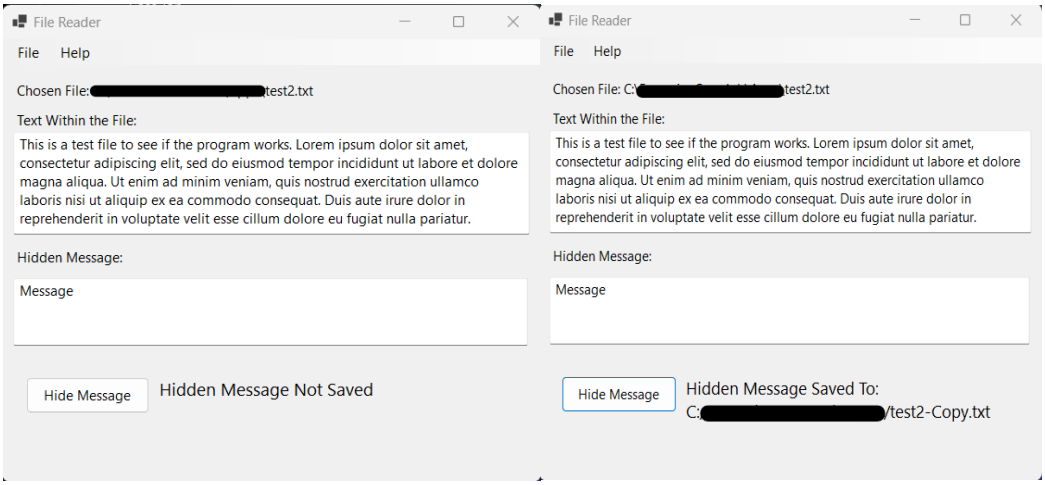

Figure 6. Text File Steganography. On the left, the user chooses a file and enters a message they wish to hide. After pressing the "Hide Message" button, the message will be hidden in a new file that is almost identical to the original, as shown on the right.

The last application that was created is a GPS tool which displays a map with markers on each spot involved in the murder case. When a location marker is clicked, there are text boxes that display the common name, address, and coordinates of the location as well as logs about the location. The logs are intended to display who was at a location, when they were there, and what they were doing there. A location can also be chosen by clicking it in the dropdown menu labeled "Select A Location." After researching various mapping tools, the decision was made to begin implementation of the GPS application using Windows Forms in C# and a control called Great Maps which can be integrated into Windows Forms. See Appendix A for more information on Great Maps. Using Great Maps allowed a map to be displayed on the window, which could be moved around, zoomed in and out, and show markers for each location. Great Maps also has many different map providers to choose from, and the Bing Satellite Map Provider was selected as it guaranteed that no real-world addresses and location names would be shown on the map. The purpose of the GPS tool is to give students a visual representation

of the locations involved in the murder case so that they can understand the timing surrounding the murder. The app also allows students to see information about what characters were connected with each location and why. There is more discussion about the GPS app in the “Future Work” section of the paper.

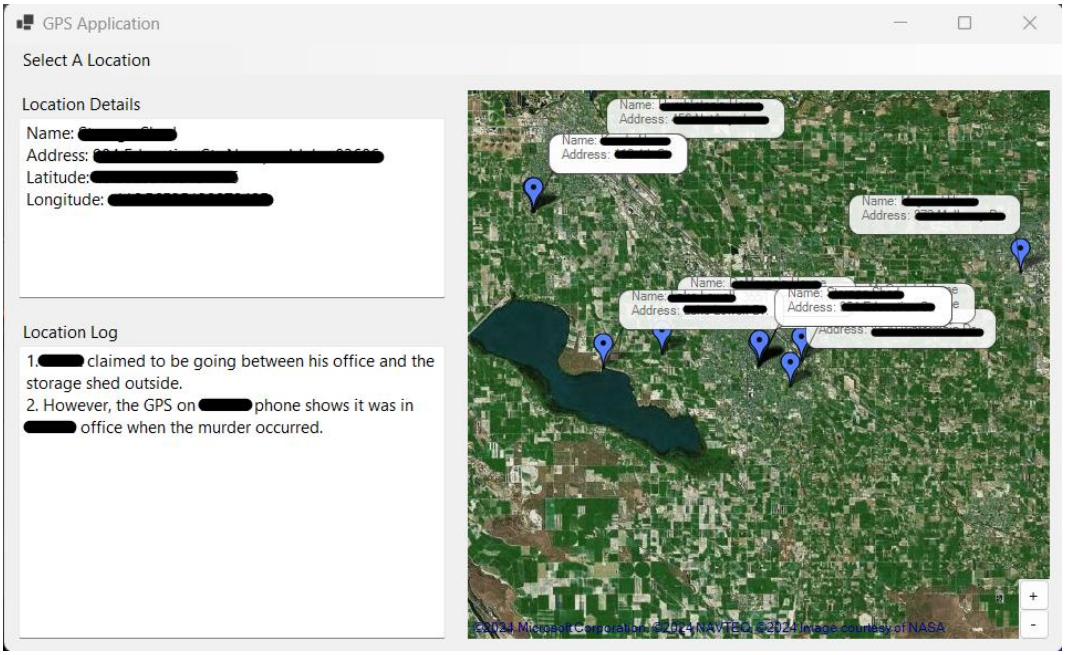

Figure 7. GPS Application. The map on the right can be moved around to view various locations that may be involved in the murder case. When a marker is selected, information will be displayed on the left side.

Results

In summary, there were a number of useful tools created to assist Dr. McCarty in the development of his work and he plans on implementing it in his next revision of the cyber forensics class. Some of the project work was foundational for helping with other parts of the project and also for helping those who will work on the forensics case project in the future. Other parts are directly applicable as courseware. There were a few challenges that could not be overcome, but also some that were able to be solved, so although not everything was completed that was hoped for, much was still accomplished.

Conclusion

Overall, the work done on forensics case project was successful, as it is now in a much better state for moving even further along. Again, although the project is not entirely completed, most of the tasks that were set out to be done were accomplished, as well as some extra tasks that weren't originally planned. Future students in NNU's cyber forensics course will soon be able to practice their skills on a fun, interactive cyber forensics case.

Future Work

There is still much to be done for the forensics case project to be operational. For the character's social media accounts, one character still needs to have accounts created for it as there were complications when originally trying to create them. Also for the social media accounts, there need to be images and posts published for each character. Publishing social media posts relies on AI images to represent each character, which was outside of the scope for this section of the project.

One task that there was not enough time for was to create applications to generate and read text message headers. The text message header apps could be similar to the email header apps, so one suggestion would be to copy the two email header applications and modify them to generate and read text message headers instead. Further research on text message headers would be required to determine how they are created and formatted.

One of the student tools, called "GPS_Simulator.sln" is functional, but still needs some work to be completed. In specific, there needs to be a change in how information for each location is logged. Currently, a couple of the logs are complete, but the

information is hard coded into the project files, meaning that students will have access to all of the information at the start. However, a better use of the GPS app would be to have the logs begin empty and let students fill out the logs themselves. Therefore, one suggestion is to keep a text or JSON file in the directory of the project that will store the logs for each location. When a textbox for the logs is updated, the program can save the data to the file, and it can also be read back into the textbox when the app is run. Changing the way the logs are stored is the best solution to complete the app because it allows students to do research on the case themselves, and to practice keeping track of that information.

Finally, one big step for completing the forensics case project is to create evidence to place on the character's hard drives. Evidence can be in the form of images, texts, emails, documents, etc. To create the evidence, some of the tools developed could be used. These tools include the "Steganography.sln" app to hide messages in images, the "File_Simulator.sln" app to hide messages in text files, and the "EmailHeaderGenerator.py" app to create email headers. If an app is created to generate text message headers, that could be used to create evidence as well. Once evidence is created, the case will be in an even better state to be used, although modifications and additions can still be made.

What Was Learned

I faced many difficulties while working on the forensics case project. Despite these difficulties, I was able to work through most of the problems and complete the tasks at hand. One of the first problems I encountered was while creating various accounts for the characters. Since none of them are real, it was a challenge to create accounts for them

without websites requesting verification of their identities. Websites do this to try to prevent spam or fraud accounts from being created, so they would sometimes ask for a phone number. Since the email and social media accounts are being made for a forensics case, I decided not to add a phone number so that there is no link to a real person. The requirement of a phone number for authentication is what led to one of these accounts' creations being part of a future work task.

I did not face too many issues while working on the applications and tools, but this work did help me learn more about programming in Python and C#. I had used Python plenty of times before, but I learned a lot more about the use of graphical user interface (GUI) libraries. For the most part, I used TKinter for my python GUIs, but I did experiment with a few others such as PySimpleGUI and PyQT. For more information on Tkinter, see Appendix B. On the other hand, updating the image steganography tool was my first time programming in C#. I used Windows Forms for C# to update the image steganography tool and to create both the text file steganography app and the GPS application. I learned a lot about Windows Forms and furthered my skills in object oriented programming through using it. The biggest challenge and growth point from using Windows Forms was when I was trying to add an interactive map to the GUI. I struggled with adding the map for a while, but eventually I was able to get it working and it was better than I had anticipated.

The most important thing that I learned from this project experience was communication in the context of project development. It is imperative to actively listen to what is wanted in a project as well as speak clearly to make sure plans for it are understood. Effective communication is an important skill to learn as it will greatly

improve one's ability to complete a project in a timely and correct manner. I only struggled with communication early on in the project, as I learned that without discussing more detailed project specifications, I would not know what exactly I needed to make an application do.

References

Bone, H. (2024, January 24). *What are email headers?* Proton. Retrieved April 21, 2024, from https://proton.me/blog/what-are-email-headers

Download Windows 10. (n.d.). Retrieved April 17, 2024, from https://www.microsoft.com/en-us/software-download/windows10

Fosshub. (2020, March 25). *Android-X86*. fosshub.com. Retrieved April 17, 2024, from https://www.fosshub.com/Android-x86.html

Kwan, M. (2013a, June 20). *Author*. Darkside. Retrieved March 17, 2024, from https://darkside.com.au/snow/author.html

Kwan, M. (2013b, June 20). *Index*. Darkside. Retrieved March 17, 2024, from https://darkside.com.au/snow/index.html

Nakivo. (2019, November 26). *How to Install Android on VMware: A Step-by-Step Guide*. Retrieved April 17, 2024, from https://www.nakivo.com/blog/installing-android-on-vmware-esxi-a-how-to-guide/

Radioman. (2013, April 16). *Gmap.NET – Great Maps for Windows Forms and* Presentation. CodeProject. Retrieved April 10, 2024, from https://www.codeproject.com/Articles/32643/GMap-NET-Great-Maps-for-Windows-Forms-and-Presenta

Soboh, H. (2014, April 22). *Steganography: Simple Implementation in C#*. CodeProject. Retrieved April 5, 2024, from https://www.codeproject.com/Tips/635715/Steganography-Simple-Implementation-in-Csharp

Tcl Developer Site. (n.d.). www.tcl.tk. Retrieved April 11, 2024, from https://www.tcl.tk/

Tkinter — Python interface to Tcl/Tk — Python 3.10.1 documentation. (n.d.). Docs.python.org. Retrieved April 11, 2024, from https://docs.python.org/3/library/tkinter.html#a-hello-world-program

Appendix A: Information on Great Maps

According to its author, Great Maps (GMap.NET) is a “powerful, free, cross platform, open source” control for Windows Forms the allows the use of “routing, geocoding, directions, and maps” from a variety of map providers such as Google, Bing, and ArcGIS (Radioman, 2013, para. 3).

The following is a code snippet provided in the article that contains some basic control options for Great Maps:

```
public MainForm()
{

        InitializeComponent();

        try
        {
           System.Net.IPHostEntry e =
                System.Net.Dns.GetHostEntry("www.google.com");
        }
        catch
        {
           MainMap.Manager.Mode = AccessMode.CacheOnly;
           MessageBox.Show("No internet connection avaible, going to CacheOnly
        mode.",
                 "GMap.NET - Demo.WindowsForms", MessageBoxButtons.OK,
                 MessageBoxIcon.Warning);
        }

        // config map
        MainMap.MapProvider = GMapProviders.OpenStreetMap;
        MainMap.Position = new PointLatLng(54.6961334816182, 25.2985095977783);
        MainMap.MinZoom = 0;
        MainMap.MaxZoom = 24;
        MainMap.Zoom = 9;

        // add your custom map db provider
        //GMap.NET.CacheProviders.MySQLPureImageCache ch = new
        GMap.NET.CacheProviders.MySQLPureImageCache();
        //ch.ConnectionString = @"server=sql2008;User Id=trolis;Persist Security
        Info=True;database=gmapnetcache;password=trolis;";
        //MainMap.Manager.SecondaryCache = ch;

        // set your proxy here if need
        //GMapProvider.WebProxy = new WebProxy("10.2.0.100", 8080);
        //GMapProvider.WebProxy.Credentials = new
        NetworkCredential("ogrenci@bilgeadam.com", "bilgeada");

        // map events
        {
```

```
            MainMap.OnPositionChanged += new
        PositionChanged(MainMap_OnPositionChanged);

            MainMap.OnTileLoadStart += new
        TileLoadStart(MainMap_OnTileLoadStart);
            MainMap.OnTileLoadComplete += new
        TileLoadComplete(MainMap_OnTileLoadComplete);

            MainMap.OnMapZoomChanged += new
        MapZoomChanged(MainMap_OnMapZoomChanged);
            MainMap.OnMapTypeChanged += new
        MapTypeChanged(MainMap_OnMapTypeChanged);

            MainMap.OnMarkerClick += new MarkerClick(MainMap_OnMarkerClick);
            MainMap.OnMarkerEnter += new MarkerEnter(MainMap_OnMarkerEnter);
            MainMap.OnMarkerLeave += new MarkerLeave(MainMap_OnMarkerLeave);

            MainMap.OnPolygonEnter += new PolygonEnter(MainMap_OnPolygonEnter);
            MainMap.OnPolygonLeave += new PolygonLeave(MainMap_OnPolygonLeave);

            MainMap.OnRouteEnter += new RouteEnter(MainMap_OnRouteEnter);
            MainMap.OnRouteLeave += new RouteLeave(MainMap_OnRouteLeave);

            MainMap.Manager.OnTileCacheComplete += new
        TileCacheComplete(OnTileCacheComplete);
            MainMap.Manager.OnTileCacheStart += new
        TileCacheStart(OnTileCacheStart);
            MainMap.Manager.OnTileCacheProgress += new
        TileCacheProgress(OnTileCacheProgress);
        }
}
```

Appendix B: Information on Tkinter

Tkinter is a graphical user interface (GUI) package for Python. Its origin comes from its connection to a Tool Command Language (Tcl) extension called Tk. Tk, authored in 1991 by John Ousterhout, allows for the creation of a GUI with Tcl and is available on most platforms (*Tcl Developer Site*, 2024).

The documentation provides the following code, which is a simple "Hello World" using Tkinter (*Tkinter — Python Interface to Tcl/Tk — Python 3.10.1 Documentation*, 2024):

```
from tkinter import *
from tkinter import ttk
root = Tk()
frm = ttk.Frame(root, padding=10)
frm.grid()
ttk.Label(frm, text="Hello World!").grid(column=0, row=0)
ttk.Button(frm, text="Quit", command=root.destroy).grid(column=1, row=0)
root.mainloop()
```

Appendix C: Sample Code

Email Header Generator

Code for “EmailHeaderGenerator.py”, which creates email headers based on user inputs. This is a custom function that takes the users inputs for the send date and time. It then uses the date and time to construct and return a string of all of the information concatenated. Before concatenating the information, it also checks and handles errors with the inputs, such as if there is an incorrect number of digits entered or invalid entries (e.g. letters or values over 59 for the second or minute inputs).

```
#Function to Create the Header Date
def CreateHeaderDate():
    #Get the day of the week of the provided date
    sendDate = datetime.date(
        int(str(yearInput.get("1.0", "end-1c")))
        , int(monthValues.index(monthVar.get()) + 1)
        , int(dayVar.get())
    )
    dayOfWeek = weekdays[sendDate.weekday()]

    #Get the time the email was sent and format it
    hour = GetHour()
    minutes = minuteInput.get("1.0", "end-1c")
    seconds = secondInput.get("1.0", "end-1c")

    #Handle TypeErrors and Exceptions for the minute,seconds, and year
textbox entries
    if len(minutes) != 2:
        raise Exception("Minutes must be two characters long")
    if len(seconds) != 2:
        raise Exception("Seconds must be two characters long")
    if len(yearInput.get("1.0", "end-1c")) != 4:
        raise Exception("Year must be four characters long")
    try:
        int(minutes)
        int(seconds)
    except:
        raise TypeError("Only numbers between 00 and 59 are allowed")
    if int(minutes) < 0 or int(minutes) > 59:
        raise Exception("Minutes must be between 00 and 59")
    if int(seconds) < 0 or int(seconds) > 59:
        raise Exception("Seconds must be between 00 and 59")

    time = str(hour) + ":" + str(minutes) + ":" + str(seconds) + " -
0600"

    #Format the information for the header date
    date = ("Date: "
        + str(dayOfWeek) + ", "
        + str(dayVar.get()) + " "
        + str(monthVar.get()) + " "
```

```
        + str(yearInput.get("1.0", "end-1c")) + " "
        + str(time)
    )

    return date
```

Email Header Reader

The following code sample is for “EmailHeaderReader.py”, which deciphers email headers generated by “EmailHeaderGenerator.py” and displays the information in a way that is easier to view. This is the function that gets called when a button is clicked to read an input email header. The first chunk of the function will allow the output textboxes to have text inserted into them. The next section clears them all. There are then several function calls to retrieve and format the data, then store them in variables. Finally, the data is inserted into the textboxes and the textboxes are disabled.

```
#Function to Read the Email Header and Display Relevant Information
def ReadHeader():
    if headerInput.get("1.0", "end-1c") != "":
        senderFirstNameDisplay.configure(state="normal")
        senderLastNameDisplay.configure(state="normal")
        senderEmailDisplay.configure(state="normal")
        recipientFirstNameDisplay.configure(state="normal")
        recipientLastNameDisplay.configure(state="normal")
        recipientEmailDisplay.configure(state="normal")
        dateDisplay.configure(state="normal")
        timeDisplay.configure(state="normal")
        subjectDisplay.configure(state="normal")
        emailBodyDisplay.configure(state="normal")
        attachmentDisplay.configure(state="normal")

        senderFirstNameDisplay.delete("1.0", tk.END)
        senderLastNameDisplay.delete("1.0", tk.END)
        senderEmailDisplay.delete("1.0", tk.END)
        recipientFirstNameDisplay.delete("1.0", tk.END)
        recipientLastNameDisplay.delete("1.0", tk.END)
        recipientEmailDisplay.delete("1.0", tk.END)
        dateDisplay.delete("1.0", tk.END)
        timeDisplay.delete("1.0", tk.END)
        subjectDisplay.delete("1.0", tk.END)
        emailBodyDisplay.delete("1.0", tk.END)
        attachmentDisplay.delete("1.0", tk.END)

        date = GetDate()
        time = GetTime(False)
        subject = GetSubject()
        senderFirst, senderLast = GetSenderName(False)
        senderEmail = GetSenderEmail()
        recipientFirst, recipientLast = GetRecipientName(False)
        recipientEmail = GetRecipientEmail()
        emailBody = GetEmailBody()
        attachmentName = GetAttachmentName()
```

```
        senderFirstNameDisplay.insert(tk.END, senderFirst)
        senderFirstNameDisplay.configure(state="disabled")

        senderLastNameDisplay.insert(tk.END, senderLast)
        senderLastNameDisplay.configure(state="disabled")

        senderEmailDisplay.insert(tk.END, senderEmail)
        senderEmailDisplay.configure(state="disabled")

        recipientFirstNameDisplay.insert(tk.END, recipientFirst)
        recipientFirstNameDisplay.configure(state="disabled")

        recipientLastNameDisplay.insert(tk.END, recipientLast)
        recipientLastNameDisplay.configure(state="disabled")

        recipientEmailDisplay.insert(tk.END, recipientEmail)
        recipientEmailDisplay.configure(state="disabled")

        dateDisplay.insert(tk.END, date)
        dateDisplay.configure(state="disabled")

        timeDisplay.insert(tk.END, time)
        timeDisplay.configure(state="disabled")

        subjectDisplay.insert(tk.END, subject)
        subjectDisplay.configure(state="disabled")

        emailBodyDisplay.insert(tk.END, emailBody)
        emailBodyDisplay.configure(state="disabled")

        attachmentDisplay.insert(tk.END, attachmentName)
        attachmentDisplay.configure(state="disabled")
```

Image Steganography Tool

The “Steganography.sln” app is the program that hides a message in an image file. The following function is one of the additions made to the app. It scans a subdirectory in the projects directory and creates a list of the files in it. It then scans through that list of files and adds the name of the file to a dropdown list on the menu. When one of the names is selected from the menu, the image will be uploaded and displayed in the app.

```
private void InitializeDefaultImages()
{
   files = directory.GetFileSystemInfos();
      foreach (var file in files) {
         ToolStripMenuItem menuItem = new ToolStripMenuItem();
         menuItem.Name = file.Name;
         menuItem.Size = new System.Drawing.Size(224, 26);
```

```
            menuItem.Text = file.Name;
            menuItem.Click += new System.EventHandler(this.defaultImagesToolStripMenuItem_Click);
            this.useDefaultImageToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] { menuItem });
        }

}
```

Text Steganography Tool

The other steganography tool, “File_Simulator.sln” hides a message in the whitespace of a text file using SNOW, which was briefly discussed earlier. This specific function hides the message in the text file when a button is clicked. It hides this message by opening up the command line and running the SNOW command with the user inputs (file name and message to hide) as its arguments.

```
/* Attempts to hide the message in the file provided by the user.
 *
 * If successful, a copy of the file with the hidden message is created with the same
 * name as the original and "-Copy" added after. This file is saved in the same
 * directory as the original.
 */
private void HideButton_Click(object sender, EventArgs e)
{
    try
    {
        //Creates a process that opens command prompt and runs the SNOW command with user input
        // * SNOW command explained at the top of the file
        System.Diagnostics.Process process = new System.Diagnostics.Process();
        System.Diagnostics.ProcessStartInfo startInfo = new System.Diagnostics.ProcessStartInfo();
        startInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
        startInfo.FileName = "cmd.exe";
        startInfo.Arguments = "/C snow -C -m \"" + Textbox2.Text.Trim() + "\" \"" +
Label1.Text.Substring(13).Replace("\\", "/") + "\" \"" + Label1.Text.Substring(13).Replace("\\",
"/").Replace(".txt", "-Copy.txt") + "\"";
        System.Diagnostics.Debug.WriteLine(startInfo.Arguments);
        process.StartInfo = startInfo;
        process.Start();
        process.WaitForExit();

        //Updates a label on the form to show that the message was successfully hidden to the displayed
file.
        Label4.Text = "Hidden Message Saved To:\n" + Label1.Text.Substring(13).Replace("\\",
"/").Replace(".txt", "-Copy.txt");
    }
    //Error Handling for if the process is unsuccessful
    catch (SecurityException ex)
```

```
    {
        MessageBox.Show($"Security error.\n\nError message: {ex.Message}\n\n" +
        $"Details:\n\n{ex.StackTrace}");
    }
}
```

GPS Simulator Tool

The last app was “GPS_Simulator.sln” which is a tool with a map that represents the fake world of the murder case. This function performs a few actions for whenever a marker on the map is clicked. First, it removes and adds the marker to the overlay to make sure the marker is on top of all of the others in case they are close together. It then gathers the location details and the location log then displays them both in their respective textboxes.

```
/* Function to display information about a location when its marker is clicked
 * Also brings that marker to the top as well as zooms in and centers that location
 *
 */
void Map_OnMarkerClick(GMap.NET.WindowsForms.GMapMarker marker, MouseEventArgs e)
{
    Map.Overlays[0].Markers.Remove(marker);
    Map.Overlays[0].Markers.Add(marker);

    string name = marker.ToolTipText.Split("Address: ")[0];
    System.Diagnostics.Debug.WriteLine(name.Substring(6));

    Location temp = location_dict[name.Substring(6).Trim()];

    LocationDetails.Text =
        name + Environment.NewLine +
        "Address:" + marker.ToolTipText.Split("Address:")[1] + ", " + temp.cityAndState + " " +
temp.zipCode + Environment.NewLine +
        "Latitude: " + marker.Position.Lat.ToString() + Environment.NewLine +
        "Longitude: " + marker.Position.Lng.ToString() + Environment.NewLine;

    LocationLog.Text = location_logs[name.Substring(6).Trim()];
}
```