Write Blocker – SOUA

(Hardware and Software Write Blocker)

By

Syed Muhammad Sohaib Abbas (00000237890)

Aden Bin Farrukh (00000216406)

Omer Nasim (00000213601)

Ureed Mustafa Dahri (00000213399)

Supervisor

Asst. Prof. Mian Muhammad Waseem Iqbal

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology,
in partial fulfillment for the requirements of B.E Degree in
Software Engineering

July, 2021

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled "**Write Blocker - SOUA**", carried out by **Syed Muhammad Sohaib Abbas, Aden Bin Farrukh, Omer Nasim, Ureed Mustafa Dahri** under the supervision of **Asst. Prof. Mian Muhammad Waseem Iqbal** for partial fulfillment of Degree of Bachelors of Computer Software Engineering, in Military College of Signals, National University of Sciences and Technology, Islamabad during the academic year 2020-2021 is correct and approved. The material that has been used from other sources it has been properly acknowledged / referred.

Approved by

Supervisor

Asst. Prof. Mian Muhammad Waseem Iqbal

Date: 21/06/2021

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Syed Muhammad Sohaib Abbas
Regn No: 00000237890
Signature: Sohaib Abbas

Aden Bin Farrukh
Regn No: 00000216406
Signature: Aden

Omer Nasim
Regn No: 00000213601
Signature: Omer

Ureed Mustafa Dahri
Regn No: 00000213399
Signature: Ureed Mustafa

Signature of Supervisor

Acknowledgements

We are thankful to our Creator Allah Subhana-Watala to have guided us throughout this work at every step and for every new thought which You setup in our minds to improve it. Indeed, we could have done nothing without Your priceless help and guidance. Whosoever helped us throughout the course of our thesis, whether our parents or any other individual was Your will, so indeed none be worthy of praise but You.

This thesis serves as a tribute to our advisor, Asst. Prof. Mian Muhammad Waseem Iqbal for the time, patience, and efforts that he has spent on us. We are indebted for the vision, knowledge, and mentality acquired from him, and we, the team members, feel privileged and proud to have benefited from his mentoring and guidance.

We would like to thank Asst. Prof. Mobeena Shehzad, for being our Final Year Project coordinator, for her guidance and support and helping us to complete this required work.

We would also like to thank our instructors, professors and all the people in Military college of Signals (NUST) who taught us and helped us to complete our study program.

We are profusely thankful to our beloved parents who raised us when we were not capable of walking and continued to support us throughout in every department of our life.

Finally, we would like to express our gratitude to our friends and all the individuals who have rendered valuable assistance to our study.

“I have no special talents. I am only passionately curious”

-

Albert Einstein

Abstract

Collection and storage of data is day by day becoming a norm. According to a Forbes article "From 2010 to 2020, the amount of data created, captured, copied, and consumed in the world increased from 1.2 trillion gigabytes to 59 trillion gigabytes, an almost 5,000% growth". Majority of today’s organizations routinely make data-driven business decisions. So, data integrity is just as important as the data itself, as changing just one bit among perhaps gigabits of data, will irrevocably alter that data and cast doubt on any prediction or decision made using it.

Therefore, in this thesis we present our solution for this problem which can easily be used by general users and sensitive organizations (strategic, banks, audit firms, law enforcement, armed forces and many others) to overcome the above stated integrity problem, that is the Write Blocker.

There are many Software and Hardware based write blockers available in the market, but are expensive. We, therefore intent to design proprietary software write blocker and hardware write blocker (prototype) to introduce forensic market space in the Pakistan. Our proposed Write Blocker (SOUA) will allow read only access to a storage device for procuring image of the digital evidence while preserving the integrity of the digital device.

Table of Contents

INTRODUCTION

1.1 Overview

So, in this digital age, where almost every system pertaining to our lives has been automated, Cybercrimes and cyber fraud have increased exponentially. To investigate these, digital forensics investigators need to acquire and investigate the data from the host machine, from which the crime has been carried out. For a sound forensic investigation, the original data/evidence must not be changed or tempered with . The main purpose of the project is to introduce a proprietary solution that is low cost and indigenous. Other proprietary solutions that are currently available in the market are too expensive and the non-proprietary solutions unreliable.

1.2 Problem Statement

The main purpose of the project is to introduce a proprietary solution that is low cost and indigenous. Other proprietary solutions that are currently available in the market are too expensive and the non-proprietary solutions unreliable. They are non-indigenous, which means they pose a huge security risk for the national organizations using these devices.

1.3 Objectives

Our goal is to developed a Write blocker prototype that will allow read only access to a storage device for procuring image of the data that is stored while preserving the integrity of the digital device and introduce it to the forensic market place in Pakistan.

1.4 Deliverables

Sr.	**Tasks**	**Deliverables**
1	Literature Review	Literature Survey

2	Requirements Specification	Software Requirements Specification document (SRS)
3	Detailed Design	Software Design Specification document (SDS)
4	Implementation	Demonstration of the proposed project
5	Deployment	Complete project with the proper documentation

Table 1.1: Deliverables

1.5 Document Conventions

Headings are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, according to their level. Font used is Times New Roman. All the main headings are of size 18 and bold. All the second level sub-headings are of size 16 and bold. All the further sub-headings are of size 14 and bold. All references in this document are provided where necessary, however where not present, the meaning is self-explanatory. All ambiguous terms have been clarified in the glossary at the end of this document.

Literature Review

2.1 Overview

In digital forensics, after identifying the digital device for investigation, the next step is data acquisition. By making a bit-by-bit image of the original data, an accurate duplicate image of the same data can be created, which would be forensically valid [1]. The forensic examination and analysis would be carried out on the duplicate image instead of the original data, as it is completely identical to the original and is a risk-free alternative. Digital forensic investigators attach the original storage device to a write blocker that is attached to a forensic workstation [2]. The write blocker prevents the forensic workstation from changing the original media, including addition, deletion, or modification of any data or information [4]. They a forensic image of the original device is taken using a software. As integrity of the digital evidence collected, holds the utmost importance to guarantee a successful digital crime investigation, a write blocker must be employed when acquiring digital evidence from a target device [3]. Without using a write blocker, the digital evidence obtained is discarded by an investigator, disregarded by a counsel, or is challenged in court as evidence tampering, by the opposing party on the assumption that the image of the digital evidence has been modified and it does not represent the original data [2]. Also, viewing the data, and other commands that are non-modifying in nature are allowed even when write protection is enabled.

2.2 Software Write Blockers

A Software write blocker prevents writing or any other modifying commands to be sent to the hard disk drive. It uses what’s called a BIOS interrupt call, to stop the device drivers from modifying the hard disk drive. BIOS (Binary Input Output System), is responsible for the computer's startup and other installation of computer's hardware. Interrupt is basically a signal that stops the OS from executing its current task and switches it over to perform another task. When triggered, the interrupt will transfer the control to its associated interrupt handler. The corresponding interrupt vector, stores this interrupt handler's address.

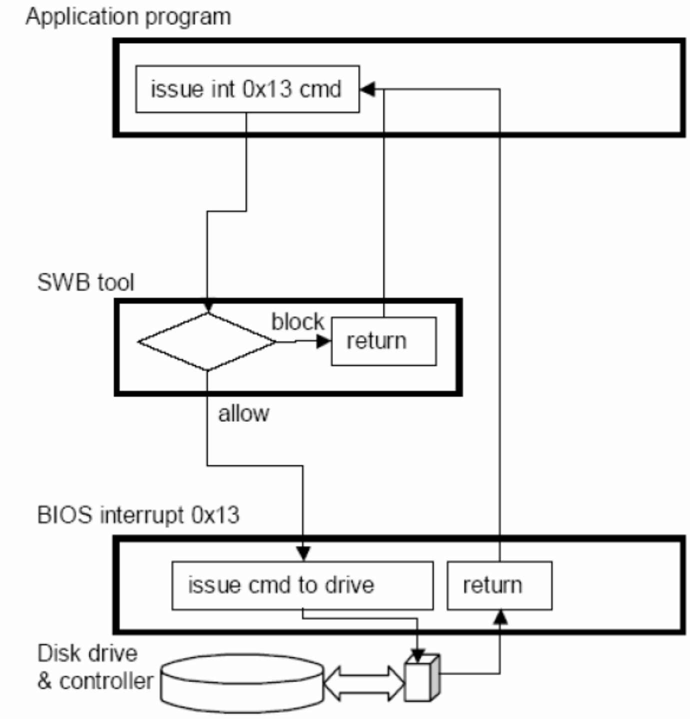


Fig 2.1: Working of a Software Write Blocker

For a command to be sent to the disk drive, the information and command about the destination drive must be placed in the appropriate hardware registers, by the application program. Then the BIOS interrupt 0x13 is then invoked for transferring the control to interrupt handler routine, as it is used to perform disk access operations. When a software write blocker is executed, it installs a new 0x13 handler in place of the previous and saves the address of the previous handler. Software write blocker now intercepts all the commands destined for some disk drive and blocks the modifying command and allows the non-modifying commands [3].

2.3 Hardware Write Blockers

A hardware write blocker (HWB) is a hardware device that attaches to a computer framework with the basic role of intercepting and preventing (or 'obstructing') any changing orders from truly arriving at the storage device. The device is associated between the computer and a storage device. A portion of its capacities incorporate observing and filtering any action that is sent or gotten between its interface associations with the computer and the storage device [6]. Hardware Write Blocker contains software that does the actual Write Blocking function.

2.4 Methods of Write Protection

Exploring more we found other methodologies for implementation write protection involving mounting a file system in read-only mode on the target device so that it can only allow read access [5]. A different approach which was used to perform write protection on the virtual machines is by creating a RAM drive in virtual machine's memory space which will allow data being saved or modified to be saved in the RAM thus preventing anything to be written on the hard disk drive. With this approach there was a problem as the commands were executed in the RAM their logs were made [5].

Another method of write blocking which includes executing program instruction for a blocking driver by a processor of the host computer. A communication is sent from a connection interface device, physically separate from, and operatively coupled to the host computer, to the blocking driver. After receiving the communication sent form the blocking driver and after operative connection of a storage drive to the connection interface device, the connected storage drive is connected to the host computer by the connection interface device. The blocking driver prevents the host computer from altering data stored on the connected storage drive [7].

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

The Software Requirements Specification (SRS) will provide a detailed description of the requirements for the Write Blocker (WB-SOUA). This SRS will allow for a complete understanding of what is to be expected of the WB that is to be constructed. The idea is to develop a WB which will be used for obtaining crucial evidence from target device while preserving its integrity. The WB shall not allow a protected drive to be changed and allow read-only access to that drive. This document highlights the features and requirements of WB-SOUA, that serves as a guide to the developers on one hand and a software validation document for the end users on the other.

3.1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

3.1.3 Intended Audience and Reading Suggestions

[Intended Audience]

- **Project Supervisor (AP Mian Muhammad Waseem Iqbal):** It will help the supervisor to guide the group members. This document will be used to check whether all the requirements have been met.

- **Group Members (Developers/Testers/Documentation Writers):** For FYP group members, this document will provide the guideline for developing and testing the project.

- **UG Project Evaluation Team**: It will help the evaluation team to evaluate the progress of FYP project. The document will provide the evaluators with the scope, requirements and details of the project to be built.

[Reading Suggestions]

The SRS begins with the title and table of contents. All level 1 and level 2 headings are given in the table of contents, but the lower subheadings are not included. Each main heading is succeeded by several subheadings, which are all in bold format. The product overview is given at the start, succeeded by the complete detailed features, including both functional and non-functional requirements. The entire interface is also described. The SRS ends with appendices, including a glossary.

3.1.4 Product Scope

The scope of SOUA Write Blocker is to provide a cost-effective yet comprehensive solution by designing a proprietary software and hardware WB (prototype) that will benefit forensic investigators in keeping integrity of the digital device.

3.1.5 References

Document Name and Version	Description	Location
IEEE SRS Template	It provides the skeleton of this document.	http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&isnumber=16016&arnumber=741940&punumber=5982
Project Synopsis	It defines the scope of the project	File attached

3.2 Overall Description

3.2.1 Product Perspective

SOUA Write Blocker allows forensic investigators to acquire digital evidence without violating the integrity of the digital evidence. It will prevent all modifying commands from reaching the target device and allow access to the device in read-only mode. Its goal is to protect the contents of the storage device from being changed so that the data is not tampered and is legitimate.

3.2.2 Product Functions

- The device will not allow any changes to be made to the protected drive.
- A prompt on the screen shall alert the user when write blocking takes places.
- The device shall report the protection status of all disks.

3.2.3 User Classes and Characteristics

SOUA WB will be primarily intended for the Law enforcement agencies and private forensic agencies. They would be mainly used by forensic investigators.

3.2.4 Operating Environment

- Microsoft Windows and Linux operating system would the software platforms on which the SOUA WB will run.
- Raspberry Pi which will function as our hardware platform

3.2.5 Design and Implementation Constraints

The device does have its own design and implementation constraints

- The write-blocker tool shall not prevent obtaining any information from or about any drive.

3.2.6 User Documentation

- Project Synopsis
- User Manual

3.2.7 Assumptions and Dependencies

- The system should work on the operating systems that are described
- Software must be in the English language
- The device will be backwards compatible with older versions of the operating systems that are described.

3.3 External Interface Requirements

3.3.1 User Interfaces

- **Login screen**: The user enters his/her username or password to log on to the device.
- **Home Screen**: A dashboard of functions that the device performs.
- **Device Selection Screen**: A window that shows the all the devices that are connected with the WB
- **Write Block/Unblock Selection Screen**: A window that gives the user the options to Write block or to Write unblock the selected device

3.3.2 Hardware Interfaces

- Using a SATA connection to a Raspberry pi, the Hard Disk drive will be connected
- The USB drive will be connected to Raspberry pi using USB – A Female Port
- The Raspberry pi would be connected to the PC/Laptop using a USB connection as well

3.3.3 Software Interfaces

- Python
- .Net Framework

3.3.4 Communications Interfaces

Communication interfaces will not be needed, as it would be a stand-alone device.

3.4 System Features

3.4.1 Authentication

3.4.1.1 Description and Priority

The WB requires the user to login using a username and a password when plugged in, so that he/she can be authenticated. This is a HIGH priority

3.4.1.2 Stimulus/Response Sequences

The system prompts a login screen, as soon as the WB is connected and detected by the computer system and asks the user for the username and password for their respective text fields. After the username and password are entered, the WB authenticates the user and displays the interface accordingly.

3.4.1.3 Functional Requirements

REQ-1: The user can login using a username and password

REQ-2: The username and the password entered are verified from the provided database

3.4.2 GUI

3.4.2.1 Description and Priority

The GUI of the entire system should be intuitive, so that the user's experience with the system becomes easier. This is a MEDIUM priority.

3.4.2.2 Stimulus/Response Sequences

If the system is easy to use and is intuitive, it would be easier to navigate by the user through the different interfaces of the system.

3.4.2.3 Functional Requirements

REQ-1: The interface has to be made in a way that it is easy to use and navigate.

3.5 Other Nonfunctional Requirements

3.5.1 Performance Requirements

- **Response Time**: The user interface screens shall not have a load time exceeding two seconds

- **Platform**: The device application shall be compatible with Linux and Windows

3.5.2 Safety Requirements

The use of the SOUA WB has no harms; nor does it have any possibility of loss or damage to the data of any external storage device connected to it

3.5.3 Security Requirements

- The WB shall be password protected.

3.5.4 Software Quality Attributes

- **Legal**: The system will follow the customer privacy policy strictly.

- **Reliability**: The system shall function normally, after restarting due to an error.

- **Ease of Use**: A single day of training would be needed to completely understand the system.

- **Usability**: The graphical user interface of the WB is to be designed with usability in such a manner that is both aesthetically appealing and easy to use for the user to navigate and use the software interface.

3.5.5 Business Rules

As per NUST Policy

3.6 Appendix A: Glossary

3.6.1 Definitions, Acronyms and Abbreviations

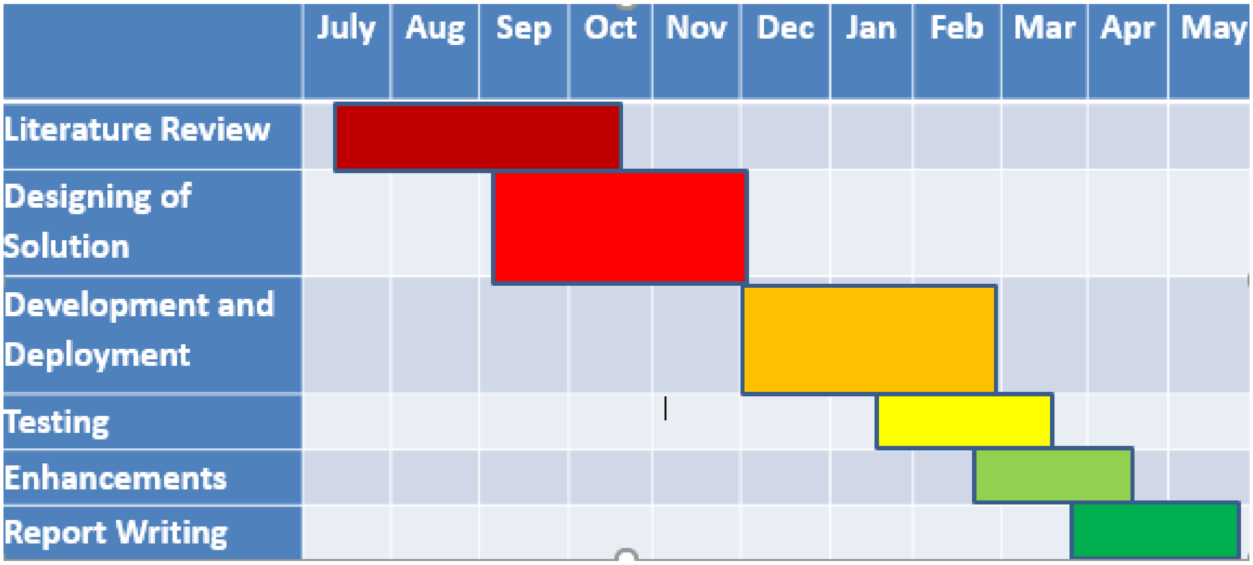
[Definitions]

- End Users – The people actually using the device or affected by it

[Acronyms]

- WB – Write Blocker
- SRS – Software Requirement Specification
- SOUA – Sohaib Omer Ureed Aden

3.7 Timeline of the Project



Design and Development

4.1 Introduction

The Software Design Document (SDD) document provides a high-level view of the entire SDD with purpose, scope, definitions, acronyms, abbreviations, references. The aim of this document is to present, in detail, the functional and non-functional aspects of the project Write Blocker (WB-SOUA). The detailed descriptions and visualizations of the Write Blocker (WB-SOUA) are provided in this document.

4.2 Purpose

The Software Design Document (SDD) describes the architecture and system design of project Write Blocker (WB-SOUA). The document is to provide in depth detail about the feature’s design and requirements of the project, to serve as a guide for the developers and a software validation document for the prospective client. Document includes classes and their inter-relationships, use cases with detailed descriptions, sequence diagrams and various flow charts.

4.3 Scope

The scope of SOUA Write Blocker is to provide a cost-effective yet comprehensive solution by designing a proprietary software and hardware WB (prototype) that will benefit forensic investigators in keeping integrity of the digital device.

4.4 Overview

The SDD begins with the title and table of contents. All level 1 and level 2 headings are given in the table of contents. Each main heading is succeeded by several sub headings, which are all numbered. Definitions and acronyms are mentioned. The scope is given at the start, succeeded by the complete detailed features, including architecture and components. The figures are also described.

4.5 References

Document Name and Version	Descripti on	Locatio n
IEEE SDD best practices	It provides the skeleton of this document.	https://ieeexplore.ieee.org/document/741934
SRS document	It defines the scope of the project	File attached

4.6 Definitions and Acronyms

[Definitions]

- Users – The intended audience using the device
- Registry – A settings database that stores configurations of external and internal hardware devices, applications that are installed and the OS that is running on the computer’s operating system

[Acronyms]

- WB – Write Blocker
- SDD – Software Design Document
- IEEE – Institute of Electrical and Electronic Engineers
- SRS – Software Requirement Specification
- SOUA – Sohaib Omer Ureed Aden
- MVC – Model View Control

4.7 System Overview

This system will use registry API for the drives to restrict read, write and execute operations that are being performed on the specific device. And users will be able to change the restrictions on these devices. Python will be used for the development as it is easy to use and compatible This system will be loaded on to a micro controller/ Raspberry Pi to make it more portable.

4.8 System Architecture

4.8.1 Architectural Design

The architectural design of the WB SOUA is Model-View-Controller (MVC) architecture. The MVC divides the system into the following modules to achieve the complete functionality

- **Model**: The model in this case is the **registry APIs**. It notifies the controller, about any changes made,
- **View:** The view in this case is the **User Interface (UI)**, It handles all the inputs and outputs for our system.
- **Controller:** The controller in this case is the **Code** that is used to send requests to the registry API. It takes input from the view and gives an appropriate response to the model.

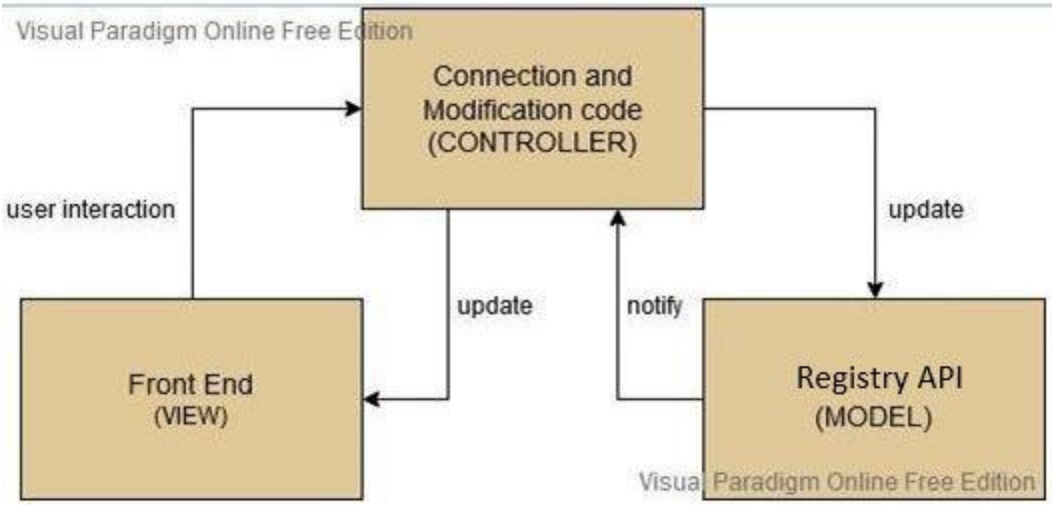

Visual Paradigm Online Free Edition
Connection and Modification code (CONTROLLER)
user interaction
update
update
notify
Front End (VIEW)
Registry API (MODEL)
Visual Paradigm Online Free Edition

Fig 4.1: MVC architecture of WB SOUA

4.8.2 Decomposition Description

The decomposition of the system is explained in the following two ways.

Use Cases

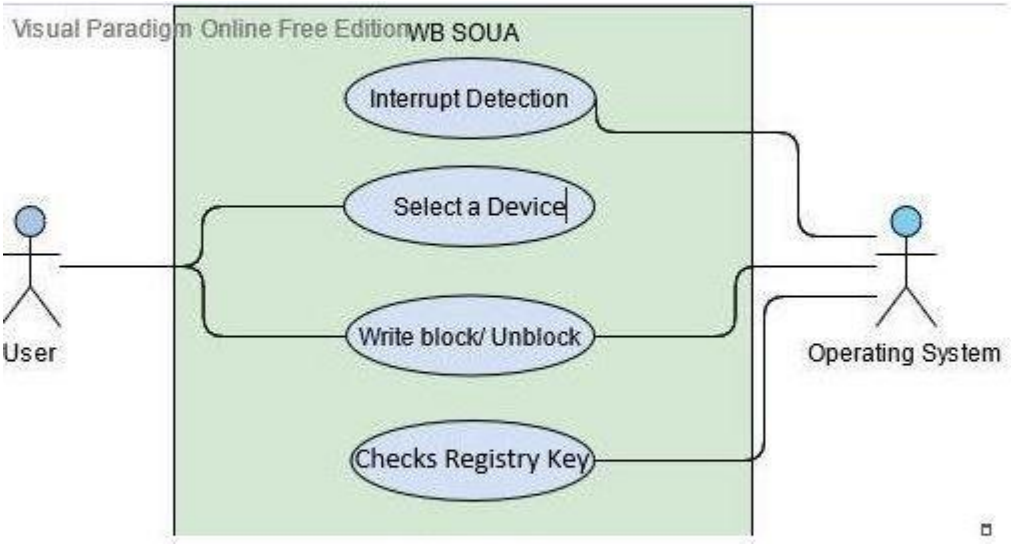


Fig 4.2: Use Case Diagram of WB SOUA

- There are two actors in this use case. The User is the Primary actor and the Operating System is the Secondary actor.

- The user can interact with the system by firstly by selecting a device. After the device selection the user can choose whether to enable write blocking on it, if it isn’t enabled and vice versa.

- The Operating System interacts with the system, when a command for write block/unblock is issued by the write blocker and the OS has to check for the registry key, that it exists or not through the Registry editor. If the key does not exist, the registry editor creates that registry key and places the value of 0 and 1 (0 is for write Unblock/ 1 is for Write bloc). If the key exists, the registry key directly turns the write blocking on or off.

Sequence Diagram

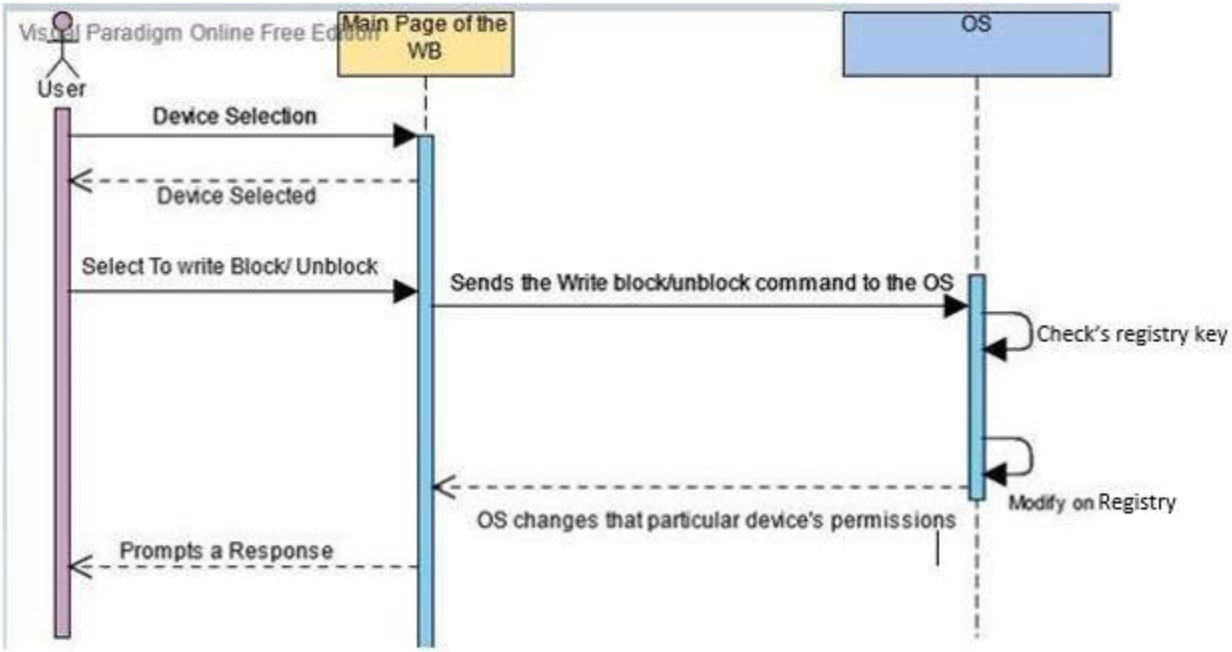
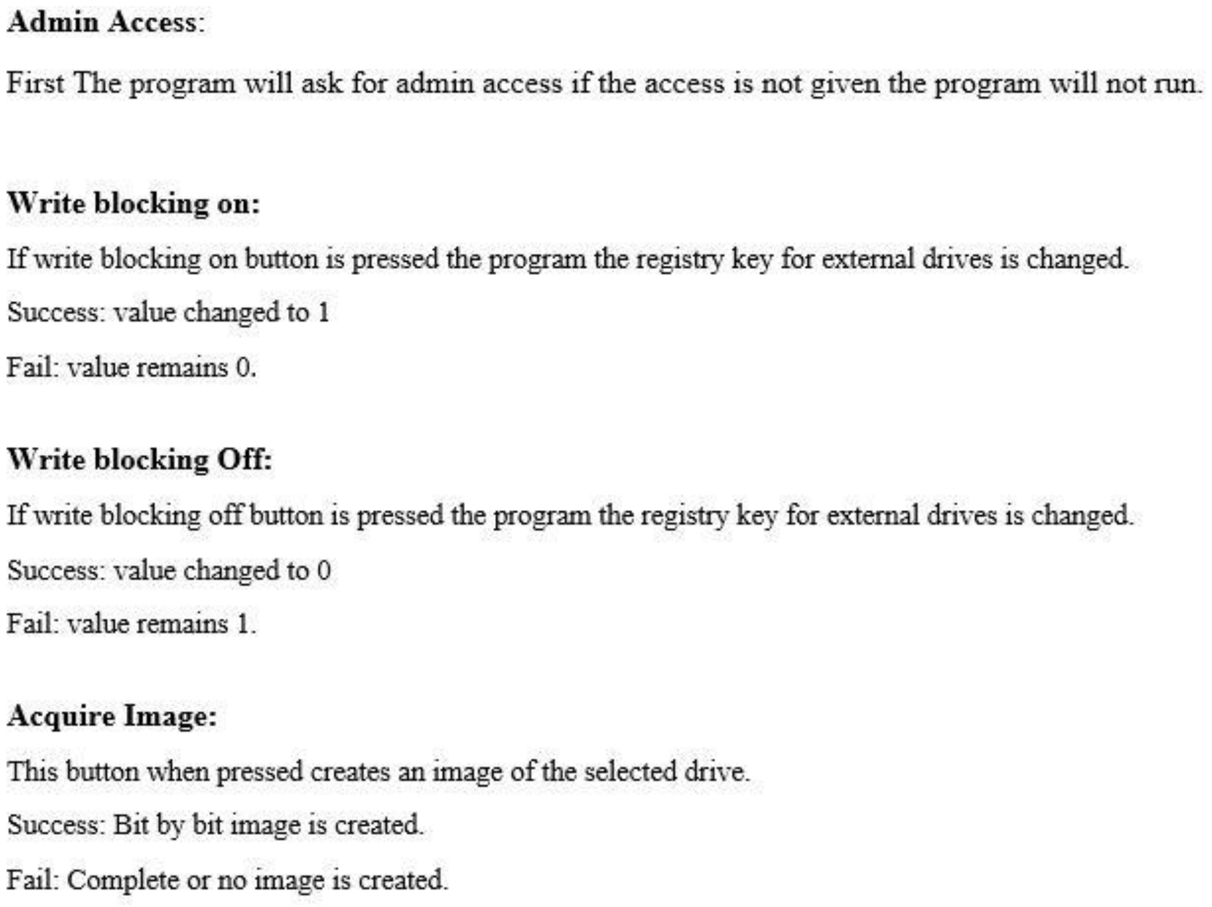

Visual Paradigm Online Free Edition
User
Main Page of the WB
OS
Device Selection
Device Selected
Select To write Block/ Unblock
Sends the Write block/unblock command to the OS
Check's registry key
Modify on Registry
OS changes that particular device's permissions
Prompts a Response

Fig 3.3: Sequence diagram for the operations performed for WB SOUA

The sequence diagram shows the sequence of events and object interactions arranged in time sequence from user's perspective. The user opens the Main window. Then a device is selected. After the device selection the user can choose whether to enable or disable write blocking on it. The write block/unblock command is sent to the Operating System and the OS has to modifies the registry by changing it's the value of registry key (from read, write and execute to read only and viceversa)

4.8.3 Test Cases



Admin Access:

First The program will ask for admin access if the access is not given the program will not run.

Write blocking on:

If write blocking on button is pressed the program the registry key for external drives is changed.

Success: value changed to 1

Fail: value remains 0.

Write blocking Off:

If write blocking off button is pressed the program the registry key for external drives is changed.

Success: value changed to 0

Fail: value remains 1.

Acquire Image:

This button when pressed creates an image of the selected drive.

Success: Bit by bit image is created.

Fail: Complete or no image is created.

4.8.4 Design Rationale

The architecture chosen was **MVC architecture**.

This was chosen particularly, as each component of the MVC has a distinct purpose which makes it easier to understand. And as components can do their work independently in certain flow, it is **highly cohesive**. The components also don’t have much interaction with one another, as once a component has completed its work, it will communicate only its state to the other component. Consequently, that component will come into action. That led us to **low coupling**.

4.9 Component Design

Write block/ Unblock Selection

- User opens the Main Page.
- Clicks on the Write block button
- Selects the Device
- Choses from the Write block/unblock button and clicks on it

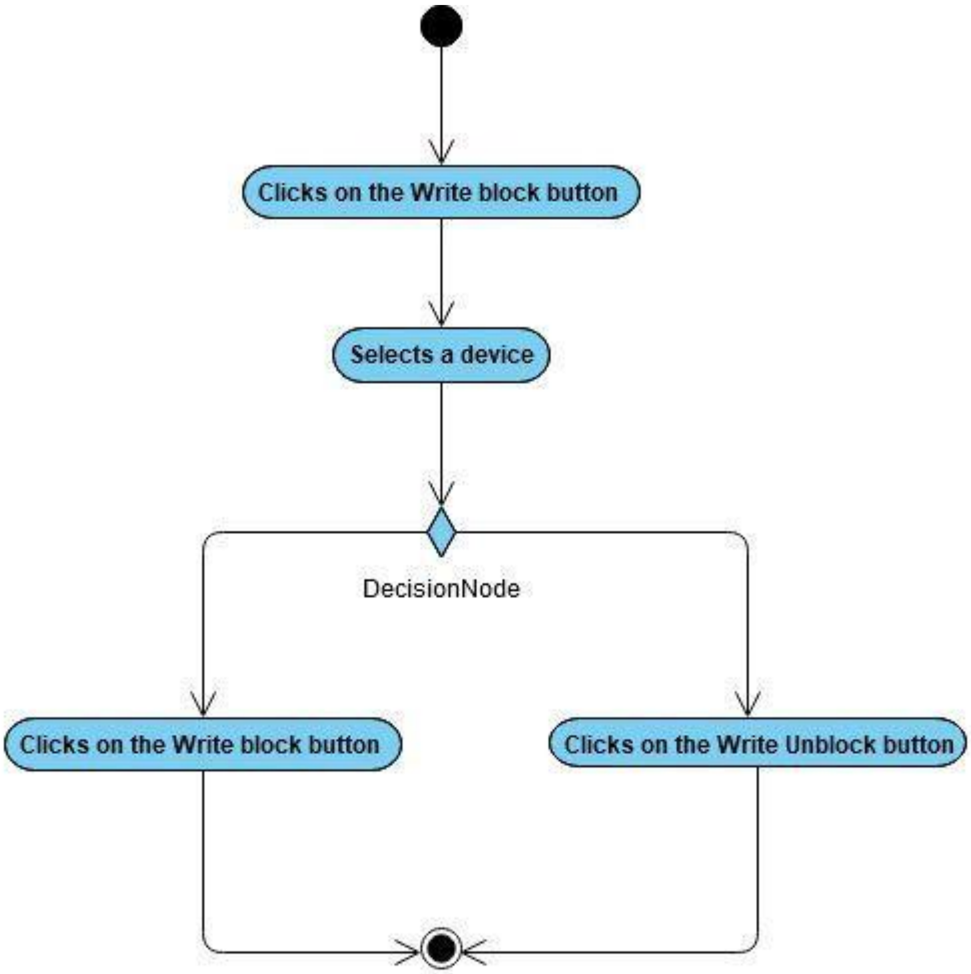

Clicks on the Write block button
Selects a device
DecisionNode
Clicks on the Write block button
Clicks on the Write Unblock button

Fig 4.1: Activity Diagram

Function of the OS upon receiving command from WB

- Write block/unblock command is given by the WB.

- The OS systems checks for the registry key value

- The OS proceeds to modify the registry

- Prompts a message back to the WB after completion

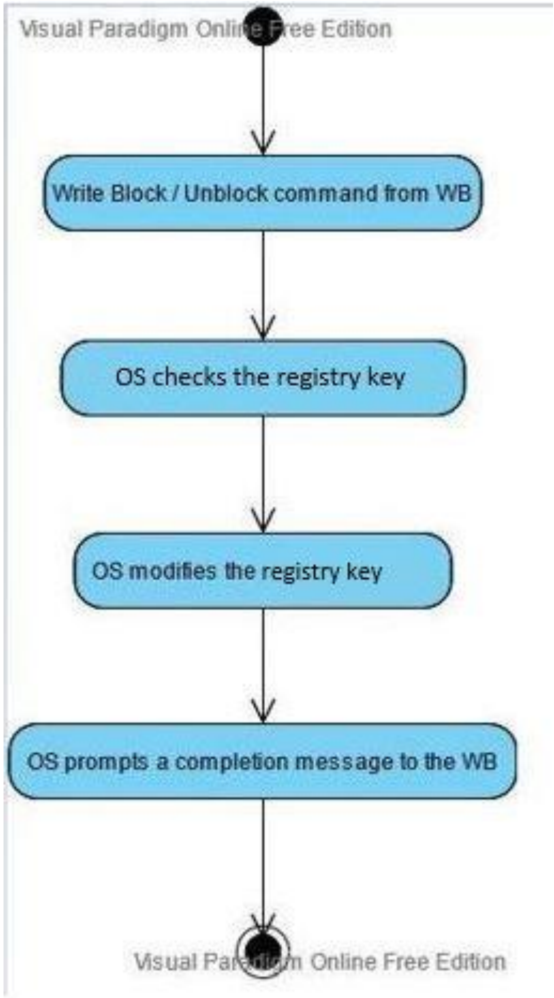

Visual Paradigm Online Free Edition
Write Block / Unblock command from WB
OS checks the registry key
OS modifies the registry key
OS prompts a completion message to the WB
Visual Paradigm Online Free Edition

Fig 4.2: Activity Diagram

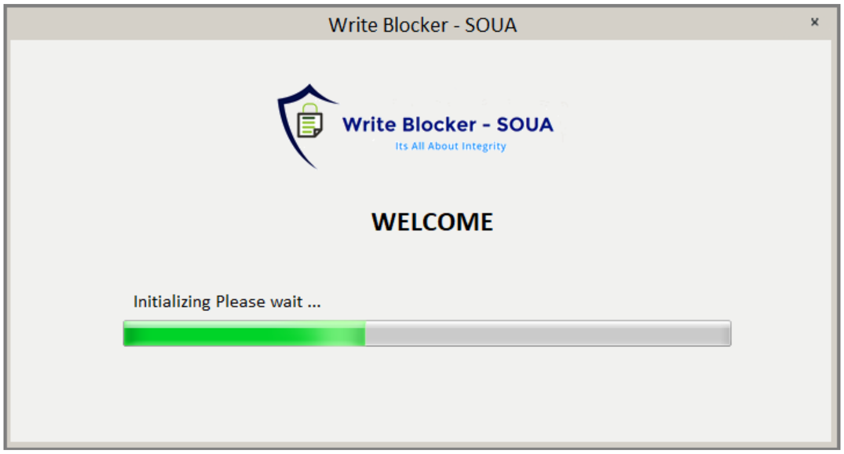
4.10 Human Interface Design

4.10.1 Overview of User Interface

The Home screen of Write blocker consists of three main buttons: Device, Write Block and Help. When we press Device button, the user will be able to see all the devices and their status (whether they are write-blocked or not). By pressing Write Block button user will be able to view write block window where device can be selected and write blocking can be turned on (by selecting write block on button) and turned off (by selecting write block off button). Additionally, if user is facing difficulty operating Write Blocker – SOUA user interface he can select Help which will cover basic guidelines to operate Write Blocker user interface.

4.10.2 Screen Images

Loading Screen:

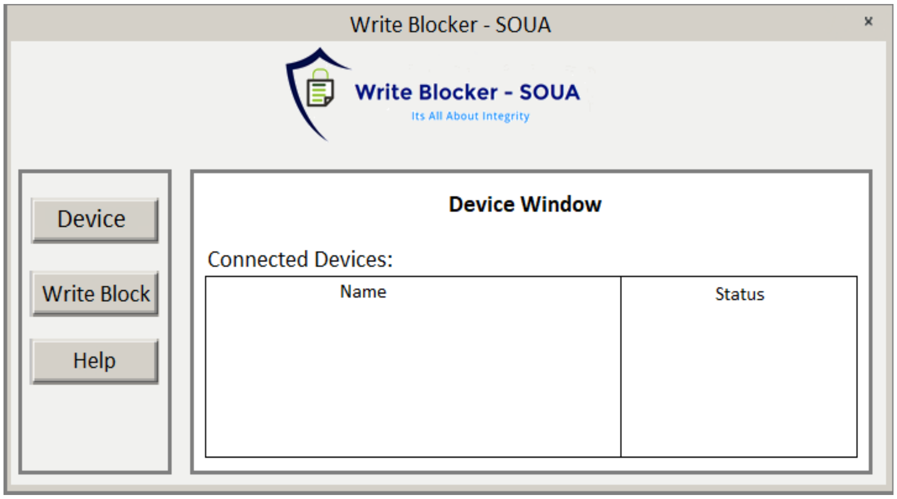


Write Blocker - SOUA

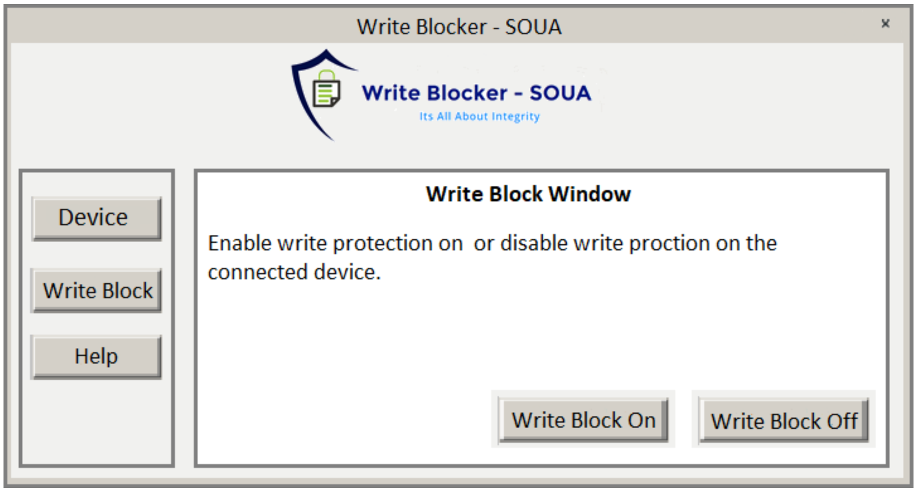
WELCOME

Initializing Please wait ...

Home Screen/Main Window:

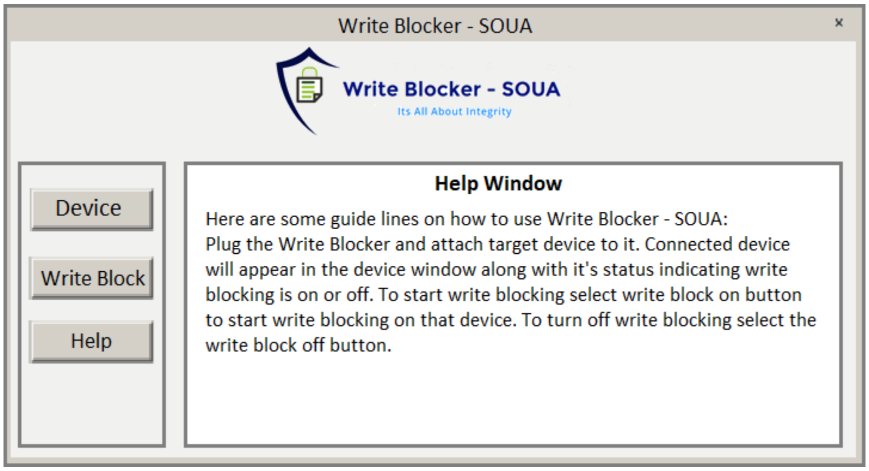


Write Block Window:



Enable write protection on or disable write proction on the connected device.

Help Window



Here are some guide lines on how to use Write Blocker - SOUA:
Plug the Write Blocker and attach target device to it. Connected device will appear in the device window along with it's status indicating write blocking is on or off. To start write blocking select write block on button to start write blocking on that device. To turn off write blocking select the write block off button.

4.10.3 Screen Objects and Actions

MAIN PAGE:

Options of Device, Write block and Help can be seen. User can view Connected Device and their status if write blocking is on or not on the device through Device window. User can start and stop Write Blocking on the device connected to write blocker by pressing Write Block on and off button in the write block window. Moreover, Users can select help if facing difficulty in operating Write Blocker in help window. This page will consist of three main buttons which are following:

1. **Connected Device**: It will show connected device and its status.

2. **Write Block**: It consists of two sub buttons write block on and off, when pressed it will start and stop write blocking on device.

3. **Help**: It will show guideline of how to use Write Blocker.

4.11 Requirements Matrix

Functional Requirement ID #	Functional Requirement	Priority
FR_1	The software should make the selected drive read only	High
FR_2	The software interface will be displayed to the user within 0.5 seconds	Low
FR_3	There shall be relevant icons of buttons and function of buttons to facilitate the users to understand	Medium
FR_4	All devices with Windows 8+, 7, 10 will be able to run this software.	Medium
FR_5	The software should be delivered with proper manual and documentation.	Medium

Implementation and Analysis

5.1 Implementation

The Write blocker through the Registry API, sends the request to the operating system to turn on write blocking. This OS checks for the registry key, that it exists or not through the Registry editor. If the key does not exist, the registry editor creates that registry key and places the value of 0 and 1 (0 is to indicate that write blocking is turned off and 1 indicates that write blocking is turned on). If the key exists, the registry key directly turns the write blocking on or off.

For Write Blocker – SOUA to work it is required that APO USB Autorun application must be installed on the host machine. APO auto run is a tool that detects USB drive connections. It searches for autorun.inf and executes it This will help auto start the Write Blocker – SOUA as we plug it in the host machine.

5.2 Pre requisites

- Python must be installed on the host machine
- APO USB Auto run is required for Write Blocker SOUA to auto run

5.3 Limitations

If a target device is connected to host machine, while the Write Blocker – SOUA is used to turn on write protection, the device will not be write protected. For Write Blocker to work it is mandatory that Write Blocker – SOUA is first plugged in the host machine and write protection is turned on. Then the target device can be connected and it will now be write protected.

5.4 Analysis

The project Write Blocker SOUA, has fulfilled most of the stated objectives like:

- Pakistan based Indigenous solution to obtain evidence without compromising integrity.
- Plug and play solution, which means no power source is required.

- Market competitive price and features.

- Exhibiting a software and hardware write blocker (prototype) which will be used for obtaining crucial evidence from target device while preserving its integrity

5.5 Future Work

Certain aspects of the project can be refined and streamlined to make it more user friendly as well as improve it overall as we have so far worked on a prototype. If time is given and work is done on it, a proper R&D can be done to further increase its efficiency and to add more features to it as well. A proper marketable product can be brought to consumers and clients in the space of Digital forensics in Pakistan and Abroad

Appendix A

Python Script for Write Blocking

```
import tkinter.font
from tkinter.ttk import Progressbar
from tkinter import *
from tkinter import ttk
import winreg
import os
import shutil
from shutil import copytree, ignore_patterns
import ctypes, sys
def is_admin():
    try:
        return ctypes.windll.shell32.IsUserAnAdmin()
    except:
        return False
if is_admin():
    print("Is Admin.")
    w = Tk()
    w.lift()


    width_of_window = 1000

    height_of_window = 750

    screen_width = w.winfo_screenwidth()

    screen_height = w.winfo_screenheight()

    x_coordinate = (screen_width / 2) - (width_of_window / 2)

    y_coordinate = (screen_height / 2) - (height_of_window / 2)

    w.geometry("%dx%d+%d+%d" % (width_of_window, height_of_window, x_coordinate,
y_coordinate))


    w.overrideredirect(1)


    s = ttk.Style()

    s.theme_use('clam')

    s.configure("red.Horizontal.TProgressbar", foreground='red', background='#4f4f4f')

    progress = Progressbar(w, style="red.Horizontal.TProgressbar", orient=HORIZONTAL,
length=1050, mode='determinate', )


#####################################################################################
```

```
    def aquiredata():
        files = os.listdir(r'F:')


        destination = r'E:/USBtransferTest2'
        try:
            for f in files:
                source = r'F:/newfolder123'
                copytree(source, destination, ignore=ignore_patterns('*.pyc', 'tmp*'))
                print("Complete")
        except Exception as e:
            print(e)
            print("Error has occured")


################################################################################


    def writeprotecton():
        now = winreg.ConnectRegistry(None, winreg.HKEY_LOCAL_MACHINE)
        hKey = winreg.OpenKeyEx(now, "SYSTEM\\CurrentControlSet\\Control", 0,
winreg.KEY_ALL_ACCESS)
```

```
   try:

            new0 = winreg.OpenKey(now, 'StorageDevicePolicies', 0, winreg.KEY_ALL_ACCESS)

            b = winreg.SetValueEx(subkey, "WriteProtect", 0, winreg.REG_DWORD, 1)

            print("Write protection truned on")


        except Exception as error1:

            new1 = winreg.CreateKey(hKey, 'StorageDevicePolicies')

            subkey = winreg.OpenKey(hKey, 'StorageDevicePolicies', 0,
winreg.KEY_ALL_ACCESS)

            b = winreg.SetValueEx(subkey, "WriteProtect", 0, winreg.REG_DWORD, 1)

            print("Write protection truned on")


######################################################################################


    def writeprotectoff():

        now = winreg.ConnectRegistry(None, winreg.HKEY_LOCAL_MACHINE)

        hKey = winreg.OpenKeyEx(now, "SYSTEM\\CurrentControlSet\\Control", 0,
winreg.KEY_ALL_ACCESS)
```

```
  try:

         new0 = winreg.OpenKey(now, 'StorageDevicePolicies', 0, winreg.KEY_ALL_ACCESS)

         b = winreg.SetValueEx(subkey, "WriteProtect", 0, winreg.REG_DWORD, 0)

         print("Write protection truned off")



      except Exception as error1:

         new1 = winreg.CreateKey(hKey, 'StorageDevicePolicies')

         subkey = winreg.OpenKey(hKey, 'StorageDevicePolicies', 0,
winreg.KEY_ALL_ACCESS)

         b = winreg.SetValueEx(subkey, "WriteProtect", 0, winreg.REG_DWORD, 0)

         print("Write protection truned off")



#####################################################################################



   def new_win():

      mainwin = Tk()

      mainwin.title('SOUA')

      mainwin.geometry('1000x750')
```

```
    logo = PhotoImage(file="write blocker logo 1.png")
    logofont = tkinter.font.Font(size=20)
    w1 = Label(mainwin, image=logo, font=logofont).pack()


    WBB1 = Button(mainwin, width=12, height=2, text='Write Block on ',
command=writeprotecton, border=0, fg='Blue',
                bg='#EFEFEF')
    WBB1Font = ('Calibri (Body)', 14, 'bold')
    WBB1.config(font=WBB1Font)
    WBB1.place(x=350, y=620)


    WBB2 = Button(mainwin, width=12, height=2, text='Write Block off ',
command=writeprotectoff, border=0,
                fg='Blue',
                bg='#EFEFEF')
    WBB2Font = ('Calibri (Body)', 14, 'bold')
    WBB2.config(font=WBB2Font)
    WBB2.place(x=550, y=620)
```

```
        WBB3 = Button(mainwin, width=12, height=2, text='Aquire Image', command=aquiredata, border=0, fg='Blue',
                      bg='#EFEFEF')
        WBB3Font = ('Calibri (Body)', 14, 'bold')
        WBB3.config(font=WBB2Font)
        WBB3.place(x=150, y=620)

        mainwin.mainloop()

################################################################################

    def bar():
        l4 = Label(w, text='Loading...', fg='blue', bg='#EEEEEE')
        lst4 = ('Calibri (Body)', 10)
        l4.config(font=lst4)
        l4.place(x=18, y=710)

        import time
        r = 0
        for i in range(100):
            progress['value'] = r
            w.update_idletasks()
            time.sleep(0.03)
            r = r + 1

        w.destroy()
        new_win()

###############################################################################

    progress.place(x=-10, y=735)

    a = 'white'
    Frame(w, width=427, height=241, bg=a).place(x=0, y=0)  # 249794
    b1 = Button(w, width=10, height=1, text='Start Program', command=bar, border=0, fg='Blue', bg='#EFEFEF')
    b1.place(x=450, y=650)

    ######## Label
    logo = PhotoImage(file="write blocker logo 1.png")
    logofont = tkinter.font.Font(size=20)
    w1 = Label(w, image=logo, font=logofont).pack()

    w.mainloop()

    # Code of your program here
else:
    ctypes.windll.shell32.ShellExecuteW(None, "runas", sys.executable, " ".join(sys.argv), None,
1)
    print("Is now Admin.")

    # Re-run the program with admin rights
```

BIBLIOGRAPHY

[1] Carlton, G.H. (2007). A Protocol for the Forensic Data Acquisition of Personal Computer Workstations. UMI 3251043. Ann Arbor, MI, ProQuest.

[2]"A Study of Forensic Imaging in the Absence of Write-Blockers" Gary C. Kessler *Embry-Riddle Aeronautical University* Gregory H. Carlton *California State Polytechnic University*

[3] Journal of Digital Forensics, Security and Law Volume 9 | Number 3 Article 4

[4] "Formal specification of a write blocker system for forensic investigation" Author One | Author Two | Author TUCS Technical Report No 718, November 2005

[5] Forensic Focus. (2010, May 11). Connecting a USB device without a write-blocker. Discussion thread. Retrieved from http://www.forensicfocus.com/Forums/viewtopic/t=5809/

[6] https://www.forbes.com/sites/gilpress/2021/12/30/54-predictions-about-the-state-of-data-in 2021/?sh=74b66e91397d

[7] https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2018/02/guardians-of-trust.pdf

[8] A Lightweight Software Write-blocker for Virtual Machine Forensics Patrick Tobin, Nhien-An Le-Khac, M-Tahar Kechadi, University College Dublin, Ireland

[9] Hardware Write Blocker Device (HWB) Specification *Version 1.0 NIST (National Institute of Standard and Technology), Technology Administration, US Department of Commerce*

[10] Lyle, J. (2012, November 30). Computer Forensics Tool Testing. In Forensics@NIST 2012. Retrieved from http://www.nist.gov/oles/upload/5- Lyle_James-CFTT.pdf