Information Security

Assignment 1 Report

Danyal Faheem

19I-2014 CS-F

Mirza Usman Baig

19I-0465 CS-F

Submitted to:

Dr. Abid Rauf

Virus implemented: Polymorphic Virus

Table of Contents

Table of Figures

Introduction

The purpose of this program is to simulate the working of a polymorphic virus with a simple "Hello World" payload. The program uses a cryptographic algorithm from the cryptography package in Python to provide the necessary encryption/decryption functionalities required for the simulation of the virus. Program is divided into multiple files for the easier encryption and decryption.

Virus Overview

Design and Working

The program is divided mainly into 4 modules:

- The decrypted decryption routine
- The encrypted payload
- The encrypted mutation engine
- The encryption techniques

When the program is executed, the decrypted decryption routine decrypts the virus payload and the mutation engine. The decrypted payload is then delivered. After which the mutation engine returns a new decryption routine and the payload and mutation engine are re-encrypted. Their signatures also change during this process. A flow diagram can be seen in Figure 1.

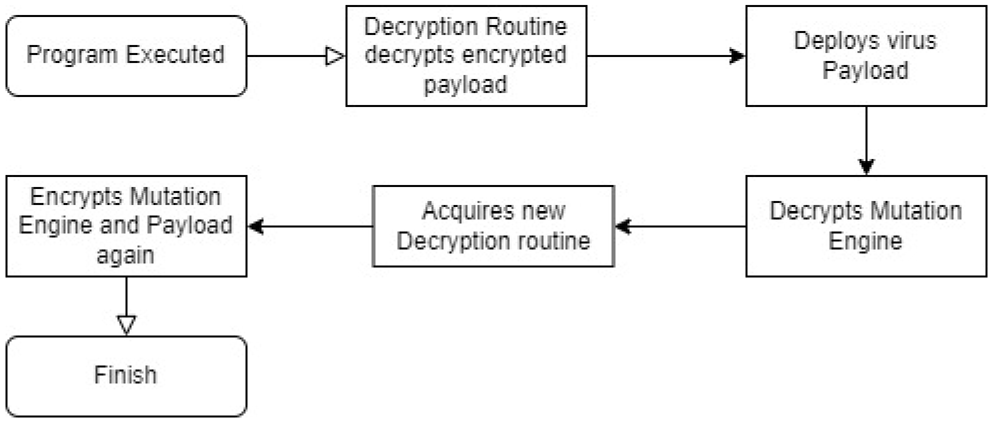
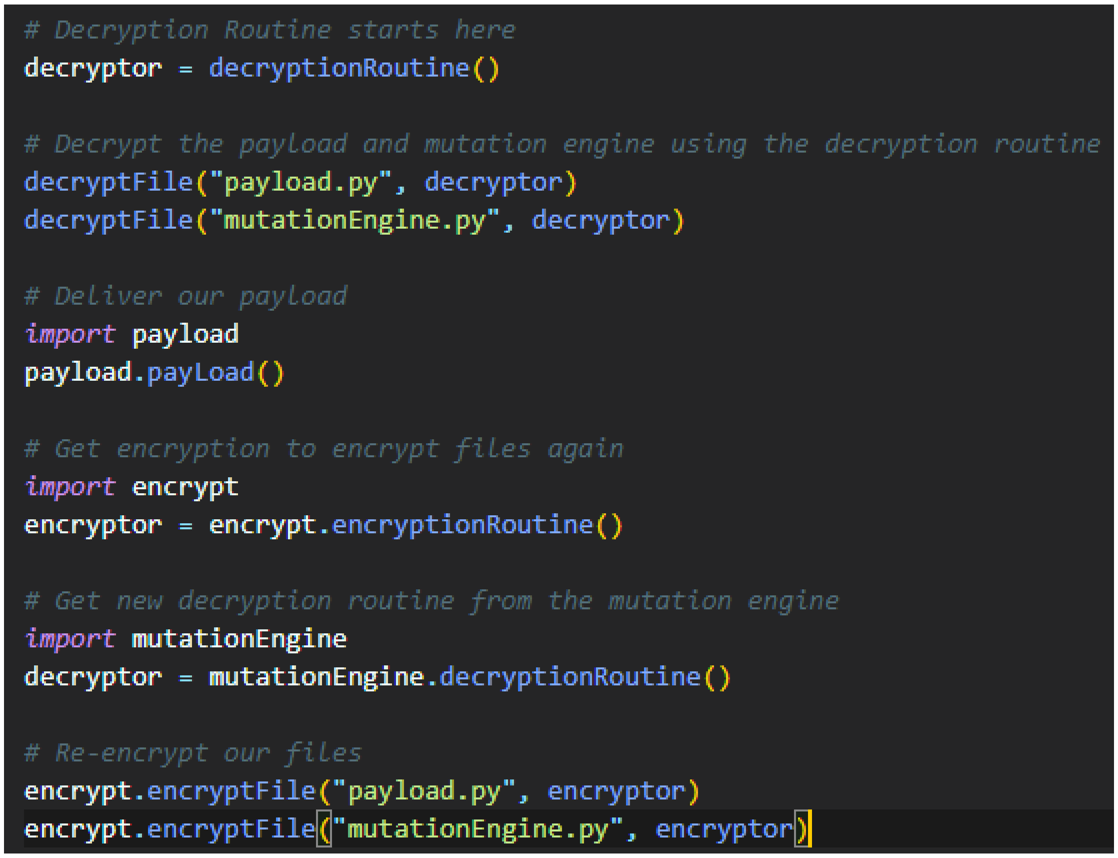

Program Executed
Decryption Routine decrypts encrypted payload
Deploys virus Payload
Decrypts Mutation Engine
Acquires new Decryption routine
Encrypts Mutation Engine and Payload again
Finish

Figure 1. Program Flowchart

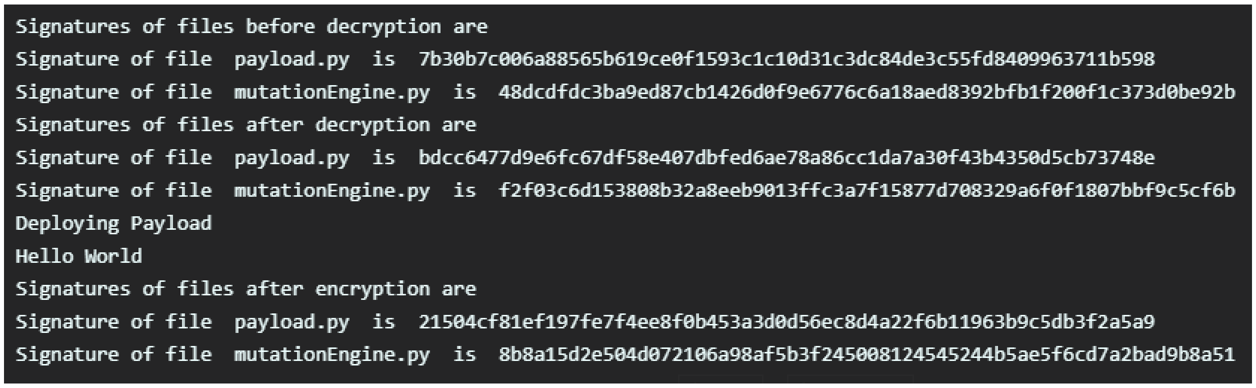
Code snapshot of the main flow of the program can be seen in Figure 2.



```python
# Decryption Routine starts here
decryptor = decryptionRoutine()

# Decrypt the payload and mutation engine using the decryption routine
decryptFile("payload.py", decryptor)
decryptFile("mutationEngine.py", decryptor)

# Deliver our payload
import payload
payload.payLoad()

# Get encryption to encrypt files again
import encrypt
encryptor = encrypt.encryptionRoutine()

# Get new decryption routine from the mutation engine
import mutationEngine
decryptor = mutationEngine.decryptionRoutine()

# Re-encrypt our files
encrypt.encryptFile("payload.py", encryptor)
encrypt.encryptFile("mutationEngine.py", encryptor)
```

Figure 2. Main Code Snapshot

A sample output of the program can be seen in Figure 3.



```
Signatures of files before decryption are
Signature of file  payload.py  is  7b30b7c006a88565b619ce0f1593c1c10d31c3dc84de3c55fd8409963711b598
Signature of file  mutationEngine.py  is  48dcdfdc3ba9ed87cb1426d0f9e6776c6a18aed8392bfb1f200f1c373d0be92b
Signatures of files after decryption are
Signature of file  payload.py  is  bdcc6477d9e6fc67df58e407dbfed6ae78a86cc1da7a30f43b4350d5cb73748e
Signature of file  mutationEngine.py  is  f2f03c6d153808b32a8eeb9013ffc3a7f15877d708329a6f0f1807bbf9c5cf6b
Deploying Payload
Hello World
Signatures of files after encryption are
Signature of file  payload.py  is  21504cf81ef197fe7f4ee8f0b453a3d0d56ec8d4a22f6b11963b9c5db3f2a5a9
Signature of file  mutationEngine.py  is  8b8a15d2e504d072106a98af5b3f245008124545244b5ae5f6cd7a2bad9b8a51
```

Figure 3. Sample output snapshot

Implementation

The virus program has been implemented using python language. Multiple different packages have been used. They are as follows:

- Cryptography: To provide our encryption and decryption functionalities.
- Hashlib: To provide the functionality for our file signatures.
- Random: To randomize the selection of the encryption/decryption schemes.

Modules

High Level View

There are 4 main modules of the program. Their interaction is shown in a high level diagram as below in Figure 4.

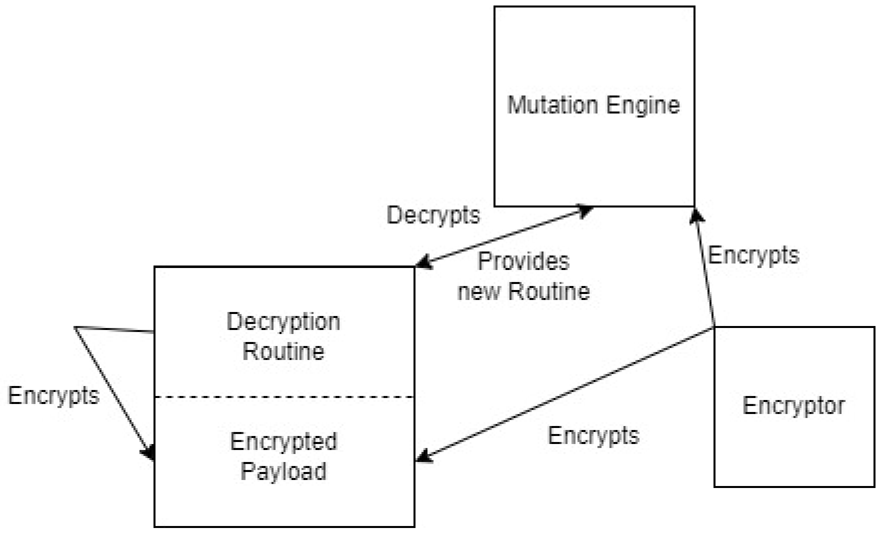
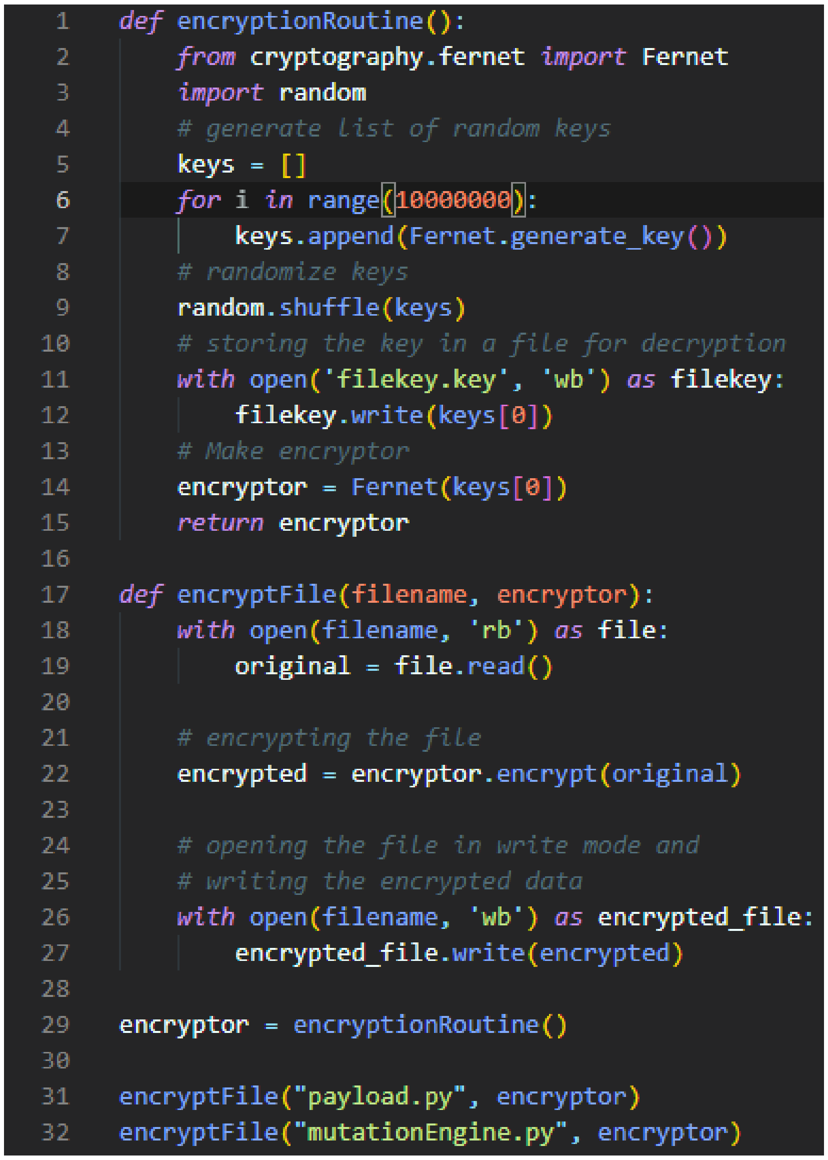


Figure 4. High Level view of modules layout

Encryptor

The encryptor is the main module that encrypts the files and provides the encryption schemes necessary. We are using the Fernet module in the cryptography module in Python to handle the encryption. Seemingly infinite number of encryption schemes are generated and one is picked to make it seem like polymorphism. The encryption scheme is then stored in a file to be used by the decryption routine.

A small snapshot of the encryption module can be seen in Figure 5.

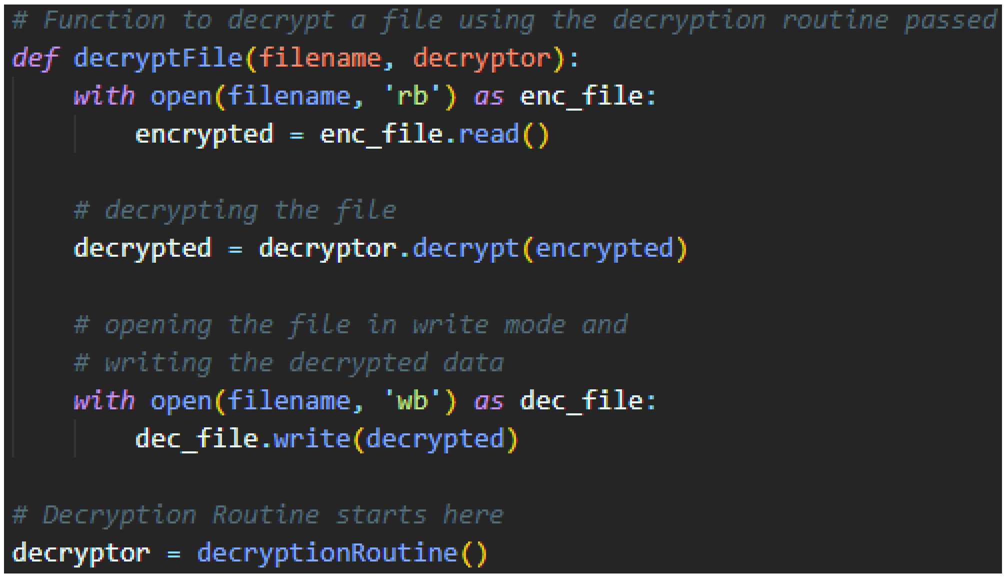


```python
def encryptionRoutine():
    from cryptography.fernet import Fernet
    import random
    # generate list of random keys
    keys = []
    for i in range(10000000):
        keys.append(Fernet.generate_key())
    # randomize keys
    random.shuffle(keys)
    # storing the key in a file for decryption
    with open('filekey.key', 'wb') as filekey:
        filekey.write(keys[0])
    # Make encryptor
    encryptor = Fernet(keys[0])
    return encryptor

def encryptFile(filename, encryptor):
    with open(filename, 'rb') as file:
        original = file.read()

    # encrypting the file
    encrypted = encryptor.encrypt(original)

    # opening the file in write mode and
    # writing the encrypted data
    with open(filename, 'wb') as encrypted_file:
        encrypted_file.write(encrypted)

encryptor = encryptionRoutine()

encryptFile("payload.py", encryptor)
encryptFile("mutationEngine.py", encryptor)
```

Figure 5. Encryption Module code snapshot

Decryption Routine

The decryption routine is provided by the mutation engine however at first, it is already stored in the main file. It creates a decryptor that decrypts the payload and the mutation engine using the key stored in the file. The decryption also uses the Fernet module as the encryption scheme. A small snapshot of the decryption code can be seen in Figure 6.

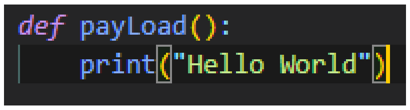


```python
# Function to decrypt a file using the decryption routine passed
def decryptFile(filename, decryptor):
    with open(filename, 'rb') as enc_file:
        encrypted = enc_file.read()

    # decrypting the file
    decrypted = decryptor.decrypt(encrypted)

    # opening the file in write mode and
    # writing the decrypted data
    with open(filename, 'wb') as dec_file:
        dec_file.write(decrypted)

# Decryption Routine starts here
decryptor = decryptionRoutine()
```

Figure 6. Decryption code snapshot

Payload

The payload in our case was a simple Hello world program which would be encrypted at the start, then decrypted by the decryptor and would deliver the payload and re-encrypt again. A snapshot of the payload code can be seen in Figure 7.

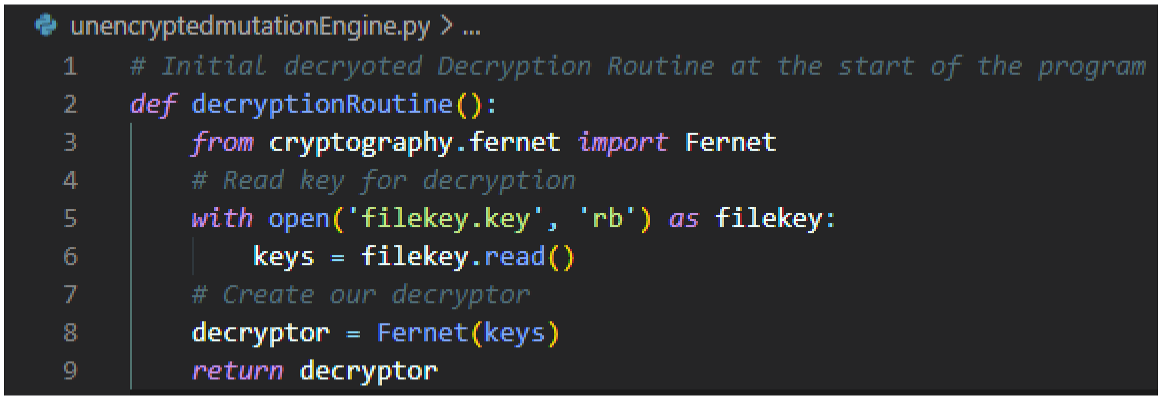


```python
def payLoad():
    print("Hello World")
```

Figure 7. Payload code snapshot

Mutation Engine

The mutation engine is the module that provides the polymorphic part of the virus. It returns a new decryption routine to the main virus body after the encryption of the virus. After which, the mutation engine is encrypted too. It is also encrypted at start. A snapshot of the mutation engine can be seen in Figure 8.


unencryptedmutationEngine.py > ...

```python
# Initial decryoted Decryption Routine at the start of the program
def decryptionRoutine():
    from cryptography.fernet import Fernet
    # Read key for decryption
    with open('filekey.key', 'rb') as filekey:
        keys = filekey.read()
    # Create our decryptor
    decryptor = Fernet(keys)
    return decryptor
```

Figure 8. MutationEngine code snapshot

Bibliography

Bhargava, A. (2022, May 4). *What is a Polymorphic Virus? (How to Create, Detect, and Prevent)*. Tutorialspoint. Retrieved September 11, 2022, from https://www.tutorialspoint.com/what-is-a-polymorphic-virus-how-to-create-detect-and-prevent

VincenzoArceri/python-virus: Simple polymorphic virus written in Python for the "Malware analysis and Design" Master course in University of Verona. (n.d.). GitHub. Retrieved September 11, 2022, from https://github.com/VincenzoArceri/python-virus

What are Polymorphic Viruses? (2022, February 22). GeeksforGeeks. Retrieved September 11, 2022, from https://www.geeksforgeeks.org/what-are-polymorphic-viruses/

Wright, R. (n.d.). *What is polymorphic virus? - Definition from WhatIs.com*. TechTarget. Retrieved September 11, 2022, from https://www.techtarget.com/searchsecurity/definition/polymorphic-malware