Console Recorder (Consent-Based) — Code Explanation & Report

Generated on: 2025-10-24 13:52:00

Title & Purpose

This report describes a consent-based console input recorder implemented in Python. Its purpose is to provide a safe tool for debugging and journaling user-entered lines in a transparent, consented manner.

High-level Summary

The program requests explicit consent before logging, records input lines with basic redaction, and writes them to a rotating log file. It is NOT intended for covert monitoring.

Code Included

Below is the improved, ready-to-run Python script used for the console recorder. Save it as `console_recorder.py` and run with Python 3.

How it Works - Step by Step

1. The program prints an explanation and requests user consent (type 'I consent').

2. If consent is given, a rotating logger is set up with file permissions restricted to the user (if possible).

3. The program reads lines from stdin in a loop; typing 'exit' (case-insensitive) stops the recorder.

4. Each line is passed through a simple redaction function (long digit sequences replaced with [REDACTED]) and written to the log with a timestamp.

5. The program handles KeyboardInterrupt and other exceptions gracefully and informs the user where the log is saved.

Key Functions & Design Choices

- redact(): Naive redaction using a regex to remove long digit sequences.

- setup_logger(): Configures a RotatingFileHandler and tries to set secure file permissions (chmod 600).

- consent_and_run(): User-facing loop that enforces consent and records lines until exit.

Design choices favor transparency (explicit consent) and safety (redaction, limited file permissions).

Caveats & Limitations

- Redaction is naive and may over-redact or miss sensitive patterns. Production redaction requires more robust patterns and testing.

- File permission changes (os.chmod) may not work on Windows or environments that prevent chmod.

- The logger uses a basic rotating handler; further protections (encryption, secured storage) are recommended for real logs.

- The consent check is case-insensitive but relies on the user entering the exact phrase. Consider different consent flows for GUI apps.

Security & Privacy Recommendations

- Avoid logging passwords or secrets; warn users explicitly and enforce input validation.
- Store logs in encrypted form (AES) if logs may contain sensitive info.
- Limit log retention (rotate and delete older backups) and restrict filesystem permissions.
- Use secure transport if sending logs over a network (TLS) and authenticate endpoints.
- Maintain an auditable consent record if logs are collected for others.

Testing Steps

1. Save the script as `console_recorder.py`.
2. Run: `python3 console_recorder.py`.
3. At the consent prompt type: I consent
4. Enter sample lines (avoid secrets). Type `exit` to stop.
5. Inspect `console_recorder.log` in the same directory; verify timestamps and redaction behavior.
6. Try KeyboardInterrupt (Ctrl+C) to test graceful exit handling.

Improvement Ideas (Optional)

- Improve redaction using named detection of credit cards, SSNs, emails, and IP addresses.
- Add optional AES encryption with a passphrase-based key or OS keyring integration.
- Add a GUI or web consent flow to capture richer consent metadata.
- Add unit tests for the redaction and logging behavior.

Contact / Next Steps

If you want, I can:

- Tune the redaction regex for specific patterns (credit card, SSN, phone numbers).
- Add AES encryption to the log file with secure key handling.
- Produce unit tests or a Docker container that runs the recorder in an isolated environment.

Code (improved, ready-to-run)

Save this block as console_recorder.py and run with Python 3.

```
#!/usr/bin/env python3
import re
import os
import logging
from logging.handlers import RotatingFileHandler
from datetime import datetime

LOG_PATH = "console_recorder.log"
EXIT_CMD = "exit"
# Redact long digit sequences of length 12+ (reasonable for card-like numbers).
REDACT_NUMS = re.compile(r"\b\d{12,}\b")

def redact(text: str) -> str:
    \"\"\"Basic redaction: replace long numeric sequences with [REDACTED].\"\"\"
    return REDACT_NUMS.sub("[REDACTED]", text)

def setup_logger(path: str, max_bytes: int = 500_000, backups: int = 3) -> logging.Logger:
    logger = logging.getLogger("console_recorder")
    logger.setLevel(logging.INFO)
    if not any(isinstance(h, RotatingFileHandler) and hasattr(h, 'baseFilename') and h.baseFilename.endswith(pa
        handler = RotatingFileHandler(path, maxBytes=max_bytes, backupCount=backups)
        fmt = logging.Formatter("%(asctime)s %(message)s", "%Y-%m-%d %H:%M:%S")
        handler.setFormatter(fmt)
        logger.addHandler(handler)
        logger.propagate = False
        try:
            open(path, "a").close()
            os.chmod(path, 0o600)
        except Exception:
            pass
    return logger

def consent_and_run():
    print("Console Recorder (for debugging). This will log lines you ENTER.")
    print("Do NOT enter passwords, credit card numbers, or other secrets.")
    consent = input("Type 'I consent' to start recording (or anything else to cancel): ").strip()
    if consent.lower() != "i consent".lower():
        print("Consent not given — exiting.")
        return
    logger = setup_logger(LOG_PATH)
    print(f"Recording started. Type '{EXIT_CMD}' alone on a line to stop and save.\n")
    try:
        while True:
            line = input()
            if line.strip().lower() == EXIT_CMD:
                print(f"Stopped. Log saved to {os.path.abspath(LOG_PATH)}")
                break
            safe_line = redact(line)
            logger.info(safe_line)
    except KeyboardInterrupt:
        print("\\nInterrupted — log saved (if anything was written).")
    except Exception as ex:
        print(f"An error occurred: {ex}")

if __name__ == "__main__":
    consent_and_run()
```

End of Report