# Sigmon — Operation & Safety Stance

A short technical and security summary

---

## What Sigmon is

Sigmon is a native macOS application that draws a live spectrum of the system's audio output. It captures whatever is playing, computes its frequency content in real time, and renders it as an animated trace. It does not record, store, or transmit anything; the audio exists only as a transient calculation on its way to the screen.

## How it operates

**Capture.** Sigmon uses Core Audio *process taps* (macOS 14.2+), a public, documented Apple API. The tap reads the system's audio stream without a virtual device — the sound continues to play normally through the speakers — and runs under the *System Audio Recording* permission, not the microphone. Capture is started and stopped explicitly by the user.

**Processing.** Each frame, the most recent block of samples is windowed and passed through a Fast Fourier Transform; the linear power is pooled onto log-spaced frequency bands, perceptually shaped, and drawn at 60 fps. The signal-processing detail is documented separately in the spectrum-rendering report. The relevant point for safety is what the pipeline *omits*: there is no write-to-disk, no buffering beyond the short analysis window, and no transmission.

**Footprint.** The app is local and self-contained. It has no network code, no user accounts, no file parsing, no inter-process server, and no third-party dependencies. Preferences (palette, shortcut, window mode) are stored in the standard per-user defaults and contain nothing sensitive.

## Safety stance

**Consent is gated by the OS, and visible.** Capture cannot begin without the user approving the System Audio Recording prompt — the operating system's consent checkpoint, which no application can bypass or pre-approve. While capture is active, macOS displays its own indicator and removes it when capture stops; this system signal is authoritative and cannot be spoofed or suppressed by the app. Sigmon additionally shows its state in-window ("Capturing at 48,000 Hz" versus "Stopped"), driven by the same internal flag as the capture itself, so the two always agree.

**Nothing leaves the device.** With no network surface, spectra and any playback metadata never go anywhere. Audio content is never written to disk.

**Minimal attack surface.** Because the app parses no external files, accepts no network input, exposes no IPC, and bundles no outside code, the classic exploitation routes — injection, untrusted-input handling, deserialization, supply-chain compromise — do not apply. In particular, an already-approved copy of Sigmon cannot be quietly commandeered into a recording tool, because it has no input channel through which an attacker could redirect its captured data.

**No uplift to an attacker.** The tap API and its permission belong to macOS and are available to any program; Sigmon neither unlocks nor bypasses them, and it exposes captured audio to nothing but the on-screen drawing. An attacker gains no capability by starting from Sigmon that they would not already have by calling the same public API directly.

## Known gap and priority

**Distribution integrity.** Sigmon is currently *ad-hoc signed* and *not notarized*, and it runs outside the App Sandbox (which the tap requires). Ad-hoc signing carries no developer identity and no tamper-evidence, so a modified build could be passed off as the genuine app — a trusted-looking shell wrapped around added recording-and-exfiltration code. This is the one place where the project's name could be abused, and it is a distribution problem rather than a property of the capture itself.

**Highest-value step:** Developer ID signing plus notarization. This gives users a verifiable guarantee that the binary came from the developer and was not altered, closing the trojan risk and removing the Gatekeeper warning. Secondary discipline: keep the app dependency-free and minimal, so the unsandboxed privilege is never exercised by code that was not written and audited in-house, and defer any auto-update mechanism unless it is itself signed and verified.

---

Scope note: this summary reflects Sigmon's current local, network-free, input-free design. The analysis of the capture permission and indicator assumes the documented behaviour of the Core Audio process-tap API on current macOS; any future feature that adds files, networking, persistence, or dependencies would warrant a fresh review.