

AliExpress's Silent Audio Script Hijacked a Bluetooth Headset

Julien Laurenceau | Published 2026-08-24 | Updated 2026-09-30

<https://pepitedata.com/aliexpress-webaudio-bluetooth-fingerprinting/>

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



TL;DR. A deep-dive investigation found that AliExpress pages run obfuscated anti-abuse scripts that build live Web Audio graphs: oscillator through analyser and processor into zero gain, connected to your actual audio device. Inaudible, but enough to keep a Windows machine from handing audio back to a multipoint headset. Blocking the two scripts fixed it. Mozilla has known since 2023. The viral retelling turned this into "websites can silently identify you with sound", which overstates it: in Firefox, this particular measurement lands in one of three values for 99.24% of users - roughly a CPU family bucket, not an identity. It is one input among many in a composite fingerprint, plausibly feeding bot and fraud scoring. No microphone, no ultrasound, no demonstrated cross-site tracking.

The bug report that started with headphones

In August 2026, [a detailed investigation by Laserphile](#) traced down an odd behaviour: while certain AliExpress pages were open, the investigator's Windows setup would not switch audio back to their Bluetooth multipoint headphones. Multipoint headsets juggle two paired devices; these pages were keeping an audio path active on the PC side, so the headset never saw the PC go quiet. Block two specific scripts and the problem disappeared.

The scripts are `collina.js` and `fireyejs.js`, obfuscated Alibaba security libraries. They are not subtle about ambition: they also collect canvas and WebGL rendering data, screen properties, hardware hints, timing information, interaction signals and automation detectors, then transmit telemetry home. What the investigation documented specifically was the audio piece: each page builds two live Web Audio

graphs. Each graph generates an oscillator signal, routes it through an analyser and a processing node, passes it through a gain node set to zero, and connects to the system audio destination.

Zero gain means you never hear it. But the browser still opens a real audio path to the OS, and on that Windows machine the OS-level power and routing behaviour followed. This is the concrete, well-supported core of the story.

Mozilla had a head start. [Bug 1863193](#) was opened on 5 November 2023 after reports that AliExpress kept a Windows audio power request active. Mozilla engineers later logged exactly the graph shape above - oscillator, analyser, processor, zero-gain node, destination. The mitigation itself is older: [Firefox 118 shipped an AudioContext fingerprinting defense in September 2023](#) that perturbs audio processing outputs slightly per-site.

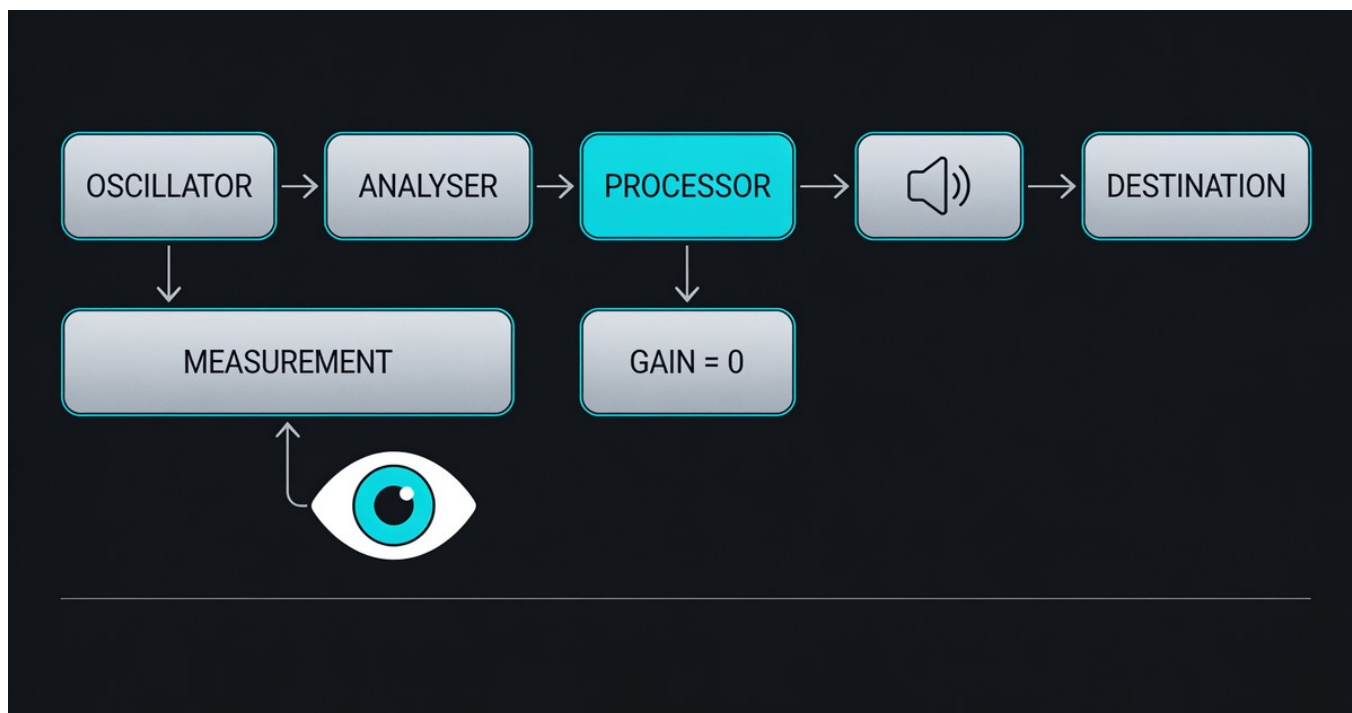
The story went wide on 20 August 2026 via [Hacker News](#), where [the engineer behind ritter.vg published a correction the same day](#) pushing back on the strongest claims. By 23 August, the Polish security outlet Niebezpiecznik had [summarised it on X](#), where it reached several hundred thousand views. As of 24 August the HN thread sat above 1,000 points with over 300 comments.

Why a website touches your audio at all

Web Audio is a legitimate API, and the fingerprinting trick exploits a property nobody designed in. Noise suppression, visualisers, games, voice-change filters, audio worklet DSP: all of it needs real signal processing in the browser, and the exact numbers that come out of floating-point audio maths vary slightly across hardware and software stacks.

Run a known waveform through a compressor or a worklet and read it back, and you get a value influenced by the machine you are on.

That is why the AliExpress graphs route through an analyser and a custom processor before killing the volume at the zero-gain node. The interesting part for them is not the sound - it is the measurement taken mid-path. The connection to the real destination matters for a different reason: some sites appear to use "did the pipeline actually run" as an automation check, since headless browsers historically stubbed audio out. We cannot prove which purpose Alibaba intends, because none of that is visible client-side.



The signal is generated, measured mid-path, and silenced at zero gain before you hear it. The measurement, not the sound, is the point.

It is worth being precise about what this is not. Nothing in the documented graphs uses the microphone. Nothing leaves your speakers as an acoustic signal for anyone to listen to. The signal is generated, analysed and silenced inside the browser's own audio pipeline. Any "silent sound tracking you" framing is wrong.

How identifying is Web Audio, really?

The honest answer is: identifying enough to be useful, not uniquely enough to matter alone. Here is where the viral version and the evidence diverge, and the measured precision of the technique is lower than the headlines and the original posts suggest.

If Web Audio produced a stable, unique value per device, it would be a serious tracking primitive - one that survives cookie clearing and works wherever scripts run. That is the implication the X summary carried. The Firefox engineer's analysis says otherwise: looking at Mozilla's own telemetry, **99.24% of Firefox observations fall into only three values**, and those values track broad CPU instruction families rather than individual machines.

So in current Firefox, with its per-site perturbation shipped in 2023, this measurement tells a site roughly "this is an AVX-era x86 CPU" versus "something ARM-ish". Useful for segmenting a bot farm from a MacBook, useless for picking you out of a crowd.

Two caveats keep this story alive anyway:

1. **Composite fingerprints add entropy piece by piece.** Canvas hashes, WebGL renderer strings, screen metrics, font lists, timing jitter, interaction patterns - each contributes bits. A Web Audio value that only splits the population three ways still refines a score built from twenty other signals. The right statement is: it adds a little, it is not the crown jewel.
2. **Firefox is the mitigated case.** The perturbation defence is not universal across browsers. On less defensive stacks the raw spread may be wider. But "may be" is doing real work there - the investigation did not demonstrate a unique identifier on any browser, and cross-browser persistence (same value in Chrome and Firefox on one machine) was not shown either. Browser implementations genuinely differ, which cuts against the "it reads your hardware truth" narrative.

The honest summary: AliExpress runs a broad fingerprinting and anti-abuse collection system in which Web Audio is one sensor among dozens. Its purpose is plausibly bot detection and fraud scoring - both legitimate problems for a marketplace the size of Alibaba - and possibly automation detection. Whether any of it feeds advertising identity or links sessions across Alibaba properties is unknown from the outside, and nobody has demonstrated it.

The reactions that matter

The Hacker News thread is worth reading because practitioners reproduced adjacent symptoms: unexpected audio-focus grabs, devices waking up, background CPU burn from pages that should be silent, and yes, other Bluetooth quirks. Several asked the reasonable product question: if a page is running live audio processing with the volume at zero, why doesn't the browser show an indicator, the way it does for camera and microphone?

That question has no good answer yet. Distinguishing abuse from legitimate WebAudio use is genuinely hard - a visualiser and a fingerprint probe are the same API calls. Indicator-on-live-audio-context is the obvious candidate UX and I would take it tomorrow.

One practical warning surfaced too: people rushing to `blocklist collina.js` and `fireyejs.js` wholesale may break login, CAPTCHA challenges or checkout on AliExpress, because those same security scripts gate those flows. If you block them, expect friction, and know that a marketplace can simply refuse service to clients that refuse its fraud checks.

What I would tell a platform team

Strip away the headlines and this is an infrastructure governance story, and it lands close to home for anyone running data platforms. What follows is what I would tell a platform team facing this class of problem, in the order I would say it.

- **Third-party scripts have hardware-level side effects.** A marketing tag manager pulling a "security" bundle gave that script control of your users' audio stack. Your dependency audit should include what APIs a script touches, not just what domains it phones.
- **Side effects are how surveillance gets discovered.** Nobody audited Alibaba's code because they suspected audio fingerprinting. It came out because someone's headphones misbehaved. Quiet failures in your own systems deserve the same curiosity.
- **Fingerprinting is a spectrum, not a switch.** "They fingerprinted me" spans "they counted my CPU family to score bot probability" and "they built a persistent cross-site identity". Regulation, disclosure duties and user harm differ enormously between those. Insist on the specific mechanism before reacting - or before writing policy.
- **Browsers are converging on defences unevenly.** Firefox perturbs audio outputs; indicator surfaces do not exist. If you architect products assuming stable client-side identity signals, assume those signals will get noisier, not quieter, and design server-side session and fraud models accordingly.

On legality: GDPR questions hang on purpose, necessity, consent and disclosure, none of which are established here. Anti-fraud processing can be legitimate interest; covert collection beyond what that requires may not be. I will not pretend the answer is knowable from a blog post and a bug tracker.

If you need an independent review of third-party dependencies, data flows and operational side effects in a data platform, see my [Data Platform Audit](#).

Sources

- Primary investigation: [Laserphile - AliExpress webpage keeping multipoint audio busy](#)
- Mozilla bug tracker: [bug 1863193 - AliExpress keeps audio power request active](#)
- Firefox engineer's correction: [ritter.vg - WebAudio and Alibaba](#)
- Firefox AudioContext mitigation: [bug 1358149](#), shipped in Firefox 118
- Discussion threads: [Hacker News](#) · [Niebezpiecznik on X](#)

A note on method: every factual claim above traces to the four primary sources listed; where the public record stops - server-side retention, cross-property linkage, intended purpose - I have said unknown rather than guessing.

Related posts

- [After Coldcard, AI found 1,029 bugs. Censored models found none.](#)
- [Ox Alpha review: test a free AI model before production](#)

- [Local LLMs, political bias, and why llama.cpp still matters](#)

PDF snapshot of the public article. The online version is authoritative; links remain clickable.