SNAPSMACK OPERABILITY AUDIT — DING DONG BELL

OPAUDIT 007 — A failed signed fetch dropped every incoming like, boost, and reply

Audit ID	2026-09-07-007
Report date	2026-09-07 (compiled from the incident record)
First observed	Fixed at 0.7.607D "SECOND KNOCK" (September 2026), diagnosed via the on-screen reason added in 0.7.606D.
Theme	One failing sub-step silently swallowing a whole class of inbound.
Surfaces	Inbound likes, boosts and replies from other fediverse servers.
State	WATCHED WORKING Boosts now show up correctly on our own site's Pixelfed page. One class of peer remains unconfirmed (see §7).
Watched live?	**Yes** — Sean; boosts display properly on the SnapSmack Pixelfed page.
Reporter	Sean + Claude.
Related	OPAUDIT 005 (the receipt log that made the reason visible); OPAUDIT 008.
Publication	Public. No exploitable detail.

1. SUMMARY

A wall of "signature verify failed" rejections looked like a cryptography problem. It wasn't. A single sub-step — fetching the sender's key — was failing, and because the whole verification depended on it, one failure silently dropped every like, boost and reply from that server.

2. WHAT A HUMAN SAW

Interactions from other servers stopped arriving, and the log filled with generic "signature verify failed" lines that gave no way to tell why.

3. ROOT CAUSE

To verify an inbound activity, SnapSmack fetches the sender's actor document (which holds their key) — and it *signed* that outbound fetch ("authorized fetch"). When a peer refused the signed fetch, the whole verification failed with "could not fetch signer," and the inbound activity was dropped. So a peer that didn't like our signed request cost us *all* of its likes, boosts and replies — and it read as a signature failure, pointing the investigation in the wrong direction.

4. EVIDENCE AT DIAGNOSIS

An earlier build (0.7.606D) had made the log print the *exact* reason a verification failed instead of a generic message. That change turned the mystery into a fact: the failures were "could not fetch signer," not digest or

key mismatches. The fix followed directly from seeing the real reason.

5. WHAT WE DID

A failed signed fetch now retries *unsigned* — the ActivityPub default that most instances serve — so the actor and its key are fetched and the activity verifies. A separate tool re-pulls entries that were dropped during the outage, so nobody had to re-post. The log now names the exact remaining reason (transport error, HTTP 401, or an SSRF block) so any leftover case is diagnosable.

6. VERIFIED LIVE

Boosts now show up properly on our own site's Pixelfed page. Confirmed by Sean.

7. EYEBALL NEXT

A few instances genuinely insist on a valid *signed* fetch (strict authorized-fetch mode) and still refuse ours. Those specific servers are named and unconfirmed — inbound from them is not yet proven, and closing that gap is the next piece of work. We can reproduce it directly: our own fediverse lab (Pixelfed, Mastodon, GoToSocial) can be switched into strict authorized-fetch mode, so this is testable against a server we control with full logs, not left to chance in the wild.

8. THE DOCTRINE THIS INCIDENT CREATED

> *A whole class of failures can hide behind one shared sub-step, and a generic error message will send you hunting in the wrong place. Make the system report the exact reason first; the fix usually follows the moment you can see what actually failed.*