SNAPSMACK OPERABILITY AUDIT — DING DONG BELL

OPAUDIT 001 — Pixelfed accepted our posts, then silently dropped them

Audit ID	2026-09-07-001
Report date	2026-09-07 (compiled from the incident record)
First observed	~0.7.347–0.7.350 (July 2026)
Theme	Every assumption about how a peer reads our output was wrong until proven against that peer.
Surfaces	Inbound Follow from Pixelfed; outbound Note delivery to Pixelfed.
State	FIXED — CONFIRMING The two named bugs are fixed and posts landed in testing; Pixelfed as a peer still behaves inconsistently and is being watched.
Watched live?	The two fixes: **yes** — Sean, on live boxes (Mastodon + pixelfed.ca), posts landed and displayed. The peer overall: still rough, see §6.
Reporter	Sean (chose the target; ran the live boxes) + Claude (diagnosed the signing and object-URL failures end to end).
Related	OPAUDIT 008 (modern-Mastodon RFC-9421 signing); OPAUDIT 007 (failed signed fetch, "Second Knock"). Same theme: peer behaviour proven only one peer at a time.
Publication	Public. No live exploitable detail; this is an interoperability post-mortem.

1. SUMMARY

This is the incident that taught SnapSmack its central federation rule. Talking to another fediverse server is not like calling your own code: what the specification says a peer *should* do and what a specific peer *actually* does are different things, and the gap between them is where posts quietly disappear. Two separate bugs, a fortnight apart, both against Pixelfed, both invisible until proven — and neither of them a crash or an error message. The system reported success the whole time.

2. WHAT A HUMAN SAW

First: for about two weeks, every attempt to follow a SnapSmack blog from Pixelfed simply failed. The follow never completed, with nothing on the SnapSmack side to explain why — the request arrived and was rejected as unverifiable.

Then, once that was fixed: deliveries to Pixelfed were *accepted* — the server answered normally — and yet no post ever appeared on Pixelfed. Everything on the sending side looked like it had worked. The post was simply not there.

3. ROOT CAUSE

Bug one — the signature never matched (the failed Follow). Every request between fediverse servers is signed, and both sides must build the exact same text to check the signature against. Pixelfed builds that text from the *path only* — it drops the query string. Our inbox address carried a query string (`?ap=inbox`). So Pixelfed signed one version of the text and we checked a different one; they could never match, and every Follow was rejected as unverifiable.

Bug two — the object address arrived mangled (the vanishing post). When Pixelfed accepts a delivery it later fetches the post object back by its address to display it. Our object address used a query string with an `&` in it (`?ap=note&post=N`). Pixelfed HTML-encoded that `&` into `&` when it dereferenced the address, so the address it actually requested was wrong and returned `404` . The post was dropped *after* being accepted — which is why the sending side saw success and the post still never appeared.

4. EVIDENCE AT DIAGNOSIS

Bug one: eight different signing-string constructions were tested against the captured signature and public key using raw `openssl dgst` . Only the path-only form returned `Verified OK` . That is direct proof of which text Pixelfed actually signs, not a guess from reading the spec.

Bug two: the Apache access log showed `PixelFedBot/1.0.0` requesting the `&` form of the object address and receiving a `404` . The drop was visible in the log once we knew to look for the encoded address.

5. WHAT WE DID

Bug one: the inbox verifier now accepts both legitimate ways of building the signing text — including Pixelfed's path-only form — so a correctly-signed Follow from Pixelfed verifies.

Bug two: object addresses were changed to a plain, path-style form with no query string (`/ap/note/p/N`), so there is no `&` left for a peer to re-encode. (Fixes shipped across 0.7.349 / 0.7.350.)

6. VERIFIED LIVE — AND THE HONEST CAVEAT

The two fixes were watched working on live boxes: posts landed and displayed on Mastodon and on pixelfed.ca. Those two specific bugs are closed.

But we will not call Pixelfed *solid*. In continued use it still behaves inconsistently, and we still see occasional drops we have not fully explained. By contrast our Mastodon side is confirmed clean across solo posts and carousels. So this incident is marked **FIXED — CONFIRMING**: the known bugs are fixed; the peer itself stays under watch.

7. EYEBALL NEXT

Everything proven here was proven *against Pixelfed*. GoToSocial and every other peer are unconfirmed on the exact same two axes — how they build the signing text, and how they encode an object address when they fetch it back. Not because there is a known bug in them, but because it was only ever proven against one peer. Each new peer must be watched on those axes before it is trusted. We now run our own Pixelfed,

Mastodon and GoToSocial instances to do exactly that — controlling both ends gives us the raw logs and every failure in real time, so these can be watched to a verdict rather than guessed at from a dropped post.

8. THE DOCTRINE THIS INCIDENT CREATED

> *"A `2xx` proves transport only, never federation." A success response proves the bytes were carried — not that the other end kept, stored, or displayed the post. Confirm the human-visible outcome on the real peer, separately from the fact that the message was accepted. This rule was learned by getting kicked in the teeth, and it now governs how every federation feature is tested.*