SNAPSMACK_EOF_HEADER Last non-empty line of this file MUST be the canonical EOF marker for this file type: an HTML comment containing five equals, space, the literal string 'SNAPSMACK EOF', space, five equals. Missing or different = truncated/corrupted. Restore before saving.

SECAUDIT 051 - Multisite Federation Control Plane & SNAP SLAPPER

SECURITY AUDIT / PUBLIC REMEDIATION RECORD

Date: 2026-08-25 Scope: the multisite/federation code added since audit 049 - `core/multisite-api.php`, `core/smackverse.php`, `core/smackverse-webcron.php`, `core/mesh-helpers.php`, `directory-api.php` and the public directory - plus the SNAP SLAPPER desktop photo editor (`tools/hub/`) after it was split out from THE HUB. Status: **two high, two medium and three low findings - all fixed in 0.7.559.**

Why this audit

The last audit reviewed the CMS as it stood. Since then a real **hub-and-spoke control plane** was built: one blog (the *hub*) can now send commands to the other blogs connected to it (the *spokes*), and a public directory on photoblogs.fyi lets blogs opt in to a shared people-finder. That is a lot of new machinery, and the most valuable kind to attack - one blog reaching into another, and a public form anyone on the internet can reach. So it got its own review: four reviewers reading the real shipped code in parallel, and every finding re-checked by hand against the source before a line was changed.

The engine room held up well. What actually goes over the wire between blogs - the cryptographic signatures on federated messages, the outbound requests, the remote-job runner, the self-healing cron - was built carefully and checked out. The holes were in the **access-control glue** around it: who is allowed to ask for what.

Finding A - a spoke blog could seize hub powers (CLOSED)

Severity: high. Privilege escalation across the trust boundary between two of your own blogs.

When a spoke connects to a hub, the two share one secret key so the hub can send the spoke commands. The catch: the spoke also holds that key - it generated it - and the hub was not checking *who* was on the other end of a request. It authenticated by matching the key and then trusted whatever the record said. Most of the hub's commands correctly required the caller to be a hub. Four did not.

That meant a spoke - or an attacker who had compromised a single spoke - could turn its own key around and, **on the hub**, do any of these:

- mint a working posting credential;
- read the email address and IP of everyone who had left a comment awaiting moderation;
- create posts and upload files;
- wipe and rewrite the blogroll.

Why it existed. One shared secret was doing two jobs - proving "the hub is calling the spoke" and, if misused, "a spoke is calling the hub" - and the direction was never enforced on those four endpoints. The other dozen commands in the same file all had the check; these were the ones that slipped through, and the comment-reading one was missing the exact guard its four sibling endpoints carried.

Fix. All four now require the caller to be a hub, using the same one-line check as their siblings. A spoke replaying its key matches a spoke record and is refused; a genuine hub call passes. A handful of read-only status endpoints leak lower-value data the same way and are gated as a tracked follow-up once their intended direction is confirmed, so a working spoke's telemetry is not broken by mistake.

Finding B - anyone on the internet could hijack a public directory listing (CLOSED)

Severity: high. Public-listing defacement and denial, no login required.

The photoblogs.fyi directory sign-up endpoint had **no authentication at all**, and a blog's address is public (it is printed on the directory page). It also trusted whatever was posted to it. So anyone could:

- re-submit a victim's address with their own name, description and avatar, overwriting an approved listing that **stayed live** - bypassing the "a human reviews new members" promise entirely; or
- send a removal for any address and delist that blog.

Why it worth reading. The endpoint took the content straight from the request body. Escaping was in place downstream, so this was not a script-injection hole - it was that the *authority* for what appeared under your blog's name was whoever sent the last POST.

Fix. The hub no longer trusts the submission. When a register or remove arrives, it calls the site back at a new read-only address on that site (`/directory-verify.php`) and reads the listing **straight from the site itself** - proving the request genuinely comes from whoever controls that domain. A registration is honoured only if the site itself reports it wants listing, and the card is built from the site's own data, never the POST. A removal is honoured only if the site itself reports it is no longer listed. The most a forged submission can now do is make the hub re-read that site's own truth: it cannot inject content, and it cannot delist a blog that still lists itself. New members still wait for human review; a verified update from an approved blog refreshes in place.

Finding C - an image fetch could be aimed at the private network (CLOSED)

Severity: medium. Server-side request forgery, reachable by a connected hub.

When the hub pushes a post to a spoke, the spoke fetches the image from a URL the hub supplies. That one fetch - alone among every outbound request in the federation engine - followed redirects to a raw hostname. A crafted image URL could pass the safety check and then bounce the fetch onto an internal address (a machine on the spoke's own network, or the cloud-metadata address that hands out credentials).

Fix. It now resolves the address once, confirms it is genuinely public, locks the fetch to that exact address, and refuses to follow redirects - the same discipline every other fetch in the engine already used. This was the only one out of step, and combined with Finding A it would have been reachable by any spoke, which is why it was fixed in the same pass.

Finding D - the photo editor was shipping the fleet's credential code (CLOSED)

Severity: medium. Unnecessary attack surface in a tool that should have none.

SNAP SLAPPER is the standalone desktop photo editor. It was recently split out from THE HUB, and the question this audit asked was whether the split left anything behind. It did - not in the code, in the build recipe. The packaging step swept **every** shared module into the editor, including the fleet credential vault and the network/discovery code, none of which the editor ever runs. It only imports two small files.

To be clear about the blast radius: **no secret was ever embedded in the program.** The master key lives in the operating system's own keychain, not in the file, so there was nothing to extract. This was dead weight, not a leak - but a local photo editor has no business carrying the code that reads your website passwords, and any future mistake in that dead code would have become reachable from a tool that should never touch the network.

Fix. The build now bundles only the two files the editor actually uses, and the credential and network modules are explicitly excluded so a stray import can never drag them back in.

The small ones (CLOSED)

- **A tracking-beacon trick on the directory.** A listing's image links are drawn inside a stylesheet rule on the public directory page. A crafted link could have smuggled in a hidden style declaration - a tracking beacon that fires when the page is viewed. No script ran and it never reached the admin, but those links are now cleaned at the source: ordinary web addresses only, no stray punctuation.
- **A one-in-a-million registration race.** The one-time key used when a blog first joins a network was consumed a fraction too late, so two simultaneous join attempts could both slip through. It is now claimed atomically - exactly one wins. This lives in the join path, so it carries a live join test before it is called done.

Reviewed and sound - no findings

- **Federated messages from other servers** are cryptographically verified before anything is trusted, and the fix from audit 047 that stops one server posing as another is intact and still fails closed.
- **The rest of the engine's outbound requests** already pin the verified address and refuse redirects; Finding C was the single exception.
- **The remote-job runner** is limited to safe operations and every risky one requires the receiving blog to have opted in - a hub cannot silently push code or settings to a spoke that did not agree.
- **The self-healing cron** that runs jobs from ordinary page loads is throttled server-side with a database lock; no visitor request can force extra work.
- **Every database query** in the reviewed code is parameterised, and **admin forms** carry the automatic cross-site-request protection that has been site-wide since June.
- **SNAP SLAPPER's own code** launches external programs safely (no shell interpretation of filenames), stays inside the chosen folder for file operations, reads no credentials, makes no network calls, and preserves photo metadata on export by design.

What changed in how we work

The two high findings share a root: a control plane is only as safe as the check on *who is asking*, and both a shared secret and an open public form make that question easy to get wrong. The directory fix in particular moved the authority for what appears under a blog's name from "whoever posted last" to "the site itself, proven by a call-back" - the right place for it. This report was held back until 0.7.559 was actually live on the fleet, because publishing a working description of an unpatched hole is the opposite of protecting the people running these blogs. That is the whole point of this page.