SECURITY AUDIT FINDING — Multisite Mesh Roster Key Broadcast

SnapSmack Alpha (multisite mesh subsystem) Finding date: 2026-06-15 **Status: REMEDIATION SHIPPED in 0.7.260 "Ejector Seat" (2026-06-15).** Finding 1 fixed (roster strips keys + spoke self-heal); Finding 2 mitigated (consent gates + duplicate unguarded SSO handler removed); Finding 3 functional break fixed (SUYB uses correct key), at-rest scoped-key hardening DEFERRED; Finding 4 still OPEN. See "REMEDIATION SHIPPED" section below. Working finding (pre-PDF). Disclosed proactively per SnapSmack transparency policy. We are NOT hiding this — it was found in our own review and is recorded openly. **Severity: Critical** if the multisite mesh is deployed on a live hub+spoke fleet (network-wide admin compromise from a single leaked key). Confirm deployment state of the mesh on the Proxmox fleet to set "exploitable now" vs "design-stage." **Document authorship:** Prepared with AI assistance (Claude — Cowork build instance). All SnapSmack code is AI-produced; see the SPL Ethical Provenance Summary / hairy-muff.php. Finding raised by Sean ("what attack surface is the hub having spoke keys?") and traced in collaboration.

Summary

The multisite mesh distributes node **API keys inside the roster** that the hub serves to peers. Any party holding **one** spoke's `api_key_remote` can call the hub and receive the **`api_key_local` of every node in the network**. `api_key_local` is the key that passes the main Bearer gate on each node, which unlocks god-mode endpoints (full DB dump, admin SSO-token mint, code push). Net effect: **one leaked key from the least-protected spoke -> admin-level compromise of the entire fleet. Wormable.**

A second, separate item: a same-session change to the desktop backup tool (SUYB) wired the wrong key type and (a) does not authenticate — fail-closed, not exploitable — and (b) increased at-rest exposure of `api_key_remote` by writing it into desktop profile files. Documented here in full; remediation = revert + redesign.

FINDING 1 — Roster broadcasts `api_key_local` (CRITICAL) — verified first-hand

Chain (4 steps, no sophistication required):

1. Obtain one spoke's `api_key_remote`. This is the weakest-guarded secret in the system: it lives on every spoke, is stored plaintext on the hub, is returned by `suyb-data.php`, and (as of this session — see Finding 3) also sits in desktop profile files.
2. Call the hub `GET /core/multisite-api.php` route `ping` **or** `peers/list` with it.
   - `ping` validates the Bearer token against `api_key_remote` for any `role='spoke'` (`core/multisite-api.php:164`) and, if this install is a hub, returns the roster (`:192 -> :199`).
   - `peers/list` is hub-only (`:212`), validates against any active node's `api_key_remote` (`:224`), and returns the roster (`:232`).
3. The roster comes from `ms_build_roster()`, which **selects and returns every active node's `api_key_local`** (`core/mesh-helpers.php:72`, emitted at `:85`).
4. `api_key_local` is the key checked by the main inbound Bearer gate on every node (`core/multisite-api.php:250` — `WHERE api_key_local = ? AND status='active'`). With it, an attacker reaches the unrestricted endpoints below.

God-mode endpoints reachable with a harvested `api_key_local` (from the audit trace; the two with NO role gate are each, alone, full compromise):

- `multisite/backup/export` (`dump=full`) — **full SQL dump incl. user table / password hashes / all settings secrets. No role gate.**

- `multisite/auth/sso-token` — **mints a one-time admin SSO token -> full admin takeover. No role gate.**
- `multisite/updates/trigger`, `multisite/skins/reinstall` — **code push** (hub-role gated).
- `multisite/settings/push` — rewrites settings incl. `akismet_key`, `ai_key_*` (hub-role gated).
- `multisite/comments/action`, `multisite/blogroll/sync`, `multisite/disconnect`, `multisite/maintenance/set` — destructive writes.

Why it is over-built (not the intended design): Sean's design intent is a **star** — hub coordinates (stats up = read-only; features/updates down), each site runs its own work and writes its **own** DB; no site reaches into another's database. That intent is honored at the data layer: cross-site changes are authenticated API calls the receiving node executes with its own code (no raw cross-DB writes). The actual cross-site caller is the **hub**, which reads spoke keys **straight from its own `snap_multisite_nodes` table** (`smack-multisite-crosspost.php` is hub-only, `:25`, and reads `api_key_local` locally, `:32`) — it never needs the roster to carry keys. Spokes ingest peer keys (`mesh-helpers.php ms_ingest_roster`, ~`:135`) but no spoke-side consumer was found (crosspost is hub-only; `blogroll.php` only renders links). The key-bearing roster is redundant for the hub and unused by spokes — latent full-mesh capability that creates the exposure for no working feature.

Remediation (low-risk): `ms_build_roster()` must emit **names/URLs/roles only — never keys**. Hub->spoke calls keep working (hub reads keys from its own DB). Only the latent, unused spoke-to-spoke path stops. PENDING: one final sweep to confirm no spoke-side code reads ingested peer keys before removal. Topology decision (Sean): star confirmed = strip keys; full-mesh wanted = per-pair encrypted on-demand distribution instead of broadcast.

FINDING 2 — No role gate on `backup/export` and `auth/sso-token` (CRITICAL) — from audit trace

Independent of the roster, these two endpoints lack the `role` check the destructive write endpoints have. A DB dump and an admin-token mint should both require, at minimum, hub-role + ideally a scoped key. **Remediation:** add role gating; move the dangerous operations behind short-lived, just-in-time tokens (the existing one-time SSO token is the right shape to extend).

FINDING 3 — SUYB key wiring: wrong key type + at-rest exposure (this session's change) — verified

The same-session SUYB change ("auto-configure spokes from the hub") stored each spoke's `api_key_remote` and sent it for backups. Two problems:

- **Functional (fail-closed, NOT exploitable):** `api_key_remote` authenticates nothing SUYB calls. Backup endpoints check `tool_api_key` via `X-Snap-Key` (`core/api-auth.php:37,46`); `multisite/backup/config` checks `api_key_local`. `api_key_remote` != either, and the hub never holds a spoke's `tool_api_key`. Result: auto-configured profiles return 401. The feature does not work; it fails safe.
- **Exposure regression (real):** the change writes `api_key_remote` into SUYB profile files on the desktop (obfuscated, not encrypted). Given Finding 1, `api_key_remote` is the exact bootstrap key for the roster harvest. So this change moved a fleet-compromise bootstrap key onto the desktop. **Remediation:** revert the `api_key_remote` wiring (`hub_discovery.py` build_profiles, `main.py` discovery dialog, `backup_engine.py` session); redesign around a dedicated **backup-only, read-only scoped key** (can download a backup + report status, nothing else) so a desktop leak = "someone read a DB copy," not fleet takeover.

FINDING 4 — Keys stored plaintext; `tool_api_key` shown in UI (HIGH) — from audit trace

`api_key_local` / `api_key_remote` are plain `varchar(255)` stored raw (`multisite-api.php` handshake insert ~:120; `snapsmack_canonical.sql:882-883`). `tool_api_key` is stored raw (`install.php:1289`, `smack-settings.php`) and **rendered in full in the admin UI input** (`smack-settings.php:831`). **Remediation:** keys a node only *checks* -> hash (store hash, compare). Keys the hub must *replay* -> encrypt at rest, never display. Mask `tool_api_key` in the UI.

Remediation order (severity-first; Sean's "fix in order of severity")

1. **Finding 1** — roster carries no keys (after the no-spoke-consumer sweep).
2. **Finding 2** — role-gate `backup/export` + `auth/sso-token`; short-lived tokens for both.
3. **Finding 3** — revert SUYB `api_key_remote` wiring; redesign on a backup-only scoped key.
4. **Finding 4** — hash/encrypt keys at rest; mask in UI.
5. THEN defense-in-depth: automatic key lifetime / rotation (auto re-handshake, no human — manual reauth across the fleet will not happen; see backup-recovery continuity).

REMEDIATION SHIPPED — 0.7.260 "Ejector Seat" (2026-06-15), linted clean, pending push

Built and PHP-linted in the main session this same day. Six PHP files + the SUYB tool.

Finding 1 — FIXED (keystone). `ms_build_roster()` now selects/returns **names, URLs, and roles only — no `api_key_local`** (`core/mesh-helpers.php`). `ms_ingest_roster()` stores no key for learned peers and **self-heals**: on each hub sync it blanks any sibling `api_key_local` a spoke stored under the old code (`UPDATE ... SET api_key_local='' WHERE roster_source = <hub>`). The no-spoke-consumer sweep was completed — a repo-wide grep confirmed **every** `api_key_local` reader is a hub-side admin page reading its own DB where `role='spoke'`; nothing consumes an ingested peer key. Topology confirmed **star** by Sean. Zero working functionality lost.

Finding 2 — MITIGATED (not the full JIT-token redesign yet). Root cause of the missing SSO role gate was found and fixed: a **duplicate, unguarded `multisite/auth/sso-token` handler executed first and shadowed the role-gated one** — the duplicate is removed, so only the hub-role-gated handler runs. In addition, the four powerful inbound actions — `backup/export`, `auth/sso-token`, `updates/trigger`, `skins/reinstall` — are now behind **per-site consent gates** (`multisite_allow_backup` / `_sso` / `_update` / `_skin`), **default OFF**, each requiring the spoke owner to enable it with password + TOTP step-up. NOTE: because a leaked `api_key_local` resolves to `role='hub'` on the target, role-gating alone would not have closed Finding 1 — the keystone strip does; these gates are defence-in-depth + consent. Short-lived JIT tokens for export/SSO remain future work.

Finding 3 — FUNCTIONAL FIX SHIPPED; AT-REST HARDENING DEFERRED (documented deviation). The `api_key_remote` wiring was reverted. SUYB now uses **`api_key_local`** — the key the hub backup page already uses — read from the hub's own DB via `suyb-data.php`. This is a **deliberate deviation** from this audit's recommended remediation (a dedicated backup-only scoped key). Reason: Sean's explicit "fix it, keep functionality, ship today." Consequence: SUYB profile files on the desktop now hold `api_key_local` (a hub-authority key), so a **desktop compromise is still fleet-significant** until the scoped key lands. Partial mitigations now in place: (a) `backup/export` is consent-gated default-off; (b) key rotation (disconnect/rejoin mints fresh keys) neutralises anything leaked. **Residual risk accepted by owner; the `api_key_backup` scoped-key redesign (generate at handshake -> expose via suyb-data -> `backup/export` authenticates the scoped key only) remains the recommended next hardening.**

Finding 4 — OPEN. Keys still plaintext; `tool_api_key` still rendered in the UI. Unchanged this release. Recommended next, alongside the scoped key.

New operational control — key rotation. Because keys may already have spread to all spoke boxes over the ~1-month live window, the rollout includes a one-time **disconnect + rejoin of every spoke** after the fixed code is deployed, minting fresh `api_key_local/_remote` on each. This is what actually neutralises already-leaked keys; it only counts once every node runs the fixed (no-broadcast) code.

New auth model (durable). Auth is required to GRANT or REDUCE security (join a hub, enable a hub-permission, push settings from the hub, disable SMACKBACK) and is FREE for actions that INCREASE security (leave a hub, disable a hub-permission). Batched operations take ONE step-up, not one per item. SMACKBACK-disable keeps its own separate gate.

Honesty note

Finding 3 is a flub introduced in a prior session by the Cowork build instance and is recorded here in full rather than quietly reverted; its functional break is now fixed, and the deviation from the recommended scoped-key remediation is documented above rather than hidden. Findings 1, 2, 4 predate this session (mesh design). The roster chain (Finding 1) and the crosspost/hub-reads-locally evidence were verified firsthand from source; Findings 2 and 4 carry file:line from the audit trace and should be re-confirmed firsthand before the PDF is cut. The shipped remediation was PHP-linted clean (`php -l`, all six files) but not yet runtime-tested on the live fleet at the time of writing.