SECURITY AUDIT FINDING — SON OF A BATCH offline poster + Unzucker authoring routes

SnapSmack Alpha (Unzucker API — new GRAMOFSMACK authoring routes; SON OF A BATCH desktop suite) Finding date: 2026-06-25 **Status: REMEDIATION SHIPPED (same day, 2026-06-25), pre-PDF.** Findings 1, 2, 4, 5 FIXED; Finding 3 MITIGATED by 1+4 with at-rest encryption deferred (documented deviation, SUYB precedent). See "REMEDIATION SHIPPED" below. Original finding shipped in commit `7a4d86f7` + adjustable-cuts follow-up; remediation in a follow-up commit. Disclosed proactively per SnapSmack transparency policy — found in our own review of code we just wrote; not hidden. **Severity: High** if the `unzucker` Bearer key is exposed or over-shared (unbounded posting + image upload to an established gram site, with the owner-authorization step that protected such sites now bypassed for these routes). Lower if the key stays single-purpose and the gram site is low-value. **Document authorship:** Prepared with AI assistance (Claude — Cowork build instance), auditing code the same instance wrote this session. All SnapSmack code is AI-produced; see the SPL Ethical Provenance Summary. Audit requested by Sean ("we'll have holes") ahead of the Flickr import.

Summary

SON OF A BATCH adds an offline GRAMOFSMACK/solo poster to the SYBU desktop tool and three new server routes to the Unzucker API (`gram/upload`, `gram/post`, `gram/verify`). To make the tool usable on an established site (post a batch from a coffee shop), the two write routes were **deliberately exempted from the non-empty-site "import bazooka" lock** that otherwise forces an owner to authorize writes to a site holding > 5 items. The carousel-mode lock is kept; the Instagram import path is unchanged.

The headline issue is that the **only volume limit on these routes is a client-side soft cap** (a ~50-image warning in the desktop UI). The server imposes no cap, so a valid or leaked `unzucker` key can write unlimited posts and upload unlimited images to a live gram site with no owner step. Two supporting issues: the `images[].path` field in `gram/post` is not checked for directory traversal (inherited from the existing import route, replicated here — and inconsistent, because the client-thumb paths *were* hardened), and the key now sits at rest in desktop profile files with materially more power than before.

Scope of this pass

Audited firsthand (code written this session): `core/unzucker-api.php` authoring routes and guard changes; `tools/sybu/sob_offline.py`, `sob_post.py`, `sob_gram.py`, `sob_solo.py` (the desktop tool). SQL paths, auth gate, input validation, file handling.

NOT audited firsthand this pass (flagged for a dedicated review): the other server-touching changes since the last audit (0.7.260, 2026-06-15) — grid composer crop/zoom/split image pipeline (`smack-post-gram.php`, `smack-edit-carousel.php`, 0.7.299–0.7.300), Flickr views/provenance import (`flkr-fckr` + core 0.7.300–0.7.301), and the ohsnap-api CORS origin addition (`https://tauri.localhost`). See "Broader surface" at the end.

Line numbers below are approximate (the build sandbox served truncated file views; the route bodies were read firsthand but exact current line numbers must be re-confirmed against the real file before a PDF is cut).

FINDING 1 — Authoring routes have no server-side volume guard; the cap is client-only (HIGH) — verified firsthand

Chain:

1. The non-empty-site lock (`core/unzucker-api.php` ~`:193`) blocks writes to a site holding > 5 items unless the owner opened an import window in admin. It is the control that stops a single `unzucker` key from flooding an established site.
2. SON OF A BATCH adds `$uz_authoring_subs = ['gram/upload','gram/post']` (`:187`) and **skips that lock for them** (`if (!$uz_is_authoring) { ... }`, `:196`). Intentional — so a photographer can post to a live site without an admin round-trip — but it removes the only volumetric/authorization guard on these routes.
3. The replacement limit (~50 images/batch) lives **only in the desktop UI** (`sob_offline.SOFT_BATCH_IMAGE_LIMIT`, surfaced as a `messagebox` warning in `sob_gram.py`/`sob_solo.py`). A modified client, a script, or a second tool holding the same key ignores it entirely.
4. Net: any holder of a valid `unzucker` key can call `gram/upload` (store images) and `gram/post` (create posts) without bound and without owner consent. `gram/post` with N split images creates N+1 posts per call (amplification).

Impact: content flooding / defacement of a live gram site and storage exhaustion, with no owner-authorization step. The carousel-mode lock still confines this to GRAMOFSMACK installs; it does not limit volume.

Remediation (pick one, ideally both): - Add a **server-side** authoring budget per key per window (e.g. mirror the ~50 soft cap as a hard server cap, or a rolling rate limit), so the client cap is enforced where it counts. - Gate the authoring exemption behind an **owner-enabled "offline poster" permission** (same shape as the 0.7.260 `multisite_allow_*` consent gates, default OFF, password+TOTP to enable) so an established site opts in once rather than being implicitly open to any `unzucker` key.

FINDING 2 — `gram/post` `images[].path` not checked for traversal (HIGH) — verified firsthand

`gram/post` validates each `images[].path` for length (<= 500) and a `.jpg/.jpeg` extension but **does not reject `../` or a leading `/`**. The value is then (a) passed to `snapsmack_generate_thumbs($path, $site_root, ...)` which reads `$site_root . '/' . $path` — an arbitrary-file read / thumbnail oracle — and (b) stored verbatim in `snap_images.img_file`, which skins render — a stored path-poisoning vector. The defensive `$valid_thumb` closure I added *does* reject traversal in the client thumb paths (regex-anchored to `img_uploads/YYYY/MM/thumbs/[ta]_*.jpg` + on-disk check), so the hardening is **inconsistent**: thumbs are checked, the main image path is not.

This gap is **inherited from the existing `unzucker/posts` import route** (same length+extension-only check, with a code comment already acknowledging "storing an arbitrary string in img_file could poison rendering code"). It was replicated into `gram/post` rather than fixed.

Remediation: reject any `path` containing `..` or a leading `/`; canonicalize (`realpath`) and confirm the resolved path stays under `img_uploads/`; apply the same to the existing `posts` route while there.

FINDING 3 — `unzucker` key at rest on the desktop now grants unbounded authoring (MEDIUM) — verified firsthand

SON OF A BATCH reads the site URL + `unzucker` key from the SYBU connection profile (base64-obfuscated, **not encrypted** — the same storage as SUYB). Given Finding 1, that key now authorizes unlimited posting + image upload to the live gram site. So a desktop compromise moved from "can run an Instagram import that the site's >5 lock would have blocked without owner consent" to "can post freely to the live site." This mirrors the open SUYB key-at-rest finding (secaudit 2026-06-15, Finding 4).

Remediation: issue a **posting-scoped, rate-limited key** distinct from the general `unzucker` import key; encrypt desktop key storage at rest; fold into the broader key-at-rest hardening already recommended for the tool family.

FINDING 4 — `gram/upload` has no per-key rate limit or storage quota (MEDIUM) — verified firsthand

`gram/upload` accepts JPEGs up to 20 MB (`:~490`) with no per-key count or rate limit; combined with Finding 1's removed volume guard and `gram/post` split amplification, a key holder can exhaust `img_uploads/` storage. JPEG MIME is verified via `finfo` and the saved filename is sanitised, so this is a volumetric/DoS issue, not a file-type or path issue.

Remediation: per-key upload rate limit + a soft storage budget; folds into Finding 1's server-side budget.

FINDING 5 — `gram/verify` discloses any post's caption/status by ID (LOW / informational) — verified firsthand

`gram/verify?post_id=` returns `post_type`, `status`, `image_count`, `trigram_id` and `caption` for **any** post id on the site, not only those created by the calling key. Because the `unzucker` key is single-tenant and already has broad read access (`site`, `ping`, the import surface), this is not a privilege escalation — but it does turn the key into a clean post-enumeration/caption oracle. Noted for completeness.

Remediation: none strictly required while the key is whole-site; if scoped keys land (Finding 3), constrain `gram/verify` to posts the key authored.

What is sound (positive controls confirmed firsthand)

- **No SQL injection surface in the new routes** — every INSERT/SELECT uses prepared statements with bound parameters; no string-built SQL.
- **Auth gate applies to all new routes** — `unzucker_auth()` (Bearer `key_type='unzucker'`, SHA-256 compared) runs before routing (`:~149`); the new routes are not reachable unauthenticated.
- **Carousel-mode lock retained** — authoring still refuses any non-`carousel` install.
- **Client thumb paths ARE traversal-hardened** (`$valid_thumb`) and uploads are JPEG-verified via `finfo` with sanitised destination filenames.
- **No CSRF surface** — Bearer API, no cookie/session auth on these routes.
- **No schema change** — all columns pre-existed; nothing weakened at the DB layer.

Remediation order (severity-first)

1. **Finding 1** — server-side authoring budget and/or owner-enabled "offline poster" consent gate. (Closes the headline risk; also covers Finding 4.)
2. **Finding 2** — block `../`leading `/` + `realpath`-confine `images[].path` in `gram/post` AND the existing `posts` route.
3. **Finding 3** — posting-scoped rate-limited key + encrypt desktop key at rest.
4. **Finding 4** — per-key upload rate/quota (with Finding 1).
5. **Finding 5** — constrain `gram/verify` once scoped keys exist.

Broader surface since 0.7.260 (NOT audited firsthand — recommend a dedicated pass)

These shipped between the last audit (2026-06-15) and this one and touch server-side input/file handling; they were not reviewed firsthand here:

- **Grid composer crop/zoom/split image pipeline** (`smack-post-gram.php`, `smack-edit-carousel.php`, 0.7.299–0.7.300). Same image-path/thumbnail family as Finding 2 — check for the same traversal/validation parity. Session-authed admin pages, so lower external surface, but worth confirming.

- **Flickr views/provenance import** (`flkr-fckr` + core 0.7.300–0.7.301, "preserve full original image"). New ingest fields (view counts, provenance URLs) and image handling — confirm output encoding of provenance URLs and upload validation parity.
- **ohsnap-api CORS origin add** (`https://tauri.localhost`, commit `53326a95`). Confirm it is an exact-origin allow (not a wildcard, and not reflecting arbitrary origins with credentials).

REMEDIATION SHIPPED — 2026-06-25 (same day)

Built and edited in `core/unzucker-api.php`, `smack-api-keys.php`, and `tools/sybu/sob_post.py`. No schema change (settings rows are created on write).

Finding 1 — FIXED. Two-part: - **Owner consent gate.** Authoring onto an established site (> 5 items) now requires the owner to have enabled offline posting — setting `gram_authoring_enabled='1'`, flipped in **Admin -> API Keys** behind `reauth_verify` (password + 2FA), with a free disable. Empty/new sites stay free. This is a one-time, persistent opt-in (no per-session friction — the property Sean required) that restores owner authorization (`unzucker-api.php` POST guard; `smack-api-keys.php enable_offline_posting/disable_offline_posting`). - **Server-side volume budget.** `uz_authoring_budget()` enforces a rolling **300 images/hour** cap across `gram/upload` + `gram/post` (settings `gram_authoring_win_start/_count`), plus a **30-image per-call** ceiling on `gram/post`. The ~50 client cap is now explicitly UX only; the server is the control. Exceed -> HTTP 429.

Finding 2 — FIXED. `gram/post` rejects any `images[].path` that is absolute, contains `..` or NUL, and additionally `realpath`-confines it under `img_uploads/` (the file exists — it was uploaded via `gram/upload`). The existing `posts` import route gets the lexical guard (no `realpath`, since its files arrive out-of-band via FTP and may not exist at check time). The `$valid_thumb` client-thumb guard was already traversal-safe; parity restored.

Finding 3 — MITIGATED; at-rest encryption DEFERRED (documented deviation). With Findings 1+4 a leaked desktop key no longer yields unbounded authoring — it hits the consent gate (blocked on an established site unless the owner opted in) and the hourly budget. The desktop key remains base64-obfuscated, **not encrypted**, identical to the open SUYB key-at-rest finding (2026-06-15 Finding 4): real encryption needs a desktop key-management story to be worth more than obfuscation, and is deferred to the same future hardening. Residual risk: on a site where the owner HAS enabled offline posting, a desktop key leak allows posting up to the hourly budget until the key is rotated. Accept or revoke the `unzucker` key to neutralise.

Finding 4 — FIXED. Folded into `uz_authoring_budget()` — `gram/upload` counts against the same hourly image budget; the 20 MB/file cap and `finfo` JPEG check are retained.

Finding 5 — FIXED. `gram/verify` no longer returns the caption — existence, `post_type`, `status`, `image_count`, `trigram_id` only (all positive verification needs). No caption-enumeration oracle.

Client (`tools/sybu/sob_post.py`). `gram/upload`, `gram/post`, and `link_trigram` now surface the server's JSON message on 401/403/429, so the user sees "Offline posting is not enabled for this site. Turn it on in Admin -> API Keys" or the rate-limit notice instead of a generic rejection.

Verification: server edits to be `php -l`'d on real-FS (sandbox has no php and was serving truncated views); the desktop-tool edits py-compile clean. Not yet runtime-tested against a live install.

Honesty note

Findings 1–5 are in code this same Cowork instance wrote this session, audited the same day, and disclosed rather than quietly patched. Finding 1 (the client-only cap) is the hole Sean anticipated ("we'll have holes") and is a direct consequence of a deliberate design choice (the bazooka exemption), not an oversight — but the design choice shipped without its server-side counterpart, which is the finding. Finding 2 is a pre-existing

gap in the import route that was replicated rather than fixed. Route bodies were read firsthand; exact line numbers are approximate (truncated sandbox views) and must be re-confirmed against the real file before a PDF is cut. The new routes were `php -l` clean (Sean, real-FS) but not yet runtime-tested against a live install.