SNAPSMACK SECURITY AUDIT

Unzucker Attack Surface Review — #023

Date: 2026-06-07
Version: 0.7.218
Scope: core/unzucker-api.php · api.php · smack-api-keys.php · tools/unzucker/*
Auditor: Claude (Sonnet 4.6) + Sean McCormick

EXECUTIVE SUMMARY

Unzucker introduces a new authenticated REST API surface (3 endpoints), a key generation admin page, and a Python desktop client with FTP/SFTP upload capability. No critical or high severity findings were identified. Auth is correctly scoped per key_type and uses SHA-256 hashing with a multi-source header fallback for FastCGI environments. The primary concerns are: SFTP host key verification is disabled (AutoAddPolicy), img_path from the POST payload is stored without server-side validation, and the ig_parser does not constrain resolved image paths to the export root directory.

EOF MARKER / TRUNCATION CHECK

File	EOF Marker Present	Status
core/unzucker-api.php	// ===== SNAPSMACK EOF =====	✓ OK
api.php	// ===== SNAPSMACK EOF =====	✓ OK
smack-api-keys.php	<?php // ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/poster.py	# ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/main.py	# ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/ig_parser.py	# ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/ftp_upload.py	# ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/config.py	# ===== SNAPSMACK EOF =====	✓ OK
tools/unzucker/exif_writer.py	# ===== SNAPSMACK EOF =====	✓ OK

All 9 source files carry valid EOF markers. No truncations detected.

FINDINGS

UZ-01 — SFTP: AutoAddPolicy disables host key verification

MEDIUM

Location: `tools/unzucker/ftp_upload.py — SFTPTransport.connect()`

paramiko.AutoAddPolicy() silently accepts any host key, eliminating SFTP's man-in-the-middle protection. An attacker on the same network can intercept the SFTP session and exfiltrate the FTP password or substitute uploaded images.

Recommendation: Switch to RejectPolicy and pre-populate known_hosts, or use key-based auth. At minimum, document the risk in the UI so users understand plain SFTP over untrusted networks is not safe.

UZ-02 — img_path stored without server-side path validation

LOW–MEDIUM

Location: `core/unzucker-api.php — POST unzucker/posts, line ~235`

The img_path value from the JSON payload is trimmed and written directly to snap_images.img_file with no extension whitelist, no path prefix check, and no length cap. A holder of a valid Unzucker API key could craft a request inserting arbitrary path strings into the database. The images themselves are never fetched by this endpoint (they are pre-FTP'd), so there is no server-side file access risk — but crafted paths could poison the image table or trigger unexpected behaviour in front-end rendering code.

Recommendation: Add a basic path sanity check: verify the path is non-empty, has a jpeg/jpg extension, and optionally starts with the expected remote_base prefix. Enforce a VARCHAR(500) length cap at the application layer to match the schema.

UZ-03 — ig_parser: resolved image paths not constrained to export root

LOW–MEDIUM

Location: `tools/unzucker/ig_parser.py — parse(), line ~147`

Media URIs from posts_1.json are resolved with os.path.normpath(os.path.join(export_root, uri)). normpath resolves '..' but os.path.join with an absolute URI would discard export_root entirely. A maliciously crafted export could reference arbitrary files on the user's machine (e.g. another image file at a known path), causing Unzucker to resize and FTP-upload that file to the server. Practical risk is low since the user controls the export file, but not zero (poisoned export from a third party).

Recommendation: After resolving abs_path, assert os.path.abspath(abs_path).startswith(os.path.abspath(export_root)) and skip the image with a warning if it fails.

UZ-04 — CSRF on API key generation and revocation (pre-existing, now higher-stakes)

LOW–MEDIUM

Location: `smack-api-keys.php — generate / revoke / delete POST handlers`

The key management forms carry no CSRF token. This is a pre-existing issue present across several admin pages. With Unzucker keys now granting write access to snap_posts and snap_images, the impact of a CSRF-triggered key generation has increased — an attacker could silently generate a valid Unzucker key by tricking a logged-in admin into loading a malicious page.

Recommendation: Pre-existing issue. Not introduced by Unzucker. Track for the broader CSRF hardening pass. Short term: the admin session requirement limits exposure.

UZ-05 — FTP (plain) transmits credentials in cleartext

LOW

Location: `tools/unzucker/ftp_upload.py — FTPTransport`

Plain FTP sends username and password over the wire unencrypted. The SFTP option exists and is the correct choice for public networks. No enforcement pushes users toward SFTP.

Recommendation: Add a UI warning when plain FTP is selected, recommending SFTP for non-LAN transfers. Consider defaulting the protocol selector to SFTP.

UZ-06 — API key and FTP password stored base64-encoded (not encrypted) in unzucker.ini

LOW

Location: `tools/unzucker/config.py — _encode() / _decode()`

Credentials in unzucker.ini are base64-encoded, which the file itself documents as 'not plaintext at a glance.' On a shared or compromised machine, any user can read and decode the file trivially. The API key grants write access to the

SnapSmack database; the FTP password grants full server file access.

Recommendation: Pre-existing design decision. Document the risk for users. Future improvement: use OS-level credential storage (keyring on Windows/macOS/Linux) for secrets.

UZ-07 — No rate limiting on unzucker/posts endpoint

LOW

Location: `core/unzucker-api.php — POST unzucker/posts`

An authenticated Unzucker key can POST to /posts as rapidly as the server allows. No per-key throttle or daily quota is enforced. A compromised or stolen key could be used to flood snap_posts and snap_images with arbitrary records.

Recommendation: Low priority given auth requirement. Consider a simple per-key rate limiter (e.g., max N posts per hour stored in a transient or DB table) for a future hardening pass.

UZ-08 — tags array depends on snap_sync_tags() for SQL safety

INFORMATIONAL

Location: `core/unzucker-api.php — line ~277; core/snap-tags.php`

The tags array from the POST body is assembled into $tag_string and passed to snap_sync_tags(). Tag safety relies entirely on snap_sync_tags() using prepared statements internally. Not audited in this pass — confirm snap_sync_tags() uses parameterised queries throughout.

Recommendation: Verify snap_sync_tags() in a dedicated tags-subsystem audit.

UZ-09 — Dead code: embed() and embed_inplace() in exif_writer.py

INFORMATIONAL

Location: `tools/unzucker/exif_writer.py — embed() / embed_inplace()`

Two functions (embed and embed_inplace) are defined but never called. Only build_exif_bytes() is used by poster.py. These are leftovers from the ft-batch-poster fork. Not a security risk but adds surface to maintain.

Recommendation: Remove or mark as unused to keep the codebase clean.

UZ-10 — GET unzucker/site exposes full taxonomy to key holder

INFORMATIONAL

Location: `core/unzucker-api.php — GET unzucker/site`

Returns all category and album names and IDs. This is required for the UI dropdowns and is intentional. Documented here for completeness — any Unzucker key holder can enumerate the site's full taxonomy without posting anything.

Recommendation: Intended behaviour. No action required.

FINDINGS SUMMARY

ID	Severity	Title	Status
UZ-01	MEDIUM	SFTP AutoAddPolicy — host key verification disabled	OPEN
UZ-02	LOW–MEDIUM	img_path stored without server-side validation	OPEN
UZ-03	LOW–MEDIUM	ig_parser paths not constrained to export root	OPEN
UZ-04	LOW–MEDIUM	CSRF on key management (pre-existing)	OPEN (pre-existing)
UZ-05	LOW	Plain FTP transmits credentials in cleartext	OPEN
UZ-06	LOW	Credentials stored base64-only in unzucker.ini	OPEN (by design)
UZ-07	LOW	No rate limiting on unzucker/posts	OPEN
UZ-08	INFORMATIONAL	snap_sync_tags() SQL safety unverified	OPEN
UZ-09	INFORMATIONAL	Dead code: embed / embed_inplace in exif_writer.py	OPEN
UZ-10	INFORMATIONAL	GET site exposes full taxonomy to key holder	INFORMATIONAL

Priority order for remediation: UZ-01 (SFTP host key) → UZ-03 (path traversal in parser) → UZ-02 (img_path validation) → UZ-04 (CSRF, pre-existing) → remaining Low items. Informational items are low urgency but should be addressed before a wider public release.