SECAUDIT 040 - FLKR FCKR: credential-at-rest, credential-in-transit, and server-side trust in client-supplied paths

SECURITY AUDIT / PUBLIC REMEDIATION RECORD

Field	Value
Audit ID	2026-08-06-040
Date	2026-08-06
Severity	**MEDIUM** - the shared step-up helper transmits the operator's **account password and a live TOTP code** to whatever URL is configured, with no scheme check, so a `http://` site URL puts full account credentials on the wire in cleartext (Finding B, and it affects every tool in the family, not just this one). The scoped import key is base64-obfuscated at rest (Finding A). Two lower items: the server stores client-supplied image paths with no containment and later `unlink()`s them (C), and accepts an unvalidated URL that becomes a rendered link `href` (D).
Component	`tools/flkr-fckr/` desktop client - `config.py` (key at rest), `main.py` (settings/step-up drive), `poster.py` (API transport), `flickr_parser.py` (export parsing), `image_prep.py` (EXIF/geo, file copy), `checkpoint.py`; `tools/_shared/snap_stepup.py` (**shared** step-up client); `core/flkrfckr-api.php` (server handler)
Status	**ALL FOUR FINDINGS CLOSED** (2026-08-06, see section 9). A was closed by porting SUYB's credential vault; the client needs a rebuild to ship. EXIF/GPS preservation is a **settled** decision, recorded in section 6 - not a finding, not open.
Reporter	Sean (chose FLKR FCKR as the next desktop-client audit) + Claude (walked the client, the shared step-up helper, and the server handler end to end)
Related	**037 (sibling desktop client SUYB - same credential-at-rest class, closed by a scrypt+Fernet vault; its deferred "HTTP admin login" sub-item is this audit's Finding B, now seen twice)**, 039 (GYSS desktop trust boundary - added the `http://` warning this tool lacks; its key-expiry sweep already fixed `flkrfckr-api.php`), 036 (SmackPress desktop credentials), 034 (path discipline on extraction), 024A (comment/caption escape-first - the reason Finding D is not an XSS), 035 (client-IP trust boundary - correctly honoured here). FLKR FCKR version at audit: **0.7.25**.
Disclosure	No exploitation known. Finding B requires the operator to configure a plaintext-`http://` URL **and** an attacker on the network path. Finding A requires local read access to the app folder. Findings C and D are **post-authentication**: they require a valid `flkrfckr` key *and* an open step-up window, which in normal operation means the site owner.

1. Summary

FLKR FCKR migrates a Flickr data export into a SMACKONEOUT (photoblog) install. It parses the unzipped export folder on the operator's machine, prepares each image locally, uploads over HTTPS multipart, and creates the `snap_images` rows, albums, collections and imported comments through `core/flkrfckr-api.php`.

It holds no Flickr credentials. This is worth stating plainly because the tool's name suggests otherwise, and because the pre-audit assumption was wrong: FLKR FCKR never talks to Flickr. It reads the offline export Flickr hands you. The only secret it stores is the scoped `flkrfckr` SnapSmack API key.

The authorization design is the strongest part of the tool and deserves saying first: the API key is explicitly *not* a credential. Every write requires an active, time-boxed, per-user window opened only by password **plus** TOTP, with no password-only fallback, no empty-site free pass, and no auto-slide. A stolen key with no open window can list album names and nothing else. That model is better than what most of the family had a month ago, and section 6 records the rest of what held up.

Four findings, none of them in that authorization model.

The meaningful one (**Finding B**) is in the machinery that *opens* the window. `tools/_shared/snap_stepup.py` POSTs `{username, password, totp_code}` to `{site_url}/api.php` with no check that `site_url` is HTTPS. A site URL typed as `http://` sends the operator's account password and a live authenticator code across the network in the clear. GYSS got exactly this warning in SECAUDIT 039 - but GYSS only leaks an API key that way, and FLKR FCKR leaks the password behind it. The helper is shared by FLKR FCKR, Unzucker, GYSS, SUYB, SYBU and Oh Snap, so one fix in one file closes it for all six.

Finding A is the family's recurring habit: the API key sits in `flkrfckr.ini` next to the executable, base64-encoded, with no file-permission tightening. Narrower blast radius than SUYB's (no FTP password, no admin password, no cloud refresh token) and materially blunted by the step-up window - but a key stolen *during* an open window is a working import credential.

Two lower findings sit on the server side, both in the same shape: the handler trusts strings the client sent it. **Finding C** - `img_file` and the thumbnail paths are stored verbatim, and `snap_manage_delete_image()` later feeds them straight to `unlink()`. **Finding D** - `author_url` is stored unvalidated and rendered as a link `href`; `flkrfckr-api.php` is the *only* writer of that column anywhere in the codebase.

Section 6 records the EXIF/GPS handling as a **settled product decision** - the importer preserves what the photographer's archive contains and does not edit it. Section 7 records a functional defect found on the way through: imported photos display no EXIF at all, because the client writes different key names than the skins read - which matters precisely *because* of the section 6 decision.

2. Finding B - step-up sends the account password and TOTP code with no HTTPS requirement (MEDIUM)

2.1 The code

`tools/_shared/snap_stepup.py:53`:

```
url = base_url.rstrip('/') + '/api.php'
resp = requests.post(url, params={'route': route},
headers={'Authorization': f'Bearer {api_key}', ...},
data=json.dumps({'username': username,
'password': password,
'totp_code': totp_code}), ...)
```

`base_url` arrives from `config.py`'s `site_url`, which is a free-text entry box in the settings bar (`main.py:237`). Nothing anywhere - not the entry widget, not `config.save()`, not `FlkrDckrClient.__init__`, not this helper - requires a scheme, checks for `https://`, or warns on `http://`. The in-app help says to enter "e.g. https://myphotoblog.com", which is guidance, not enforcement.

TLS verification itself is intact: every request goes through `requests` with the default `verify=True`, and there is no `verify=False` or `CERT_NONE` anywhere in this tool. The gap is not weak TLS. It is **no TLS at all** when the operator types `http://`.

2.2 Impact

On a plaintext URL, an attacker on the network path (hostile Wi-Fi, compromised router, ISP segment) reads, in order:

- the **account password** - not an application password, the login password for

the SnapSmack admin user the key is bound to;

- a **live TOTP code**, valid for its remaining window and replayable within it;
- the **Bearer key** on that request and every subsequent one;
- the entire import stream - titles, descriptions, comments and full-resolution

originals.

Password plus a live second factor is the whole authentication set. This is a strictly larger exposure than the one 039 documented for GYSS, where the same mistake leaks a scoped key that cannot write on its own.

Severity is held at MEDIUM rather than higher because it requires the operator to have configured plaintext HTTP *and* an attacker positioned on the path. But the tool does nothing to discourage the first condition, and a photoblog owner setting up on a LAN or a staging box is exactly the person who types `http://`.

2.3 Recommendation

In `snap_stepup.request_authorization()`, before the POST:

- refuse a non-`https://` `base_url` outright, with `localhost` / `127.0.0.1` / `::1` exempted (a local install has no network path to attack). Credentials are categorically different from an API key - this one should be a refusal, not a warning;
- for the non-credential calls in `poster.py`, mirror 039's GYSS treatment: warn once, clearly, that the key travels in the clear, and require confirmation.

Because the helper is shared, fixing it here fixes FLKR FCKR, Unzucker, GYSS, SUYB, SYBU and Oh Snap in one edit. This also closes the sub-item SECAUDIT 037 deferred for `hub_discovery.py` - that is the same defect in a second tool, which is the signal that it belongs in the shared layer.

3. Finding A - scoped API key base64-obfuscated on disk (MEDIUM)

3.1 The code

`config.py` is candid in its own docstring - *"The API key is stored base64-obfuscated - not encrypted, just not plaintext at a glance"* - which is more honest than SUYB's `_enc` field naming was before 037. The storage:

```
def _encode_pw(plain: str) -> str:
return base64.b64encode(plain.encode()).decode()
```

`flkrfckr.ini` is written with `open(_config_path(), 'w')` and no permission tightening; SUYB moved to owner-only `chmod 0600` in 0.7.18 and FLKR FCKR never got that pass. The file sits next to the executable by design (portable-app convention, same as Unzucker and SUYB).

3.2 Impact - and why it is narrower than 037

What reading the folder yields, and what it does not:

Scoped `flkrfckr` API key	**Exposed** (base64)
Site password / admin password	Not stored - only the *username* is persisted, for prefill (`main.py:919`)
TOTP secret or codes	Not stored, ever
Flickr credentials	**None exist** - the tool reads an offline export
FTP/SFTP credentials	None - uploads go over the same HTTPS API
Cloud OAuth refresh tokens	None
Checkpoint file	Holds `site_url` and Flickr-ID->image-ID mappings; **no secrets**
Log files	No secrets written; the key is never logged

So the worst case is one scoped key, and the step-up window means that key alone **cannot write** - it can call `flkrfckr/ping` and list album names. The real exposure is a key stolen while a window is open (default 4 hours, hard cap 24), during which it is full import-write access to the photoblog: create images, albums, collections, and auto-approved comments.

SECAUDIT 039 (sweep) additionally now enforces `expires_at` on this handler, so a stolen key dies with the mandated ≤4-week lifetime rather than living forever.

3.3 Recommendation

The fix already exists in the family and does not need designing: SUYB 0.7.19's `secret_vault.py` (scrypt N=32768/r=8/p=1 -> Fernet). It was built for exactly this constraint - portable, no machine-bound keychain, passphrase at unlock - and FLKR FCKR has one short-lived secret rather than SUYB's five, so the port is smaller.

Whether it is worth a passphrase prompt on a tool the operator runs once, for a key they are told to revoke when the import finishes (the help text already says so, twice), is an owner call. The floor, which is not a call and should happen either way: `chmod 0600` on `flkrfckr.ini`, best-effort and wrapped so it never breaks on FAT/Windows - the same treatment SUYB 0.7.18 applied.

4. Finding C - server stores client-supplied image paths uncontained, then deletes them (LOW)

`core/flkrfckr-api.php:393` takes `img_file` from the request body and inserts it verbatim (line 516); `img_thumb_square` / `img_thumb_aspect` the same. Nothing requires those values to be the ones the server itself just returned from `flkrfckr/upload`, to be relative, or to live under `img_uploads/`.

`smack-manage.php:24-35` then trusts the stored value:

```
$stmt = $pdo->prepare("SELECT img_file FROM snap_images WHERE id = ?");
...
if ($img && !empty($img['img_file']) && file_exists($img['img_file'])) {
$pi = pathinfo($img['img_file']);
$td = $pi['dirname'] . '/thumbs';
@unlink($td . '/t_' . $pi['basename']);
@unlink($td . '/a_' . $pi['basename']);
@unlink($img['img_file']);
}
```

A record created with `img_file` set to a traversing or absolute path deletes that file when the owner later deletes the image in admin - a stored, delayed, owner-triggered arbitrary unlink, bounded by the web-server user's permissions. The derived `thumbs/t_` and `a_` siblings extend it slightly.

This is **post-authentication**: it needs a valid key *and* an open step-up window, so in normal operation the only person who can do it is the owner. It is recorded because the server should not extend the reach of a tampered client or an intercepted response - the same principle 037 applied to `restores_to` and 034 applied to skin ZIP extraction.

Recommendation (unconditional, no tradeoff): validate on insert - reject absolute paths, drive letters, UNC, NUL bytes and any `..` segment; require the normalized path to stay under `img_uploads/`. Same rule for both thumbnail fields.

5. Finding D - unvalidated `author_url` becomes a rendered link href (LOW)

`flkrfckr/comments` (line 745) takes `author_url` and stores it as `snap_community_comments.guest_url` (line 761). `core/community-component.php:313` renders it:

```
<a class="ss-commenter ss-commenter-link"
href="<?php echo htmlspecialchars($c['guest_url']); ?>"
target="_blank" rel="noopener noreferrer">
```

`htmlspecialchars()` is correct and prevents attribute breakout - this is the 024A escape-first discipline working as intended - but it does not stop a `javascript:` scheme in an `href`. The client only ever constructs

`https://www.flickr.com/people/<nsid>/` (`flickr_parser.py:341`), so this is not reachable through normal use; it is reachable by anything that can call the API with a window open.

Two things make it worth fixing rather than filing under "the client is well-behaved":

1. `core/flkrfckr-api.php:761` is the **only** writer of `guest_url` in the

entire codebase - the public comment path (`process-community-comment.php`) never sets it. So this one import route is the sole source of a rendered link href on a photoblog, and it validates nothing.

2. The *content* inside that URL is third-party data - the NSID comes from the

Flickr sidecar, i.e. from whoever commented on your photos.

Recommendation: `filter_var($author_url, FILTER_VALIDATE_URL)` plus an explicit `http`/`https` scheme allowlist on insert; store NULL otherwise.

Related, recorded as a decision rather than a finding: imported comments are inserted `status = 'visible'` - auto-approved, bypassing whatever moderation the site otherwise applies. That is defensible (they were already public on Flickr, and the import is owner-initiated), but it should be a stated decision rather than an implicit one. The comment *body* is safe: the client runs `html.unescape()` on it (`flickr_parser.py:336`), which actively reverses Flickr's escaping, and the only reason that is not stored XSS is that the renderer escapes first (line 324). Worth knowing that the escape-first rule from 024A is what is holding here.

6. EXIF and GPS preservation - settled decision (2026-08-06, Sean)

GPS and EXIF are preserved, deliberately. This is not a finding and is not open. Recorded here so a later audit does not re-raise it, and so the behaviour is documented rather than incidental.

The behaviour: `image_prep.prepare()` copies JPEG/PNG/WebP sources **byte-for-byte** (`shutil.copy2`, line 172), so the file published on the site is the photographer's original with its complete EXIF - GPS, camera, lens, serial number, copyright fields - intact. Separately, `_build_exif_json()` merges Flickr's `geo` lat/lon into `img_exif` (lines 105-107) whenever the sidecar carries it.

The decision, in Sean's terms: most photographers do not regard location data in their own archive as a leak, and **a migration tool does not get to decide what data is imported**. FLKR FCKR's job is to move an archive from Flickr to SnapSmack without editing it. Byte-for-byte is also what preserves image quality (no re-compression of an already-compressed original), so the same copy path serves both goals. Stripping metadata would be the tool making an editorial choice about someone else's work.

Two facts recorded alongside it, neither of them a reason to revisit:

- No skin displays the coordinates. Every skin reads a fixed allowlist of EXIF

keys (`camera`, `lens`, `focal`, `film`, `iso`, `aperture`, `shutter`, `flash`

- e.g. `skins/aurora/layout.php:124-134`), and `latitude`/`longitude` appear in

no PHP in the codebase. The DB copy is stored, not rendered.

- Photos marked private on Flickr import as drafts by default (the `Private ->`

setting, honoured at `poster.py:480`), but the file itself is uploaded to `img_uploads/YYYY/MM/` and reachable by direct URL - draft status hides the listing, not the bytes. That is how every image in SnapSmack works, not something this importer introduces.

The live consequence of this decision is **section 7**: the camera data the importer is preserving currently reaches no skin, because the key names do not match core's. That is worth fixing on the strength of this decision, not despite it.

7. Functional defect found in passing (not a security finding)

Imported photos display **no EXIF at all** in any skin. The client writes PIL's raw tag names into `img_exif` - `Make`, `Model`, `LensModel`, `FNumber`, `ExposureTime`, `ISOSpeedRatings`, `FocalLength`, `Flash`, `DateTimeOriginal` (`image_prep.py:70-92`). Core's own ingest writes lowercase semantic keys - `camera`, `lens`, `focal`, `iso`, `aperture`, `shutter` (`core/image-ingest.php:292-297`) - and that is what the skins read. The names do not match, so every field resolves empty.

The fix is a rename in `_read_exif()`'s output mapping to match `core/image-ingest.php`. Flagged here because a Flickr archive import is exactly the case where camera data matters, and because it is the same file as Finding E.

8. What was already correct (and stays)

Recorded so the findings do not obscure what held up.

Property	Where	Verdict
Import key is session continuity, not a credential - every write needs an active per-user leased window	`flkrfckr-api.php:294-310`	Correct - the strongest part of the design
Step-up is password **+** TOTP, always; no password-only fallback; 2FA enrolment required	`flkrfckr-api.php:250-261`	Correct
Window can only be opened for the user the key is bound to - no impersonation	`flkrfckr-api.php:246`	Correct
Legacy keys with NULL `user_id` are refused for all writes	`flkrfckr-api.php:304`	Correct
Row owner forced to the key's user; client-supplied user ignored	`flkrfckr-api.php:521`	Correct
Key expiry + `key_type` scoping enforced	`flkrfckr-api.php:64-83`	Correct (closed by 039's sweep)
Bearer format constrained to 64 hex; key stored server-side as SHA-256	`flkrfckr-api.php:60-63`	Correct
Auth failures IP-rate-limited, 5 strikes -> 7-day ban, via `snap_ip_is_bannable`	`flkrfckr-api.php:132-154`	Correct - honours the 035 boundary
Client IP resolved through the single trusted accessor (not raw `REMOTE_ADDR`)	`flkrfckr-api.php:116-121`	Correct - 035 discipline
Install-mode lock: writes refused unless `site_mode === 'photoblog'`	`flkrfckr-api.php:301`	Correct
Every SQL statement parameterised	handler-wide	Correct - no injection surface found
Upload MIME-sniffed with `finfo`, extension forced from detected type, client filename stripped of path chars, 25 MB cap	`flkrfckr-api.php:627-647`	Correct
Client-supplied thumbnails independently MIME-checked and size-capped; all-or-nothing, never a half set	`flkrfckr-api.php:689-714`	Correct

Property	Where	Verdict
`img_uploads/.htaccess` blocks PHP execution (written at install, restored by recovery)	`install.php:1413`, `core/recovery-engine.php:417`	Correct
Comment rendering escapes first - body, guest name and href all through `htmlspecialchars`	`core/community-component.php:297,313,324`	Correct - 024A discipline holding
Export parsing is index-based (`os.listdir` of the chosen folder), so a crafted sidecar cannot traverse out	`flickr_parser.py:121-179`	Correct - no traversal
Name sidecar and all export JSON are data-only (`json.load`), no code execution on load	`flickr_parser.py:294-308`	Correct
Checkpoint is JSON via temp-file + atomic `os.replace`; no `pickle`; holds no secrets	`checkpoint.py:143-154`	Correct
TLS verification left at `requests`' default everywhere; no `verify=False` in the tool	tool-wide	Correct
Password and TOTP never persisted - only the username, for prefill	`main.py:918-923`, `config.py:55-57`	Correct
No secrets written to the log file; logs pruned at 14 days	`main.py:61-72`	Correct
`subprocess` call is list-form with a constant opener and a constant directory	`main.py:710-712`	Correct - no injection
Duplicate detection keyed on `flickr:<id>`, so re-runs are idempotent	`flkrfckr-api.php:440-452`	Correct

9. Remediation applied (2026-08-06)

Finding B - CLOSED. `tools/_shared/snap_stepup.py` now refuses to transmit credentials over a channel that would expose them. `_insecure_reason()` requires `https://`, exempting only loopback (`localhost`, `127.0.0.0/8`, `::1`) where there is no network path to sit on. A scheme-less URL is refused with the correction shown, and a non-http(s) scheme is refused outright.

The check runs in **two** places, deliberately. In `request_authorization()` it guards direct callers. In `authorize_interactive()` it runs **before the password dialog is shown** - refusing afterwards would mean the operator had already typed their password for a destination the tool was never going to send it to, and the retry loop would then re-prompt forever against an error no amount of retyping could fix.

This is a hard refusal, not the warn-and-confirm SECAUDIT 039 gave GYSS. The difference is what is on the wire: GYSS exposes a scoped key that cannot write on its own, this endpoint exposes the account password and a replayable TOTP code.

Because the helper is shared, this closes the same gap in **Unzucker, GYSS, SUYB, SYBU and Oh Snap** simultaneously, and resolves the `hub_discovery.py` sub-item SECAUDIT 037 deferred.

Findings C and D - CLOSED. Both validators now live in a new side-effect-free `core/api-input-safety.php` (no DB, no session, no headers, no includes) so every desktop-tool handler can adopt them and so they are directly testable - the same shape as `core/client-ip.php`, the SECAUDIT 035 boundary helper.

- `snap_api_safe_upload_path()` rejects absolute, drive-letter, UNC, backslash,

NUL-bearing, empty-segment, dot and dot-dot paths, and requires the value to sit under `img_uploads/` with a conservative charset. Root containment alone was explicitly **not** sufficient: `core/db.php` never leaves the install and is still catastrophic when passed to `unlink()`. The rule enforced is the real invariant - these columns may only name a file the upload endpoint itself wrote. Applied to `img_file` and both thumbnail columns.

- `snap_api_safe_link()` requires a well-formed `http`/`https` URL and returns

NULL otherwise, so a rejected value stores as NULL and the commenter's name renders unlinked. `htmlspecialchars()` at the render site stops attribute breakout but not a `javascript:` scheme; escaping is not scheme validation.

Finding A - CLOSED. The API key is now encrypted at rest. `tools/_shared/snap_vault.py` is a parameterised lift of the vault SECAUDIT 037 built for SUYB 0.7.19 - scrypt-derived master key (N=32768, r=8, p=1), Fernet sealing, passphrase never written. `init()` scopes both the keyring entry and the verifier per tool, so a `vault.meta` copied between tools fails closed rather than half-opening. FLKR FCKR gained a **Key** control for on/off/re-key, with an optional machine-bound cache of the unlock key so the passphrase is not demanded on every launch; that cache never travels with the folder.

The lifecycle is read-then-switch-then-write in both directions, so an interruption cannot strand the key, and a locked vault **raises** rather than silently rewriting the key in the weaker base64 form. Legacy base64 keys keep loading, so no existing install is stranded. Verified by `tools/flkr-fckr/tests/test_vault.py` (12 tests, all passing), including that no plaintext or base64 residue of the key remains in `flkrfckr.ini` after enabling, that a tampered ciphertext is rejected, and that a full on -> re-key -> off -> on cycle never loses the key. The GUI was constructed end to end in a smoke test with the Key window built.

SUYB was deliberately not migrated onto the shared module - it is shipped, tested, and wired through four other modules; converging it is worth doing as its own change with its own test pass, not as a side effect of this one.

The floor, applied alongside it. `config.py` now writes `flkrfckr.ini` with `chmod 0600`, best-effort and swallowed so a permissions failure can never cost the operator their settings. Stated plainly because it would be easy to over-read: this is a real control on Linux, **near-cosmetic on Windows** (the call succeeds but NTFS ignores the POSIX bits - the mode still reads `0666` afterwards), and it does **not** make the key confidential at rest. It is still base64, which is an encoding. The `secret_vault.py` port remains the actual fix and remains an owner decision.

Applying it surfaced a related gap worth recording, because it is the same credential seen from the other side. `.gitignore` matched `tools/*/config.ini`, which covered only the tools that happen to use that exact filename. It did not cover:

- `tools/flkr-fckr/flkrfckr.ini` - the scoped `flkrfckr` API key;
- `tools/smackattack-scanner/gobsmacked-scanner.ini` - an `api_key` **and** a

`db_password`.

Running either from source would therefore leave a live credential in the working tree as an untracked file, one `git add -A` from being committed to a public repository. (`tools/unzucker/unzucker.ini` was checked and is **not** affected - Unzucker keeps its secrets in the OS keyring and its ini holds only URL and paths. It is the pattern the others should follow.)

No exposure occurred. `git log --all --diff-filter=A -- 'tools/*/*.ini'` returns empty across every branch: no tool `.ini` has ever been committed, and none of these files exist on the operator's disk. This was a latent trap, not an incident, and **no key rotation is required**. The rule now matches every `.ini` under `tools/` rather than naming files one at a time, plus import checkpoints and the dated logs; verified that no `.ini` was previously tracked, so nothing was orphaned by the change.

Verification. `tests/api-input-safety-regression.php` - **48 checks, passing**: 5 legitimate upload paths accepted; 23 hostile ones rejected (traversal, absolute, drive-letter, UNC, NUL, empty segment, shell metacharacters,

quotes, prefix lookalikes, wrong-case prefix, over-length); 4 legitimate links accepted; 11 hostile ones rejected (`javascript:` in two cases, `data:`, `vbscript:`, `file:`, newline-split scheme, CRLF header injection, protocol-relative, over-length). The suite also asserts the handler still calls all three path checks and the link check, and that the Python HTTPS guard is present in both of its call sites, so a PHP-only run still fails loudly if the Python side is reverted.

The HTTPS refusal was exercised against 12 URL shapes (https, trailing slash, uppercase scheme, `localhost:8080`, `127.0.0.1`, `127.0.1.1`, `[::1]`, plaintext remote, LAN-over-http, scheme-less, `ftp://`, whitespace-padded) - all correct - and a direct call to a plaintext remote returns `ok=False` **without making a network request**.

10. Recommended fix order

1. ~~**Finding B** - `snap_stepup.py` HTTPS requirement.~~ **DONE** (section 9).
2. ~~**Findings C and D** - server-side path containment and URL scheme validation.~~

DONE (section 9).

3. ~~**Section 7** - EXIF key names.~~ **DONE** - client rewritten and a REBUILD EXIF

maintenance pass added to recover already-imported photos without re-importing.

4. ~~**Finding A** - credential vault + `chmod 0600` floor + the `.gitignore` gap.~~

DONE (section 9). Nothing in this report remains outstanding.

~~Also still open: whether to warn on `http://` for the non-credential calls.~~ **DONE** - Connect and Start Import both warn and require confirmation now.

11. Verification

- Every file in `tools/flkr-fckr/` was read in full (3,295 lines across seven

modules), plus `tools/_shared/snap_stepup.py` and the whole of `core/flkrfckr-api.php`.

- Finding C was confirmed by tracing the stored value to its consumer:

`flkrfckr-api.php` insert -> `snap_images.img_file` -> `snap_manage_delete_image()` `unlink()`.

- Finding D's "sole writer" claim was verified by grepping `guest_url` across

every PHP file in the repo: six hits, five of them the reader in `community-component.php`, one the writer in `flkrfckr-api.php`.

- Section 6's "not displayed" claim was verified against the skin EXIF allowlist

and by grepping `latitude`/`longitude` across all PHP (no hits outside the importer and installer).

- The absence of Flickr credentials was verified by reading the parser end to end
- it consumes `albums.json`, `photo_*.json`, `contacts_part*.json` and image

files from a local folder, and makes no network call to Flickr.

- No remediation has been applied. Nothing in this report has been fixed yet.