SECAUDIT 047 — Full Multi-Dimension Security Audit of 0.7.525

Audit date: 2026-08-14

Target release: SnapSmack 0.7.525D (live dev track)

Method: Source-level review across nine security dimensions, followed by an adversarial re-audit of the fixes

Remediation release: 0.7.526D

Publication status: Closed. Superseded in part by SECAUDIT 048, which live-tested these fixes and found two items this review missed.

Executive summary

SECAUDIT 047 was a full, source-level security review of the entire SnapSmack core, read against the actual shipping code (0.7.525D) rather than a stale branch. Nine reviewers each took one dimension — authentication, authorization, SQL injection, path/local-file access, file upload and code execution, federation and server-side request forgery, cross-site request forgery, secrets and the updater, and cross-site scripting — and each was required to find the real code, prove a real caller could reach it, and propose a minimal fix. Two further reviewers then re-attacked the patched code.

The review found four critical issues, seven high, three medium, and nine low. The single most important theme: the authorization layer was effectively missing. Admin pages decided access one page at a time, and almost none actually checked the caller's role, so a content-only editor was a latent full administrator. Alongside that were a recovery path that could write attacker files into the site (code execution), a federation check that never tied the signing key to the claimed author (account takeover), a download-link secret that was a shared default, and a media uploader that accepted any file type.

All four critical, all seven high, and all three medium findings were fixed in 0.7.526D, along with eight of the nine low findings. The ninth low (an updater fail-open in a rare state) was deliberately deferred because the updater is the only channel by which fixes reach installed sites. An adversarial re-audit of the patched code found four residual gaps, all of which were also fixed before release.

This was a code review, not a live attack. The follow-up live penetration test (SECAUDIT 048) confirmed the upload, CSRF, download-link, and role-gate fixes held under real attack, and caught two things this review did not: a leftover

installer left an unauthenticated recovery console, and one of the cross-site scripting fixes below was incomplete.

Scope and method

The review covered the whole SnapSmack core plus the Smack Central hub login. It was reset onto tag `v0.7.525D` first, because a previous pass had mistakenly run against a stable-track checkout roughly four hundred versions behind the live code; every finding here is confirmed present in the shipping release.

Nine parallel reviewers each owned one dimension and had to demonstrate reachability, not merely pattern-match. Two adversarial verifiers then re-checked every fix against the patched tree and were specifically tasked with finding what the first pass missed.

Findings at a glance

#	Severity	Finding	Resolution
1	CRITICAL	A content editor could promote itself to administrator	Fixed in 0.7.526D
2	CRITICAL	A content editor could create or delete administrators	Fixed in 0.7.526D
3	CRITICAL	Recovery-kit import could write files into the site (code execution)	Fixed in 0.7.526D
4	CRITICAL	Federation signature check did not bind the key to the claimed account (takeover)	Fixed in 0.7.526D
5	HIGH	Admin pages did not check the caller's role (root cause of 1 and 2)	Fixed in 0.7.526D
6	HIGH	The hub login had no brute-force lockout	Fixed in 0.7.526D
7	HIGH	The download-link secret was a shared hardcoded default	Fixed in 0.7.526D
8	HIGH	Delete/approve/ban actions were plain links (CSRF via GET)	Fixed in 0.7.526D
9	HIGH	The media library accepted any file type (webshell upload)	Fixed in 0.7.526D
10	HIGH	A remote user could edit or delete another user's federated comments	Fixed in 0.7.526D
11	HIGH	Script injection via `javascript:` links on the public Photo Challenge board	Fixed in 0.7.526D
12	MEDIUM	Backup-restore and FTP-push were plain links (CSRF via GET)	Fixed in 0.7.526D

#	Severity	Finding	Resolution
13	MEDIUM	The active-skin setting could point outside the skins folder	Fixed in 0.7.526D
14	MEDIUM	Script injection via `javascript:` links in the admin fediverse panel	Fixed in 0.7.526D
15	LOW	Password-reset tokens were stored in plain text	Fixed in 0.7.526D
16	LOW	The single-sign-on cookie could travel without the Secure flag	Fixed in 0.7.526D
17	LOW	Login timing revealed whether a username existed	Fixed in 0.7.526D
18	LOW	The updater could skip its signature check in a rare state	Deferred (see below)
19	LOW	The remote-follow link could redirect anywhere	Fixed in 0.7.526D
20	LOW	The RSS fetcher could be aimed at internal addresses (SSRF)	Fixed in 0.7.526D
21	LOW	Logout could be triggered by a cross-site request	Fixed in 0.7.526D
22	LOW	The page's canonical/OpenGraph URL was echoed unescaped	Fixed in 0.7.526D (incomplete — see 048)

Critical finding 1 — editor self-promotion to administrator

What happened

The self-service account editor exposed the role field and saved whatever role was submitted, without checking that the person making the change was an administrator. A logged-in editor could open their own account and set their role to administrator.

Why it mattered

This is a direct jump from the lowest content role to full site owner, opening every administrative capability: settings, accounts, secrets, the updater, and federation control. In practice it is site takeover from a content account.

Correction

The account page is now behind the central administrator gate, and the role field is administrator-only. An editor can no longer view or change roles.

Critical finding 2 — editor creating or deleting administrators

What happened

The user-management page had no role check, so an editor could open it and reach the actions that create, delete, or re-role any account.

Why it mattered

An editor could quietly create a new administrator for persistence, or delete the real administrator to lock the owner out. It reaches the same outcome as finding 1 by a second route.

Correction

The page is now administrator-only, and account changes verify administrator status before running.

Critical finding 3 — recovery-kit import could execute code

What happened

The recovery/restore engine wrote each file from a recovery archive to a location taken from the archive itself, without constraining that location. A crafted archive could climb out of the intended folder, or use a disguised double extension, to drop an executable file into the web root.

Why it mattered

Recovery code is powerful by nature — it can replace files and change the database. Leaving the destination attacker-controlled turned a restore into remote code execution on the server. The browser-side import also lacked a cross-site request token.

Correction

Restore targets are now confined to the site folder, reject climbing paths and absolute paths, and reject any file whose name contains an executable extension in any position (so a disguised name cannot slip through). The browser import now requires a valid request token.

Critical finding 4 — federation account takeover

What happened

When another server sent a signed message, the signature check fetched the signing key's owner but never required that key to belong to the same server as the account the message claimed to come from. A remote server could present a message validly signed with its own key while claiming to be any other account.

Why it mattered

It allowed impersonation of any federated account and overwriting of its posts and comments — a federation-level takeover of identity.

Correction

The signing key is now bound to the claimed author: the key's origin must match the account's origin, and where the account declares its key, that key must match the one that signed. This is enforced consistently for create, update, and delete messages.

High findings

High finding 5 — admin pages did not check the caller's role

This was the root cause of findings 1 and 2. There was no central place that asked "is this person an administrator?"; each page was on its own, and almost none checked. The fix adds one central gate and a list of administrator-only pages, evaluated on every admin request. The list is deny-by-default for configuration pages: a new content page is editor-reachable, but a new configuration or system page must be added to the list. (SECAUDIT 048 later found three system pages that had been left off that list.)

High finding 6 — the hub login had no lockout

The Smack Central hub login accepted unlimited password attempts, and because the hub uses its own database it was not covered by the spoke's rate limiting. The fix adds a self-contained lockout after repeated failures, plus an even-timing dummy check so a locked or unknown account cannot be told apart by response time.

High finding 7 — the download-link secret was a shared default

Download links are signed with a per-site secret, but that secret was never generated, so every install fell back to the same built-in default. Anyone who knew the default could forge valid download links for any site. The fix has each install generate and store its own random secret, self-healing if it is missing, while the link check remains constant-time. (Impact was limited to already-public images, but the signature was effectively meaningless before the fix.)

High finding 8 — destructive actions were plain links (CSRF)

Delete, approve, ban, restore, and moderation actions across thirteen pages were ordinary links with no anti-forgery token, so a booby-trapped link — or an image tag in an email — could perform them silently in a logged-in administrator's browser. The fix introduces a shared request-token helper and threads a

one-time token through every such link, verified before the action runs.

High finding 9 — the media library accepted any file type

The media uploader trusted the filename extension, so a script file could be uploaded into a web-served folder and executed. The fix derives the file type from the actual file contents, accepts only real image types, and configures the upload folder to refuse to execute code. (The live test in 048 confirmed a script upload is rejected and a disguised image/script polyglot is stored inert.)

High finding 10 — remote editing of other users' comments

Incoming federated update and delete messages for a comment matched on the comment's identifier alone, not on the author, so a remote user could edit or delete a comment they did not write. The fix requires the acting account to match the stored author before an update or delete is applied.

High finding 11 — script injection on the public challenge board

Submitted links, including federated content, were shown on the public Photo Challenge board and hall of fame without checking the link type, so a `javascript:` link could run code in every visitor's browser. The fix accepts only normal web links, both when content is received and when it is displayed.

Medium findings

Medium finding 12 — backup-restore and FTP-push were plain links

The same cross-site-request weakness as finding 8, on the highest-impact operations. Backup restore and FTP actions now require a valid request token.

Medium finding 13 — the active-skin setting could escape the skins folder

The active-skin value could be pointed outside the skins directory (stored, and reachable through a forged request). Only installed skins with a cleaned name are now accepted, and the cleaned name is used for both the path and the stored setting.

Medium finding 14 — script injection in the admin fediverse panel

A `javascript:` link could run inside the administrator's fediverse view. This is closed both by finding 4 (such an account cannot pass signature verification) and by the display-time link-type check.

Low-severity findings

- **15 — reset tokens in plain text (fixed).** Password-reset tokens are now

stored hashed, so a database read cannot reuse a live token.

- **16 — sign-on cookie missing Secure (fixed).** The single-sign-on cookie now sets the Secure flag when the site is served over HTTPS through a proxy.
- **17 — login timing enumeration (fixed).** The login now performs an even-timing dummy check for unknown usernames so response time no longer reveals whether an account exists.
- **18 — updater fail-open (deferred, accepted).** In a narrow state the updater could proceed without completing its signature check. This was left untouched on purpose: the updater is the only way fixes reach installed sites, and a mistake here could lock every site out of all future updates. It is documented for a deliberate, separately-tested change, not represented as fixed.
- **19 — open redirect on remote-follow (fixed).** The remote-follow link is now restricted to the same site over HTTPS.
- **20 — RSS fetcher SSRF (fixed).** The feed fetcher now refuses addresses that resolve to private or reserved ranges, limits redirects, and verifies TLS.
- **21 — logout via cross-site request (fixed).** Forged background logout requests from other sites are now dropped, without changing any real logout link.
- **22 — canonical/OpenGraph URL echoed unescaped (fixed, but incomplete).** The canonical and OpenGraph URLs are now escaped. This fix missed the sibling JSON-LD data block, which still echoed content into an inline script unescaped; SECAUDIT 048 proved that exploitable and it was fixed in 0.7.528D.

Adversarial re-audit

Two verifiers re-attacked the patched code and found four residual gaps, all fixed before 0.7.526D shipped:

1 The federated-comment binding from finding 10 covered update and delete but not the create path, so a create message could still overwrite another author's comment. The create path now applies the same author binding.

1 The recovery guard from finding 3 checked only the final extension, so a disguised double extension slipped through. It now inspects every part of the filename.

1 The solo-post uploader still trusted the client filename extension. It now derives the type from file contents, images only.

1 The first-run secret heal from finding 7 could create two secrets under

concurrent requests and orphan the stored FTP password. It now takes a lock and re-reads under it.

Defenses that held up

The review confirmed several controls were already sound: SQL access is consistently parameterized; passwords are properly hashed; secrets use secure randomness; there are no hardcoded credentials in the source; and the normal image-upload and skin-install paths were well defended, with the package updater using signed, verify-before-extract releases.

Limits of this audit

This was a source-level review. It reasons about reachability from reading the code; it is not a live attack test. The live penetration test in SECAUDIT 048 was commissioned specifically to close that gap and did find two issues this review missed (the leftover installer console, and the incomplete JSON-LD escape noted in finding 22).

Remediation verification

Every fix is marked in the source, all changed files passed syntax validation and retained their required end-of-file markers, and the four re-audit gaps above were re-verified. Shipped in 0.7.526D. See `2026-08-15-048-live-penetration-test-0.7.527.md` for the live follow-up.