SECAUDIT 048 — Live Penetration Test of 0.7.527

Test date: 2026-08-15

Target release: SnapSmack 0.7.527D

Test environment: Disposable GRAMOFSMACK installation, isolated from the production network hub

Remediation releases: 0.7.528D (serious findings) and 0.7.530D (header-delivery correction)

Publication status: The serious 0.7.528D fixes are verified live. The HSTS/CSP delivery correction (0.7.530D) is pending a live header check; publish only after that check passes.

Executive summary

SECAUDIT 048 was SnapSmack's first live penetration test. Earlier reviews read the code and reasoned about what an attacker might reach. This test attacked a running installation as both an unauthenticated visitor and an authenticated content-only editor.

The live test found two critical installation-state failures, one high-severity script-injection flaw, and one medium-severity authorization gap. The critical failures could expose recovery and deployment machinery after installation. The script-injection flaw allowed stored content to escape a JSON-LD data block and execute in a visitor's browser. The authorization gap let an editor reach three system pages that should have been restricted to administrators.

All critical, high, and medium findings were corrected in 0.7.528D and the fixes were confirmed on the live upgraded site. The test also identified four low-severity hardening items. Two (an editor diagnostics leak and a password-reset CSRF token) were completed in 0.7.528D. The other two — adding HSTS and a Content-Security-Policy — were where this report was initially wrong: the 0.7.528D code placed those headers in a protected file the updater never delivers, so they did not reach the live site. That delivery defect is corrected in 0.7.530D and remains open until the live response headers are confirmed.

The exercise also verified that many defenses added during SECAUDIT 047 held up under real attack: administrative authentication, sensitive-file blocking, path-traversal protection, upload restrictions, signed download links, destructive-action CSRF tokens, session-cookie protection, login response handling, password-reset rate limiting, and public-registration controls.

Scope and method

The test used a disposable site with no production content and no connection to

the SnapSmack network hub. Testing covered two realistic threat positions:

1. an unauthenticated internet visitor; and
2. an attacker who had obtained the password of a content-only editor.

The second position matters because editor accounts are intentionally allowed to create and manage content but must never gain control of users, site-wide configuration, recovery, updates, secrets, or system maintenance.

Tests included access-control checks, installation-state checks, controlled upload attempts, stored and reflected script-injection probes, path traversal, sensitive-file requests, destructive-action CSRF review, download-token tampering, session behavior, login behavior, password-reset behavior, and editor-role boundary checks.

No production site, production account, private archive, or third-party system was used as a target.

Findings at a glance

#	Severity	Finding	Live proof	Resolution
1	CRITICAL	Installed site could still expose installer recovery/patch functions	Confirmed	Fixed in 0.7.528D; verified live
2	CRITICAL	Bootstrap deployer could become available again after installer removal	Confirmed	Fixed in 0.7.528D; verified live
3	HIGH	Stored/reflected script injection through JSON-LD output	Confirmed, script executed	Fixed in 0.7.528D; verified live
4	MEDIUM	Three system pages were missing from the editor restriction boundary	Confirmed	Fixed in 0.7.528D
5	LOW	HSTS response header absent	Confirmed by headers	**Open** — 528D code did not ship; delivery fixed in 0.7.530D, awaiting live check
6	LOW	Content-Security-Policy header absent	Confirmed by headers	**Open** — 528D code did not ship; delivery fixed in 0.7.530D, awaiting live check
7	LOW	Editor dashboard exposes unnecessary server diagnostics	Confirmed	Fixed in 0.7.528D
8	LOW	Password-reset request form lacks a CSRF token	Confirmed	Fixed in 0.7.528D

Critical finding 1 — installed-site recovery surface

What happened

SnapSmack correctly recognized that the site was already installed during the normal installation path, but two exceptional installer modes were evaluated before the installed-state lock fully took effect. That left recovery and schema-patching capabilities reachable without first entering the authenticated administration area.

Why it mattered

Installation and recovery code has unusual power: it can inspect or replace files and alter database structure. Leaving any of that machinery reachable after installation creates an unauthenticated route around the normal admin, two-factor, and recovery controls. The practical consequence was potential server takeover or destructive database changes.

Correction

Once an installed SnapSmack database is detected, `install.php` now refuses all installer actions. Recovery for an existing site is available only through the authenticated Disaster interface. The blank-server recovery path remains available only when no installed site exists. Verified live: the installed site's `install.php` no longer exposes recovery or schema-patch actions.

Critical finding 2 — bootstrap deployer reactivation

What happened

The small bootstrap deployer used an overly narrow test to decide whether a site already existed. Removing one installation file — an otherwise sensible cleanup step — could make the bootstrapper believe deployment was available again even though the application and database configuration were present.

Why it mattered

A deployment bootstrapper can write the application onto the server. If it reactivates on a live installation, an unauthenticated visitor may be able to replace or redeploy files over the existing site.

Correction

`setup.php` now locks immediately when either of the durable installed-site markers exists. It returns a refusal page instead of entering any deployment

path. Removing the main installer no longer re-arms the bootstrapper. Verified live: `setup.php` returns HTTP 403 on the installed site.

High finding — JSON-LD script injection

What happened

Photo titles, captions, ALT text, tags, and the current page URL were serialized into a JSON-LD `<script>` element. The JSON itself was valid, but the output did not hex-escape HTML-significant characters. A stored closing-script sequence could therefore terminate the data block and introduce executable markup. The test demonstrated both stored and reflected execution on the disposable site.

Why it mattered

An attacker able to submit or edit content could run script in another visitor's browser under the blog's origin. Depending on the victim and page, that could alter content, perform authenticated actions, or expose information available to the browser. The HttpOnly session cookie correctly prevented direct script access to the session identifier, limiting — but not eliminating — the impact.

Correction

JSON embedded in script elements is now emitted with the full set of hex-escaping flags for tags, ampersands, apostrophes, and quotation marks. The same protection was applied to the inline configuration JSON. The resulting JSON-LD remains valid for search engines while closing the HTML parser boundary. Verified live: the tested closing-script payload no longer breaks out of the JSON-LD block.

Medium finding — incomplete editor boundary

What happened

SECAUDIT 047 introduced a central admin-only page boundary, but three system pages were omitted from its registry. A content-only editor could open traffic statistics and reach audit/backfill functions that write settings or image records.

Why it mattered

The role model promises that editors can manage content without controlling the site. Any omitted system page weakens that promise and may provide a path to configuration changes, bulk mutation, or future privilege escalation.

Correction

The audit, backfill, and statistics pages are now part of the central admin-only registry. The existing gate returns a refusal for editor sessions before page logic runs. Administrator-authenticated backup tooling is unaffected.

Low-severity hardening

Completed in 0.7.528D

- **Editor diagnostics:** the content-only editor dashboard no longer shows server software, language version, filesystem/load, or scheduler-path details.
- **Reset-request CSRF:** the password-reset request form now carries a token. The endpoint was already rate-limited and already returned the same response for known and unknown addresses.

Header-delivery correction (0.7.530D) — open until verified

- **HSTS and CSP:** these were written in 0.7.528D but placed in a protected configuration file the updater does not deliver to existing installs, so the live upgraded site sent neither header. This was a delivery defect, not a code gap. 0.7.530D moves the header emission into a shipping file loaded on every response and unprotects the configuration file (which holds no per-install data). These findings remain **open** until the live response headers are confirmed to include HSTS and the initial CSP after 0.7.530D is deployed.

Defenses that held up

The following controls were exercised against the running site and resisted the test:

- unauthenticated access to normal admin pages was redirected to sign-in;
- sensitive configuration, repository, environment, migration, and database files were not publicly readable;
- path-traversal and local-file probes did not escape their intended routes;
- executable uploads were rejected;
- a deliberately malformed image/program polyglot was stored as inert image data and did not execute;
- signed download links rejected forged and modified tokens;
- destructive media and post actions carried CSRF tokens;
- the session identifier remained HttpOnly;
- login failures did not expose sensitive detail;
- password-reset requests were rate-limited and did not reveal whether an address

belonged to an account;

- public self-registration was disabled.

This list is important: a penetration report should record not only what failed, but which security claims were actually tested and survived.

Limits of this test

Federation was disabled on the disposable site, so live ActivityPub signature forgery and remote-actor behavior were outside this pass. The actor/key binding introduced after SECAUDIT 047 was reviewed in code but must receive a separate live federation test when an isolated federated fixture is available. Live SQL injection, the OAuth/API surface, and two-factor bypass were also out of scope for this pass.

This was a targeted test, not proof that no other vulnerability exists.

Remediation verification

- 0.7.528D is deployed and its serious fixes are confirmed on the live upgraded site: the installer and bootstrapper both refuse access, and the tested JSON-LD closing-script payload no longer breaks out of its script element.
- 0.7.530D corrects the HSTS/CSP delivery defect. Closure of findings 5 and 6 requires one live header check, with a cache-busted request, confirming HSTS and the initial CSP are present after 0.7.530D is deployed.
- All changed source files passed syntax validation and retained their required end-of-file markers.

Closure gate for publication

Publish this report only after all of the following are true:

1. the disposable test site is confirmed on 0.7.530D or later;
2. the installed-site installer and bootstrapper both refuse access (met on 528D);
3. the JSON-LD injection fixture no longer escapes its script element (met on 528D);
4. the editor account is refused by all three corrected system pages;
5. HSTS and the initial CSP appear in the live response headers;
6. this section is replaced with the deployment date and verification result;
7. the final PDF is rendered and visually inspected.