SnapSmack Security Audit

Second Review — Alpha 0.7.22 · Fixes in 0.7.23 · April 25, 2026

Executive Summary

This second audit was conducted immediately following the 0.7.22 security hardening release, which resolved all eight findings from Audit 1. The purpose of this document is to confirm those fixes are in place and to report any new findings identified in the updated codebase.

This audit identified **2 new critical findings**, **1 medium finding**, **1 low finding**, and **1 high finding carried forward** (CSRF — previously noted, intentionally deferred). The 2 critical findings and the low finding were fixed immediately in 0.7.23. No authentication bypass or privilege escalation vulnerabilities were found.

#	File	Severity	Finding	Status
1	core/contact-form.php	CRITICAL	Email header injection	FIXED 0.7.23
2	smack-central/sc-enemy-api.php	CRITICAL	Race condition in rate limiter	FIXED 0.7.23
3	privacy-policy.php	MEDIUM	Raw admin HTML on public page	ACCEPTED RISK
4	smack-central/sc-release.php	LOW	Weak temp file randomness	FIXED 0.7.23
5	Multiple AJAX endpoints	HIGH	CSRF tokens missing	DEFERRED

Findings

1. Email Header Injection — core/contact-form.php

CRITICAL Lines 111–113

User-supplied **$name** is interpolated directly into the mail subject, and **$email** is placed into the From: and Reply-To: headers. While FILTER_VALIDATE_EMAIL rejects most addresses containing newlines, it does not sanitise **$name** at all. A name containing \r\n sequences can inject arbitrary mail headers, enabling spam relay and BCC injection.

Vulnerable code:

```
$subject = "[$site_name] Contact form message from $name"; // $name unsanitised $headers = "From: $email\r\nReply-To: $email\r\nX-Mailer: SnapSmack";
```

Recommended fix — strip CRLF from all user inputs before use in headers:

```
$name = preg_replace('/[\r\n]+/', ' ', $name); $subject = "[$site_name] Contact form message from $name"; // $email is already FILTER_VALIDATE_EMAIL validated — CRLF in addresses is rejected // but strip for defence-in-depth: $email = preg_replace('/[\r\n]+/', '', $email); $headers = "From: $email\r\nReply-To: $email\r\nX-Mailer: SnapSmack";
```

---

2. Race Condition in Rate Limiter — smack-central/sc-enemy-api.php

CRITICAL Lines 70–85 · ste_rate_limit()

The STE API rate limiter uses a per-IP JSON file in /tmp with no file locking. Under concurrent load, multiple requests read the same stale count before any write completes — all see count=N, all increment to N+1, and all pass. An attacker sending 100+ concurrent requests can effectively bypass the 100 req/min limit entirely.

Vulnerable code:

```
$file = sys_get_temp_dir() . '/ste_' . md5($ip) . '.json'; $data = file_exists($file) ? json_decode(file_get_contents($file), true) : [...]; // RACE: all concurrent threads read old count here $data['count']++; file_put_contents($file, json_encode($data)); // last write wins
```

Recommended fix — use exclusive file locking:

```
$fp = fopen($file, 'c+'); flock($fp, LOCK_EX); $data = json_decode(stream_get_contents($fp), true) ?? ['count' => 0, 'window' => time()]; if (time() - $data['window'] > 60) $data = ['count' => 0, 'window' => time()]; $data['count']++; ftruncate($fp, 0); rewind($fp); fwrite($fp, json_encode($data)); flock($fp, LOCK_UN); fclose($fp); if ($data['count'] > $limit) { ste_json(['ok' => false, 'error' => 'Rate limit exceeded.'], 429); }
```

---

3. Raw Admin HTML on Public Page — privacy-policy.php

MEDIUM Line 78

The privacy policy content from snap_settings is echoed unescaped to public visitors. This is intentional by design — the admin is expected to write HTML — and carries the same risk profile as the custom head scripts feature (also admin-entered HTML). However, unlike head scripts (which now live in data/custom-head.html and are watched by SMACKBACK), privacy policy content lives in the database with no file-integrity monitoring. A compromised admin account or SQL injection would result in stored XSS on a public-facing page.

Note: This is not a bug in the traditional sense — raw HTML output is the intended behaviour. The risk is the attack surface it creates if the DB is compromised.
Recommended mitigations:
1. Add a SMACKBACK-style hash of privacy_policy_content to detect unexpected changes. 2. Document in the admin UI that this field accepts HTML and that JS will execute. 3. Long term: consider running content through a whitelist sanitiser (HTML Purifier) while preserving valid structural HTML.

---

4. Weak Temp File Randomness — smack-central/sc-release.php

LOW Line 172

Temporary zip filenames are constructed with time() and rand(1000, 9999). This gives only ~9,000 possible filenames within any given second, making the path predictable. Exploitability is low — requires access to the server's temp directory — but is easily hardened.

Current code:

```
$tmp_src = sys_get_temp_dir() . '/sc_gh_' . time() . '_' . rand(1000, 9999) . '.zip';
```

Recommended fix:

```
$tmp_src = sys_get_temp_dir() . '/sc_gh_' . bin2hex(random_bytes(16)) . '.zip';
```

---

5. CSRF Tokens Missing on AJAX Endpoints (Deferred)

HIGH process-like.php · process-reaction.php · process-community-comment.php

State-changing POST endpoints (liking posts, setting reactions, posting comments) accept requests without validating a CSRF token. A malicious third-party page could trigger these actions on behalf of a logged-in visitor. This was flagged as HIGH in Audit 1 and remains open by design — implementation requires coordinated changes across multiple JS engines and PHP endpoints.

Status: deferred. Partial mitigation exists via SameSite=Lax session cookies, which blocks cross-site form submissions in modern browsers but not all AJAX attack vectors.

Audit 1 Findings — Confirmed Fixed in 0.7.22

Finding	File	Original Severity	Status

Open redirect	community-auth.php	CRITICAL	**FIXED**
Logo/favicon upload — no MIME check	smack-settings.php	CRITICAL	**FIXED**
Path traversal via skin manifest	smack-edit.php	HIGH	**FIXED**
Session fixation in 2FA flow	login.php	MEDIUM	**FIXED**
No rate limit on admin password reset	password-reset.php	HIGH	**FIXED**
Weak slug validation	smack-post-solo.php, smack-post-longform.php	MEDIUM	**FIXED**
Raw DB error in installer	install.php	LOW	**FIXED**
No security response headers	core/constants.php	LOW	**FIXED**

Overall Assessment

The codebase continues to demonstrate strong foundational security practices: parameterised queries are used consistently throughout, community session management is well-implemented, the release installer now performs Ed25519 signature verification, and all eight Audit 1 findings have been corrected. No authentication bypass, privilege escalation, or SQL injection vulnerabilities were identified in this review.

The two new critical findings (email header injection and the rate-limiter race condition) are straightforward to fix and should be addressed before the next release. The privacy policy XSS risk is a design trade-off worth documenting clearly. CSRF remains the largest deferred item.

This document is part of SnapSmack's internal security audit series. Previous: 2026-04-25-A-snapsmack-security-audit.pdf