SECAUDIT 038

Gallery skin JavaScript package boundary

AUDIT ID	2026-08-05-038
DATE	2026-08-05
SEVERITY	MEDIUM (defense-in-depth)
COMPONENT	Gallery skins, content scanner, Smack Central Skin Packager, shared engine inventory
STATUS	REMEDIATED - closed in 0.7.500D
REPORTER	Sean + Claude + Codex independent closure review
RELATED	SECAUDIT 019, 025, 026, 030, 034
DISCLOSURE	No exploitation is known.

1. Summary

SnapSmack's skin architecture already had a reviewed JavaScript engine inventory, but the Gallery packaging boundary did not fully enforce that design. Several official skin directories still contained their own `.js` files or direct script references. Earlier checks looked for dangerous JavaScript patterns; they did not categorically reject every JavaScript carrier inside a skin package.

A signature proves which release process produced a package and that its bytes were not changed afterward. It does not prove that browser-executable code inside the signed package is safe. Skin-local copies also survive fixes made to the shared engine, creating stale and inconsistent code paths across installed sites.

The policy is now explicit and enforced: a Gallery skin ships **no JavaScript**. Reusable browser behaviour lives in core-owned `assets/js/`, is registered in `core/manifest-inventory.php`, and is requested through an inert JSON manifest. The repository scanner detects executable JavaScript carriers, and the Smack Central packager refuses to sign or publish a skin with any blocking finding.

All skin-shipped JavaScript found during remediation was removed or moved to the shared engine library. The completed scan reports zero blocking findings across the official skin set.

2. Finding - the package boundary was descriptive, not mandatory

2.1 What existed

SnapSmack already encouraged skins to declare shared engines with `require_scripts`. The footer loader resolved those identifiers through the core manifest inventory and loaded core-owned files. However, a skin could still contain executable browser content and reach the packager:

- a bundled `.js` file;
- an inline `<script>` block or HTML event handler;
- a `javascript:` URI or external/skin-relative script source;
- an active `<iframe>`, `<object>`, or `<embed>` element.

The official tree contained multiple examples of skin-local JavaScript, including navigation and presentation engines. These files were legitimate features, not evidence of malicious code. They demonstrated that the intended boundary was not yet an enforced publishing rule.

2.2 Security impact

JavaScript in a skin executes in the site's browser origin. A malicious or compromised package could read browser-accessible content and tokens, alter links or administrative interfaces, exfiltrate data under the visitor's credentials, or persist as an overlooked per-skin copy after the shared engine was repaired.

This was not an unauthenticated remote-code-execution path. Installing and publishing Gallery skins remained a privileged, signed workflow. The issue was a missing defense at that workflow's trust boundary, so the finding is rated MEDIUM defense-in-depth. No exploitation is known.

3. Remediation

3.1 Repository clean scanner

`tools/skin-scan.php` now scans each skin and emits blocking findings for browser-executable carriers. It rejects bundled JavaScript, inline handlers and script blocks, JavaScript URIs, external or skin-local script sources, and active embedded-content elements.

The permitted loading mechanism is the core-owned footer loader, which resolves script paths from the manifest inventory instead of a skin-controlled URL. A direct tag pointing at a known core engine is reported as a warning so it can be migrated without treating reviewed core bytes as a skin-local payload.

The command-line scanner reports every official skin, returns a non-zero exit code when any blocker exists, and can emit structured JSON for automation.

3.2 Fail-closed packaging gate

Smack Central invokes the same scanner immediately before packaging. Any blocker rejects the build before ZIP creation, signature generation, or registry publication. The error identifies the finding type and source location.

A clean repository is useful evidence; the packaging gate is the control that prevents a later regression from becoming a signed Gallery release.

3.3 Shared engine consolidation

Reusable behaviour was moved into core-owned engines, including Alfred navigation, community, and Glide. The shared files are registered in the core inventory and affected skins request them through `require_scripts`. More than 500 lines of duplicate or misplaced skin JavaScript were removed, restoring one patch point per engine.

4. Verification

The closure review verified:

- scanner coverage for bundled files and known inline, URI, external-script, and embedded-content carriers;
- non-zero failure for a blocking finding and success for the clean official tree;
- Smack Central executes the scanner before packaging, signing, and registry publication;
- official skin directories contain no `.js` files after the purge;
- moved engines are under `assets/js/` and registered in the core inventory;
- affected manifests request shared engines instead of loading local copies;
- the complete official skin scan reports zero blocking findings.

The remediation is present in commits `df694083` and `8e649b4c`, both included in the development release tag `v0.7.500D`.

5. Residual trust and limitations

Skins still contain PHP presentation templates. The JavaScript clean gate does not make an arbitrary third-party skin harmless and does not replace signature, manifest, ZIP-path, or server-side code review controls.

The scanner is a publishing-policy gate, not a general-purpose HTML or PHP parser. Its line-based rules cover supported skin authoring patterns and provide precise source locations. Future template syntax capable of producing executable browser content must be added to the scanner before it is accepted in Gallery skins.

Owner-authored custom code outside the Gallery workflow remains a separate, explicitly privileged capability. It must not become an exception that allows JavaScript back into signed Gallery packages.

6. Closure

The architecture now says one thing in both design and enforcement: skin packages choose shared behaviour; they do not carry browser code. Signatures establish provenance, the clean gate establishes content policy, and the shared engine inventory provides one reviewed and repairable implementation.

The official skin set is clean, the packager fails closed, and no exploitation is known.

Disposition: CLOSED in 0.7.500D.