SECAUDIT 052 - `llms.txt` Agent-Execution Surface

SECURITY AUDIT / PUBLIC REMEDIATION RECORD

Audit date: 2026-08-27 **Remediation release:** SnapSmack 0.7.575 **Scope:** `core/site-files.php`, generated `/llms.txt`, output trust boundaries, release integrity, and regression coverage **Review:** Sean McCormick and OpenAI Codex; independent live-fleet verification remains a deployment acceptance step **Public status:** Finding remediated in source; publish and deploy through the signed BITCHIN' release channel

1. Executive summary

SnapSmack automatically publishes `/llms.txt` beside `robots.txt`. The file describes the photoblog, identifies SnapSmack, exposes the site's home and archive locations, and expresses the photographer's AI-training preference. It is written for machine readers, including search systems and AI agents.

`llms.txt` is not executable code. Merely requesting it cannot run a command on a visitor's computer or on the SnapSmack server. The risk arises in a different system: an external AI agent may read machine-oriented documentation, treat prose inside it as trusted instructions, and possess tools capable of running shell commands, downloading software, or modifying files. If attacker-controlled text can enter that documentation, it may become an indirect prompt-injection or execution path in the agent that consumes it.

The pre-audit SnapSmack template contained no package-install commands, fetch-and-run examples, code fences, registry references, or `llms-full.txt`. There was therefore no direct command-execution payload in the shipped template. One preventive high-severity boundary gap was nevertheless confirmed: publisher-controlled `site_name`, `site_description`, and `site_url` values were inserted directly into agent-readable Markdown. A malicious or compromised setting could manufacture headings, links, role-labelled instructions, or executable-looking command text.

Release 0.7.575 closes that boundary. Publisher fields are now reduced to inert, single-line descriptive text; suspicious descriptions are omitted; unsafe URLs are rejected; generated links are restricted to the validated current site and hard-coded SnapSmack attribution; and regression tests pin the behavior against representative prompt-injection and command strings.

2. What was reviewed

The audit examined the complete local generation path rather than treating the static file as an isolated artifact:

- `snapsmack_generate_llms()` in `core/site-files.php`;
- the site settings copied into the generated document;
- the fixed strings and URLs added by SnapSmack;
- the write path used by Global Configuration, Maintenance rebuilds, and pushed

multisite AI-training policy changes;

- the relationship between generated output and SMACKBACK's signed release

manifest;

- whether any path generates `llms-full.txt`;
- whether updater protection would preserve stale or modified output;
- regression coverage for agent-oriented input rather than browser-only XSS.

The review did not treat the third-party agent as part of SnapSmack's trusted computing base. SnapSmack cannot force an unrelated AI tool to distinguish data from instructions. It can prevent its own generated document from becoming a convenient carrier for attacker-selected instructions.

3. Threat model

3.1 Required conditions for exploitation

A practical exploit requires all of the following:

1. An attacker gains influence over text or a link written into `llms.txt`.
2. An external AI agent reads that file or content transitively referenced by it.
3. The agent treats the attacker-controlled content as an instruction instead of

untrusted data.

4. The agent has permission to execute commands, install packages, fetch files,

or perform another consequential action.

5. The agent executes the instruction without an adequate approval or trust

boundary.

Removing any one condition breaks the chain. SnapSmack directly controls the first condition and can materially reduce the second by limiting generated links. The consuming agent remains responsible for conditions three through five.

3.2 Time-delayed dependency takeover

Machine-readable documentation becomes particularly dangerous when it contains installation examples. A package name or domain may be safe when written but later expire, be abandoned, or be transferred. An instruction such as an install command can then begin retrieving attacker-controlled material without the SnapSmack site changing at all. This is why the policy prohibits actionable package, image, registry, and fetch-and-run syntax rather than merely checking that a named dependency is safe today.

3.3 Fleet multiplication

SnapSmack generates the file from one code path across a fleet. A defect in the generator can be reproduced on every updated installation. Conversely, fixing the generator once and delivering it through the signed update channel provides a consistent fleet-wide boundary. Live verification must still be per-site, because old files, failed writes, host drift, or caching can leave one installation different from the source template.

3.4 Trust boundaries

Material	Trust decision
SnapSmack generator code	Trusted only when delivered through the signed release and verified by SMACKBACK
Fixed SnapSmack prose	Reviewed first-party output
Site name and description	Publisher-controlled data; never instructions
Site URL	Publisher-controlled configuration; validate before creating links
`https://snapsmack.ca`	Explicit first-party attribution target
Discovered or publisher-supplied external URL	Not permitted in generated `llms.txt`

4. Finding 052-A - publisher fields crossed into an agent instruction surface

Severity: HIGH, preventive **Exploitability qualifier:** exploitation additionally requires a consuming agent that follows untrusted instructions and has consequential tools **Affected code:** pre-0.7.575 `snapsmack_generate_llms()`

4.1 Previous behavior

The generator interpolated three settings directly:

- `site_name` became the top-level Markdown heading;
- `site_description` became a Markdown blockquote;
- `site_url` became the Home and Archive link prefix.

Those settings are normally edited by the site administrator, which lowers the probability of an unauthenticated exploit. It does not make the boundary safe. Settings can be imported, restored, synchronized, copied from another system, or modified after an administrative compromise. More importantly, the generator's contract claimed to produce descriptive machine-readable data but did not enforce that contract.

4.2 Why ordinary HTML escaping is insufficient

The output is plain Markdown-oriented text, not HTML rendered into a browser DOM. Escaping `<script>` would not stop any of these shapes:

- a newline followed by a new `## SYSTEM` heading;
- `Ignore previous instructions` prose;
- a Markdown link to an attacker-controlled host;
- a fenced or inline command example;
- `pip install`, `npx`, `docker run`, or `curl ... | sh`;
- a `javascript:`, `data:`, or credential-bearing configured site URL.

The correct boundary is semantic and structural: publisher fields must remain plain description and must not create document structure, executable references, or arbitrary navigation targets.

4.3 Impact

SnapSmack itself would not execute the content. The affected party would be a person using an inadequately sandboxed external agent to inspect the site. At worst, that agent could execute attacker-selected code with whatever local permissions the user granted it. Because the precise impact belongs to another program and depends on its permissions, the finding is rated HIGH as a preventive publishing-boundary failure rather than described as direct SnapSmack remote code execution.

5. Remediation in 0.7.575

5.1 Inert descriptive-text boundary

The new `snapsmack_llms_plain_text()` boundary:

- decodes entities and removes HTML tags;
- removes control characters and collapses all content to one line;
- applies explicit length limits: 120 characters for the name and 500 for the

description;

- rejects prompt-override phrases such as instructions to ignore prior rules;
- rejects system/developer/assistant role-labelled prompt text;
- rejects actionable package commands for pip, npm, pnpm, yarn, npx, gem,

Cargo, Go, Docker, and similar forms covered by the regression set;

- rejects command substitution, inline code markers, and pipe-to-shell syntax;
- removes publisher-provided URL schemes and Markdown control characters;
- omits suspicious descriptions instead of attempting to preserve or reinterpret

their meaning.

The description is emitted with an explicit label:

```
Publisher description (descriptive text only)
```

That label is defense in depth for machine readers. The security control remains the filtering boundary; a prose label alone is not a sandbox.

5.2 URL boundary

The new `snapsmack_llms_site_url()` boundary accepts only a syntactically valid HTTP or HTTPS URL with a hostname. It rejects:

- control characters and embedded whitespace;
- `javascript:`, `data:`, `file:`, and other non-web schemes;
- embedded usernames or passwords;
- malformed or incomplete URLs.

If the site URL is invalid, the generator omits the Key Pages section. It does not fabricate an `example.com` link and does not echo the unsafe setting.

5.3 Link policy

Generated links are limited to:

- the validated current site's Home and Archive locations; and
- the fixed, first-party `https://snapsmack.ca` attribution link.

Site descriptions cannot add their own links. The audit deliberately rejected an earlier proposal requiring every link to use the same hostname, because that would prohibit legitimate first-party SnapSmack attribution without reducing the relevant risk. The boundary is controlled ownership, not string equality.

5.4 No `llms-full.txt`

No SnapSmack source path generates `llms-full.txt`. The expanded variant creates more space for inherited links, setup examples, and stale dependency references, while SnapSmack's discovery and attribution needs are met by the short file. It will not be added without a separate security review.

6. Finding 052-B - integrity policy correction

Severity: INFORMATIONAL **Disposition:** Design corrected

The draft threat specification proposed placing every generated `llms.txt` under one signed release hash. That is not technically correct. Each installation has a legitimate site name, description, URL, AI policy, and SnapSmack version; therefore each generated file can differ while remaining valid.

The integrity boundary is `core/site-files.php`, the shipped generator. PHP source is included in the core signed release manifest and monitored by SMACKBACK. Fleet output verification compares the live file with output freshly generated from that site's current settings. A universal hash would either produce constant false alarms or force all sites to publish identical and incorrect data.

`llms.txt` was not added to `protected_paths.json`. That file controls what the updater refuses to overwrite; it is not a substitute for generated-output validation. Protecting `llms.txt` there could preserve a stale or poisoned file instead of allowing the reviewed generator to rebuild it.

7. Regression verification

`tests/llmstxt-agent-safety-regression.php` contains 21 checks. The suite verifies:

- safe names and ordinary photographic descriptions survive;
- the site's valid archive URL is produced correctly;
- first-party SnapSmack attribution remains present;
- prompt-override and role-injection phrases are omitted;
- representative pip, npm, npx, Docker, curl, shell-pipe, and inline-code forms

do not reach output;

- a publisher field cannot manufacture a Markdown heading or link;
- `javascript:` URLs and credential-bearing URLs are rejected;
- publisher-provided external URLs are absent;
- the generator retains both the text and URL boundary helpers;
- no `llms-full.txt` output path appears.

Check	Result
PHP syntax: `core/site-files.php`	PASS

Check	Result
PHP syntax: regression suite	PASS
Agent-safety regression suite	PASS - 21 checks
Git whitespace/error check	PASS
EOF marker check on changed source/test	PASS

The test corpus is intentionally not advertised as a complete dictionary of every possible natural-language jailbreak. The structural controls - one-line plain text, no publisher URLs, no Markdown controls, strict generated URLs - do most of the work. The phrase and command checks provide additional fail-closed coverage for known dangerous shapes.

8. Residual risk and limitations

8.1 A consuming agent can still be unsafe

SnapSmack cannot guarantee that another vendor's agent handles web content safely. A vulnerable agent may follow malicious instructions from an ordinary post, comment, linked webpage, README, issue, or search result. This remediation ensures SnapSmack's purpose-built `llms.txt` generator does not grant publisher settings extra structural or executable authority.

8.2 Administrative compromise remains serious

An attacker with full site administration may be able to alter templates, PHP, or the web root directly, depending on the compromise. Sanitizing settings does not turn an administratively compromised server into a trusted publisher. SMACKBACK, signed updates, hosting controls, and incident response remain the controls for that broader threat.

8.3 Domain custody is operational

The fixed `snapsmack.ca` attribution link is safe only while the project retains and securely administers that domain. Domain registration and account security remain operational dependencies. The important distinction is that the target is deliberate, first-party, and visible in reviewed source rather than supplied by arbitrary site settings or a transient package registry.

8.4 Live state follows deployment

Passing source tests does not prove every public site has updated and rebuilt its file. Release acceptance requires signed packaging, fleet update, regeneration, and live comparison. Until that occurs, the report must distinguish source remediation from fleet verification.

9. Release and fleet acceptance procedure

1. Commit the generator, regression test, release note, public audit entry, and

this report as one reviewable release slice.

2. Push to `dev` through the guarded release workflow.
3. Create the next sequential BITCHIN' candidate only after the complete release

candidate is ready; do not create a stable tag.

4. Package and publish through Smack Central's signed development channel.
5. Update the Hub and spokes.
6. Rebuild crawler files through Maintenance or by saving Global Configuration.
7. Fetch `/llms.txt` over HTTPS from every fleet domain.
8. Compare each live file with expected output generated from that site's current settings.
9. Confirm no live `/llms-full.txt` exists.
10. Record any stale file, failed write, CDN cache discrepancy, or host-specific divergence as a deployment failure rather than weakening the expected output.

10. Locked policy after SECAUDIT 052

- `llms.txt` is descriptive discovery and attribution data only.
- Publisher settings are data, never instructions.
- No actionable installation commands, package references, code examples, command substitutions, or fetch-and-run syntax.
- No publisher-supplied links.
- Generated links are limited to the validated current site and reviewed, hard-coded SnapSmack-controlled targets.
- The shipped generator is protected and monitored as code.
- Per-site output is verified against per-site expected generation, not a false universal hash.
- SnapSmack does not publish `llms-full.txt`.
- Suspicious input fails closed by omission.

11. Final disposition

SECAUDIT 052 found no executable reference in SnapSmack's existing `llms.txt` template and no direct server-side execution path. It did identify a real trust boundary that should be fixed before it becomes an incident: administrator-set descriptive fields could shape a document consumed as instructions by external agents.

The source remediation is complete in 0.7.575. It preserves useful discovery, photographer policy, and SnapSmack attribution while preventing those fields from adding Markdown structure, arbitrary links, or known command and prompt-injection forms. The remaining work is ordinary release discipline: signed delivery, regeneration, and fleet-wide live verification.