Security Review

security-audit (Claude Code Skill)

Date: 2026-02-06

Auditor: Claude Code (Automated Security Analysis)

Method: Static code analysis, configuration review, dependency analysis

Scope: Full review (all categories)

Baseline: security-audit-security-audit-2026-02-06.pdf (Score: 90)

Executive Summary

This is a follow-up security review of the security-audit Claude Code skill project. Since the initial audit (score 90/100), two low-severity findings have been resolved: L1 (IDE configuration is no longer tracked in git) and L2 (output directories are now excluded in `.gitignore`). The score improved from 90 to 92/100 (A). Two medium-severity findings remain open: an overly permissive Bash execution rule in the local Claude settings (`Bash(bash:*)`) and the use of the `--no-sandbox` flag in the Chrome headless command. No new findings were introduced by recent changes (review naming, cross-platform Chrome detection, README updates). No critical or high-severity findings exist.

Results Overview

0	0	2	0	6
CRITICAL	HIGH	MEDIUM	LOW	POSITIVE

Findings Overview

ID	SEVERITY	FINDING	FILE	OWASP	CWE
M1	MEDIUM	Overly Broad Bash Permission in Local Settings	`.claude/settings.local.json`	A05:2021	CWE-269
M2	MEDIUM	`--no-sandbox` Flag in Chrome Headless Command	`SKILL.md`	A05:2021	CWE-693
~~L1~~	RESOLVED	~~IDE Configuration Committed to Repository~~	~~`.idea/`~~	~~A05:2021~~	~~CWE-538~~
~~L2~~	RESOLVED	~~Missing Output Directory~~	~~`.gitignore`~~	–	~~CWE-538~~

ID	SEVERITY	FINDING	FILE	OWASP	CWE
		~~Exclusions in .gitignore~~			

Detailed Findings

M1 – Overly Broad Bash Permission in Local Settings

MEDIUM

Description: The local Claude Code settings file contains the permission rule `Bash(bash:*)`, which grants unrestricted bash command execution without user confirmation. This violates the principle of least privilege. If other skills or prompts are executed within this project context, they could leverage this broad permission to execute arbitrary commands without the user being prompted for approval.

Affected Code

File: `.claude/settings.local.json`, Lines 3-7

```
"permissions": {
    "allow": [
      "Bash(git add:*)",
      "Bash(git commit:*)",
      "Bash(bash:*)"
    ]
  }
```

Recommended Fix

```
"permissions": {
    "allow": [
      "Bash(git add:*)",
      "Bash(git commit:*)"
    ]
  }
```

OWASP: A05:2021 Security Misconfiguration CWE: CWE-269 Improper Privilege Management

M2 – `--no-sandbox` Flag in Chrome Headless Command

MEDIUM

Description: The Chrome headless command in the skill prompt uses `--no-sandbox`, which disables Chrome's sandbox security. While this flag is sometimes needed in Docker containers running as root, it is unnecessary and insecure on local macOS execution where the skill is designed to run. If the HTML file being rendered were to contain malicious content, the lack of sandbox would increase the attack surface.

Affected Code

File: `SKILL.md`, Lines 158-164

```
<resolved-chrome-path> \
  --headless --disable-gpu --no-sandbox \
  --print-to-pdf="docs/security-audit/[filename].pdf" \
  --print-to-pdf-no-header \
  "docs/security-audit/[filename].html"
```

Recommended Fix

```
<resolved-chrome-path> \
  --headless --disable-gpu \
  --print-to-pdf="docs/security-audit/[filename].pdf" \
  --print-to-pdf-no-header \
  "docs/security-audit/[filename].html"
```

OWASP: A05:2021 Security Misconfiguration CWE: CWE-693 Protection Mechanism Failure

L1 – IDE Configuration Committed to Repository ✓ RESOLVED

RESOLVED

Resolution: The `.idea/` directory is no longer tracked in git. The `.gitignore` rule was already in place, and the files have been removed from the git index. Running `git ls-files .idea/` returns empty.

L2 – Missing Output Directory Exclusions in .gitignore ✓ RESOLVED

RESOLVED

Resolution: Added `docs/epics/` and `docs/security-audit/` to `.gitignore`. Generated audit output files will no longer appear as untracked files when the skill is run against any repository.

Applied Fix

```
docs/epics/
docs/security-audit/
```

Positive Findings

No Hardcoded Secrets or Credentials: The codebase contains no API keys, tokens, passwords, or other sensitive credentials. All files are documentation or configuration only.

Robust .gitignore Configuration: Generated output files (PDF, HTML, epics) are properly excluded from version control, with explicit exceptions for the report template.

Safe Bash Defaults in Install Script: The `install.sh` script uses `set -euo pipefail`, ensuring that the script exits on errors, unset variables, and pipe failures.

Install Script Path Validation: The installer properly checks if the target path exists, differentiates between symlinks and regular files, and handles update vs. fresh install scenarios safely.

Zero Runtime Dependencies: The project has no `package.json`, `requirements.txt`, or similar dependency manifests. This eliminates the entire class of supply-chain vulnerabilities.

Symlink-Based Installation: The installation uses symbolic links rather than file copies, ensuring updates are immediately reflected without re-installation.

Risk Matrix

	LOW (IMPACT)	MEDIUM (IMPACT)	HIGH (IMPACT)	CRITICAL (IMPACT)
High (Likelihood)				
Medium (Likelihood)		M1		
Low (Likelihood)		M2		

References

OWASP Top 10:2021: https://owasp.org/Top10/

OWASP A05:2021 – Security Misconfiguration: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

CWE-269 – Improper Privilege Management: https://cwe.mitre.org/data/definitions/269.html

CWE-538 – Insertion of Sensitive Information into Externally-Accessible File or Directory: https://cwe.mitre.org/data/definitions/538.html

CWE-693 – Protection Mechanism Failure: https://cwe.mitre.org/data/definitions/693.html

OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/

CWE (Common Weakness Enumeration): https://cwe.mitre.org/

NIST NVD (National Vulnerability Database): https://nvd.nist.gov/

Docker Security Best Practices: https://docs.docker.com/engine/security/

OWASP Docker Security Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html