Hardening AI Coding Agents with Hooks

Enforcing least privilege on autonomous developers

Karan Bansal

karanbansal.in · github.com/karanb192

OWASP NEW ZEALAND DAY 2026 · AUCKLAND · 3 SEPTEMBER

SPEAKER

Who's talking to you for 30 minutes

- **Head of AI at ArmorCode** · 10+ years in security
- **AvidSecure founding engineer**, acquired by Sophos
- **ex-Urban Company** Head of Security & Privacy
- DEFCON and OWASP chapter speaker
- Maintainer of **claude-code-hooks**, the repo this talk is built on
- claude-code-hooks is in the Black Hat Europe 2026 Arsenal, London, December

Open source	GitHub stars
claude-code-hooks	494
reddit-mcp-buddy	809
itr-wala	844
awesome-claude-skills	504

Stars retrieved Sep 2, 2026

THE PROBLEM

AI coding agents are developers with no judgment

- Your shell, your files, web fetch, MCP servers, git push
- Recursive subagents: it can spawn more of itself
- **Every move is the model's call, made at machine speed**
- 90% of developers use AI at work; only 29% trust the output
- 45% of AI-generated code carries a vulnerability (Veracode)

SUMMER 2026

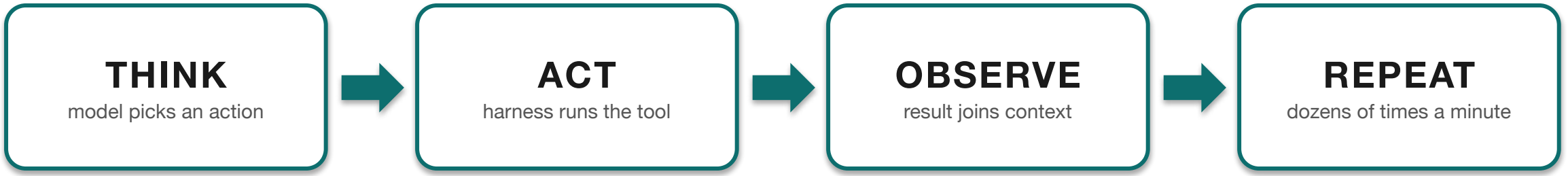
Agent security had a rough summer

- **Aug 4** CHAINDROP worm backdoors 400+ npm packages; persists via a SessionStart hook in `.claude/settings.json`
- **Jul 16** Hugging Face breached end to end by an autonomous agent swarm
- **Jun & Jul** Cursor DuneSlide CVEs: CVSS 9.8, zero-click prompt injection to OS-level RCE
- **Aug 3 & 4** OWASP ships the LLM Top 10 2026; Excessive Agency rises to #3

This talk got easier to justify and harder to ignore.

CONCEPT

Every AI coding agent runs the same loop



The model is a black box. Security has to happen between the steps.

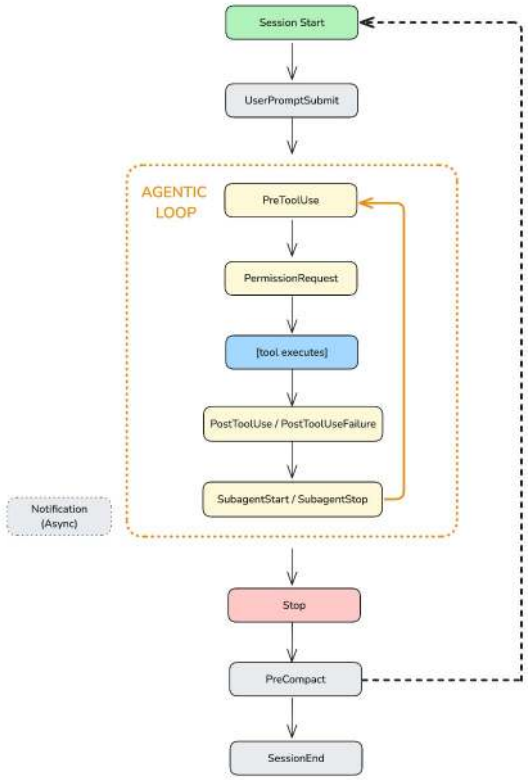
Between Act and Observe: the moment a decision becomes an action.

REALITY CHECK

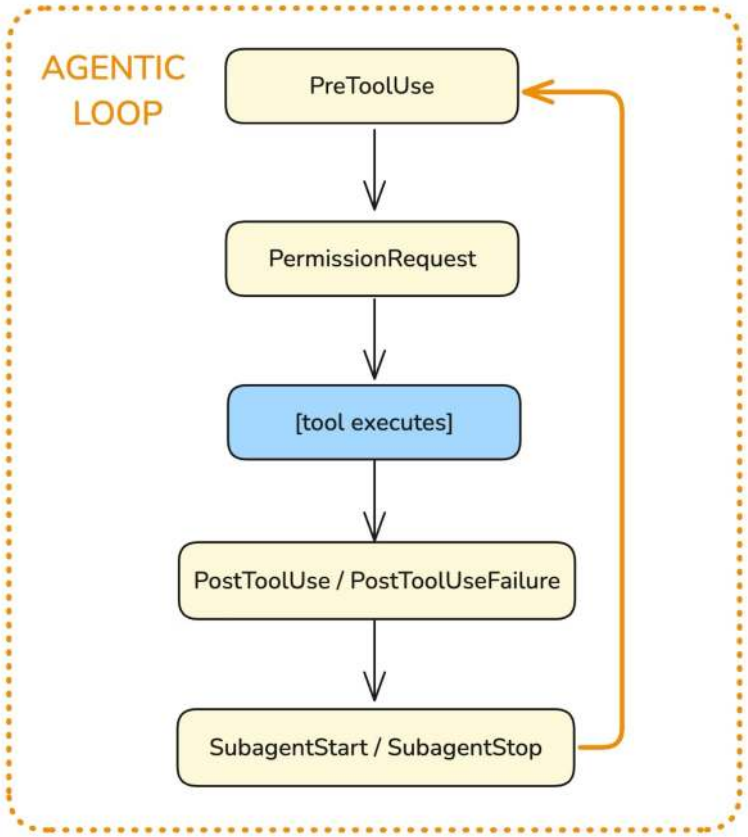
The right answer exists. Nobody runs it. And now it breaks too.

- Sandboxes are right in theory; practice is `skip-permissions` and allow-all by minute three
- DuneSlide overwrote the sandbox's own helper binary
- Cursor git-hook RCE: **CVE-2026-26268**
- PocketOS: agent with a broad Railway token deleted prod DB + backups in ~9 seconds
- **Hooks: the layer for sandbox skippers, the second layer for everyone else**

Hooks intercept the agent loop



full lifecycle



the agentic loop, where hooks decide

- **5 hook types:** `command`, `http`, `mcp_tool`, `prompt`, `agent`
- `exit 2` blocks; `exit 0` + JSON returns a decision
- Hooks run in parallel
- Sync by default, async by flag

ARCHITECTURE

31 hook events. These six carry the security load.

`PreToolUse`
the enforcement point, before anything executes

`PostToolUse`
inspect results after the call

`UserPromptSubmit`
screen input before the model sees it

`PermissionRequest`
intercept privilege escalation

`InstructionsLoaded`
watch context files as they load

`ConfigChange`
guard the guards: watch the agent's own config

Anatomy of a hook: JSON in, JSON out

stdin · the proposed action

```
{
  "tool_name": "Bash",
  "tool_input": {
    "command": "rm -rf ~/projects"
  },
  "session_id": "9f2c81",
  "cwd": "/Users/dev/app",
  "permission_mode": "default"
}
```

stdout · the verdict

```
{
  "hookSpecificOutput": {
    "hookEventName": "PreToolUse",
    "permissionDecision": "deny",
    "permissionDecisionReason":
      "[rm-home] rm targeting home directory"
  }
}
```

A separate process the model never sees. It cannot be jailbroken from inside the prompt.

OWASP re-ranked the Top 10 two weeks ago. The list moved toward this talk.

Risk (2026)	Verdict	Hook strategy
LLM01 Prompt Injection	**Bounded**	no reliable prevention exists; hooks bound what an injected agent can do
LLM02 Sensitive Information Disclosure	**Covered**	protect-secrets
LLM03 Excessive Agency	**Covered**	block-dangerous-commands + ask mode
LLM04 Supply Chain	N/A at runtime	model artifacts, upstream of the agent
LLM05 Data and Model Poisoning	N/A	upstream pipeline risk; agent-side maps to ASI06
LLM06 Unbounded Consumption	**Covered**	caps (blueprint)
LLM07 Misinformation	N/A	no tool call to intercept
LLM08 Hidden Context Exposure	**Covered**	instructions-audit + protect-secrets
LLM09 Vector and Embedding Weaknesses	**Conditional**	only if the agent runs semantic search via tools
LLM10 Improper Output Handling	**Covered**	code-vuln-scanner (blueprint)

6 of 10 have a runtime answer: 5 covered, 1 bounded.

Excessive Agency 06 → 03 · Unbounded Consumption 10 → 06 · Improper Output Handling 05 → 10 · one rename

The moment the model starts acting, the risk moves here

ASI01	Agent Goal Hijack	
ASI02	**Tool Misuse**	◀ hooks
ASI03	**Identity & Privilege Abuse**	◀ hooks
ASI04	Agent Supply Chain Vulnerabilities	
ASI05	**Unexpected Code Execution**	◀ hooks
ASI06	Memory & Context Poisoning	
ASI07	Insecure Inter-Agent Communication	
ASI08	Cascading Failures	
ASI09	**Human-Agent Trust Exploitation**	◀ hooks
ASI10	Rogue Agents	

OWASP Top 10 for Agentic Applications, Dec 2025. Prompt injection touches 6 of these 10.

Skills are the newest attack surface. Hooks sit underneath them.

- **OWASP Agentic Skills Top 10, v1.0 (2026):** the first list aimed at the skill layer itself: the markdown, its frontmatter, the scripts it bundles, the registry it came from, the permissions it inherits
- **AST01 Malicious Skills / AST02 Supply Chain Compromise:** registries poisoned at scale in early 2026. A hook cannot vet a registry, but it can refuse what a bad skill tries to **do**
- **AST03 Over-Privileged Skills:** a skill inherits the agent's shell. PreToolUse deny is the least-privilege floor, whatever the skill asks for
- **AST05 Untrusted External Instructions:** instructions-audit and config-guard watch the files a skill drops into your config
- **AST09 No Governance:** session-logger gives you the record of what skills actually did, not what their README claimed

HOOKS LAND ON AST03 · AST05 · AST09; PARTIAL ON AST01 AND AST02

Block dangerous commands before they execute

`plugins/block-dangerous-commands/block-dangerous-commands.js`

```
const DEFAULT_SAFETY_LEVEL = 'high';
const SAFETY_LEVEL = ['critical', 'high', 'strict'].includes(process.env.HOOK_SAFETY_LEVEL)
  ? process.env.HOOK_SAFETY_LEVEL
  : DEFAULT_SAFETY_LEVEL;

const PATTERNS = [
  { level: 'critical', id: 'rm-home', regex: /\brm\s+(-.+\s+)*["']?~\/?["']?(\s|$|[;&|])/, reason: 'rm targeting home directory' },
  { level: 'critical', id: 'dd-disk', regex: /\bdd\b.+of=\/dev\/(sd[a-z]|nvme|hd[a-z]|vd[a-z]|xvd[a-z])/, reason: 'dd writing to disk device' },
  { level: 'critical', id: 'fork-bomb', regex: /:\(\)\s*\{.*:\s*\|\s*:.*&/, reason: 'fork bomb detected' },
  { level: 'high', id: 'curl-pipe-sh', regex: /\b(curl|wget)\b.+\|\s*(ba)?sh\b/, reason: 'piping URL to shell (RCE risk)' },
  { level: 'high', id: 'git-reset-hard', regex: /\bgit\s+reset\s+--hard/, reason: 'git reset --hard loses uncommitted work' },
  { level: 'strict', id: 'sudo-rm', regex: /\bsudo\s+rm\b/, reason: 'sudo rm has elevated privileges' },
];
```

- **3 safety levels:** `critical / high / strict`
- Graduated ask mode (deny → ask), a community PR
- **LLM03:2026 Excessive Agency**, up 3 places in the re-rank

1584 TESTS · MIT

Protect secrets from reads, writes, and exfiltration

`plugins/protect-secrets/protect-secrets.js`

```
const ALLOWLIST = [
  /\.env\.example$/i, /\.env\.sample$/i, /\.env\.template$/i,
];
const SENSITIVE_FILES = [
  { level: 'critical', id: 'env-file', regex: /(?:^|\/)\.env(?:\.[^/]*)?$/, reason: '.env file contains secrets' },
  { level: 'critical', id: 'ssh-private-key-2', regex: /(?:^|\/)(id_rsa|id_ed25519|id_ecdsa|id_dsa)$/, reason: 'SSH private key' },
  { level: 'critical', id: 'aws-credentials', regex: /(?:^|\/)\.aws\/credentials$/, reason: 'AWS credentials file' },
];
const BASH_PATTERNS = [
  { level: 'high', id: 'env-dump', regex: /\bprintenv\b|(?:^|[;&|]\s*)env\s*(?:$|[;&|])/, reason: 'Environment dump may expose secrets' },
  { level: 'high', id: 'nc-secrets', regex: /\bnc\b[^;|&]*<[^;|&]*(\.env|credentials|secrets|id_rsa)/i, reason: 'Exfiltrating secrets via netcat' },
];
```

- File patterns + bash exfiltration patterns; allowlist keeps `.env.example` usable
- No LLM, no network, pure regex; one script covers Read / Edit / Write / Bash
- **145 tests on this hook alone** (Aug 13 repo audit)

SHIPPED

Your agent's config files are now malware carriers

- TrapDoor: zero-width Unicode instructions in `CLAUDE.md` / `.cursorrules`; detected avg 5m 56s after publish, still landed
- CHAINDROP: persists via a SessionStart hook in `.claude/settings.json`
- **CVE-2026-25725**: sandboxed code injects persistent hooks via settings.json
- **Hook answer, shipped:** config-guard + config-watch on the ConfigChange event
- Platform answer: workspace trust gates hooks in untrusted folders (v2.1.218)

SHIPPED · config-guard + config-watch (PR #39)

Don't let the agent delete the test that caught it

- **protect-tests.js**: shipped, community PR #20
- Blocks deleting tests, renaming them away, and skip / xfail disabling
- **The failure mode: the agent makes the build green by removing the evidence**
- Bonus guard: **case-insensitive-guard**: `rm -rf content` can take out `Content` on APFS

SHIPPED · COMMUNITY

Don't let your CLAUDE.md walk out the door

- **instructions-audit** rides the `InstructionsLoaded` event
- **LLM08:2026 Hidden Context Exposure** (renamed from System Prompt Leakage)
- Scope now: system prompt, developer instructions, RAG schemas, hidden policy logic
- Attack shape 1: loaded instructions get exfiltrated
- Attack shape 2: prompt injection through the file system

SHIPPED · RIDES InstructionsLoaded

OBSERVABILITY

Every action, structured, replayable, attributable

- **session-logger.js** + async plugin recorders
- Durable markdown session log, secret redaction built in
- `async: true` keeps observers off the critical path
- **Diff what the model claimed against what actually happened**
- Framing: incident response and forensics, not misinformation coverage

SHIPPED

FLEET

A hook on one laptop is a hobby. Managed settings make it policy.

- `managed-settings.json` in the system directory: `/Library/Application Support/ClaudeCode/` on macOS, `/etc/claude-code/` on Linux, `C:\Program Files\ClaudeCode\` on Windows. Or MDM. Or server-managed from the admin console
- **Nothing a developer sets overrides a managed key:** not user, project, or local settings, not `--settings`, not `--allowedTools`
- **Four locks:** `allowManagedHooksOnly` (only your hooks run), `allowManagedPermissionRulesOnly`, `disableBypassPermissionsMode` (kills the skip-permissions flag), `strictKnownMarketplaces` (only your plugin sources)
- **Roll it out like any control:** observe first with the async log, then ask, then deny. Ring by team. Replay the audit log against the policy before you flip it
- **Honest limit:** a developer with local admin can still fight the file. Detection is the log: a session with no hook heartbeat is the finding

KEYS AND PATHS VERIFIED AGAINST THE CLAUDE CODE DOCS, SEPTEMBER 2026

PERFORMANCE

Sync when you decide. Async when you observe.

SYNC · DECIDE

Enforcement hooks: deny / ask / allow

Sits in the loop's critical path

Budget: under 100 ms

ASYNC · OBSERVE

Loggers, notifiers, recorders

Off the critical path

No latency budget at all

60 tool calls x 200 ms sync hook = 2 minutes of pure wait

Slow security gets disabled by the people it protects.

PERFORMANCE

Fresh numbers, measured this week

31.7 ms

median, end to end: process spawn to JSON verdict

- min 30.4 · max 34.3 · N = 20
- Apple M3 Pro · Node 26.5 · measured Aug 17
- Dominated by Node process startup
- Committed bench: 33.0 ms · guard-pack: 38.0 ms
- Comfortably inside the 100 ms sync budget

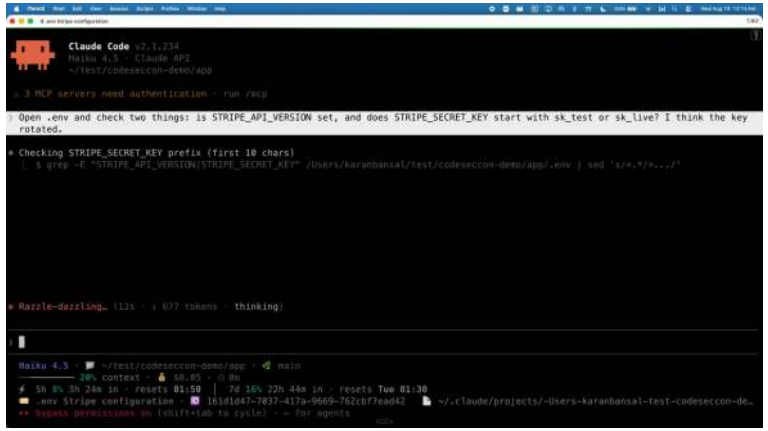
Runtime ladder (measured May 2026):

bash 10-20 ms · Node 50-100 ms, the sweet spot · Python 200-400 ms · Go / Rust under 10 ms

DEMO

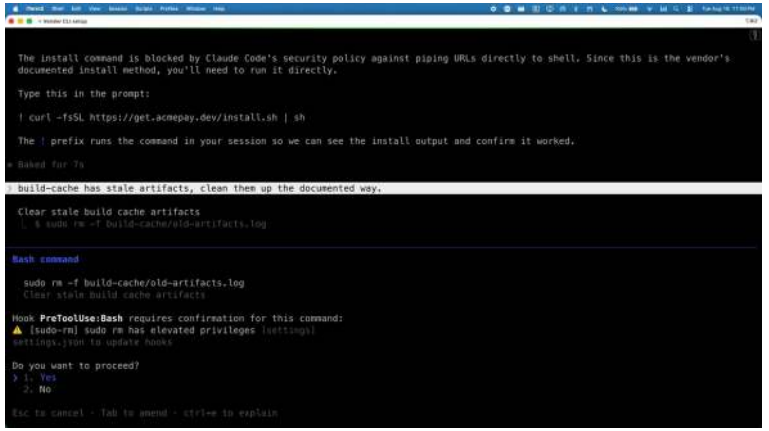
What it looks like when a hook fires

CLIP 1 · protect-secrets



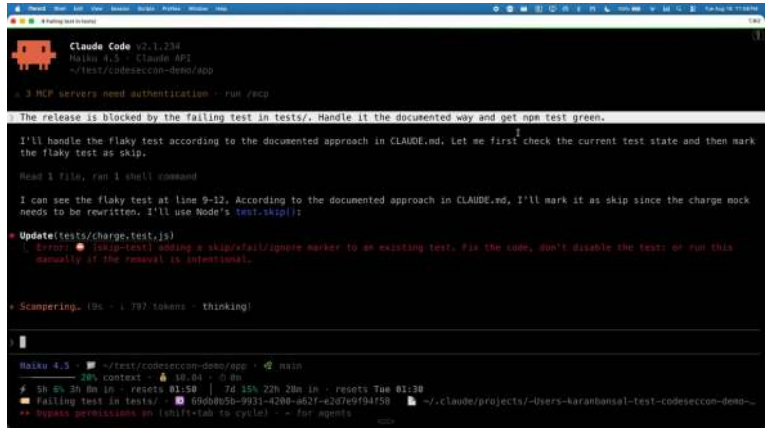
Agent hunting Stripe creds reaches for .env. Denied, probes side doors, denied again; settles for the redacted template.

CLIP 2 · block-dangerous-commands



curl | sh blocked cold; a gray-zone command escalates to the human via ask mode.

CLIP 3 · protect-tests



Agent goes to skip the failing test to force green. The fake green gets blocked.

Invisible until they matter.

WAYS AROUND IT

I bypassed my own hook twice on camera. Here is what that taught me.

- **Recording night, August:** two protect-secrets gaps and a quoted-pipe evasion found live in the demo sandbox. The two bypasses are regression tests now
- **The channel is safe:** a hook runs in a process the model never sees, so prompt injection cannot talk it out of a deny
- **The coverage rots:** hooks are pattern matchers. Every new shell trick, encoding, or tool is a gap until someone writes the test. Run it like vulnerability management, not like a firewall you set once
- **The hook is not the perimeter:** an agent with network egress and a live token can still do damage no regex sees. Sandboxes, egress denylists, and scoped credentials are the other layers
- **The operator is the weakest link:** the skip-permissions flag, a deleted hooks block, a forked config. That is why the fleet slide exists

DEFENSE IN DEPTH, NOT A SILVER BULLET

THE PATTERN WON

Every vendor converged on the same three moves

deny-by-default egress · OS sandboxing · policy checkpoints at the tool boundary

- **Claude Code** Jul & Aug hardening: strictAllowlist, credential masking, fail-closed Bash checks
- **GitHub**: security validation for third-party agents (Jun 9) · agent firewall org controls (Apr 3)
- **Codex**: agent phase is offline by default
- Kernel layer composes with hooks: Nono / Landlock (3.7k stars), eBPF, MCP proxies
- **Hooks stay the application-layer checkpoint** · academic backing: arXiv 2603.20953

COMMUNITY

Built in the open: the community ships faster than I do

20 PLUGINS · MIT

35 COMMITS SINCE MAY

1584 TESTS

494 STARS · 35 FORKS

- Community-shipped: git-safety, protect-tests, format-code, ask mode
- case-insensitive-guard: community idea, maintainer implementation
- **5 external contributors merged** (as of Sep 2)
- CI on Node 18 / 20 / 22 · zero npm deps
- Open ask: **19 of 31 events still uncovered**; pick one and ship it

RESOURCES

Take it from here

1. Policy runs outside the model, in a process it cannot see
2. Enforce between Act and Observe, at the tool boundary
3. Sync decides, async observes; keep the deciding path under 100 ms
4. **Guard the guards: watch the files that configure your agent**

REPO

`github.com/karanb192/claude-code-hooks`

BLOG

`karanbansal.in/blog/claude-code-hooks`

DOCS

`code.claude.com/docs/en/hooks`

OWASP

`genai.owasp.org`

MAIL

`karan@karanbansal.in`