CHEAT SHEET

Coding Agent Security

Practical defense-in-depth for teams using AI coding agents.

Don't combine untrusted input, reachable secrets, and open egress in one agent.

Christian Schneider
Security Architect · Hacker · Trainer
christian-schneider.net

01 Everything it reads is an instruction

Models don't reliably separate instructions from data.

Ask who could have influenced it. Issues, READMEs, web pages, docs, even error-tracker events can carry hidden instructions. If an attacker could have shaped any part of it, drop auto-approve and permissions for that task.

Inspect foreign repos from a sandbox, not your IDE. Rules files steer the model; MCP configs, hook installers, build scripts, and devcontainer commands execute code once their trigger fires: browse restricted, run only contained.

Treat the agent's output as untrusted too. Review diffs like a stranger's PR, especially new dependencies and lockfile changes; a hijacked agent ships plausible-looking code.

Stop — don't just restart — when behavior turns odd. A fresh session clears context, not poisoned rules, memory, or configs; inspection and escalation live in **06**.

02 Gate actions with code, not vigilance

Approval prompts are the last human checkpoint: make policy enforceable.

Require approval for shell, network, installs, pushes. Most agents ship an auto-approve mode; know yours by name and keep it off on your daily machine.

Deny-rule secret paths and dangerous commands. Block reads of `.env*`, SSH keys, and cloud credential paths. Path denial is a tripwire, not a wall: isolation and secrets off disk are the real controls.

Enforce policy as hooks before each tool call. Prefer narrow allowlists: deny patterns are obfuscatable, and even allowed interpreters run arbitrary code. Keep hooks outside the agent's write reach.

Gate git push through an approval proxy. Intercept and hold pushes for review, and close the direct route to the VCS host by network and credential design. (FINOS GitProxy)

Fix the setup when you reflex-approve. Approval fatigue is a predictable failure mode: sandbox more and scope tighter instead of clicking faster.

03 Contain it: sandbox and starve the network

Containment does the real work; dialogs are just the seatbelt.

Sandbox any task touching untrusted input. Use your own hardened definition, never the repo's checked-in one: workspace-only, non-root, no host Docker or SSH sockets, fail closed. For outright hostile repos, prefer a throwaway VM. (devcontainers · Anthropic sandbox-runtime)

Work on a disposable clone, mounted alone. Keep home, dotfiles, and other repos out; export a reviewed patch back, and inspect `.git` config and hooks: normal diffs don't show them.

Default-deny outbound traffic with an allowlist. Permit package mirrors and your VCS host only; block the host gateway, internal ranges, and metadata endpoints. Hostnames can't tell your repo from an attacker's, so broker writes separately.

Route DNS through your approved resolver. Block direct and encrypted DNS around it, or DNS tunneling stays open as an exfiltration channel.

Apply the same rules to agents in CI. Ephemeral runners, job-scoped short-lived tokens, restricted egress. Keep untrusted pull requests out of secret-bearing workflows.

Never run auto-approve outside a sandbox. Reserve full autonomy for disposable, credential-free environments, never your workstation.

04 Put secrets out of reach

Nothing sensitive lying around to read: brokers beat handing tokens over.

Replace plaintext .env files with schemas. Commit names and types; resolve values at runtime from a secret manager with best-effort redaction. Code the agent runs still reads what the app reads, so pair with scoped tokens. (varlock)

Prefer brokers over readable tokens. Keychains raise the bar over dotfiles, but an authorized process can still use the token; best is a broker performing the operation without revealing it.

Scope every token to the task. Fine-grained, short-lived, per-repository: an agent fixing one repo needs no org-wide write access.

Sanitize what you paste into sessions. Error messages and config dumps carry tokens and customer data; transcripts can persist locally and vendor-side; check your retention settings.

Turn on server-side push protection where available. Pattern-based blocking catches known secret formats, not everything; scan locally as early warning. (gitleaks · TruffleHog)

05 Vet extension surfaces like dependencies

MCP servers, skills, rules files, packages: your new supply chain.

Read MCP tool descriptions before connecting. Clients that expose tool metadata to the model let a malicious description steer behavior, even if that tool is never called.

Pin digests and watch tool definitions. Immutable digests over `@latest`; change detection catches the rug pull. Scan foreign MCP configs only from a disposable sandbox, since scanning can execute them. (Snyk Agent Scan, ex-Invariant mcp-scan)

Run MCP servers isolated and least-privilege. Containerize with network isolation on; give each its own narrow token, read-only where possible, never your personal one. (Stacklok ToolHive)

Review rules and memory files like code. They are system-prompt extensions; diff them in every PR and audit what the agent has "remembered."

Verify agent-suggested packages before installing. Models hallucinate package names and attackers register them, and post-install scripts run the moment the agent installs, before any review; check existence, age, and provenance first.

Uninstall what you don't use. Every unused server, skill, and tool is reachable attack surface without benefit.

06 Assume prevention fails

Allowlisted channels leak; detection decides incident vs. breach.

Turn on, test, and read the logs. Proxies, sandboxes, and hooks log only what you enable: capture successful high-risk actions too, not just denials, and route alerts where someone reads them.

Plant tripwires an attacker will trip. Fake credentials that alert on use turn silent exfiltration into a fire alarm; verify the alert path actually fires. (Thinkst Canarytokens)

Escalate odd behavior as a security signal. A project that suddenly "acts different" means: inspect rules, memory, and configs; tell your team, not just your prompt.

Stop, preserve, contain, then revoke. Halt the agent, keep transcripts and logs, revoke exposed sessions and tokens, rotate per your incident plan. Rebuild sandboxes rather than trusting cleaned ones.

IF YOU DO NOTHING ELSE THIS MONTH

1. **Turn off standing auto-approve — today.** The zero-cost stopgap while you build the rest; enforce it in admin or team config.
2. **Move secrets out of the workspace.** Deny-rule secret paths this week; start the broker and secret-manager migration this month.
3. **Sandbox tasks that touch strangers' text.** Your own hardened, fail-closed definition: no credentials, restricted egress.
4. **Inventory and vet every MCP server and skill.** Remove the unused, pin the rest, watch for definitions that change after approval.
5. **Default-deny outbound traffic where supported.** Allowlist mirrors and your VCS host; expect churn in week one.