

Lesson 09 — Secret and Credential Isolation

The prompt is an egress channel. Every secret that enters an AI agent's context is a secret that has left your trust boundary. This lesson is the discipline that keeps that from mattering.

Audience: Devs + Security + Operators

AI OpSec

Secrets

Notes: Scope first, rotate second. The goal is that a leaked prompt is a survivable event.

What you will learn

- **Scope secrets** — per-repo, per-project, per-API, never org-wide
 - **Rotate them** — per run for agent runs, daily for engagement tokens, never "set and forget"
 - **Keep agents from seeing more than they need** — the never-to-an-agent list, pre-send scanners, distinct MCP trust boundaries
 - Outcome: a leaked prompt is a nuisance, not an incident
-

The problem with long-lived admin tokens in agent context

One leaked prompt = full access.

A long-lived admin token pasted into an agent's context window is indistinguishable, to the outside world, from that same token pasted into a public Slack channel. Both end up in logs, both end up in model provider training buffers, both are one prompt-injection away from being read back verbatim to an attacker.

- Agents log context. Model providers log context. You will not see every copy.
 - "It was only in the prompt" is not a mitigation — it is a distribution mechanism.
-

Scope narrows blast radius

- **Per-repo:** a GitHub token scoped to one repo cannot touch the other 400
- **Per-project:** an OpenAI key with a spend cap and a project tag cannot drain the org budget
- **Per-API:** a Stripe restricted key scoped to "read customers" cannot issue refunds
- **Short TTL:** a 1-hour STS session is worthless to an attacker who finds it in a 24-hour-old log

Scope is the primary control. Rotation is the secondary control. Both matter, but scope matters first — an unscoped token that rotates hourly is still an admin token for an hour.

Per-project `.env` files

- **One file, one scope.** Each project's `.env` contains only the credentials that project needs.
- **No shared vault for agents.** A "dev secrets" file with every key you've ever used is a liability, not a convenience.
- Never copy an `.env` between projects. If project B needs the same API, mint a new key for project B.
- `chmod 600`, `.gitignore`, and never paste the contents into a chat window.

```
project-a/.env      # keys for project A only
project-b/.env      # keys for project B only
# no ~/.env, no ~/secrets.env, no shared .env
```

The never-to-an-agent list

- **Long-lived PATs** — classic GitHub/GitLab tokens with repo-admin or org-admin
- **Root API keys** — the default org key for OpenAI, Anthropic, AWS, Stripe, etc.
- **SSH private keys** — `~/.ssh/id_rsa`, `~/.ssh/id_ed25519`, any unencrypted key
- **Cloud credentials with console access** — anything that grants web-console login, not just API

If one of these ends up in an agent's context, treat it as compromised. Revoke, do not rationalise.

Rotation cadence

- **Agent-run tokens:** rotate *per run*. Mint at start, revoke at exit. TTL in minutes.
- **Engagement-bound tokens:** rotate *daily*. A fresh token each morning, old one revoked.
- **Interactive dev tokens:** rotate *weekly* at most, and only if scoped narrowly to begin with.
- **Never "set and forget."** A token with no rotation schedule is a token you will forget to rotate.

Rotation without scope is hygiene theatre. Scope without rotation is slow decay. You need both.

Secret scanning your prompts

- **Pre-send hook:** intercept the prompt before it leaves your machine, grep for known secret shapes, refuse to send on match
- Patterns that catch 90% of incidents with zero false positives:

```
AKIA[0-9A-Z]{16}          # AWS access key ID
ghp_[A-Za-z0-9]{20,}      # GitHub PAT (new format)
xoxb-[A-Za-z0-9-]+        # Slack bot token
-----BEGIN [A-Z ]+PRIVATE KEY-----
sk-[A-Za-z0-9]{20,}       # OpenAI / Anthropic API key
```

The scanner's job is not to be clever. Its job is to be loud when you do something you'll regret.

Credential helpers and IDE plugins

- AI coding plugins read open files, recent clipboard, env vars, and sometimes the running process list
 - Credential helpers (git-credential, osxkeychain, gh auth) store tokens that the IDE's AI assistant can and does include in context
 - A `.env` open in a neighbouring tab can end up in the prompt *as context*, not as a deliberate paste
 - **Close secret files before you prompt.** Audit which plugins you have installed. Assume every plugin is a context source until proven otherwise.
-

MCP server credentials

- Each MCP server is a distinct trust boundary. A compromised or malicious MCP reads every token it is given.
- **Do not give all MCPs the same token.** A shared "MCP token" defeats the entire point of per-server isolation.
- Mint a token per MCP, scoped to exactly what that MCP needs, with the shortest TTL the server tolerates
- Review every new MCP the way you would review a new vendor: what token, what scope, what rotation, who runs it

An MCP you trust today is an MCP whose publisher account got compromised tomorrow. Design for that.

Rules to remember

- Scope first, rotate second. An unscoped rotating token is still too powerful.
 - Per-project `.env` — one file, one scope. Never shared across projects or agents.
 - Long-lived admin tokens never go into agent context. Ever.
 - Rotate per run for agent runs. Rotate daily for engagement tokens.
 - Run a secret scanner over every prompt before it leaves your machine.
 - Treat each MCP server as a separate trust boundary. Distinct tokens.
-

