Secret Management for Agents — One-Page Checklist

Companion to Agentic AI Security Training — Chapter 7
Use this checklist when issuing, storing, or rotating any credential an agent will use.

Principle in one sentence

Secrets are how an agent inherits authority over the systems it interacts with — keep that authority short-lived, narrowly scoped, redacted from logs, and absent from the prompt context.

Five locations to inspect for leaks

1. **Environment.** `OPENAI_API_KEY=sk-...` in `.bashrc`, on a CI runner, in a checked-in `.env`. Long-lived, broadly scoped, infrequently rotated.
2. **Agent's own context.** Secrets in system prompts, agent files (`CLAUDE.md`, `AGENT.md`, `.cursor/rules`), or retrieval results — sent on every model call.
3. **Tool arguments.** `slack_post_message(token="xoxb-...")` — captured in transcript, harness audit, provider logs.
4. **Tool outputs and fetched content.** A page contains a secret in plain text; the agent's summary includes it.
5. **Persistent files.** Logs, transcripts, scratch files — every secret the agent has ever handled.

L1 — environment variables (the floor)

- [ ] No cleartext secrets in workspaces or configuration files.
- [ ] No secrets in checked-in code, dotfiles in repos, agent files, or test fixtures.
- [ ] `.env` files never committed to version control.
- [ ] No secrets in skill files, plugin configurations, MCP server configurations, or scheduled-task scripts.
- [ ] Per-host environment files (`.envrc`, systemd unit) populated at boot from a more secure source.
- [ ] CI/CD secrets injected at runtime; strict no-echo policy.

L2 — password manager or KMS

- [ ] No cleartext secrets on the host filesystem either.
- [ ] Secrets pulled from 1Password / Bitwarden / Vaultwarden / cloud KMS into the process environment at session start, never written to disk.
- [ ] `op run` (1Password CLI) wraps each agent's launch command; per-agent scoped sets.
- [ ] `aws-vault exec` for AWS, issuing short-lived STS tokens via OS keychain.
- [ ] `gcloud auth application-default print-access-token` for GCP — short-lived OAuth tokens on demand.

L3 — vault with just-in-time issuance

- [ ] Vault (HashiCorp Vault, Akeyless, AWS Secrets Manager + IAM dynamic credentials) issues credentials at tool-call time.
- [ ] Database access: 5-minute Postgres role per session; revoked at session end.
- [ ] Source control: fine-grained PAT, single repo, 1-hour expiry.
- [ ] Cloud access: STS / OIDC federation, 15-minute tokens scoped to single resource.
- [ ] Per-tool-call credential — the PEP requests, embeds in outbound request, discards.
- [ ] Each issuance traceable to the agent, the tool call, and the resulting action.

Output redaction (PostToolUse hooks)

- [ ] Redact API-key formats: `sk-...`, `xoxb-...`, `gho_...`, `AKIA...`.
- [ ] Redact JWT-like strings.
- [ ] Redact long base64 blobs above a configurable length threshold.
- [ ] Redact content following `Authorization: Bearer`.
- [ ] Redact organisation-specific secret patterns.
- [ ] Use `detect-secrets`, `gitleaks`, or `trufflehog` pattern sets.
- [ ] Replace rather than remove: `sk-...abcd` → `<REDACTED:openai_api_key>`.
- [ ] Redact tool arguments too (PreToolUse), not only outputs.

Per-agent identity

- [ ] No shared credentials between agents.
- [ ] GitHub: fine-grained PAT, one repo, specific permissions, ≤ 30-day expiry.
- [ ] Slack: per-bot OAuth, channel-scoped, rotated quarterly.
- [ ] Cloud: per-agent IAM role, OIDC federation, no long-lived access keys.
- [ ] Database: per-agent role, just the tables/columns needed.
- [ ] No tokens in the agent's filesystem.

Rotation and revocation

- [ ] Long-lived tokens rotated every 30 days; webhook secrets every 90 days.
- [ ] Rotation automated, not manual.
- [ ] Revocation as kill switch: one command, single source of truth.
- [ ] Revocation drilled: "What breaks if I revoke the coding agent's GitHub token right now?"
- [ ] Any secret that has appeared in a log, transcript, or tool output is rotated immediately.

Anti-patterns to remove

- [ ] API keys exported from `.bashrc`.

- [ ] Shared `.env` committed to a private repo.
- [ ] Tokens in long-lived agent files for "context".
- [ ] Personal OAuth tokens used for agents.
- [ ] Verbose debug logging that captures tool arguments.
- [ ] Reliance on the model to redact secrets.
- [ ] Token sharing across multiple agents.
- [ ] Temporary production access granted informally.