ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 10 / 30 · AI AGENT EXPOSURE

When AI Agents Leak Their Own Memory

112 Exposed Session Servers

How `.claude`, `.codex`, `.cursor`, `.openclaw` and `.hermes` folders turn open file servers into credential leaks

Two-week observation window ending 18 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

112
Servers exposing hidden AI agent folders over HTTP; 100 of them hold real user files

3,145
Secret and session matches, excluding bundled skill documentation (some vendor source matches remain)

695
User session, subagent, tool and credential file paths

919 / 553
AI model keys and command-and-control or callback URLs, the two largest match categories

What Is in This Report. 12 chapters and two appendices.

How .claude, .codex, .cursor, .openclaw and .hermes folders turn open file servers into credential leaks

01	Executive Summary	04
02	How AI Agents Store Their Memory on Disk	07
03	What Leaks: Secret Categories	11
04	Case Study: One Workspace on Two Servers	14
05	Backups, Trajectories and Resets	16
06	Co-Hosted Multi-Agent Servers	19
07	Bundled Files and Who Owns These Servers	22
08	What Defenders Should Do	24
09	Detection Rules	26
10	Indicators	29
11	Scope and Limits	31
12	Sources	33
A	Indicators of compromise (0)	35
B	Detection rules (4 files)	36

IN SHORT

Odin by Brandefense found 112 servers that expose the hidden home folders of five AI coding and task agents over plain HTTP. One hundred of them hold real user files. These folders keep full session transcripts, compaction summaries and backup copies, so API keys and bot tokens survive cleanup. A quarter of the servers run two or three agents and reuse keys across them. Block non-loopback file servers, deny dotfile paths at the proxy, and wipe agent session folders whenever you rotate keys.

01

CHAPTER 01

Executive Summary

IN SHORT

Odin by Brandefense found 112 servers that expose the hidden home folders of five AI coding and task agents over plain HTTP. One hundred of them hold real user files. These folders keep full session transcripts, compaction summaries and backup copies, so API keys and bot tokens survive cleanup. A quarter of the servers run two or three agents and reuse keys across them. Block non-loopback file servers, deny dotfile paths at the proxy, and wipe agent session folders whenever you rotate keys.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 04

01 Executive Summary

Developers now run AI coding agents on rented Linux servers instead of laptops. They paste API keys, cloud tokens and database URLs into the agent's conversation. The agent writes all of it to hidden folders in the home directory. When the owner later starts a quick file server in that home folder, every one of those files can be downloaded by anyone.

1. **AI agents keep secrets in transcripts, summaries and backups, not just config files.**
   Session logs, automatic context summaries, pre-edit file history and reset backups all hold full keys and shell output. Closing the terminal or resetting the session does not remove them. One open file server publishes all of it.

2. **Nearly one quarter of exposed servers run two or three AI agents together.**
   Of the 112 servers, 26 (23.2%) host two or three agent stacks in the same account. On several of them, the same API key or bot token appears in more than one agent. Rotating the key in one config file is not enough.

3. **Bundled skill files inflate secret scans.**
   Open-source skill packs ship example keys and local URLs, above all in Hermes folders. Defenders who scan agent folders must exclude them before opening an incident.

Three Decisions for Leaders

1. **Ban non-loopback file servers on engineering and cloud hosts.** Block unauthenticated directory servers bound to all interfaces on ports such as 4000, 8000, 8080, 8888 and 9999. Deny hidden dotfile paths at every reverse proxy.
2. **Treat agent session folders as secret stores.** Subagent summaries, file history and backup snapshots hold the same keys as config files. Wipe them every time a key, token or SSH key is rotated.
3. **Route exposed servers to remediation, not to block lists.** Most of these servers belong to careless developers. Exclude bundled skill documentation from scans, rotate the exposed keys, and fix the server.

The Study in Numbers

Number	What it is
112	Servers exposing hidden AI agent folders over HTTP; 100 of them hold real user files
3,145	Secret and session matches, excluding bundled skill documentation (some vendor source matches remain)
695	User session, subagent, tool and credential file paths
26 (23.2%)	Servers running two (20) or three (6) AI agent stacks side by side
919 / 553	AI model keys and command-and-control or callback URLs, the two largest match categories
83 (74.1%)	Servers that most likely belong to developers who exposed their home folder by mistake

How to use this report.

Security leaders can read Chapter 1, Chapter 8 and Chapter 11. Cloud and DevSecOps teams should read Chapters 2 to 6 for the file layouts. SOC and detection engineers should start with Chapters 8 and 9.

02

CHAPTER 02

How AI Agents Store Their Memory on Disk

IN SHORT

Each agent writes its conversations, tool output and credentials to a hidden folder in the user's home directory. A quick file server started in that home directory turns those folders into a public download.

INSIDE THIS CHAPTER

How the exposure happens 07

Matches by agent 07

Separating user files from vendor files 07

Four layers of agent memory 08

Which ports the servers use 09

02 How AI Agents Store Their Memory on Disk

How the exposure happens

Developers rent Linux virtual private servers (VPS) and install several command-line agents under `/root` or `/home/<user>`. To move files or preview a web build, they run `python3 -m http.server` or turn on Nginx `autoindex on` in their home folder. Neither tool hides dotfile folders. Every agent folder becomes readable over HTTP without a password.

The five agents in this study use these folders:

Agent	Hidden folder	What it is
Claude Code	`/.claude/`	Anthropic command-line coding agent with subagents
Hermes Agent	`/.hermes/`	Open-source autonomous agent with multi-profile bots and skill packs
OpenClaw	`/.openclaw/`	Open-source autonomous agent gateway with trajectories and auth profiles
Cursor IDE	`/.cursor/` and `/.cursor-server/`	IDE with remote SSH server and agent transcripts
OpenAI Codex CLI	`/.codex/`	OpenAI command-line coding agent with rollout logs

Matches by agent

The five folders hold 3,145 secret and session matches on 112 servers, excluding bundled skill documentation.

Agent	Matches	Servers (with user files)
Claude Code	988	26 (25)
Hermes Agent	967	75 (58)
OpenClaw	932	21 (19)
Cursor IDE	184	6 (3)
OpenAI Codex CLI	74	16 (14)

The server counts add up to 144 agent installs on 112 servers, because 26 servers run more than one agent. On the 99 servers whose earlier directory listings show user files, there are 119 installs.

Hermes appears on the most servers, but few matches per server. On 11 single-agent Hermes servers, the snapshot held only bundled OpenSSL or npm files or upstream source code. Claude Code and OpenClaw tell the opposite story. They show few servers but many secrets per server, mostly inside live `.jsonl` session logs.

Separating user files from vendor files

The servers expose 695 user-state host and path pairs. Vendor bundles and upstream runtime source are not counted as user data. They include 29 copies of `/.hermes/hermes-agent/agent/redact.py`, the module Hermes uses to hide secrets.

The other 695 pairs are user sessions, subagent files, tool output and credential stores. Thirteen single-agent servers held only vendor or source paths in their earlier listing, which leaves 99 servers with user files. A full listing of Host H dated 18 November 2026 adds a Hermes session log, for 100 servers in total.

The earlier listings and the November listings count different things, so no combined match total is given. The 3,145 figure counts matches, not confirmed leaks.

Four layers of agent memory

The 695 user paths fall into four layers.

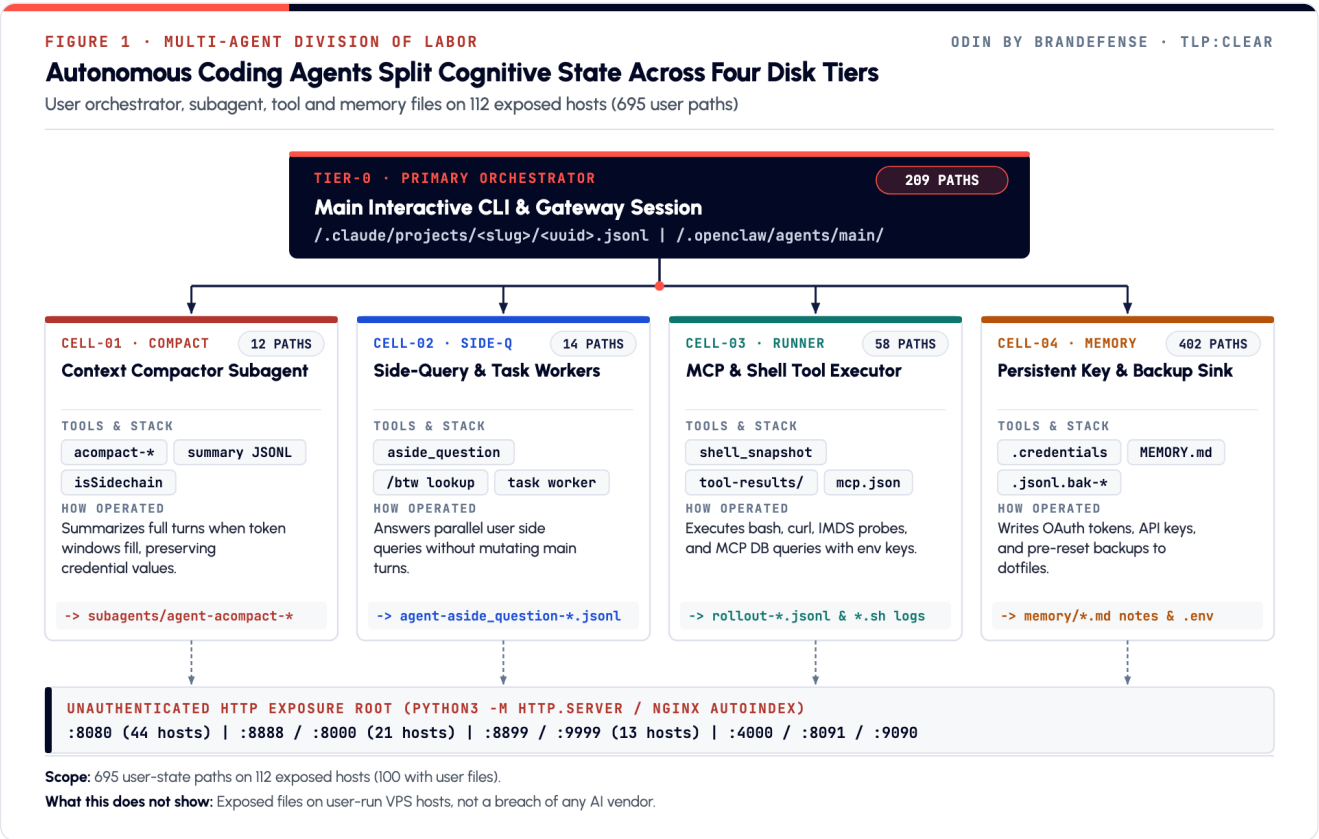


Figure 1: Autonomous coding agents split their working memory across four disk layers

Layer	Paths	What it holds	Example files
Primary session	209	Full conversation: prompts, pasted `.env` blocks, MCP parameters, raw tool output	`/.claude/projects/<slug>/<uuid>.jsonl`, `.trajectory.jsonl`, `rollout-*.jsonl`
Compaction and side-query subagents	26 (12 compaction, 14 side-query and task)	Summaries written when the context window fills, and parallel side questions	`subagents/agent-acompact-*.jsonl`, `agent-aside_question-*.jsonl`
Tool execution	58	Shell commands, metadata queries, MCP database calls and their output	`shell_snapshots/*.sh`, `tool-results/*.txt`
Persistent credential and backup stores	402 (218 secret stores, 172 backup or history snapshots, 12 memory notes)	OAuth tokens, API keys, pre-reset backups, Markdown notes	`.credentials.json`, `auth.json`, `auth-profiles.json`, `.jsonl.bak-*`

The compaction layer matters most for cleanup. When a session fills its context window, Claude Code starts an `agent-acompact-*` subagent. It writes a dense summary of the work so far to a new file. Keys, Tailscale tokens and command-and-control (C2) URLs in the earlier turns carry over into that summary. They stay on disk even after the user deletes the original script. Whether these credentials still worked was not tested.

Which ports the servers use

These servers rarely use port 443. They sit on development ports that match `python3 -m http.server`, `miniserve` or a container preview.

Port	Servers	Share of 112
8080	44	39.3%
8888	11	9.8%
8000	10	8.9%
8899	9	8.0%
80	5	4.5%
9999 (4), 8081 (4), 18080 (3), 4000 (2), 19999 (2)	15	
Other high ports, such as 3000, 3003, 8090, 8091, 9090, 9123	18	

CHAPTER 03

What Leaks: Secret Categories

IN SHORT

AI model keys and C2 or callback URLs make up nearly half of all matches. The most damaging finds are the infrastructure secrets that let someone move further: SSH keys, GitHub tokens, cloud tokens and mesh VPN keys.

INSIDE THIS CHAPTER

The categories 11

Which secrets matter most 12

03 What Leaks: Secret Categories

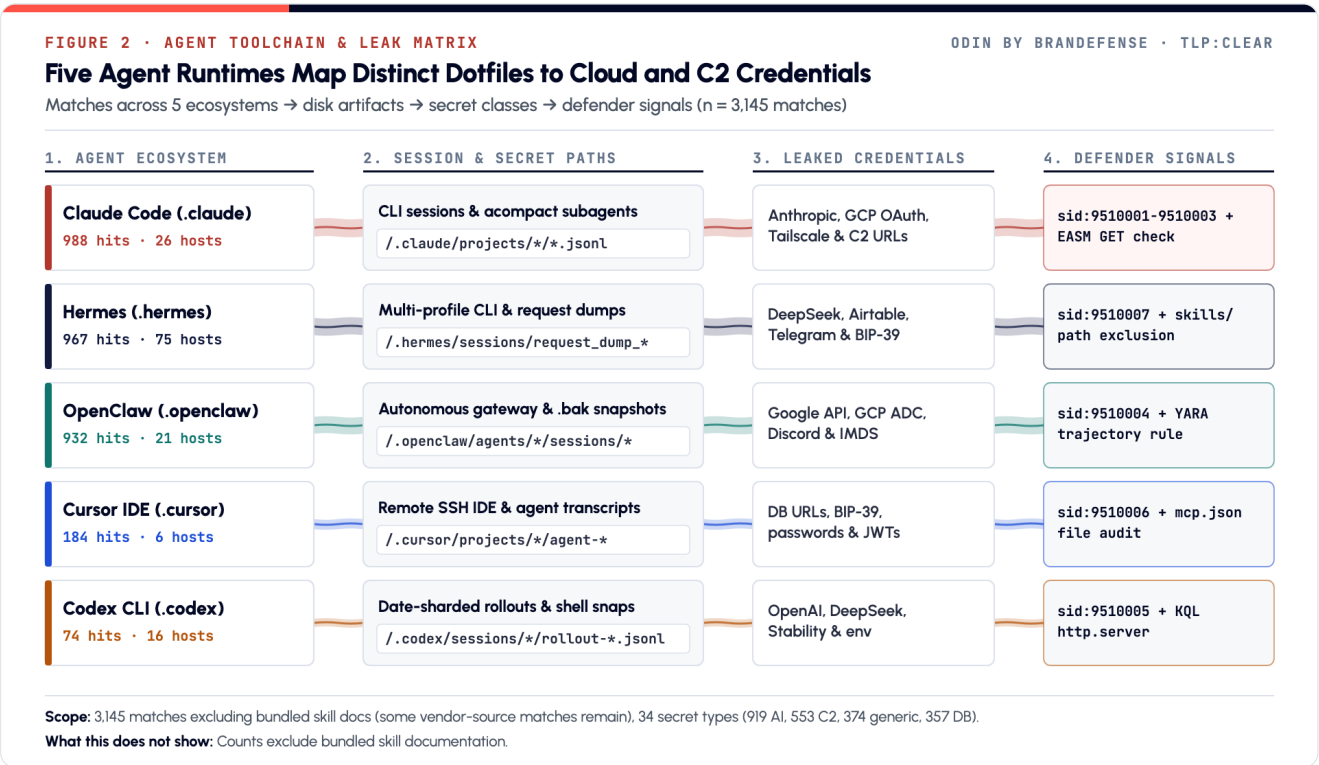


Figure 2: Five agent runtimes map distinct dotfiles to cloud and C2 credentials

The categories

The 3,145 matches cover 34 secret types, grouped below. Some vendor source matches are still inside these counts.

Category	Matches	Share	Leading types
AI model keys	919	29.2%	DeepSeek (425), Stability AI (204), Anthropic (110), OpenRouter (102), OpenAI (72)
C2 and callback URLs	553	17.6%	C2-pattern endpoints (550), reverse shell strings (3)
Generic passwords and secrets	374	11.9%	Passwords (195), generic secrets (179)
Database and SaaS	357	11.4%	Airtable keys (266), database connection strings (91)
Bot and drop channels	353	11.2%	Telegram bot tokens (285), Discord bot tokens (42), out-of-band callbacks (16), Pastebin drops (10)
Cloud and mesh access	251	8.0%	Google API keys (115), GCP OAuth secrets (67), DigitalOcean tokens (48), Vultr keys (14), GCP application default credentials (2)
Crypto and SSH keys	215	6.8%	Private keys (98), Tailscale keys (48), JWTs (45), BIP-39 seed phrases (23)
Source control	111	3.5%	GitHub tokens (90), GitHub personal access tokens (21)
Exchange and monitoring	12	0.4%	Binance keys (8), Sentry DSNs (4)

The "C2 and callback" label describes the URL pattern. It does not prove that each URL is attacker infrastructure. Many are webhooks or test endpoints the agent was asked to work with.

Which secrets matter most

AI provider keys and C2-pattern URLs account for nearly half of all matches. Damage, however, depends on what lets someone move further. These are the lateral secrets in the set:

Secret	Count	Note
PEM private keys	Up to 68	98 matches minus 30 that are regex patterns inside `redact.py` source code; not yet checked for other bundle rows
GitHub tokens	111	Source code and CI access
GCP OAuth secrets	67	Cloud project access
DigitalOcean tokens	48	Droplet and account control
Tailscale keys	48	Entry into a private mesh network
Telegram and Discord bot tokens	327	Control of the bots and their message channels

CHAPTER 04

Case Study: One Workspace on Two Servers

IN SHORT

Two European VPS servers expose the same Claude Code workspace. One block of credentials appears in the main sessions, six compaction summaries, two side-query logs, file history snapshots and a Markdown memory note.

INSIDE THIS CHAPTER

What the two servers share 14
The file layers on Host A and Host B 14
Why the counts repeat 14

04 Case Study: One Workspace on Two Servers

What the two servers share

Host A and Host B sit on European VPS networks. Host A shows 258 matches and Host B shows 254. Both expose the same Claude Code project folder name and the same session and subagent file names. That points to one `/home/<user>/.claude` workspace copied to two separate servers.

When the workspace was copied between the two servers is unknown.

The file layers on Host A and Host B

Layer	Path pattern under `/.claude/`	Present on	Role and secrets
OAuth store	`.credentials.json`	Host A	Claude OAuth access token, refresh token and scopes
Pre-edit file history	`file-history/<session>/<snapshot>@v2, @v3`	Both	Versioned copies of scripts before secrets were removed
Primary sessions	`projects/<slug>/<uuid>.jsonl` (three sessions; a fourth on Host B)	Both	Full transcripts with Anthropic, OpenAI and Telegram keys
Compaction subagents	`projects/<slug>/<uuid>/subagents/agent-acompact-*.jsonl` (six files)	Both	Summaries that concentrate GCP OAuth, DigitalOcean, GitHub and Tailscale keys
Side-query workers	`.../subagents/agent-aside_question-*.jsonl` (two files)	Both	Side questions saved to disk next to the parent session's tokens
Memory note	`projects/<slug>/memory/*.md`	Both	A plaintext Markdown sheet listing cloud and bot credentials

Why the counts repeat

On Host A, seven provider and bot token types each show exactly 28 matches. The host also shows 44 C2-pattern URLs and 15 password matches. The uniform count of 28 comes from copying, not from 28 different keys.

One multi-cloud credential block in the Markdown memory note was copied into the primary sessions. It was copied again into six compaction files, two side-query logs and two file-history versions. Each copy counts as a new match.

On Host B, the copied workspace adds one more session file. Its C2-pattern URLs rise from 44 to 67.

The lesson for responders: deleting the memory note does not remove the keys. They also live in every compaction summary, side query and history snapshot.

CHAPTER 05

Backups, Trajectories and Resets

IN SHORT

Autonomous agents record shell probes and cloud metadata output, and they write append-only backup files. Resetting a session leaves its old secrets in sibling backup files.

INSIDE THIS CHAPTER

OpenClaw: trajectories, metadata probes and a reverse shell	16
Backup sprawl after resets	16
Codex: auth files, rollouts and shell snapshots	17
Cursor and Hermes	17
Where the deepest reservoir sits	17

05 Backups, Trajectories and Resets

Many engineers assume that an agent `/reset` command or a config edit removes exposed keys from disk. The files on these servers show the opposite. Agent runtimes checkpoint and back up their state to survive crashes and updates.

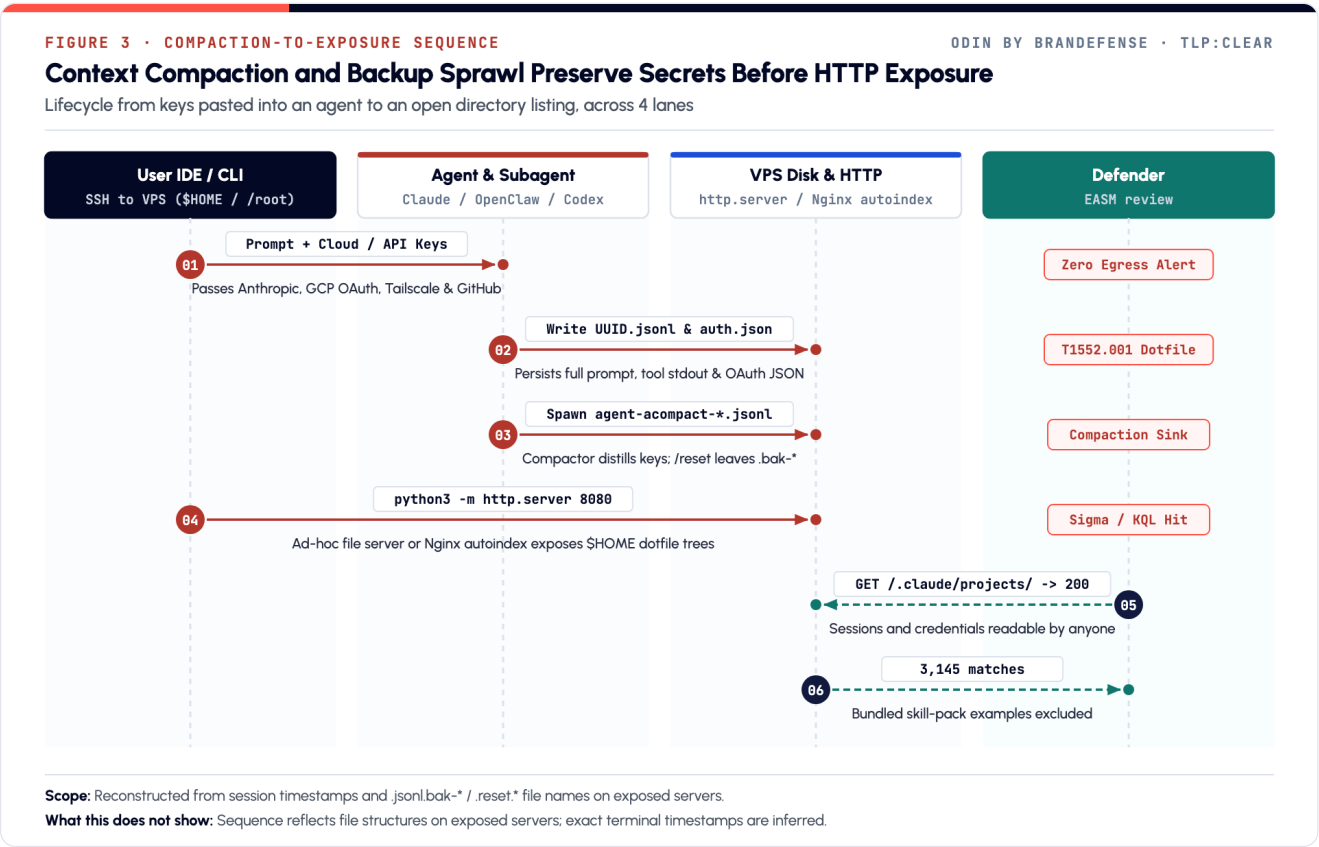


Figure 3: Context compaction and backup sprawl preserve secrets before HTTP exposure

OpenClaw: trajectories, metadata probes and a reverse shell

Host C sits on a German cloud VPS network. Its exposed tree holds 285 files and 183 matches. It was listed again on 18 November 2026.

One OpenClaw session file on Host C records a query to the cloud instance metadata service (IMDS) at `169.254.169.254`. It also records a listing of Chrome DevTools targets on the loopback address `127.0.0.1:9222`. The same file holds GCP application default credentials, a Discord bot token and an inline Python socket reverse shell. A sibling `.trajectory.jsonl` file holds a GCP OAuth client secret.

These strings show that the agent discussed, generated or ran these commands on its own server. They do not show any breach of a third-party target.

Backup sprawl after resets

On Host C and two other OpenClaw servers, resetting a session left two kinds of copies behind. One is a `.jsonl.bak-*` file, which on Host C still held a Google API key. The other is a `.jsonl.reset.*` file. The same folders kept `openclaw.json.bak` through `.bak.4`, a `.clobbered.*` copy and a hand-named pre-fix backup.

Another OpenClaw server, Host D on an Asia-Pacific cloud VPS network, shows 158 matches. Of these, 154 are C2-pattern URLs. It pairs seven `.trajectory.jsonl` files with a cron run log under `/.openclaw/cron/runs/`.

Codex: auth files, rollouts and shell snapshots

Across 16 Codex servers (74 matches), the Codex CLI stores secrets in several places:

File	What it holds
`/.codex/auth.json`	Bearer tokens
`/.codex/agentrouter.env`	Custom router keys
`/.codex/config.toml`	Provider keys in TOML, including a DeepSeek key on one server
`/.codex/sessions/<YYYY>/<MM>/<DD>/rollout-*.jsonl`	Date-sharded session logs
`/.codex/shell_snapshots/*.sh`	Saved shell state scripts

Cursor and Hermes

Across 6 Cursor servers (184 matches), remote SSH sessions expose agent transcripts under `/.cursor/projects/<slug>/agent-transcripts/`. Three servers show them. Those servers also expose editor buffers under `/.cursor-server/data/User/History/`.

Hermes users split their bots into named profiles. Each profile has its own `auth.json` or `.env` under `/.hermes/profiles/<profile>/`. Hermes also writes raw request dumps to `/.hermes/sessions/request_dump_*.json`. Two Hermes servers with this layout show 61 and 70 matches.

Where the deepest reservoir sits

The persistent credential and backup layer holds 402 of the 695 user paths. That breaks down into 218 secret stores, 172 backup or history snapshots and 12 memory notes. Tool execution adds 58 more. Together these are the files most often missed during cleanup.

CHAPTER 06

Co-Hosted Multi-Agent Servers

IN SHORT

Of the 112 servers, 26 (23.2%) run two or three AI agents in one home folder. On several of them, the same keys were copied across agents.

INSIDE THIS CHAPTER

How many agents per server 19

Which agents run together 19

Three ways keys spread across agents 20

06 Co-Hosted Multi-Agent Servers

How many agents per server

Most servers run one agent, but a large minority stack several together.

Agents per server	Servers	Share
One	86	76.8%
Two	20	17.9%
Three	6	5.4%

Which agents run together

Stack	Servers	Share of 112	Example server	Matches and notes
Hermes + OpenClaw	6	5.4%	Host C, German VPS	183 matches in 285 files; the same Anthropic and Discord keys in two agents, IMDS query, reverse shell string
Claude + Codex	5	4.5%	Host B, European VPS	254 matches; copied Claude subagent tree, Codex in the same home, 67 C2-pattern URLs
Claude + OpenClaw	3	2.7%	US VPS	70 matches; three custom OpenClaw personas and plugin catalogs
Claude + Hermes + OpenClaw	2	1.8%	Host E, Asia-Pacific VPS	156 matches; Claude memory note about OpenClaw bots, 88 C2-pattern URLs
Claude + Codex + OpenClaw	2	1.8%	Host G, Asia-Pacific VPS	97 matches; the same DeepSeek key in a Codex rollout and an OpenClaw script
Codex + Hermes	2	1.8%	European VPS	59 matches; `/.codex/auth.json` next to Hermes profile `.env` files
Claude + Hermes	2	1.8%	Asia-Pacific VPS	25 matches; Claude project history next to Hermes config backups
Claude + Codex + Cursor	1	0.9%	Host F, Chinese VPS	125 matches in 116 files; remote Cursor and VS Code IDE, Claude MCP and Codex CLI
Cursor + Hermes, Codex + Cursor, Claude + Codex + Hermes	3	2.7%	Three servers	8 to 17 matches each; remote Cursor SSH server next to CLI agents

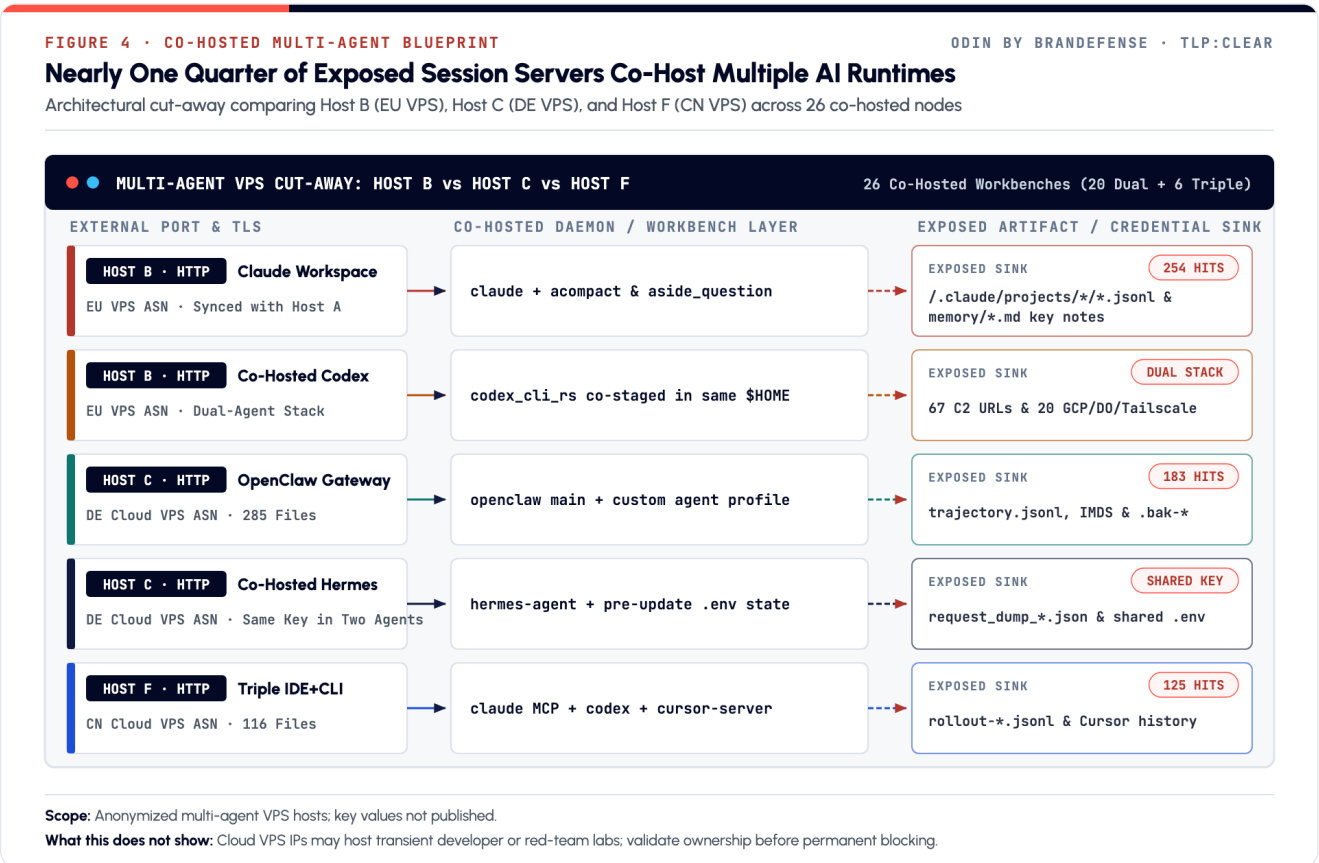


Figure 4: Nearly one quarter of exposed session servers co-host multiple AI runtimes

Three ways keys spread across agents

No key value or key hash is published.

Key copying between two agents (Host C).

OpenClaw stores provider keys for two agent profiles in `/.openclaw/agents/main/agent/auth-profiles.json`. Hermes keeps a pre-update `.env` copy under `/.hermes/state-snapshots/`. Both files hold the same Anthropic API key. An OpenClaw session file and that Hermes `.env` snapshot also hold the same Discord bot token.

One agent managing another (Host E and Host G).

Owners use Claude Code or Codex interactively to write and debug background OpenClaw or Hermes bots on the same server. On Host E, a Claude Code memory note about OpenClaw bots sits next to a pre-migration OpenClaw folder and 88 C2-pattern URLs. On Host G, a Codex rollout and an OpenClaw shell script contain the same DeepSeek key.

A full remote workstation (Host F).

Host F, on a Chinese cloud VPS network, combines remote Cursor and VS Code SSH servers with Claude Code MCP sessions and Codex CLI rollouts. Codex session files on the server carry dates up to 17 November 2026. The host exposes 86 Stability AI keys, 22 password records and 12 generic secrets.

For defenders, the rule is simple. When a key leaks from one agent folder, search every other agent folder on that host for the same key.

07

CHAPTER 07

Bundled Files and Who Owns These Servers

IN SHORT

Secret scans that do not exclude skill packs overstate Hermes leaks, because those packs ship example keys and local URLs. Most exposed servers belong to careless developers, not attackers.

INSIDE THIS CHAPTER

Bundled skill files 22

Who owns these servers 22

07 Bundled Files and Who Owns These Servers

Treating every match in a dotfile folder as a live leak floods analysts with false alerts. Treating every exposed server as hostile blames careless engineers. Both mistakes are easy to avoid.

Bundled skill files

Across all servers, 266 distinct secret values were found. Of these, 135 turned out to be example values from open-source skill packs and test fixtures. The other 131 are session secrets. The 131 include low-entropy password values, which this report does not publish in any form.

Host H shows the effect on one server. It runs Hermes on a Chinese cloud VPS and exposes 22 files. Its full listing on 18 November 2026 held 31 distinct secret values. Twenty of them (64.5%) were bundled examples under `/.hermes/skills/`:

Source	Values	What they are
ComfyUI skill reference	16	Local REST examples such as `http://127.0.0.1:8188/queue`, `/interrupt` and `/system_stats` in `/.hermes/skills/creative/comfyui/references/rest-api.md`
Other skill files	4	Placeholders in `SKILL.md`, `backends.md` and a skills backup archive
Hermes session log	11	One C2-pattern URL and ten password values in `/.hermes/sessions/`

Excluding `/.hermes/skills/` removes all 20 examples and keeps the 11 real session values. An earlier listing of Host H showed only one match, a private-key regex pattern inside `redact.py`.

Who owns these servers

Ownership cannot be confirmed from exposed files alone. We weighed three explanations across the 112 servers. They are working hypotheses, not attributions.

Hypothesis	Servers	What supports it	Response
H1: Developer or AI engineering server exposed by mistake	83 (74.1%)	Single-agent installs serving SaaS, trading or study projects through an ad-hoc `http.server`	Fix the exposure and rotate keys; do not block permanently
H2: Multi-agent builder or red-team lab	21 (18.8%)	Two or three IDE and CLI agents on one host (Host F, Host G and a US host), keys copied across tools, few C2-pattern URLs	Check corporate egress for accidental tunnel use; rotate matched keys
H3: Workbench with offensive content	8 (7.1%)	Dense C2-pattern URLs (Hosts A, B, D, E and two others) or recorded metadata probes and a reverse shell (Host C)	Hunt egress logs for contact; check how long the cloud IP has been leased before any block

Even in the H3 group, C2 and reverse shell strings in a session log show that an agent discussed, generated or ran those commands on the server. They do not confirm any breach of a third-party target. The owners of all 112 servers are treated as users, not operators.

CHAPTER 08

What Defenders Should Do

IN SHORT

This chapter lists the actions defenders should take, in priority order.

08 What Defenders Should Do

Within 24 hours

- [ ] Search your own external attack surface for HTTP 200 responses on `/.claude/`, `/.codex/`, `/.cursor/`, `/.openclaw/` and `/.hermes/`.
- [ ] Add a reverse-proxy deny rule for these dotfile paths in Nginx, Caddy, Apache and CloudFront. Confirm that a request returns 403 or 404.
- [ ] Block public ingress to development ports 3000, 3003, 4000, 8000, 8080, 8091, 8888, 8899, 9090 and 9999 on cloud VPS and jump hosts.

Within 72 hours

- [ ] Require `--bind 127.0.0.1` for `python3 -m http.server`, `npx serve` and similar tools on engineering hosts. Alert on any launch in `/root` or `/home/` that binds to all interfaces.
- [ ] When rotating a key, also delete `subagents/agent-acompact-*.jsonl`, `file-history/`, `.jsonl.bak-*`, `.reset.*` and `shell_snapshots/` in `~/.claude/`, `~/.openclaw/` and `~/.codex/`.
- [ ] Replace hard-coded `DATABASE_URL` values in `~/.cursor/mcp.json` and keys in agent memory notes with an OS keychain or short-lived vault injection.

Within 7 to 30 days

- [ ] Restrict cloud metadata access. Enforce IMDSv2 with hop limit 1 on AWS EC2, require the `Metadata-Flavor: Google` header and workload identity on GCP, and use the token-hardened metadata mode on Alibaba ECS. On DigitalOcean and other Linux VPS hosts, block `169.254.169.254` with a per-user firewall rule.
- [ ] Hash your own active Anthropic, OpenAI, GCP, Tailscale and bot keys, and check every agent folder on each engineering host for copies.
- [ ] Exclude `/.hermes/skills/`, `node_modules/` and test fixtures from secret scanning before opening incidents.
- [ ] Deploy the detection rules in Chapter 9 and confirm that all eight Suricata signatures load.

09

CHAPTER 09

Detection Rules

IN SHORT

This chapter describes each published rule and where to deploy it.

INSIDE THIS CHAPTER

How the Suricata signatures work 26

How the other rules work 26

How to sort an event 27

09 Detection Rules

The full rules are in Appendix B. They cover both sides of the problem: your own engineers serving dotfiles, and outsiders fetching them.

Rule file	What it catches
`odin_ai_agent_session_exposure.rules`	Suricata 7: eight signatures for HTTP requests to agent credential and session paths, and for open directory listings that expose agent folders
`sentinel_ai_agent_workspace_exposure_hunt.kql`	Microsoft Sentinel and Defender XDR: home-folder file servers on public interfaces, writes to agent secret files, agent-started metadata or reverse shell commands, and public connections to ad-hoc ports
`sigma_exposed_ai_coding_agent_http_serving.yml`	Sigma (3 rules): non-loopback file servers in home folders, web log hits on agent dotfiles, and agent runtimes that start metadata probes or socket shells
`odin_exposed_ai_agent_session_artifacts.yar`	YARA (4 rules): Claude credentials and compaction files, OpenClaw trajectories and auth profiles, Codex rollouts and auth files, Cursor MCP configs and Hermes session dumps

How the Suricata signatures work

SID	Looks for	ATT&CK
9510001	`GET` request for `/.claude/.credentials.json`	T1552.001
9510002	`GET` under `/.claude/projects/` for a `.jsonl` file, `memory/` or `tool-results/`	T1213
9510003	`GET` for `subagents/agent-acompact-*.jsonl`	T1552.001
9510004	`GET` under `/.openclaw/agents/` for `.jsonl`, `auth-profiles.json` or `models.json`	T1552.001
9510005	`GET` under `/.codex/sessions/` for a `rollout-*.jsonl` file	T1213
9510006	`GET` under `/.cursor/` for `mcp.json` or an agent transcript	T1552.001
9510007	`GET` under `/.hermes/` for request dumps, session files, state snapshots or profile `auth.json` and `.env`, excluding `/skills/`	T1552.001
9510008	HTTP 200 directory listing ("Index of /" or "Directory listing for /") with links to any of the five agent folders	T1213

Suppress authorized internal scanners with explicit `/32` entries, not broad subnet exceptions.

How the other rules work

The KQL hunt has a 14-day lookback and five branches. Two branches catch a file server started from a home folder without a loopback bind. One joins Linux `auditd` command and working-directory records; the other reads a served-directory argument from Defender process events.

The third branch flags agent or web processes writing compaction, trajectory, backup or auth files. The fourth flags an agent process running a metadata query, a socket import, a DevTools listing or `/dev/tcp/`. The fifth flags public inbound connections to Python, Node or `miniserve` on the ad-hoc ports listed in Chapter 2.

The query excludes approved build hosts through two placeholder names. Replace them with your own host names before running it.

The Sigma rules need Sysmon for Linux or `auditd` process events with the working directory, plus reverse-proxy access logs.

Each YARA rule is limited to files under 20 MB. The four rules matched all 12 synthetic positive samples and none of 4 negative samples in a test of their logic. Every rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma, Kusto.Language); the rules were not replayed against live traffic.

How to sort an event

What the event shows	Action
A file server bound only to `127.0.0.1` or `::1`	Ignore, local development
A path under `/.hermes/skills/`, `node_modules/`, a test file or `SKILL.md`	Compare with the upstream package; rotate any key that differs
A single-agent session or credential file exposed	Rotate keys and fix the exposure
Several agents co-hosted with shared keys	Rotate keys and audit every agent folder
Compaction files, metadata probes or reverse shell strings	Investigate as a priority
A low-signal workspace file only	Hold for review

10

CHAPTER 10

Indicators

IN SHORT

This chapter lists the published indicators and explains what stays withheld.

10 Indicators

This report publishes no indicators. All 112 servers are exposed developer or operator machines, and their owners are the victims of the leak. Publishing their addresses, ports, usernames, project names or file names would help others harvest the same credentials.

The secret values are withheld in every form, including their hashes. Defenders should hunt with the path patterns in Chapters 2 to 6 and the rules in Appendix B, and check their own keys against their own agent folders.

11

CHAPTER 11

Scope and Limits

> IN SHORT
>
> This chapter sets out what the evidence can show and what it cannot.

11 Scope and Limits

- **Window.** The findings come from a two-week observation window ending 18 November 2026. Some servers also have earlier directory listings; those listings are used only for counts and are not dated in this report.
- **Matches are not confirmed leaks.** The 3,145 matches still include some vendor bundle and source code matches. The validity of the exposed credentials was not tested.
- **Ownership is unknown.** The three hypotheses in Chapter 7 are a working model. Exposed files show server state, not who owns the server.
- **C2-pattern strings are not attacks.** URLs and shell strings in session logs show what an agent discussed or ran. They do not confirm a breach of any third party.
- **No AI vendor was breached.** These are files on user-run servers, exposed by the users' own file servers.
- **Rule testing.** Every published rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma and Microsoft's Kusto.Language analyzer, checked on 20 November 2026). The rules were not replayed against live traffic, so test and tune them in your own environment before production use.

12

CHAPTER 12

Sources

IN SHORT

This chapter lists the public references used in the report.

12 Sources

1. Anthropic. Claude Code documentation: session storage, memory and subagents. https://docs.anthropic.com/en/docs/claude-code/overview (verified 18 November 2026)
2. OpenAI. Codex CLI repository: configuration, authentication and rollout logs. https://github.com/openai/codex (verified 18 November 2026)
3. Cursor. Model Context Protocol documentation. https://cursor.com/docs/mcp (verified 18 November 2026)
4. Nous Research. Hermes Agent repository. https://github.com/NousResearch/hermes-agent (verified 18 November 2026)
5. OpenClaw Project. OpenClaw repository. https://github.com/openclaw/openclaw (verified 18 November 2026)
6. MITRE ATT&CK. T1552.001 Unsecured Credentials: Credentials In Files. https://attack.mitre.org/techniques/T1552/001/ (verified 18 November 2026)
7. MITRE ATT&CK. T1213 Data from Information Repositories. https://attack.mitre.org/techniques/T1213/ (verified 18 November 2026)
8. Python Software Foundation. `http.server` documentation and security warning. https://docs.python.org/3/library/http.server.html (verified 18 November 2026)
9. F5 Nginx. `ngx_http_autoindex_module` documentation. https://nginx.org/en/docs/http/ngx_http_autoindex_module.html (verified 18 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

No individual indicators are published for this study; Appendix A explains why. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (0) 35

B. Detection rules (4 files) 36

A Indicators of Compromise

No individual addresses or file hashes are published for this study. The hosts in this report are exposed or unattributed systems rather than confirmed adversary infrastructure, so they stay anonymous. Use the detection rules in Appendix B.

B Detection Rules

Each rule file loads in its own engine (Suricata 7, YARA-X, Kusto, pySigma). Tune thresholds and allow-lists to your environment before alerting or blocking. The same files can be downloaded from the online edition.

odin_ai_agent_session_exposure.rules

SURICATA

```
# Odin by Brandefense - Monograph #10 Suricata 7.x Exposed AI Agent Session & Subagent Suite
# Classification: TLP:CLEAR
# Scope Note: Detects unauthenticated HTTP GET exposure and HTTP 200 autoindex serving of
#             .claude, .openclaw, .codex, .cursor, and .hermes session/credential stores.

# 1. Claude Code OAuth and API bearer keyring exposure (/.claude/.credentials.json)
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed Claude Code Credentials JSON Access";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.claude/.credentials.json"; nocase;
fast_pattern; metadata:created_at 2026_11_18, mitre_technique_id T1552_001; classtype:credential-theft; sid:
9510001; rev:1;)

# 2. Claude Code primary orchestrator project transcript or memory exposure (/.claude/projects/)
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed Claude Code Project Session JSONL or Memory";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.claude/projects/"; nocase;
fast_pattern; pcre:"/(\.jsonl|\/memory\/|\/tool-results\/)/i"; metadata:created_at 2026_11_18, mitre_technique_id
T1213; classtype:policy-violation; sid:9510002; rev:1;)

# 3. Claude Code spawned context-compactor subagent exposure (subagents/agent-acompact-*.jsonl)
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed Claude Subagent Context Compactor JSONL";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"subagents/agent-acompact-"; nocase;
fast_pattern; content:".jsonl"; nocase; metadata:created_at 2026_11_18, mitre_technique_id T1552_001;
classtype:credential-theft; sid:9510003; rev:1;)

# 4. OpenClaw autonomous agent trajectory, backup snapshot, or auth-profiles exposure (/.openclaw/agents/)
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed OpenClaw Agent Trajectory or Auth Profile";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.openclaw/agents/"; nocase;
fast_pattern; pcre:"/(\.jsonl|auth-profiles\.json|models\.json)/i"; metadata:created_at 2026_11_18,
mitre_technique_id T1552_001; classtype:credential-theft; sid:9510004; rev:1;)

# 5. OpenAI Codex CLI rollout session transcript exposure (/.codex/sessions/.../rollout-*.jsonl or /.codex/
sessions/rollout-*)
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed OpenAI Codex CLI Rollout Session JSONL";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.codex/sessions/"; nocase;
fast_pattern; content:"rollout-"; nocase; content:".jsonl"; nocase; metadata:created_at 2026_11_18,
mitre_technique_id T1213; classtype:policy-violation; sid:9510005; rev:1;)

# 6. Cursor IDE Model Context Protocol config (/.cursor/mcp.json) or agent transcript JSONL exposure
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed Cursor MCP Config or Agent Transcript";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.cursor/"; nocase; fast_pattern;
pcre:"/(mcp\.json|agent-transcripts\/.*\.jsonl)/i"; metadata:created_at 2026_11_18, mitre_technique_id T1552_001;
classtype:credential-theft; sid:9510006; rev:1;)

# 7. Hermes Agent live session, request_dump, state-snapshot, or profile auth exposure
alert http any any -> any any (msg:"ODIN-AI-AGENT Exposed Hermes Session Dump or Profile Auth Store";
flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.hermes/"; nocase; fast_pattern;
pcre:"/(sessions\/(request_dump_|session_|\d{8}_)|state-snapshots\/|profiles\/[^\/]+\/(auth\.json|\.env))/i";
metadata:created_at 2026_11_18, mitre_technique_id T1552_001; classtype:credential-theft; sid:9510007; rev:3;)

# 8. HTTP 200 Autoindex open-directory listing exposing dotfile AI agent state roots (Nginx/Apache & Python
http.server)
alert http any any -> any any (msg:"ODIN-AI-AGENT HTTP 200 Open Directory Listing Exposing AI Agent Dotfiles";
flow:established,from_server; http.stat_code; content:"200"; file.data; content:"href="; fast_pattern; pcre:"/
(Index of \/|Directory listing for \/)/i"; pcre:"/href=[\x22\x27]\/?(\.claude|\.openclaw|\.codex|\.cursor|
\.hermes)\//i"; metadata:created_at 2026_11_18, mitre_technique_id T1213; classtype:policy-violation; sid:
9510008; rev:2;)
```

odin_exposed_ai_agent_session_artifacts.yar

YARA

```
/*
   Odin by Brandefense - Monograph #10 YARA Suite for Exposed AI Agent Session & Credential Artifacts
   Classification: TLP:CLEAR
   Scope: 4 deterministic structural rules identifying exposed Claude Code, OpenClaw, OpenAI Codex CLI,
          Cursor IDE, and Hermes Agent credential stores and subagent/trajectory JSONL files while
          avoiding false positives on bundled OSS skill documentation and standard web configs.
*/

rule ODIN_Claude_Credentials_And_Subagent_Compaction
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects exposed Claude Code .credentials.json OAuth stores and acompact/aside_question
subagent JSONL transcripts"
        mitre_attack = "T1552.001"
    strings:
        $oauth1 = "\"claudeAiOauth\"" ascii
        $oauth2 = "\"accessToken\"" ascii
        $oauth3 = "\"refreshToken\"" ascii
        $sub_parent = "\"parentUuid\"" ascii
        $sub_side = "\"isSidechain\":true" ascii
        $sub_acompact = "acompact-" ascii
        $sub_aside = "aside_question-" ascii
    condition:
        filesize < 20MB and (
            ($oauth1 and $oauth2 and $oauth3) or
            ($sub_parent and $sub_side and ($sub_acompact or $sub_aside))
        )
}

rule ODIN_OpenClaw_Trajectory_AuthProfiles_And_DeviceKey
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects exposed OpenClaw auth-profiles.json, device.json private identity keys,
and .trajectory.jsonl logs"
        mitre_attack = "T1552.001"
    strings:
        $ap1 = "\"version\":" ascii
        $ap2 = "\"profiles\":" ascii
        $ap3 = "\"provider\":" ascii
        $ap4 = "\"apiKey\"" ascii
        $ap5 = "\"access\"" ascii
        $ap6 = "\"token\"" ascii
        $ap7 = "\"key\"" ascii
        $tr1 = "\"trajectoryVersion\":\"openclaw." ascii
        $tr2 = "\"toolCall\"" ascii
        $dv_id = "\"deviceId\"" ascii
        $dv_ts = "\"createdAtMs\"" ascii
        $dv_pub = "\"publicKey\"" ascii
        $dv_prv = "\"privateKey\"" ascii
        $dv_prvpem = "\"privateKeyPem\"" ascii
        $dv_pem = "-----BEGIN PRIVATE KEY-----" ascii
    condition:
        filesize < 20MB and (
            ($ap1 and $ap2 and $ap3 and ($ap4 or $ap5 or $ap6 or $ap7)) or
            ($tr1 and $tr2) or
            ($dv_id and $dv_ts and (($dv_pub and $dv_prv) or ($dv_prvpem and $dv_pem)))
        )
}

rule ODIN_Codex_Rollout_Auth_And_ShellSnapshot
{
    meta:
```

```
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects exposed OpenAI Codex CLI rollout-*.jsonl session logs and auth.json token stores"
        mitre_attack = "T1552.001"
    strings:
        $ro1 = "\"session_meta\"" ascii
        $ro2 = "\"originator\":\"codex_cli" ascii
        $ro3 = "\"response_item\"" ascii
        $au1 = "\"OPENAI_API_KEY\"" ascii
        $au2 = "\"tokens\"" ascii
        $au3 = "\"access_token\"" ascii
        $au4 = "\"refresh_token\"" ascii
        $au5 = "\"id_token\"" ascii
        $au6 = /"OPENAI_API_KEY"[ \t]*:[ \t]*"sk-/ ascii
    condition:
        filesize < 20MB and (
            ($ro1 and $ro2 and $ro3) or
            ($au6) or
            ($au2 and $au3 and $au4) or
            ($au1 and $au2 and ($au3 or $au4 or $au5))
        )
}

rule ODIN_Cursor_MCP_Transcript_And_Hermes_SessionDump
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects exposed Cursor mcp.json with embedded secrets, Cursor agent-transcripts JSONL, and
Hermes request_dump or auth.json credential stores (version/providers/credential_pool layout)"
        mitre_attack = "T1552.001"
    strings:
        $mcp1 = "\"mcpServers\"" ascii
        $mcp2 = "\"env\"" ascii
        $mcp_sec1 = "DATABASE_URL" ascii
        $mcp_sec2 = "API_KEY" ascii
        $mcp_sec3 = "GITHUB_PERSONAL_ACCESS_TOKEN" ascii
        $cur1 = "\"cursorAgentTranscript\":true" ascii
        $cur2 = "\"composerId\"" ascii
        $her_d1 = "\"hermes_agent_version\"" ascii
        $her_d2 = "\"request_dump_id\"" ascii
        $her_r1 = "\"session_id\"" ascii
        $her_r2 = "\"reason\"" ascii
        $her_r3 = "\"request\"" ascii
        $her_r4 = "\"messages\"" ascii
        $her_p1 = "\"version\"" ascii
        $her_p2 = "\"providers\"" ascii
        $her_p3 = "\"active_provider\"" ascii
        $her_p4 = "\"credential_pool\"" ascii
        $her_p5 = "\"access_token\"" ascii
        $her_p6 = "\"refresh_token\"" ascii
        $her_p7 = "\"agent_key\"" ascii
        $her_p8 = "\"api_key\"" ascii
    condition:
        filesize < 20MB and (
            ($mcp1 and $mcp2 and ($mcp_sec1 or $mcp_sec2 or $mcp_sec3)) or
            ($cur1 and $cur2) or
            ($her_d1 and $her_d2) or
            ($her_r1 and $her_r2 and $her_r3 and $her_r4) or
            ($her_p1 and $her_p2 and ($her_p3 or $her_p4) and ($her_p5 or $her_p6 or $her_p7 or $her_p8))
        )
}
```

sentinel_ai_agent_workspace_exposure_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph #10 Microsoft Sentinel & Defender XDR Hunting Query (TLP:CLEAR)
let Lookback = 14d;
let ApprovedInternalBuildHosts = dynamic(["replace-with-your-build-host-01.example", "replace-with-your-build-
host-02.example"]); // placeholders: list your own approved build hosts
let ExposedSessionPorts = dynamic([3000, 3003, 4000, 8000, 8080, 8081, 8090, 8091, 8888, 8899, 9090, 9999, 18080,
19999]);
let WatchlistSessionHostIPs = dynamic([]);
let AgentRuntimeAndWebProcesses = dynamic(["claude", "codex", "openclaw", "hermes", "node", "python", "python3",
"nginx", "caddy", "httpd", "apache2", "miniserve", "php"]);
let SensitiveAgentPathRegex = @"(?i)(/\.claude/\.credentials\.json|subagents/agent-acompact-|subagents/agent-
aside_question-|/memory/reference_api_keys\.md|\.trajectory\.jsonl|\.jsonl\.bak-|\.jsonl\.reset\.|auth-
profiles\.json|/identity/device\.json|/\.codex/sessions/.*/rollout-|/\.codex/auth\.json|/\.cursor/mcp\.json|/
agent-transcripts/|/\.hermes/sessions/(request_dump_|\d{8}_)|/\.hermes/state-snapshots/|/\.hermes/profiles/[^/]+/
(auth\.json|\.env))";
let AuditdHttpExecve =
    Syslog
    | where TimeGenerated >= ago(Lookback)
    | extend Timestamp = TimeGenerated, DeviceName = Computer
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where Facility in ("audit", "authpriv", "daemon", "user")
    | where SyslogMessage contains "EXECVE" or SyslogMessage contains "cmd=" or SyslogMessage contains
"http.server"
    | extend AuditEventId = extract(@"msg=audit\(([0-9]+(?:\.[0-9]+)?:[0-9]+)\)", 1, SyslogMessage),
             NormalizedCmd = replace_string(replace_regex(SyslogMessage, @"a\d+=""", " "), @"""", "")
    | where isnotempty(AuditEventId)
    | where NormalizedCmd matches regex @"(?i)(-m\s+http\.server|SimpleHTTPServer|http-server|npx\s+serve|
\bminiserve\b|file-server|-S\s+0\.0\.0\.0:)"
    | project ExecTimestamp = Timestamp, DeviceName, AuditEventId, NormalizedCmd;
let AuditdHomeCwd =
    Syslog
    | where TimeGenerated >= ago(Lookback)
    | extend DeviceName = Computer
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where Facility in ("audit", "authpriv", "daemon", "user")
    | where SyslogMessage contains "type=CWD" or SyslogMessage contains "cwd="
    | extend AuditEventId = extract(@"msg=audit\(([0-9]+(?:\.[0-9]+)?:[0-9]+)\)", 1, SyslogMessage),
             WorkingDirectory = extract(@"(?i)\bcwd=""?(/root[^\s""]*|/home/[^\s""]+)", 1, SyslogMessage)
    | where isnotempty(AuditEventId) and isnotempty(WorkingDirectory)
    | project DeviceName, AuditEventId, WorkingDirectory;
union isfuzzy=true
(
    AuditdHttpExecve
    | join kind=inner (AuditdHomeCwd) on DeviceName, AuditEventId
    | extend Timestamp = ExecTimestamp,
             HuntSignal = "Auditd_Joined_EXECVE_CWD_NonLoopback_HTTP_Server_In_Home",
             InitiatingProcessFileName = "auditd_execve_cwd",
             TargetItem = WorkingDirectory,
             ArtifactDetail = strcat("audit(", AuditEventId, ") cwd=", WorkingDirectory, " :: ", NormalizedCmd)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem, ArtifactDetail
),
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where FileName in~ ("python", "python3", "node", "npx", "miniserve", "caddy", "php")
    | where ProcessCommandLine matches regex @"(?i)(-m\s+http\.server|SimpleHTTPServer|http-server|npx\s+serve|
\bminiserve\b|file-server|-S\s+0\.0\.0\.0:)"
    | extend ServedDirArg = extract(@"(?i)(?:--directory|-d|--root|-t|\bminiserve|\bserve|\bhttp-server)\s+
(['""]?(?:/root|/home/|[^\s'""]*/\.(?:claude|openclaw|codex|cursor|hermes))[^\s'""]*)", 1, ProcessCommandLine)
    | where isnotempty(ServedDirArg)
    | extend HuntSignal = "CLI_Explicit_Home_Or_Dotfile_Path_NonLoopback_HTTP_Server",
             TargetItem = trim(@"'""", ServedDirArg),
             ArtifactDetail = strcat("cmd=", ProcessCommandLine)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName = FileName, TargetItem, ArtifactDetail
),
(
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
```

```
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | where InitiatingProcessFileName in~ (AgentRuntimeAndWebProcesses)
    | extend FullArtifactPath = strcat(FolderPath, "/", FileName)
    | where FullArtifactPath matches regex @"(?i)/\.(claude|openclaw|codex|cursor|hermes)/"
    | where FullArtifactPath matches regex SensitiveAgentPathRegex
    | extend HuntSignal = "AI_Agent_Session_Or_Compaction_Secret_Sink_Write",
             ArtifactDetail = strcat(ActionType, " :: ", FullArtifactPath)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = FileName, ArtifactDetail
),
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where InitiatingProcessFileName in~ ("claude", "codex", "openclaw", "hermes", "node", "python", "python3")
        or InitiatingProcessCommandLine matches regex @"(?i)\b(claude|openclaw|codex|hermes-agent)\b"
    | where ProcessCommandLine contains "169.254.169.254/metadata"
        or ProcessCommandLine contains "import socket"
        or ProcessCommandLine contains "127.0.0.1:9222/json/list"
        or ProcessCommandLine contains "/dev/tcp/"
    | extend HuntSignal = "AI_Agent_Runtime_IMDS_Or_ReverseShell_Execution",
             ArtifactDetail = ProcessCommandLine
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = FileName, ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback) and ActionType in ("ConnectionSuccess", "InboundConnectionAccepted")
    | where (array_length(WatchlistSessionHostIPs) > 0 and RemoteIP in (WatchlistSessionHostIPs))
        or (InitiatingProcessFileName in~ ("python", "python3", "miniserve", "node", "caddy", "php") and
LocalPort in (ExposedSessionPorts) and RemoteIPType == "Public")
    | extend HuntSignal = iff(RemoteIP in (WatchlistSessionHostIPs), "Egress_To_Watchlist_AI_Session_Host",
"Public_Connection_On_AdHoc_Agent_HTTP_Port"),
             ArtifactDetail = strcat(RemoteIP, ":", tostring(RemotePort), " (local:", tostring(LocalPort), ")")
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = RemoteIP, ArtifactDetail
)
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
    EventCount = count(),
    SampleArtifacts = make_set(ArtifactDetail, 5)
    by DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem
| where EventCount >= 1
| order by EventCount desc
```

sigma_exposed_ai_coding_agent_http_serving.yml

SIGMA

```
title: Ad-Hoc Non-Loopback HTTP File Server Launched in Linux Home or AI Agent Root
id: 6c8f3b21-4d92-4a1e-9b75-10a100000001
status: experimental
description: Detects ad-hoc static HTTP file servers (python http.server, npx serve, http-server, miniserve, php
-S) started in /root or /home without explicit loopback binding, which exposes
hidden .claude, .openclaw, .codex, .cursor, and .hermes session directories. Requires a Linux process_creation
source that populates CurrentDirectory alongside CommandLine (such as Sysmon for Linux EventID 1 or Linux auditd
EXECVE and CWD records correlated by shared audit event ID via Laurel/Auditbeat), or an explicit home/dotfile
directory argument on CommandLine.
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://docs.python.org/3/library/http.server.html
  - https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/pdf/security_hardening/auditing-the-
system_security-hardening
  - https://attack.mitre.org/techniques/T1552/001/
tags:
  - attack.credential_access
  - attack.t1552.001
  - attack.collection
  - attack.t1213
logsource:
  category: process_creation
```

```
    product: linux
  detection:
    selection_cli:
      CommandLine|contains:
        - ' -m http.server'
        - 'SimpleHTTPServer'
        - 'http-server'
        - 'npx serve'
        - 'miniserve'
        - ' -S 0.0.0.0:'
    selection_workdir:
      CurrentDirectory|startswith:
        - '/root'
        - '/home/'
    selection_explicit_dir_arg:
      CommandLine|contains:
        - ' /root'
        - ' /home/'
        - '/.claude'
        - '/.openclaw'
        - '/.codex'
        - '/.cursor'
        - '/.hermes'
    condition: selection_cli and (selection_workdir or selection_explicit_dir_arg)
  falsepositives:
    - Intentional air-gapped lab file transfers restricted by host firewall rules (enforce explicit --bind
  127.0.0.1).
  level: high
  ---
  title: Unauthenticated Web Server Access to Exposed AI Agent Session and Credential Dotfiles
  id: 6c8f3b21-4d92-4a1e-9b75-10a100000002
  status: experimental
  description: Detects HTTP 200 access to exposed Claude Code, OpenClaw, OpenAI Codex, Cursor IDE, and Hermes Agent
  session transcripts, subagent compaction logs, and credential stores while excluding bundled Hermes OSS skill
  documentation.
  author: Odin by Brandefense
  date: 2026/11/18
  references:
    - https://docs.anthropic.com/en/docs/claude-code/overview
    - https://attack.mitre.org/techniques/T1552/001/
  tags:
    - attack.credential_access
    - attack.t1552.001
    - attack.collection
    - attack.t1213
  logsource:
    category: webserver
  detection:
    selection_status:
      sc-status: 200
    selection_uris:
      cs-uri-stem|contains:
        - '/.claude/.credentials.json'
        - '/.claude/projects/'
        - 'subagents/agent-acompact-'
        - 'subagents/agent-aside_question-'
        - '/.openclaw/agents/'
        - '/.openclaw/identity/device.json'
        - '/.codex/sessions/'
        - '/.codex/auth.json'
        - '/.cursor/mcp.json'
        - '/agent-transcripts/'
        - '/.hermes/sessions/'
        - '/.hermes/state-snapshots/'
        - '/.hermes/profiles/'
    condition: selection_status and selection_uris
  falsepositives:
    - Authorized forensic artifact mirrors gated behind reverse-proxy mTLS authentication.
  level: critical
  ---
  title: Autonomous AI Agent Runtime Spawning Cloud IMDS Probe or Python Socket Reverse Shell
  id: 6c8f3b21-4d92-4a1e-9b75-10a100000003
```

```
status: experimental
description: Detects AI coding or gateway runtimes (OpenClaw, Claude Code, Codex CLI, Hermes) spawning inline
Python socket reverse shells, headless Chrome CDP port 9222 enumeration, or cloud IMDS credential queries
observed in exposed trajectories.
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://attack.mitre.org/techniques/T1059/006/
  - https://attack.mitre.org/techniques/T1552/005/
tags:
  - attack.execution
  - attack.t1059.006
  - attack.credential_access
  - attack.t1552.005
logsource:
  category: process_creation
  product: linux
detection:
  selection_parent:
    ParentCommandLine|contains:
      - 'openclaw'
      - 'claude'
      - 'codex'
      - 'hermes'
  selection_payload:
    CommandLine|contains:
      - '169.254.169.254/metadata'
      - 'import socket'
      - '127.0.0.1:9222/json/list'
      - '/dev/tcp/'
  condition: selection_parent and selection_payload
falsepositives:
  - Authorized cloud security posture validation run inside isolated sandbox containers.
level: high
```

ODIN

About Odin Research

Odin Research is the public research series of Odin by Brandefense. It documents adversary infrastructure seen on the internet and turns each finding into indicators and detection rules that defenders can use. Victims and third parties stay anonymous, addresses appear only for confirmed adversary infrastructure, and no working attack code is published.

Odin is the agentic analyst inside the Brandefense threat intelligence platform, which unifies Cyber Threat Intelligence, Digital Risk Protection and External Attack Surface Management.

Online edition, indicators and rules · brandefense.io/odin-research/reports/when-ai-agents-leak-their-own-memory/
Short Read · brandefense.io/odin-research/blog/ai-coding-agents-are-leaking-keys-through-open-file-servers/
Defense Kit · brandefense.io/odin-research/defense-kit/