ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 16 / 30 · AI AGENT EXPOSURE

Unattended AI Agents Exposed
Hermes Kanban Boards, Cron Jobs and Leaked Keys

How 87 open Hermes Agent servers leak task boards, schedules, retry dumps and co-hosted TLS keys

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

87
Distinct exposed hosts behind the 1,100 directory entries analysed, holding 235,644 indexed files

14,914
Indexed files on one German server that exposes all eight subsystems

277
Certificate files exposed next to 7,643 Hermes files on one US server, including two TLS private keys

68 to 42
Skills the agent itself merged and archived in one 591-second Curator run

What Is in This Report. 11 chapters and two appendices.

How 87 open Hermes Agent servers leak task boards, schedules, retry dumps and co-hosted TLS keys

01	Executive Summary	04
02	What a Hermes Daemon Leaves on Disk	07
03	Task Boards, Schedules and 31 Tools	11
04	Curator Runs, Retry Dumps and Snapshots	14
05	Shared Servers: Agent State Next to TLS Keys	17
06	Separating Real Daemons From Skill Mirrors	20
07	What Defenders Should Do	24
08	Detection Rules	26
09	Indicators	29
10	Scope and Limits	31
11	Sources	33
A	Indicators of compromise (0)	35
B	Detection rules (4 files)	36

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense reviewed 87 servers exposing Hermes Agent folders over open HTTP. On 47 of them (54.0%), the files show unattended automation: schedules, task boards, skill curators and upgrade snapshots. Retry dumps keep full API requests with bearer headers. One US server also exposed TLS private keys for two web panels. Bind file servers on agent hosts to loopback, keep certificate keys out of home folders, and purge dumps and snapshots when you rotate keys.

01

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense reviewed 87 servers exposing Hermes Agent folders over open HTTP. On 47 of them (54.0%), the files show unattended automation: schedules, task boards, skill curators and upgrade snapshots. Retry dumps keep full API requests with bearer headers. One US server also exposed TLS private keys for two web panels. Bind file servers on agent hosts to loopback, keep certificate keys out of home folders, and purge dumps and snapshots when you rotate keys.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 04

01 Executive Summary

Hermes Agent is an open-source AI agent from Nous Research. It keeps its state in a hidden `~/.hermes` folder. Unlike a chat assistant, it can run as a background service. It schedules recurring jobs, splits projects into task cards for worker agents, and tidies its own skill library without a human prompt. This report looks at what that folder reveals when a server shares it over plain HTTP.

All 87 servers are exposed systems, not attacker infrastructure. Their owners are users who left a directory listing open. Every address, domain, path name and secret value in this report is withheld.

1. **Over half of exposed Hermes servers run unattended automation, not chats.**
   On 47 of 87 hosts (54.0%), the exposed files belong to background subsystems. Skill Curators appear on 41 hosts, code language servers on 7, upgrade snapshots on 6, schedulers on 5 and task boards on 4. One German server exposes all eight subsystems in 14,914 files.

2. **Retry dumps and upgrade snapshots keep old credentials after rotation.**
   When the model provider returns a temporary error, Hermes writes the whole failed request to disk. That file holds the bearer header, the agent's persona and memory, and 31 tool definitions. Upgrade snapshots and `.env` backups keep older keys next to the live ones.

3. **One open file server exposed agent state and web-panel TLS keys together.**
   On a US virtual server, a Python file server started in `/root` exposed 7,643 Hermes files and 2,222 scheduler files. The same listing exposed 277 `acme.sh` certificate files, including ECDSA private keys for two co-hosted web-panel domains.

Three Decisions for Leaders

1. **Treat unattended agents as privileged service accounts.** Any agent that runs scheduled jobs, task queues or email skills needs registration, an outbound allowlist and short-lived, scoped tokens.
2. **Ban open file servers on hosts that hold agent state or keys.** Ad-hoc Python or Node listeners in `/root` or `/home` expose every hidden folder beside them, including snapshots, dumps and certificate keys.
3. **Triage by what is exposed, not by the folder name.** A host that shows only bundled skill documents is noise. A host that shows task databases, schedules, retry dumps or private keys needs action.

The Study in Numbers

Number	What it is
87	Distinct exposed hosts behind the 1,100 directory entries analysed, holding 235,644 indexed files
47 (54.0%)	Hosts exposing background subsystems: schedulers, task boards, Curators, language servers or snapshots
14,914	Indexed files on one German server that exposes all eight subsystems
277	Certificate files exposed next to 7,643 Hermes files on one US server, including two TLS private keys
68 to 42	Skills the agent itself merged and archived in one 591-second Curator run
11	Verified secret findings: private keys, bot tokens, API keys and endpoint strings

How to use this report.

Security leaders can read Chapter 1 and Chapter 7. Platform and cloud teams should read Chapters 2 to 5. SOC and detection engineers should start with Chapters 6, 8 and 9.

02

CHAPTER 02

What a Hermes Daemon Leaves on Disk

IN SHORT

A Hermes install stores scheduler jobs, task boards, curator runs, logs and snapshots under one folder. When a high-port HTTP listener serves that folder, all of it becomes readable.

INSIDE THIS CHAPTER

From chat transcripts to background services 07
Nine path families 07
Where the hosts are and how they listen 08
How the subsystems fit together 08
The fullest example: one server, eight subsystems 09

02 What a Hermes Daemon Leaves on Disk

From chat transcripts to background services

An earlier Odin report looked at exposed `.claude`, `.codex` and `.cursor` folders. Those mostly held interactive coding sessions. The `.hermes` folders in this study look different. They belong to an agent built to keep working when nobody is typing.

Odin by Brandefense analysed 1,100 open `.hermes` directory entries. They sit on 87 distinct IPv4 hosts, which together list 235,644 indexed files.

Nine path families

Of the 1,100 entries, 859 (78.1%) map to a named subsystem. The other 241 (21.9%) are root credential files (`.env`, `auth.json`, `config.yaml`), named bot profiles, runtime routers and paste buffers.

Subsystem	Path under `~/.hermes`	Entries	Share of 1,100	Hosts (of 87)	What it does
Skills library	`skills/`, `skills/.archive/`	499	45.4%	68 (78.2%)	Playbooks such as `kanban-worker`, `cron-pipelines` and `himalaya`, plus bundled docs
Daemon logs	`logs/`	190	17.3%	57 (65.5%)	`agent.log`, `gateway.log` and `tui_gateway_crash.log`
Memory Curator	`logs/curator/`	124	11.3%	41 (47.1%)	Unattended skill merging and pruning (`run.json`, `REPORT.md`)
Kanban boards	`kanban/`, `kanban.db`	14	1.3%	4 (4.6%)	SQLite task queues, project boards and `t_<id>` worker folders
State snapshots	`state-snapshots/`	11	1.0%	6 (6.9%)	`*-pre-update/` copies of `.env` and `config.yaml`
Language server	`lsp/`	8	0.7%	7 (8.0%)	Headless Language Server Protocol runtime for code navigation
Durable memory	`memories/`, `SOUL.md`	6	0.5%	2 (2.3%)	`MEMORY.md` and `USER.md`, injected into every prompt
Cron scheduler	`cron/`	4	0.4%	5 (5.7%)	`jobs.json`, `executions.db`, `.tick.lock` and `cron/output/<job_id>/`
Session dumps	`sessions/`	3	0.3%	4 (4.6%)	`request_dump_*.json` retry payloads and `.jsonl` transcripts

Top-level counts understate how deep some folders go. This likely explains why the Cron and session rows show more hosts than entries. On one US server, `cron/` alone holds 2,222 files, and `sessions/` holds dozens of 87 to 256 KB request dumps.

Where the hosts are and how they listen

Half of the hosts are in China. The rest spread across ten other countries.

Country	Hosts	Share
China	44	50.6%
United States	14	16.1%
Germany	8	9.2%
Singapore	8	9.2%
Indonesia	4	4.6%
France, Japan, Canada	2 each	
South Korea, Malaysia, Netherlands	1 each	

Almost every exposure comes from a plain HTTP server on a non-standard port, bound to all interfaces.

Port	Hosts	Share
8080	38	43.7%
8000	8	9.2%
8899	8	9.2%
8888	7	8.0%
9090	4	4.6%
8765, 9999, 18080, 80	3 each	3.4% each

How the subsystems fit together

The Hermes TUI gateway (`tui_gateway/entry.py`) and a policy router (`runtime-controller/runtime_router.py`) sit on top. Below them, four units work on their own: the Cron scheduler, the Kanban board manager, the skill and language-server executor, and the Curator. All four write into the same plaintext credential and snapshot files on disk.

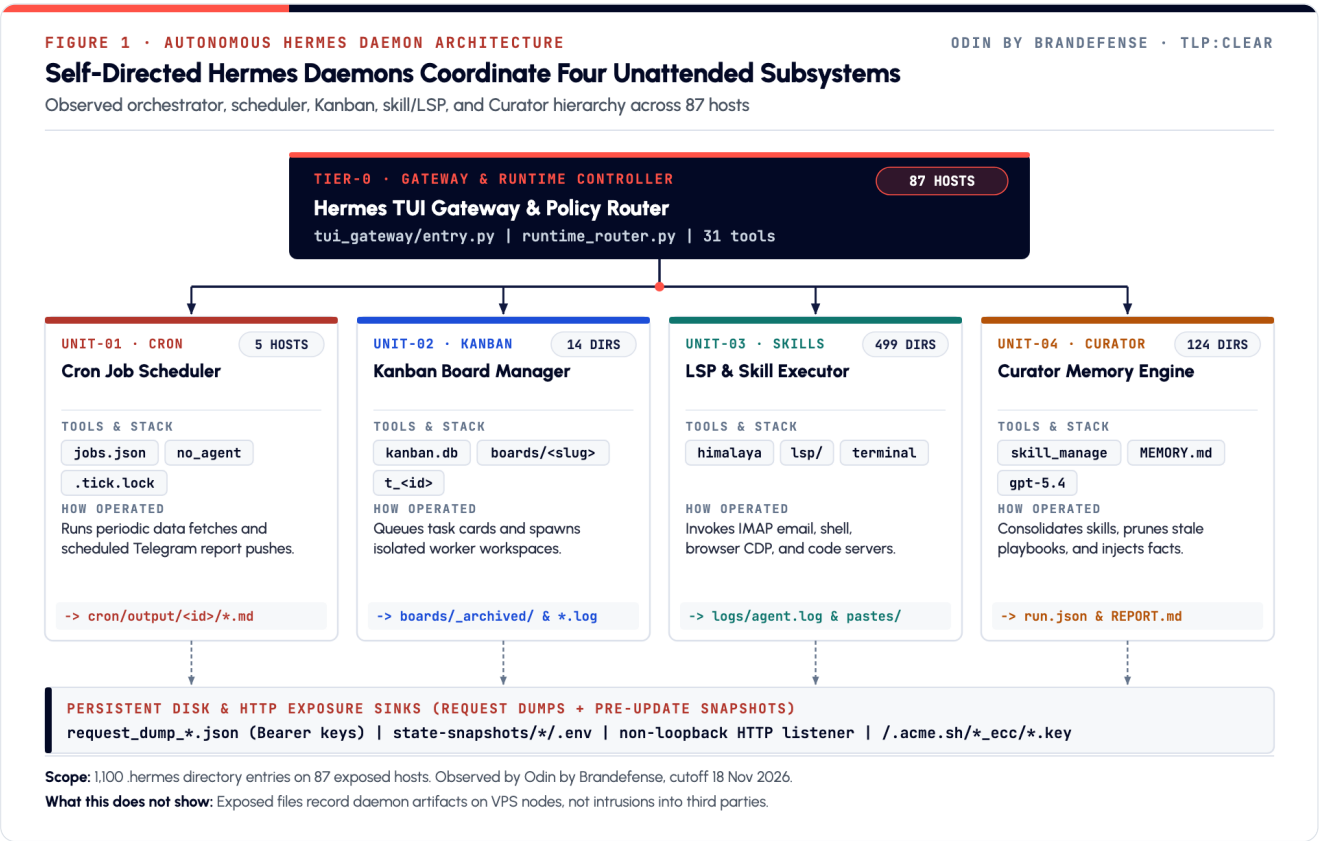


Figure 1: Self-directed Hermes daemons coordinate four unattended subsystems

The fullest example: one server, eight subsystems

A virtual server in Germany (Host A in Figure 4, Chapter 5) exposes all eight runtime subsystems at once. It lists 14,914 indexed files.

Exposed path	Files	What it shows
`/.hermes/`	14,914	Full tree: `cron`, `kanban`, `curator`, `lsp`, `sessions`, `skills`, `logs`, `state-snapshots`
`/hermes-webui.old.*/.venv/`	883	Virtual environment kept from an older Hermes WebUI install
`/hermes-webui.old.*/tests/`	652	A second WebUI backup left under the HTTP root
`/.hermes/profiles/<name>/`	176	A dedicated persona profile with its own config and state
`/.hermes/logs/curator/`	26	Past Curator run folders
`/.config/` and project logs	30	Service configs (21 files) and agent-managed project logs (9 files)
`/.hermes/.env`	1	Root environment file holding a Telegram bot token

Two WebUI backups, a custom profile and 26 Curator entries point to a workbench that has been upgraded many times. This is a long-lived install, not a one-off test.

CHAPTER 03

Task Boards, Schedules and 31 Tools

IN SHORT

Hermes replaces prompting with two scheduling tools: SQLite Kanban boards that hand cards to isolated workers, and a Cron dispatcher that runs scripts and model briefings on a timer. Monitor both on engineering hosts.

INSIDE THIS CHAPTER

The tools the agent can call 11

Kanban boards and Cron jobs 12

Skills that reach beyond the agent 12

03 Task Boards, Schedules and 31 Tools

The tools the agent can call

Retry dumps on one server contain the agent's full tool list. Hermes registers 31 tools in five groups.

Group	Count	Tools	What they allow
Headless browser	10	`browser_navigate`, `browser_snapshot`, `browser_vision`, `browser_click` and 6 DOM or tab helpers	Drive a Chromium browser and read pages visually
Scheduling and delegation	4	`cronjob`, `delegate_task`, `mixture_of_agents`, `todo`	Schedule jobs, fan out to sub-agents, track cards
Code and shell	5	`execute_code`, `terminal`, `process`, `patch`, `write_file`	Run shell commands, manage processes, patch files
Files and retrieval	5	`read_file`, `search_files`, `session_search`, `web_extract`, `web_search`	Search local files and past sessions, scrape the web
Memory, skills and messaging	7	`memory`, `skill_manage`, `skill_view`, `skills_list`, `send_message` and 2 runtime helpers	Update memory, prune skills, push to Telegram or Discord

An agent with `terminal`, `cronjob` and `send_message` can run shell commands on a timer and report the results to a chat bot. No person needs to be present.

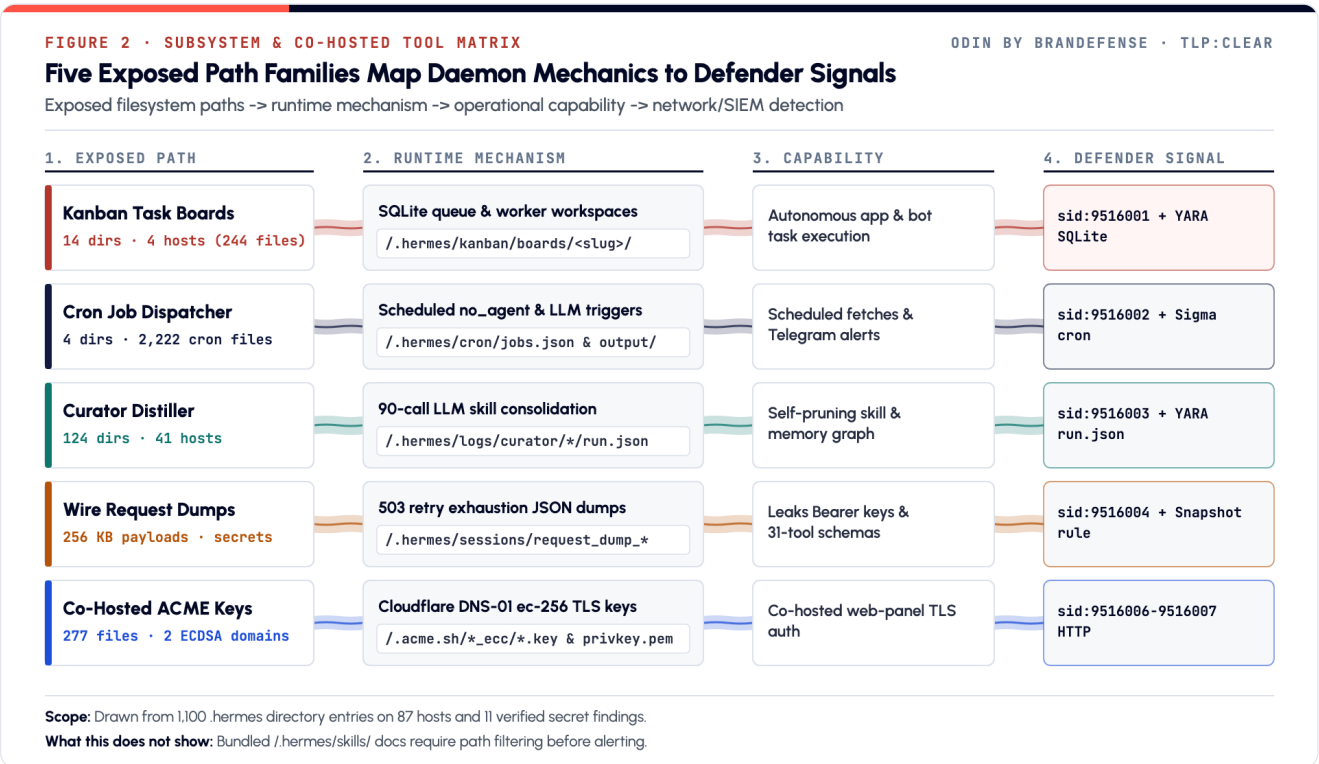


Figure 2: Five exposed path families map daemon mechanics to defender signals

Kanban boards and Cron jobs

Multi-subsystem hosts pair a `cron-pipelines` skill with two Kanban skills: `kanban-orchestrator` and `kanban-worker`. The orchestrator breaks a project into SQLite cards. Each worker runs a card in its own `workspaces/t_<id>/` folder. Task state lives in `kanban.db` and in one `kanban.db` per board.

The Cron dispatcher reads `cron/jobs.json`. It runs two kinds of job. Some are plain `no_agent` Python scripts, guarded by `.tick.lock` and logged in `executions.db`. Others call the model and push a briefing to Telegram.

Host (country)	Exposed path	Files or size	What it holds
Host C (KR)	`kanban/boards/<board>/`	76 files, 258,048-byte DB	Live board database and task cards
Host C (KR)	`kanban/boards/_archived/`	82 files, 105,904-byte log	Archived worker folders and worker logs
Host C (KR)	`cron/output/<job_id>/`	1 file	Scheduled Markdown summary that records a callback endpoint
Host D (DE)	`kanban/boards/`	49 files	Two German-language project boards (23 and 26 files)
Host B (US)	`cron/`, `jobs.json`	2,222 files, 12,048-byte JSON	An hourly `no_agent` fetch job and a thrice-weekly report job
Host E (US)	`cron/output/<job_id>/`	88 files	Scheduled output, with four Telegram bot tokens in `.hermes_history`
Host G (CN)	`kanban.db`, `cron/executions.db`	143,360 bytes in total	Task board (118,784 bytes) and Cron execution log (24,576 bytes)

Job IDs, board names and script names are withheld.

Skills that reach beyond the agent

The skill manifest on one US server (`.skills_prompt_snapshot.json`, 57,659 bytes) shows how Hermes extends its own tools.

- **Delegation to other coding agents.** Hermes ships playbooks for `claude-code`, `codex` and `opencode`. They tell the daemon to start separate Claude Code or OpenAI Codex CLI processes for multi-PR coding work. The same US server also exposes a `/.codex/` folder with 688 files, including the `codex-linux-sandbox` and `codex-execve-wrapper` binaries. A Korean server has the Claude Code CLI wrapper installed under `/.hermes/node/lib/node_modules/`.
- **Terminal email.** Several servers stage `skills/email/himalaya/`. This playbook wires the Rust `himalaya` client into the `terminal` tool, so the agent can read IMAP mail and send through SMTP. The same hosts stage `skills/mcp/native-mcp/` and `skills/red-teaming/`.

04

CHAPTER 04

Curator Runs, Retry Dumps and Snapshots

IN SHORT

Hermes keeps three kinds of file that outlive a key rotation: Curator run logs, raw retry dumps and pre-update snapshots. Rotating a key is not enough; these files must be purged too.

INSIDE THIS CHAPTER

Inside one Curator run 14

Four files that hold secrets 15

04 Curator Runs, Retry Dumps and Snapshots

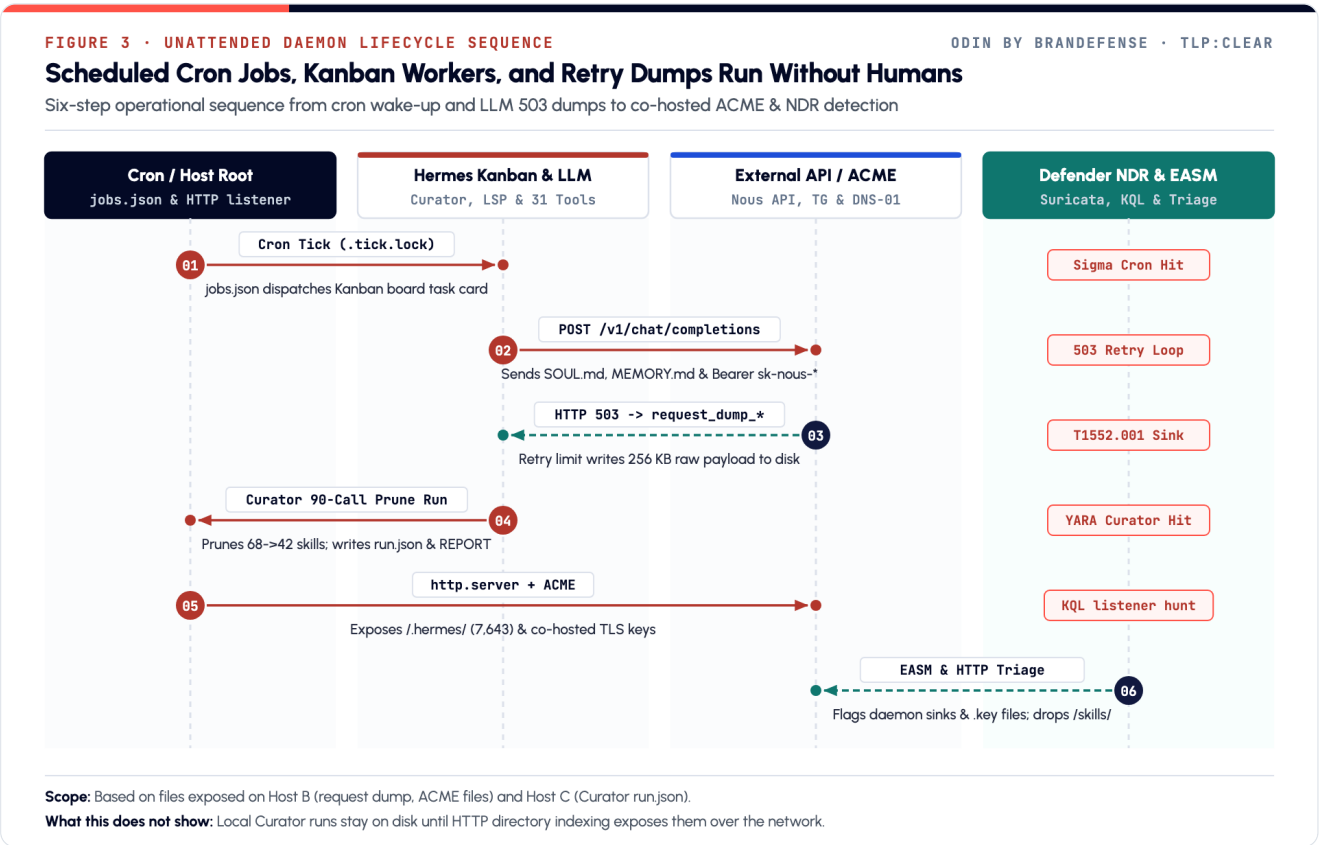


Figure 3: Scheduled Cron jobs, Kanban workers and retry dumps run without humans

Inside one Curator run

The Curator is the second most common background subsystem after logs. It appears on 41 hosts (47.1%), in 124 directory entries. It reviews the skills the agent has written and merges or archives them.

A Korean server (Host C in Figure 4, Chapter 5) exposes one full run. The run log, `run.json`, is 34,236 bytes. The summary, `REPORT.md`, is 8,294 bytes. The log carries its own timestamp of 17 June 2026. The run took 591.25 seconds (9 minutes 51 seconds) and used model `gpt-5.4` through provider `copilot`.

Stage	What happened
1. Pre-check	Reviewed 68 agent-written skills without calling the model. None were marked stale or reactivated; one was seeded.
2. Model pass	Made 90 tool calls: `skill_manage` 35, `read_file` 31, `terminal` 12, `search_files` 6, `skill_view` 5, `execute_code` 1. Merged nine skill clusters into broader skills.
3. Clean-up	Found a missing path, located the support files with `terminal` and moved them. Archived 26 skills (68 down to 42) into `skills/.archive/`, which exposes 100 files. Rewrote no Cron jobs.

The merges are ordinary engineering housekeeping. `kanban-orchestrator` and `kanban-worker` became `kanban-operations`. Six GitHub skills became `github-operations`. Debugging, document and travel-planning skills were also combined. The point for defenders is that the agent edits its own playbooks and runs shell commands while nobody watches.

Four files that hold secrets

Mechanism	File and size	When it is written	What it keeps
Retry dump	`sessions/request_dump_*.json`, 87 to 256 KB each	Upstream HTTP 503 or timeout (`max_retries_exhausted`)	The full `POST /v1/chat/completions` body sent to `inference-api.nousresearch[.]com`, with the `Authorization: Bearer` header, `SOUL.md`, `MEMORY.md`, 31 tool definitions and terminal output
Pre-update snapshot	`state-snapshots/*-pre-update/.env`	Automatically before `hermes update`	Older `.env` and `config.yaml` with Anthropic, OpenRouter, Telegram and Discord keys
Config backups	`.env.bak.*` (23,157 bytes) and 8 `config.yaml.bak.*` copies	Config or router edits	Plaintext key rings and provider fallback lists across many edits
Crash logs	`logs/tui_gateway_crash.log` (219,915 bytes) and `gateway-exit-diag.log` (40,786 bytes)	`SIGTERM` to the gateway	Thread stacks, module paths, process IDs and restart rhythm

The retry dump is the most direct leak. One case on the US server (Host B in Figure 4, Chapter 5) shows how it happens. A user asked Hermes, in Chinese, to report the current fallback provider and model without showing any API key. The agent searched `state-snapshots/` and old request dumps.

The upstream model (`deepseek/deepseek-v4-flash`) then returned HTTP 503. The gateway wrote the whole 256,668-byte request to `sessions/`, bearer header included. The user's request to hide the key did not stop the client from saving it.

CHAPTER 05

Shared Servers: Agent State Next to TLS Keys

IN SHORT

An open file server in `/root` does not stop at `.hermes`. On one US server it also exposed web-panel TLS private keys, a Codex install and backups; on a Korean server it exposed PM2 web workers.

INSIDE THIS CHAPTER

Language servers on seven hosts 17

The US server: Hermes, Codex and ACME keys under one root 17

The Korean server: Kanban, Curator and PM2 18

Multi-profile servers 18

05 Shared Servers: Agent State Next to TLS Keys



Figure 4: Flagship servers pair autonomous Hermes stacks with co-hosted TLS keys and PM2 apps

Language servers on seven hosts

Seven hosts stage `~/.hermes/lsp/`, a headless Language Server Protocol toolchain. On the Korean server it holds 1,791 files. It lets the agent navigate and refactor code across services on the same machine. A US server also carries `lsp/package.json`.

The US server: Hermes, Codex and ACME keys under one root

The US virtual server (Host B) exposes 30 open-directory sites. A retry dump dated May 2026 contains a `ps aux` listing captured by the agent's own `terminal` tool. It shows a bash login shell that changed into `/root` and started a Python script importing `http.server`.

That script served a generated analytics dashboard from `/root` on a high TCP port. Because `http.server` has no path limits, it served everything else in `/root` too. The files cannot tell us whether the owner or the agent started the listener.

Exposed folder	Files	What it holds
`/.hermes/`	7,643	2,222 Cron files with `jobs.json`, `kanban.db` (114,688 bytes), `runtime_router.py`, `.env` (23,252 bytes), retry dumps and a pre-update snapshot
`/` and `/backups/`	6,258	5,974 root files plus 284 backup files, including a recovery archive and project archives
`/.codex/`	688	OpenAI Codex CLI: `logs_2.sqlite-wal` (4.19 MB), shell snapshots, sessions and sandbox binaries
`/.acme.sh/`	277	169 DNS API hooks, 26 notification scripts and two ECDSA `ec-256` certificate folders
Panel domain 2 `_ecc/`	10	Private key (227 bytes), `privkey.pem`, `backup/key.bak`, `fullchain.cer` and a `.conf` with `Le_Webroot='dns_cf'`
Panel domain 1 `_ecc/`	6	Private key, seen on 18 November 2026

The certificate request config for the second domain sets `extendedKeyUsage=serverAuth,clientAuth`. Both certificates were issued through Cloudflare DNS-01 validation (`dns_cf`). Anyone holding these keys could impersonate the two web panels until the certificates are revoked.

The folders share a root, but that is all the files show. They do not tell us whether the agent managed the panels, or who runs the server.

The Korean server: Kanban, Curator and PM2

The Korean server exposes 86 open-directory sites and 4,774 files across six Hermes subsystems.

Component	Files	Note
Active Kanban board	76	Includes the 258,048-byte `kanban.db`
Archived boards	82	Worker folders and logs
`lsp/`	1,791	Language server runtime
`node/lib/`	1,779	Node.js modules, including the Claude Code CLI wrapper
Curator runs	2 folders	Includes the run described in Chapter 4
`/.pm2/`	17	PM2 logs, including `dashboard-error.log`, for Node and Express workers bound locally
Application folder	7	Name withheld
`/.hermes/.env`	1	Holds a Discord bot token

Multi-profile servers

A third US server (Host F) lists 8,440 files across four subsystems: `curator`, `logs`, `lsp` and `skills`. It runs separate bot profiles under `profiles/<name>/.env`. One profile holds a DeepSeek API key and a Telegram bot token. A Chinese server shows the same profile layout.

CHAPTER 06

Separating Real Daemons From Skill Mirrors

IN SHORT

Skill folders are the most common path but the weakest signal. Logs and Curator runs mark a working daemon; task boards, schedules, dumps and keys mark a host that needs action.

INSIDE THIS CHAPTER

Why skills inflate the count 20

The verified secret findings 21

Four kinds of host (analytical model) 21

Other explanations to weigh 21

06 Separating Real Daemons From Skill Mirrors

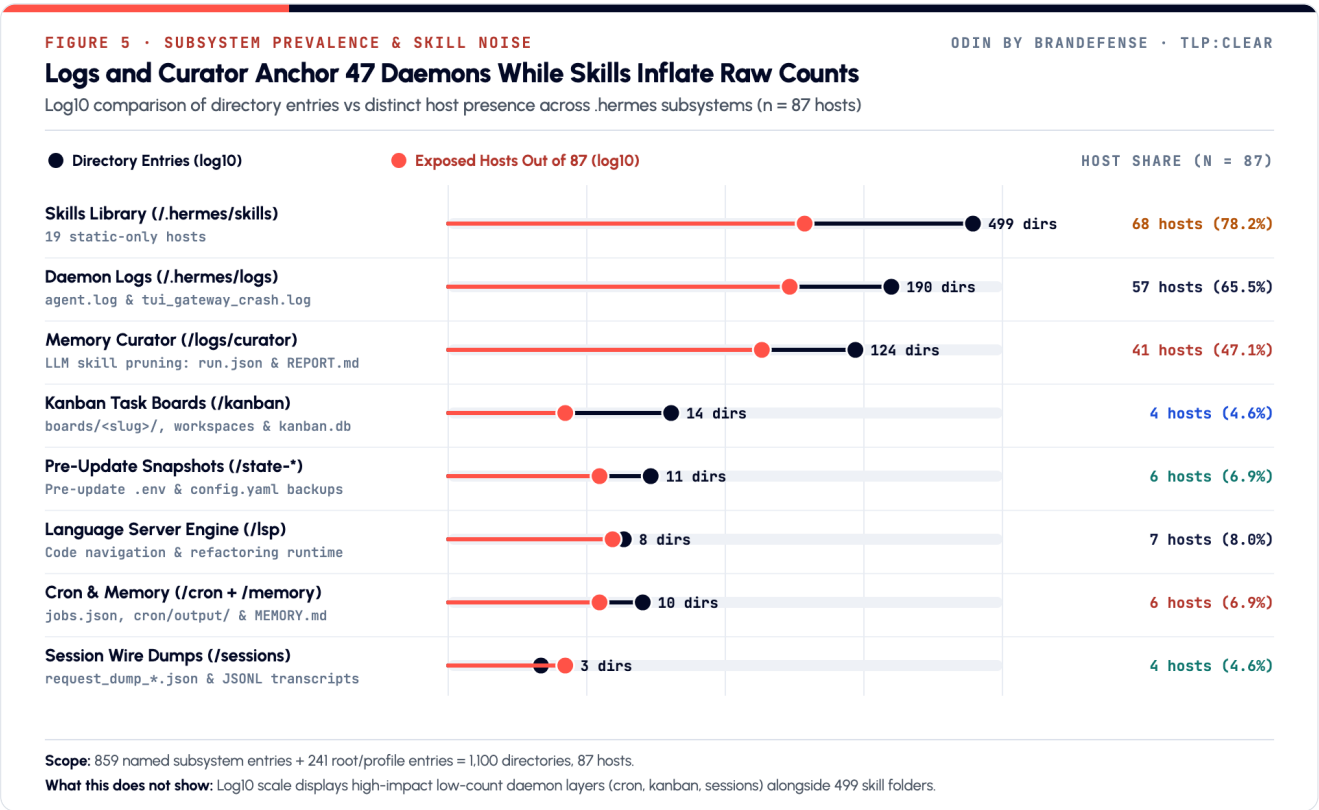


Figure 5: Logs and Curator runs anchor 47 daemons while skills inflate raw counts

Why skills inflate the count

The `skills/` folder makes up 499 of 1,100 entries (45.4%) and appears on 68 hosts. Much of it is bundled open-source documentation. Some bundled docs contain example strings that look like secrets, such as localhost REST examples. Alerting on every `/.hermes/` path would flood analysts with this material.

Logs (57 hosts) and Curator runs (41 hosts) are better markers of a running daemon. Task boards, Cron files, retry dumps and private keys are the paths that call for containment.

The verified secret findings

Eleven secret findings are verified against the original files. They hold 10 distinct values, because one private key appears in two files. All 11 are real exposures; none is a test fixture.

Type	Count
TLS private keys	3
Telegram bot tokens	2
Callback endpoint strings	2
Anthropic API key	1
DeepSeek API key	1
Discord bot token	1
Pastebin drop link	1

Seven of the 11 are application or corporate secrets: the three key files, root `.env` files and a profile token. Four come from agent session files: request dumps, snapshot `.env` files and Cron output. This is a verified sample, not a census. We do not claim a ratio of real secrets to fixtures.

Four kinds of host (analytical model)

The table below is our analytical model. It groups the 87 hosts by what their files show. The files record what was on disk, not what the owner intended.

Model group	Hosts	Share	What the files show	Suggested response
H1: Static skills or minimal config	25	28.7%	19 hosts expose only `skills/` (one lists 11,899 static files); 6 expose root `.env` or `auth.json` without logs	Suppress skill hits; review only if `.env` or `auth.json` is exposed
H2: Active gateway with logs or memory	15	17.2%	`logs/` or `memories/` next to `skills/`, but no Curator, Kanban or Cron	Rotate exposed model and bot keys; close the listener
H3: Self-directed workflow daemon	46	52.9%	32 Curator or snapshot hosts plus 14 multi-subsystem Kanban, Cron or language-server hosts	Check egress for unsanctioned automation; purge snapshots and dumps when rotating keys
H4: Daemon plus co-hosted web-panel keys	1	1.1%	The US server with 7,643 Hermes files, 2,222 Cron files, Codex and two ACME key folders	Revoke the TLS certificates and bot tokens; move `acme.sh` out of `/root`

H3 and H4 together are the 47 hosts with background subsystems.

Other explanations to weigh

Three explanations apply across all four groups. None of them points to an attack.

- **Authorized testing.** Some hosts stage `skills/red-teaming/` playbooks next to normal developer tools. Offensive skill templates inside a bundled skill tree fit an authorized test bench or a default checkout. They do not show intrusion activity.

- **Research benches.** Heavy Curator use (41 hosts) and Kanban use (4 hosts), including the multi-profile servers, fit engineering teams or labs testing unattended agent memory and task splitting.
- **Decoys and unconfigured mirrors.** Some of the 25 static H1 hosts may be decoys or unused mirrors that attract scanners.

Across all groups, the files record past runs and local configuration. They do not show attacks on third-party networks.

07

CHAPTER 07

What Defenders Should Do

IN SHORT

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

07 What Defenders Should Do

Within 24 hours

- [ ] Deny hidden-folder URIs at the web edge (Nginx, Caddy, Apache, CDN). Requests for `/.hermes`, `/.acme.sh`, `/.pm2` and `/.codex` should return 403 or 404, with `autoindex off`.
- [ ] Find and stop non-loopback `python3 -m http.server` processes, or scripts that import `http.server`, running in `/root` or `/home/`. Local previews must use `--bind 127.0.0.1`.
- [ ] If you find your own host in this pattern, revoke any TLS certificate whose key sat in an exposed `acme.sh` folder.

Within 72 hours

- [ ] Move `acme.sh` keys out of home folders, for example with `--cert-home /etc/ssl/acme`, and set `chmod 600`.
- [ ] Each time you rotate a model, Telegram or Discord token, also delete `sessions/request_dump_*.json`, `state-snapshots/*-pre-update/.env` and `.env.bak.*`.
- [ ] Review `himalaya` mail credentials, and the permissions of any `claude-code` or `codex` sub-process, on shared hosts.

Within 7 to 30 days

- [ ] Register every unattended agent Cron job and Kanban board as a service account. Give it token spend caps and an egress allowlist.
- [ ] Deploy the Suricata signatures 9516001 to 9516010 and check that 9516005 excludes `/skills/`.
- [ ] Compare internal vault entries against exposed-secret alerts, and check certificate transparency logs for unexpected certificates on your panel domains.

CHAPTER 08

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 27

08 Detection Rules

The full rules are in Appendix B. Each rule excludes the bundled Hermes skills folder or loopback traffic, so static mirrors and local previews do not fire.

Rule file	What it catches
`odin_hermes_daemon_and_panel_exposure.rules`	Suricata 7: ten signatures for requests to Hermes daemon paths and `acme.sh` keys, and for responses that deliver keys, configs, dumps or directory listings
`sentinel_unattended_ai_daemon_cron_and_acme_leak.kql`	Microsoft Sentinel and Defender XDR: open file-server commands, writes to daemon and key files, unattended Hermes processes, and listeners on daemon ports
`sigma_hermes_autonomous_agent_cron_and_daemon.yml`	Sigma (3 rules): Hermes gateway and Curator processes, web servers returning daemon files or keys, and ad-hoc file servers in dotfile paths
`odin_hermes_kanban_curator_and_request_dumps.yar`	YARA (4 rules): Curator run files, retry dumps and crash logs, Kanban and Cron stores, and ACME key material

How the Suricata signatures work

Signatures 9516001 to 9516006 watch requests. Each one sets a flowbit. Signatures 9516007 to 9516009 only fire on a 200 response in a flow where that flowbit is set. Suricata 7.0 or later is needed, with the `http.uri`, `http.stat_code` and `file.data` buffers.

SID	Looks for	Why
9516001	`GET` for `/.hermes/kanban` with `/boards/`, `kanban.db` or `/workspaces/t_<hex>`	Task board and worker folders (T1213)
9516002	`GET` for `/.hermes/cron/` with `jobs.json`, `executions.db`, `.tick.lock` or `/output/`	Scheduler jobs and output (T1213)
9516003	`GET` for `/.hermes/logs/curator/` with `run.json`, `REPORT.md` or a dated run folder	Curator runs (T1213)
9516004	`GET` for `request_dump_*.json` under `/.hermes/sessions/`	Retry dumps with bearer headers (T1552.001)
9516005	`GET` under `/.hermes/` for `state-snapshots/`, a profile `auth.json` or `.env`, or `tui_gateway_crash.log`, excluding `/skills/`	Snapshot and profile secrets (T1552.001)
9516006	`GET` under `/.acme.sh/` for `.key`, `privkey.pem`, `key.bak` or an `_ecc` `.conf`	Certificate keys and DNS-01 configs (T1552.004)
9516007	200 response body with `-----BEGIN ... PRIVATE KEY-----`	A private key was actually served
9516008	200 response body with `Le_RealKeyPath=` and `Le_Webroot=` or `Le_Keylength=`	An `acme.sh` domain config was served
9516009	200 response body with `Hermes Agent Persona` and `max_retries_exhausted` or `upstream_timeout`	A retry dump was served
9516010	200 listing (`Index of /` or `Directory listing for /`) that links to `.hermes/` or `.acme.sh/`	An open directory exposing either folder

Suppress your own authorized attack-surface scanners by exact source address in `threshold.conf`, not by CIDR ranges.

How the other rules work

The KQL query has four branches and a 14-day lookback. Results are grouped per device and day, and device-days with two or more distinct signals come first.

Branch	Looks for
Process	`http.server`, `SimpleHTTPServer`, `http-server`, `npx serve` or `miniserve` started from `/root`, `/home/`, `.hermes` or `.acme.sh`, without a loopback bind
File	Files created, changed or renamed under `.hermes` daemon folders or `acme.sh` `.key` files
Daemon	`tui_gateway/entry.py`, `runtime_router.py`, `hermes curator` or `kanban-orchestrator`, or a Hermes parent that starts `himalaya` or `acme.sh`
Network	A web-server process listening on a daemon port (for example 8080, 8899, 8765, 18080) on a non-loopback address, or accepting public connections

The network branch matters because a custom wrapper script can import `http.server` without naming it on the command line. Fill the `ApprovedInternalBuildHosts` list with exact host names of sanctioned automation runners.

The three Sigma rules need Linux `auditd` and web access logs. The first flags Hermes gateway, router and Curator processes, excluding `pytest`. The second flags a web server returning 200 for daemon files or `acme.sh` key files; check `cs-username` to confirm anonymous access. The third flags non-loopback file servers or root Python wrappers in dotfile paths.

YARA rule	Fires on
`ODIN_Hermes_Curator_Distillation_Run_And_Report`	Curator `run.json` and `REPORT.md` content
`ODIN_Hermes_Request_Dump_And_Gateway_Crash`	Retry dumps and gateway crash logs
`ODIN_Hermes_Kanban_Board_And_Cron_Scheduler`	Kanban SQLite databases and Cron job stores
`ODIN_Cohosted_Panel_ACME_TLS_Material`	`acme.sh` key and config files

Each YARA rule is limited to files under 20 MB and needs YARA 4.3 or later.

What triage sees	Action
Listener bound to loopback only	Ignore: local development
Only `skills/` or `node_modules/` paths	Compare with the upstream package before closing
Gateway logs or a root `.env`	Review and rotate keys
Any Kanban, Cron, Curator or retry-dump path	Treat as an unattended daemon exposure
Daemon paths plus an exposed `acme.sh` `.key` or private key	Most urgent: revoke certificates and tokens

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. Their logic was checked in Python against sample files and negative controls. Load and test each rule in staging before production use.

09

CHAPTER 09

Indicators

IN SHORT

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

INSIDE THIS CHAPTER

What to hunt for instead 29

MITRE ATT&CK mapping 29

09 Indicators

No indicator from this study is published as a value. All 28 point to exposed owners, not to attackers.

Type	Count	Status
Flagship exposed Hermes hosts (Hosts A to F)	6	Withheld: exposed owners
Co-hosted web-panel domains	2	Withheld: exposed owners
File hashes of exposed key, database, Cron, Curator, dump and crash-log files	10	Withheld: owners' files
Hashes of exposed secret values	10	Withheld: secret material

Host G and the other labelled hosts appear in the text but are not counted as indicators.

Withheld: all host addresses and ports, both panel domains, board, job, profile and script names, and every file or secret-value hash.

What to hunt for instead

The patterns below are stable across installs and safe to publish.

Pattern	Where to look	Typical size
`/.hermes/kanban.db`, `kanban/boards/<slug>/kanban.db`	Web logs, file scans	114,688 to 258,048 bytes
`/.hermes/cron/jobs.json`, `cron/executions.db`	Web logs, file scans	12,048 and 24,576 bytes
`/.hermes/logs/curator/<run>/run.json`, `REPORT.md`	Web logs, file scans	34,236 and 8,294 bytes
`/.hermes/sessions/request_dump_*.json`	Web logs, file scans	87 to 256 KB
`/.hermes/logs/tui_gateway_crash.log`	Web logs, file scans	219,915 bytes
`/.hermes/state-snapshots/*-pre-update/.env`, `.env.bak.*`	Web logs, file scans	23,157 bytes for one backup
`/.acme.sh/*_ecc/*.key`, `privkey.pem`, `backup/key.bak`	Web logs, file scans	227 bytes for one key
`Directory listing for /` or `Index of /` linking to `.hermes/`	HTTP responses	n/a

MITRE ATT&CK mapping

These techniques describe what an attacker could do with the exposed files. They do not mean the owners were attacked.

Tactic	Technique	Exposure	Detection
Collection	T1213 Data from Information Repositories	Kanban boards, Cron stores and Curator runs readable over HTTP	SIDs 9516001 to 9516003, 9516010; Sigma rule 2
Credential Access	T1552.001 Unsecured Credentials: Credentials In Files	Bearer headers in retry dumps; keys in `.env`, snapshots and backups	SIDs 9516004, 9516005, 9516009; YARA dump rule
Credential Access	T1552.004 Unsecured Credentials: Private Keys	`acme.sh` ECDSA keys for two web-panel domains	SIDs 9516006 to 9516008; YARA ACME rule

10

CHAPTER 10

Scope and Limits

IN SHORT

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

10 Scope and Limits

- **Window.** The findings come from a two-week observation window ending 18 November 2026. Files on the exposed servers carry their own earlier dates. These include a retry dump dated May 2026 and a Curator run log dated 17 June 2026, both written by the agents themselves. Some servers were first listed before this window; no dates are given for those listings. They are file timestamps, not the span of our observation.
- **A sample, not a census.** The counts describe these hosts, not every Hermes install online.
- **Files, not activity.** Directory listings show what was stored on disk. They do not show running processes, intent, or whether a scheduled job ran at a given time.
- **Owners are not attackers.** All 87 hosts are exposed systems. Red-team skills on some hosts fit authorized testing as well as misuse.
- **No downstream compromise shown.** Nothing here shows a breach of the model providers, of the panel domains' users, or of any third-party network.
- **Unknown listener origin.** On the US server, the files cannot tell us whether a person or the agent started the file server.
- **Verified secrets only.** The 11 secret findings are those matched to original files. Other secrets may exist on these hosts.
- **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.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. Nous Research. Hermes Agent documentation. https://hermes-agent.nousresearch.com/docs (verified 18 November 2026)
2. Nous Research. hermes-agent repository. https://github.com/NousResearch/hermes-agent (verified 18 November 2026)
3. acmesh-official. acme.sh repository. https://github.com/acmesh-official/acme.sh (verified 18 November 2026)
4. PM2. Quick start and log management. https://pm2.keymetrics.io/docs/usage/quick-start/ (verified 18 November 2026)
5. Pimalaya. Himalaya CLI email client. https://github.com/pimalaya/himalaya (verified 18 November 2026)
6. MITRE ATT&CK. T1213 Data from Information Repositories. https://attack.mitre.org/techniques/T1213/ (verified 18 November 2026)
7. MITRE ATT&CK. T1552.004 Unsecured Credentials: Private Keys. https://attack.mitre.org/techniques/T1552/004/ (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)

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_hermes_daemon_and_panel_exposure.rules

SURICATA

```
# Odin by Brandefense - Monograph #16 Suricata 7.x Autonomous Hermes Daemon & Co-Hosted ACME TLS Suite
# Classification: TLP:CLEAR
# Scope Note: Rules 9516001-9516006 detect attempted unauthenticated HTTP GET requests for exposed
#             .hermes autonomous daemon subsystems (Kanban boards, Cron schedulers, Curator runs,
#             request_dump wire logs, and state-snapshots) and co-hosted /.acme.sh/ private-key URIs,
#             setting flowbit odin.hermes.sensitive_uri_req. Rules 9516007-9516010 inspect inbound
#             from_server HTTP 200 responses across separate rules for PEM private-key bodies,
#             acme.sh DNS-01 configs, Hermes retry-dump JSON payloads, and Nginx/Apache or Python
#             http.server open-directory listings

# 1. Attempted HTTP GET for Hermes autonomous Kanban board, SQLite database, or worker workspace (/.hermes/
kanban/)
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for Autonomous Kanban Board or SQLite
Queue"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.hermes/kanban"; nocase;
fast_pattern; pcre:"/(\/boards\/|kanban\.db|\/workspaces\/t_[0-9a-f]+)/i";
flowbits:set,odin.hermes.sensitive_uri_req; metadata:created_at 2026_11_18, mitre_technique_id T1213;
classtype:policy-violation; sid:9516001; rev:3;)

# 2. Attempted HTTP GET for Hermes application Cron schedule, SQLite execution store, or output (/.hermes/cron/)
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for Unattended Cron Jobs or Scheduled
Output"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.hermes/cron/"; nocase;
fast_pattern; pcre:"/(jobs\.json|executions\.db|\.tick\.lock|\/output\/)/i";
flowbits:set,odin.hermes.sensitive_uri_req; metadata:created_at 2026_11_18, mitre_technique_id T1213;
classtype:policy-violation; sid:9516002; rev:3;)

# 3. Attempted HTTP GET for Hermes automated Curator skill-distillation run.json or REPORT.md (/.hermes/logs/
curator/)
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for Curator Memory Distillation Run or
Report"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.hermes/logs/curator/";
nocase; fast_pattern; pcre:"/(run\.json|REPORT\.md|\d{8}-\d{6}\/)/i"; flowbits:set,odin.hermes.sensitive_uri_req;
metadata:created_at 2026_11_18, mitre_technique_id T1213; classtype:policy-violation; sid:9516003; rev:3;)

# 4. Attempted HTTP GET for Hermes raw HTTP inference wire payload dump (/.hermes/sessions/request_dump_*.json)
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for Raw LLM Inference Request Dump
JSON"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"request_dump_"; nocase;
fast_pattern; content:"/.hermes/sessions/"; nocase; content:".json"; nocase;
flowbits:set,odin.hermes.sensitive_uri_req; metadata:created_at 2026_11_18, mitre_technique_id T1552_001;
classtype:credential-theft; sid:9516004; rev:3;)

# 5. Attempted HTTP GET for Hermes pre-update state-snapshots or bot profile secrets
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for Pre-Update State Snapshot or Bot
Profile Secrets"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.hermes/"; nocase;
fast_pattern; pcre:"/(state-snapshots\/|profiles\/[^\/]+\/(auth\.json|\.env)|tui_gateway_crash\.log)/i";
flowbits:set,odin.hermes.sensitive_uri_req; metadata:created_at 2026_11_18, mitre_technique_id T1552_001;
classtype:credential-theft; sid:9516005; rev:5;)

# 6. Attempted HTTP GET for co-hosted acme.sh ECDSA/RSA private key (.key, privkey.pem, key.bak) or DNS-01 config
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Attempted HTTP GET for ACME TLS Private Key or DNS-01
Config"; flow:established,to_server; http.method; content:"GET"; http.uri; content:"/.acme.sh/"; nocase;
fast_pattern; pcre:"/(\.key|privkey\.pem|key\.bak|_ecc\/.*\.conf)$/i";
flowbits:set,odin.hermes.sensitive_uri_req; metadata:created_at 2026_11_18, mitre_technique_id T1552_004;
classtype:credential-theft; sid:9516006; rev:3;)

# 7. Confirmed HTTP 200 server response delivering exposed PEM ECDSA/RSA private-key body
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Confirmed HTTP 200 Response Serving PEM TLS Private Key
Body"; flow:established,from_server; flowbits:isset,odin.hermes.sensitive_uri_req; http.stat_code; content:"200";
file.data; content:"PRIVATE KEY-----"; fast_pattern; pcre:"/-----BEGIN (EC |RSA |OPENSSH )?PRIVATE KEY-----/";
metadata:created_at 2026_11_18, mitre_technique_id T1552_004; classtype:credential-theft; sid:9516007; rev:3;)

# 8. Confirmed HTTP 200 server response delivering exposed acme.sh DNS-01 domain configuration
```

```
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Confirmed HTTP 200 Response Serving ACME DNS-01
Configuration"; flow:established,from_server; flowbits:isset,odin.hermes.sensitive_uri_req; http.stat_code;
content:"200"; file.data; content:"Le_RealKeyPath="; fast_pattern; pcre:"/(Le_Webroot=|Le_Keylength=)/";
metadata:created_at 2026_11_18, mitre_technique_id T1552_004; classtype:credential-theft; sid:9516008; rev:3;)

# 9. Confirmed HTTP 200 server response delivering Hermes retry-exhaustion request_dump wire payload
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Confirmed HTTP 200 Response Serving Hermes Retry-
Exhaustion Wire Dump"; flow:established,from_server; flowbits:isset,odin.hermes.sensitive_uri_req;
http.stat_code; content:"200"; file.data; content:"Hermes Agent Persona"; fast_pattern; pcre:"/
\x22(max_retries_exhausted|upstream_timeout)\x22/"; metadata:created_at 2026_11_18, mitre_technique_id T1552_001;
classtype:credential-theft; sid:9516009; rev:3;)

# 10. Confirmed HTTP 200 open-directory listing (Apache/Nginx Index of / or Python http.server Directory listing
for /)
alert http any any -> any any (msg:"ODIN-HERMES-DAEMON Confirmed HTTP 200 Open Directory Listing Exposing Hermes
or ACME Root"; flow:established,from_server; http.stat_code; content:"200"; file.data; content:"href="; nocase;
fast_pattern; pcre:"/(Index of \/|Directory listing for \/)/i"; pcre:"/href=[\x22\x27]\/?(\.hermes|\.acme\.sh)\//
i"; metadata:created_at 2026_11_18, mitre_technique_id T1213; classtype:policy-violation; sid:9516010; rev:3;)
```

odin_hermes_kanban_curator_and_request_dumps.yar

YARA

```
/*
   Odin by Brandefense - Monograph #16 YARA Suite for Autonomous Hermes Daemons & Co-Hosted ACME Artifacts
   Classification: TLP:CLEAR
   Scope: 4 deterministic structural rules identifying exposed Hermes Curator distillation logs
          (run.json / REPORT.md), raw HTTP retry-exhaustion request dumps and TUI gateway crash logs,
          Kanban SQLite task boards and Cron job schedules, and co-hosted acme.sh ECDSA TLS material,
          while avoiding false positives on bundled OSS skill documentation and npm package files.
*/

rule ODIN_Hermes_Curator_Distillation_Run_And_Report
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects unattended Hermes Curator LLM skill-consolidation run.json and REPORT.md outputs"
        mitre_attack = "T1213"
    strings:
        $rj1 = "\"auto_transitions\"" ascii
        $rj2 = "\"cron_jobs_rewritten\"" ascii
        $rj3 = "\"skill_manage\"" ascii
        $rp1 = "## Auto-transitions (pure, no LLM)" ascii
        $rp2 = "## LLM consolidation pass" ascii
        $rp3 = "hermes curator restore" ascii
    condition:
        filesize < 20MB and (
            ($rj1 and $rj2 and $rj3) or
            ($rp1 and $rp2 and $rp3)
        )
}

rule ODIN_Hermes_Request_Dump_And_Gateway_Crash
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects raw Hermes LLM inference request_dump_*.json payloads and tui_gateway_crash.log
stack traces"
        mitre_attack = "T1552.001"
    strings:
        $rd1 = "\"session_id\"" ascii
        $rd2 = "# Hermes Agent Persona" ascii
        $rd_r1 = "\"max_retries_exhausted\"" ascii
        $rd_r2 = "\"upstream_timeout\"" ascii
        $gw1 = "hermes-agent/tui_gateway/entry.py" ascii
        $gw2 = "main-thread stack at signal delivery:" ascii
```

```
    condition:
        filesize < 20MB and (
            ($rd1 and $rd2 and ($rd_r1 or $rd_r2)) or
            ($gw1 and $gw2)
        )
}

rule ODIN_Hermes_Kanban_Board_And_Cron_Scheduler
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects Hermes Kanban SQLite board databases, worker task logs, and application Cron job
definitions"
        mitre_attack = "T1213"
    strings:
        $cr1 = "\"job_id\"" ascii
        $cr2 = "\"schedule\"" ascii
        $cr3 = "/.hermes/" ascii
        $sq_hdr = "SQLite format 3" ascii
        $kb_ddl1 = "CREATE TABLE kanban_boards" ascii
        $kb_ddl2 = "CREATE TABLE kanban_cards" ascii
        $kb_ddl3 = "CREATE TABLE cron_executions" ascii
        $ko1 = "# Hermes Cron Execution Output" ascii
        $ko2 = "board_slug:" ascii
        $ko3 = "/.hermes/kanban/boards/" ascii
    condition:
        filesize < 20MB and (
            ($cr1 and $cr2 and $cr3) or
            ($sq_hdr and ($kb_ddl1 or $kb_ddl2 or $kb_ddl3)) or
            ($ko1 and $ko2 and $ko3)
        )
}

rule ODIN_Cohosted_Panel_ACME_TLS_Material
{
    meta:
        author = "Odin by Brandefense"
        classification = "TLP:CLEAR"
        date = "2026-11-18"
        description = "Detects exposed acme.sh ECDSA DNS-01 configuration and dual-use CSR files for co-hosted
web-panel TLS domains"
        mitre_attack = "T1552.004"
    strings:
        $ac1 = "Le_Domain=" ascii
        $ac2 = "/.acme.sh/" ascii
        $ac3 = "Le_Keylength='ec-256'" ascii
        $csr1 = "[ req_distinguished_name ]" ascii
        $csr2 = "extendedKeyUsage=serverAuth,clientAuth" ascii
        $csr3 = "subjectAltName=DNS:" ascii
    condition:
        filesize < 20MB and (
            ($ac1 and $ac2 and $ac3) or
            ($csr1 and $csr2 and $csr3)
        )
}
```

sentinel_unattended_ai_daemon_cron_and_acme_leak.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Monograph #16 Sentinel & Defender XDR Hunt for Autonomous Hermes Daemons & ACME Leaks
// Classification: TLP:CLEAR
// Uses regex and contains operators (not has/has_any) so punctuation-bearing path fragments match as substrings.
// Correlates non-loopback HTTP serving launched in /root or /home via command-line or network listener
telemetry,
// file writes/modifications inside Hermes Kanban/Cron/Curator or ACME private-key sinks, and unattended daemon
runs.
let Lookback = 14d;
```

```
let LoopbackAddresses = dynamic(["127.0.0.1", "::1", "0:0:0:0:0:0:0:1"]);
let ApprovedInternalBuildHosts = dynamic(["ci-artifact-cache-01.corp.internal", "devbox-
mirror-01.corp.internal"]);
let ExposedDaemonPorts = dynamic([3000, 5000, 6000, 7788, 8000, 8080, 8081, 8088, 8099, 8765, 8787, 8888, 8898,
8899, 8900, 9090, 9999, 18080]);
let WebServerProcesses = dynamic(["python", "python3", "nginx", "caddy", "httpd", "apache2", "miniserve", "node",
"php"]);
let HttpServerCommandRegex = @"(?i)(http\.server|SimpleHTTPServer|http-server|npx\s+serve|miniserve|-S\s+0\.0\.
0\.0:)";
let RootOrHermesContextRegex = @"(?i)(/root|/home/|\.hermes|\.acme\.sh)";
let WrapperListenerRegex = @"(?i)(python3?\s+\S*server\S*\.py|--port\s+\d+|0\.0\.0\.0:\d+)";
let SensitiveSinkRegex = @"(?i)(/\.hermes/(kanban|cron|logs/curator|state-snapshots)/|kanban\.db|jobs\.json|
executions\.db|request_dump_|tui_gateway_crash\.log|/\.acme\.sh/|privkey\.pem)";
let HermesDaemonCommandRegex = @"(?i)(tui_gateway/entry\.py|runtime_router\.py|hermes\s+curator|/\.hermes/cron/|
kanban-orchestrator)";
let HermesChildCommandRegex = @"(?i)(himalaya|acme\.sh|dns_cf)";
union isfuzzy=true
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where FileName in~ ("python", "python3", "node", "npx", "miniserve", "caddy", "php", "bash", "sh")
    | where ProcessCommandLine matches regex HttpServerCommandRegex
        or (ProcessCommandLine matches regex RootOrHermesContextRegex and ProcessCommandLine matches regex
WrapperListenerRegex)
    | where ProcessCommandLine matches regex RootOrHermesContextRegex
        or InitiatingProcessCommandLine matches regex RootOrHermesContextRegex
    | extend HuntSignal = "NonLoopback_HTTP_Listener_Command_In_Root_Or_Hermes_Home",
             ArtifactDetail = strcat(FileName, " :: ", ProcessCommandLine)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName = FileName, TargetItem = FileName,
ArtifactDetail
),
(
    DeviceFileEvents
    | where Timestamp >= ago(Lookback)
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where ActionType in ("FileCreated", "FileModified", "FileRenamed")
    | extend FullPath = strcat(FolderPath, "/", FileName)
    | where FullPath matches regex @"(?i)/\.(hermes|acme\.sh)(/|$)"
    | where FullPath matches regex SensitiveSinkRegex or FileName endswith ".key"
    | extend HuntSignal = "Hermes_Autonomous_Sink_Or_ACME_Key_File_Write",
             ArtifactDetail = strcat(ActionType, " :: ", FullPath)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = FileName, ArtifactDetail
),
(
    DeviceProcessEvents
    | where Timestamp >= ago(Lookback)
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where ProcessCommandLine matches regex HermesDaemonCommandRegex
        or (InitiatingProcessCommandLine matches regex @"(?i)hermes" and ProcessCommandLine matches regex
HermesChildCommandRegex)
    | extend HuntSignal = "Unattended_Hermes_Daemon_Curator_Cron_Or_Skill_Execution",
             ArtifactDetail = ProcessCommandLine
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = FileName, ArtifactDetail
),
(
    DeviceNetworkEvents
    | where Timestamp >= ago(Lookback)
    | where DeviceName !in~ (ApprovedInternalBuildHosts)
    | where ActionType in ("ListeningConnectionCreated", "InboundConnectionAccepted", "ConnectionSuccess")
    | where InitiatingProcessFileName in~ (WebServerProcesses)
    | where LocalPort in (ExposedDaemonPorts) and not(LocalIP in (LoopbackAddresses))
    | where RemoteIPType == "Public" or ActionType == "ListeningConnectionCreated"
    | extend HuntSignal = "NonLoopback_WebServer_Listener_Or_Public_Connection_On_Daemon_Port",
             ArtifactDetail = strcat(ActionType, " ", LocalIP, ":", tostring(LocalPort), " <- ", RemoteIP, " ::
", InitiatingProcessCommandLine)
    | project Timestamp, DeviceName, HuntSignal, InitiatingProcessFileName, TargetItem = strcat(LocalIP, ":",
tostring(LocalPort)), ArtifactDetail
)
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
```

```
    EventCount = count(),
    DistinctSignals = dcount(HuntSignal),
    SignalsObserved = make_set(HuntSignal, 4),
    SampleArtifacts = make_set(ArtifactDetail, 5),
    Processes = make_set(InitiatingProcessFileName, 5),
    TargetItems = make_set(TargetItem, 5)
    by DeviceName, TimeWindow = bin(Timestamp, 1d)
| where EventCount >= 1
| order by DistinctSignals desc, EventCount desc
```

sigma_hermes_autonomous_agent_cron_and_daemon.yml

SIGMA

```
title: Unattended Hermes Autonomous Daemon Cron, Curator, or TUI Gateway Execution
id: 7d4e9a12-5c16-4b88-8e41-16a100000001
status: experimental
description: Detects unattended execution of the Hermes autonomous agent gateway (tui_gateway/entry.py,
runtime_router.py), application scheduler jobs (/.hermes/cron/), or automated Curator skill-pruning passes on
Linux hosts.
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://hermes-agent.nousresearch.com/docs
  - https://github.com/NousResearch/hermes-agent
tags:
  - attack.execution
  - attack.t1059.006
logsource:
  category: process_creation
  product: linux
detection:
  selection_hermes_daemon:
    CommandLine|contains:
      - 'tui_gateway/entry.py'
      - 'runtime-controller/runtime_router.py'
      - 'runtime-controller/run_gateway.py'
      - 'hermes curator'
      - '/.hermes/cron/'
      - '/.hermes/kanban/'
  condition: selection_hermes_daemon
falsepositives:
  - Sanctioned engineering sandbox hosts explicitly registered to run unattended Hermes automation agents.
level: medium
---
title: Web Server HTTP 200 Serving of Hermes Daemon Sinks or ACME TLS Private Keys
id: 7d4e9a12-5c16-4b88-8e41-16a100000002
status: experimental
description: Detects HTTP 200 responses serving exposed Hermes autonomous daemon state sinks (kanban, cron, logs/
curator, request_dump_*, state-snapshots, or profile secrets) or co-hosted acme.sh TLS private-key files (.key,
privkey.pem, key.bak) while excluding bundled Hermes OSS skill Markdown files. Because HTTP status 200 alone does
not prove authentication was absent, confirm anonymous access via cs-username or reverse-proxy auth fields, and
treat broader /.acme.sh/ directory hits without private-key filenames as review signals.
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://github.com/NousResearch/hermes-agent
  - https://attack.mitre.org/techniques/T1552/004/
tags:
  - attack.credential_access
  - attack.t1552.001
  - attack.t1552.004
  - attack.collection
  - attack.t1213
logsource:
  category: webserver
detection:
  selection_status:
    sc-status: 200
  selection_hermes_sinks:
```

```
    cs-uri-stem|contains:
      - '/.hermes/kanban/'
      - '/.hermes/kanban.db'
      - '/.hermes/cron/'
      - '/.hermes/logs/curator/'
      - '/.hermes/sessions/request_dump_'
      - '/.hermes/state-snapshots/'
      - '/.hermes/profiles/'
      - '/.hermes/.env'
      - '/.hermes/auth.json'
  selection_acme_private_key_path:
    cs-uri-stem|contains: '/.acme.sh/'
    cs-uri-stem|endswith:
      - '.key'
      - 'privkey.pem'
      - 'key.bak'
  condition: selection_status and (selection_hermes_sinks or selection_acme_private_key_path)
falsepositives:
  - Internal forensic artifact mirrors protected by reverse-proxy or mutual TLS authentication (verify empty cs-
username before escalating).
level: critical
---
title: Non-Loopback Ad-Hoc HTTP Directory Server or Root Python Wrapper Launched in Dotfile Path
id: 7d4e9a12-5c16-4b88-8e41-16a100000003
status: experimental
description: Detects ad-hoc Python, Node, PHP, or miniserve HTTP directory listeners and root/home Python server
wrapper invocations (such as cd /root && python3 <script>.py) referencing /root, /home/, .hermes, or .acme.sh
without an explicit loopback bind argument (127.0.0.1, localhost, or ::1). Because process_creation command lines
cannot inspect import http.server inside arbitrary custom script names launched without directory or server-name
cues, pair this rule with socket/network listener telemetry (Sentinel KQL DeviceNetworkEvents
ListeningConnectionCreated).
author: Odin by Brandefense
date: 2026/11/18
references:
  - https://docs.python.org/3/library/http.server.html
  - https://github.com/acmesh-official/acme.sh
tags:
  - attack.collection
  - attack.t1213
logsource:
  category: process_creation
  product: linux
detection:
  selection_http_listener:
    CommandLine|contains:
      - 'http.server'
      - 'SimpleHTTPServer'
      - 'http-server'
      - 'npx serve'
      - 'miniserve'
      - '-S 0.0.0.0:'
  selection_python_exec:
    CommandLine|contains:
      - 'python '
      - 'python3 '
  selection_python_wrapper_context:
    CommandLine|contains:
      - 'cd /root &&'
      - 'cd /home/'
      - 'server.py'
      - '-server.py'
      - '_server.py'
  selection_home_or_dotfile_context:
    CommandLine|contains:
      - '/root'
      - '/home/'
      - '.hermes'
      - '.acme.sh'
  condition: ((selection_http_listener and selection_home_or_dotfile_context) or (selection_python_exec and
```

```
  selection_python_wrapper_context and selection_home_or_dotfile_context))
  falsepositives:
    - Non-networked local utility scripts invoked via cd /root && python3 (correlate with
  ListeningConnectionCreated on ports 3000-18080).
  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/unattended-ai-agents-exposed/
Short Read · brandefense.io/odin-research/blog/unattended-ai-agents-are-leaving-their-keys-on-open-servers/
Defense Kit · brandefense.io/odin-research/defense-kit/