t-defence

Open source vulnerabilty hunting: Hermes Agent

Dashboard OAuth PKCE flow persisted Anthropic refresh token without restrictive file permissions

#TDefenceBusiness

Malware Lab

Summary

Our Malware Lab

T-Defence Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malware.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

Executive Summary

The dashboard OAuth flow in `hermes_cli/web_server.py` persists the Anthropic PKCE response (`accessToken`, `refreshToken`, `expiresAt`) to `~/.hermes/.anthropic_oauth.json`. In affected builds, the write was performed with `Path.write_text(...)` and no subsequent permission restriction. Under the default `umask 022` used by macOS and most Linux distributions, the credential file was created with mode `0644` — group - and world-readable.

Every other credential writer in the codebase (`agent/anthropic_adapter.py`, `hermes_cli/auth.py`) explicitly restricts permissions after writing, so the dashboard path was an outlier that left a long-lived Anthropic refresh token readable by other local accounts and by any process running under a different UID with read access to the operator's home directory.

The issue has been remediated on `main`. The credential write is now atomic and private from creation, and the same pattern has been applied uniformly across the other OAuth writers.

Autori:

- Gaetano Zappulla: CISO

Affected versions

- `hermes-agent` builds whose dashboard PKCE writer (`_save_anthropic_oauth_creds` in `hermes_cli/web_server.py`) used `Path.write_text(...)` without a permission restriction.
- Confirmed against hermes-agent 0.10.0 (source tarball `hermes-agent-main`, extracted 2026-04-23). The pre-fix range is observable in the git history of `hermes_cli/web_server.py`.

Patched versions

- Fixed on `main`. Users should update to the latest `main` / current release, in which `_save_anthropic_oauth_creds` writes the credential atomically with `0600` permissions from creation time.

Impact

Trust boundary crossed: confidentiality of credentials at rest.

A local user other than the operator — or any process running under a different UID with read access to the operator's home (for example, a misconfigured backup agent or a shared CI node) — could read the persisted Anthropic refresh token. With that token an attacker could mint access tokens against the provider account and continue to do so until the token was rotated/revoked.

CVSS rationale (7.3 / High):

Metric	Value	Reason
Attack Vector	Local (L)	Requires read access to the operator's home on the same host.
Attack Complexity	Low (L)	No special conditions; the file is simply world-readable.
Privileges Required	Low (L)	A separate, unprivileged local account suffices.
User Interaction	None (N)	No operator action needed at exploit time.
Scope	Changed (C)	The token grants access to a distinct authority (the Anthropic API account).
Confidentiality	High (H)	Full disclosure of a long-lived refresh token.
Integrity	Low (L)	The token can be used to mutate provider-side state indirectly.
Availability	None (N)	No availability impact.

Technical detail

Root cause (pre-fix)

`hermes_cli/web_server.py → _save_anthropic_oauth_creds` serialized the PKCE payload and wrote it via `Path.write_text(...)` with no following `chmod`. The resulting file inherited the process umask. Under the common default `umask 022` the file mode was `0644`.

Correct patterns already existed elsewhere in the tree and were not applied here:

- `agent/anthropic_adapter.py:643-646 — cred_path.chmod(0o600)` after write.
- `hermes_cli/auth.py:1186-1192 — os.chmod(tmp_path, stat.S_IRUSR | stat.S_IWUSR)` on the temp file.

Reproduction (pre-fix)

1. Use a pre-fix build (or revert `_save_anthropic_oauth_creds` to `Path.write_text(…)`).
2. `umask 022 && hermes`, then open the dashboard OAuth flow and complete the Anthropic PKCE exchange.
3. Inspect the file mode:
   - Linux: `stat -c '%a %n' ~/.hermes/.anthropic_oauth.json → 644 …/.anthropic_oauth.json`
   - macOS: `stat -f '%A %N' ~/.hermes/.anthropic_oauth.json`
4. From any other local account: `cat ~<operator>/.hermes/.anthropic_oauth.json` discloses the refresh token.

Remediation (current `main`)

The credential is written to a per-process temporary file created with restrictive mode at creation time, then atomically moved into place:

```
def _save_anthropic_oauth_creds(access_token, refresh_token,
expires_at_ms):
    from agent.anthropic_adapter import _HERMES_OAUTH_FILE
    payload = {
        "accessToken": access_token,
        "refreshToken": refresh_token,
        "expiresAt": expires_at_ms,
    }
    _HERMES_OAUTH_FILE.parent.mkdir(parents=True, exist_ok=True)
    oauth_payload = json.dumps(payload, indent=2) + "\n"
    tmp_path = _HERMES_OAUTH_FILE.with_name(

f"{_HERMES_OAUTH_FILE.name}.tmp.{os.getpid()}.{secrets.token_hex(8)}"
    )
    try:
        fd = os.open(tmp_path, os.O_WRONLY | os.O_CREAT | os.O_TRUNC,
0o600)
        with os.fdopen(fd, "w", encoding="utf-8") as handle:
            handle.write(oauth_payload)
            handle.flush()
            os.fsync(handle.fileno())
        os.replace(tmp_path, _HERMES_OAUTH_FILE)
        os.chmod(_HERMES_OAUTH_FILE, 0o600)
        try:
            dir_fd = os.open(str(_HERMES_OAUTH_FILE.parent), os.O_RDONLY)
        except OSError:
            dir_fd = None
        if dir_fd is not None:
            try:
                os.fsync(dir_fd)
            finally:
                os.close(dir_fd)
    finally:
        try:
            if tmp_path.exists():
                tmp_path.unlink()
        except OSError:
            ...
```

Why this closes the issue:

- `os.open(..., 0o600)` sets the mode at creation time, independent of umask.
- `os.replace` is atomic on POSIX and Windows (same filesystem), so readers never observe an intermediate, wider-permission file — closing the `write_text` + post-hoc `chmod` TOCTOU window during which the file briefly inherited the process umask.
- The subsequent `os.chmod(..., 0o600)` is defensive against a pre-existing destination with wider permissions.
- The directory `fsync` makes the replace durable across an unclean shutdown.

The maintainer confirmed the same atomic, private-from-creation pattern is mirrored in the other OAuth writers – `agent/google_oauth.py` (PR #19673) and `tools/mcp_oauth.py` (PR #21148) — and closed the report as already-fixed on `main`.

Regression coverage

A regression test asserts the final file mode, e.g.:

```
assert stat.filemode(os.stat(p).st_mode) == '-rw-------'
```

Mitigation for existing deployments

On shared or multi-user hosts where a pre-fix build previously completed the dashboard PKCE flow:

1. Rotate the Anthropic refresh token, since it may already have been exposed.
2. Restrict any existing on-disk credential files:
3. `chmod 600 ~/.hermes/.anthropic_oauth.json`
4. `for f in ~/.hermes/profiles/*/.anthropic_oauth.json; do chmod 600 "$f"; done`
5. Update to the latest main / current release so future writes are private from creation.

Suggested hardening

Factor the atomic-`0600` write into a single shared helper (e.g. `agent/secret_file.py::write_secret_atomic`) and call it from every credential writer. The legacy `write_text + chmod` sequence still in use in some writers has a brief window during which the file is world-readable under the default umask; a uniform helper removes both that residual window and the risk of a future missed `chmod`.

Timeline

Date	Event
2026-04-23	Issue identified against the 0.10.0 source tarball.
2026-04-24	Reported privately via GitHub Security Advisories (GHSA-hp9q-8898-334v).
2026-04-24	Maintainer confirmed the issue was already remediated on `main`; advisory closed with credit.
(publication date)	Public disclosure.

Credits

- Vulnerability discovered, analyzed and reported by Gaetano Zappulla, CISO at Defence Tech SpA.
- Remediation and confirmation: NousResearch / hermes-agent maintainers.

References

- `SECURITY.md` (credential storage): https://github.com/NousResearch/hermes-agent/blob/main/SECURITY.md
- CWE-276: https://cwe.mitre.org/data/definitions/276.html
- Comparable correct writers: `agent/anthropic_adapter.py`, `hermes_cli/auth.py`
- Related uniform-pattern PRs: `agent/google_oauth.py` (#19673), `tools/mcp_oauth.py` (#21148)

t-defence

Next | Donexit | Foramil | Innodesi

Via Giacomo Peroni, 452 - 00131 Roma
tel. 06.45752720 - info@defencetech.it
www.t-defence.it

#TDefenceBusiness