nesquena / **hermes-webui** Public

Code | Issues 82 | Pull requests 4 | Actions | Projects | Security and quali

Commit 88dc8bb

3 people authored last week · ✓ 3 / 3 · Verified

```
fix: isolate profile .env secrets on switch (#351)

* fix: isolate profile .env secrets on switch

* fix: move direct os.environ set after _reload_dotenv to survive profile isolation

The profile env isolation in _reload_dotenv now clears previously tracked
env keys before re-reading .env. When apply_onboarding_setup set
os.environ BEFORE _reload_dotenv, the key was immediately cleared.
Move the belt-and-braces os.environ set to AFTER _reload_dotenv so
the API key survives regardless of profile tracking state.

Co-Authored-By: Claude Opus 4.6 (1M context) <noreply@anthropic.com>

---------

Co-authored-by: Nathan Esquenazi <nesquena@gmail.com>
Co-authored-by: Claude Opus 4.6 (1M context) <noreply@anthropic.com>
```

master (#351) · v0.50.133 ••• v0.50.12

1 parent 1fee123 commit 88dc8bb

3 files changed +91 -5 lines changed

Top

Filter files…

- api
  - onboarding.py
  - profiles.py
- tests
  - test_profile_env_isolation.py

3 files changed +91 -5 lines changed

Search within code

api/onboarding.py

```
@@ -479,9 +479,6 @@ def apply_onboarding_setup(body: dict) -> dict:
479 479
480 480         if api_key:
481 481             _write_env_file(env_path, {provider_meta["env_var"]: api_key})
482     -         # Belt-and-braces: set directly on os.environ so the value is visible to
483     -         # any code in the same process that reads it before the next request cycle.
484     -         os.environ[provider_meta["env_var"]] = api_key
485 482
486 483         # Reload the hermes_cli provider/config cache so the next streaming call
487 484         # picks up the new key without requiring a server restart.
@@ -491,6 +488,12 @@ def apply_onboarding_setup(body: dict) -> dict:
491 488         except Exception:
492 489             pass
493 490
    491 +     # Belt-and-braces: set directly on os.environ AFTER _reload_dotenv so the
    492 +     # value survives even if _reload_dotenv cleared it (e.g. when _write_env_file
    493 +     # wrote to disk but the profile isolation tracking hasn't seen it yet).
    494 +     if api_key:
    495 +         os.environ[provider_meta["env_var"]] = api_key
    496 +
494 497         try:
495 498             # hermes_cli may cache config at import time; ask it to reload if possible.
496 499             from hermes_cli.config import reload as _cli_reload
```

api/profiles.py

```
@@ -26,6 +26,7 @@
26 26  # ── Module state ─────────────────────────────────────────────────────────
27 27  _active_profile = 'default'
28 28  _profile_lock = threading.Lock()
    29 + _loaded_profile_env_keys: set[str] = set()
```

```
29  30
30  31      def _resolve_base_hermes_home() -> Path:
31  32          """Return the BASE ~/.hermes directory — the root that contains profiles/.
    ↕       @@ -120,11 +121,24 @@ def _set_hermes_home(home: Path):
120 121
121 122
122 123     def _reload_dotenv(home: Path):
123         -       """Load .env from the profile dir into os.environ (additive)."""
        124 +       """Load .env from the profile dir into os.environ with profile isolation.
        125 +
        126 +       Clears env vars that were loaded from the previously active profile before
        127 +       applying the current profile's .env. This prevents API keys and other
        128 +       profile-scoped secrets from leaking across profile switches.
        129 +       """
        130 +       global _loaded_profile_env_keys
        131 +
        132 +       # Remove keys loaded from the previous profile first.
        133 +       for key in list(_loaded_profile_env_keys):
        134 +           os.environ.pop(key, None)
        135 +       _loaded_profile_env_keys = set()
        136 +
124 137         env_path = home / '.env'
125 138         if not env_path.exists():
126 139             return
127 140         try:
        141 +           loaded_keys: set[str] = set()
128 142             for line in env_path.read_text().splitlines():
129 143                 line = line.strip()
130 144                 if line and not line.startswith('#') and '=' in line:
    ↕       @@ -133,8 +147,10 @@ def _reload_dotenv(home: Path):
133 147                     v = v.strip().strip('"').strip("'")
134 148                     if k and v:
135 149                         os.environ[k] = v
        150 +                       loaded_keys.add(k)
        151 +           _loaded_profile_env_keys = loaded_keys
136 152         except Exception:
137         -           pass
        153 +           _loaded_profile_env_keys = set()
138 154
```

```
139  155
140  156     def init_profile_state() -> None:
```

tests/test_profile_env_isolation.py

```
     @@ -0,0 +1,67 @@
1    + import importlib
2    + import os
3    + import sys
4    + from pathlib import Path
5    +
6    +
7    + def test_profile_switch_clears_previous_profile_env_vars(monkeypatch, tmp_path):
8    +     base = tmp_path / ".hermes"
9    +     (base / "profiles" / "p1").mkdir(parents=True)
10   +     (base / "profiles" / "p2").mkdir(parents=True)
11   +     (base / "profiles" / "p1" / ".env").write_text(
12   +         "OPENAI_API_KEY=secret-from-p1\nCUSTOM_TOKEN=token-from-p1\n",
13   +         encoding="utf-8",
14   +     )
15   +
16   +     monkeypatch.setenv("HERMES_BASE_HOME", str(base))
17   +     monkeypatch.delenv("HERMES_HOME", raising=False)
18   +     monkeypatch.delenv("OPENAI_API_KEY", raising=False)
19   +     monkeypatch.delenv("CUSTOM_TOKEN", raising=False)
20   +
21   +     sys.modules.pop("api.profiles", None)
22   +     profiles = importlib.import_module("api.profiles")
23   +     profiles = importlib.reload(profiles)
24   +
25   +     profiles.init_profile_state()
26   +     profiles.switch_profile("p1")
27   +     assert os.environ.get("OPENAI_API_KEY") == "secret-from-p1"
28   +     assert os.environ.get("CUSTOM_TOKEN") == "token-from-p1"
29   +
30   +     profiles.switch_profile("p2")
31   +     assert os.environ.get("OPENAI_API_KEY") is None
32   +     assert os.environ.get("CUSTOM_TOKEN") is None
33   +     assert profiles.get_active_profile_name() == "p2"
34   +
```

```
35 +
36 + def test_profile_switch_replaces_overlapping_keys(monkeypatch, tmp_path):
37 +     base = tmp_path / ".hermes"
38 +     (base / "profiles" / "p1").mkdir(parents=True)
39 +     (base / "profiles" / "p2").mkdir(parents=True)
40 +     (base / "profiles" / "p1" / ".env").write_text(
41 +         "OPENAI_API_KEY=secret-from-p1\nONLY_P1=one\n",
42 +         encoding="utf-8",
43 +     )
44 +     (base / "profiles" / "p2" / ".env").write_text(
45 +         "OPENAI_API_KEY=secret-from-p2\nONLY_P2=two\n",
46 +         encoding="utf-8",
47 +     )
48 +
49 +     monkeypatch.setenv("HERMES_BASE_HOME", str(base))
50 +     monkeypatch.delenv("HERMES_HOME", raising=False)
51 +     monkeypatch.delenv("OPENAI_API_KEY", raising=False)
52 +     monkeypatch.delenv("ONLY_P1", raising=False)
53 +     monkeypatch.delenv("ONLY_P2", raising=False)
54 +
55 +     sys.modules.pop("api.profiles", None)
56 +     profiles = importlib.import_module("api.profiles")
57 +     profiles = importlib.reload(profiles)
58 +
59 +     profiles.init_profile_state()
60 +     profiles.switch_profile("p1")
61 +     assert os.environ.get("OPENAI_API_KEY") == "secret-from-p1"
62 +     assert os.environ.get("ONLY_P1") == "one"
63 +
64 +     profiles.switch_profile("p2")
65 +     assert os.environ.get("OPENAI_API_KEY") == "secret-from-p2"
66 +     assert os.environ.get("ONLY_P1") is None
67 +     assert os.environ.get("ONLY_P2") == "two"
```

Comments 0

Please sign in to comment.