fix: validate bootstrap agent names before filesystem writes #2274

Merged **WillemJiang** merged 2 commits into `bytedance:main` from `Hinotoi-agent:fix/bootstrap-agent…` 2 days ago

Hinotoi-agent commented 2 days ago • edited Contributor

Summary

This PR hardens bootstrap-mode custom-agent creation by enforcing the existing backend agent-name validation before any filesystem path construction occurs.

- bootstrap-mode `make_lead_agent()` now validates `configurable["agent_name"]` instead of bypassing validation when `is_bootstrap=true`
- `setup_agent()` now validates `runtime.context["agent_name"]` before building `agent_dir` or writing `config.yaml` / `SOUL.md`
- the cleanup path in `setup_agent()` is hardened so invalid-name failures do not dereference an uninitialized `agent_dir`
- regression tests now cover traversal and absolute-path agent names in both the bootstrap entry path and the filesystem-writing tool path

Security issues covered

Issue	Impact	Severity
Bootstrap-mode agent-name validation bypass	attacker-controlled invalid `agent_name` values can reach custom-agent filesystem paths during bootstrap flows	High
Unvalidated `setup_agent` filesystem writes	traversal or absolute-path `agent_name` values can redirect `config.yaml` / `SOUL.md` writes outside the intended agents directory, subject to runtime filesystem permissions	High

Before this PR

- bootstrap mode skipped `load_agent_config()`, which was the only backend path enforcing the `^[A-Za-z0-9-]+$` custom-agent name pattern
- `setup_agent()` trusted `runtime.context["agent_name"]` and used it directly in `paths.agent_dir(agent_name)` before writing files
- absolute-path and traversal-style agent names were not rejected before filesystem path construction
- regression tests did not cover invalid bootstrap-mode agent names or invalid `setup_agent()` runtime agent names

After this PR

- bootstrap-mode lead-agent creation rejects invalid custom agent names before any bootstrap agent is created
- `setup_agent()` rejects invalid runtime agent names before directory creation or file writes
- cleanup logic no longer assumes an initialized `agent_dir` after early validation failures
- regression tests lock in both the bootstrap entry validation and the tool-level path-safety validation

Why this matters

- the broken boundary was: request/runtime-controlled `agent_name` -> filesystem path construction -> file writes
- if a bootstrap flow could be reached with an attacker-controlled invalid agent name, the pre-patch path could redirect writes outside the intended `agents/` directory
- even when constrained by runtime write permissions, this is the wrong trust boundary for custom-agent creation and should fail before any path is built

Attack flow

```
attacker-controlled bootstrap agent_name
    -> bootstrap-mode make_lead_agent() skips load_agent_config() validation
        -> setup_agent() uses runtime.context.agent_name in paths.agent_dir()
            -> config.yaml / SOUL.md write outside intended agents directory
```

Affected code

Issue	Files
Bootstrap-mode	`backend/packages/harness/deerflow/agents/lead_agent/agent.py`, `backend/packages/harness/deerflow/config/agents_config.py`

Issue	Files
validation bypass	
Unvalidated `setup_agent` writes	`backend/packages/harness/deerflow/tools/builtins/setup_agent_tool.py`, `backend/packages/harness/deerflow/config/agents_config.py`, `backend/packages/harness/deerflow/config/paths.py`

Root cause

Issue 1: bootstrap-mode validation bypass

- `make_lead_agent()` only called `load_agent_config(agent_name)` when `is_bootstrap` was false
- the existing name-validation rule lived inside `load_agent_config()`, so bootstrap mode accidentally bypassed the only backend validation gate

Issue 2: unvalidated filesystem writes in `setup_agent`

- `setup_agent()` read `runtime.context["agent_name"]` and passed it directly into `paths.agent_dir(agent_name)` before writing files
- this allowed request/runtime-controlled path components to influence filesystem writes without prior validation

CVSS assessment

Issue	CVSS v3.1	Vector
Bootstrap-mode agent-name validation bypass	8.1 High	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H`
Unvalidated `setup_agent` filesystem writes	8.1 High	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H`

Rationale:

- the validated primitive is attacker-influenced file write outside the intended custom-agent directory boundary
- confidentiality impact is not the primary claim here, but integrity and availability can be materially affected if writable target paths are reachable in deployment

Safe reproduction steps

1. Bootstrap-mode validation bypass

1. Prepare a bootstrap-mode `RunnableConfig` with `is_bootstrap=true` and an invalid `agent_name` such as `../../../tmp/evil`.
2. Call `make_lead_agent()` with that configuration.
3. Observe that the pre-patch code accepted the invalid name into the bootstrap path instead of rejecting it early.

2. Unvalidated `setup_agent` writes

1. Invoke `setup_agent()` with a runtime context containing an invalid `agent_name` such as `../../../tmp/evil` or an absolute path.
2. Let `setup_agent()` derive `agent_dir = paths.agent_dir(agent_name)`.
3. Observe that the pre-patch code proceeded toward directory creation and `config.yaml` / `SOUL.md` writes using the invalid path-derived location.

Prefer safe local test harnesses; no destructive payload is required.

Expected vulnerable behavior

- bootstrap flows should never accept traversal or absolute-path custom agent names
- `setup_agent()` should never build filesystem paths from unvalidated runtime agent names
- pre-patch behavior allowed invalid names to reach filesystem path construction instead of failing immediately

Changes in this PR

- add shared `validate_agent_name()` helper in `agents_config.py`
- reuse that helper inside `load_agent_config()` so validation remains centralized
- validate `configurable["agent_name"]` in `make_lead_agent()` even when `is_bootstrap=true`
- validate `runtime.context["agent_name"]` in `setup_agent()` before any filesystem path use
- guard cleanup so invalid-name failures do not access an uninitialized `agent_dir`
- add regression tests for invalid bootstrap agent names and invalid `setup_agent()` runtime agent names

Files changed

Category	Files	W
Shared validation	`backend/packages/harness/deerflow/config/agents_config.py`	added `valida` helper `load_a`
Bootstrap hardening	`backend/packages/harness/deerflow/agents/lead_agent/agent.py`	validat before

Category	Files	W
		lead-ag
Tool hardening	`backend/packages/harness/deerflow/tools/builtins/setup_agent_tool.py`	validat `agent_` constru cleanu
Regression tests	`backend/tests/test_lead_agent_model_resolution.py`, `backend/tests/test_setup_agent_tool.py`	add tra absolu covera

Maintainer impact

- this is a narrow backend hardening patch
- normal valid custom-agent creation flows remain unchanged
- frontend and unrelated runtime systems are untouched
- centralizing validation into a shared helper reduces the chance of future bootstrap-vs-non-bootstrap drift

Suggested fix rationale

- the right boundary is to reject invalid custom agent names before any filesystem path is constructed
- applying the same validation rule in both bootstrap entry and tool-level write paths makes the behavior consistent and durable
- the regression tests are sufficient to prevent the exact bootstrap/path-safety drift that allowed this gap

Type of change

- [x] Security fix
- [x] Tests
- [ ] Documentation update
- [ ] Refactor with no behavior change

Test plan

- [x] validate bootstrap-mode invalid agent names are rejected
- [x] validate `setup_agent()` rejects traversal-style runtime agent names before writing
- [x] validate `setup_agent()` rejects absolute-path runtime agent names before writing
- [ ] full backend test suite

Executed with:

- `cd backend && uv sync`
- `cd backend && PYTHONPATH=. uv run pytest tests/test_lead_agent_model_resolution.py tests/test_setup_agent_tool.py -q`

If something was not run:

- the full backend test suite was not run for this PR body update; only the targeted regression tests above were executed for this patch

Disclosure notes

- claims here are intentionally bounded to the validated bootstrap/path-construction and filesystem-write code paths
- this PR does not claim arbitrary write success to every path, only that invalid names reached filesystem path construction and write attempts outside the intended agents directory boundary subject to runtime permissions
- no unrelated files were changed outside the minimal validation and regression-test scope

`fix: validate bootstrap agent names before filesystem writes` ✓ `e998921`

WillemJiang requested a review from **Copilot** 2 days ago

Copilot started reviewing on behalf of **WillemJiang** 2 days ago View session

Copilot AI reviewed 2 days ago

View reviewed changes

Copilot AI left a comment Contributor

Pull request overview

Hardens custom-agent creation paths by ensuring `agent_name` is validated (against the existing backend pattern) before any filesystem path construction or writes occur, including in bootstrap mode.

Changes:

- Added a shared `validate_agent_name()` helper and reused it from `load_agent_config()`.
- Validated `configurable["agent_name"]` in `make_lead_agent()` even when `is_bootstrap=true`.

- Validated `runtime.context["agent_name"]` in `setup_agent()` before directory creation/writes and hardened cleanup for early failures.

Reviewed changes

Copilot reviewed 5 out of 5 changed files in this pull request and generated 2 comments.

▶ Show a summary per file

`backend/packages/harness/deerflow/config/agents_config.py` Outdated Show resolved

`backend/tests/test_setup_agent_tool.py` Show resolved

`fix: tighten bootstrap agent-name validation` `7594153`

Hinotoi-agent commented 2 days ago — Contributor, Author

Addressed the two Copilot review findings in follow-up commit `7594153`.

What changed:

- tightened `validate_agent_name()` to reject non-string inputs and use `fullmatch()` instead of `match()`
- updated the traversal regression test to assert against a controlled outside directory under the test temp hierarchy rather than `/tmp/evil`

Revalidated with:

- `cd backend && PYTHONPATH=. uv run pytest tests/test_lead_agent_model_resolution.py tests/test_setup_agent_tool.py -q`
- `10 passed`

I also resolved the associated review threads after pushing the fix.

WillemJiang approved these changes 2 days ago

View reviewed changes

WillemJiang merged commit **`2176b2b`** into `bytedance:main` 2 days ago

5 checks passed

View details

Reviewers

Copilot

WillemJiang

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

3 participants