

Agentenplattform: Laufzeitumgebung und Sicherheitsmodell

Interne Plattform, auf der LLM-Agenten Geschäftsszenarien unter einer ausführbaren Sicherheitsrichtlinie abarbeiten.

Zeitraum: 07/2026 – heute

Rolle: Entwickler der Laufzeitumgebung und des Sicherheitsmoduls im Plattformteam

Quellcode nicht öffentlich

AUSGANGSLAGE

Agentenszenarien laufen über Stunden oder Tage und warten zwischendurch auf die Freigabe eines Menschen. Ein Neustart des Workers darf dabei keinen Fortschritt kosten. Gleichzeitig muss für jeden Aufruf nachvollziehbar sein, wer ihn ausgelöst hat, in welcher Zone er läuft und welche Regel ihn erlaubt hat. Bis dahin lag das Berechtigungsmodell nur als Dokument vor.

LÖSUNG

- Ein Szenario wird als Snapshot beschrieben und in einen LangGraph-Graphen kompiliert, den Temporal ausführt. Jeder Knoten läuft als eigene Activity, damit Abstürze und Wartezeiten auf Freigaben den Ablauf nicht abbrechen.
- Die Kompilierung ist eine reine Funktion: Derselbe Snapshot ergibt immer denselben Graphen. Sonst bricht Temporal beim Wiederabspielen der Historie mit einem Nichtdeterminismus-Fehler ab.
- In Graphzustand und Historie liegen nur Verweise auf Artefakte, keine Nutzdaten.
- Fehler sind in drei Klassen getrennt. Vorübergehende Fehler werden wiederholt, fatale Fehler und Ablehnungen der Richtlinie nicht, fachliche Ergebnisse wie eine Datenabweichung sind ein normaler Übergang im Graphen.
- Das Sicherheitsmodul bildet Zonen, Rollen, Datenklassen, Autonomiestufen und absolute Verbote als Python-Code ab. `decide()` prüft jeden Aufruf in sieben Schritten in fester Reihenfolge und gibt die zugrunde liegende Regel mit zurück.
- Java-Dienste fragen das Modul über HTTP mit mTLS ab. Ohne Eintrag im Append-only-Journal wird keine Entscheidung ausgegeben.

ERGEBNISSE

Stand 09/2026, belegt durch Code und Tests

- Determinismus der Kompilierung per Test über Prozessgrenzen geprüft: gleicher Snapshot, gleicher Hash in getrennten Prozessen.
- Laufzeitumgebung: 176 Tests in der Standalone-Version, 58 Tests im Plattform-Repository.
- Sicherheitsmodell im Code: 8 Zonen mit 10 erlaubten Übergängen, Rollenmatrix 12×9 , 22 Aktionsarten, 10 absolute Verbote, Autonomiestufen L0–L3 mit Notabschaltung.
- Sicherheitsmodul: 145 Tests, Typprüfung mit mypy im Strict-Modus.
- Nächster Schritt: Anbindung des Sicherheitsmoduls an den Core-Dienst der Plattform.

TECHNOLOGIEN

Python · Temporal · LangGraph · gRPC · HTTP mit mTLS · Policy as Code · mypy strict · pytest