

Drei Sicherheitsregeln, die dein Agent nicht umgehen kann

AWS hat mit Dogwood eine Open-Source-Schicht veröffentlicht, die jeden Tool-Aufruf deines KI-Agenten gegen feste Regeln prüft — bevor er ausgeführt wird. Hier sind drei einsatzfertige Regeln nach den offiziellen Beispielen: Zahlungs-Freigabe, Datenschutz-Sequenz und Session-Fenster.

3 REGELN ZUM KOPIEREN

OFFIZIELLE SYNTAX

AWS OPEN SOURCE

APACHE-2.0

QUELLE INKLUSIVE

Worum es geht: Dogwood ist eine Runtime-Verification-Schicht von AWS für KI-Agenten. Deine Regeln stehen als Policies **zwischen Agent und Tools**: Erlaubtes läuft ungebremst durch, Kritisches wird vor der Ausführung gestoppt oder wartet auf deine Freigabe. Die Sprache baut auf Cedar auf und kann zusätzlich Abläufe prüfen — also nicht nur „darf er das?“, sondern auch „darf er das *jetzt, nach dem, was vorher passiert ist?*“.

1 So greifen die Regeln

Eine Policy besteht aus **permit** (erlauben) oder **forbid** (verbieten) plus Bedingungen. Das Besondere sind die zeitlichen Bedingungen: **when formerly within 1h { ... }** heißt „nur, wenn in der letzten Stunde vorher Folgendes passiert ist“. Damit bildest du genau die Fälle ab, an denen reine Prompt-Regeln scheitern.

Der Ablauf mit dem Referenz-Interpreter: **dogwood validate** prüft deine Policy-Datei, **dogwood lower** übersetzt sie für die Laufzeit, **dogwood replay** spielt aufgezeichnete Agenten-Läufe gegen deine Regeln ab.

2 Regel 1 — Zahlung nur mit frischer Freigabe

Zahlungen bleiben unter einer Betragsgrenze **und** brauchen eine Freigabe, die höchstens eine Stunde alt ist. Ohne beides wird der Tool-Aufruf gar nicht erst ausgeführt.

```
// Zahlung: unter 1.000 und Approve-Aktion in der letzten Stunde
permit(principal, action == Action::"SendPayment", resource)
when { context.input.amount < 1000 }
when formerly within 1h {
  Action::"Approve"::request{ approver: context.input.approver }
};
```

Anpassen: Betragsgrenze, Zeitfenster und den Action-Namen deines Zahlungs-Tools.

3 Regel 2 — nach vertraulichen Daten kein Versand nach außen

Die Regel aus dem Reel: Sobald der Agent vertrauliche Daten gelesen hat, ist externer Versand für die nächsten 24 Stunden blockiert — egal, wie überzeugend ein Prompt ihn zum Gegenteil drängt.

```
// Sequenz-Schutz: erst Vertrauliches gelesen → kein SendExternal
forbid(principal, action == Action::"SendExternal", resource)
when temporal {
  formerly within 24h
    Action::"ReadConfidential"::request{ input.user: context.input.user }
};
```

Anpassen: die Action-Namen auf dein Lese- und dein Versand-Tool mappen.

4 Regel 3 — kritische Aktionen nur in aktiver Session

Wörtlich aus der offiziellen Doku: Lese-Zugriffe funktionieren nur innerhalb einer Stunde nach dem Login desselben Nutzers. Dasselbe Muster passt auf jede kritische Aktion, die an eine frische Anmeldung oder Freischaltung gebunden sein soll.

```
@id("read_after_login")
permit (
  principal,
  action == Drupe::Action::"Read",
  resource
)
when temporal {
  formerly within 1h Drupe::Action::"Login"::request{ input.user: context.input.user }
};
```

VOR DEM EINSATZ

Regel 1 und 3 folgen den Beispielen aus der offiziellen Doku, Regel 2 überträgt das dokumentierte Sequenz-Muster auf den Versand-Fall. Lass vor dem Produktiv-Einsatz jede Datei durch **dogwood validate** laufen und teste sie mit **dogwood replay** gegen echte Agenten-Läufe. Und: Dogwood regelt Tool-Aufrufe und Abläufe — es ist ein Policy-Wächter, kein Hardware-Notausschalter.

5 Quelle & Einstieg

- **Offizieller AWS-Artikel:** aws.amazon.com/blogs/opensource/introducing-dogwood-runtime-verification-for-ai-agents/
- **Repo mit Doku und Referenz-Interpreter:** github.com/dogwood-policy/dogwood (Apache-2.0)
- **Einbinden** (Rust/Cargo): `dogwood-language = { git = "https://github.com/dogwood-policy/dogwood.git" }`

■ // AUF EINEN BLICK

Die Kurzfassung

Vier Punkte, die das Prinzip zusammenhalten.

- ✓ Policies stehen **vor dem Tool-Aufruf** — geprüft wird, bevor etwas passiert
- ✓ **Zeitliche Bedingungen** (formerly within) prüfen Abläufe, nicht nur einzelne Rechte
- ✓ Der Agent bleibt **autonom** — nur die kritischen Aufrufe werden gestoppt oder warten auf Freigabe
- ✓ Open Source unter **Apache-2.0**, Validierung und Replay-Tests im Repo enthalten

■ // BLEIB DRAN

KI-Agenten-Releases, getestet statt gehypt.

Ich schaue mir solche Veröffentlichungen an, bevor ich sie empfehle — mit den echten Voraussetzungen und Grenzen. Folg mir für die nächste Einordnung.

@ki.borish

Fakten verifiziert am 29.08.2026 aus aws.amazon.com/blogs/opensource (Dogwood-Ankündigung) und dem README von github.com/dogwood-policy/dogwood. Dogwood ist junge Open-Source-Software; Syntax und Befehle können sich mit neuen Releases ändern. Prüfe Policies vor dem Produktiv-Einsatz mit `dogwood validate` gegen die aktuelle Doku.