



id: SI_003

model: mistralai/mistral-large-2512

seeds:

04_risiko: sårbarhet

03_relasjon: tillit

05_handling: implementer

01_perspektiv: bruker

02_tidhorisont: langsiktig

Instans SI_003

Seeds: bruker (menneskelig forankring) | langsiktig (systemisk motstandskraft) | tillit (dynamisk tillitsmodell) | sårbarhet (bevisst eksponering) | implementer (inkrementell materialisering)

Kritisk perspektiv: Agentisk KI som sosiotechnisk tillitsøkosystem

Agentisk KI er ikke et teknisk problem, men et **langsiktig tillitsforhandlingsprosjekt** mellom mennesker, kode og systemer. Den største sårbarheten ligger ikke i teknologien, men i at vi undervurderer hvordan autonome agenter forandrer brukerens rolle – fra operatør til tilsynshaver. Dette krever en arkitektur som **eksponerer sårbarhet bevisst** for å bygge tillit, ikke skjuler den bak kompleksitet.

Arkitekturprinsipper: “Tillitsrammer” over kontrollmekanismer

1. Zero Trust som sosial kontrakt

- IAM må gå fra statisk RBAC til **dynamiske tillitsnivåer** (f.eks. “trust tiers” basert på agentens historikk, risikonivå og brukerfeedback).
- Eksempel: En agent som har kjørt 1000 feilfrie transaksjoner får “tillitsscore” +10, som gir midlertidig utvidet tilgang (men med runtime-policyer som krever menneskelig godkjenning for nye handlinger).
- Implementer: Bruk **Open Policy Agent (OPA)** med tilpassede regler som evaluerer både tekniske og sosiale signaler (f.eks. brukerklager, avvik fra historisk adferd).



2. Sårbarhetsdrevet isolasjon

- Ikke isoler agenter for å kontrollere dem, men for å **gjøre feil synlige og reversible**.
- Eksempel: Hver agent kjører i en **ephemeral container** med:
 - **Tidsbegrensede credentials** (rotasjon hver 15 min)
 - **Rate-limited API-kall** (f.eks. maks 5 kall/sekund mot legacy-systemer)
 - **Automatisk “circuit breaker”** som stopper agenten hvis den utløser >3 feilhendelser på rad
- Implementer: **Kyverno** for Kubernetes-policyer + **Linkerd** for service mesh-throttling.

3. Audit som tillitsbyggende ritual

- Logging må fange **ikke bare handlinger, men beslutningsprosessen** – inkludert prompts, token-bruk og hvorfor agenten valgte en handling.
- Eksempel: Hver agent-transaksjon logges med:
 - **Prompt-historikk** (inkl. brukerens opprinnelige instruks)
 - **Policy-evalueringer** (hvorfor OPA tillot handlingen)
 - **Sideeffekter** (f.eks. “endret 3 rader i database X, triggerte webhook Y”)
- Implementer: **OpenTelemetry** + **Loki** for konsolidert logging, med **automatisk varsling** til brukeren ved uvanlige mønstre.

IAM: Dynamisk tillit > statisk tilgang

- **Service accounts med “tillitsscore”:**
 - Hver agent får en **temporær identitet** (f.eks. SPIFFE/SPIRE) med credentials som roteres hver 5. minutt.
 - Tilgang gis basert på **kontekst** (tid, sted, risikonivå) + **historisk adferd** (f.eks. “agenten har aldri misbrukt API Z før”).
- Eksempel: En agent som skal oppdatere en pasientjournal får:
 - **15-minutters token** med scope `read:pasientdata` + `write:journal_notater`
 - **Menneskelig godkjenning** kreves for å eskalere til `write:medisinering`



Blindsoner: Emergent sårbarhet

1. Latent coupling i legacy-systemer

- Risiko: Agenter kan utløse kjedereaksjoner i systemer med **implisitte avhengigheter** (f.eks. en endring i API A trigger en batch-jobb i system B som ingen visste om).
- Løsning: **Chaos engineering for agenter** – test med vilkårlige input for å avdekke skjulte avhengigheter før produksjon.

2. Dataens sårbarhet

- Risiko: Agenter som opererer på **dårlig kvalitet eller inkonsistente data** kan forsterke feil (f.eks. en agent som automatisk korrigerer “feil” adressefelt basert på utdatert register).
- Løsning: **Data-kvalitets-sensorer** som stopper agenten hvis den møter data utenfor forventede terskler.

3. Emergent adferd

- Risiko: Agenter som optimaliserer for lokale mål kan skape systemiske problemer (f.eks. en agent som bestiller for mange varer for å unngå “out of stock”, og skaper overskudd i lageret).
- Løsning: **Globale “guardrails”** (f.eks. maks 10% endring i lagerbeholdning per dag) + **automatisk eskalering** til menneske ved avvik.

Neste steg (6-9 måneder)

1. Bygg en “tillitslab” med 3 agenter

- Velg **et lav-risiko domene** (f.eks. intern dokumenthåndtering) og implementer:
 - OPA for runtime-policyer
 - Ephemeral credentials (SPIFFE/SPIRE)
 - Audit-pipeline (OpenTelemetry + Loki)
- Mål: Bevis at agenter kan operere **sikkert** og at brukerne **forstår** hva som skjer.

2. Kjør “sårbarhetsøvelser”

- Test med **fiendtlige agenter** (f.eks. en agent som prøver å eskalere tilgang) og **dårlige data** (f.eks. korrupte API-responser).
- Mål: Identifiser hvor tillitsmodellen bryter sammen før produksjon.



3. Designer en “tillitsdashboard” for brukere

- Vis **i sanntid**:
 - Hvilke handlinger agenten har utført
 - Hvilke policyer som ble evaluert
 - Hvordan brukeren kan overstyre eller stoppe agenten
 - Mål: Gjør agentens adferd **gjennomsiktig** for å bygge langsiktig tillit.
-

SI_003s dissens: De fleste fokuserer på kontroll – vi fokuserer på **sårbarhet som tillitsbygger**. Agentisk KI vil feile; spørsmålet er om systemet er designet for å **gjøre feilene synlige, reversible og lærerike**.