



id: SI_005

model: mistralai/mistral-large-2512

seeds:

04_risiko: resiliens

03_relasjon: makt

05_handling: utforsk

01_perspektiv: operasjonell

02_tidhorisont: langsiktig

Instans SI_005

Seeds: operasjonell (hverdagsstyring) | langsiktig (systemisk motstandskraft) | makt (kontroll og eskalering) | resiliens (tilbakevendingsevne) | utforsk (kontrollert eksperimentering)

Operasjonell makt med langsiktig resiliens: En motgift mot agentisk kaos

Agentisk KI er ikke et teknologisk problem, men et **maktfordelingsproblem** – hvem eller hva skal ha autonomi til å endre systemtilstander, og hvordan sikrer vi at denne makten ikke korrupperer eller kollapser under egen vekt? I et landskap med legacy-systemer, uforutsigbare API-er og begrensede ressurser, må vi **operasjonelt begrense makt mens vi langsiktig bygger resiliens** – ikke motsatt.

Arkitekturprinsipper: Isolasjon, observasjon, kompensasjon

1. Maktbegrensning via “operasjonelle celler”

- Del systemet inn i **domene-spesifikke celler** (f.eks. “lønn”, “pasientjournal”, “eksterne API-er”) med strenge grensekontroller.
- Eksempel: En agent i “lønn”-cellen kan kun kalle et spesifikt `/calculate` -endpoint, og aldri skrive direkte til database. Bruk **OPA** (Open Policy Agent) som policy-motor for runtime-sjekk av hver handling.
- **Resiliens-mekanisme:** Hver celle har en “kill switch” som aktiveres ved uforutsette mønstre (f.eks. 100 kall/sekund til samme endepunkt).



2. Langsiktig sporbarhet: Execution Graph som “DNA”

- Modellér agentens handlinger som en **dynamisk graf** hvor hver node er en handling (API-kall, datamodifikasjon) og hver kant er en avhengighet.
- Eksempel: Graf-node for `POST /invoice` inkluderer input-data, prompt, agent-ID, timestamp, og en **kryptografisk hash** av forrige tilstand. Lagres i en **immutable ledger** (f.eks. AWS QLDB eller en lokal PostgreSQL med WAL-logging).
- **Utforskningsverktøy**: Bruk grafvisualisering (f.eks. Neo4j) for å spore emergent behavior – f.eks. uventede kjedereaksjoner mellom “lønn” og “eksterne API-er”.

3. IAM for agenter: Dynamisk makt, midlertidig tillit

- **Ephemeral credentials** med levetid på 5–30 minutter, generert via **HashiCorp Vault** eller **AWS STS**.
- **Token-scoping**: Hver agent får et JWT med claims som spesifiserer:
 - Tillatte handlinger (f.eks. `["read:employee", "write:invoice"]`).
 - Tidsvindu (`exp` + `nbf`).
 - **Kontekstuelle begrensninger** (f.eks. `max_amount: 10000` for betalinger).
- **Resiliens**: Automatisk nøkkelrotasjon hver 24. time, med manuell override for kritiske agenter.

4. API-sikkerhet: Behandle eksterne kall som fiendtlige

- **Circuit breakers** (f.eks. **Hystrix** eller **Resilience4j**) som stopper agenter etter 3 feilede kall til samme endepunkt.
- **Rate-limiting per agent**: Maks 10 kall/minutt til eksterne API-er, med **eksponentiell backoff** ved feil.
- **Kontraktshåndheving**: Bruk **Pact** eller **OpenAPI-specs** for å validere at agenten kun sender forventede payloads. Eksempel: Hvis en agent sender `{"amount": "1000000"}` til en betalings-API som forventer `{"amount": 1000}`, blokkeres kallet.

5. Audit-pipeline: Rekonstruer agentens “tankeprosess”

- **Logging-standard**: Hver handling logges med:
 - **Prompt** (rå input til agenten).
 - **Beslutningsgrunnlag** (f.eks. “Valgte å endre faktura pga. avvik i regel X”).
 - **Handlingskjedet** (f.eks. `GET /employee → POST /invoice → PATCH /status`).



■ **Sideeffekter** (f.eks. “Endret `employee.salary` fra 50000 til 55000”).

- **Lagring**: Send loggene til **Elasticsearch** (for søk) + **S3 Glacier** (for arkivering). Bruk **Fluentd** for å konsolidere logger fra distribuerte systemer.
- **Resiliens**: Implementer **automatisk rollback** ved avvik. Eksempel: Hvis en agent endrer 100 fakturaer i løpet av 1 minutt, trigger systemet en `compensating action` som nullstiller endringene.

Blindsoner og worst-case-scenarier

1. **Latent coupling**: Agenten oppdager en “snarvei” mellom systemer som ikke er dokumentert (f.eks. at `POST /invoice` også oppdaterer `employee.last_activity`). Løsning: Bruk **chaos engineering** (f.eks. **Gremlin**) for å teste uforutsette koblinger før produksjon.
2. **Emergent behavior**: Agenten lærer å utnytte svakheter i legacy-systemer (f.eks. at et gammelt system godtar `NULL` -verdier som “godkjent”). Løsning: Implementer **statistiske anomalideteksjon** (f.eks. **Isolation Forest**) på agentens handlinger.
3. **Integritetsrisiko**: Agenten får tilgang til et system med høy risiko (f.eks. kjernejournaler). Løsning: Krev **manuell godkjenning** for alle handlinger i slike systemer, og logg godkjennerens identitet.

Neste steg (konkrete handlinger)

1. **Bygg en “policy-sandkasse” med OPA**
 - Definer **5 kritiske regler** for agentene (f.eks. “Ingen direkte database-skriv”, “Maks 1000 NOK per transaksjon”) og implementer dem i OPA.
 - Tidsramme: 4 uker (inkl. testing mot et isolert miljø).
2. **Pilot med en “low-risk” agent**
 - Velg en agent som kun leser data (f.eks. en “rapportgenerator”) og kjør den i produksjon med **full logging og manuell review** av alle handlinger.
 - Tidsramme: 6 uker (inkl. evaluering av emergent behavior).
3. **Implementer ephemeral credentials via Vault**
 - Konfigurer HashiCorp Vault til å utstede **tidsbegrensede tokens** for agentene, med automatisk rotasjon.
 - Tidsramme: 3 uker (inkl. integrasjon med IAM-plattformen).



Avsluttende perspektiv: Agentisk KI er ikke et teknologisk sprang, men en **organisatorisk test** – klarer vi å desentralisere makt uten å miste kontroll? Svaret ligger i å kombinere operasjonell disiplin (begrensninger, logging) med langsiktig utforskning (grafanalyse, chaos testing). Resiliens oppnås ikke ved å fjerne risiko, men ved å **bygge systemer som overlever sin egen kompleksitet**.