



id: SI\_006

model: mistralai/mistral-large-2512

seeds:

04\_risiko: sårbarhet

03\_relasjon: tillit

05\_handling: forhandle

01\_perspektiv: økonomisk

02\_tidhorisont: kortsiktig

## Instans SI\_006

**Seeds:** økonomisk press, kortsiktig pilotvindu, tillit som valuta, sårbarhet i legacy-systemer, forhandle autonomi-grenser

### Analyse: Agentisk KI som økonomisk sårbarhetsmultiplikator

Dere står foran en kortsiktig pilot (6–9 måneder) med økonomiske begrensninger – men agentisk KI introduserer sårbarheter som ikke lar seg løse med teknologi alene. Nøkkelen er å forhandle autonomi som en **tillitssone**, ikke en teknisk silo.

#### 1. Tillit som økonomisk buffer

I et legacy-landskap med manglende Zero Trust og RBAC/ABAC-gap, er tillit den reelle sikkerhetsmekanismen. Dere må **begrense agentenes autonomi til det dere har råd til å feile med:**

- **Økonomisk avgrensning:** Definer maksimal kostnad per agent-handling (f.eks. maks 100 API-kall/dag, maks 5 databasetransaksjoner/time). Bruk rate-limiting og kostnadsbudsjetter som hard grense.
- **Tillitssone-modell:** Del systemene inn i **tre soner** basert på sårbarhet og økonomisk risiko:
  - **Grønn sone** (lav risiko): Moderne mikrotjenester med idempotente API-er (f.eks. interne HR-systemer). Tillat autonomi med ephemeral credentials (15-min levetid).
  - **Gul sone** (middels risiko): Legacy-systemer med uforutsigbare sideeffekter (f.eks. økonomimoduler). Krev menneskelig godkjenning for



endringer via en “break-glass”-mekanisme (f.eks. Slack-approval med 5-min timeout).

- **Rød sone** (høy risiko): Systemer med transaksjonell integritet (f.eks. pensjonsberegninger). Forby autonomi helt – agenten kan kun foreslå handlinger som må godkjennes manuelt.

## 2. Forhandle autonomi-grenser som kontrakt

Agentenes “execution graph” må forhandles som en **juridisk-teknisk hybrid**:

- **Policy-as-code med økonomisk logikk**: Bruk OPA (Open Policy Agent) til å definere regler som kostnadsbegrenser handlinger. Eksempel:

```
allow {  
  input.action == "update_invoice"  
  input.cost_impact < 1000 # NOK  
  time.hour_of_day(input.time) >= 8  
  time.hour_of_day(input.time) <= 16  
}
```

- **Sårbarhetsbasert scoping**: For hver agent, lag en **risikoprofil** basert på:
  - Datakvalitet i kilde-API-er (f.eks. “API X har 20% feilrate i felt Y – agenten må validere før bruk”).
  - Økonomisk eksponering (f.eks. “Agenten kan ikke endre kontrakter over 500k NOK uten revisjon”).
  - Tidsvindu (f.eks. “Agenten kan kun kjøre i arbeidstid, med manuell override utenfor”).

## 3. Kortsiktige løsninger for langvarige sårbarheter

Med begrensede ressurser må dere prioritere økonomisk effektive tiltak som reduserer sårbarhet uten å kreve full arkitekturrevisjon:

- **Audit-pipeline som kostnadsdokumentasjon**: Bruk eksisterende logging (f.eks. ELK-stack) til å spore:
  - Agentens prompts (lagres i 30 dager, anonymisert).
  - API-kall med kostnadsestimater (f.eks. “Kall til ekstern betalingstjeneste: 0,50 NOK”).
  - Sideeffekter (f.eks. “Endret 3 rader i database X – estimert risiko: middels”).
  - Kritiske handlinger trigger automatisk varsel til Slack/Teams (f.eks. “Agent SI\_006 forsøkte å slette brukerdata – handling blokkert”).



- **Rollback som økonomisk sikring:** For hver agent-handling, lag en **automatisk kompenserende handling** som kan kjøres ved feil. Eksempel:
    - Agent oppretter faktura → lagre reverserings-skript som sletter fakturaen hvis den ikke godkjennes innen 24 timer.
    - Bruk idempotency keys for alle eksterne API-kall (f.eks. UUID per transaksjon) for å unngå dobbeltkjøring.
  - **Runaway-deteksjon via kostnadsmetrikker:** Overvåk agentens kumulativ kostnad per time. Hvis den overstiger en terskel (f.eks. 10x normalen), stopp agenten automatisk og send varsel.
- 

## Konkrete neste steg

### 1. Kartlegg sårbarhetsøkonomien:

- Lag en risiko/kostnads-matrise for alle systemer agenten skal aksessere (f.eks. "Legacy-fakturerings: høy risiko, lav kostnad per kall" vs. "Pensjonssystem: lav risiko, høy kostnad per feil").
- Prioriter rød sone-systemer for manuell godkjenning og grønn sone-systemer for autonomi.

### 2. Implementer tillitsbasert scoping i IAM:

- Opprett ephemeral service accounts for agenter med tidsbegrensede tokens (f.eks. 1 time levetid, automatisk rotasjon).
- Bruk OPA til å definere økonomiske grenser (f.eks. "Agenten kan ikke bruke mer enn 500 NOK/dag på eksterne API-kall").

### 3. Bygg en "økonomisk audit"-pipeline:

- Koble eksisterende logging til et kostnadsdashboard (f.eks. Grafana) som viser agentens kumulative kostnad og risikoeksponering i sanntid.
  - Sett opp automatiske varsler for handlinger som overstiger budsjett (f.eks. "Agent SI\_006 har brukt 80% av dagens API-budsjett").
- 

**Dissens:** Hvis dere venter på "perfekte" løsninger (Zero Trust, OPA, transaksjonelle garantier), vil piloten mislykkes. Tillit og økonomiske begrensninger er de eneste levedyktige kontrollmekanismene i et kortsiktig og sårbart landskap.