



id: SI_002

model: anthropic/claude-sonnet-4.6

seeds:

04_risiko: mulighet

03_relasjon: makt

05_handling: implementer

01_perspektiv: operasjonell

02_tidhorisont: syklisk

Instans SI_002

Seeds: operasjonell | syklisk | makt | mulighet | implementer

Kjerneperspektiv: Syklisk maktbalanse som styringsmodell

Mainstream-tilnærmingen fokuserer på statisk kontroll. Min linse er annerledes: autonome agenter opererer i **sykliske maktdynamikker** — tilgang akkumuleres, eskalerer, og må aktivt tømmes. Styringsmodellen må speile denne rytmen.

Operasjonell realitet: Makt som flytende ressurs

Agenter akkumulerer effektiv makt gjennom sesjoner — cachede tokens, åpne tilkoblinger, implisitte tilstander i downstream-systemer. Uten syklisk nullstilling bygger dere **latent autoritet** ingen har godkjent eksplisitt.

Konkret tiltak: Innfør “credential epoch”-sykluser på maks 15 minutter for agentsesjoner. Etter hver epoch: full token-invalidering, ny ABAC-evaluering, loggforseglet checkpoint. OPA kan evaluere policy per epoch-grense, ikke bare ved oppstart.



Implementer: Execution Graph som syklisk kontrollstruktur

Ikke modeller agentens handlinger som en lineær DAG — modeller det som **sykliske tilstandsmaskiner** med eksplisitte “return-to-safe-state”-overganger. Hvert node i grafen må ha:

- Definert reversibel handling (compensating action)
- Maks antall iterasjoner (loop-breaker på f.eks. 3 forsøk)
- Eskaleringsterskel til menneskelig operatør

Legacy-systemer uten idempotente API-er får automatisk **skrivesperr** som default — kun lesetilgang inntil kontrakten er verifisert.

Mulighet som risikovektor — undervurdert blindsoner

Typisk undervurdert: **“Opportunity coupling”** — agenten oppdager utilsiktede handlingsrom fordi systemlandskapet ikke er designet for å nekte. En agent med lesertilgang til et fagsystem kan via API-kjeding konstruere skrivetilgang ingen enkelt policy blokkerte. Dette er emergent makt, ikke konfigurasjonsfeil.

Mottak: Implementer en “blast radius”-kalkulator som simulerer maksimal rekkevidde for hver agentrolle før produksjonssetting. Verktøy: Cartography (AWS/GCP graph-analyse) tilpasset intern topologi.

Dissens fra mainstream

De fleste råd sier “start med logging, bygg kontroll etterpå.” Jeg anbefaler motsatt: **implementer hard operasjonell grense først** — én agent, ett system, null ekstern API-tilgang — og åpne syklisk basert på bevist adferd. Logging uten grenser gir observasjon uten kontroll.

Neste steg

1. **Uke 1-2:** Definer tre “autonomy tiers” med hard-kodede handlingsgrenser. Ingen agent opererer uten tier-klassifisering.
2. **Uke 3-4:** Implementer credential epoch-syklus (15 min) og loggførseglet checkpoint i pilotmiljø.
3. **Uke 5-6:** Kjør blast-radius-simulering på pilotagens topologi — stopp piloten hvis rekkevidde overstiger tier-grensen.