Bilag 1 - vedlegg B
Trondheim kommunes sikkerhetsarkitektur

Kontaktperson vedr. bruk av prinsipper

Arnstein Vestad,
Sikkerhetsarkitekt, Trondheim kommune
E-post: arnstein.vestad@trondheim.kommune.no

Tlf. 934 350 95

Innhold

1 Sikkerhetsprinsipper

I arbeidet med en sikkerhetsarkitektur er det viktig å ta stiling til hvilke prinsipper som skal være førende for arkitekturen. Disse prinsippene representerer den underliggende tankegangen i arkitekturen og vil danne som retningslinjer for videre arbeid med realisering og anvendelse av den.

Arkitekturprinsipper operer på et høyere abstraksjonsnivå enn sikkerhetskrav og representerer langsiktige mål og strategier. Kjennetegn for prinsipper er at de gjerne er teknologinøytrale og endres sjeldnere enn konkrete sikkerhetsmekanismer.

Trondheim Kommune har allerede et etablert sett med generelle overordnede arkitekturprinsipper. Disse prinsippene gjelder også for sikkerhetsarkitekturen. Dette dokumentet definerer et sett med prinsipper som omhandler sikkerhet spesifikt.

Man kan liste opp en lang rekke av sikkerhetsprinsipper som bør anvendes. Vi har valgt ut de som vi mener vil være mest førende for konkrete valg i arbeidet med arkitekturen. Prinsippene er basert på etablert beste praksis, samt Trondheim Kommunes mål for sikkerhet som dokumentert i Trondheim Kommunes Informasjonssikkerhetsstrategi. Sikkerhetsprinsippene er listet opp nedenfor:

- Sikkerhet i dybden
- Enkelhet i design, operasjon og administrasjon
- Innebygd personvern
- Sikkerhet i hele livsløpet
- Sporbarhet
- Arkitektur med preventive, detekterende og korrigerende mekanismer
- Minimere tillit
- Separasjon av sikkerhetskomponenter (compartmentalization)
- Fokus på brukervennlighet og tilgjengelighet

Prinsippene er beskrevet i detalj i de påfølgende kapitlene.

1.1 Sikkerhet i dybden

Sikkerhet i dybden
Begrunnelse
Man benytter flere lag med sikkerhetsmekanismer/barrierer (security controls). Dette prinsippet er benyttet i sikkerhetsarkitekturen fordi man må anta at svikt i sikkerhetsmekanismene kan forekomme. Dersom dette skjer vil det da finnes andre mekanismer som kan beskytte systemet.
Konsekvenser / eksempler på anvendelse
• Systemer og informasjon bør ha flere uavhengige sikkerhetsmekanismer. Dette er spesielt viktig for mekanismer som skal beskytte systemer og informasjon med høy kritikalitet. • Systemer plasseres i soner med sikkerhetsbarrierer mellom sonene. For tilgang til sensitiv informasjon kreves det minimum to uavhengige barrierer. • Tjenester som gir tilgang til sensitiv informasjon krever både autentisering av sluttbruker og av applikasjonen. • Brannmur på hver enkelt maskin i tillegg til ekstern brannnmur

1.2 Enkelhet i design, operasjon og administrasjon

Enkelhet i design, operasjon og administrasjon
Begrunnelse
Komplekse sikkerhetssystemer gir større mulighet for feil i design, implementasjon og bruk. Et prinsipp i arkitekturen er derfor at man velger så enkle mekanismer som mulig for å oppfylle sikkerhetsmålene.
Konsekvenser / eksempler på anvendelse
• Velge så enkle mekanismer som mulig. • Mulige fremtidige behov skal ikke automatisk gi grunnlag for å velge mer komplekse løsninger. • Fleksibilitet er ikke nødvendigvis ønskelig da dette øker kompleksitet • Sikkerhetsfunksjonalitet bør om mulig trekkes ut i separate komponenter som kun har dette som oppgave.

1.3 Innebygd personvern

Innebygd personvern
Begrunnelse
Trondheim Kommune har store mengder personopplysninger i sine systemer og er dermed underlagt kravene i Personopplysningsloven. Det skal tas hensyn til personvern i alle utviklingsfaser av et system eller en løsning. Punktene under er hentet fra Datatilsynets «Syv steg til innebygd personvern»: • Vær i forkant, forebygg fremfor å reparere • Gjør personvern til standardinnstilling • Bygg personvern inn i designet • Skap full funksjonalitet: Både-og, ikke enten-eller • Ivareta informasjonssikkerheten fra start til slutt • Vis åpenhet • Respekter brukerens personvern
Konsekvenser / eksempler på anvendelse
• Det må være fokus på personvern og hvordan dette ivaretas i hele livssyklusen til løsningen, fra design til operasjon • Ikke samle og behandle flere personopplysninger enn nødvendig • Sikre rutiner for sletting og oppdatering av personopplysninger • Implementasjon tar høyde for at det er personopplysninger som behandles • Mekanismer for å håndtere personopplysninger er åpne og dokumentert

1.4 Sikkerhet i hele livsløpet

Sikkerhet i hele livsløpet
Begrunnelse
For å oppnå sikre løsninger kreves det fokus på sikkerhet gjennom hele livsløpet til applikasjonen, helt fra utviklingsløpet planlegges til løsningen en gang legges ned og tas ut av drift. Sikkerhet må planlegges for allerede i oppstarten av prosjektet. Det må tidlig gjennomføres RoS analyse og nødvendige sikkerhetsaktivteter må planlegges og budsjetteres for. Selve utviklingsløpet må også ha fokus på sikkerhet helt fra starten og følge beste praksis for sikker utviklingsmetodikk. Behovet for å arbeide med sikkerhet stopper ikke i det systemet er levert og prosjektet avsluttes. En løsning som er sikker i dag er kanskje ikke sikker om et år eller to fordi det blir funnet nye sårbarheter i 3. parts komponenter som er benyttet i løsningen (ref. OpenSSL og Heartbleed). Endringer i omkringliggende miljø og trusselbilde kan også gjøre at det er behov for å tilpasse sikkerhetsmekanismer i løsningen.
Konsekvenser / eksempler på anvendelse
• Fokus på sikkerhet i alle faser av livsløpet for en løsning: planlegging, krav, design, implementasjon, test, drift og operasjon. • Sikkerhetsaktivteter må tas med i planer og budsjetter for alle faser av prosjektet fra start til slutt. • Man må sikre at man er i stand til å oppgradere og videreutvikle løsningen i takt med endringer i trusselbilder og kjente sårbarheter.

1.5 Sporbarhet

Sporbarhet
Begrunnelse
Dersom det skulle inntreffe en uønsket hendelse er det viktig at man har en arkitektur som sikrer at man kan undersøke hva som har skjedd og hvem/hva som har utløst hendelsen.
Konsekvenser / eksempler på anvendelse
• Man har en klar policy for hva som skal logges og hvordan. • Personvernhensyn må ivaretas i logging og prosesser for håndtering av logger.

1.6 Arkitektur med preventive, detekterende og korrigerende mekanismer

Arkitektur med preventive, detekterende og korrigerende mekanismer
Begrunnelse
Det er ikke mulig å sikre seg 100% mot sikkerhetsbrudd. Det vil alltid kunne være sårbarheter eller bakveier som kan utnyttes og menneskelige feil kan skje. Arkitekturen må ta høyde for at sikkerhetsmekanismer kan svikte, og det må finnes mekanismer for å oppdage at dette har skjedd og planer for hvordan man skal reagere for gjenopprette normal situasjon og minimere skadeomfang.
Konsekvenser / eksempler på anvendelse
• Arkitekturen må ha mekanismer for deteksjon av innbrudd/angrep • Det må finnes planer for hvordan man skal reagere på et evt. innbrudd (hendelseshåndtering)

1.7 Minimere tillit

Minimere tillit
Begrunnelse
Systemer og systemkomponenter som håndterer sensitive data bør i størst mulig grad søke å beskytte seg selv å ha minst mulig tillit til omkringliggende systemer og nettverk.
Konsekvenser / eksempler på anvendelse
• Alle personer og systemer og nettverk skal ha et klart definert nivå av tillit. Tillitsnivået kan være definert gjennom roller eller plassering i sikkerhetssoner. • Beskytte mot angrep både fra innsiden og utsiden • Minimale rettigheter (least privilege) • Minimere tillit til nettverk – kommunikasjon sikres ende til ende • Beskyttelsesmekanismer plasseres så nært verdien som mulig. • Systemer bør sikre seg selv i stedet for å stole på at eksterne systemer skal etablere nødvendige sikkerhetsbarrierer. • Brannmur på hver enkelt maskin i tillegg til ekstern brannnmur

1.8 Separasjon av sikkerhetskomponenter (compartmentalization)

Separasjon av sikkerhetskomponenter (compartmentalization)
Begrunnelse
Komponenter som inneholder sikkerhetsfunksjoner eller behandler sensitiv informasjon bør i minst mulig grad blandes sammen med komponenter som utfører andre funksjoner. Dette vil sikre at sikkerhetskomponenten ikke kan bli påvirket av feil eller sikkerhetsbrudd i de andre komponentene. Separasjon bør praktiserer både på programvareprosessnivå, maskinnivå og nettverksnivå.
Konsekvenser / eksempler på anvendelse
• Prosesser som håndterer sensitiv informasjon eller sikkerhetsfunksjoner bør ikke håndtere andre funksjoner som ikke er direkte relatert til dette. • Tjenester kjører på egne maskiner som ikke kjører andre tjenester • Separate nettverkssoner til hver tjeneste

1.9 Fokus på brukervennlighet og tilgjengelighet

Fokus på brukervennlighet og tilgjengelighet
Begrunnelse
Sikkerhetsarkitekturen skal støtte opp rundt visjonen bak «digitalt førstevalg» hvor kommunens tjenester skal åpnes opp og gjøres tilgjengelig for publikum. Dersom disse tjenestene skal bli brukt må sikkerhetsmekanismene være så enkle og brukervennlig som mulig. Unødvendig strenge krav til sikkerhet og bruk av komplekse sikkerhetsmekanismer vil redusere bruken og dermed verdien av tjenestene.
Konsekvenser / eksempler på anvendelse
• Sikkerhetskrav skal ikke være strengere enn absolutt nødvendig dersom dette medfører sikkerhetsmekanismer som vil gjøre tjenestene vanskeligere å bruke og dermed mindre tilgjengelig for publikum. • Tilby autentiseringsmekanismer som er tilpasset nivået på informasjonen • Risiko og sårbarhetsanalyse (RoS) danner utgangspunkt for å vurdere hvilke sikkerhetsmekanismer som er strengt nødvendige og hvilke som kan lettes på.