IN2120 – Cybersikkerhet

Del 8: Systemsikkerhet

Nils Petter Wien



Høst 2026

Universitetet i Oslo

Viktigheten av systemsikkerhet

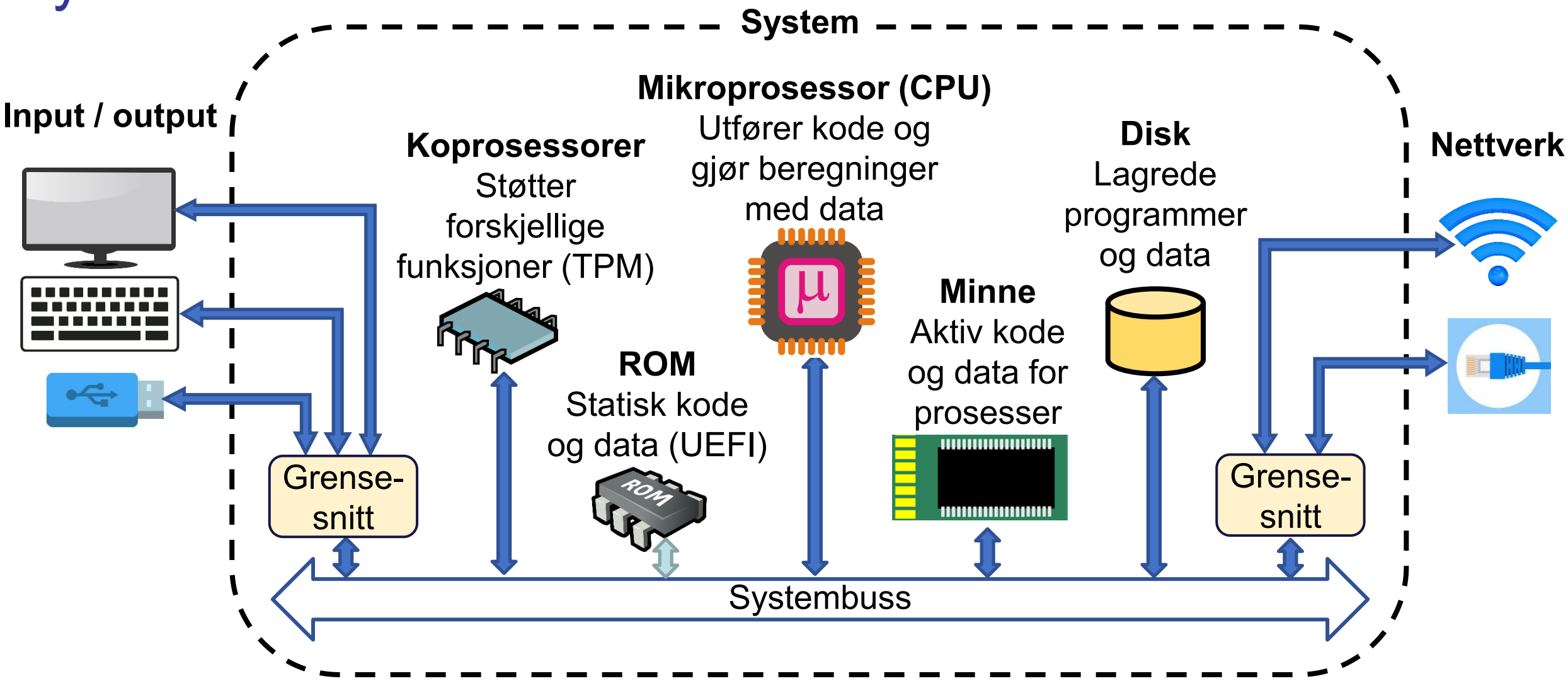


- *«Å bruke kryptering på internett tilsvarer å bruke en pansret bil for å levere kredittkortinformasjon fra en som bor i en pappkartong til en som bor på en parkbenk.»*

  Eugene Spafford

- Moralen er at god kommunikasjonssikkerhet er bortkastet hvis systemsikkerheten er svak.

Systemarkitektur



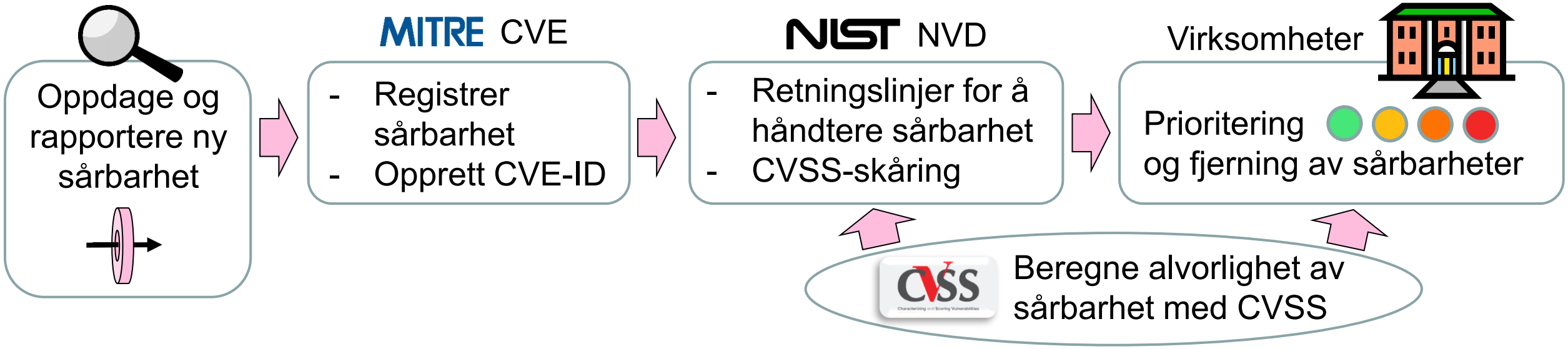
Kontinuerlig sikkerhetsoppdatering

- Nye sårbarheter oppdages kontinuerlig
  - Krever kontinuerlig sikkerhetsoppdatering

- Sikkerhetsoppdatering kan ta tid, slik at angripere ofte utnytter kjente sårbarheter
  - Testing og verifikasjon av oppdateringer
  - Oppdateringer kan ødelegge for implementasjonen din

- En zero-day-sårbarhet er en sårbarhet som leverandøren ikke kjenner til, eller ikke har en tilgjengelig sikkerhetsoppdatering for.

CVE (Common Vulnerabilities and Exposures)
CVSS (Common Vulnerability Scoring System)



- CVSS (Common Vulnerability Scoring System) beregner alvorlighet av en sårbarhet fra 0 til 10. Virksomheter prioriterer fjerning/mitigering av sårbarheter ut ifra alvorlighet.
  - Kritisk: 9,0–10
  - Høy: 7,0–8,9
  - Middels: 4,0–6,9
  - Lav: 0,1–3,9

Tilnærminger for å styrke systemsikkerheten

- Fjern feil og sårbarheter i OS (operativsystemet)
  - Nye versjoner og sikkerhetsoppdateringer



- Legg til sikkerhetsfunksjoner i OS og CPU (Central Processing Unit)
  - Privilegienivåer, NX (No eXecute), ASLR (Address Space Layout Randomization)



- Monitorering av systemsikkerhet
  - Antivirusverktøy
  - Brannmur, inntrengingsdeteksjon



- Virtualiseringsteknologi
  - Beskytte prosesser ved å separere virtuelle maskiner



- Tiltrodd beregning, f.eks.
  - Sikker oppstart med UEFI



  - Sikker maskinvare på plattformen, f.eks. TPM (Trusted Platform Module)



Privilegienivåer for prosesser i et system

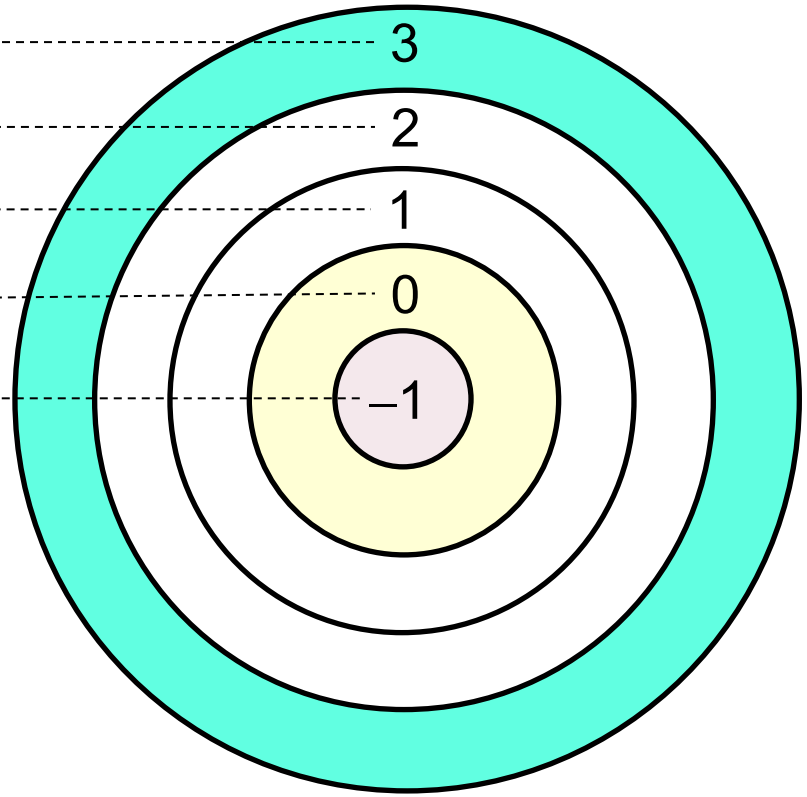
Nivå 3: Applikasjoner (brukermodus)

Nivå 2: (som regel ubrukt)

Nivå 1: (som regel ubrukt)

Nivå 0: OS: Unix root, Win. adm. (kernelmodus)

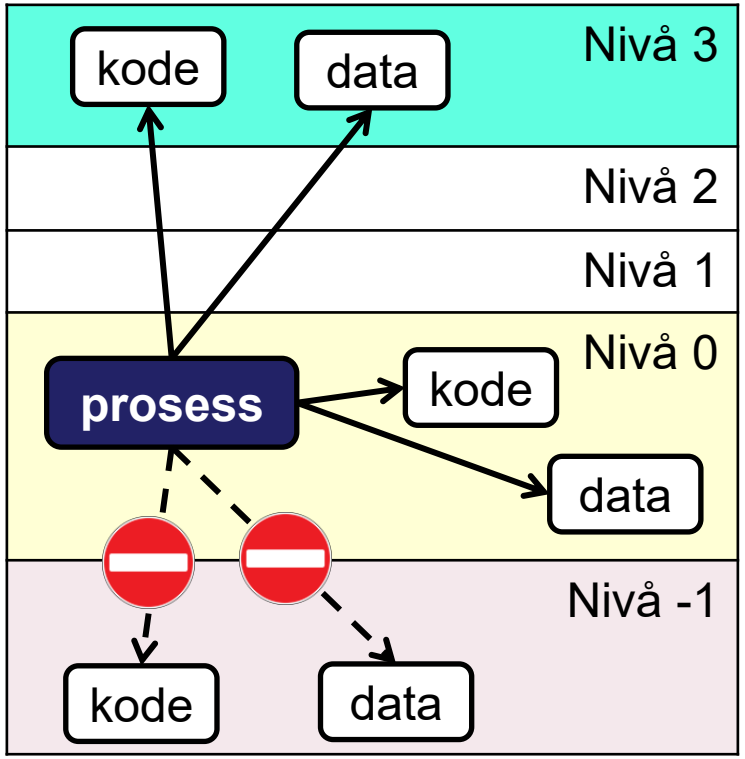
Nivå –1: Hypervisor (hypervisormodus)



- Modellen ble introdusert i CPU-arkitekturen på 80-tallet
  - Nivå 0: mest privilegert (høyest)
  - Nivå 3: minst privilegert (lavest)
- Nytt mest privilegert «nivå -1» for hypervisor ble innført ca. 2006
- Illustrert metaforisk som «beskyttelsesringer»

Tilgangskontroll basert på privilegienivåene

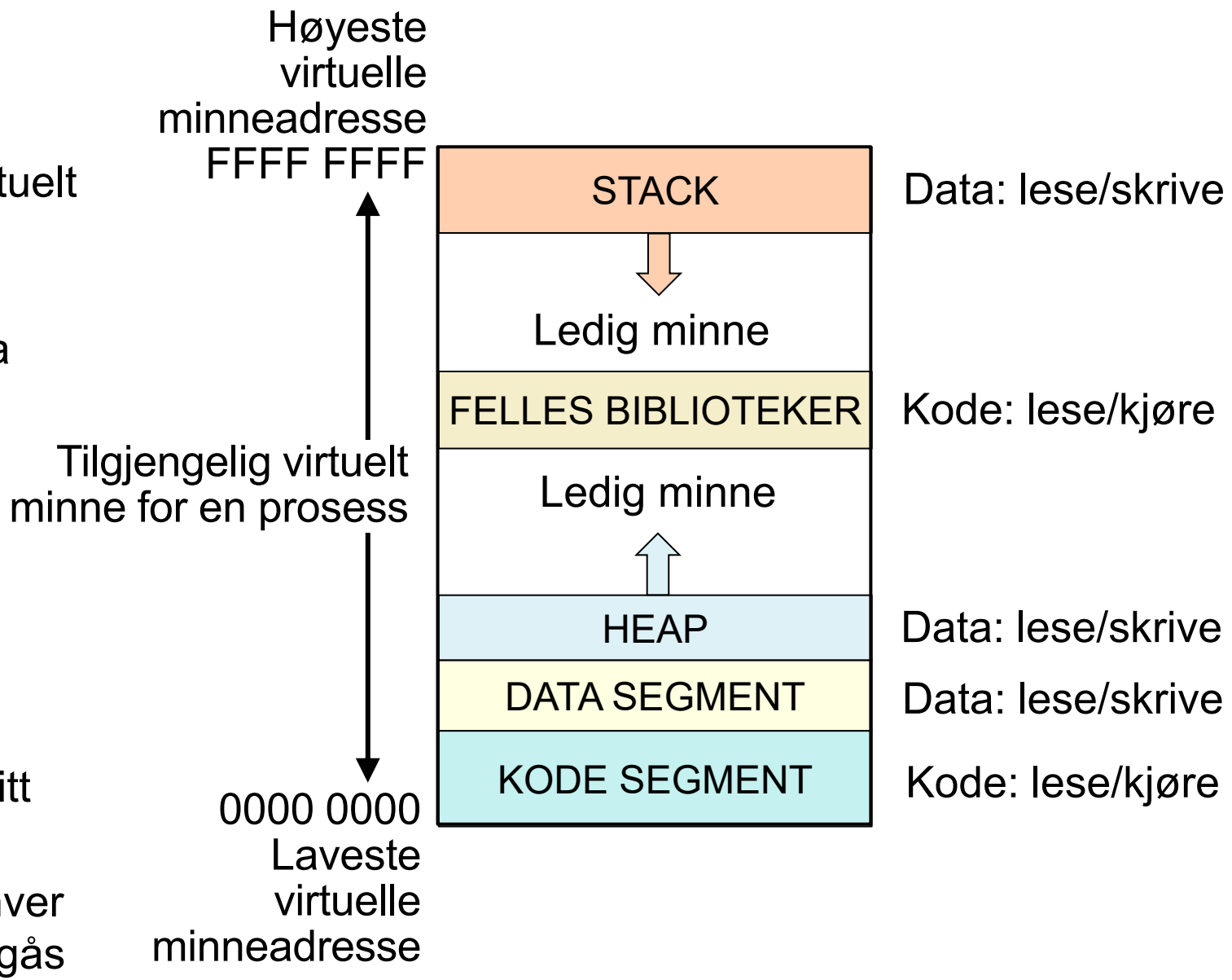
- En prosess kan få tilgang til, og endre data og programvare på samme eller lavere privilegienivå som seg selv.
- En prosess i kernelmodus (nivå 0) kan få tilgang til data og SW på nivå 0, 1, 2 og 3
  - men ikke på nivå -1
- Målet til angriper er å få tilgang til kernel (nivå 0) eller hypervisormodus (nivå -1), f.eks.
  - gjennom exploits
  - ved å lure brukerne til å installere skadevare



Eksempel

Virtuelt minne for en prosess

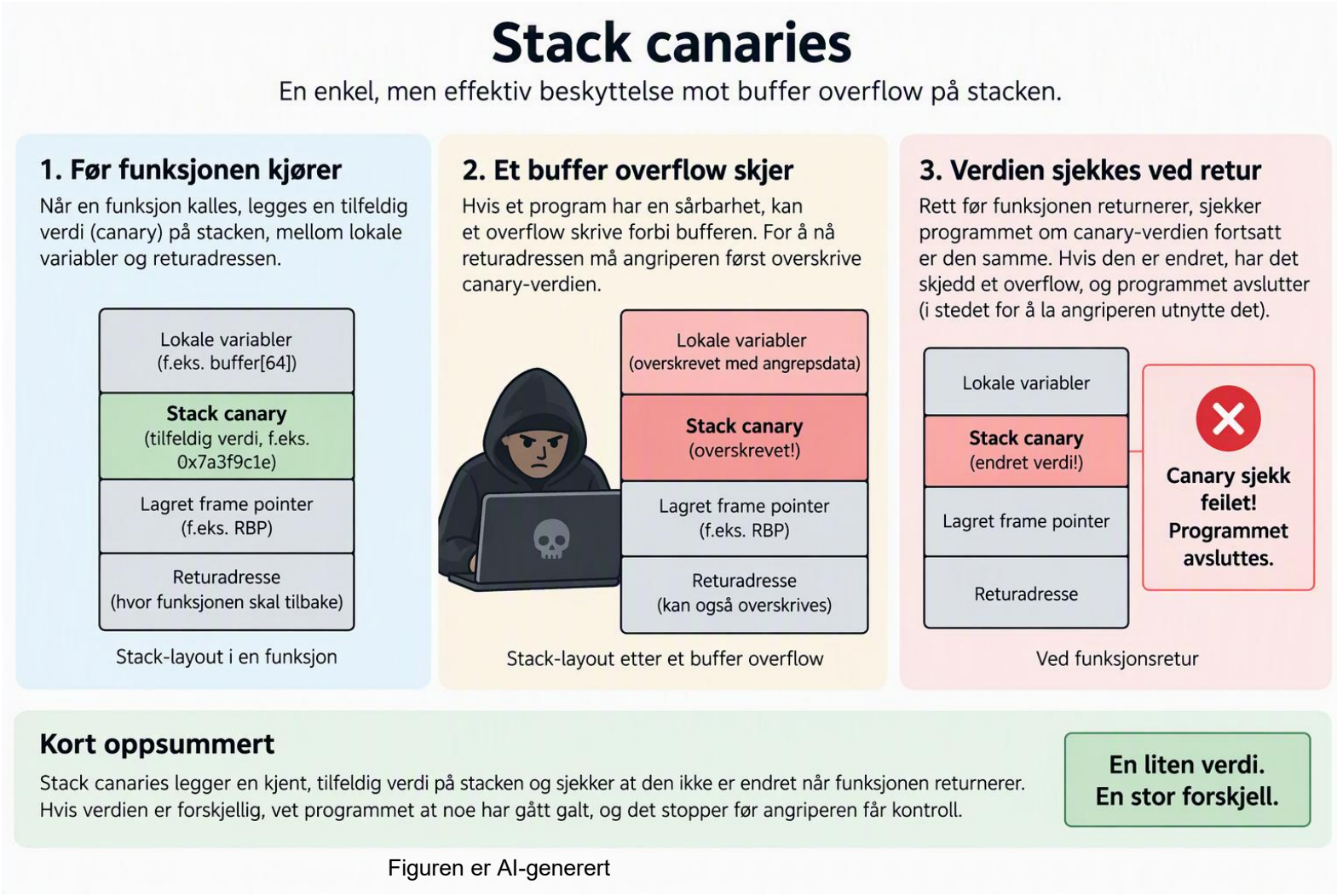
- Hver 32-bits prosess har eget 4GB virtuelt minneområde.
- Virtuelle minneadresser oversettes av CPU til fysiske minneadresser før data leses og skrives i det fysiske system-minnet.
- Selv om alle prosessene har samme virtuelle adresseområde, har de i virkeligheten separate fysiske adresseområder.
- En prosess har ikke tilgang til fysiske adresser for andre prosesser, kun til sitt eget fysiske adresseområde.
- Virtuelle minneadresser vil være like hver gang prosessen kjører. Dette kan unngås med ASLR (se senere).



Buffer overflow – mottiltak

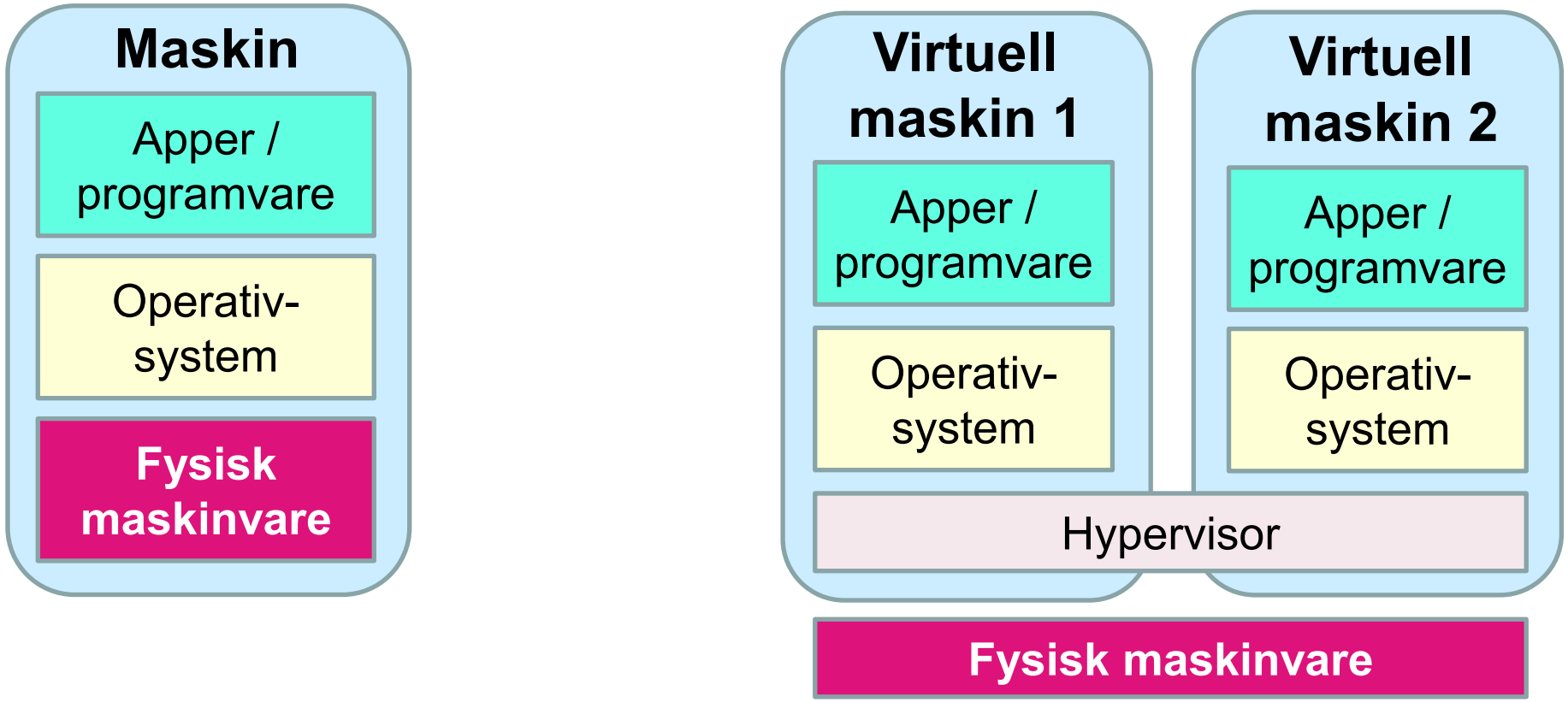
- No Execute (NX): stacken merkes som ikke-kjørbar, og CPU-en nekter å kjøre kode der
- Stack canaries: tilfeldig verdi lagres på stacken. Sjekker om data har blitt skrevet over før funksjonen returnerer
- ASLR (Address Space Layout Randomization): angriper må vite hvor i minnet angrepskoden som skal kjøre ligger. ASLR gjør det vanskeligere å finne denne adressen
- Bruk av (sikrere) programmeringsspråk og statisk analyse som unngår sårbarhet for buffer overflow
- Skriv bedre kode (innebygd sikkerhet / DevSecOps)

Buffer overflow – mottiltak



Figuren er AI-generert

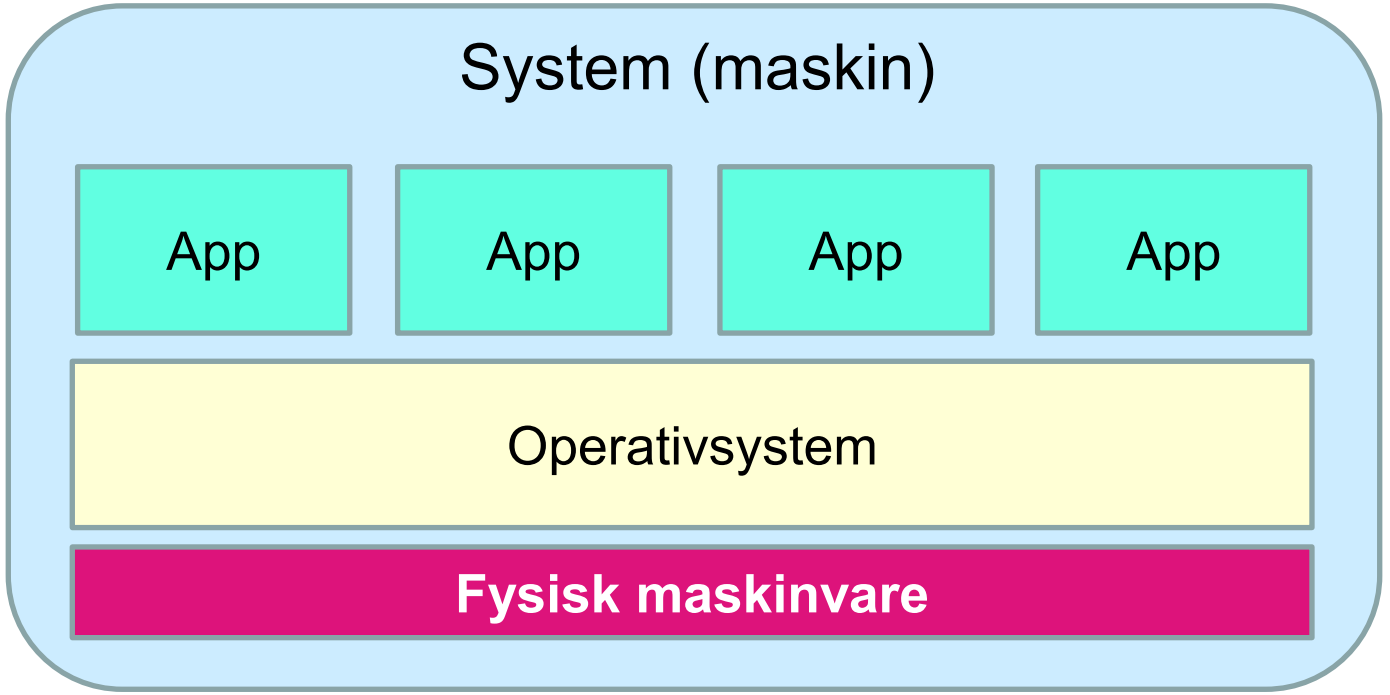
Maskin og «virtuell maskin»



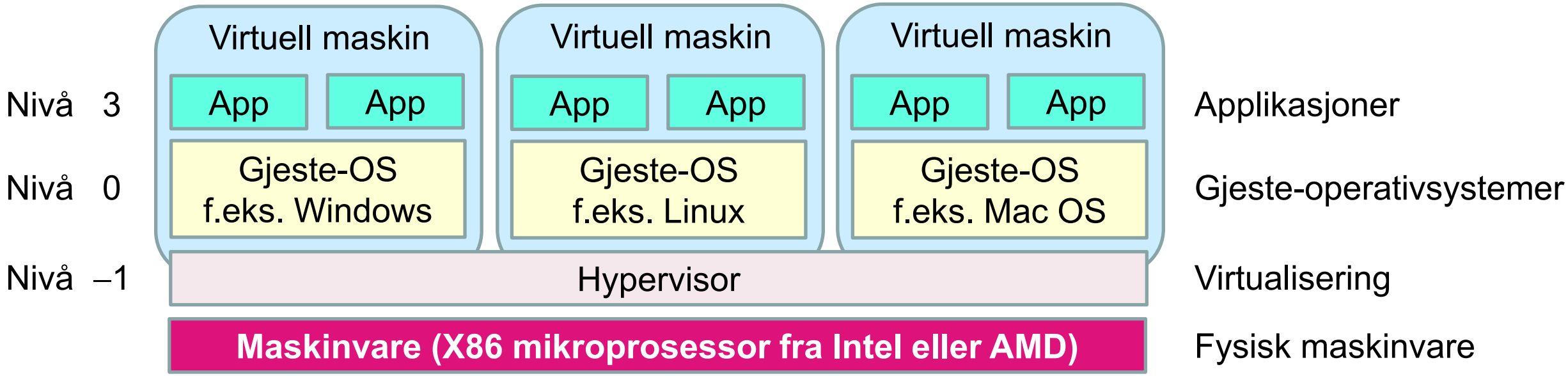
Tradisjonell system-stack

Nivå 3: Brukerapplikasjoner

Nivå 0: OS, Linux root, Win. adm.

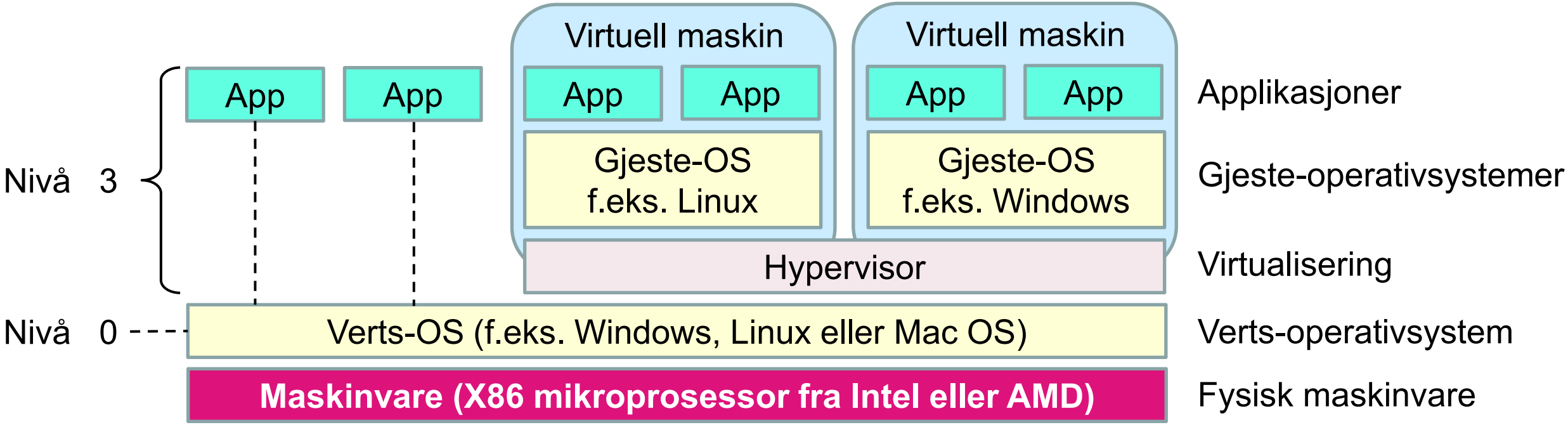


Type 1 virtualisering (native)



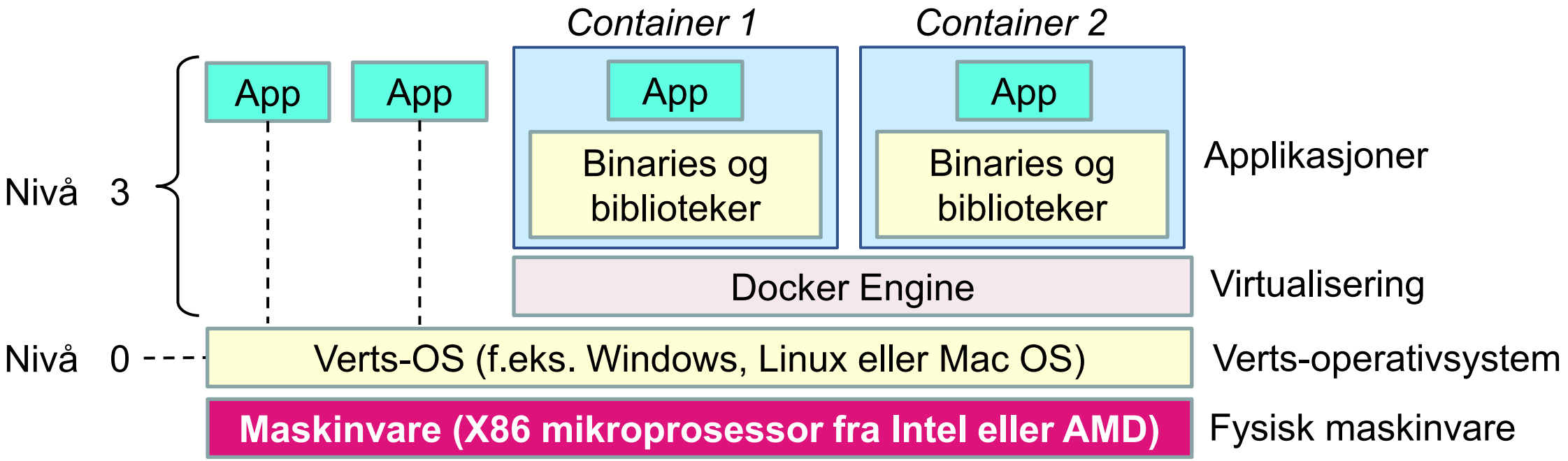
- Type 1 virtualisering er den mest direkte og mest effektive måten å kjøre virtuelle maskiner på.
- Hypervisor er programvare som spiller rollen som fysisk maskinvare.
- Type 1-virtualisering gir en logisk struktur på privilegienivåene ved at hypervisoren er mer privilegert enn gjeste-operativsystemene som den styrer.
- Gjeste-operativsystemene kjører med privilegienivå 0, som de nettopp er designet for.

Type 2 virtualisering (vertsbasert)



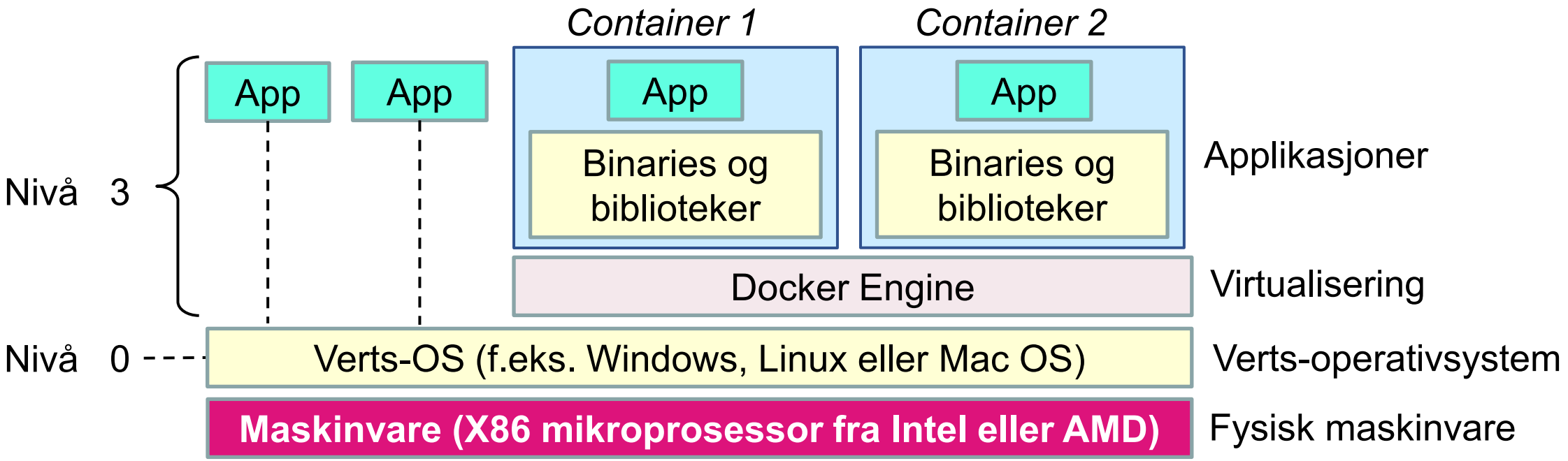
- Hypervisor-programmet blir installert som en vanlig applikasjon på vertsoperativsystemet.
- Enkelt for vanlige brukere å installere og bruke.
- Type 2 bruker maskinvarestøtte (VT-x/AMD-V), men har fortsatt mer overhead enn type 1. Verts-OS-et styrer ressursene, og I/O går via vertens drivere.

Docker Engine og containere (vertsbasert)



- Docker Engine gjør det mulig for containeriserte applikasjoner å kjøre på ulike plattformer.
- Docker Engine er en forenklet virtualisering, som gjør gjeste-OS unødvendig.

Docker Engine og containere (vertsbasert)



- Eliminerer avhengigheter på OS-plattformer, fordi avhengighetene er innebygd i containeren.
- En container er en programvareenhet med kode og alle dens avhengigheter, slik at applikasjonen kjører raskt og pålitelig, og kan lett porteres mellom ulike plattformer som har Docker Engine.

- Containere deler kjerne med verten
  - Svakere isolasjon enn VM-er fører til potensielt svakere sikkerhet

Bruk av virtualisering

- Skyleverandører driver store serverparker
  - Hver kunde får sin egen VM
  - Mange kunder deler den samme maskinvaren
  - Lett å migrere VM mellom servere for å øke/redusere kapasiteten



Amazon EC2 Datasenter

- Testing og programvareanalyse
  - Potensielt skadelige eksperimenter kan trygt utføres i isolerte omgivelser
  - Ta et snapshot av den nåværende tilstanden til operativsystemet
  - Systemet kan til enhver tid settes tilbake til snapshot-tilstanden
  - Analyse av skadevare



Virtualisering og sikkerhetsmål

- Gjeste-(operativ)systemene må ikke kunne aksessere eller bli påvirket av hverandre
- Gjeste-(operativ)systemene må ikke kunne påvirke hypervisor-programmet
- Gjeste-(operativ)systemene må ikke kunne detektere at de er virtualisert

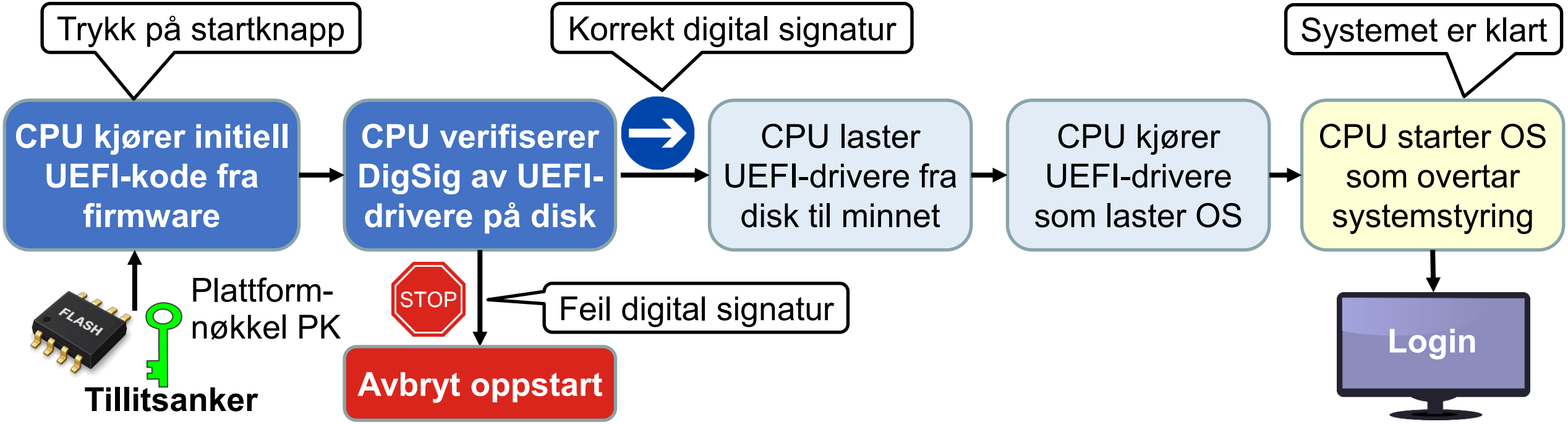
Tiltrodd beregning

- Tiltrodd beregning (trusted computing) betyr at aspekter ved sikkerheten i et system er forankret i maskinvare på en eller annen måte.

- Maskinvare anses som mer robust mot sikkerhetstrusler enn programvare.

- Eksempler:
  - sikker oppstart med UEFI (*Unified Extensible Firmware Interface*)
  - TPM (Trusted Platform Module)
  - TEE (Trusted Execution Environment)

Tiltrodd beregning - sikker oppstart med UEFI



- UEFI erstatter BIOS i moderne computere. Styrer oppstartsekvensen.
- Programmoduler for oppstart er digitalt signert.
- UEFI-kode i flash-minne initierer oppstartsekvensen.
- Signaturene sjekkes mot en liste over godkjente nøkler (db) og en svarteliste (dbx). Platform Key (PK) beskytter disse listene mot endring.
- Hvis en feil signatur detekteres vil oppstartsekvensen avbrytes.

Tiltrodd beregning - TPM

- TPM (Trusted Platform Module) er en sikkerhetsbrikke (koprosessor) som finnes i de fleste moderne PC-er. Den støtter blant annet:
  - målt oppstart (measured boot): registrerer hva som startes
  - attestering: beviser oppstartstilstanden overfor tredjeparter
  - forsegling av nøkler (sealed storage): f.eks. diskkrypteringsnøkkelen til BitLocker

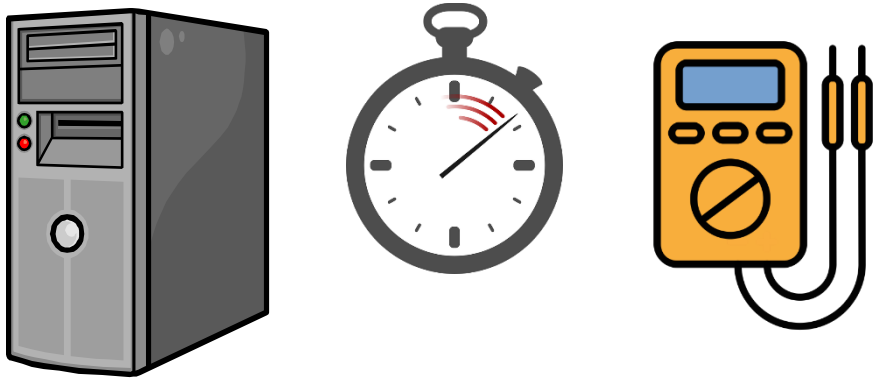
- Sikker oppstart *stopper* ukjent kode, mens målt oppstart *registrerer* den

Tiltrodd beregning - TEE

- Data og beregninger beskyttes av maskinvare i prosessoren, ikke bare av operativsystemet
  - Selv et kompromittert OS eller en hypervisor skal ikke kunne lese dataene

- Eksempler:
  - ARM TrustZone: brukes i mobiltelefoner, f.eks. for fingeravtrykk og betaling
  - Intel SGX/TDX og AMD SEV: brukes i skyen for å beskytte kundedata mot leverandøren

Sidekanaler



- En sidekanal er en utilsiktet informasjonslekkasje som skyldes den fysiske implementasjonen av et system, ikke selve grensesnittet

- Angriperen observerer bieffekter og trekker slutninger om hemmelige data
  - Tidsforbruk: ulik behandlingstid kan avsløre f.eks. bit i en kryptonøkkel
  - Strømforbruk, lyd og elektromagnetisk stråling

Skjulte kanaler

- En *skjult kanal* er en mekanisme som ikke er designet for kommunikasjon, men som kan misbrukes for å overføre informasjon.
- Kommunikasjon mellom to parter som *ikke har lov* til å sende dataene
- Båndbredden (kapasiteten) til skjulte kanaler er typisk svært begrenset.

- Eksempler:
  - **Air-gap:** data sendes ut fra isolerte maskiner via viftestøy eller blinkende lysdioder.
  - **App-samarbeid på mobil:** én app har tilgang til kontakter, en annen har nettilgang. De «snakker» ved å endre og lese av volumet på telefonen.
  - **DNS-tunnelering:** data skjules i DNS-oppslag for å komme forbi brannmuren.

Slutt på presentasjonen