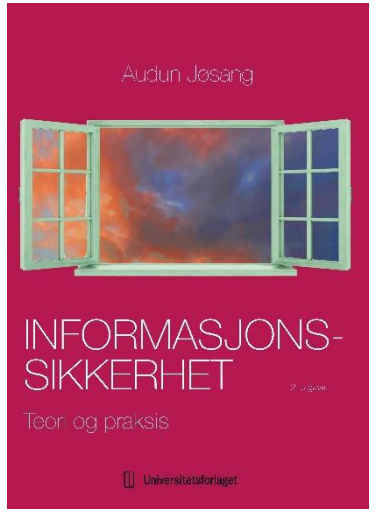
Kapittel 3: Systemsikkerhet

Informasjonssikkerhet: Teori og praksis

Audun Jøsang

2. utg. 2023

Universitetsforlaget



Oversikt

- Systemarkitektur
- Sikkerhet i operativsystemer
- Virtualiseringsarkitekturer
- Tiltrodd beregning
- Sikker oppstart
- Side-kanaler og skjulte kanaler

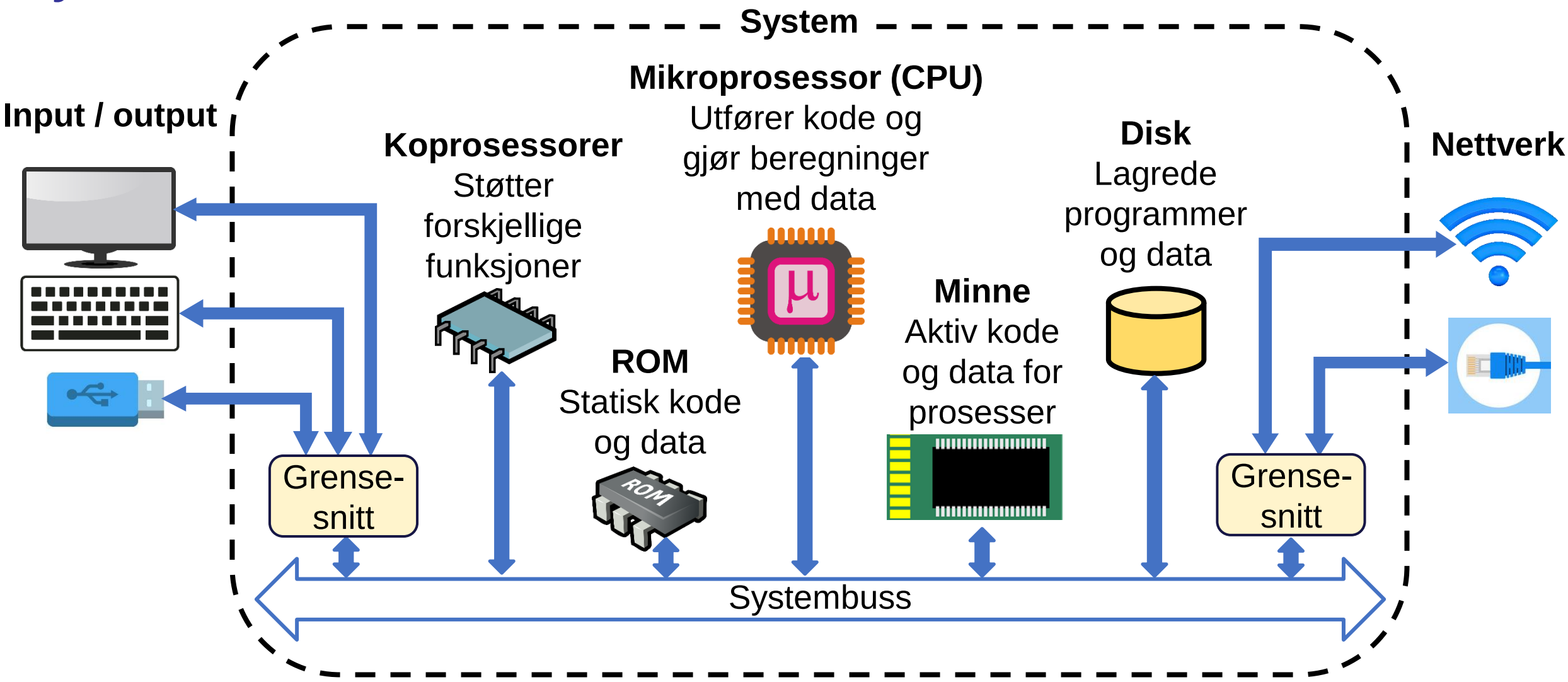
Viktigheten av systemsikkerhet

- *«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.»*

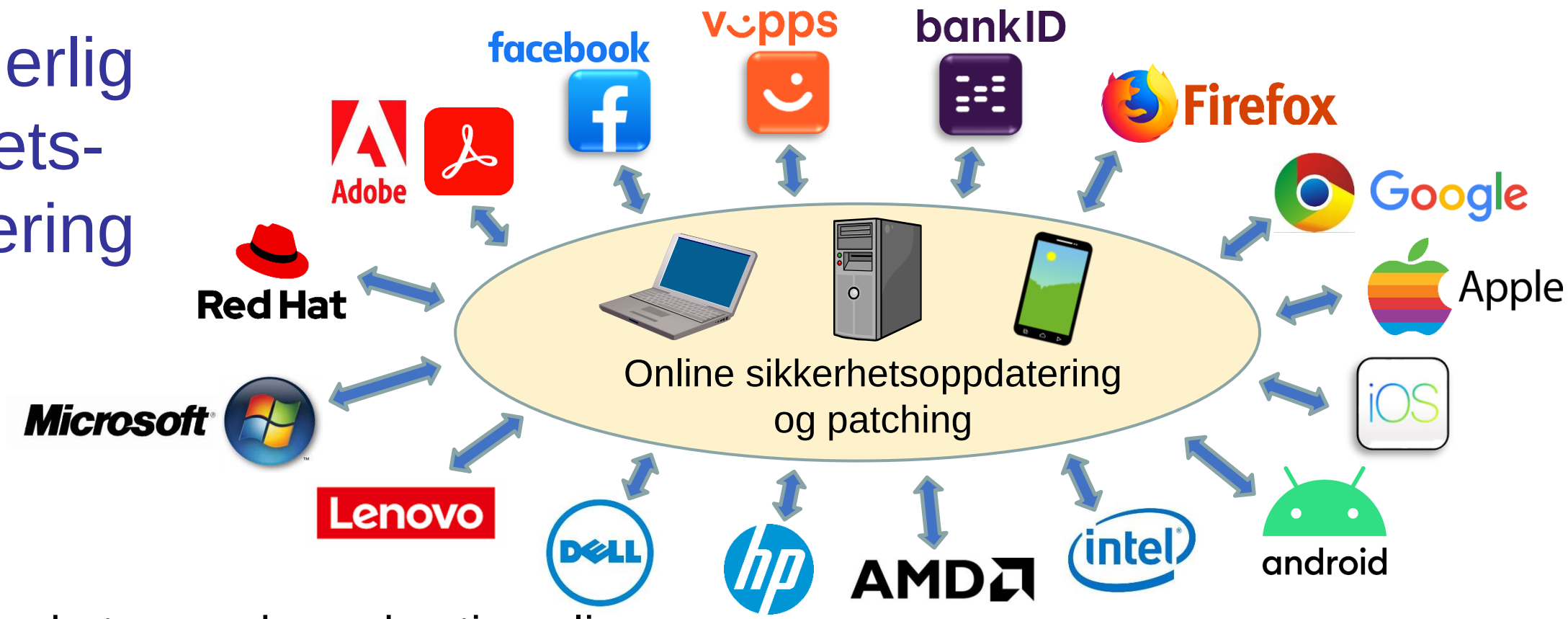
  Eugene Spafford

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

Systemarkitektur

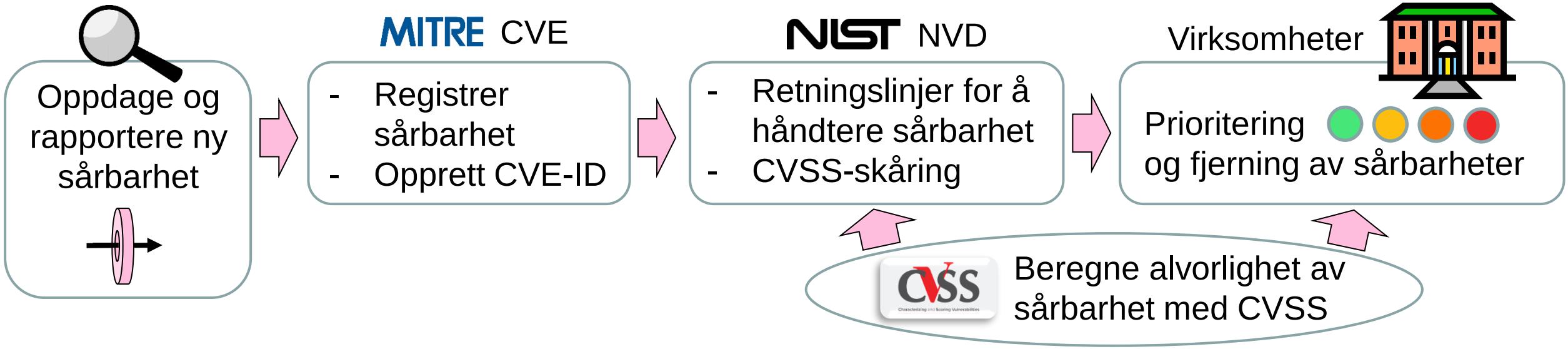


Kontinuerlig sikkerhets-oppdatering



- Nye sårbarheter oppdages kontinuerlig
- Krever kontinuerlig online sikkerhetsoppdatering
- En nulldags- (zero-day) sårbarhet er kun kjent for angriper, slik at programvareprodusenten har hatt null dager (zero days) til å fjerne sårbarheten
- Sikkerhetsoppdatering kan ta tid, slik at angripere ofte utnytter kjente sårbarheter

CVE (Common Vulnerability Enumeration)
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 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)

OS privilegienivåer

- Hierarkiske privilegienivåer ble introdusert i X86 CPU-arkitekturen til Intel (og AMD) i 1985 (Intel 80386), med 4 nivåer
  - Nivå 0: høyest
  - Nivå 3: lavest
- Nytt høyest nivå -1 innført ca. 2006
- Illustrert metaforisk som “beskyttelsesringer”

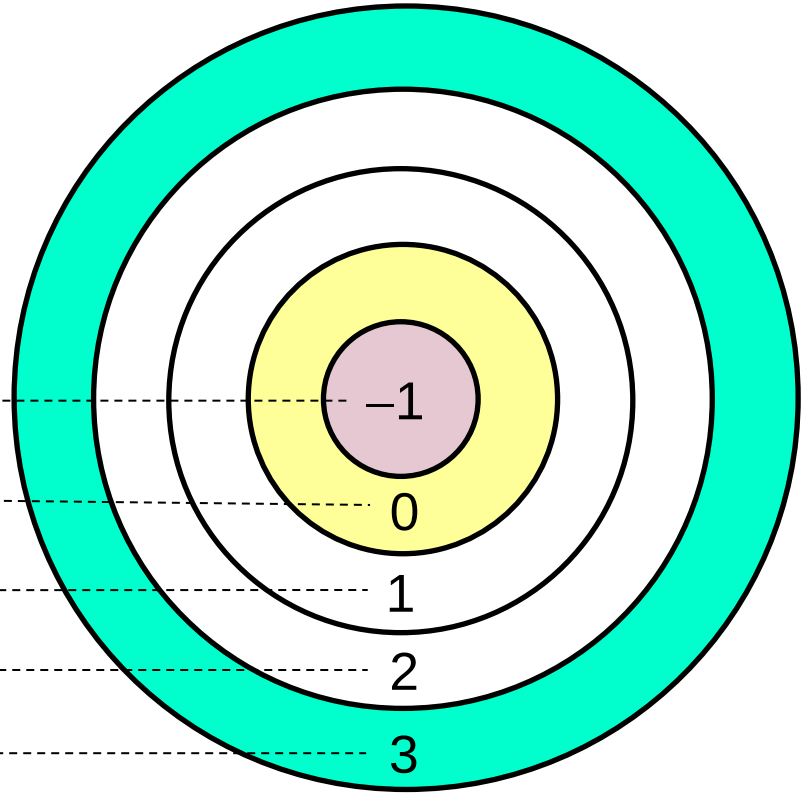
Nivå –1: Hypervisormodus

Nivå 0: Kernelmodus (Unix root, Win. Adm.)

Nivå 1: (som regel ubrukt)

Nivå 2: (som regel ubrukt)

Nivå 3: Applikasjon- og brukermodus



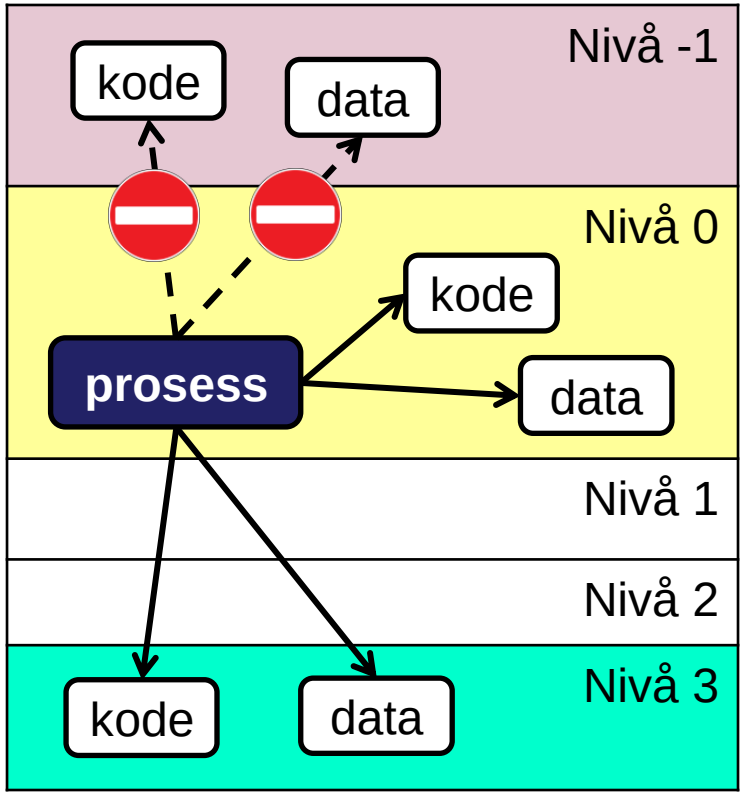
Hva skjedde med nivå 1 og 2 ?

- Det viste seg etterhvert at den hierarkiske sikkerheten som de fire beskyttelsesringene ga, ikke samsvarte særlig godt med behovene til systemprogrammereren, og at det ga liten eller ingen forbedring i forhold til den enkle sikkerhetsmodellen med bare to nivåer. Beskyttelsesringer kunne implementeres effektiv i maskinvare, men det var lite annet å si om dem.
- Å implementere operativsystemer med mange sikkerhetsnivåer viste seg å være en blindgate.

Maurice Wilkes (1994)

Prinsipp for bruk av privilegienivåene

- En prosess kan få tilgang til, og endre data og programvare på samme eller lavere privilegie-nivå som seg selv.
- En prosess som kjører 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



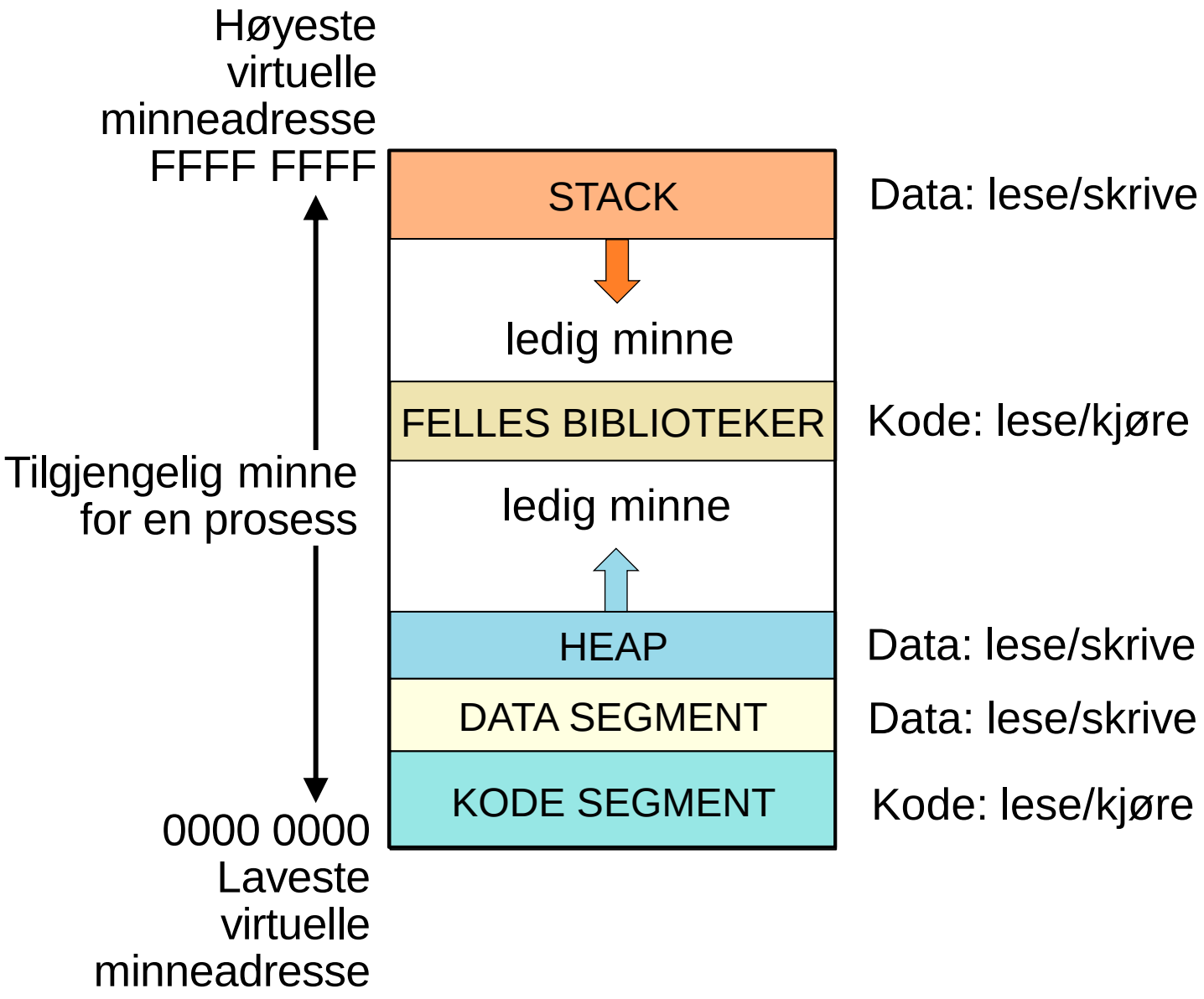
Eksempel

Buffer overflow

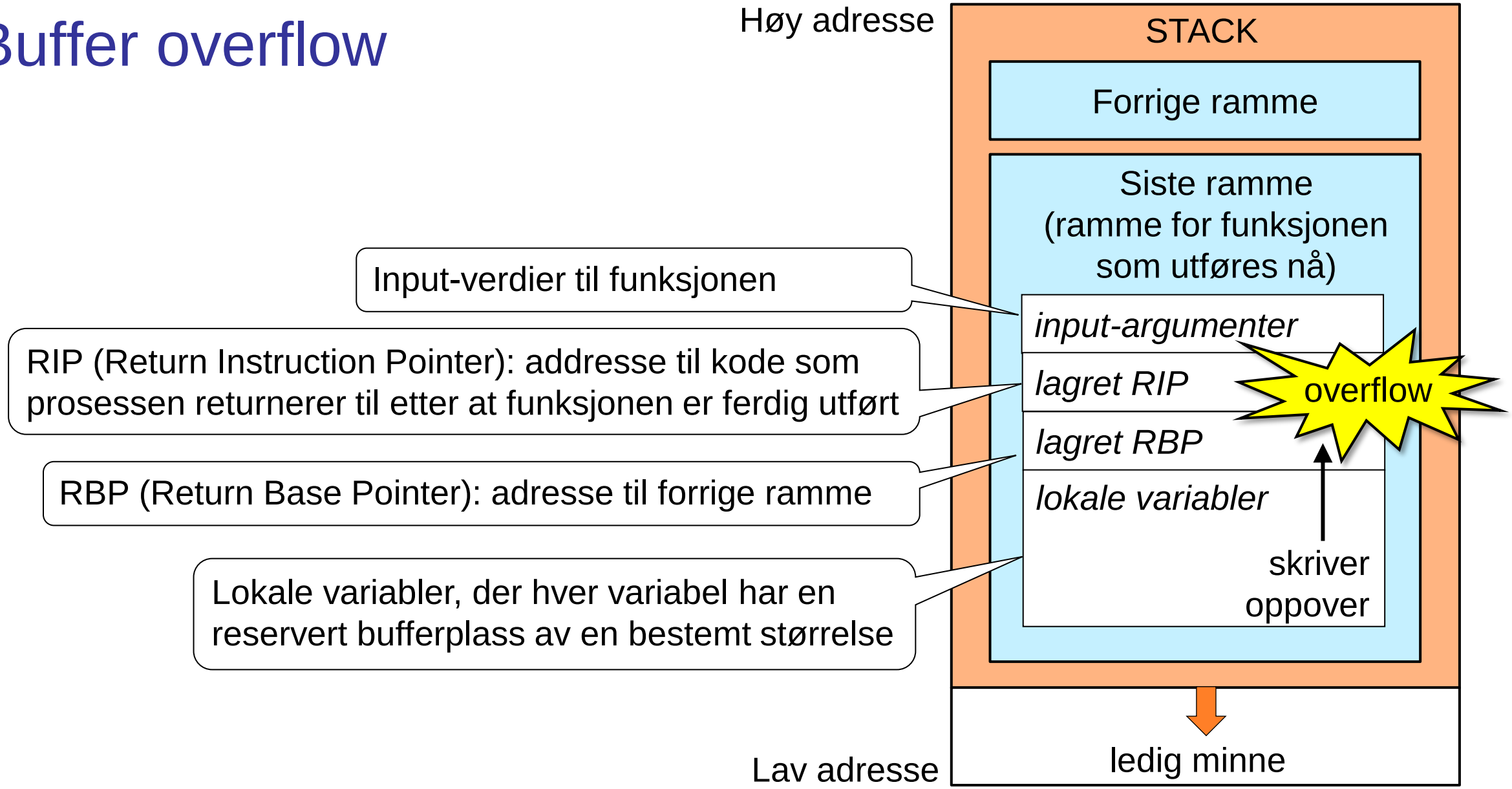
- Buffer overflow er en klassisk sårbarhet
  - Utnyttet av f.eks. Morris-worm og SQL-slammer
- Går i prinsippet utpå at minnet overskrives og blir korrupt som får angriper til å kjøre egen kode
- Disse vil ta kjøres med samme privileger som det opprinnelige programmet
  - Er sårbarheten i root/admin program vil angriper kunne kjøre sin kode med de privelegiene
- Det er flere måter å oppnå en buffer overflow og vi vil fokusere på en stack-basert versjon
- Dette krever litt innsikt i hvordan (virtuelt) minne for en prosess fungerer

Virtuelt minne for en prosess

- Hver prosess har et eget virtuelt minneområde på 4 GB.
- Virtuelle minneadresser oversettes av OS til fysiske minneadresser før de leses og skrives i system-minnet.
- Selv om alle prosessene har samme virtuelle adresseområde, har de i virkeligheten separate fysiske adresseområder.
- Dette prinsippet gjør at en prosess ikke har tilgang til fysiske adresser for andre prosesser, kun til sitt eget fysiske adresseområde, indirekte gjennom oversettelsen fra virtuelle til fysiske minneadresser.
- Betyr også at (virtuelle) adresser vil være like hver gang man kjører (finnes dog tiltak mot dette)



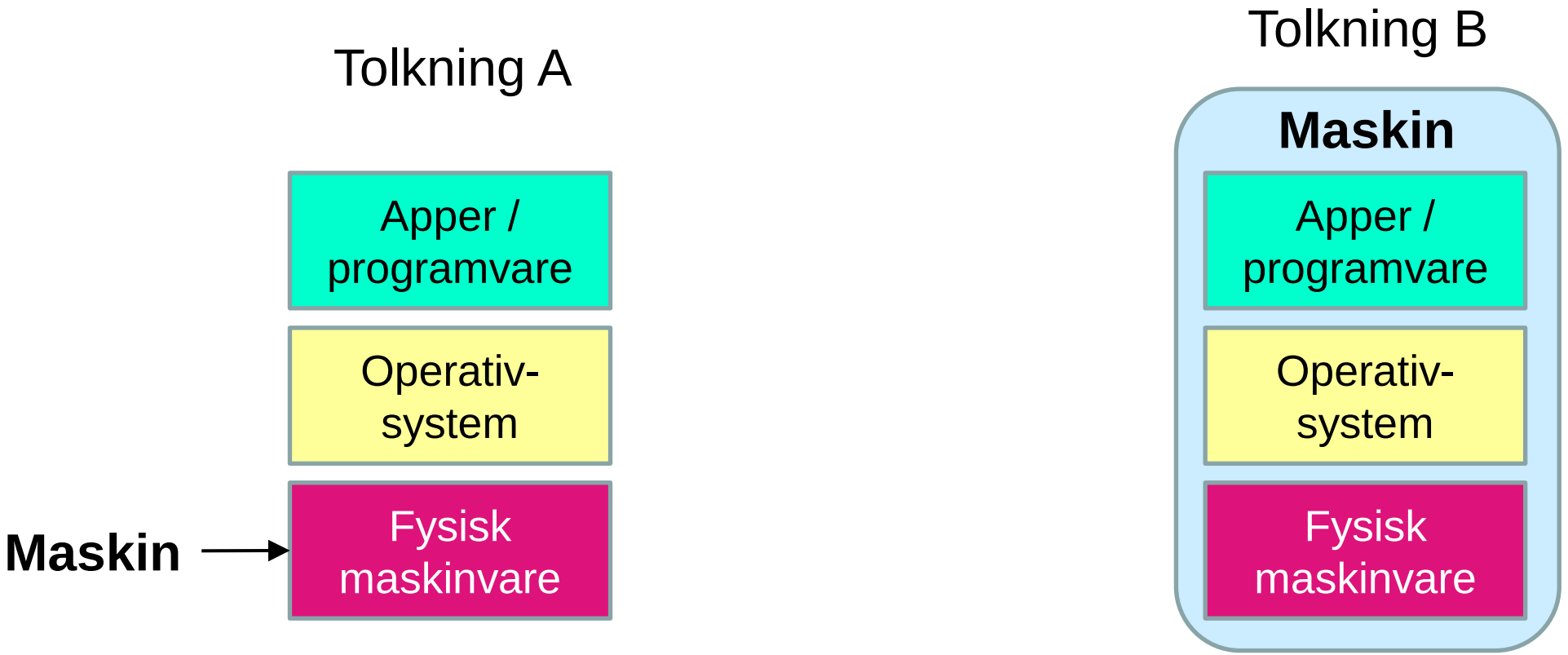
Buffer overflow



Buffer overflow – mottiltak

- No Execute (NX): OS tillater ikke å kjøre kode på stacken
- Stack canaries: tilfeldig verdi lagret annet sted som sjekker om data har blitt skrevet over
- 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
- Skriv bedre kode (innebygd sikkerhet / DevSecOps)

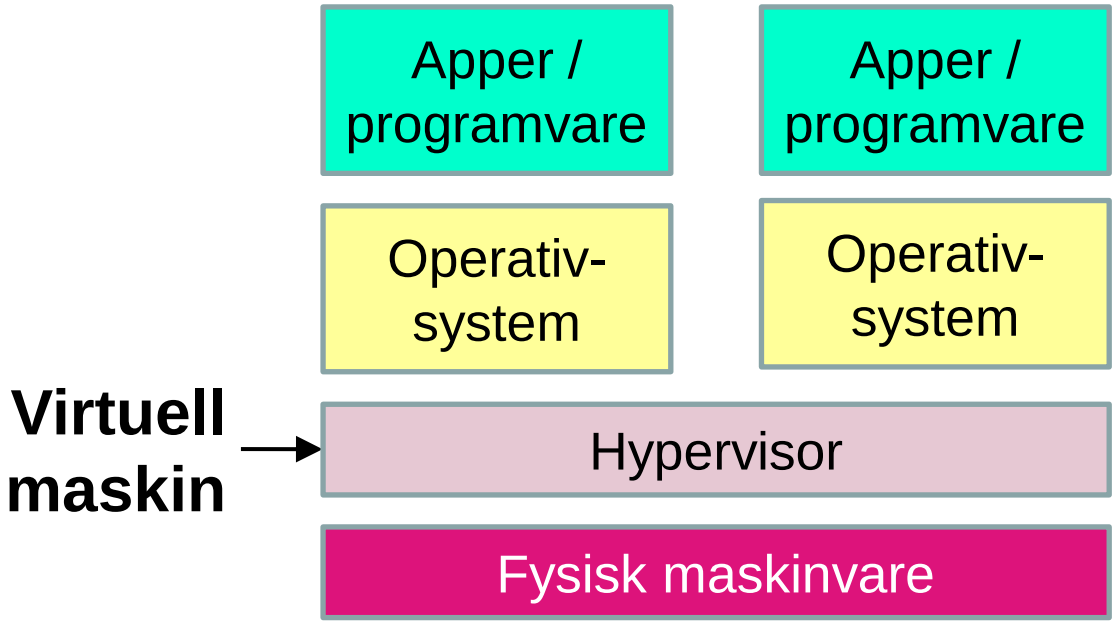
Ulike tolkninger av begrepet “maskin”



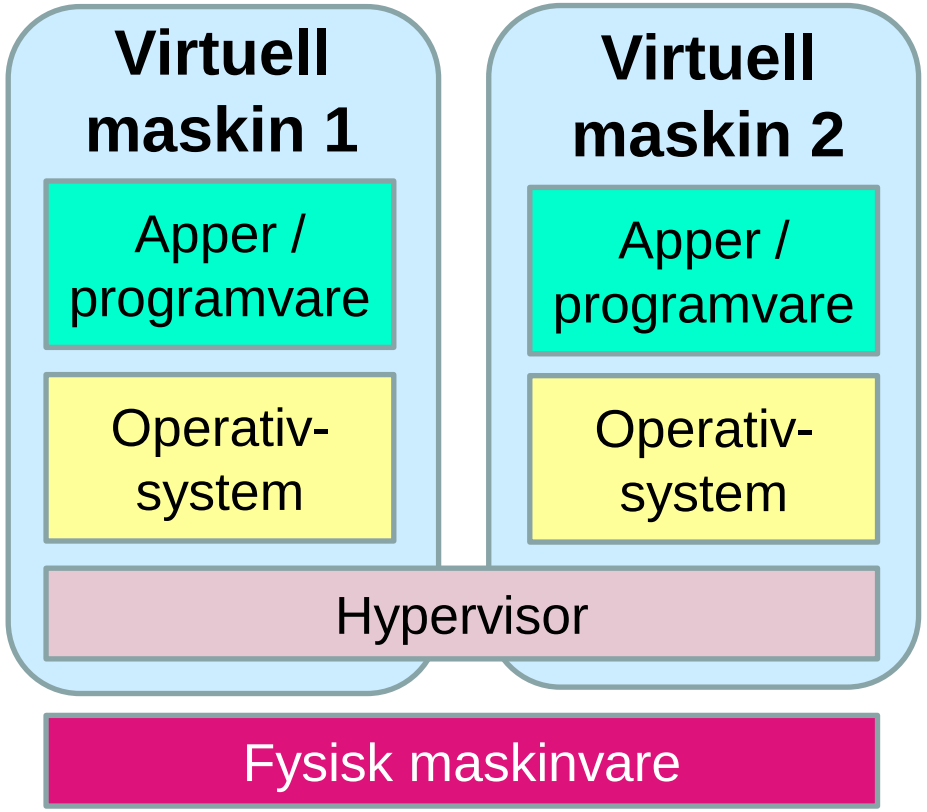
- «Maskin» kan bety maskinvare eller datamaskin.

Ulike tolkninger av begrepet “virtuell maskin”

Tolkning A:
Virtuell maskinvare

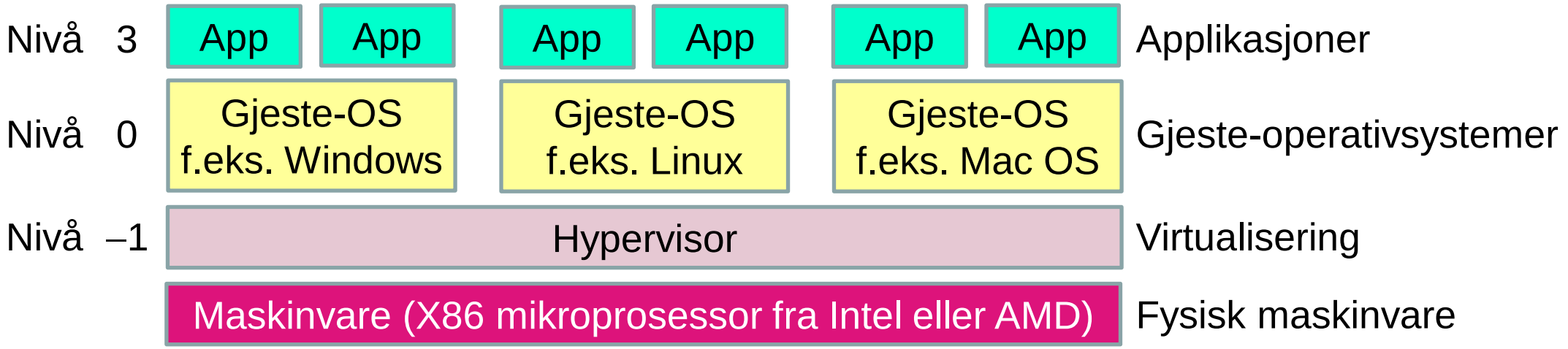


Tolkning B:
Virtuell datamaskin



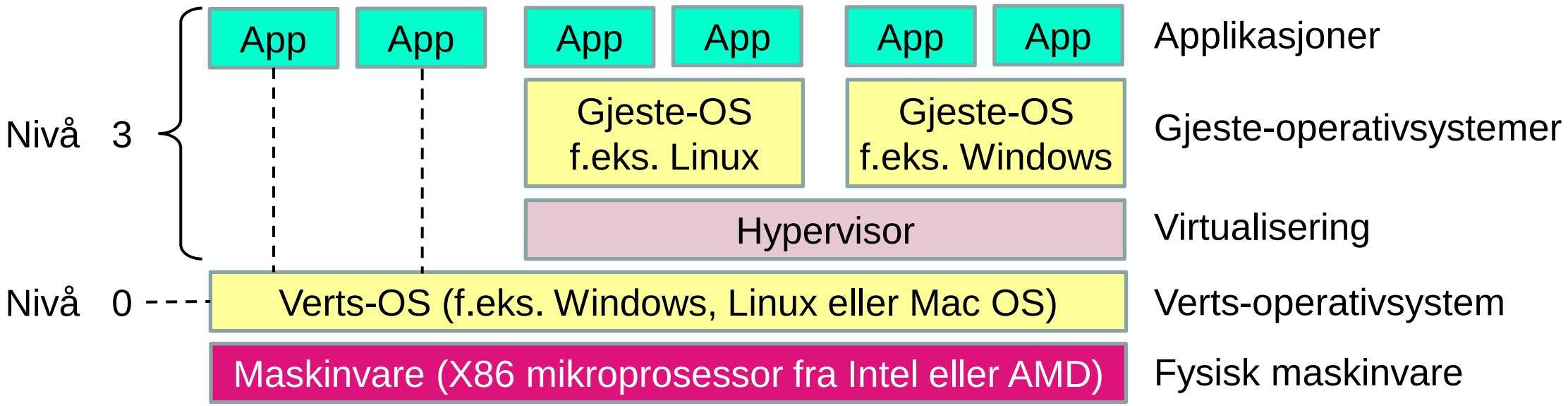
- Tolkning B er mest vanlig.

Type 1 virtualisering (native)



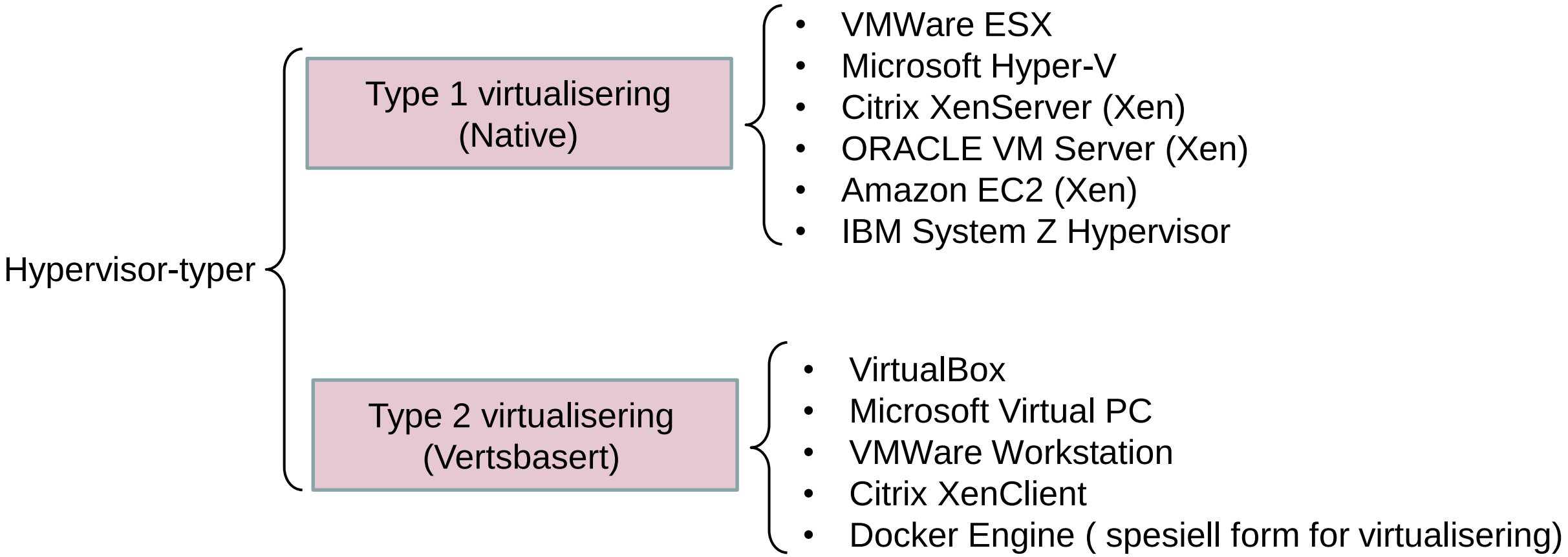
- Type 1 virtualisering er den mest direkte og mest effektive måten å kjøre virtuelle maskiner på.
- Hypervisor er programvare som spiller rollen som fysisk maskinvare.
- Type 1-virtualisering gir en logisk struktur på privilegienivåene ved at hypervisoren er mer privilegert enn gjeste-operativsystemene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

Type 2 virtualisering (vertsbasert)

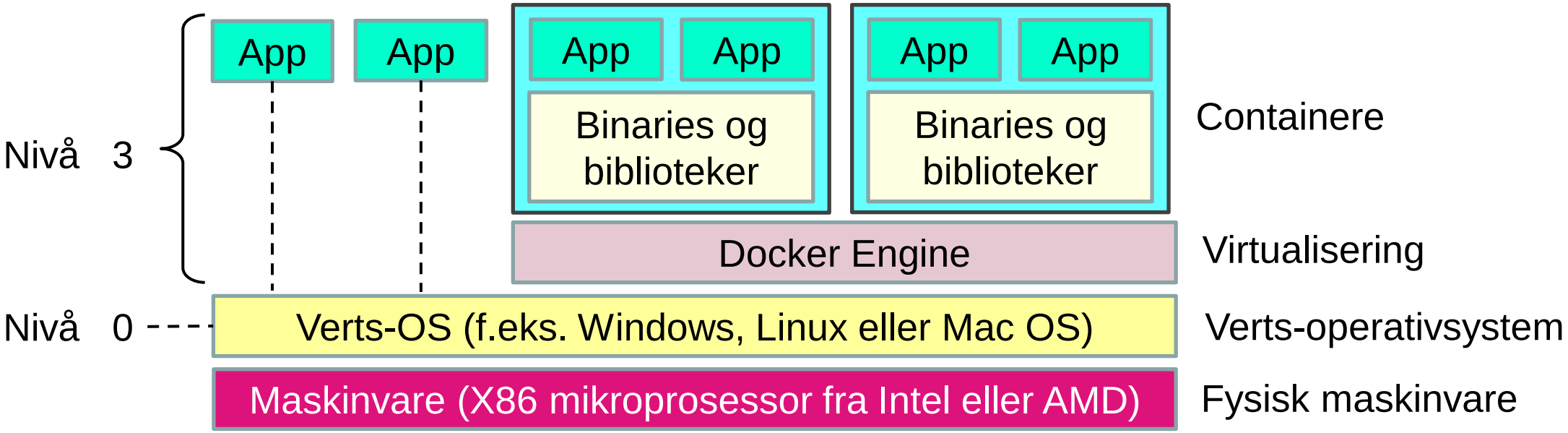


- Hypervisor-programmet blir installert som en vanlig applikasjon på vertsoperativsystemet.
- Enkelt for vanlige brukere å installere og bruke.
- Nivå -1 brukes ikke: hypervisor og gjeste-OS kjører som vanlige applikasjoner på nivå 3.
- Operativsystemer trenger å utføre privilegerte funksjoner som krever nivå 0 (eller nivå -1).
- Privilegerte funksjoner må kalles gjennom verts-operativsystemet, noe som forårsaker ekstra forsinkelse i prosesseringen. Derfor er type 2 en relativt ineffektiv virtualiseringsarkitektur.

Produkter for virtualisering



Docker Engine og containere (vertsbasert)



- Docker Engine gjør det mulig for containeriserte applikasjoner å kjøre uansett plattform.
- Docker Engine er en form for virtualisering som gjør gjeste-OS unødvendig.
- 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.

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

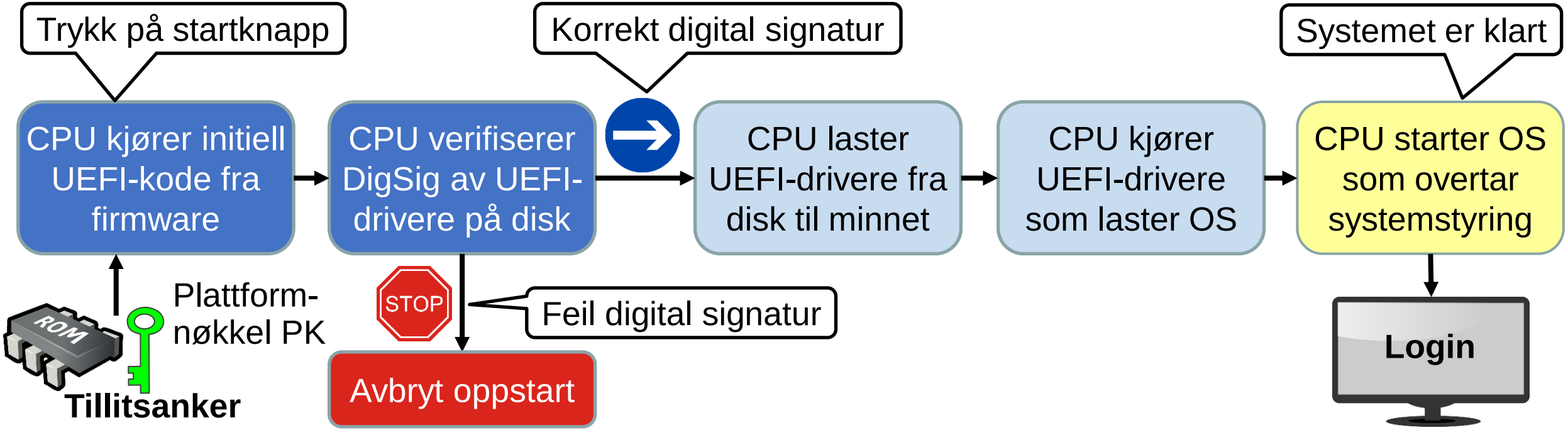
Tiltrodd beregning

- 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 (secure boot)* med UEFI.
  - *TPM (Trusted Platform Module)* er en koprosessor bygget inn i mange ulike systemer, TPM støtter tre spesifikke sikkerhetsfunksjoner: 1) sikker oppstart (secure boot), 2) attestering av sikker tilstand til tredjeparter (remote attestation), 3) disk-kryptering (sealed storage). Sikker oppstart med TPM er forskjellig fra sikker oppstart med UEFI.

  - *TEE (Trusted Execution Environment)* betyr at data og beregninger til en prosess beskyttes med maskinvareteknologi i mikroprosessoren, ikke bare av sikkerhetsfunksjoner i operativsystemet. Intel SGX (Software Guard Extensions) er et eksempel på slik teknologi.
  - *Manipuleringsbestandig fysisk innkapsling* er vanskelig å trenge gjennom for en angriper. En innkapsling er manipuleringssikker hvis den i tillegg kan detektere forsøk på fysisk manipulering, og eventuelt automatisk slette sensitiv informasjon som kryptonøkler. Et eksempel er IBM 4765 Secure Coprocessor

Sikker oppstart med UEFI (*Unified Extensible Firmware Interface*)



- UEFI erstatter BIOS i moderne computere. Styrer oppstartsekvensen.
- Programmoduler for oppstart er digitalt signert av computerleverandøren.
- UEFI-kode i ROM er usignert men antas å være korrekt, og initierer oppstartsekvensen.
- Programmoduler som lastes sjekkes for korrekt digital signatur med Platform Key.
- Hvis en feil signatur detekteres vil oppstartsekvensen avbrytes.

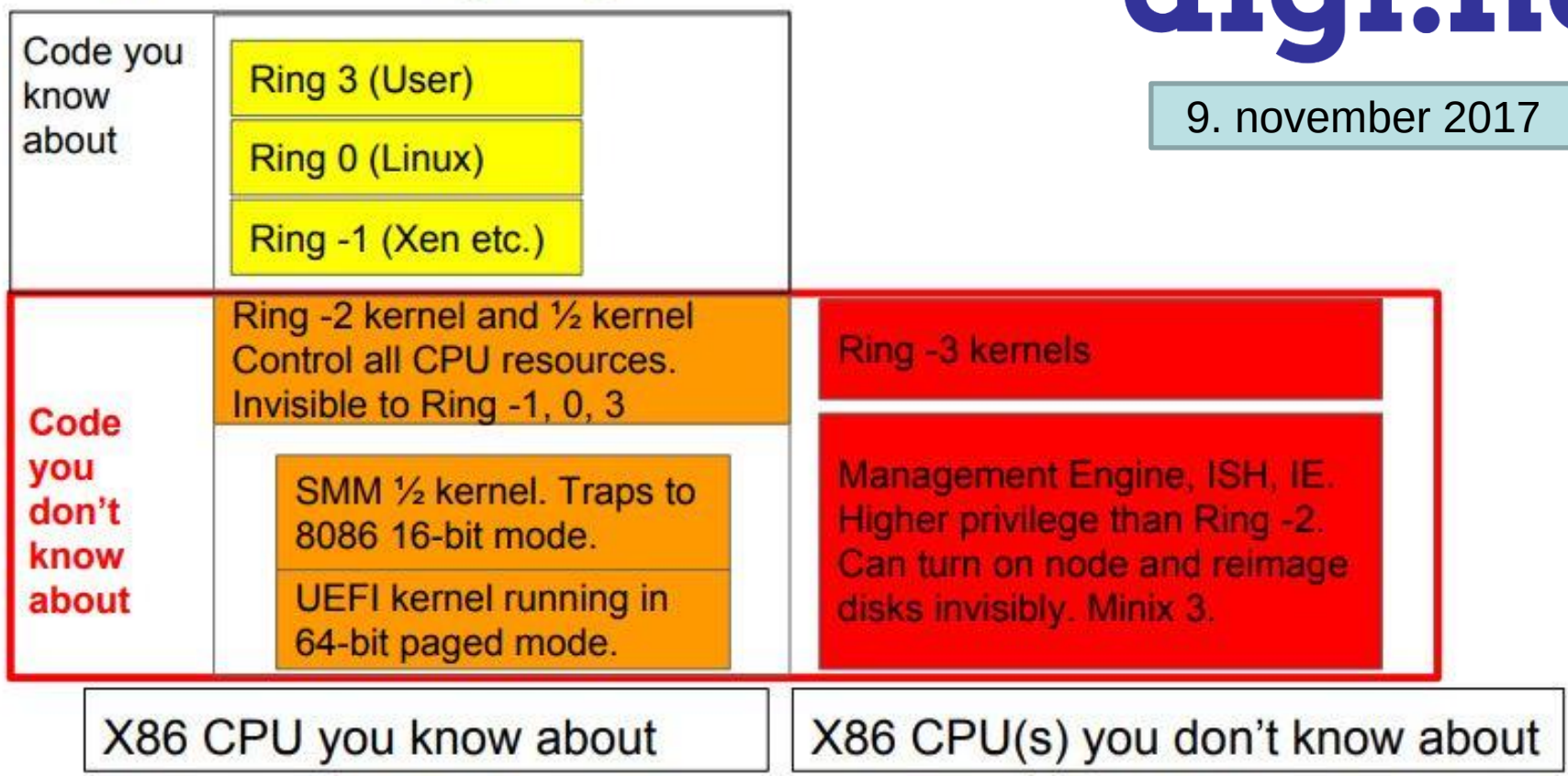
Sidekanaler og skjulte kanaler

- En *sidekanal* er en utilsiktet kanal som avgir informasjon som skyldes den fysiske implementering av et system
- Eksempel på sidekanaler er: tidsforbruk for instruksjon i CPU kan si noe om verdi (f.eks. av enkelte bits i krypteringsnøkkel), strømforbruk, lyd, stråling
- En *skjult kanal er en* mekanisme som ikke er designet for kommunikasjon, men som misbrukes (med vilje) for å overføre informasjon (på en måte som bryter med sikkerhetspolicy)
- Meltdown og Spectre er eksempler på sidekanaler i mikroprosessorer
  - Moderne prosessorer «gjetter» (mulige) neste instruksjoner (selv om de kan bryte sikkerhetspolicyer), og data lever fortsatt i cache selv om de ikke velges
  - Forsinkelse kan forårsake informasjonslekkasje mens sårbarheten kan få andre prosesser til å lese dens data

Det er ikke Windows, Linux eller MacOS som har den egentlige kontrollen over pc-en din

Hørt om Minix, det skjulte operativsystemet til mange Intel-baserte pc-er?

The operating systems



digi.no

9. november 2017

Intel ME (Management Engine)

- *Intel ME (Management Engine)* er et lite subsystem med lavt strømforbruk som er integrert med Intel mikroprosessorer.
- Intel ME kjører MINIX-OS, som er aktivt til enhver tid så lenge mikroprosessoren har strøm.
- Intel ME har full tilgang til systemmaskinvare, inkludert systemminne, skjerm, tastatur, kamera, mikrofon, periferiutstyr og nettverk.
- *Intel AMT (Advanced Management Technology)* er en løsning for fjernadministrasjon av arbeidsstasjoner. Den kan brukes på servere og klient-computere som kjører Intel-prosessorer. Brukere av Intel AMT er vanligvis store organisasjoner, ikke hjemmebrukere.
- AMT må være aktivert i UEFI/BIOS for å kunne brukes, og er deaktivert som standard når servere leveres til kunden.
- AMT fjernadministrasjon kan slå på, konfigurere, styre eller slette programvare på computere som kjører Intel-prosessorer.
- I motsetning til typiske plattformadministrasjonsløsninger som krever OS-støtte, fungerer AMT selv om ikke noe operativsystem (Windows, Linux, MacOS, hypervisor) er installert.

Slutt på presentasjonen