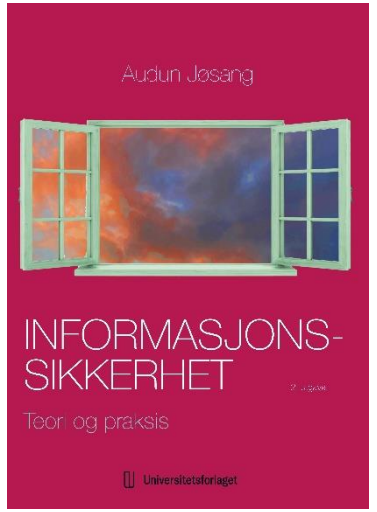
Kapittel 11:
Innebygd informasjonssikkerhet

Informasjonssikkerhet: Teori og praksis

Audun Jøsang

2. utg. 2023

Universitetsforlaget



Oversikt

- Innebygd informasjonssikkerhet
- Trusselmodellering og STRIDE
- Applikasjonssikkerhet og OWASP Top 10
- Sikker programvareutvikling og DevOps
- Sikkerhet i skyen

Innebygd informasjonssikkerhet og personvern

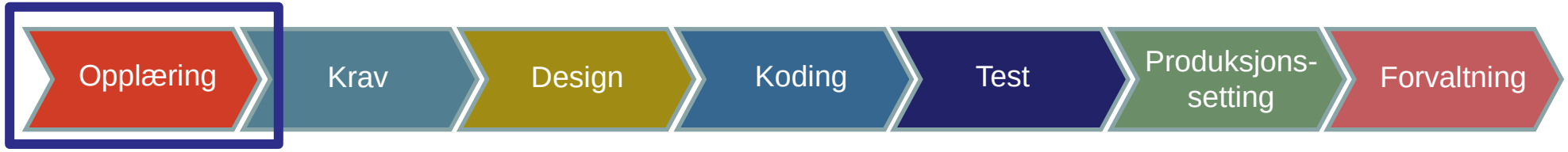
- «Innebygd» informasjonssikkerhet og personvern betyr at det tas eksplisitt hensyn til informasjonssikkerhet og personvern i hele livssyklusen til programvare og applikasjoner.
- Et viktig mål er å finne og redusere sårbarheter tidlig i utviklingsprosessen slik at det blir færre hendelser og sårbarheter å håndtere under drift.
- Microsoft så dette behovet tidlig og har vært ledende her og utviklet *Microsoft Security Development Lifecycle (SDL)*
  - Denne er nå en integrert del av programvareutviklingsprosessen hos Microsoft (og andre)
- Vi kan beskrive livssyklusen som en prosess av 7 faser:

Fase 1: Opplæring

Alle som deltar i utvikling og drift av digitale tjenester og applikasjoner skal ha basiskunnskap og forståelse for

- Informasjonssikkerhet og personvern
- Risikovurderinger
- Trusselmodellering

Dette er egentlig en selvfølge. Det ville være uforsvarlig å utdanne bygningsarkitekter og ingeniører uten å gi dem kunnskap om branntrygghet, fordi arkitekter og ingeniører da ville bygget brannfeller inn i våre bygninger. På samme måte er det uforsvarlig å utdanne informatikere og dataingeniører uten obligatoriske kurs om informasjonssikkerhet, fordi de ferdigutdannede da nødvendigvis ville bygget en sårbar IKT-infrastruktur.



Krystallklare signaler fra myndighetene

- Alle må ha kunnskap om digital sikkerhet.

… men det har ikke alltid vært slik

- Nasjonal strategi fra 2012 nevnte ikke opplæring i informasjonssikkerhet med et eneste ord.
- Stortingsmelding 18-2014 foreslo å spare penger ved at små fagområder kun skulle undervises ved ett lærested.
- Utdanningsdirektoratet forslo i 2014 at kun ett lærested skulle undervise informasjonssikkerhet.
- Fagmiljøene sa tydelig fra at informasjonssikkerhet må undervises overalt.

COMPUTERWORLD

15.08.2014

Audun Jøsang

Cybersikkerhet starter med it-utdanningen

https://www.cw.no/debatt-it-karriere-sikkerhet/cybersikkerhet-starter-med-it-utdanningen/622463

Endret visjon fra myndighetene om IT-sikkerhet i utdanningen

2014

Kunnskapsdepartementet

SAK (Samarbeid, Arbeidsdeling og Konsentrasjon)

IT-sikkerhet bør bare undervises ved ett eneste lærested !

2019

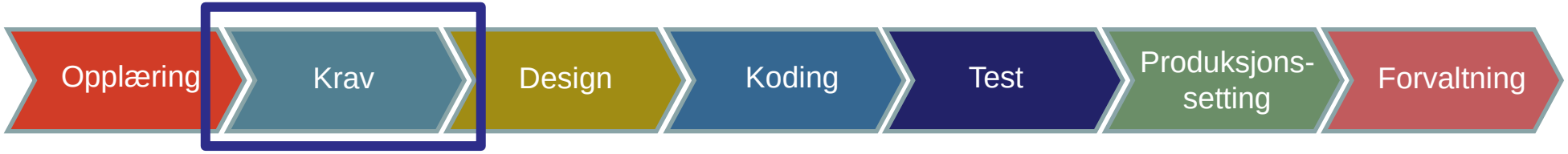
Kunnskapsdepartementet

Nasjonal strategi for digital sikkerhetskompetanse 2019

Digital sikkerhet skal undervises overalt !

Fase 2: Krav om informasjonssikkerhet og personvern

- Kilder til krav om informasjonssikkerhet og personvern:
  1. Krav som følger av god praksis for adekvat sikkerhet i applikasjoner og forretningsprosesser.
  2. Krav om å begrense sikkerhetsrisiko til et akseptabelt nivå.
  3. Juridiske lovbestemte, regulatoriske og kontraktsmessige krav til informasjonssikkerhet og personvern.
- Sikkerhetskrav må kontinuerlig oppdateres for å gjenspeile endringer i nødvendig funksjonalitet, trusselslandskap, lover, forskrifter, reguleringer,…
- Bør gjøres tidlig i utviklingsløpet (innledende design- og planleggingsfasen)
- OWASP Top 10 / ASVS definerer beste praksis for krav om applikasjonssikkerhet



OWASP

- Open Web Application Security Project (OWASP)
  - Ideell organisasjon med mål om å forbedre sikkerheten til applikasjoner og tjenester på internett
  - Gjennom råd, veiledning og verktøy
  - Involverer bedrifter, utdanningsinstitusjoner og enkeltpersoner fra hele verden
  - Flere parallelle prosjekter
- OWASP ASVS (Application Security Verification Standard)
  - Standard med mål om å definere beste praksis for sikker utvikling og testing
- OWASP Top 10
  - Rangerer de 10 mest kritiske sikkerhetsrisikoene for nettapplikasjoner
  - Gir råd om hvordan sårbarhet og risiko kan reduseres
  - Oppdateres med noen års mellomrom
  - Siste revisjon kom ut i 2021, forrige versjon fra 2017
  - Kan være en veldig god kilde for å komme i gang med trusselmodellering

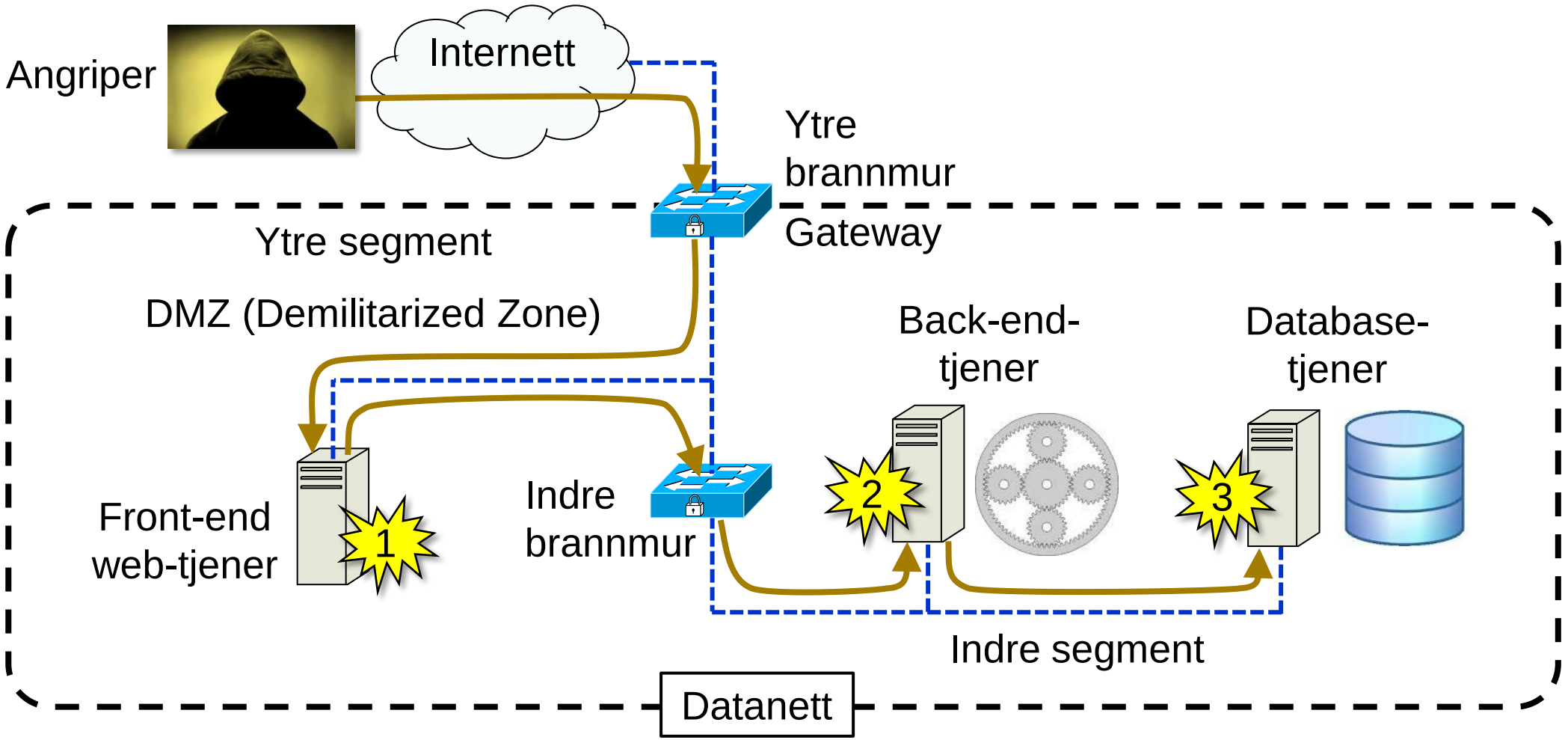
OWASP Top 10 (2021)

1. **Brudd på tilgangskontroll**
   - Angripere utnytter feil i hvordan tilgangskontroll er håndhevet
   - Kan f.eks. være å lese eller endre andre brukeres data
2. **Kryptografiske feil**
   - Feil relatert til krypto som ofte medfører at sensitiv data eksponeres eller kompromittering av system
3. **Injeksjon**
   - Manipulert input-data sendes til en applikasjon som en del av en forespørsel/kommando som lurer applikasjonen til å utføre utilsiktede eller uautoriserte handlinger
   - Flere varianter, inkludert SQL-injeksjon
   - Cross-site scripting (XXS) er (i 2021 versjonen) en del av denne kategorien
4. **Usikkert design**
   - Risiko relatert til feil i design
5. **Feilkonfigurert sikkerhet**
   - Risiko som skyldes feil i konfigurering
   - F.eks. usikker standardkonfigurering som ikke endres, feilmeldinger som avslører sensitiv informasjon

OWASP Top 10 (2021)

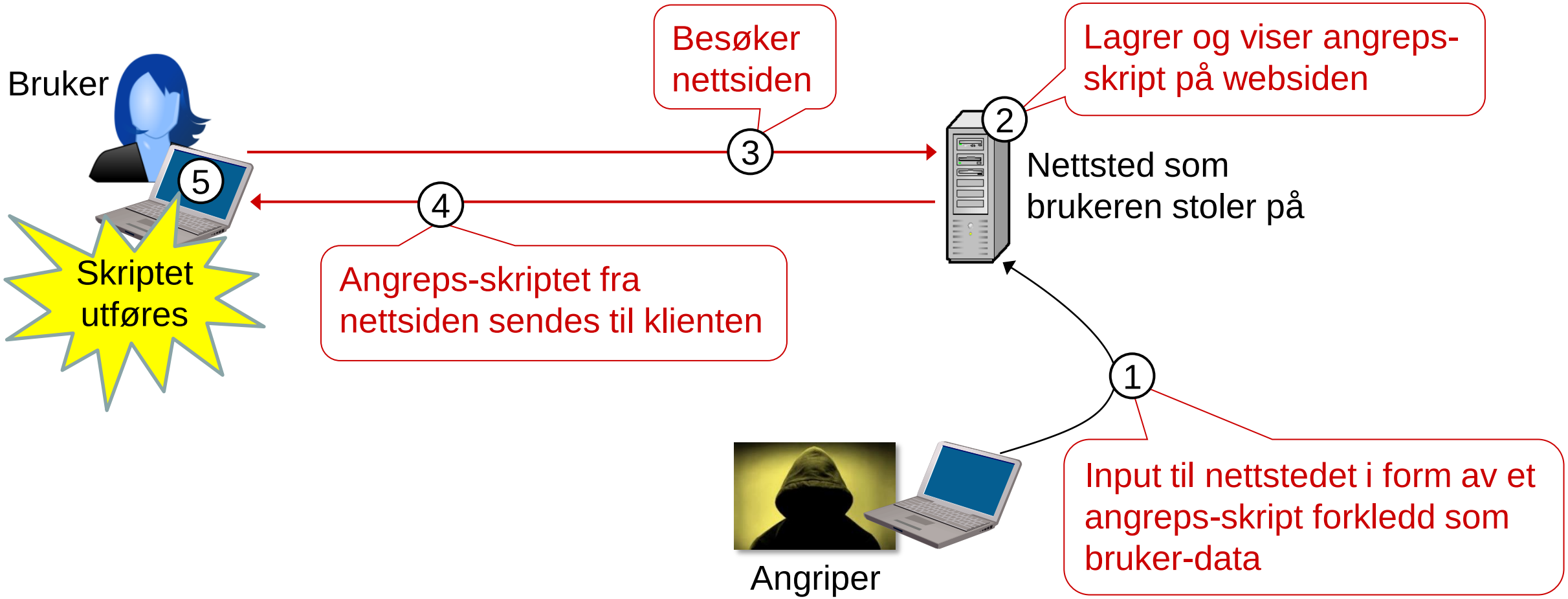
6. **Sårbare og utdaterte komponenter**
   - Utnyttelse av sårbar/utdatert komponent
   - F.eks. gjennom bruk av (eksternt) bibliotek eller programvaremodul som kjører med samme privileger som applikasjonen hvor de brukes
7. **Feil i identifisering og autentisering**
   - Sjekk av brukers identitet, autentisering eller styring av økt er ofte implementert feil
   - F.eks. dårlige og standard passord, reset av passord, økt-identifikator i URL, …
8. **Feil i (data og programvare) integritet**
   - F.eks. applikasjon bruker modul fra ukjent kilde eller auto-oppdatering uten god nok sjekk av integritet
9. **Utilstrekkelig logging og overvåking**
   - Medfører at man ikke kan detektere og håndtere brudd
10. **Server side request forgery (SSRF)**
    - Angriper får tjener-applikasjon til å gjøre forespørsel til et domene spesifisert av angrep
    - F.eks. kan det medføre at angriper kan lese/oppdatere interne ressurser.

Applikasjonssikkerhet – forhindre angrep gjennom apper



- OWASP Top 10 beskriver typiske web-sårbarheter, og hvordan de kan forhndres

Injeksjon: XSS (Cross-Site Scripting)



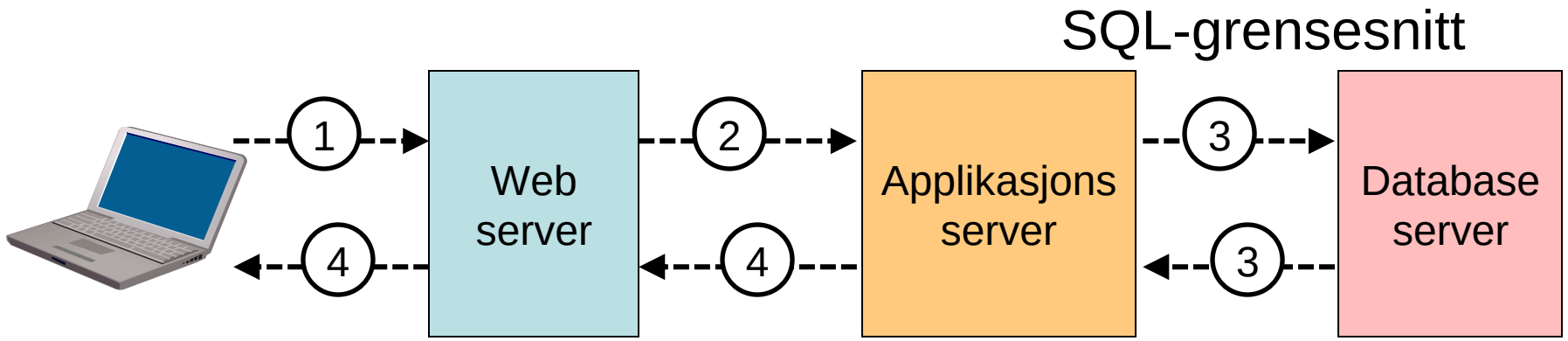
- XSS benytter applikasjonssårbarhet på server for å angripe på klienten

Stored XSS

- Mange web-applikasjoner lar brukere sende info som lagres vedvarende på server (i database, filsystem, …) og vises senere til alle brukere som besøker nettstedet.
  - Typisk eksempel: sosiale nettverk, meldingstjenester, oppslagstavler osv..
- Angriperen laster opp data som inneholder skadelig skript til serveren.
- Hver gang den sårbare nettsiden besøkes, vil det skadelige skriptet lastes ned til nettleseren og kjøres i klientnettleseren.
- Angriperen trenger bare å injisere skript én gang.

SQL i web-stacken

1. Hent input fra et nettskjema via HTTP-metoder som POST eller GET, og send det til en applikasjon på serversiden.
2. Søkeprosess åpner forbindelse til SQL-databasen.
3. Spør database med SQL og henter svar.
4. Behandler SQL-svar og sende resultater tilbake til brukeren.



Hva er SQL?

- Structured Query Language: Grensesnitt til relasjonsdatabasesystemer.
- Gjør det mulig å sette inn, oppdatere, slette og hente data i en database.
- SQL er en ANSI og ISO Standard som er my brukt mye i webapplikasjoner.
- Eksempel:

```
SELECT * FROM users WHERE userid = ‘$userid’ AND password = ‘$password’
```

- Kommandoen henter info om bruker med oppgitt bruker-ID og riktig passord, som kan resulterer i innlogging.

Hva er SQL-injeksjon?

- Databasesystemet feiltolker inputdata
  - Angriper skjuler SQL-kommandoer som inputdata
  - Forkledde SQL-kommandoer = 'injiserte' SQL-kommandoer
- Med SQL-injeksjon kan en angriper få full kontroll over database
  - uansett hvor sikkert det underliggende systemet er
  - uansett hvor godt brannmuren er konfigurert,
- Sårbarhet eksisterer når nettapplikasjonen ikke klarer å filtrere inputdata før den sendes til databasen
- Dette er en sårbarhet i webapplikasjonen, ikke i SQL-database.

Eksempel SQL injeksjon

- F.eks., en innloggingsside som forventer at bruker oppgir bruker-ID og password
- Angriper skriver i feltet for bruker-ID: `' OR 1=1--`
- Og lar feltet for passord stå åpent
- Resultatet av SQL-kommandoen er:

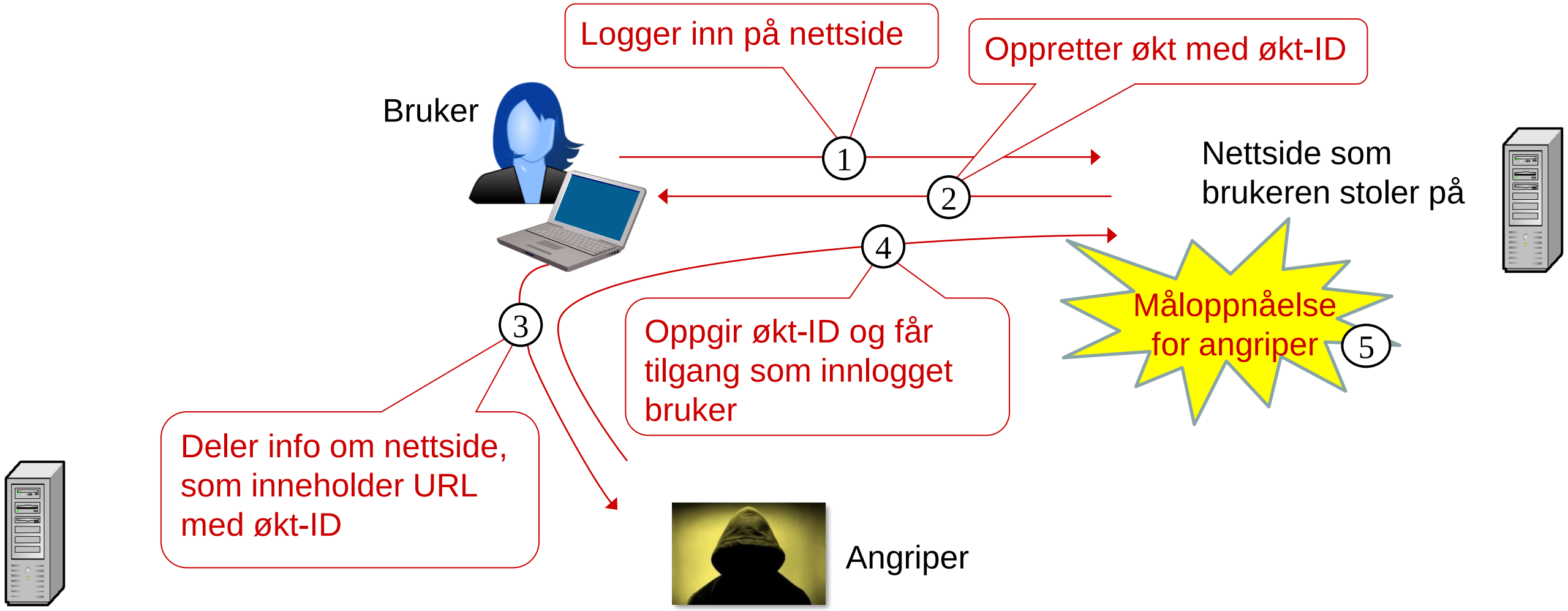
```
SELECT * FROM users WHERE userid = '' OR 1=1-- ' AND password ='
```

- 1=1 er alltid TRUE slik at “WHERE” alltid vil stemme, uansett passord.
- - - gjør at SQL serveren ignorerer alt som følger etter
- Resultatet er at applikasjonen returnerer info om alle brukere

Injeksjon: Sårbarhet og sikkerhetstiltak

- Injeksjon er en generell angrepsteknikk som går ut på å legge skadelig data i systemer og applikasjoner slik at disse feiler
  - Angriper kan styre systemer og applikasjoner
- Følg OWASP Top 10 anbefalinger for å hindre injeksjon
- Generelt må all input fra bruker filtreres, og skadelig data fjernes
- Tiltak mot SQL-injeksjon kan gjøres på database-server
  - Parametriserte SQL-kommandoer forhindrer SQL-injeksjon fordi verdiene for parameterne legges til etter at malen er kompilert med en fast struktur. Skadelige input-verdier ville resultert i en SQL-kommando med forventet struktur, men med parametere som ikke ville gitt noen treff i databasen.

Feil i identifisering og autentisering: Tyveri av økt-ID



Stjålet økt-identifikator: Sårbarhet og sikkerhetstiltak

- Brukerautentisering gir ikke nødvendigvis kontinuerlig autentiseringssikkerhet
  - Brukerautentisering er bare på ett tidspunkt
- Usikker implementering av økt med en statisk økt-ID som sendes i URL
  - Dette kan dessverre misbrukes
  - Hvem som helst kan stjele økten bare ved å kjenne økt-ID
- Følg OWASP Top 10 anbefalinger for økt-ID
  - Eksempler på kontroller for økt-ID:
  - Koble økt-ID til IP-adresse eller TLS-økt-ID

Fase 3: Sikker design, og Fase 4: Sikker koding

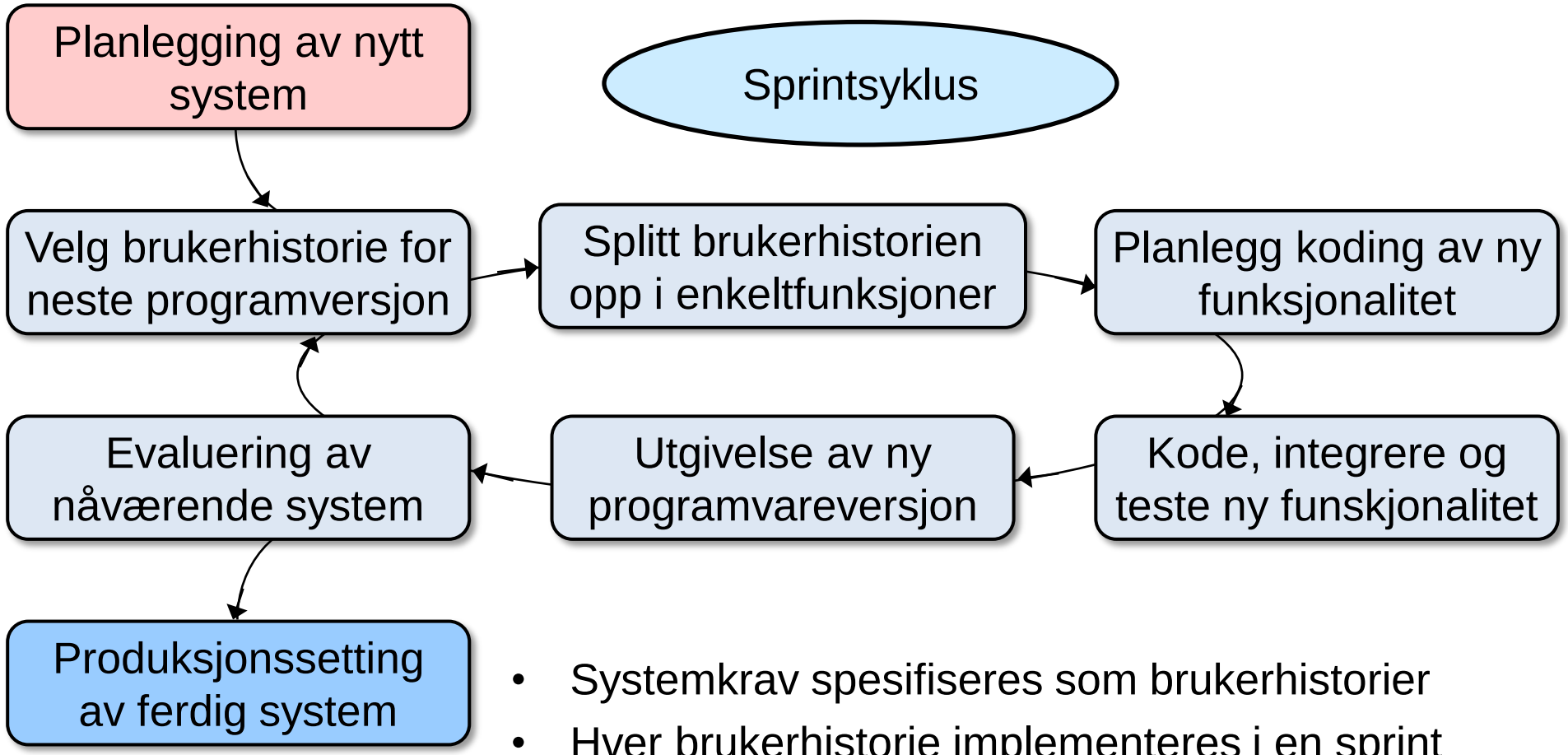
3. Sikker design

- Viktig del å spesifisere «sikre funksjoner», som er godt designet med hensyn til sikkerhet.
- Sikkerhetsfunksjoner som krypto, autentisering, logging etc er viktig. Designer må ha god kompetanse på disse.
- Trusselmodellering (mer om dette senere) er en viktig del her for å unngå sårbarheter

4. Sikker koding

- Målet med sikker koding er å unngå at sårbarheter bygges inn i systemet under koding og at sikkerhetsfunksjoner fungerer i henhold til krav og design
- Moderne utvikling benytter i stor grad tredjepartskomponenter, og det er viktig å forstå innvirkningen disse kan ha på sikkerheten. Usikre komponenter skal ikke brukes.
- Verktøy for kodeskanning og statisk analyse, gjennomgang av kode samt bruk av «sikre» programmeringsspråk er viktig her.

Smidig programvareutvikling



- Systemkrav spesifiseres som brukerhistorier
- Hver brukerhistorie implementeres i en sprint
- Fortsetter så lenge det er igjen brukerhistorier
- Systemet er ferdig når alle brukerhistorier er laget

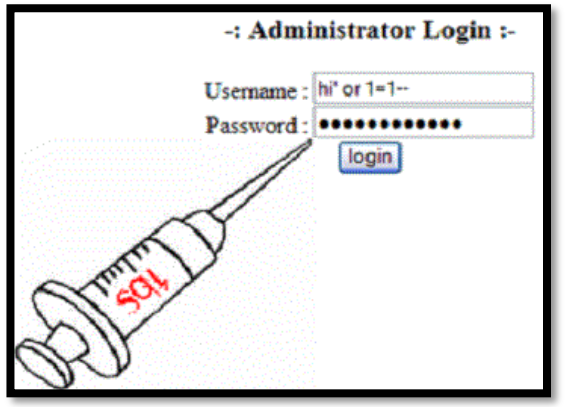
Sikker smidig programvareutvikling

- Sikker smidig programvareutvikling har tilleggsaktiviteter
  - … ved oppstart av prosjektet
  - … i hver sprintsyklus
  - … ved avsluttende evaluering av system
- «Sikker smidig» blir nødvendigvis litt mindre smidig

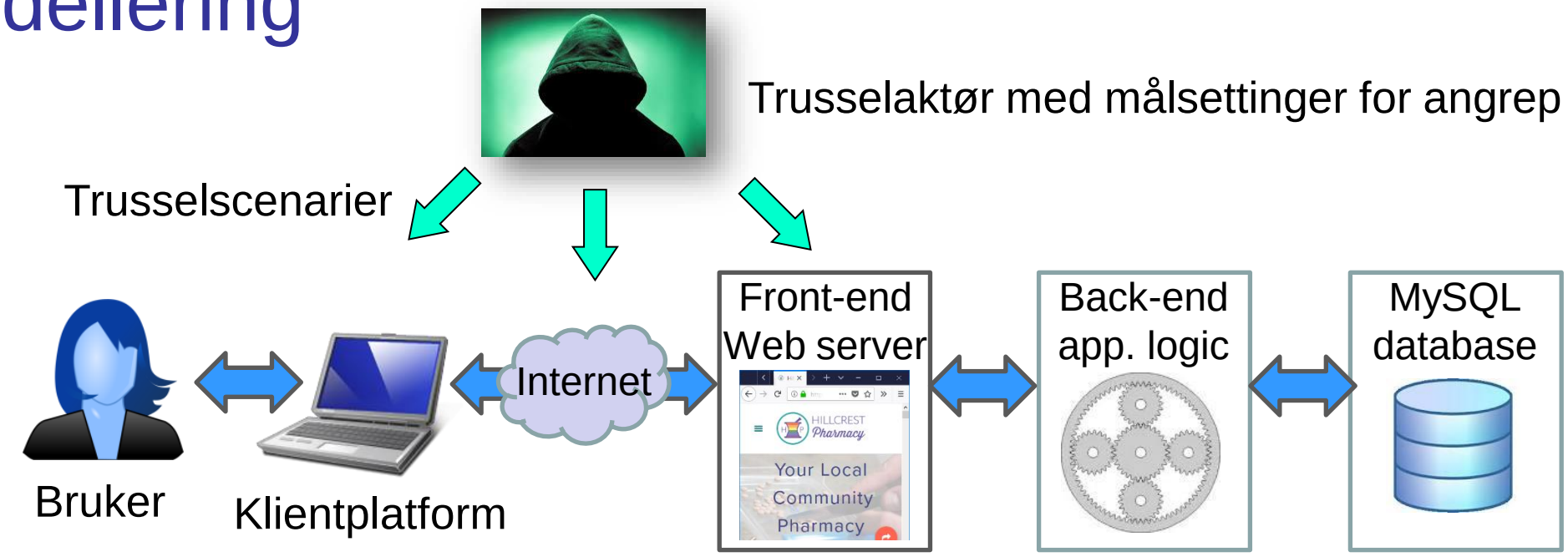
Funksjonell og ikke-funksjonell sikkerhet

- Funksjonell sikkerhet:
  - Sikkerhetskrav som utgjør en egen brukerhistorie
  - Sikkerhetsfunksjon som er nødvendig for brukerhistorien
  - Sikkerhetsfunksjon som gir en «synlig» tjeneste til brukere
  - F.eks.: Brukerautentisering

- Ikke-funksjonell sikkerhet:
  - Sikkerhetskrav som ikke er en egen brukerhistorie
  - Sikkerhetsfunksjon som er unødvendig for brukerhistorien men nødvendig for å hindre trusler
  - Sikkerhetsfunksjon som er «usynlig» for brukeren
  - F.eks.: Inputvalidering mot SQL-injection og XSS



Trusselmodellering



- Trusselmodellering er å identifisere, analysere og beskrive relevante angrepsscenarier.
- I sikker smidig programvareutvikling skal trusselmodellering og en enkel risikovurdering utføres som del av hver sprint.
- Tenk: Hvordan kan denne nye funksjonen misbrukes eller angripes? Hvilke verdier kan bli skadet? Hvilke konsekvenser kan det få?
- Stopp eller reduser trusselen (fjern sårbarheter) under sprinten.

Brukerhistorie og “Use Case”

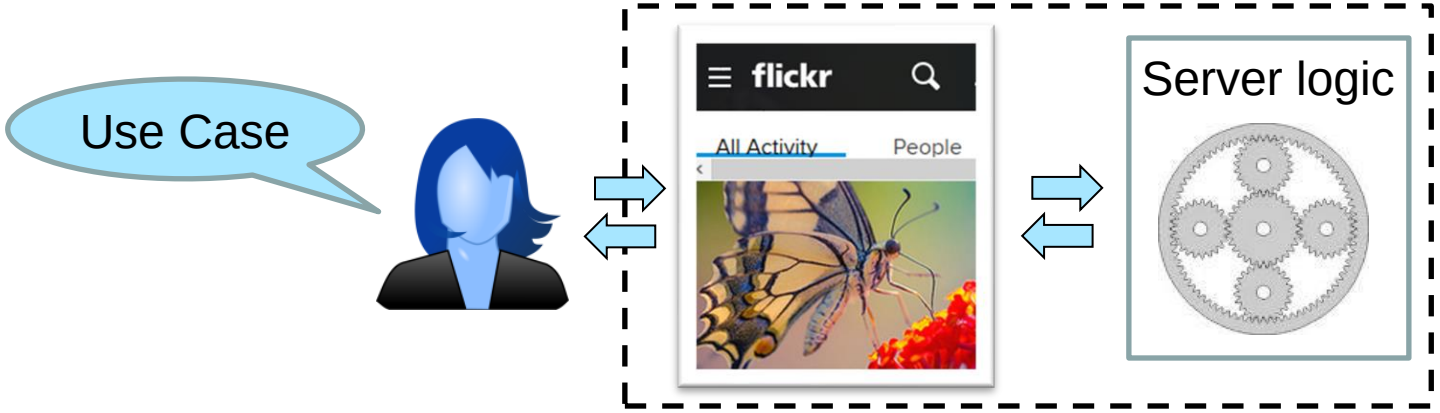
Brukerhistorie – Sett fra brukerperspektiv:

Som [bruker] ønsker jeg [funksjon] for å nå [resultat].

Eksempel: *Som Flickr-bruker, ønsker jeg å kunne definere tilgangskontroll, slik at jeg kan velge hvem som kan se de forskjellige bildene jeg har lagt ut.*

Use Case – Sett fra designerens perspektiv:

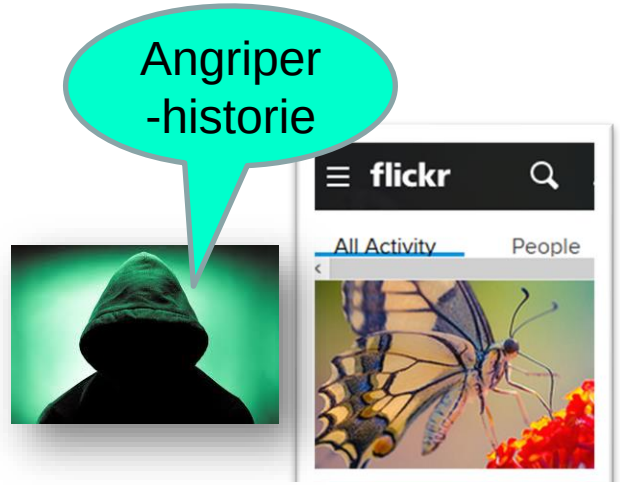
Beskrivelse av et sett med interaksjoner mellom et system og én eller flere aktører (der en "aktør" kan være en bruker eller et annet system).



Angriperhistorie og "Misuse Case" (Angripers målsetting og Trusselscenario)

Angriperhistorie – Angripers målsetting:

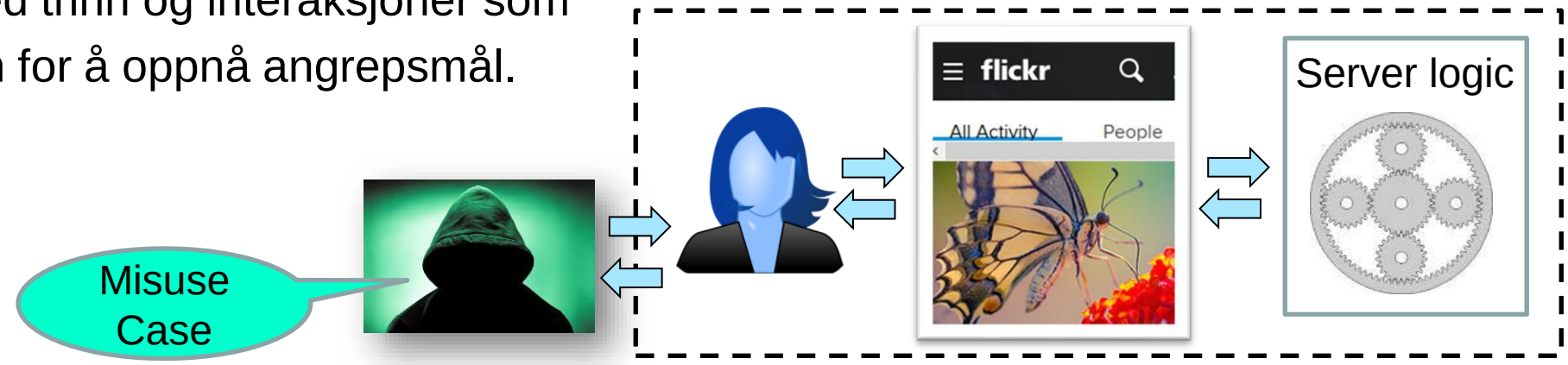
Som [angriper] ønsker jeg [funksjon] for å nå [resultat]. Eksempel:
Som angriper ønsker jeg å hacke Flickr-kontoer for å stjele private bilder og personlig info.



Misuse Case (Trusselscenario)

Sett fra trusselaktøren sitt perspektiv:

Beskrivelse av et sett med trinn og interaksjoner som må utføres av angriperen for å oppnå angrepsmål.



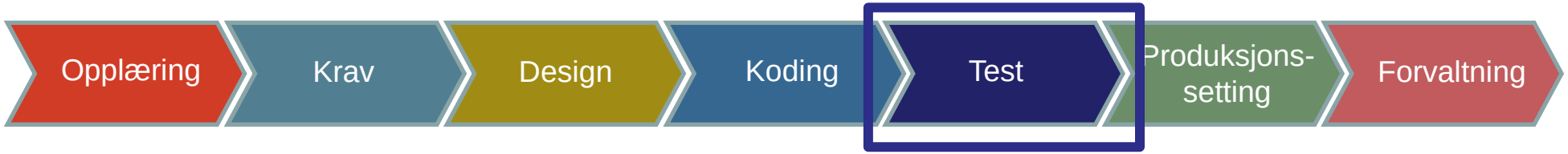
STRIDE Trusselmodellering for programvareutvikling

- Det finnes flere rammeverk for trusselmodellering.
- STRIDE er et populært rammeverk utviklet av Microsoft, hvor navnet kommer fra forbokstavene til 6 kategorier
- Hensikten er at disse kategoriene skal hjelpe å identifisere trusler og sårbarheter

Bokstav	Trusselkategori	Beskrivelse	Sikkerhetsbrudd
S	*Spoofing* Identitetstyveri	Kan en angriper få uautorisert tilgang ved å stjele en annens identitet?	Brudd på autentisitet
T	*Tampering* Tukling	Kan en angriper endre konfigurasjon eller data som prosesseres av systemet?	Brudd på integritet
R	*Repudiation* Benekting	Kan en angriper benekte misbruk fordi vi mangler spor og logger som peker ut angriperen?	Brudd på sporbar- og ubenektelighet
I	*Information disclosure* Datatyveri og -lekkasje	Kan en angriper få tilgang til konfidensielle og personlige data?	Brudd på konfidensialitet
D	*Denial of service* Tjenestenekt	Kan en angriper blokkere eller minske tilgjengeligheten til systemet?	Brudd på tilgjengelighet
E	*Elevation of privilege* Utvidede tilganger	Kan en angriper oppnå utvidede tilganger som en privilegert bruker?	Brudd på tilgangskontroll

Fase 5: Sikkerhetstesting

- Mål er å avdekke sårbarheter som ikke har blitt oppdaget i design- eller kodefasene.
- **Dynamisk testing/sårbarhetsanalyse** av den fullstendige programvaren sjekker funksjonalitet som blir synlig når alle komponentene er integrert sammen. Sjekker blant annet at bruker får tilgang til informasjon/funksjonalitet den skal (og ikke informasjon som bruker ikke skal ha tilgang til).
- **Penetrasjonstesting** går et steg videre og er et (autorisert) simulert angrep for å evaluere sikkerheten («etisk hacking» brukes ofte om dette). Skiller mellom
  - hvitbokstesting der angriper har informasjon om system på forhånd
  - svartbokstesting der angriper ikke har slik informasjon.
- **Fuzztesting** forsøker å fremprovosere feil i systemet ved å gi korrupte inputverdier (tilfeldig eller misformet data).



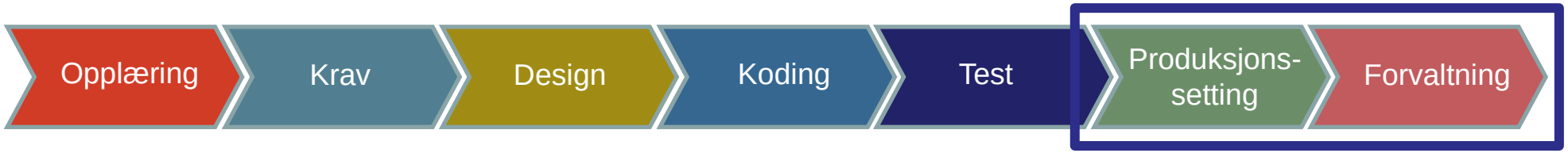
Fase 6: Produksjonssetting og Fase 7: Forvaltning

6. Produksjonssetting

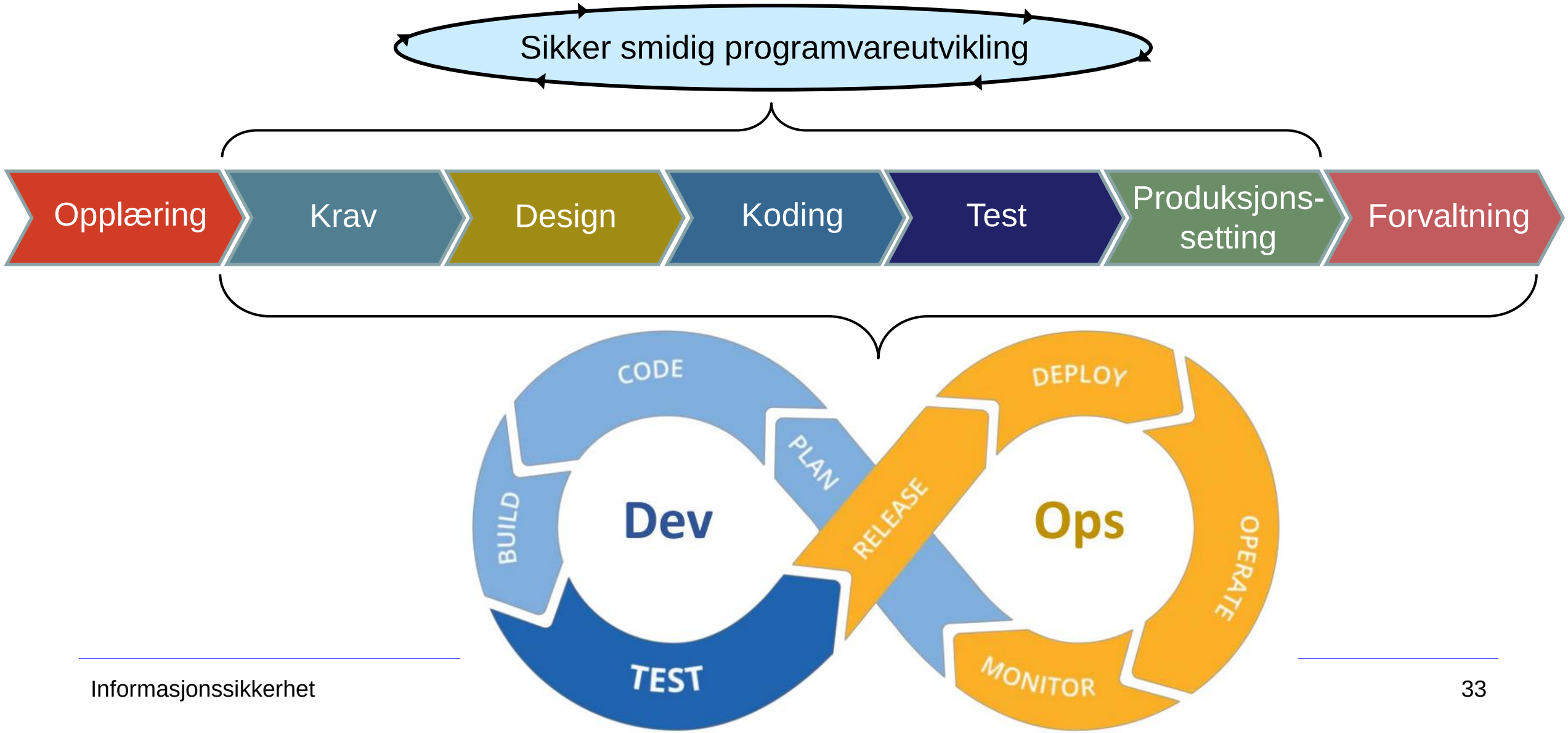
- **Plan for drift, vedlikehold og hendelseshåndtering** må definere prosedyrer for drift (inkl. patching), avviksrapportering og hendelseshåndtering (mer om dette neste uke)
- **Formell godkjenning av produksjonssetting** vil kreve at det verifiseres og dokumenteres at alle krav til sikkerhet og personvern er oppfylt og identifiserte sårbarheter er tilstrekkelig fjernet. Formelt ansvar/mandat må defineres og relevant data og dokumentasjon arkiveres.

7. Forvaltning

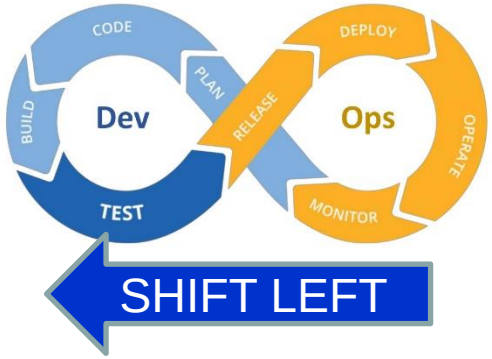
- **Drift og vedlikehold** innebærer at prosedyrer og rutiner for drift og vedlikehold av programvare skal følges, også over tid. Revisjoner bør gjennomføres regelmessig og et ledelsessystem for informasjonssikkerhet bør være på plass. Man må klart definere hva som skal logges og hvordan loggene håndteres.
- **Avviks- og hendelseshåndtering** Avvik og hendelser skal rapporteres som beskrevet i planene. (Mer om hendelseshåndtering neste uke).



DevOps: Sikker smidig programvareutvikling i skyen

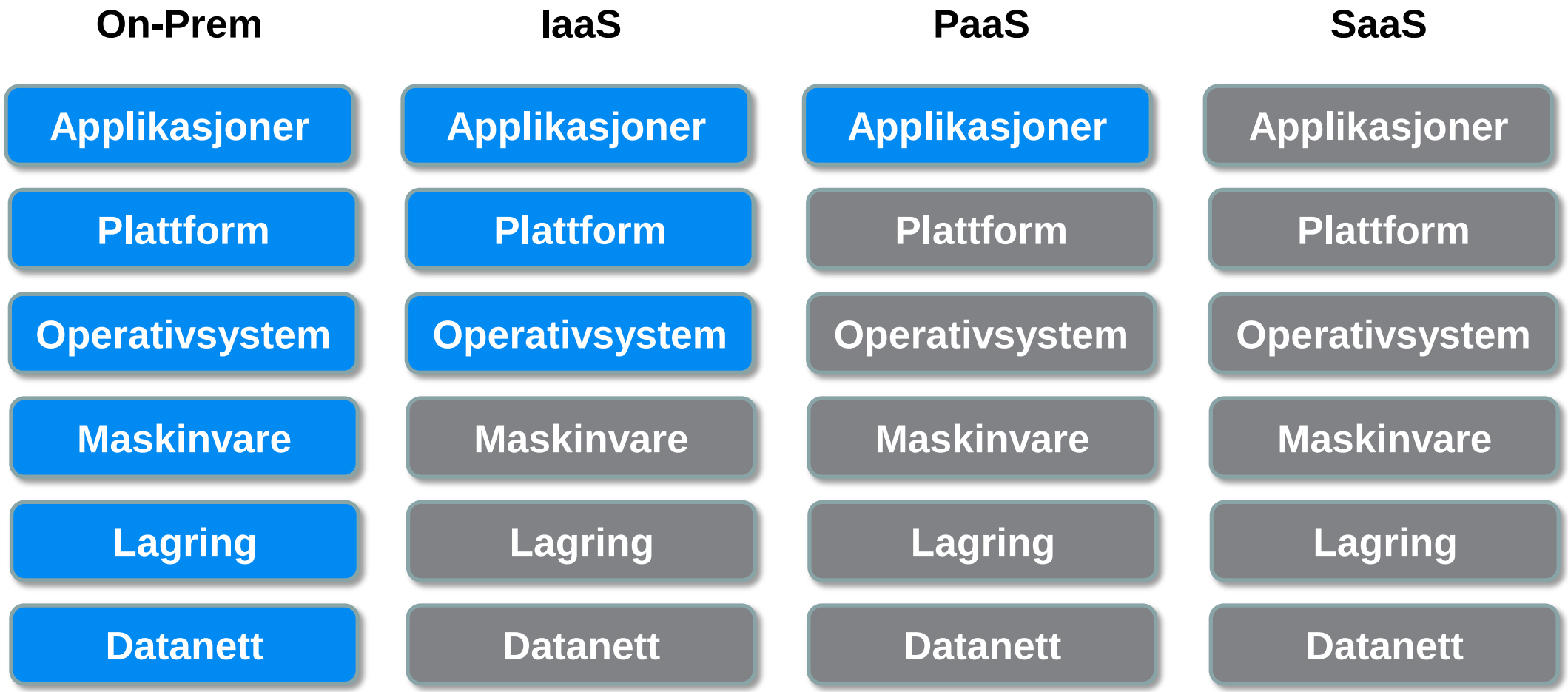


DevSecOps of SHIFT LEFT



- *DevOps*, også kalt *DevSecOps*, er som navnet indikerer å kombinere programvareutvikling (Development) og drift (Operations).
- DevOps tar sikte på å forkorte syklusen rundt utvikling og drift av programvare og gi kontinuerlig levering med høy kvalitet og sikkerhet (Security).
- DevOps er videreutvikling av sikker smidig programvareutvikling, ved å inkluderer drift/forvaltning, mens sikker smidig bare går frem til produksjonssetting.
- Alle verktøy som trengs i hele DevOps-syklusen ligger i skyen, fra utviklingsverktøy og testverktøy til verktøy for produksjonssetting, drift og monitorering. Skymodeller for DevOps må være enten IaaS eller PaaS.
- Begrepet «SHIFT LEFT» betyr at det fokuseres mer på sikkerhet på venstre side av 8-tallet, slik at det blir færre sikkerhetssårbarheter og -hendelser under drift.

Delt ansvar ved ulike skymodeller



Delt ansvar for sikkerhet ved ulike skymodeller

Sikkerhet i skyen

- Skyleverandører har ofte betydelig kompetanse og ressurser for å opprettholde høy grad av sikkerhet i sine infrastrukturer, f.eks. ved rask sikkerhetsoppdatering, effektiv sikkerhetsmonitorering og deteksjon, og hendelsesrespons.
- Aspekter som kan true sikkerhet ved bruk av skyløsninger er f.eks.:
  - Potensiell korrupsjon eller utro tjenere hos skyleverandøren. Sannsynlighet for dette er vanligvis svært lav, men kan øke i land der nivået av korrupsjon er relativt betydelig og rettsikkerheten er relativt lav.
  - Lovlig tilgang til virksomhetens data fra myndigheter i land der skyleverandøren er lokalisert. Dette kan skje som del av etterretning eller etterforskning av kriminalitet, uten at virksomheten nødvendigvis blir informert.
  - Identitets- og tilgangshåndtering ved bruk av skytjenester kan utgjøre en sårbarhet hvis den er dårlig implementert. Skytjenester er teknisk sett tilgjengelig fra hele Internett, slik at det trengs robuste løsninger for beskyttelse av tilgang.

Slutt på presentasjonen