Oslo

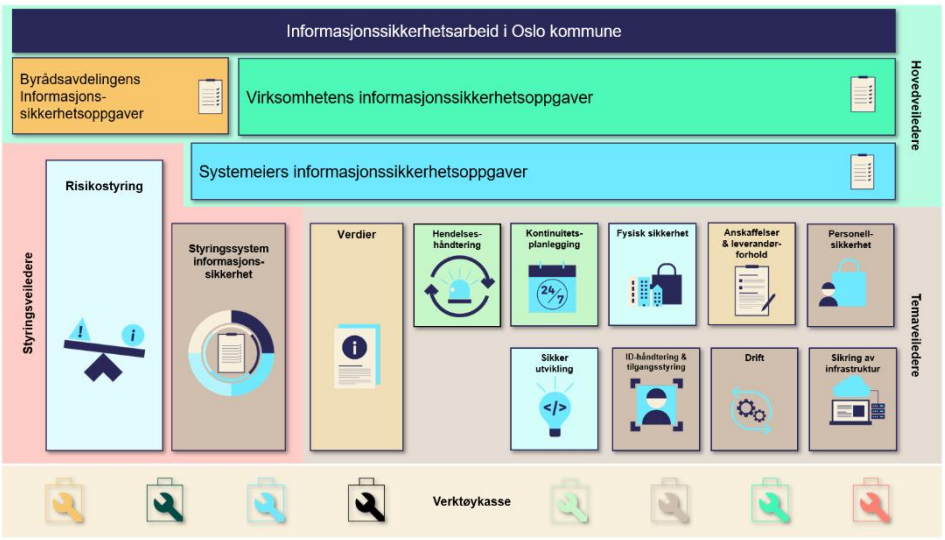
Byrådsavdeling for finans
Laget av: Oslo Origo

# Temaveileder for sikker programvareutvikling

Rammeverk for informasjonssikkerhet

# Innhold

# 1 Om temaveilederen




Denne temaveilederen handler om sikkerhet i programvareutvikling. Temaveilederen tar utgangspunkt i faser, roller og oppgaver inspirert av DevOps (se kapittel 2.3) som sørger for innebygd sikkerhet i alle faser av programvareutviklingen.

Temaveilederen er strukturert etter ulike faser i utviklings- og driftsprosessen, med tilhørende roller og oppgaver som sikrer innebygd informasjonssikkerhet i hele livsløpet av digitale tjenester. Til tross for denne inndelingen etter fasene i DevOps, kan temaveilederen også brukes med andre metoder for programvareutvikling. Dette gjør det enklere å levere programvare raskere, med færre feil, og mulighet for kontinuerlige forbedringer og tilpasninger.

<

Oppgavene er utledet fra beste praksis for sikkerhet i programutvikling. Oppgavene søker også å operasjonalisere krav til informasjonssikkerhet i lovverk, kommunens interne styringsdokumenter, anbefalte sikkerhetstiltak i ISO 27002 og NSMs grunnprinsipper.

Hvert tema i denne temaveilederen begynner med en kort beskrivelse om formålet med fasen og hvilken rolle informasjonssikkerhet har i fasen. Deretter følger én eller flere tilhørende oppgaver fordelt på de ulike rollene beskrevet under.

Oppgavene i denne temaveilederen er fordelt på flere faser og roller i programvareutvikling. For å lettere identifisere dem er de fargekodet og merket med ikoner slik:

| | |
|---|---|
| Systemeier | ***Oppgave**. Systemeiers[1] oppgaver har brun bakgrunn.* |
| Teamlead | ***Oppgave**. Teamlead sine oppgaver har rosa bakgrunn.* |
| Techlead | ***Oppgave**. Techlead sine oppgaver har gul bakgrunn.* |
| Utvikler | ***Oppgave**. Utviklers oppgaver har lilla bakgrunn.* |
| | *Eventuelle hjelpemidler og verktøy kommunen har utviklet for å bistå i oppgaven. Hjelpemidler i <klammer> er planlagt, men eksisterer ikke ennå.* |
| | *Eventuell tilleggsinformasjon eller anbefalinger.* |

*Tabell 1: Fargekoding og ikoner for rollene i programvareutvikling*

## 1.1 Formål med temaveilederen


*Figur 1: Temaveilederens funksjon i operativ programvareutvikling*

Formålet med denne temaveilederen er å gi veiledning til sikker programvareutvikling. Veilederen beskriver de ulike fasene i utviklingsprosessen med oppgaver som sørger for et forsvarlig informasjonssikkerhetsnivå. Oppgavene er fordelt på rollene: systemeier, teamleder, techlead og utvikler. Som Figur 1 over viser, fungerer veilederen også som en tolk som viser sammenhengen mellom regelverket og det operative nivået i programvareutvikling.

## 1.2 Definisjoner av Hovedansvarlig, Utførende, Konsulteres og Informeres (HUKI)

For å sikre at alle oppgaver har de rollene som er nødvendig og en klar ansvarsfordeling, er det satt opp en HUKI-matrise for oppgavene i hver fase. I HUKI-

[1] *I denne veilederen har vi valgt å avgrense systemeierbegrepet til kun å omfatte budsjett-, ressurs- og risikoeierskap med hovedfokus på aktiviteter relatert til sikker programvareutvikling*

matrisene brukes bokstavene **H**(ovedansvarlig), **U**(tførende), **K**(onsulteres) og **I**(nformeres).

I denne veilederen er det valgt følgende definisjon av disse rollene:

**H**: Dette er den som har ansvar for å ta initiativ til å gjennomføre oppgaven og som påser at den blir utført med ønsket resultat. Det kan kun være én person med denne rollen for hver oppgave. Implisitt i **H** ligger også at vedkommende kan være **U**tførende.

**U**: Dette er den som utfører selve arbeidet for å løse oppgaven. Det kan være flere som innehar denne rollen.

**K**: Dette er den som skal konsulteres ved gjennomføringen av oppgaven og som kan stoppe eller endre innholdet i foreslått løsning av oppgaven. Ikke alle oppgaver trenger noen som konsulteres.

**I**: Dette er den som skal informeres om status for gjennomføringen og resultatet av oppgaven. Det kan være flere som innehar denne rollen.

# 2 Sikker programvareutvikling i Oslo kommune

Sikkerhet må være en integrert del av utvikling og drift for å beskytte brukere og data. Veilederen bygger på prinsippene for innebygd personvern (GDPR art. 25) og informasjonssikkerhet (GDPR art. 32), som begge vektlegger tidlig planlegging og implementering av sikkerhetstiltak.

Veilederen er strukturert etter faser i programvareutvikling og drift, men krever ikke en bestemt metodikk eller teknologi. Faseinndelingen bygger på DevOps og viser til hvilken kontekst man befinner seg i – som planlegging, programmering eller testing (se begrepsoversikt Vedlegg A).

Utvikling skjer ofte iterativt, og det er normalt å gå tilbake til tidligere faser. For eksempel kan bruk av sensitive data i testfasen kreve ny risikovurdering i planleggingsfasen for å sikre konfidensialitet, integritet og tilgjengelighet (GDPR art. 5.1.f).

## 2.1 Hva er sikker programvareutvikling?

Sikker programvareutvikling betyr å bygge inn sikkerhet fra start. Løsninger må altså oppfylle både funksjonelle krav og samtidig beskytte data. Sikkerhet må være en del av hele prosessen, fra planlegging til drift. Dette reduserer risikoen for sårbarheter og angrep. Samtidig sikrer det robuste og pålitelige digitale tjenester som ivaretar brukernes tillit og oppfyller Oslo kommunes sikkerhetskrav.

## 2.2 Hvordan få til sikker programvareutvikling?

For å oppnå sikker programvareutvikling i Oslo kommune, må sikkerhet være en integrert del av hele utviklingsprosessen. Alle i teamet, fra utviklere til ledere, bør ha grunnleggende sikkerhetskunnskap og følge beste praksis. Regelmessig

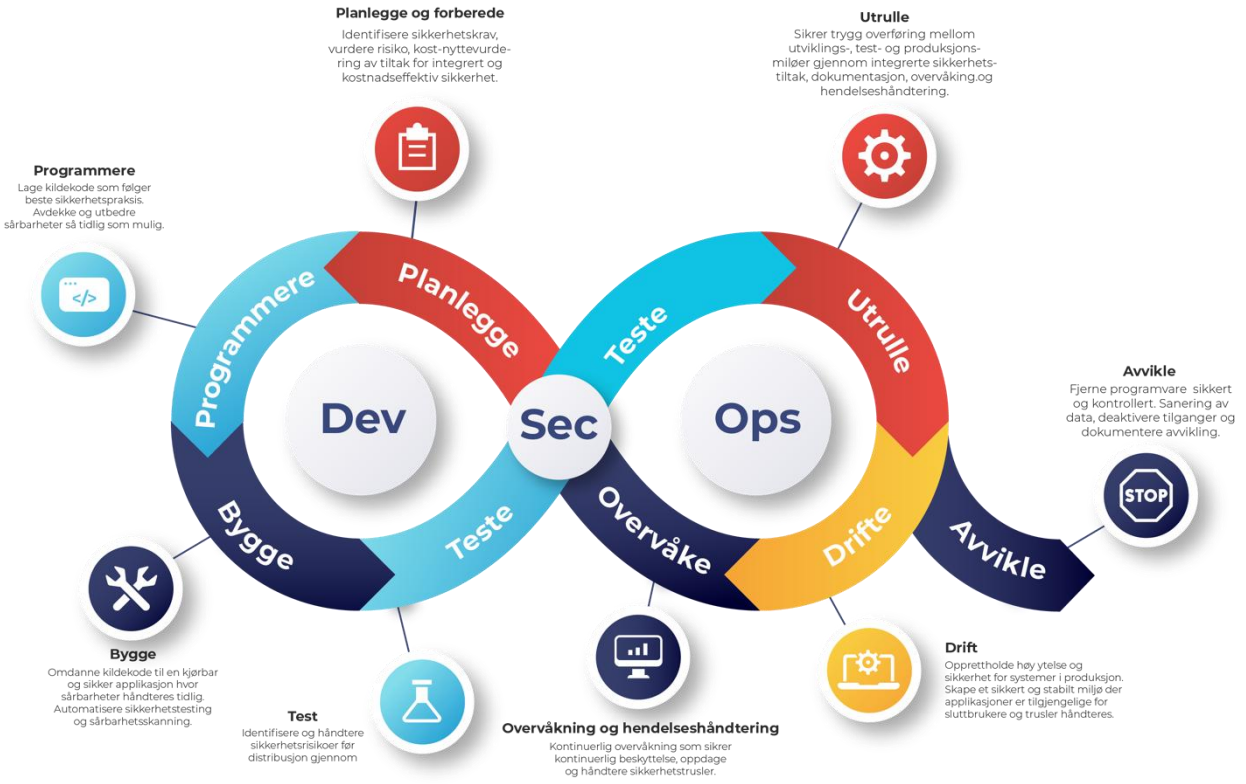
sikkerhetsopplæring og faglig kompetanseutvikling er avgjørende for å holde seg oppdatert.

Sikkerhet kan integreres i utviklingen uten at det går på bekostning av effektiviteten ved for eksempel ved å bruke relevante verktøy fra skyplattformer. Gode rutiner for risikostyring er viktig, inkludert riktig prioritering av ressurser til sikkerhetsverktøy og tydelige retningslinjer for å oppdage og håndtere trusler.

Samarbeid mellom beslutningstagere, utviklingsteam og IT-sikkerhetseksperter er avgjørende for å sikre at en oppfyller sikkerhetskravene. Videre gjør systemovervåkning og varsling i sanntid det mulig å identifisere og håndtere trusler. Dette bidrar til kontinuerlig forbedring av sikkerheten.

## 2.3 Hva er DevOps?

DevOps er en tilnærming til programvareutvikling og IT-drift som fremmer samarbeid ved å slå sammen utvikling (Dev) og drift (Ops) i samme team. Målet er å forbedre hastighet, effektivitet og kvalitet på leveranser av programvare ved å bryte ned siloene mellom programvareutvikling og -drift.




*Figur 2: DevOps livssyklus [1]*

I en DevOps-kultur jobber ressurser med utviklings- og driftskompetanse tett sammen gjennom hele programvarelivssyklusen, fra planlegging og utvikling til testing, distribusjon, drift og vedlikehold.

«DevSecOps» er et begrep som ofte brukes om sikkerhet (Sec) som integrert del av programvareutvikling basert på DevOps-tilnærmingen. I denne temaveilederen har vi valgt å bare bruke «DevOps»-begrepet og at det også omfatter sikkerhet i alle faser.

## 2.4 Roller i programvareutvikling

Nedenfor beskrives ulike roller i utviklingsprosjekter, med ansvar fordelt for økonomiske rammer, sikkerhet og risikohåndtering. Denne rollefordelingen legger et solid grunnlag for å ivareta sikkerhet i alle deler av utviklingsprosessen.

| | | |
|---|---|---|
|  | ***Systemeier*** | Ansvarlig for beslutning av risiko, samt allokering av budsjett til sikkerhetstiltak og risikoreduserende tiltak. Sørger for at sikkerhetsinvesteringer er innenfor budsjettet. I denne veilederen har vi valgt å avgrense systemeierbegrepet til kun å omfatte budsjett-, ressurs- og risikoeierskap. Det er også et bevisst valg å tydeliggjøre at denne rollen er ansvarlig for risiko, har budsjettansvar, gjør prioriteringer, og sørger for at ressurser brukes på en fornuftig måte. |
|  | ***Teamlead*** | Koordinerer teamets arbeid, sørger for at prosesser følger sikkerhetsprinsipper og at risikohåndtering er integrert. Fjerner hindringer og sørger for kontinuerlig forbedring og kompetanseutvikling. |
|  | ***Techlead*** | Ansvarlig for teknisk arkitektur og kodekvalitet med fokus på sikkerhet. Utfører risikoanalyser, anbefaler tiltak, og sørger for at sikkerhet er integrert i utviklingsprosessen og bygge- og utrullingsprosessen. |
|  | ***Utvikler*** | Implementerer sikker kode basert på beste praksis og utfører sikkerhetstesting. Rapporterer risikoer, oppdaterer tredjepartsavhengigheter, og overvåker applikasjonens sikkerhet i produksjon. |

*Tabell 2: Rollebeskrivelser i sikker programvareutvikling*

# 3 Oppgaver i sikker programvareutvikling

Dette kapitlet beskriver de ulike fasene i utviklings- og driftsprosessen, og hvilke oppgaver som bidrar til at informasjonssikkerhet er en integrert del av programvareutviklingen. Hver enkelt fase med de tilhørende oppgavene kan leses som frittstående kapitler.

## 3.1 Planlegge

I planleggingsfasen identifiseres sikkerhetskrav, risiko vurderes, og kost-nyttevurderinger av sikkerhetstiltak gjennomføres. Dette sikrer at sikkerhet integreres fra start og at tiltakene er både effektive og kostnadsmessig forsvarlige. Systemeier vurderer risiko opp mot kostnader for å finne en balansert løsning. Resultatet er en plan som legger til rette for trygg utvikling og effektiv ressursbruk. Oppgavene nedenfor støtter dette målet.

### 3.1.1 Finansiering og ressurser

For å sikre at informasjonssikkerhet er en integrert del av utviklingsprosessen, må nødvendige ressurser være tilgjengelig fra starten av prosjektet. Finansiering skal dekke anskaffelse og drift av sikkerhetsverktøy, opplæring av teammedlemmer og gjennomføring av risikoreduserende tiltak gjennom hele programvarens livsløp.

Ressursplanleggingen må sikre at teamet har kapasitet og kompetanse til å ivareta sikkerhetsoppgavene i alle faser. Systemeier har ansvar for å avsette midler og prioritere tiltak basert på virksomhetens risikovurderinger, slik at sikkerhet kan gjennomføres uten at økonomiske eller organisatoriske hindringer svekker arbeidet.

### 3.1.2 Oppgaver i planleggingsfasen

| **Oppgave** | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Fastsette akseptabelt risikonivå**<br>Akseptabelt risikonivå er fastsatt for løsningen, slik at beslutninger om sikkerhetstiltak kan forankres og prioriteres ut fra virksomhetens risikotoleranse. | K | H | I | I |
| **Gjennomføre innledende risikoanalyse**<br>Gjennomføre en innledende risikoanalyse for å identifisere risikoer og nødvendige tiltak tidlig i prosjektet. Bruk gjerne trusselmodellering for dette. | K | H | U | U |
| **Akseptere restrisiko**<br>Akseptere restrisiko etter gjennomført risikoanalyse og evaluering av tiltak, slik at videre håndtering baseres på bevisste og informerte beslutninger. | K | H | | |

| | | | | |
|---|---|---|---|---|
| **Planlegge og iverksette tiltak**<br>Planlegge og iverksette tiltak identifisert gjennom risikoanalyse, slik at risikoen reduseres til akseptabelt nivå. | | I | H | U |
| **Etablere rutiner for utviklingsfasene**<br>Etablere rutiner for sikker utvikling i alle faser, slik at arbeidet gjennomføres strukturert og risiko reduseres. | | H | U | U |
| **Etablere plan for kompetanseheving**<br>Etablere og iverksette opplæringsplan innen sikker programvareutvikling, slik at teamet har nødvendig kompetanse for å ivareta sikkerhet gjennom samtlige faser i utviklingsprosessen. | | H | | |
| **Sikre finansiering for utviklingsfasene**<br>Allokere økonomiske midler til sikkerhetstiltak og aktiviteter gjennom alle utviklingsfaser, slik at nødvendige tiltak kan gjennomføres uten økonomiske hindringer. | H | I | | |
| **Sikre tilstrekkelige ressurser**<br>Sikre tilstrekkelig bemanning og ressurser i alle faser av programvareutviklingen, slik at sikkerhetsarbeidet utføres effektivt og grundig. | | H | I | I |
|  | Veileder for hendelseshåndtering<br>Veileder for kontinuitetsplanlegging<br>OWASP Threat Modelling | | | |
|  | Bruk OWASP Threat Modelling for å strukturere risikoanalysen og tydeliggjøre trusler tidlig i utviklingsløpet. Suppler med kommunens veiledere for hendelseshåndtering og kontinuitetsplanlegging for å sikre helhetlig risikostyring. | | | |

## 3.2 Programmere

I programmeringsfasen utvikles kildekode som følger beste sikkerhetspraksis og som avdekker og utbedrer sårbarheter så tidlig som mulig.

Alle i utviklingsteamet har et ansvar for i fellesskap å sikre at det som programmeres følger avtalte standarder for sikkerhet og holder seg oppdatert på trusler og sårbarheter. Dette krever at det settes av nok tid til dette arbeidet og at det ses på som en integrert og helt nødvendig del av utviklingsarbeidet. De andre rollene i teamet skal støtte og legge til rette for at programmererne kan gjennomføre sikker utvikling.

### 3.2.1 Oppgaver i programmeringsfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Kompetanse i programmeringsfasen**<br>Gjennomføre regelmessig opplæring i sikker programvareutvikling slik at teamet kontinuerlig oppdaterer og vedlikeholder nødvendig kompetanse. | | H | KU | U |
| **Vedlikeholde tredjepartsbiblioteker og rammeverk**<br>Velge kun pålitelige tredjepartsbiblioteker og rammeverk, og holde dem oppdatert for å unngå kjente sårbarheter. | | | H | U |
| **Begrensede rettigheter**<br>Begrense utvikleres rettigheter i lokale utviklingsmiljøer til det nødvendig for å redusere risikoen for uønskede endringer og kompromittering. | | H | K | I |
| **Benytte kontinuerlig kildekodekontroll**<br>Benytte verktøy for kontinuerlig kontroll av kildekode slik at sårbarheter, feil og sikkerhetsavvik oppdages tidlig. | | | H | U |
| **Manuell kodegjennomgang**<br>Gjennomføre regelmessig manuell kildekodegjennomgang for å avdekke sikkerhetsrelaterte feil og svakheter som automatiske verktøy kan overse. | | | H | U |
| **Kjøre automatiserte enhetstester**<br>Kjøre automatiserte enhetstester og integrasjonstester kontinuerlig for å sikre at sikkerhetskrav ivaretas, og at avvik oppdages og korrigeres tidlig. | | | H | U |
|  | OWASP Top Ten<br>OWASP Cheat Sheets<br>IDE-plugins for sikker koding (f.eks. SonarLint) | | | |
|  | Bruk etablerte kodestandarder og verktøy som gir tidlige varsler om sårbarheter. Sørg for kontinuerlig opplæring slik at teamet er oppdatert på nye trusler og rammeverk. | | | |

## 3.3 Bygge programvare

I byggefasen omdannes kildekode til kjørbar og sikker programvare, der sårbarheter håndteres tidlig. Byggefasen automatiserer sikkerhetstesting og sårbarhetsskanning for å sikre at programvaren oppfyller alle sikkerhetskrav før videre testing og distribusjon.

Gjennom kontinuerlig systemovervåkning av sikkerhetstester sørger byggefasen for en stabil og sikker applikasjon som er klar til neste steg.

### 3.3.1 Oppgaver i byggefasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Kompetanseheving i byggefasen**<br>Etablere kompetanse i å konfigurere automatiserte sikkerhetstester, sårbarhetsskanning og statisk kodeanalyse for å sikre korrekt bruk og tidlig håndtering av funn. | | H | KU | U |
| **Konfigurere sikkerhetsverktøy**<br>Sikkerhetsverktøy konfigureres for automatiserte sikkerhetstester, sårbarhetsskanning og statisk kodeanalyse, slik at sårbarheter avdekkes tidlig. | | | H | U |
| **Kjøre automatiserte sikkerhetstester**<br>Sikkerhetstester kjøres automatisk i bygg for å avdekke sårbarheter tidlig og sikre kontinuerlig verifisering av sikkerhetskrav.<br>Sørge for å loggføre og tilgjengeliggjøre resultater fra automatiserte sikkerhetstester for videre analyse og oppfølging. | | | H | U |
| **Oppfølging av avvik**<br>Følge opp avvik funnet i analyser ved å iverksette nødvendige tiltak for å redusere risiko og forbedre sikkerheten. | | H | KU | U |
| | SAST (Static Application Security Testing)<br>DAST (Dynamic Application Security Testing)<br>SCA (Software Composition Analysis, f.eks. OWASP Dependency-Check, Snyk) | | | |
| | Integrer sikkerhetsskanninger i byggeprosessen for å oppdage feil tidlig. Bruk f.eks. SCA for å avdekke sårbarheter i tredjepartsbiblioteker. | | | |

## 3.4 Teste programvare

I testfasen identifiseres og håndteres sikkerhetsrisikoer før programvaren rulles ut. Her brukes avanserte tester som sårbarhetsskanninger, kodeanalyse og funksjonelle tester for å validere sikkerhet, ytelse og funksjonalitet. Testing verifiserer også at sikkerhetstiltak som kryptering og tilgangskontroller fungerer som forventet. Dette

sikrer at programvaren er stabil og trygg, slik at eventuelle problemer kan utbedres før publisering.

### 3.4.1 Oppgaver i testfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Kompetanseheving i sikkerhetstesting**<br>Teamet foretar løpende opplæring i testverktøy og testmetoder for å sikre korrekt gjennomføring og tolkning av sikkerhetstester | | H | KU | U |
| **Sikre trygg bruk av testdata**<br>Bruk av testdata skal være i tråd med regelverk og sikkerhetskrav. Det skal benyttes anonymiserte eller syntetiske data, og sensitiv informasjon skal ikke brukes i test uten særskilt risikovurdering. | | H | U | U |
| **Kontinuerlig sikkerhetstesting**<br>Sikkerhetstesting gjøres som del av testprosessen, både for automatiserte og manuelle tester. Resultatene fra sikkerhetstestene dokumenteres og gjennomgås. | | K | H | U |
| **Vurdere ytterligere sikkerhetstester**<br>Ytterlig sikkerhetstester vurderes, inkludert penetrasjonstesting. | I | H | | |
| **Oppfølging av testresultatene**<br>Risikovurdering oppdateres basert på testresultatene og nye risikoreduserende tiltak vurderes. | I | H | KU | U |
| Penetrasjonstesting (f.eks. Burp Suite, OWASP ZAP)<br>OWASP Web Security Testing Guide<br>Testdata-verktøy for syntetiske data (f.eks. DataFaker, Faker)<br>Rammeverk for automatisert test (JUnit, pytest, Cypress) | | | | |
| Suppler funksjonelle tester med penetrasjonstesting for å fange opp komplekse sårbarheter. Bruk alltid fiktive testdata i samsvar med personvernkrav. | | | | |

## 3.5 Rulle ut programvare

I utrullingsfasen fremheves viktigheten av integrerte sikkerhetstiltak som signering for integritetsbekreftelse, autentisering for identitetskontroll og tilgangsbegrensning. Dette sikrer trygg overføring mellom utviklings-, test- og produksjonsmiljøer. Ved å kombinere grundig dokumentasjon, kontinuerlig overvåking og beredskap for hendelseshåndtering reduseres risikoen for feil og sikkerhetsproblemer.

### 3.5.1 Oppgaver i utrullingsfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Følge etablerte utrullingsrutiner**<br>Følge etablerte rutiner for utrulling slik at prosessen utføres kontrollert, forutsigbart og sikkert. | | | H | U |
| **Automatisere og overvåke utrullinger**<br>Sette opp og vedlikeholde automatiserte rutiner for utrulling, samt overvåke systemet etter distribusjon for rask oppdagelse og håndtering av avvik. | | I | H | U |
| **Verifisere funksjonalitet og sikkerhet**<br>Verifisere utrullingen med hensyn til funksjonalitet og sikkerhet i produksjonsmiljøet, slik at løsningen møter fastsatte krav og unngår innføring av nye risikoer. | | | H | U |
| **Tilrettelegge for og teste tilbakerulling (rollback)**<br>Utform og test prosedyre for trygg tilbakeføring til forrige versjon dersom det oppstår feil eller sikkerhetsbrudd ved utrulling. | | | H | U |
| **Oppdatere og vedlike utrullingsdokumentasjon**<br>Oppdatere og forbedre dokumentasjon av utrullingsprosessen basert på erfaringer og tilbakemeldinger, for å sikre kontinuerlig læring og forbedring. | | H | U | I |

CI/CD pipelines (GitHub Actions, GitLab CI, Jenkins)
Signeringsverktøy for kode og artefakter
Integritetskontroller (SHA256/512-sjekksummer)

Sørg for at publiseringsrutiner inkluderer signering og integritetskontroller. Legg inn logging og overvåkning slik at feil eller angrep raskt kan oppdages.

## 3.6 Drift

I driftsfasen opprettholdes høy ytelse og sikkerhet mens systemene er i produksjon. Dette skaper et sikkert og stabilt miljø der applikasjonene alltid er tilgjengelige for sluttbrukere, og der trusler håndteres forebyggende. Ved å implementere automatiserte prosesser for overvåking, logging og oppdateringer identifiseres og håndteres potensielle trusler raskt og effektivt, uten forsinkelser som kan påvirke systemets stabilitet.

### 3.6.1 Oppgaver i driftsfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Sikre tjenestekvalitet**<br>Sikre at forventet tjenestetilgjengelighet og -ytelse er i tråd med avtaler og krav (SLA), slik at eventuelle avvik raskt kan oppdages og håndteres. | K | H | U | U |
| **Revidere og justere tilganger**<br>Gjennomgå og justere bruker- og systemtilganger jevnlig, slik at tilgangsstyring alltid samsvarer med gjeldende retningslinjer. | | H | KU | U |
| **Håndtere endringer i produksjonsmiljø**<br>Gjennomføre, dokumentere og loggføre endringer systematisk, slik at endringshistorikk og sikker drift ivaretas. | | I | H | U |
| **Automatisert sikkerhetskopiering og gjenoppretting**<br>Automatisere sikkerhetskopiering, samt jevnlig teste og dokumentere gjenoppretting for å sikre tilgjengelighet og integritet av data. | | | H | U |
| **Håndtere sikkerhetsoppdateringer**<br>Identifisere, gjennomføre og kontrollere sikkerhetsoppdateringer i henhold til etablerte rutiner for å beskytte systemet mot sårbarheter. | | | H | U |
| **Sårbarhetsanalyse og validering av tiltak**<br>Driftssystemet er jevnlig analysert for nye sårbarheter. Sikkerhetstiltakene er testet og validert for å sikre at de fungerer som forutsatt. | | K | H | U |
| Overvåkning og logging (f.eks. Datadog, Prometheus, ELK Stack)<br>Sårbarhetsskannere (f.eks. Nessus, OpenVAS)<br>SIEM-løsning (f.eks. Splunk, Microsoft Sentinel) | | | | |
| Kombiner kontinuerlig overvåkning med jevnlige sårbarhetsskanninger.<br>Dokumenter og evaluer hendelser systematisk for læring og forbedring. | | | | |

## 3.7 Systemovervåkning og hendelseshåndtering

I systemovervåknings- og hendelseshåndteringsfasen sikres kontinuerlig beskyttelse, og uforutsette hendelser håndteres. Dette gjør det mulig å oppdage og håndtere sikkerhetstrusler og så raskt som mulig komme tilbake til en normal driftsituasjon. Kontinuerlig systemovervåkning vil hjelpe teamet med å finne avvik og sårbarheter

tidlig. Fokus i denne fasen er forebyggende sikkerhet. Dette reduserer skade og gjør systemet mer robust.

### 3.7.1 Oppgaver systemovervåknings- og hendelseshåndteringsfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Opplæring og øvelser i sikkerhetshendelser**<br>Gjennomføre regelmessig opplæring og øvelser i håndtering av sikkerhetshendelser, slik at teamet er forberedt og kompetent ved reelle hendelser. | | H | KU | U |
| **Definere og vedlikeholde planverk**<br>Definere og oppdatere prosesser og rutiner for håndtering av sikkerhetshendelser, slik at de alltid er i samsvar med eksisterende rutiner for beredskap og hendelseshåndtering. | I | H | U | I |
| **Verktøy for systemovervåking**<br>Aktivere nødvendige verktøy for systemovervåkning og logging. Verifisere at varslinger fungerer som de skal og når nødvendige parter. | | I | H | U |
| **Etablere rutiner for eskalering**<br>Etablere rutiner for eskalering, og beslutte ekstraordinære sikkerhetstiltak som skal iverksettes ved kritiske hendelser. | K | H | I | |
| **Informere eksterne parter**<br>Sørge for at relevante eksterne aktører varsles og involveres ved alvorlige sikkerhetshendelser, i tråd med etablerte varslingsrutiner og avtaler. | H | U | I | |
| **Verifisere loggfunksjonalitet**<br>Sikre at loggføring er aktivert, fungerer som forventet og gir tilstrekkelig innsikt for å kunne oppdage, analysere og håndtere sikkerhetshendelser. | | | H | U |
| **Evaluere og lære av hendelser**<br>Hendelser både på overordnet og operativt nivå skal evalueres, og rapporteres for videre læring. Rutiner og planer skal oppdateres fortløpende med relevante læringspunkter. | I | H | U | I |
| Veileder for hendelseshåndtering<br>Veileder for kontinuitetsplanlegging | | | | |
| Etabler rutiner for hendelseshåndtering og kontinuitetsplanlegging for rask respons. Evaluer hendelser og oppdater prosesser løpende for å styrke beredskapen. | | | | |

## 3.8 Avvikle programvare

I avviklingsfasen sikres det at programvaren fjernes fra drift på en sikker og kontrollert måte, der sensitiv informasjon håndteres forsvarlig og ressurser frigis trygt. I denne fasen er det viktig å gjennomføre sanering av data, deaktivere tilgang og dokumentere avvikling, for å redusere risiko for uautorisert tilgang eller dataeksponering etter at systemet er tatt ut av bruk.

### 3.8.1 Oppgaver i avviklingsfasen

| Oppgave | Systemeier | Teamlead | Techlead | Utvikler |
|---|---|---|---|---|
| **Foreta beslutning om avvikling**<br>Organisatoriske og tekniske avhengigheter og konsekvenser dokumenteres og vurderes. Avvikling risikovurderes, besluttes og planlegges. | K | H | I | I |
| **Arkivering og dataoverføring**<br>Data som skal bevares overføres eller arkiveres iht. regulatoriske krav og beslutninger. | K | H | U | U |
| **Sletting og teknisk sanering**<br>Teknisk sanering av database, programvare, servere, nettverksåpninger osv. utføres og sensitive data slettes. | I | K | H | U |
| **Håndtering og fjerning av tilganger**<br>Alle fysiske og digitale tilganger som var nødvendig for drift av programvaren, inkludert til eksterne API-er, servere, databaser, nøkler og sertifikater fjernes. | I | K | H | U |
| Verktøy for sanering og sletting<br>Dokumentasjonsverktøy for sporbarhet (Confluence, SharePoint, etc.) | | | | |
| Bruk etablerte rutiner og verktøy for sikker sletting av data og avvikling av tilganger. Dokumenter prosessen for å ivareta krav til sporbarhet og etterlevelse. | | | | |

# Vedlegg A Begrepsoversikt

Formålet med denne begrepsoversikten er å skape en felles forståelse av begreper som anvendes i temaveilederen.

| Begrep | Definisjon |
|---|---|

| | |
|---|---|
| DevOps | DevOps er en tilnærming til programvareutvikling og IT-drift som fremmer samarbeid, automatisering og integrasjon mellom utviklingsteam (Dev) og driftsteam (Ops) for å muliggjøre raskere, mer pålitelige og kontinuerlige leveranser av programvare. |
| DevSecOps | DevSecOps er et begrep en utvidelse av DevOps-praksisen som integrerer sikkerhet (Sec) i hele utviklings- og driftsprosessen, slik at sikkerhetstiltak og kontroller blir en naturlig del av programvareutviklingen, fra kodebygging til distribusjon, uten å ofre hastighet eller smidighet. |
| Drift | Drift er den kontinuerlige administrasjonen, overvåkingen og vedlikeholdet av IT-systemer, programvare og infrastruktur for å sikre at de fungerer som forventet. |
| Fase | En fase i DevOps-syklusen. En fase kan også ses på som konteksten man er i relatert til programvareutvikling. Eksempler på dette er to av kontekstene en utvikler skifter raskt mellom: programmering og test. Utvikler vil typisk veksle mellom å programmere et stykke programvare for så å lage tester for den nye programvarebiten. |
| Informasjonssikkerhet | Informasjonssikkerhet er praksisen med å beskytte informasjon mot uautorisert tilgang, endring, tap eller ødeleggelse, ved å sikre konfidensialitet, integritet og tilgjengelighet av data gjennom passende sikkerhetstiltak og kontroller. |
| Penetrasjonstesting | Penetrasjonstesting er en kontrollert sikkerhetstest der man simulerer angrep for å identifisere og utnytte sårbarheter i et system, nettverk eller applikasjon. |
| Personopplysningssikkerhet | Personopplysningssikkerhet handler om å beskytte personopplysninger mot at uvedkommende får tilgang, tap, misbruk eller annen kompromittering. Målet er å sikre at disse opplysningene behandles trygt og ansvarlig i henhold til loven (som GDPR i Europa). |
| Programvare | Programvare er en samling instruksjoner som får datamaskiner til å utføre oppgaver. Den kan være systemprogramvare som operativsystemer, eller applikasjoner som brukes av brukere. |
| Programvareutvikling | Programvareutvikling er prosessen med å lage programvare, som inkluderer planlegging, koding, testing og implementering for å møte brukerens krav. |
| Restrisiko | Restrisiko er risikoen som gjenstår etter at relevante sikkerhetstiltak og kontroller er implementert. Denne risikoen er kjent og bevisst akseptert av virksomheten, ofte fordi kostnaden eller innsatsen ved å fjerne risikoen helt overstiger den potensielle gevinsten. |
| Riskoanalyse | Risikoanalyse er en systematisk prosess for å identifisere, vurdere og prioritere risikoer. Formålet er å forstå mulige trusler, sårbarheter og konsekvenser for virksomhetens systemer og data, samt å gi grunnlag for beslutninger om tiltak som skal redusere eller akseptere risikoen. |
| Sikker koding | Sikker koding er praksisen med å utvikle programvare som er motstandsdyktig mot sikkerhetssårbarheter ved å anvende beste sikkerhetspraksis, teknikker og verktøy tidlig i utviklingsprosessen. |

| | |
|---|---|
| | Målet er å forhindre vanlige sikkerhetsfeil som kan utnyttes senere, ved å bygge inn sikkerhet i koden fra starten av. |
| Sikkerhetsarkitektur | Sikkerhetsarkitektur er hvordan sikkerhetstiltak og kontroller er strukturert og utformet i et system. Den beskytter informasjon og ressurser mot trusler, sårbarheter og uautorisert tilgang. |
| Statisk kodeanalyse | Statisk kodeanalyse er en metode for å undersøke kildekoden uten å kjøre programmet, med mål om å identifisere feil, sårbarheter og avvik fra kodestandarder. |
| Systemeier | Ansvarlig for beslutninger knyttet til aksept eller avvisning av risikoer, samt allokering av budsjett til sikkerhetstiltak og risikoreduserende tiltak. Vurderer om sikkerhetsinvesteringer er innenfor budsjettet. |
| Systemutvikling | Systemutvikling er prosessen med å planlegge, designe, utvikle, teste og implementere programvareløsninger som tilfredsstiller spesifikke behov og krav. |
| Teamlead | En teamlead for programvareutvikling koordinerer teamets arbeid, sørger for at sikkerhetsprinsipper følges og at risikohåndtering er en del av prosessene. De fjerner hindringer og sikrer at teamet stadig forbedres og utvikler sin kompetanse. |
| Techlead | En techlead for programvareutvikling har ansvar for teknisk arkitektur og kodekvalitet med fokus på sikkerhet. De utfører risikovurderinger, anbefaler tiltak, og sørger for at sikkerhet er integrert i utviklingsprosessen og automatiserte leveranseløp. |
| Utvikler | En utvikler skriver sikker kode basert på prinsipper for sikker koding, og utfører sikkerhetstesting. De rapporterer risikoer, oppdaterer tredjepartskomponenter og følger med på applikasjonens sikkerhet når den er i drift. |