Introduktion til software sikkerhed Del 6

Del 6 – En mere sikker udviklingsproces.

Af Martin Edwin Schjødt Nielsen

University College

En mere sikker udviklingsproces

Sikkerhed bør som udgangspunkt ikke være en parallel sideløbende proces eller noget, man tænker ind efter, man har udviklet et system. Sikkerhed bør indarbejdes løbende, på lige fod med systemets øvrige krav. Alle sikkerhedsovervejelser og tiltag bør indgå som en fast del af en udviklingspraksis og -proces.

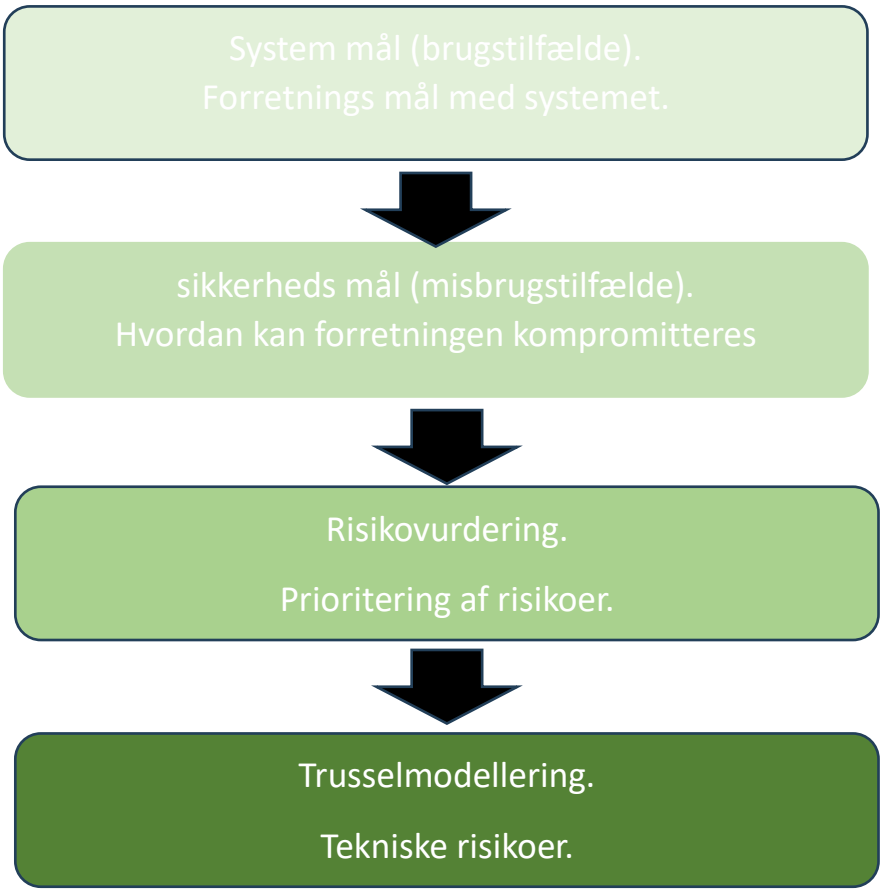
I dette afsnit gives et eksempel på, hvordan en praksis og proces kan sammensættes. Det er vigtigt at understrege, at eksemplet på ingen måder er den eneste rigtige tilgang. Hvert team har sin egen tilgang og proces. Men i processen bør overvejelser fra Software sikkerhedens 3 søjler og De 7 berøringspunkter indgå.

Software sikkerhedens 3 søjler

Der er tre overordnede fokuspunkter inden for sikker softwareudvikling. Disse tre benævnes Software sikkerhedens 3 søjle. De tre søjler er: Anvendt risikostyring, berøringspunkter og viden/vidensdeling.

Anvendt risikostyring

Denne søjle specificerer at der bør anvendes risikostyring i forbindelse med udvikling af software såvel som den løbende drift. Desuden bør der være fastsatte rammer for risikostyring. Både på det forretningsmæssige niveau og det tekniske niveau. Risikostyring på forretnings niveau afdækkes med misbrugstilfælde, herefter afdækkes de tekniske risikoer gennem trusselsmodellering.



Figur 1 Taksonomi med risikostyring på forretningsmæssigt niveau og teknisk niveau

berøringspunkter

Sikkerhed bør ikke være noget, der tænkes ind efter systemet er udviklet. Det bør tænkes ind og udvikles på lige fod med det øvrige system. Dette uddybes i afsnittet "De 7 berøringspunkter.

Viden/vidensdeling

Softwareudvikling er en disciplin baseret på viden, og software sikkerhed er ingen undtagelse. Kompetencer inden for software sikkerhed bygger på en kombination af erfaringer og viden. Det er derfor vigtigt at sikre vidensdeling i teamet såvel som i virksomheden.

Herudover bør man anvende standarder såsom Owasp application security verification standard og orienterer sig i awarenes lister som F.eks. Owasp top-10

De 7 berøringspunkter

De 7 berøringspunkter er alle områder, der bør berøres i løbet af en udviklingsproces. Hvert punkt bliver beskrevet i de følgende underafsnit.

Misbrugstilfælde

Misbrugstilfælde bliver beskrevet i del 3 af denne serie. Alle misbrugstilfælde udledes fra systemets brugstilfælde. Brugstilfælde udgør systemets mål, og misbrugstilfælde udgør systemets sikkerhedsmål.

Det er værd at bemærke, at brugstilfælde er dynamiske, da målene for et system ofte ændrer sig løbende i udviklingsprocessen. Dette betyder, at misbrugstilfældene også er dynamiske og derfor bør evalueres løbende.

Sikkerhedskrav

Sikkerhedskrav bliver udledt fra systemets sikkerhedsmål og skal indarbejdes på lige fod med de øvrige krav til softwaren. I seriens del 5 blev trusselsmodellering brugt til at omsætte sikkerhedsmål til sikkerhedskrav. Dette kan være en fremgangsmåde. Desuden kan man vælge at indarbejde sikkerhedskravene ind i et eksisterende krav. F.eks. ”Den autentificerede bruger kan se de filer, han er ejer af, og derfor er autoriseret til at se.”.

Risiko analyse af arkitektur

Systemarkitekturen bør vurderes for eventuelle risici. Dette udføres med trusselsmodellerings fremgangsmåden beskrevet i afsnittet Trusselsmodellering.

Review af kode

Kodegennemgang (Code Review) er en af de mest kendte praksisser inden for softwareudvikling, og dette gør sig også gældende inden for software sikkerhed. Man bør benytte sig af to tilgange: manuel kodegennemgang udført af et menneske. Her kan der med fordel kigges efter fejl i kodens forretnings- og domænelogik. Derudover bør der anvendes et automatiseret statisk kodeanalyseværktøj til Taint-analyse og kode skanning for at detektere kode, der er uhensigtsmæssig i forhold til sikkerhed.

Herudover (**ikke i stedet for**) kan man også anvende såkaldte "Linters", som er statisk kodeanalyseværktøjer, der integreres ind i en IDE eller teksteditor og kan identificere dårlig praksis mens der skrives software.

Penetration testing

Penetration testing er en blackbox-test, hvor man tester systemet med forskellige inputs og forsøger at omgå sikkerheden. En penetrationstest kan være noget så simpelt som at skrive et lille stykke kode i et

tekstfelt, der er beregnet til f.eks. en persons fornavn. En mere avanceret tilgang kan også anvendes, hvor man typisk bruger et værktøj som f.eks. Metasploit framework, Burp suite , OWAP ZAP

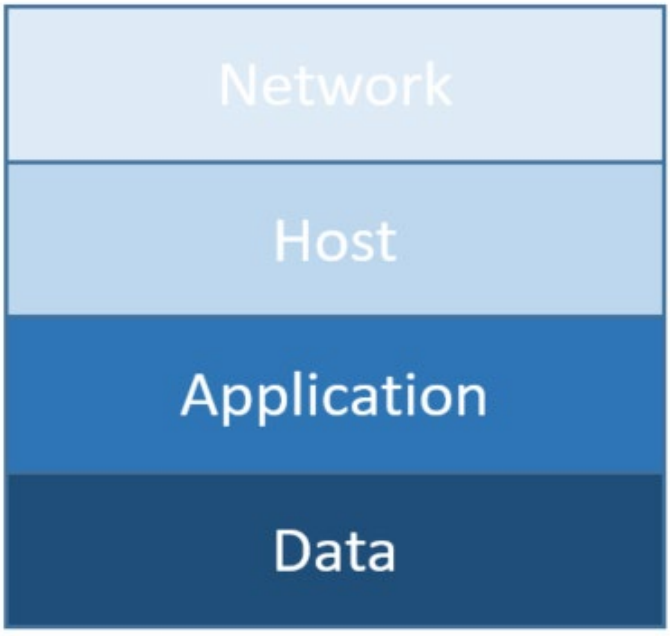
Penetrationstests bør udføres løbende på systemet, og dele heraf kan automatiseres med F.eks. scripts, eller dertil egnet værktøjer.

Risiko baseret sikkerheds test

Risikobaseret sikkerhedstest tager udgangspunkt i, at områder med høj risiko kræver mere dybdegående tests. En enkelt penetrationstest er ikke tilstrækkelig til at validere sikkerheden i disse tilfælde. For eksempel kan et kritisk område med høj risiko kræve, at mindst 90% af koden er dækket af unit tests (code coverage) som en foranstaltning for at reducere risikoen. Hvis der tidligere er identificeret et misbrugstilfælde med en meget høj risikofaktor, kan der stilles særligt strenge krav til den del af koden, der implementerer dette misbrugstilfælde.

Sikkerheds operationer.

Et udviklingsteam bør havde et overordnet kendskab til alt den infrastruktur som softwaren driftes på. Dette dækker alt fra operativsystemer til netværk. En større forståelse for infrastrukturen som applikationen driftes på, vil give softwareudviklerne et mere holistisk perspektiv på softwaren og sikkerheden. Dette taler ind til den sikkerheds strategi man kalder *Defense in depth.* Vist i Figur 2



Figur 2 Defense in depth

Tanken bag strategien er at sikkerhed opretholdelse i flere lag. Her er det relevant for udviklingsteamet at vide hvor sikkerheden bliver opretholdt, og ikke opretholdt. Og kordinerer sikkerhedsindsatsen med F.eks. involveret drifts/infrastruktur teams.

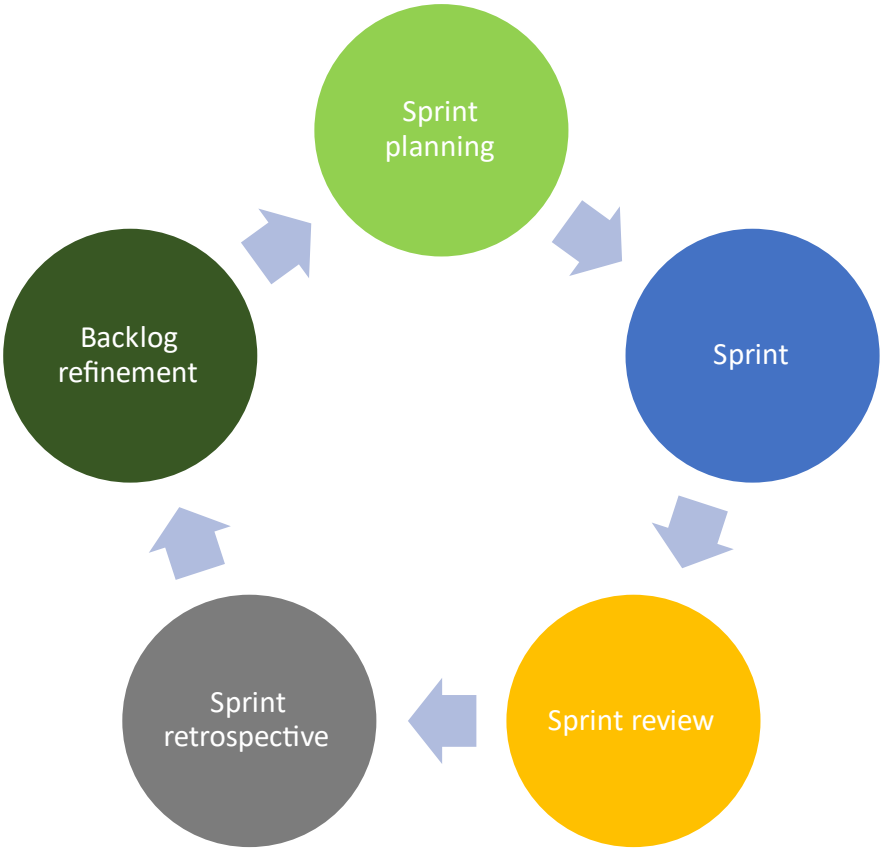
Opsamling - Et eksempel på en mere sikker udviklingsproces.

I dette afsnit introduceres et eksempel på hvordan man kunne indarbejde software sikkerhedens 3 søjler i en udviklingsproces. Jeg har valgt Scrum som rammeværktøj til udviklingsprocessen, dette må på en ingen måder tolkes som at være den eneste rigtige løsning, blot **et** eksempel. De 3 søjler bør kunne indarbejdes i

næste alle udviklingsprocesser. Hvordan de skal indarbejdes, er dog op til de enkelte teams som arbejder med processen.

Scrum

En komplet gennemgang af SCRUM er udenfor rammerne af dette dokument. Men overordnet defineres processen i SCRUM som vist i Figur 3

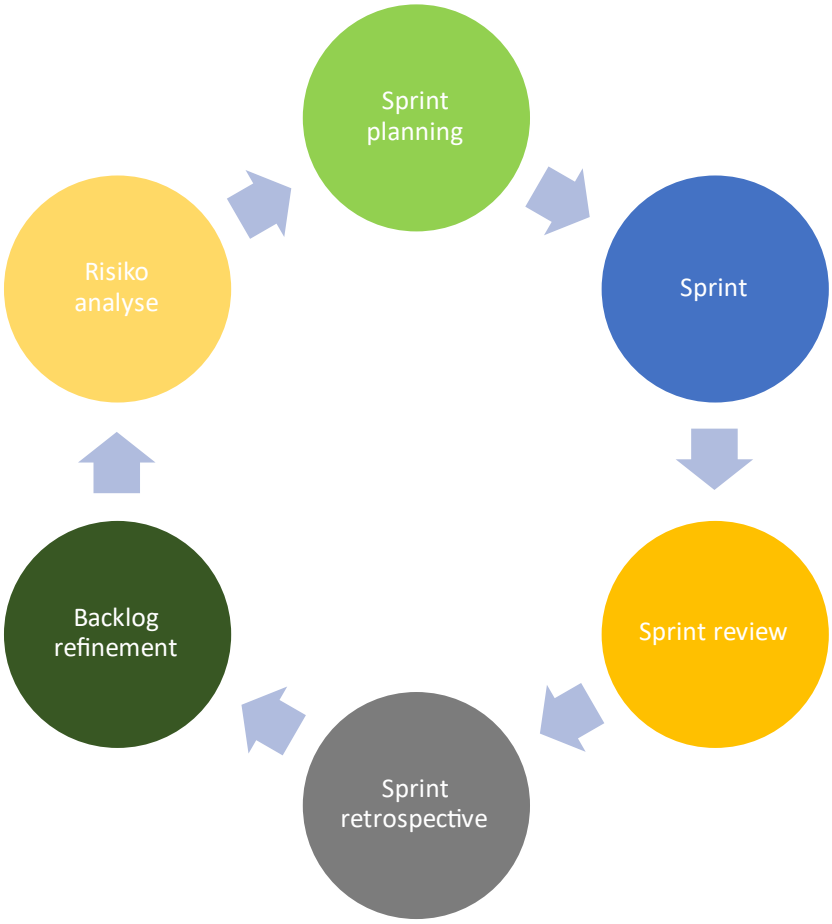


Figur 3 Scrum processen

Mere information om Scrum kan findes i dokumentet Scrum guide, der kan hentes gratis på siden www.scrum.org/resources/scrum-guide

Anvendt risikostyring i Scrum processen.

Det formodes at udarbejdelse og ændringer af systemets brugstilfælde bliver til under et backlog refinement møde. Som en opfølgning på dette kunne der udarbejdes nye risiko analyser som beskrevet i del 4 risikovurdering og styring. Figur 4 viser dette i SCRUM processen.

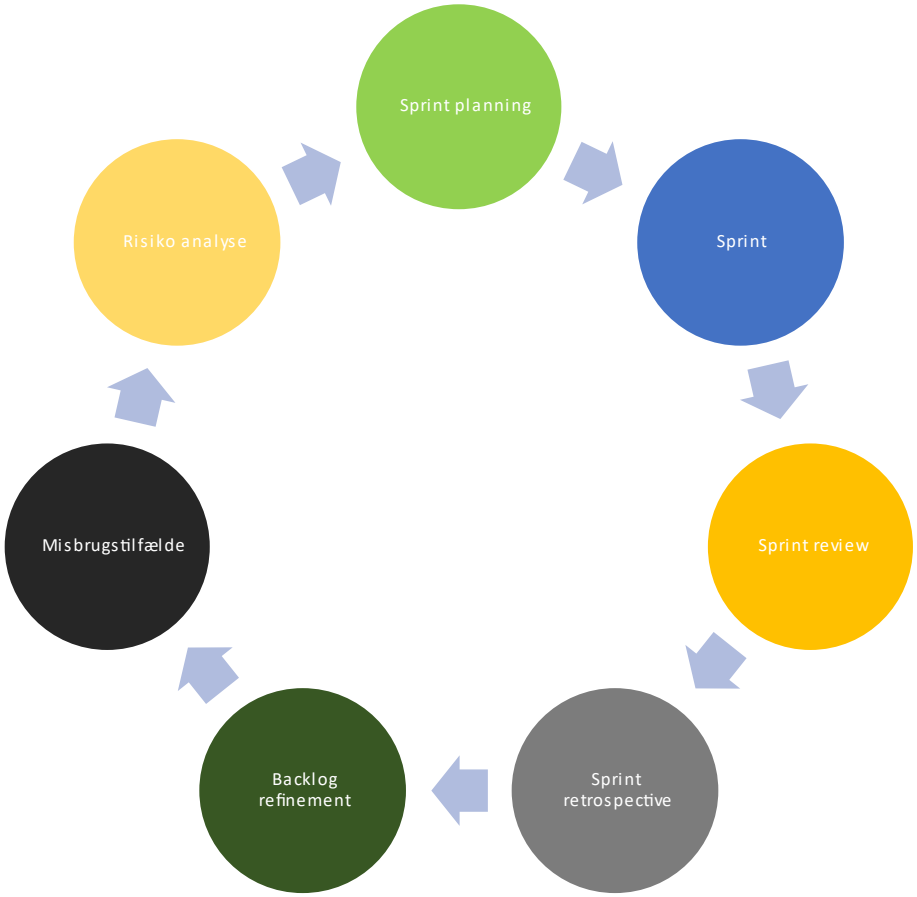


Figur 4 Risiko analyse i Scrum

Her er det dog værd og bemærke at misbrugstilfælde som skaber input til Risiko analysen endnu ikke er indsat i processen. Risikoanalysen kunne også være en del af backlog refinement mødet, dette er op til det enkelte team.

De 7 berøringspunkt i Scrum processen.

Hvert af de 7 berøringspunkter kan indarbejdes i SCRUM. Indledningsvist kan vi indarbejde udledning af misbrugstilfælde som sin egen selvstændig aktivitet, som vist i Figur 5



Figur 5 Misbrugstilfælde i Scrum processen

Da kravspecifikationerne for systemmålene typisk ændres eller udvides i backlog refinement mødet, er det naturligt at dette møde efterfølges af en session hvor man evaluerer om de ændret krav har udledt nogle nye misbrugstilfælde. Eller ændret noget på de eksisterende tilfælde. Her kan man benytte sig af fremgangsmåden der blev beskrevet i seriens del 3. Altså man brainstormer ud fra et diagram med brugstilfælde og misbrugstilfælde, og vurder om der nogle nye misbrugstilfælde, eller ændringer ift. de eksisterende misbrugstilfælde.

Berøringspunktet *Risikoanalyse af arkitektur* kan afdækkes med trusselsmodellering hvor arkitekturen tegnes op. Da Risiko analysen skaber input til trusselsmodellering i form af prioriteret og risikovurderet misbrugstilfælde, er det naturligt at trusselsmodellerings aktiviteten kommer efter risikoanalysen.

Figur 6 Trusselsmodellering i Scrum

Trusselsmodellering aktiviteten skaber også sikkerhedskrav der fungerer som input til Sprint planning og tages i betragtning med øvrige system krav, hvilket også afdækker berøringspunktet sikkerhedskrav.

Berøringspunkterne Penetration testing og risikobaseret sikkerheds test af kode, afdækkes ved at test cases skrives ind i *Definition of done* for hvert enkelt krav. Det vil sige at inden sprintet påbegyndes, bør test cases være fastlagt. Ende videre kan der opstilles dynamiske sikkerheds test (DAST).

Berøringspunktet kode review kan afdækkes delvist med en praksis i teamet om at alt kode skal være peer reviewet inden dette kommer i produktion. Bruges der versionsstyring kan man aftale at kode ikke kan integreres ind i et depot(repository) uden at først være peer reviewet af en anden udvikler. Særligt fejl i forretningslogikken er vigtigt at kigge efter for sårbarheder. Der kan også med fordel opstilles krav om høj test *code coverage* for at minimerer antallet af fejl. Ud over det manuelle review af koden, bør der anvendes statisk kode analyse til taint analyser og kode sårbarheds skanninger(SAST). Depot udbydere såsom Gitlab og Github understøtter automatisk taint analyse og kode sårbarheds skanninger til de fleste større programmeringssprog. Man kan også selv lave en opsætning med F.eks. SonarQube.