4 Koding

Sjekklisten er dynamisk, ikke uttømmende og skal oppdateres regelmessig.
Dersom du har innspill til noen av punktene, vil vi gjerne høre fra deg.

Mulige tiltak for sikker koding

- Lage liste over godkjente verktøy og biblioteker
- Skanning av avhengigheter for kjente sårbarheter eller utdaterte versjoner
- Manuell kodegjennomlesning
- Statisk kodeanalyse med sikkerhetsregler

Bruk godkjente verktøy og rammer

- Etabler en liste over
  - godkjente verktøy med tilhørende sikkerhetsfunksjonalitet som skal bidra til å automatisere og håndheve sikkerhetsrutiner i kodingen
  - godkjente støttekomponenter
  - tillatte tredjepartskomponenter og utviklingsverktøy. Unngå deling av personopplysninger gjennom tredjepartsbiblioteker - bruk heller syntetiske data
- Beskriv i listene hva de ulike verktøy og støttekomponenter skal brukes til. Dette inkluderer nye sikkerhetsanalyser, funksjonalitet og beskyttelse. Verktøy og støttekomponenter skal risikovurderes og analyseres for sårbarheter.
- Hold listene oppdatert i henhold til virksomhetens retningslinjer. Det innebærer at nye verktøy og versjoner må gjennomgås, og tilstrebe at disse benyttes.
- Bruk kun godkjente verktøy og støttekomponenter fra listen. Eventuelle unntak bør dokumenteres og godkjennes av sikkerhetsansvarlig.

Kodemønstre og kodemaler for mye brukt/opprettet funksjonalitet bør generaliseres, kvalitetssjekkes og dokumenteres som gjeldende mønster/mal. Eksempler er hvordan databasekall skal lages og hvilken struktur det skal være.

Eksempel på godkjente verktøy og støttekomponenter som må med i lister:

- kodebibliotek
- programmeringsspråk
- versjonskontroll
- testverktøy
- infrastruktur
- overvåkingsverktøy
- loggingserver
- tredjeparts rammeverk og APIer

Eksempel på praktisk tilnærming:
Virksomheten har besluttet en metodikk for koding ved utvikling og videreutvikling. Denne gjelder dermed for alle prosjekter og prosjektledere. De benytter et system for oppgavehåndtering, slik som Jira eller Confluence. Alle oppgaver har en dedikert eier som også har ansvar for personvern og sikkerhet i oppgaven. Status på alle aktive oppgaver gjøres på morgenmøtet for teamet og utfordringer som har dukket opp rundt for eksempel personvern diskuteres. Kodekontroll må gjøres

av en annen utvikler. Dette er viktig for å unngå personavhengigheter i koden samt øke kodekvalitet, men vel så viktig for å få en ekstra person til å se gjennom om personvern og sikkerhet er ivaretatt. Det bygges versjoner for testing og det settes en eier for produksjonssetting.

Ugyldiggjør utrygge funksjoner og moduler

- Utrygge funksjoner og moduler håndteres av bibliotek, eksempelvis OWASP Dependency Check.
- Deaktiver unødig sporing, logging og innsamling av personopplysninger.

Statisk kodeanalyse og kodegjennomgang

- Etabler virksomhetsrutiner og/eller -sjekklister for statisk analyse og gjennomgang av kode.
- Analyser og gjennomgå kildekoden regelmessig, og hver gang den bygges.
- Kontroller dataflyt, lagring og mellomlagring.
- Statisk analyse for personvern gjennomføres i hovedsak manuelt, da det er begrenset med «automatiserte» verktøy for kodeanalyse for personvern. Det kan være vanskelig å fange mønstre da data alene ikke nødvendigvis er å anse som personopplysninger, men at kobling av ulike typer data kan gi personopplysninger.
- Viktig at den som gjør arbeidet (reviewer) har god kunnskap både om personvernprinsippene og om kravene til innebygd personvern.
- Ha forskjellige nivå av skanning, eksempelvis for utviklere, sikkerhetsrådgivere og den som er ansvarlig for produksjonssetting.
- Etabler føringer for hva som skal skannes, og når.

Eksempel på verktøy for statisk kodeanalyse:

- RIPS PHP Static Code Analysis Tool
- OWASP LAPSE+ Static Code Analysis Tool
- SonarQube
- Checkmarx.

Eksempel på hvordan statisk kodeanalyse kan gjennomføres:

- Regelmessig skanning, for eksempel daglig eller ukentlig.
- Skanning gjøres av både Scan Master og Dev Brancher, og det gjøres en full sikkerhetsskanning.
- Ved nybygging (On build scanning) sjekkes minimum sikkerhetskrav av Dev Brancher eller Scan Master (eks. SQL-Injection).
- «On Demand scanning» gjøres av brukerens IDE og er en full sikkerhetsskanning
- Bruk sjekklister for kodegjennomgang:
    - Eks. A1. Finn alle steder med databaseaksess. Konkateneres spørringene med skitne variabler?
    - Eks. A4. Hvordan blokkeres brukere fra andre brukeres data?

Egenskaper man kan se etter i et verktøy for kodeanalyse:

- Det er designet for sikkerhet.
- Det støtter flere nivåer (ulike programmeringsspråk og plattformer).
- Det må være mulig å utvide, verktøyet bør kunne utvides til å inkludere nye angreps- og forsvarsteknikker.
- Det må være nyttig både for sikkerhetsanalytikere og utviklere.

- Det støtter eksisterende utviklingsprosesser.
- Det gir mening for flere som har eierskap til utviklingen (for eksempel grensesnitt med målinger som kan støtte avgjørelser om produksjonssetting, kontrollere kostnader ved endringer, og gi informasjon som er nødvendig for vedlikehold).

Hvorfor sikker koding?

- Personvernforordningen skal komme til anvendelse når programvare skal utvikles, jf. artikkel 25.
- Verktøy, støttesystem og infrastruktur skal være «state of the art», det vil si den nyeste og mest oppdaterte versjonen av teknologien, jf. artikkel 25.
- Det skal foreligge en forankring i ledelsen om sikker og felles metodikk for
    - koding
    - å oppdage og fjerne sårbarheter fra koden
    - bruk av automatiske verktøy for kodeanalyse
    - statisk kodeanalyse og kodegjennomgang