5 Test

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

Test om personvern- og sikkerhetskravene som ble satt i kravsetting, faktisk er implementert, og om de er riktig implementert

Husk at personopplysningsregelverket også gjelder for utviklings- og testmiljø.

- Etabler en helhetsforståelse for funksjonalitet og informasjon.
- Test om krav og de designmessige komponentene har ivaretatt sikkerhet- og personvernkrav. Dette kan for eksempel gjøres ved å lage standard testscenarier basert på funksjonelle krav som kan gjenbrukes på tvers i virksomheten.
- Utarbeid en sjekkliste på om alle komponentene – for å være i samsvar – er bygget inn. Dette inkluderer også nye komponenter som opprinnelig ikke var spesifisert i kravsetting.

  Eksempler:
  - Alle innstillinger skal som standard, være konfigurert med den mest personvernvennlige innstillingen.
  - De skal kunne eksportere og importere den registrertes informasjon (dataportabilitet).
  - Lagres data på rett sted?
  - Er dataene som samles inn nødvendig?
  - Den registrerte skal kunne gi samtykke (også mindreårige og umyndige personer).
  - Den registrerte skal kunne nekte eller trekke tilbake et samtykke.
  - Er det mulig å avslutte en kontrakt/avtale, installere, avinstallere, aktivere og deaktivere et program, tjeneste, teknisk komponent eller et system?
  - Tilgangskontroll
    - bare for autoriserte
    - tilgjengelig for den registrerte
    - gi tilgang til å rette, blokkere eller slette personopplysninger
  - Kunne sende inn forespørsler eller klager knyttet til personvern og sikkerhet.
  - Den registrerte kan motsette seg profilering.
  - Få informasjon om hvordan automatiske avgjørelser tas, hva som samles inn, behandles, hvor personopplysninger lagres, åpenhet med videre

- Test at varslingsrutiner finnes og fungerer, at mottak fungerer (for eksempel utarbeide et utkast til tekst som man kan benytte ved en hendelse inkludert tiltak for å hindre at lignende skjer igjen), samt hva som er gjort for å forhindre innvirkning på den registrerte.
- Bruk syntetiske personopplysninger og sjekk riktighet, lovlighet, rettferdighet, formål, nøyaktighet, dataminimalisert, lagring og robusthet.
  - Bruk reelle data kun ved kvalitetssikring når programvaren er implementert i produksjonsmiljø og brukere skal ta i bruk programvare. Test på reelle data gjøres av fagpersoner som er autoriserte for opplysningene, eksempelvis superbrukere for barnevernprogram i barnevernssektoren, superbrukere for legejournal på legekontor, superbruker på skolesystem på skolen.
- Søk etter reelle personopplysninger og sjekk hvorfor de fremkommer. Dette kan for eksempel gjøres ved jevnlig skanning etter mulige lekkasjer av fødselsnummer.

Sikkerhetstest ved å utfordre sårbarheter i programvaren

- **Dynamisk testing:** Test funksjonalitet i kjørende kode ved å bruke verktøy eller manuell gjennomgang som analyserer hvordan programvaren oppfører seg ved ulike brukerrettigheter og kritiske sikkerhetsfeil. Testing skal sikre at brukere kun får tilgang til den informasjon og funksjonalitet som de har rettigheter til. Verifiserer at forsøk på å tilegne seg uautorisert informasjon logges som sikkerhetsfeil.
  - Whitelisting er å foretrekke framfor blacklisting i hva som kan skrives inn i programvaren.
  - Gjør sårbarhetsanalyse av applikasjons- og nettverkslag. Analysen skal måle effekten av iverksatte tekniske og organisatoriske sikkerhetstiltak. Bruk testverktøy. Test for eksempel på standardpassord, passord i filer, SQL injection, script injection, logger, backups, tmp-områder, Cross-site scripting, manglende validation.
  - Test tilgangsstyring og den faktiske tilgangen til brukere:
    - Eksempel for GET requests: logg inn bruker 1, kopier alle lenker, logge ut av bruker 1. Logg inn bruker 2 osv. Sjekk alle lenker. Test alle rettighetsnivå, benytt minimum to brukere i hvert tilgangsnivå.
    - Enkel testing av ikke-teknikere, bruk for eksempel nettleser og sjekk om GET requests i nettleser URL gir tilgang.
    - Avansert testing av teknikere, bruk for eksempel burpsuite og burp repeater for å undersøke POST requests og session cookies.
    - Alle parametere som kan itereres (eks. Id=42, 43, 44 etc.) skal itereres og sjekkes for tilgangskontroll.
    - Undersøk cookie og sesjonshåndtering, for eksempel hvordan innlogging og tilgangskontroll fungerer basert på cookie (Genereres ny cookie etter innlogging, blir sesjonen destruert på serverside eller slettes den kun fra brukerens nettleser?).
- **Fuzz testing:** Testaktiviteten gjennomføres ved å fremprovosere feil i programvaren. Dette kan gjøres ved å bruke verktøy som sender tilfeldig og misformet data (feil inputverdier) i alle mulige input-felt til programvaren, enten manuelt eller ved bruk av intelligente verktøy som analyserer sårbarheter i webapplikasjoner. Dersom programvaren har flere grensesnitt, bør man etterstrebe å teste hvert enkelt av dem. Det bør brukes verktøy som kan analysere output og applisere sikkerhetskontekst.
- **Penetrasjonstesting/sårbarhetsanalyse:** Kjør penetrasjonstest eller sårbarhetsanalyse av nyutviklet programvare før produksjonssetting, ellers regelmessig for å oppdage sårbarheter. Benytt helst ulike testere for hver gang. Sikkerhetstester som dette skal være lovlig og autorisert forsøk på å finne, utnytte og avdekke sårbarheter. Deretter skal det implementeres tiltak som gjør programvaren mer robust. Merk at det kan være påkrevd å ha på plass underskrevne avtaler ved penetrasjonstesting.
  - Kan gjennomføres både internt og fra ekstern hold, vurder hva er mest hensiktsmessig i forhold til ny funksjonalitet/tjeneste.
  - Bruk sikkerhetseksperter både for gjennomføring og analysering av resultat.
  - Test de sikkerhetstekniske tiltakene ved innebygd personvern. Eksempler på sårbarheter som kan avdekkes er passord, installere Remote Controls på nettlesere, klienter eller servere, dumpe databaser, dataoverføring eller lagrede data inkludert caching i minne, forstyrre eller overta sesjoner eller oppkoblinger, utnyttelse av cookies, knekke kryptering, utnytte mangelfull tilgangsstyring m.m.
- Test i flere instanser
  - Testmiljø, for eksempel kodegjennomgang av applikasjonen (testing av sårbarheter i kode).

    - Integrasjons- og systemtestmiljø, for eksempel statisk og dynamisk test av applikasjon.
    - Testing i produksjonsinstansen, for eksempel fuzzing av applikasjon, sårbarhetsskann og penetrasjonstesting av applikasjon og infrastruktur.
- Ha regime for automatisk kjøring av testsett før produksjonssetting, for eksempel sikkerhetstester som kjøres hver gang en applikasjon bygges eller videreutvikles. Dersom det oppstår en feil i produksjon, kan man unngå tilsvarende feil ved å lage automatisk testing av feilen og legge denne til i testsettet.

Eksempler på verktøy som kan benyttes ved sikkerhetstesting:

- OWASP testing guide mot OWASP top 10 sårbarheter, SANS/CIS 20 sårbarheter
- Whitebox fuzz testing
- Burp suite og burp repeater for kodegjennomgang
- Bug bash og «Feedback Hub»

Gjennomgang av angrepsflaten

- Verifiser at angrepsvektorer som er avdekket i designfasen er håndtert.
- Verifiser at nye angrepsvektorer introdusert under implementasjon er identifisert og håndtert.
- Gjennomgå trusselvurderingen opp mot nyutviklet programvare.
- Revurder personvernskonsekvensene som ble gjort i kravsettingsfasen, ettersom krav kan endre seg underveis i utviklingsprosessen uten å ha blitt gjennomgått.
- Se spesielt på hvorvidt det overføres mer data enn hva applikasjonen har bruk for, og dermed at slik data er uten beskyttelse og lovlighet.

Hvorfor stille krav til verifikasjon og test?

- Personvern- og sikkerhetskrav som ble satt i kravsetting skal være implementert, og riktig implementert, jf. artikkel 5, 25, 35 og 11-23.
- Feil inputverdier skal ikke bidra til konfidensialitet-, integritet eller tilgjengelighetsbrudd, jf. artikkel 32.
- Angrepsflaten skal ikke ha sårbarheter som kan bidra til konfidensialitet-, integritet eller tilgjengelighetsbrudd, jf. artikkel 32.