Ohjelmisto-turvallisuus kehitystyössä

- Roolit ja osaamistarpeet

TRAFICOM
Liikenne- ja viestintävirasto
Kyberturvallisuuskeskus

Liikenne- ja viestintävirasto Traficom
Kyberturvallisuuskeskus
ISSN 2669-8757 (verkkojulkaisu)
ISBN 987-952-425-017-7

Traficomin julkaisuja
12/2026

Julkaisun nimi
Ohjelmistoturvallisuus kehitystyössä – Roolit ja osaamistarpeet

Tekijät
Miina Saros ja Karoliina Mancuso, Solita

Sisällysluettelo

Johdanto

Oppaan tarkoitus

Tämän Liikenne- ja viestintävirasto Traficomin Kyberturvallisuuskeskuksen laatiman oppaan tarkoituksena on tukea organisaatioita vahvistamaan turvallisen ohjelmistokehityksen osaamista. Tavoitteena on varmistaa, että ohjelmistojen turvallisuus huomioidaan järjestelmällisesti koko ohjelmiston elinkaaren ajan suunnittelusta ja toteutuksesta aina ylläpitoon ja käytöstä poistoon asti. Opas kuvaa, millaista osaamista eri ohjelmistokehityksen rooleissa tarvitaan, jotta tietoturva voidaan sisällyttää luontevaksi osaksi arjen tekemistä ja päätöksentekoa.

Oppaan tavoitteena on tukea organisaatioita tunnistamaan ja kehittämään ohjelmistokehityksen roolikohtaista tietoturvaosaamista. Tavoitteena on varmistaa, että jokaisella roolilla on edellytykset huomioida turvallisuus osana omaa työtään, yhteistyötä ja päätöksentekoa, ja että tietoturva kehittyy osaksi organisaation arkea ja toimintakulttuuria. Kun turvallinen ohjelmistokehitys on osa jokapäiväistä tekemistä, se tukee paitsi yksittäisten ohjelmistojen myös koko organisaation ja yhteiskunnan kykyä toimia häiriöttömästi digitaalisessa ympäristössä.

Kuinka opasta käytetään?

Johdanto-osio luo perustan ymmärrykselle siitä, mitä turvallinen ohjelmistokehitys tarkoittaa ja miksi turvallisuuden huomioiminen on olennainen osa ohjelmistojen laatua ja luotettavuutta. Lisäksi johdanto kuvaa, miten turvallisuuteen liittyvät näkökulmat korostuvat eri tavoin ohjelmiston elinkaaren eri vaiheissa.

Oppaan ydin koostuu roolikohtaisista osioista, joissa kuvataan kunkin ohjelmistokehityksen roolin kannalta keskeiset osaamistarpeet turvallisuuden varmistamiseksi. Roolikohtaiset kuvaukset auttavat tunnistamaan, millaista tietoa ja taitoa tarvitaan eri vaiheissa, jotta turvallisuus tulee huomioiduksi osana päivittäistä kehitystyötä ja yhteistyötä muiden roolien kanssa. Voit tarkastella opasta oman roolisi näkökulmasta arvioidaksesi omia osaamisalueitasi tai hyödyntää sitä laajemmin koko kehitystiimin tukena, kun haluatte vahvistaa yhteistä ymmärrystä turvallisesta ohjelmistokehityksestä.

Huomioi, että oppaassa kuvatut roolit ja niihin liitetyt osaamistarpeet eivät ole tyhjentäviä, eikä roolien nimet tai vastuut ole kaikissa organisaatioissa samanlaisia. Roolit ja tehtävien painotukset voivat vaihdella organisaation rakenteen, koon ja toimintatapojen mukaan. Opasta voi soveltaa joustavasti eri tilanteisiin ja täydentää niillä rooleilla, jotka ovat omassa organisaatiossa keskeisiä oman organisaation ohjelmistokehityksessä.

Termistö

Termi	Selitys
Artefakti	Ohjelmistokehityksessä syntyvä lopputuotos, kuten lähdekoodi, käännetty ohjelmisto, konttikuva tai dokumentaatio.
CI/CD-putki	Automaattinen kehitys- ja julkaisuketju, jossa koodi testataan, rakennetaan ja julkaistaan hallitusti.
DAST (Dynamic Application Security Testing)	Dynaaminen tietoturvaskannaus, joka testaa käynnissä olevaa ohjelmistoa ja etsii ajonaikaisia haavoittuvuuksia.
Deny by default	Periaate, jossa kaikki pääsy on estetty oletusarvoisesti ja sallitaan vain erikseen määritellyt oikeudet.
Fuzz-testaus	Testausmenetelmä, jossa ohjelmistolle syötetään virheellistä tai odottamatonta dataa kaatumisten ja heikkouksien löytämiseksi.
Hyökkäyspinta	Kaikki ne kohdat, joiden kautta ohjelmistoon voidaan yrittää vaikuttaa, kuten rajapinnat, käyttäjät, integraatiot ja konfiguraatiot.
IaC-skanneri (Infrastructure as Code -skanneri)	Työkalu, joka tarkistaa infrastruktuurikoodin, kuten pilvi- ja palvelinmääritykset, ja etsii niistä virheellisiä tai tietoturvaa heikentäviä asetuksia ennen käyttöönottoa.
IAST (Interactive Application Security Testing)	Testausmenetelmä, joka tarkkailee sovelluksen toimintaa sen suoritushetkellä ja yhdistää staattisen ja dynaamisen testauksen piirteitä.
Konfiguraatio	Ohjelmiston tai ympäristön asetukset, joilla määritellään, miten järjestelmä toimii eri tilanteissa.
Konttiskannaus (Container Skanning)	Työkalu tai menetelmä, joka tarkistaa konttikuvat ja niiden sisältämät kirjastot, käyttöjärjestelmäkomponentit ja asetukset tunnettuja haavoittuvuuksia ja virheellisiä konfiguraatioita vastaan ennen käyttöönottoa.
Käytöstä poisto	Ohjelmiston hallittu alasajo, jossa data, käyttöoikeudet ja tekniset riippuvuudet poistetaan turvallisesti.
Lokitus	Ohjelmiston tapahtumien kirjaaminen, jotta poikkeamat, virheet ja tietoturvatilanteet voidaan havaita ja tutkia.
Monitorointi	Jatkuva seuranta, jolla havaitaan häiriöt, poikkeamat ja mahdolliset tietoturvauhat mahdollisimman varhain.
Observointityökalu	Työkalu, jolla seurataan ohjelmiston toimintaa, suorituskykyä ja poikkeamia esimerkiksi mittareiden ja lokien avulla.
OWASP (Open Worldwide Application Security Project)	Kansainvälinen yhteisö, joka tuottaa ohjeita, viitekehyksiä ja listauksia ohjelmistoturvallisuuden tueksi.
Pääsynhallinta	Mekanismit, joilla määritellään kuka saa tehdä mitä ohjelmistossa ja mihin tietoihin käyttäjällä on oikeus.

Termi	Selitys
Regressiotestaus	Testaus, jolla varmistetaan, etteivät muutokset tai korjaukset riko aiemmin toimineita ominaisuuksia.
Rollback-suunnitelma	Etukäteen määritelty toimintamalli, jolla ohjelmisto voidaan palauttaa aiempaan toimivaan versioon ongelmatilanteessa.
Runbook	Käytännön toimintaohje häiriö- tai poikkeamatilanteisiin, joka tukee nopeaa ja johdonmukaista reagointia.
Salaisuuksien hallinta	Käytännöt ja työkalut, joilla käsitellään tunnistetietoja kuten salasanoja, avaimia ja sertifikaatteja turvallisesti.
Salaisuusskannaus (Secret Scanning)	Työkalu tai menetelmä, joka etsii lähdekoodista ja versiohallinnasta vahingossa tallennettuja salasanoja, avaimia ja muita luottamuksellisia tietoja.
SBOM (Software Bill of Materials)	Luettelo ohjelmiston sisältämistä komponenteista, kirjastoista ja niiden versioista.
Secure by Design	Suunnitteluperiaate, jossa tietoturva rakennetaan osaksi ratkaisua alusta alkaen.
Secure coding	Turvallisen koodauksen periaatteet ja käytännöt, joilla vältetään yleisimmät ohjelmistovirheet ja haavoittuvuudet.
SCA (Software Composition Analysis)	Menetelmä tai työkalu, jolla tunnistetaan ohjelmiston käyttämät kolmansien osapuolten komponentit ja niiden haavoittuvuudet.
SSDLC (Secure Software Development Lifecycle)	Tietoturvallinen ohjelmistokehityksen elinkaarimalli, jossa turvallisuus huomioidaan kaikissa vaiheissa.
SAST (Static Application Security Testing)	Staattinen tietoturvaskannaus, joka analysoi lähdekoodia haavoittuvuuksien löytämiseksi ilman ohjelman suorittamista.
Telemetria	Ohjelmiston tuottama mittaus- ja seurantatieto, jota käytetään toiminnan ja tietoturvan seuraamiseen.
Toimitusketjuturvallisuus	Ohjelmiston turvallisuuden varmistaminen myös silloin, kun mukana on kolmansien osapuolten komponentteja ja palveluita.
Tunkeutumistestaus/ penetraatiotestaus	Hallittu ja luvallinen hyökkäystestaus, jossa pyritään löytämään ja hyödyntämään järjestelmän haavoittuvuuksia kuten todellinen hyökkääjä tekisi.
Uhkamallinnus	Menetelmä, jolla tunnistetaan, mihin ohjelmistoon kohdistuu uhkia, mitä pitää suojata ja miten riskejä voidaan pienentää.
Vähimmän oikeuden periaate	Periaate, jonka mukaan käyttäjälle annetaan vain ne oikeudet, joita hän tarvitsee työnsä tekemiseen.

Mitä tarkoittaa turvallinen ohjelmisto?

Turvallinen ohjelmisto on suunniteltu ja toteutettu niin, että sen käyttö, kehitys ja ylläpito huomioivat järjestelmällisesti tietoturvariskit koko ohjelmiston elinkaaren ajan. Se suojaa sekä käyttäjiään että käsiteltävää tietoa, perustuu turvallisen ohjelmistokehityksen periaatteisiin ja mahdollistaa tehokkaan poikkeamien hallinnan. Turvallisuutta ei saavuteta täydellisellä virheettömyydellä, vaan uhkamallinnuksella, ennakoivalla suunnittelulla ja selkeillä toimintamalleilla tietoturvapoikkeamien ja haavoittuvuuksien tunnistamiseksi, ilmoittamiseksi ja korjaamiseksi.

Turvallinen ohjelmistotuottaja erottautuu läpinäkyvyydellä. Se kertoo avoimesti, miten tietoturva on huomioitu kehityksessä ja ylläpidossa, sekä tarjoaa selkeät kanavat haavoittuvuuksien ilmoittamiseen. Läpinäkyvyys lisää luottamusta ja antaa myös ohjelmiston hankkijalle mahdollisuuden arvioida, miten vastuullisesti ohjelmiston turvallisuutta johdetaan.

Hyvin tuotettu ohjelmisto ei ole vain toimiva – se on myös turvallinen.

Ohjelmistoturvallisuus on kaikkien vastuulla

Ohjelmistoturvallisuus ei synny yksittäisen asiantuntijan tai tiimin toimesta, vaan se rakentuu koko organisaation yhteisellä toiminnalla. Johto määrittää suuntaviivat ja resurssit, hankinnoista vastaavat tekevät ratkaisevia valintoja, kehittäjät toteuttavat turvallisuutta käytännössä ja käyttäjät vaikuttavat turvallisuuteen omalla toiminnallaan.

Turvallisuus rakentuu pala palalta arjen päätöksistä, valinnoista ja toimintatavoista. Kun jokainen tuo oman asiantuntemuksensa ja vastuunsa osaksi kokonaisuutta, turvallisuudesta tulee kestävä ja luonteva osa ohjelmistojen elinkaarta. Siksi on tärkeää tunnistaa, miten voi itse edistää turvallisuutta ja että organisaatio tarjoaa siihen riittävästi tukea ja osaamista.

Kun turvallisuus otetaan huomioon läpi koko ohjelmiston elinkaaren ja osaksi jokapäiväisiä tehtäviä, riskit pienenevät ja lopputulos on luotettavampi. Turvallisuudesta tulee osa organisaation kulttuuria: tapa ajatella, tehdä päätöksiä ja toimia yhdessä.

Turvallisuus ei ole erillistä lisätyötä, vaan olennainen osa laadukasta ja vastuullista toimintaa.

Ohjelmiston elinkaari ja tietoturvan painopisteet

Ohjelmiston elinkaari koostuu useista eri vaiheista, joilla kaikilla on oma roolinsa turvallisen ja toimivan ohjelmiston toteutuksessa. Tässä kappaleessa kuvataan ohjelmistokehityksen elinkaaren vaiheet ja kunkin vaiheen keskeiset tietoturvan painopisteet. Lisäksi esitellään, mitä tietoturvan näkökulmia ja ratkaisuja eri vaiheissa tulisi huomioida.

Tietoturvallisen ohjelmistokehityksen prosessista käytetään lyhennettä **SSDLC (Secure Software Development Lifecycle)**. SSDLC varmistaa, että tietoturvatoimenpiteet sisällytetään kaikkiin ohjelmistokehityksen elinkaaren vaiheisiin. Tämä auttaa tunnistamaan tietoturvariskit ja korjaamaan haavoittuvuudet mahdollisimman aikaisessa vaiheessa, mikä vähentää myöhemmissä vaiheissa tarvittavia korjaustöitä, viivästyksiä ja lisäkustannuksia. Näin ohjelmistojen tietoturva paranee ja kehitys etenee suunnitelmallisesti.

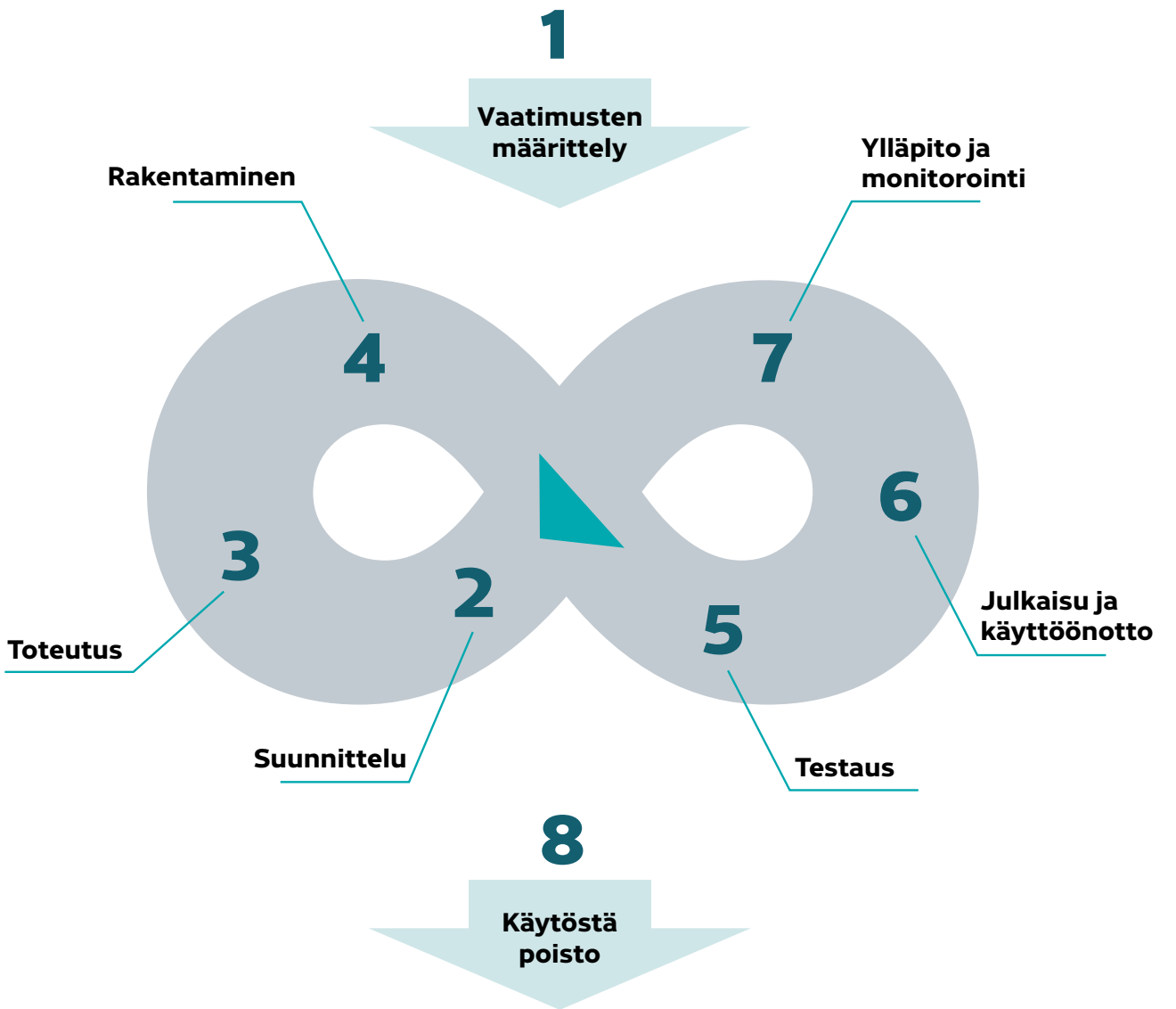
Moderni ohjelmistokehitys on iteratiivinen, jatkuva prosessi, joka ei pääty, kun ohjelmisto julkaistaan. SSDLC-prosessin merkittävimmät edut ohjelmistokehityksessä ovat seuraavat:

1. Haavoittuvuuksien minimoiminen
2. Tunnistamattomien ja mitigoimattomien haavoittuvuuksien hyväksikäyttömahdollisuuden minimoiminen
3. Haavoittuvuuksia aiheuttavien tekijöiden huomioiminen ja niihin puuttuminen, jotta estetään haavoittuvuuksien uusiutuminen

Tietoturva on tehokkainta silloin, kun se rakennetaan mukaan alusta alkaen.

Ohjelmistokehityksen elinkaari

Ohjelmistokehityksen elinkaari koostuu useista vaiheista, joista jokaisella on oma tärkeä roolinsa ohjelmiston turvallisuuden, laadun ja toimivuuden varmistamisessa. Kussakin vaiheessa arvioidaan, miten ohjelmisto täyttää sille asetetut vaatimukset ja mitä tietoturvan sekä tietosuojan näkökulmia on otettava huomioon, jotta ratkaisu on turvallinen koko elinkaarensa ajan. Ohjelmistokehitys on jatkuva ja toistuva prosessi, jossa tietoturva kulkee mukana suunnittelusta toteutukseen, ylläpitoon ja käytöstä poistoon asti, ja toimii yhtenä keskeisenä perusteena sekä suunnitteluratkaisujen arvioinnille että niiden toteutukselle. Näin varmistetaan, että ohjelmiston turvallisuus rakentuu vaiheittain ja pysyy mukana muutoksissa myös myöhemmissä elinkaaren vaiheissa.



ELINKAAREN VAIHE 1: VAATIMUSTEN MÄÄRITTELY

Vaatimusten määrittely on ohjelmistokehityksen ensimmäinen elinkaaren vaihe. Sen tavoitteena on tunnistaa, mitä ohjelmiston tulee pystyä tekemään, mitä tietoa siinä käsitellään ja missä käyttöympäristöissä sitä tullaan käyttämään.

Toiminnallisten vaatimusten määrittämisen jälkeen kartoitetaan ohjelmistoon kohdistuvat ulkoiset vaatimukset sekä kolmansien osapuolten vaatimukset. Ulkoiset vaatimukset voivat perustua esimerkiksi sääntelyyn ja standardeihin. Kolmansien osapuolten vaatimukset taas liittyvät sopimuksiin, palveluntarjoajien ehtoihin tai asiakkaiden erityisvaatimuksiin. Näiden tunnistaminen on olennaista, jotta ohjelmisto täyttää kaikki sille asetetut vaatimukset, mukaan lukien käyttäjäorganisaation ja tietojen käsittelyn edellytykset.

Ulkoisten vaatimusten lisäksi tulee huomioida myös organisaation ja ohjelmiston käyttöönottavan tahon sisäiset tietoturva- ja tietojenkäsittelyperiaatteet ja ohjeistukset, jotka ohjaavat ohjelmiston suunnittelua ja toiminnallisuuksia.

Kun vaatimukset on tunnistettu, ohjelmistolle määritetään tietoturvan vähimmäistaso. Se kuvaa ne perusvaatimukset, jotka ohjelmiston on täytettävä, jotta seuraava elinkaaren vaihe – suunnittelu – voidaan toteuttaa myös tietoturvan näkökulmasta.

Vähimmäistason määrittämisessä voidaan hyödyntää yleisesti käytössä olevia viitekehyksiä. Esimerkkeinä **OWASP-projektin** (Open Worldwide Application Security Project) Application Security Verification Standard (ASVS) sekä National Institute of Standards and Technologyn Secure Software Development Framework (NIST SSDF) viitekehykset.

Turvallisuus alkaa siitä, mitä vaaditaan.

ELINKAAREN VAIHE 2: SUUNNITTELU

Suunnitteluvaiheessa määritellään ohjelmiston teknologiaratkaisut vaatimusten ja ohjelmiston operatiivisten tarpeiden perusteella. Teknologioiden valinnassa on tärkeää varmistaa, että ne ovat nykyaikaisia, laajalti käytössä ja aktiivisesti ylläpidettyjä. Näin varmistetaan ratkaisun pitkäaikainen ylläpidettävyys, suorituskyky ja tietoturvapäivitysten saatavuus myös tulevaisuudessa.

Suunnittelun lähtökohtana tulee olla **Secure by Design** -ajattelu, jossa tietoturva rakennetaan osaksi arkkitehtuuria jo suunnitteluvaiheessa. Tavoitteena on varmistaa, että ohjelmisto ja sen käyttäjät ovat suojassa haitallisilta toimijoilta ja tietoturvauhkilta. Suunnittelussa tulee esimerkiksi huomioida salasanakäytännöt, tiedonsiirron ja -säilytyksen suojaus, käyttöoikeuksien rajaukset sekä tarpeettomien ominaisuuksien ja palveluiden poistaminen käytöstä.

Suunnitteluvaiheen keskeinen työkalu on uhkamallinnus, jonka avulla tunnistetaan ohjelmiston tietoturvaa uhkaavat tekijät, kuten haavoittuvat komponentit, riskialttiit integraatiot ja kriittiset tietovirrat. Uhkamallinnus auttaa tunnistamaan suojattavat kohteet, kuten järjestelmät, verkot, käyttäjät ja tietovarannot, sekä suunnittelemaan niille tarkoituksenmukaiset suojamekanismit.

Hyvin toteutettu suunnitteluvaihe luo perustan turvalliselle ja helposti ylläpidettävälle ohjelmistolle sekä vähentää myöhempien korjausten tarvetta.

Uhkamallinnus paljastaa riskit, suunnittelu poistaa ne.

ELINKAAREN VAIHE 3: TOTEUTUS

Toteutusvaiheessa kehitetään ohjelmiston lähdekoodi ja yhdistetään tarvittavat valmiit ohjelmistokomponentit. Nykyaikaisessa ohjelmistokehityksessä yhdistyvät itsetuotettu koodi sekä kolmansien osapuolten avoimen lähdekoodin kirjastot ja palvelut, joista muodostuu ohjelmiston kokonaisuus.

Lähdekoodi muodostaa ohjelmiston rakennuspalikat, joista koostuu käyttäjälle näkyvä ratkaisu. Hyvä koodi on luettavaa, toimivaa ja turvallista. Puutteet näissä ominaisuuksissa voivat heikentää ohjelmiston ylläpidettävyyttä, suorituskykyä ja turvallisuutta – niin käyttäjien, datan kuin infrastruktuurinkin osalta.
Tietoturva tulee huomioida jokaisessa kehitysvaiheessa. Sekä itsetuotettuun että kolmansien osapuolten komponentteihin voi liittyä haavoittuvuuksia, kuten logiikka- ja konfiguraatiovirheitä, puutteellista syötteen validointia tai vahingossa lähdekoodiin jääneitä salaisuuksia. Tällaisia voivat olla esimerkiksi kovakoodatut käyttäjätunnukset, salasanat tai API-avaimet.

Toteutusvaiheen tietoturvariskejä voidaan hallita **staattisilla tietoturvaskannereilla (Static Application Security Testing, SAST)**, jotka analysoivat lähdekoodia ja tunnistavat tunnettuja haavoittuvuuksia jo ennen ohjelman suorittamista. Lisäksi on suositeltavaa käyttää **komponenttianalyysityökaluja (Software Composition Analysis, SCA)**, jotka tunnistavat kolmansien osapuolten kirjastojen ja riippuvuuksien haavoittuvuudet.

Tietoturvallinen kehitystapa sisältää myös koodikatselmoinnit, pariohjelmoinnin, automaattiset testit ja **CI/CD-putken** valvonnan. Näiden avulla mahdolliset virheet ja riskit havaitaan mahdollisimman varhaisessa vaiheessa.

Turvallinen koodi on yhtä tärkeää kuin toimiva koodi.

ELINKAAREN VAIHE 4: RAKENTAMINEN

Rakentamisvaiheessa lähdekoodi käännetään ja rakennetaan toimivaksi ohjelmistoksi. Kääntäminen ja kokoaminen tulee tehdä turvallisessa ja eriytetyssä ympäristössä, jossa varmistetaan, että ohjelmiston käyttämät riippuvuudet ja kirjastot ovat luotettavia, ajantasaisia ja peräisin varmennetuista lähteistä.

Riippuvuuksien hallintaa varten rakentamisvaiheessa on suositeltavaa tuottaa **Software Bill of Materials (SBOM)**, joka on luettelo kaikista ohjelmiston käyttämistä komponenteista ja kirjastoista. SBOM mahdollistaa komponenttihaavoittuvuuksien seurannan, jäljitettävyyden ja korjausten hallinnan koko ohjelmiston elinkaaren ajan.

Rakentamisvaiheen keskeinen tavoite on prosessin toistettavuus ja vakaus. Riippuvuudet ja paketit haetaan aina samoista lähteistä samoilla menetelmillä, ja laadunvarmistustestit suoritetaan yhdenmukaisesti jokaisella ajokerralla. Optimitilanteessa rakentaminen tapahtuu automaattisessa CI/CD-putkessa, jossa on sisäänrakennettuja tietoturvatarkastuksia. Jos jokin tarkastuksista epäonnistuu, julkaisu estetään, kunnes havaitut tietoturvaongelmat on korjattu tai todettu merkityksettömiksi rakennettavan ohjelmiston kannalta.

Rakentamisvaiheen tietoturvauhkia voidaan hallita käyttämällä tietoturvaskannereita, jotka havaitsevat haavoittuvuuksia sekä lähdekoodissa että **infrastruktuurikoodissa (Infrastructure as Code, IaC scanner)** ja **salaisuusskannereita (secret scanner)**, jotka tunnistavat konfiguraatiotiedostoihin jääneet salaisuudet.

Lisäksi on suositeltavaa allekirjoittaa sovelluksen rakentamisessa käytetyt paketit ja tallentaa niiden tarkistussummat. Näin ohjelmiston eheys voidaan varmentaa myöhemmissä vaiheissa. Turvallinen ja toistettava rakentamisprosessi muodostaa perustan luotettavalle julkaisu- ja käyttöönottoympäristölle.

Turvallinen ja toistettava rakentamisprosessi on perusta turvalliselle ohjelmistolle.

ELINKAAREN VAIHE 5: TESTAUS

Testausvaiheen tarkoituksena on varmistaa, että ohjelmisto toimii suunnitellusti ja täyttää sille asetetut laatu- ja tietoturvavaatimukset. Ohjelmiston tulee toimia luotettavasti erilaisissa käyttötilanteissa, kuormituksissa ja ympäristöissä.

Testausvaiheessa varmistetaan, että toteutus vastaa vaatimustenmäärittelyvaiheessa kuvattuja tietoturva- ja toiminnallisia vaatimuksia. Samalla varmistetaan, että arkkitehtuuri ja tietoturvakontrollit toimivat käytännössä. Tämä tehdään sekä automatisoiduilla että manuaalisilla testeillä, kuten toiminnallisilla testitapauksilla, kuormitustestauksilla ja tietoturvatestauksilla.

Tietoturvatestauksessa voidaan hyödyntää useita menetelmiä. Staattisen analyysin (SAST) rinnalla käytetään **dynaamisia tietoturvaskannereita (Dynamic Application Security Testing, DAST)**, jotka tunnistavat ajonaikaisia haavoittuvuuksia, kuten syötteiden validointivirheitä tai puutteellisia autentikointimekanismeja. Yhä useammin käytetään myös **interaktiivista testausta (IAST)**, joka yhdistää staattisen ja dynaamisen testauksen vahvuudet.

Automaattiset testaukset muodostavat ohjelmistolle tietoturvan perustason, jota vasten myöhemmät testaukset suoritetaan. Tämä auttaa tunnistamaan uudet tietoturvaan ja ohjelmiston toimintaan vaikuttavat muutokset sekä vähentämään väärien positiivisten määrää, joissa testaus havaitsee virheen tai haavoittuvuuden virheellisesti.

Testausvaiheeseen voi sisältyä myös **tunkeutumis- ja fuzz-testausta**, joiden avulla pyritään löytämään sellaisia heikkouksia, joita automatisoidut menetelmät eivät havaitse. Lisäksi testidatan käsittelyssä on huomioitava tietosuoja: testauksessa ei tule käyttää tuotantodataa, vaan anonymisoitua tai synteettistä, eli keksittyä, aineistoa.

Huolellinen testausvaihe varmistaa ohjelmiston turvallisuuden jo ennen sen julkaisua ja luo luotettavan perustan jatkuvalle laadunvarmistukselle.

Kun testaus on kattavaa, turvallisuus ei jää oletusten varaan.

ELINKAAREN VAIHE 6: JULKAISU JA KÄYTTÖÖNOTTO

Julkaisu- ja käyttöönottovaiheessa testattu ohjelmisto julkaistaan käyttäjille ja otetaan käyttöön tuotantoympäristössä. Vaiheen tavoitteena on varmistaa, että julkaisu tapahtuu hallitusti, tietoturvallisesti ja dokumentoitujen prosessien mukaisesti.

Julkaisuvaiheen hallitun toteuttamisen keskeinen tavoite on varmistaa ohjelmiston tietoturva ja eheys. Julkaisu suoritetaan käyttämällä varmennettuja ja allekirjoitettuja artefakteja dokumentoitujen ja automatisoitujen prosessien mukaisesti. Julkaisun toteutuksesta tulisi vastata eri henkilön tai tiimin kuin ohjelmiston rakentamisesta, jotta vastuut jakautuvat selkeästi ja tahalliset väärinkäytökset voidaan estää. Tässä vaiheessa yhtenä tietoturvakontrollina toimii julkaisussa käytettävien konttikuvien tietoturvaskannaus **konttiskannerilla (Container Scanning)**.

Ennen käyttöönottoa viimeistellään tuotantoympäristön konfiguraatiot, kuten tietokannat, rajapinnat ja infrastruktuuri. Tässä yhteydessä varmistetaan, että tietoturvakontrollit, kuten pääsynhallinta, verkkoeristykset ja salattujen yhteyksien asetukset, on otettu käyttöön ja testattu. Lisäksi on tärkeää varmistaa, että salaisuudet ja pääsytunnukset hallitaan turvallisesti, eivätkä testitunnukset tai kovakoodatut avaimet päädy tuotantoon.

Rollback- tai palautussuunnitelman tulisi olla dokumentoituna ennen julkaisua. Niiden avulla järjestelmä voidaan nopeasti palauttaa aiempaan versioon, jos käyttöönotossa havaitaan virheitä tai tietoturvaongelmia.

Käyttöönoton jälkeen suoritetaan tuotantovalidointi ja seuranta, joilla varmistetaan järjestelmän vakaus ja tietoturva. Tämän jälkeen ohjelmisto siirtyy ylläpitovaiheeseen, jossa sen toimintaa seurataan jatkuvasti, havaittuja haavoittuvuuksia korjataan ja ohjelmistoa kehitetään edelleen.

Turvallinen julkaisu on hallittu tapahtuma, ei tekninen rutiini.

ELINKAAREN VAIHE 7: YLLÄPITO JA MONITOROINTI

Ylläpitovaiheessa varmistetaan ohjelmiston jatkuva toiminta, suorituskyky ja tietoturva käyttöönoton jälkeen. Ohjelmistoa kehitetään ja päivitetään sekä toiminnallisten tarpeiden että tietoturvavaatimusten mukaisesti, jotta se säilyy luotettavana muuttuvassa toimintaympäristössä.

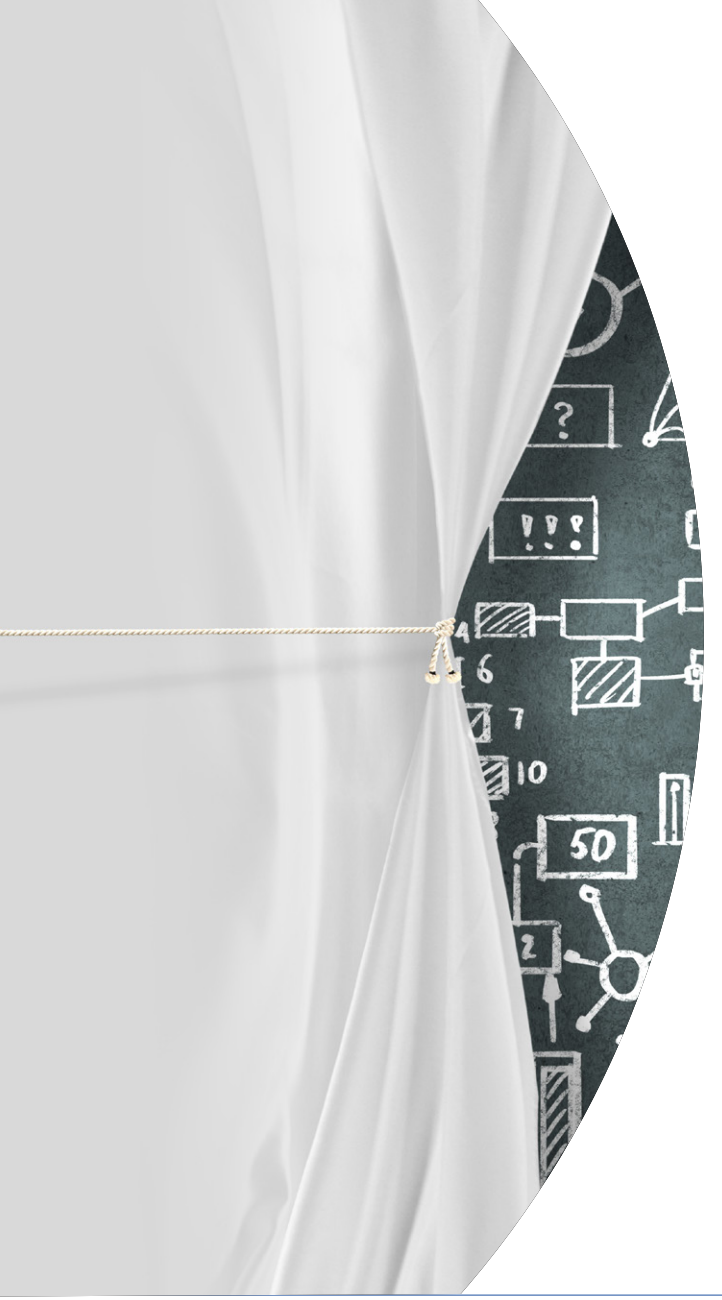
Ennen ylläpitoon siirtymistä on tärkeää varmistaa, että ylläpitoprosessit ja vastuut on määritetty, dokumentoitu ja siirretty kehityksestä tuotantoon. Tämä mahdollistaa muutosten, korjausten ja päivitysten hallitun ja jäljitettävän toteuttamisen koko ohjelmiston elinkaaren ajan.

Ylläpitovaiheen keskeisiä tehtäviä ovat haavoittuvuuksien hallinta, ohjelmistokomponenttien ja riippuvuuksien säännöllinen päivittäminen sekä salaisuuksien ja käyttöoikeuksien tarkistaminen. Ylläpitovaiheessa on kyettävä reagoimaan nopeasti havaittuihin virheisiin ja uhkiin sekä toteuttamaan tarvittavat korjaustoimenpiteet turvallisesti.

Monitorointi on osa ylläpitovaihetta ja tarkoittaa ohjelmiston toiminnan, suorituskyvyn ja tietoturvan jatkuvaa seurantaa. Seuranta perustuu lokien, mittareiden, hälytysten ja tietoturvaohjelmistojen hyödyntämiseen, ja sen tavoitteena on havaita poikkeamat, kuten suorituskyvyn heikkeneminen tai luvattomat pääsyt, mahdollisimman varhaisessa vaiheessa.

Monitoroinnin onnistuminen edellyttää selkeitä prosesseja ja vastuita sekä aktiivista reagointia havaittuihin poikkeamiin. Hälytysten käsittelyä tukevat dokumentoidut toimintamallit (runbookit). Tehokas monitorointi tarjoaa arvokasta tietoa ohjelmiston tilasta sekä tukee sen jatkuvaa kehittämistä ja tietoturvan ylläpitämistä.

Ylläpito suojaa ohjelmiston, monitorointi paljastaa poikkeamat ajoissa.



Turvallinen käytöstä poisto on osa vastuullista kehitystä.

ELINKAAREN VAIHE 8: KÄYTÖSTÄ POISTO

Ohjelmiston käytöstä poisto on ohjelmiston elinkaaren viimeinen vaihe, jossa ohjelmiston kehitys ja ylläpito päättyvät ja järjestelmä ajetaan hallitusti alas. Kuten kaikissa ohjelmiston elinkaaren vaiheissa, tulee ohjelmiston käytöstä poistossa varmistua, ettei se aiheuta tietoturvariskejä tai häiriöitä muille toteutuksille.

Ohjelmiston käytöstä poisto on toteutettava turvallisesti ja hallitusti, jotta ohjelmiston käyttöympäristöön ja käyttäjiin kohdistuvat uhat voidaan hallita. Prosessin alkuvaiheessa on suositeltavaa toteuttaa uhkamallinnus, jonka avulla tunnistetaan alasajoon liittyvät riskit ja varaudutaan niihin etukäteen.

Käytöstä poiston aikataulu tulee suunnitella ja viestiä eri sidosryhmille – kehittäjille, ylläpidolle ja käyttäjille. Puutteellinen viestintä eri sidosryhmien välillä voi johtaa siihen, että ohjelmistoa ei poisteta asianmukaisesti, ja sen seurauksena ohjelmistoon tai sen osiin voi jäädä pääsy. Tämä altistaa ohjelmiston haavoittuvuuksien hyödyntämiselle.

Käytöstä poiston yhteydessä tehdään joukko tietoturvallisia toimenpiteitä. Näitä ovat esimerkiksi tietojen turvallinen hävittäminen ja anonymisointi, käyttöoikeuksien poistaminen, palvelinten ja rajapintojen alasajo sekä sertifikaattien, domainien ja lisenssien mitätöinti. Lisäksi käytöstä poiston eteneminen dokumentoidaan ja lopputulos hyväksytään, jotta voidaan varmistaa, että kaikki suunnitellut vaiheet on suoritettu asianmukaisesti ja ohjelmisto on poistettu käytöstä turvallisesti.

Lopuksi on suositeltavaa ylläpitää lyhytaikaista jälkiseurantaa, jolla varmistetaan, ettei käytöstä poistettuun ohjelmistoon kohdistu odottamattomia yhteyksiä tai yrityksiä hyödyntää vanhoja komponentteja. Hyvin toteutettu käytöstä poisto päättää ohjelmiston elinkaaren hallitusti ja turvallisesti.

Ohjelmistoturvallisuuden osaamistarpeet rooleittain

Arkkitehti

Tässä osiossa käsitellään arkkitehdin tietoturvaan liittyviä osaamistarpeita ohjelmiston elinkaaren eri vaiheissa. Osiossa kuvataan, mitä arkkitehdin tulee huomioida ja hallita, jotta ohjelmiston turvallisuus rakentuu johdonmukaisesti alusta loppuun.

Arkkitehti rakentaa turvallisuuden perustaksi, ei irrallisiksi ratkaisuiksi.

MIKSI OHJELMISTOTURVALLISUUS KUULUU ARKKITEHDILLE?

Arkkitehdin vastuulla on varmistaa, että ohjelmistosta muodostuu toimiva, turvallinen ja kestävä kokonaisuus. Ratkaisujen tulee olla linjassa liiketoiminnan tavoitteiden ja organisaation teknisten periaatteiden kanssa. Työhön sisältyy ohjelmiston rakenteen määrittely, keskeisten teknologioiden valinta sekä ratkaisujen ohjaaminen siten, että suorituskyky, laajennettavuus ja tietoturva toteutuvat. Lisäksi arkkitehti tukee kehitystiimejä koko elinkaaren ajan, arvioi vaihtoehtoja, huolehtii kokonaisuuden eheydestä ja varmistaa, että eri osat toimivat saumattomasti yhdessä.

Arkkitehdin tietoturvaosaaminen täydentää tätä kokonaisvastuuta. Tehtävä edellyttää kykyä tunnistaa tekniset ja toiminnalliset riskit, suunnitella riskit kestävät ratkaisut ja sisällyttää tietoturvaperiaatteet osaksi arkkitehtuuria. Nämä taidot ohjaavat toteutusta ja varmistavat, että ohjelmiston turvallisuus rakentuu johdonmukaisesti suunnittelusta aina käytöstä poistoon asti.

Ohjelmistoarkkitehti

Ohjelmistoarkkitehdin vastuulla on yksittäisen ohjelmiston tekninen rakenne ja keskeiset suunnitteluratkaisut. Työhön sisältyy teknisten yksityiskohtien ja arkkitehtuurin määrittely sekä niiden teknisten standardien valinta, joiden mukaan ohjelmisto rakennetaan. Lisäksi vastuulla ovat suorituskyky, skaalautuvuus ja tietoturvan huomioiminen osana kokonaisratkaisua. Ohjelmistoarkkitehti tukee kehitystiimiä teknologioiden valinnassa ja auttaa niiden integroimisessa kokonaisratkaisuun, jotta ohjelmisto toimii luotettavasti ja on helposti ylläpidettävä.

Tiivistelmä

ARKKITEHDIN OSAAMISTARPEET

- Ymmärtää organisaation liiketoimintatarpeet ja osaa sovittaa yhteen liiketoiminnan tavoitteet, kehitystiimin ratkaisut ja infrastruktuurin vaatimukset.
- Osaa toimia yhteistyössä kehittäjien, ylläpidon ja muiden sidosryhmien kanssa varmistaen, että kokonaisratkaisu on sekä turvallinen että toteuttamiskelpoinen.
- Ymmärtää, että turvallinen arkkitehtuuri rakentuu läpi koko ohjelmiston elinkaaren ja huomioi tietoturvan johdonmukaisesti kaikissa vaiheissa.
- Tunnistaa ohjelmiston rakenteisiin, tietovirtoihin, rajapintoihin ja riippuvuuksiin liittyvät riskit ja osaa kääntää ne selkeiksi ja toteutettaviksi vaatimuksiksi.
- Osaa määrittää ja soveltaa keskeiset tietoturvaperiaatteet, kuten vähimmän oikeuden periaatteen, tiedon suojauksen ja turvallisen rajapintasuunnittelun.
- Suunnittelee arkkitehtuurin siten, että hyökkäyspinta minimoidaan ja tunnettuja malleja sekä parhaita käytäntöjä hyödynnetään tarkoituksenmukaisesti.
- Ymmärtää toteutuksen vaikutukset arkkitehtuuriin ja osaa varmistaa, että secure coding -periaatteet, kehitysputket ja tekniset ratkaisut tukevat suunniteltua turvallisuustasoa.
- Osaa arvioida testauksessa havaittujen poikkeamien ja haavoittuvuuksien vaikutukset arkkitehtuuriin ja ohjata korjauksia kokonaisuuden näkökulmasta.
- Ymmärtää julkaisun, käyttöönoton, ylläpidon ja käytöstä poiston tietoturvavaatimukset ja osaa varmistaa, että ratkaisut tukevat hallittua muutosten ja elinkaaren hallintaa.

Integraatioarkkitehti
Integraatioarkkitehti suunnittelee ohjelmistointegraatiot liiketoiminnallisten tarpeiden pohjalta ja varmistaa, että järjestelmät, sovellukset ja tietokannat voivat vaihtaa tietoa luotettavasti ja keskeytyksettä. Vastuulla on myös integraatioiden rakenne ja toteutus sekä sen varmistaminen, etteivät ratkaisut vaaranna datan eheyttä. Integraatioarkkitehdin tulee huolehtia integraatioiden dokumentoinnista, jotta kokonaisuus on ymmärrettävä, ylläpidettävä ja helposti kehitettävissä.

Tietoturva-arkkitehti
Tietoturva-arkkitehti suunnittelee, toteuttaa ja valvoo sovelluksen teknologiavalintojen ja prosessien tietoturvaa läpi koko ohjelmiston elinkaaren. Rooliin kuuluu varmistaa, että tietoturvaratkaisut ovat riittävät ja toteutettu niin, etteivät ne heikennä sovelluksen suorituskykyä tai sitä pyörittävän ympäristön tehokkuutta. Lisäksi tietoturva-arkkitehti suorittaa sovellukselle tietoturvatestauksia ja varmistaa, että havaitut riskit käsitellään asianmukaisesti.

Muut arkkitehtiroolit
Ohjelmistonkehitykseen liittyy myös muita arkkitehtirooleja, kuten yritys- ja liiketoiminta-arkkitehti, joiden vastuut eivät kohdistu suoraan yksittäiseen sovellukseen. Yritysarkkitehdit määrittävät organisaatiotason periaatteet, rakenteet ja teknologialinjaukset, jotka ohjaavat tietoturvan toteuttamista yhdenmukaisesti eri järjestelmissä ja ratkaisuissa. Liiketoiminta-arkkitehti varmistaa, että tietoturvavaatimukset tukevat liiketoiminnan tavoitteita ja prosesseja sekä että turvallisuuteen liittyvät riskit ja vaikutukset ymmärretään osana kokonaisuutta.

VAATIMUSTEN MÄÄRITTELY

Vaatimusten määrittelyssä arkkitehdin vastuulla on varmistaa, että ohjelmiston toiminnallisuuksiin, suorituskykyyn, integraatioihin ja tietoturvaan liittyvät lainsäädännölliset, organisaation sisäiset sekä sopimukselliset vaatimukset tunnistetaan ja kirjataan selkeästi. Lisäksi arkkitehti jäsentää ja konkretisoi ohjelmistolle asetetut tietoturvavaatimukset teknisiksi ratkaisuiksi, kuten suojausmenetelmiksi, salausratkaisuiksi ja pääsynhallintaperiaatteiksi.

Selkeät ja kattavat vaatimukset luovat perustan turvalliselle toteutukselle.

Osaa tunnistaa uhkamallinnuksen tuottamat tietoturvavaatimukset

Uhkamallinnuksen avulla tunnistetaan ohjelmistoon kohdistuvat keskeiset uhat jo määrittelyvaiheessa. Arkkitehdin tehtävänä on jäsentää nämä uhat selkeiksi ja toteutettaviksi vaatimuksiksi. Tämä edellyttää ohjelmiston tietovirtojen, rajapintojen, käyttäjäryhmien ja ulkoisten riippuvuuksien tunnistamista sekä ymmärrystä siitä, miten erilaiset uhkavektorit voivat vaikuttaa ohjelmiston toimintaan ja tietojen käsittelyyn. Arkkitehdin tulee dokumentoida uhkamallinnuksen havainnot siten, että ne ohjaavat seuraavan vaiheen suunnitteluratkaisuja ja mahdollistavat oikeiden suojaus- ja valvontamekanismien määrittelyn. Kun vaatimukset ovat selkeät ja kattavat, voidaan suunnitteluvaiheessa

rakentaa arkkitehtuuri, joka kestää tunnistetut riskit ja tukee ohjelmiston turvallista toteutusta.

Osaa määrittää turvallisen pääsynhallinnan vaatimukset
Arkkitehdin tulee määrittelyvaiheessa tunnistaa ja dokumentoida pääsynhallintaa koskevat vaatimukset ja varmistaa, että ne perustuvat vähimmän oikeuden periaatteeseen ja “deny by default” -malliin. Tämä edellyttää käyttäjäryhmien ja roolien määrittelyä sekä sen arviointia, mihin tietoihin ja toimintoihin eri käyttäjillä on työnkuvansa perusteella todellinen tarve päästä. Vaatimuksissa tulee edellyttää, ettei käyttäjillä ole mahdollisuutta muuttaa tai korottaa omia oikeuksiaan, ja että pääsyn tarkistaminen toteutetaan ohjelmiston kaikilla tasoilla – käyttöliittymässä, taustapalveluissa ja rajapinnoissa – yhdenmukaisin kontrollein. Lisäksi tulee huomioida vaatimukset pääsyoikeuksien poistolle, jotta oikeudet voidaan estää tai sulkea heti, kun tarve päättyy. Näiden vaatimusten selkeyttäminen luo pohjan suunnitteluvaiheessa rakennettavalle turvalliselle ja johdonmukaiselle pääsynhallinnalle.

Ymmärtää tiedon suojaamisen periaatteet ja niihin liittyvät vaatimukset
Määrittelyvaiheessa arkkitehdin tulee tunnistaa ohjelmistossa käsiteltävät tiedot ja niihin liittyvät lainsäädäntö-, sopimus- ja organisaatiokohtaiset velvoitteet. Tietojen luokittelu ja suojaustarpeiden määrittely muodostavat perustan niille kontrollitasoille, joita suunnitteluvaiheessa tarvitaan. Lisäksi arkkitehdin on määriteltävä vaatimukset turvallisille salausmekanismeille, salausavainten hallinnalle sekä tiedon ja liikenteen suojaamiselle sekä siirron että säilytyksen aikana. Näiden vaatimusten dokumentointi varmistaa, että tietojen suojaamiseen liittyvät tarpeet voidaan toteuttaa johdonmukaisesti myöhemmissä elinkaaren vaiheissa.

Osaa määrittää arkkitehtuurin ja rajapintojen suojausvaatimukset
Turvallinen arkkitehtuuri edellyttää, että suojaustarpeet ja rajapintojen käsittelemä tieto tunnistetaan ja dokumentoidaan heti elinkaaren alussa ennen suunnittelua. Tämä vaatii ymmärrystä hiekkalaatikoiden, kapseloinnin, konttien ja verkkojen eriyttämisen vaikutuksista hyökkäyspintojen hallintaan. Lisäksi on tunnistettava, mitä tietoja rajapinnat palauttavat ja miten ne on suojattava. Näiden vaatimusten määrittely luo perustan turvallisen arkkitehtuurin suunnittelulle.

Ymmärtää lähdekoodin suojauksen ja riippuvuuksien hallinnan vaatimukset
Lähdekoodin ja ohjelmistossa käytettävien komponenttien turvallisuus edellyttää, että niihin liittyvät vaatimukset tunnistetaan ja dokumentoidaan jo määrittelyvaiheessa. Arkkitehdin tulee määritellä periaatteet, joiden mukaisesti riippuvuudet valitaan luotettavista ja aktiivisesti ylläpidetyistä lähteistä, sekä varmistaa, että komponenttien käyttö kuvataan läpinäkyvästi osana suunnittelua ja dokumentaatiota. Vaatimuksiin tulee sisällyttää riskialttiiden komponenttien ja vaarallisten toiminnallisuuksien tunnistaminen sekä haavoittuvuuksien hallinnan periaatteet, kuten löydösten priorisointi ja korjaustarpeiden käsittely. Näiden pohjalta voidaan suunnitteluvaiheessa rakentaa ratkaisu, joka tukee turvallista riippuvuuksienhallintaa ja koodin suojausta koko elinkaaren ajan.

Ymmärtää infrastruktuurin ja toimintaympäristön suojausvaatimukset
Infrastruktuurin ja toimintaympäristön suojaaminen alkaa siitä, että ohjelmistolle asetetut tietoturvavaatimukset tunnistetaan ja käännetään ympäristöä koskeviksi vaatimuksiksi jo määrittelyvaiheessa. Arkkitehdin tulee tunnistaa, mitä omaisuuksia ja järjestelmän osia vaatimukset koskevat, mitä riskejä niihin liittyy ja millaista suojaustasoa ne edellyttävät. Näiden vaatimusten dokumentointi luo perustan suunnitteluvaiheen suojausratkaisuille, jotka minimoivat hyökkäyspintoja ja tukevat ohjelmiston turvallista toimintaa koko elinkaaren ajan.

Ymmärtää valvonnan ja reagoinnin vaatimukset arkkitehtuurille

Valvonnan ja reagoinnin toteuttaminen edellyttää, että arkkitehtuuri tukee vaadittujen lokitietojen keräämistä, säilytystä ja suojaamista. Arkkitehtuurin tulee mahdollistaa tietoturvalokit, virhelokit ja suorituskykylokit niin, että ne voidaan toteuttaa lainsäädännön ja tietoturvaperiaatteiden mukaisesti. Tämä edellyttää ymmärrystä siitä, millaiset lokit voivat sisältää sensitiivistä tietoa ja miten ne on suojattava, sekä varmistusta siitä, ettei lokitus paljasta ohjelmiston sisäisiä toimintoja tavalla, joka heikentäisi turvallisuutta. Rakenteiden on tuettava tehokasta valvontaa ja reagointia ilman, että ne muodostavat uusia riskejä ohjelmistolle tai tiedolle.

Osaa huomioida elinkaarenhallinnan ja käytöstä poiston vaatimukset jo määrittelyvaiheessa

Elinkaarenhallinta edellyttää, että ohjelmiston suunnittelussa varaudutaan myös sen käytöstä poistoon. Arkkitehtuurissa on tunnistettava toiminnallisuudet, teknologiat ja tietotyypit, joilla on vaikutusta siihen, miten ohjelmisto voidaan aikanaan purkaa, siirtää tai korvata. Ratkaisujen tulee tukea järjestelmän siirrettävyyttä toiseen ympäristöön ja mahdollistaa tietojen turvallinen poistaminen tai siirtäminen ilman kohtuutonta teknistä velkaa. Ennakoiva suunnittelu varmistaa, että ohjelmiston elinkaaren viimeinen vaihe voidaan toteuttaa hallitusti ja turvallisesti.

Keskeiset toimet

- Jäsennä uhkamallinnuksessa tunnistetut tietoturvavaatimukset toteutettaviksi vaatimuksiksi
- Määrittele turvallisen pääsynhallinnan vaatimukset
- Tunnista ohjelmistossa käsiteltävät tiedot ja niihin liittyvät suojausvelvoitteet
- Määritä arkkitehtuurin, rajapintojen ja lähdekoodin suojaustarpeet sekä riippuvuuksien hallinnan vaatimukset
- Käännä ohjelmistolle asetetut tietoturvavaatimukset infrastruktuurin vaatimuksiksi
- Huomioi valvonnan ja reagoinnin sekä käytöstä poiston vaatimukset jo määrittelyvaiheessa

Arkkitehti muuttaa riskit ja velvoitteet toteuttaviksi ratkaisuiksi.

SUUNNITTELU

Suunnitteluvaiheessa arkkitehti vastaa ohjelmiston rakenteen ja integraatioiden suunnittelusta siten, että ne täyttävät tunnistetut vaatimukset, vastaavat sovittuja standardeja ja hyviä tietoturvakäytänteitä. Lisäksi määritellään, mitä suojataan, millä menetelmillä ja missä ympäristössä. Arkkitehti osallistuu uhkamallinnukseen, arvioi tarvittavat turvallisuusnäkökulmat ja tekniset ratkaisut sekä varmistaa, että suunnitellut ratkaisut tukevat turvallista toteutusta.

Osaa tukea uhkamallinnusta arkkitehtuurin näkökulmasta

Uhkamallinnus edellyttää kykyä hahmottaa ohjelmiston rakenteet ja tunnistaa, mitä osia tulee suojata. Arkkitehdin on kyettävä tunnistamaan uhkamallinnuksessa suojattavat kohteet, kuten tiedot, infrastruktuurin osat, teknologiat, komponentit, konfiguraatiot ja lähdekoodi. Lisäksi on tärkeää osata tuoda esiin arkkitehtuurin ratkaisut ja riippuvuudet, joilla on vaikutusta uhkien toteutumiseen ja niiden hallintaan. Arkkitehtuurin kokonaiskuvan selkeä ymmärrys varmistaa, että uhkamallinnus perustuu realistisiin oletuksiin ja että riskeihin voidaan vaikuttaa jo suunnitteluvaiheessa.

Ymmärtää riskianalyysin periaatteet ja osaa tukea analyysin tekemistä
Riskianalyysi perustuu uhkamallinnuksessa tunnistettuihin riskeihin. Arkkitehdin tehtävänä on tukea analyysia tarjoamalla yhteinen ymmärrys suunnitelluista tietoturvakontrolleista. Tämä edellyttää kykyä selittää, miten arkkitehtuuriin sisältyvät ratkaisut vähentävät riskien todennäköisyyttä ja pienentävät niiden vaikutusta. Arkkitehtuurin näkökulma varmistaa, että riskianalyysi pohjautuu realistiseen käsitykseen järjestelmän rakenteesta ja teknisistä suojausmahdollisuuksista.

Osaa suunnitella turvallisen arkkitehtuurin ja soveltaa tunnettuja malleja
Turvallisen arkkitehtuurin suunnittelu edellyttää kykyä tunnistaa ohjelmiston rakenteista ne kohdat, joissa tietoturva on erityisen kriittinen. Arkkitehtuurin tulee perustua ratkaisuihin, jotka hyödyntävät tunnettuja ja toimiviksi todettuja malleja, ja suunnittelussa on huomioitava niiden tarjoamat periaatteet ja kontrollit. On tärkeää, että valitut arkkitehtuurimallit toteutetaan hyvien käytäntöjen mukaisesti, jotta rakenteet muodostavat kestävän ja turvallisen perustan ohjelmiston myöhemmille kehitys- ja käyttövaiheille.

Osaa suunnitella ohjelmistoon soveltuvan ja turvallisen käyttöoikeusmallin
Käyttöoikeuksien suunnittelu edellyttää ymmärrystä eri mallien, kuten RBAC:n ja ABAC:n, eroista ja soveltuvuudesta ohjelmiston tarpeisiin. Mallin tulee varmistaa käyttö- ja pääsyoikeuksien tarkistus ohjelmiston kaikilla eri tasoilla ja hyödyntää mekanismeja, joita voidaan käyttää johdonmukaisesti eri tilanteissa autorisointivirheiden välttämiseksi. Turvallinen käyttöoikeusmalli perustuu oletukseen, että pääsy on estetty kaikkialle, ellei oikeutta ole erikseen myönnetty. Tämä varmistaa, että pääsyoikeudet myönnetään resurssikohtaisesti ja vähimpien oikeuksien periaatteen mukaisesti.

Osaa suunnitella tiedon suojausperiaatteet
Suunnitteluvaiheessa arkkitehdin tulee laatia periaatteet, joilla ohjelmistossa käsiteltävä tieto suojataan sekä säilytyksen aikana että siirrettäessä. Tämä edellyttää ymmärrystä salauksen merkityksestä, käytettävissä olevista moderneista salausmenetelmistä ja siitä, miten suojaustaso valitaan suojattavan tiedon luonteen ja vaatimusten perusteella. Arkkitehdin tulee myös suunnitella malli, jossa tiedonsiirto on salattua ja jossa salausavainten säilytys ja hallinta toteutuvat turvallisesti osana ympäristön kokonaisarkkitehtuuria. Näin syntyvät suojausperiaatteet muodostavat perustan ratkaisulle, joka suojaa tietoa johdonmukaisesti ohjelmiston kaikissa käyttötilanteissa.

Osaa tunnistaa ja sisällyttää proaktiiviset tietoturvakontrollit suunnitteluun
Suunnitteluvaiheessa arkkitehdin tulee tunnistaa ne kontrollit, joilla ohjelmiston tietoturvan tasoa voidaan vahvistaa kehityksen alkuvaiheista lähtien. Tämä edellyttää ymmärrystä kriittisimmistä tietoturvaan vaikuttavista tekijöistä sekä niiden vaikutuksista suunniteltavaan ratkaisuun. Ennakoivia toimia ovat muun muassa tietoturvan huomioiminen heti ohjelmistokehitysprosessin alussa, vahva pääsynhallinta, kryptografian ja tiedon salauksen asianmukainen käyttö, syötteiden varmentaminen ja poikkeustilanteiden hallinta, turvallisten oletuskonfiguraatioiden määrittely sekä komponenttien ja digitaalisen identiteetin hallinnan turvaaminen. Lisäksi arkkitehdin on määriteltävä vaatimukset selaimen tietoturvaominaisuuksien hyödyntämiselle, valvonnalle ja lokitukselle sekä mekanismeille, joilla estetään palvelinpuolen pyyntöjen väärentäminen. Näiden kontrollien sisällyttäminen suunnitteluun luo perustan ratkaisulle, joka kestää tunnetut hyökkäystavat ja tukee turvallista toteutusta.

Keskeiset toimet

- Tunnista uhkamallinnuksessa arkkitehtuurin kriittiset kohteet ja kuvaa ne selkeästi.
- Kuvaa riskianalyysissa, miten arkkitehtuuriratkaisut pienentävät tunnistettuja riskejä.
- Hyödynnä arkkitehtuurin suunnittelussa tunnettuja ja toimiviksi todettuja malleja ja valitse niistä ratkaisut, jotka tukevat ohjelmiston turvallisuutta.
- Suunnittele käyttöoikeusmalli, joka tarkistaa pääsyoikeuden ohjelmiston kaikilla eri tasoilla ja noudattaa vähimmän oikeuden periaatetta.
- Määritä tiedon suojausperiaatteet, jotka turvaavat datan sekä siirrettäessä että säilytettäessä.
- Suunnittele ennakoivat tietoturvakontrollit, kuten pääsynhallinta, salaus, syötteiden varmentaminen ja turvalliset oletuskonfiguraatiot.

TOTEUTUS

Arkkitehti vastaa toteutusvaiheessa ohjelmistototeutuksen laadun varmistamisesta omasta näkökulmastaan. Tehtäviin kuuluu koodikatselmointeihin osallistuminen, tarvittaessa ohjelmointityöhön osallistuminen, suunnitellun arkkitehtuurin ja tietoturvavaatimusten noudattamisen varmistaminen sekä sen huolehtiminen, että koodissa toteutuvat aiemmin määritellyt tietoturvaperiaatteet.

Ymmärtää vaatimusten mukaiseen toteutukseen liittyvät periaatteet

Toteutusvaiheessa arkkitehdin tulee ymmärtää, miten vaatimuksiin perustuvat ratkaisut (kuten pääsynhallinta, tiedon suojaaminen ja salaisuuksien hallinta) toteutuvat käytännössä osana ohjelmiston rakentamista. Rooliin kuuluu myös kehittäjien tukeminen CI/CD-putken suunnittelussa ja toteutuksessa siten, että arkkitehtuuri ja putki tukevat toisiaan: arkkitehtuurin rakenteet ohjaavat putken toteutusta, ja putken ratkaisut puolestaan vaikuttavat arkkitehtuurin teknisiin valintoihin. Tämä ymmärrys varmistaa, että toteutus pysyy linjassa suunniteltujen periaatteiden kanssa eikä tuo ohjelmistoon uusia tietoturvariskejä.

Ymmärtää secure coding -käytäntöjen merkityksen toteutukselle

Arkkitehdin tulee ymmärtää secure coding -käytännöt eli turvallisen koodin kirjoittamiseen liittyvät periaatteet siinä määrin, että hän pystyy arvioimaan niiden toteutumista osana ohjelmiston rakentamista. Secure coding tarkoittaa tapoja kirjoittaa koodia niin, ettei siihen synny tarpeettomia haavoittuvuuksia ja että yleisesti tunnetut virhetyypit vältetään jo toteutuksen aikana. Tämä edellyttää osallistumista koodikatselmointeihin ja kykyä varmistaa, että lähdekoodi toteuttaa suunnitteluvaiheessa määritellyt kontrollit ja noudattaa turvallisen toteutuksen perusperiaatteita. Tämän osaamisen avulla arkkitehti tukee kehittäjiä ja varmistaa, että ohjelmiston turvallisuus rakentuu johdonmukaisesti myös kooditasolla.

Osaa tukea kehittäjiä salaisuuksien hallinnan ja arkkitehtuuriratkaisujen yhteensovittamisessa

Toteutusvaiheessa arkkitehdin tulee varmistaa, että salaisuuksien hallinta – kuten avainten, tunnisteiden ja muiden luottamuksellisten tietojen käsittely – on linjassa arkkitehtuurissa määriteltyjen periaatteiden ja suojausratkaisujen kanssa. Osaamiseen kuuluu

Arkkitehti ohjaa toteutusta kohti turvallista lopputulosta.

kehittäjien tukeminen käytännöissä, joilla salaisuudet säilytetään turvallisesti ja erotetaan ohjelmistokoodista, sekä sen varmistaminen, että valitut arkkitehtuuriratkaisut mahdollistavat turvallisen salaisuuksien hallinnan koko kehityksen ajan. Tämä yhteistyö tukee sekä tietoturvaa että arkkitehtuurin yhtenäistä toteutusta.

Keskeiset toimet

- Varmista koodikatselmoinneissa, että toteutus noudattaa suunniteltuja tietoturvakontrolleja.
- Tue kehittäjiä CI/CD-putken suunnittelussa niin, että putki ei heikennä arkkitehtuurin tietoturvaa.
- Tarkista, että pääsynhallinnan, tiedon suojaamisen ja salaisuuksien hallinnan vaatimukset toteutuvat koodissa ja putkessa.
- Ohjaa kehittäjiä soveltamaan secure coding -käytäntöjä ja korjaamaan niihin liittyvät puutteet.
- Varmista, että salaisuuksien hallinta on toteutettu arkkitehtuurin periaatteiden mukaisesti ja erillään ohjelmistokoodista.

TESTAUS

Testausvaiheessa arkkitehti arvioi testauksessa havaittujen puutteiden ja haavoittuvuuksien vaikutusta ohjelmistoarkkitehtuuriin. Arkkitehti suunnittelee ja ohjaa tarvittavat muutokset tai korjaukset ja varmistaa, että arkkitehtoniset ratkaisut tukevat ohjelmiston tietoturvaa myös testauksen jälkeen.

Osaa arvioida vaatimusten toteutumista testauksen perusteella

Testausvaiheessa arkkitehdin on tärkeää ymmärtää toteutuksen kokonaisuus siinä määrin, että pystyy arvioimaan testitulosten perusteella, täyttyvätkö suunnitteluvaiheessa asetetut arkkitehtuuriin ja tietoturvaan liittyvät vaatimukset. Tämä edellyttää kykyä tulkita testien löydöksiä arkkitehtuurin näkökulmasta, tunnistaa missä kohdin rakenteet tai kontrollit poikkeavat suunnitellusta ja arvioida, miten havainnot vaikuttavat ohjelmiston turvalliseen toimintaan. Tämän osaamisen avulla arkkitehti voi ohjata korjauksia ja varmistaa, että toteutus pysyy linjassa suunnittelun tavoitteiden kanssa.

Osaa tunnistaa löydöksistä aiemmin huomaamatta jääneet uhkat

Testausvaiheessa arkkitehdin tulee kyetä arvioimaan toteutettua arkkitehtuuria siltä osin, paljastuuko testitulosten perusteella uusia tai aiemmin huomaamatta jääneitä uhkia. Tämä edellyttää kykyä hahmottaa, miten löydökset vaikuttavat arkkitehtuurin toimivuuteen ja tietoturvaan, sekä suunnitella tarkoituksenmukaiset lieventämiskeinot riskien pienentämiseksi. Tämän osaamisen avulla arkkitehti varmistaa, että arkkitehtuuri kehittyy testauksen havaintojen pohjalta ja tukee ohjelmiston turvallista toimintaa myös jatkossa.

Ymmärtää ohjelmiston toiminnallisuudet korjausten suunnittelemiseksi

Testausvaiheessa arkkitehdin tulee ymmärtää ohjelmiston toiminnallisuudet ja niiden väliset riippuvuudet siinä laajuudessa, että hän pystyy suunnittelemaan tarkoituksenmukaiset korjaukset havaittuihin virheisiin ja poikkeamiin. Tämä edellyttää kykyä arvioida, miten löydökset vaikuttavat kokonaisuuteen, missä kohdin arkkitehtuuriin tarvitaan muutoksia ja miten korjaukset voidaan toteuttaa niin, etteivät ne aiheuta uusia riskejä tai heikennä ohjelmiston rakennetta. Tämän osaamisen avulla arkkitehti voi ohjata korjauksia tehokkaasti ja varmistaa, että ohjelmisto säilyy toimivana ja turvallisena testauksessa esiin nousseiden havaintojen jälkeen.

Testaus paljastaa, arkkitehti suunnittelee korjaukset.

Ymmärtää haavoittuvuuksien vaikutukset ja osaa suunnitella korjaukset
Testausvaiheessa arkkitehdin tulee ymmärtää ohjelmiston rakenne ja toiminnallisuudet siinä määrin, että testauksessa havaittujen haavoittuvuuksien vaikutus kokonaisuuteen voidaan arvioida. Tämä edellyttää kykyä tunnistaa järjestelmän kriittiset komponentit ja arvioida, missä kohdin arkkitehtuuriin tarvitaan muutoksia riskien pienentämiseksi. Lisäksi on osattava suunnitella korjaukset siten, etteivät ne aiheuta uusia haavoittuvuuksia tai heikennä ohjelmiston rakennetta ja turvallisuutta.

Keskeiset toimet

- Arvioi testituloksista toteutuuko arkkitehtuuri suunnitellulla tavalla ja ohjaa tarvittavat korjaukset.
- Tunnista testauksessa ilmenneet arkkitehtuuririskit ja suunnittele niille tarkoituksenmukaiset lieventämiskeinot.
- Selvitä havaittujen virheiden vaikutus ohjelmiston rakenteeseen ja suunnittele tarvittavat muutokset kokonaisuuden ehdoilla.
- Tunnista haavoittuvuuksien kohdistuminen kriittisiin komponentteihin ja määritä korjaukset, jotka poistavat riskin ilman, että arkkitehtuuri heikkenee.

Tuotantoon vain se, mitkä on tarkistettu, eheä ja hyväksytty.

JULKAISU JA KÄYTTÖÖNOTTO

Julkaisu- ja käyttöönottovaiheessa arkkitehdin vastuulla on hallitun käyttöönoton suunnittelu sekä sen varmistaminen, että ohjelmiston integraatiot, rajapinnat ja konfiguraatiot toimivat suunnitellusti tuotantoympäristössä. Lisäksi rooliin kuuluu pääsyoikeuksien määrittäminen vähimpien oikeuksien periaatteen mukaisesti sekä lokituksen, seurannan ja muiden valvontamekanismien toimivuuden varmistaminen ennen käyttöönottoa ja sen jälkeen. Arkkitehti myös osallistuu tarvittaessa käyttäjien koulutukseen ja ohjeistamiseen.

Osaa suunnitella turvallisen käyttöönottostrategian
Julkaisuvaiheessa arkkitehdin tulee suunnitella käyttöönottostrategia, joka varmistaa ohjelmiston eheyden ja estää hyväksymättömien tai haitallisten komponenttien päätymisen tuotantoon. Tämä edellyttää vaatimusten määrittämistä sille, että kaikki julkaistavat konttikuvat, paketit, konfiguraatiot ja muut osat ovat allekirjoitettuja ja että niiden eheys voidaan varmentaa ennen käyttöönottoa. Arkkitehdin tulee varmistaa, että julkaisuputki toimii suojatussa CI/CD-ympäristössä, jossa tuotantoasennuksen suorittaa valtuutettu kehitystiimin jäsen. Vaihtoehtoisesti asennus voidaan toteuttaa automatisoidusti osana CI/CD-putkea. Käyttöönottostrategiaa suunniteltaessa arkkitehdin tulee varmistaa, että järjestelmään on määritelty riittävä lokitus ja että oleelliset tapahtumat ohjautuvat valvontaratkaisuihin, jotta virheet ja poikkeamat voidaan havaita ja käsitellä hallitusti käyttöönoton jälkeen.

Osaa suunnitella tuotantoympäristön käyttöoikeushallinnan
Tuotantoympäristön käyttöoikeuksien suunnittelu edellyttää, että oikeudet perustuvat vähimmän oikeuden periaatteeseen ja rajaavat pääsyn vain siihen, mikä on välttämätöntä julkaisun toteuttamiseksi ja ylläpidon suorittamiseksi. Arkkitehdin tulee varmistaa, että julkaisun käynnistämiseen tarvittavat oikeudet on määritetty vain erikseen valtuutetuille käyttäjille ja että käyttöoikeuksien myöntämistä, muuttamista ja poistamista koskevat periaatteet tukevat turvallista ja hallittua käyttöönottoprosessia. Näin käyttöoikeushal-

linta muodostaa perustan sille, ettei tuotantoympäristö altistu tarpeettomille riskeille julkaisun yhteydessä.

Osaa tukea käyttöönottoa ja ohjata jatkokehitystä Julkaisu- ja käyttöönottovaiheessa arkkitehti tukee käyttöönottoa varmistamalla, että toteutuksessa käytetyt ratkaisut ymmärretään oikein, ja että ratkaisu integraatioineen toimii suunnitellusti. Tämä voi tarkoittaa kokonaisratkaisun esittelyä sidosryhmille, teknisten riippuvuuksien selventämistä sekä käyttöönottoon liittyvien riskien arviointia. Lisäksi arkkitehdin tehtävänä on tunnistaa julkaisun yhteydessä esiin nousevat rakenteelliset kehitystarpeet ja dokumentoida ne, jotta ne voidaan huomioida ohjelmiston jatkokehityksessä ja arkkitehtuurin kehittämisessä.

Arkkitehti pitää arkkitehtuurin eheänä myös jatkuvien muutosten keskellä.

Keskeiset toimet

- Varmista, että kaikki julkaistavat konttikuvat, paketit ja konfiguraatiot ovat allekirjoitettuja ja niiden eheys voidaan todentaa ennen tuotantoon vientiä.
- Suunnittele julkaisuputki toimimaan suojatussa CI/CD-ympäristössä, jossa tuotantojulkaisun käynnistäminen on rajattu valtuutetuille henkilöille.
- Tarkista, että tuotantoon vietävien komponenttien hash-arvot täsmäävät kehitysvaiheen arvoihin ennen käyttöönottoa.
- Varmista, että julkaisuun ja käyttöönottoon liittyvät käyttöoikeudet perustuvat vähimmän oikeuden periaatteeseen.
- Tue sidosryhmiä julkaisu- ja käyttöönottovaiheessa varmistamalla ratkaisun ymmärrettävyys ja toimivuus.
- Tunnista ja dokumentoi jatkokehitystarpeet arkkitehtuurin hallitun kehittämisen varmistamiseksi.

YLLÄPITO JA MONITOROINTI

Ohjelmiston käyttöönoton jälkeen arkkitehti osallistuu ylläpitovaiheessa ohjelmiston kehittämiseen, päivityksiin ja tietoturvaparannusten suunnitteluun. Lisäksi vastuulla on arvioida arkkitehtuurin sopeutumiskykyä muutoksiin ja varmistaa, että korjaukset ja uudet ominaisuudet eivät heikennä ohjelmiston turvallisuutta tai eheyttä.

Osaa suunnitella päivitysten ja tietoturvaparannusten hallinnan

Ylläpitovaiheessa arkkitehdin tehtävänä on varmistaa, että ohjelmiston tietoturva ja toiminnallisuus säilyvät suunnitellulla tasolla, kun ohjelmistoon tehdään päivityksiä ja muutoksia. Tätä varten arkkitehdin tulee määrittää perustaso, jonka avulla muutosten yhteydessä voidaan tunnistaa poikkeavuudet ohjelmistossa yhdessä kehitystiimin kanssa. Kun poikkeavuuksia havaitaan, arkkitehti osallistuu korjaus- ja parannustoimenpiteiden suunnitteluun ja arvioi niiden vaikutuksen julkaistuun ohjelmistoversioon. Tietoturvan osalta arkkitehti auttaa tunnistamaan havaittujen haavoittuvuuksien vaikutukset ohjelmistolle ja varmistaa, että kriittiset riskit tunnistetaan. Arkkitehdin asiantuntemus ohjelmiston rakenteista ja käytössä olevista ratkaisuista tukee kehittäjiä oikeiden toimenpiteiden valinnassa. Näin ylläpidon aikana voidaan hallita muutoksia hallitusti ja säilyttää ohjelmiston eheys ja turvallisuus.

Keskeiset toimet

- Tunnista poikkeamat ohjelmiston perustasosta ja arvioi niiden merkitys yhdessä kehitystiimin kanssa.
- Osallistu korjaus- ja parannustoimenpiteiden suunnitteluun ja arvioi muutosten vaikutukset julkaistuun versioon.
- Arvioi havaittujen haavoittuvuuksien vaikutus ohjelmistoon ja varmista, että kriittiset riskit tunnistetaan.

Arkkitehti suunnittelee järjestelmän alasajon yhtä huolellisesti kuin sen rakentamisen.

KÄYTÖSTÄ POISTO

Ohjelmiston elinkaaren päättyessä arkkitehti varmistaa, että käytöstä poisto tapahtuu hallitusti ja turvallisesti. Arkkitehdin vastuulla on suunnitella alasajoprosessi, jossa ohjelmiston rajapinnat suljetaan, tunnistetiedot poistetaan ja järjestelmän jäännöstiedot käsitellään tietoturvallisesti. Näin varmistetaan, ettei vanhan ohjelmiston kautta voida paljastaa tietoja käytössä olevista teknologioista tai ratkaisuista.

Ymmärtää alasajosuunnitelman periaatteet ja osaa varmistaa riippuvuuksien huomioimisen

Käytöstä poiston vaiheessa arkkitehti osallistuu kehitystiimin kanssa alasajosuunnitelman suunnitteluun ja varmistaa, että prosessi on hallittu ja turvallinen. Arkkitehdin tehtävänä on tunnistaa järjestelmän riippuvuudet ja arvioida niiden vaikutukset, jotta alasajo ei aiheuta häiriöitä muihin järjestelmiin tai palveluihin. Lisäksi arkkitehti huolehtii siitä, että tietoturva otetaan huomioon koko prosessin ajan, esimerkiksi datan käsittelyssä ja komponenttien poistamisessa. Näin arkkitehti tukee kehitystiimiä varmistamaan, että käytöstä poisto tapahtuu hallitusti ja ilman riskejä järjestelmän kokonaisuudelle.

Ymmärtää tietojen siirron ja säilytyksen vaatimukset käytöstä poiston jälkeen

Käytöstä poiston vaiheessa arkkitehti varmistaa, että kriittisten tietojen siirto ja arkistointi toteutetaan teknisesti turvallisesti ja säilytysratkaisut täyttävät lakisääteiset ja organisaation vaatimukset. Tämä koskee erityisesti loki- ja audit-tietoja, joiden säilytysvelvoitteet voivat olla tarkasti määriteltyjä. Arkkitehdin tehtävänä on arvioida ja ohjata teknisiä ratkaisuja niin, että tiedon eheys, luottamuksellisuus ja saatavuus säilyvät koko siirto- ja säilytysprosessin ajan. Näin arkkitehti tukee kehitystiimiä ja varmistaa, että käytöstä poisto ei aiheuta tietoturvariskejä.

Osaa arvioida rajapintojen sulkemisen vaikutukset ja hallita riskit

Käytöstä poiston vaiheessa arkkitehti varmistaa, että järjestelmän rajapinnat suljetaan hallitusti ja teknisesti turvallisesti. Auki jääneet rajapinnat voivat mahdollistaa pääsyn vanhan järjestelmän tietoihin, mikä voi sisältää esimerkiksi järjestelmä- tai käyttäjätietoja, jos niiden poistosta ei ole huolehdittu. Arkkitehdin tehtävänä on arvioida rajapintojen sulkemisen vaikutukset kokonaisarkkitehtuuriin ja ohjata teknisiä ratkaisuja niin, että tietoturvariskejä ei synny. Näin arkkitehti tukee kehitystiimiä varmistamaan, että käytöstä poisto ei jätä avoimia reittejä vanhoihin tietoihin.

Osaa tunnistetietojen ja salausavainten poistamisen periaatteet ja tekniset ratkaisut

Käytöstä poiston vaiheessa arkkitehti varmistaa, että ohjelmiston tekniset tunnistetiedot poistetaan hallitusti ja turvallisesti. Näihin voi kuulua komponentti- ja versiotiedot, konfiguraatiot sekä asennuspolut, jotka paljastuessaan voisivat antaa hyökkääjälle tietoa vanhasta tai uudesta järjestelmästä ja helpottaa hyökkäysten suunnittelua. Lisäksi arkkitehti varmistaa, että salausavaimet tuhotaan asianmukaisesti, jotta ne eivät jää järjestelmän mukana käyttöön. Arkkitehdin tehtävänä on ohjata teknisiä ratkaisuja ja tarkistaa, että poistoprosessi toteutetaan tietoturvavaatimusten mukaisesti, mikä vähentää riskiä tietojen väärinkäytöstä käytöstä poiston jälkeen.

Keskeiset toimet

- Tunnista järjestelmän riippuvuudet ja arvioi niiden vaikutukset, jotta alasajo ei aiheuta häiriöitä muihin järjestelmiin.
- Varmista, että kriittiset tiedot siirretään ja arkistoidaan turvallisesti ja säilytysratkaisut täyttävät lakisääteiset vaatimukset.
- Ohjaa rajapintojen turvallinen sulkeminen ja varmista, ettei vanhoihin tietoihin jää pääsyä.
- Tarkista, että tunnistetiedot ja salausavaimet poistetaan hallitusti ja tietoturvavaatimusten mukaisesti.

Ohjelmistokehittäjä

Tässä osiossa käsitellään ohjelmistokehittäjän tietoturvaan liittyviä osaamistarpeita ohjelmistokehityksen eri vaiheissa. Osiossa kuvataan, mitä kehittäjän on hallittava ja huomioitava, jotta ohjelmisto voidaan toteuttaa turvallisesti ja ylläpitää sen tietoturvan tasoa koko elinkaaren ajan.

MIKSI OHJELMISTOTURVALLISUUS KUULUU OHJELMISTOKEHITTÄJÄLLE?

Ohjelmistokehittäjä vastaa itsenäisesti tai yhdessä muun kehitystiimin kanssa ohjelmiston rakentamisesta, sen toiminnallisuuksien toteuttamisesta ja laadukkaasta teknisestä toteutuksesta. Tehtäviin kuuluu ohjelmiston lähdekoodin kirjoittaminen ja ylläpito, uusien ominaisuuksien toteuttaminen sekä havaittujen virheiden korjaaminen. Lisäksi on varmistettava, että ratkaisut ovat suorituskykyisiä, turvallisia ja helposti ylläpidettäviä.

Ohjelmistokehittäjän tietoturvataidot kattavat koko ohjelmistokehityksen elinkaaren vaatimusten tulkinnasta suunnitteluun, toteutukseen ja käytöstä poistoon. Ohjelmiston arkkitehtuurin ja siihen liittyvien riskien ymmärtäminen on välttämätöntä, jotta lähdekoodi, komponentit ja kehitysput-

Turvallinen ohjelmisto ei synny työkaluilla, vaan kehittäjän osaamisella.

ket voidaan toteuttaa turvallisesti. Näiden taitojen avulla varmistetaan, että ohjelmisto on suojattu niin suunnittelu-, toteutus- kuin ylläpitovaiheessakin ja reagoi muutoksiin hallitusti.

Kehitystyössä hyödynnetään sekä itse kirjoitettua koodia että valmiita ohjelmistokomponentteja ja kirjastoja, minkä vuoksi toimitusketjun turvallisuus on olennainen osa kehittäjän vastuuta. Tämä edellyttää osaamista ja ymmärrystä komponenttien luotettavuuden arvioinnista ja sen varmistamista, että käytettävät kirjastot ovat ajantasaisia ja turvallisia.

Ohjelmistokehittäjät voivat erikoistua käyttöliittymän toteuttamiseen (front end) tai palvelinpuolen ratkaisuihin (back end), mutta molemmat vastaavat omalta osaltaan siitä, että ohjelmisto toimii luotettavasti ja täyttää sille asetetut vaatimukset.

Front end -kehittäjä
Front end -kehittäjä vastaa ohjelmiston käyttöliittymän ja sen toiminnallisuuksien toteuttamisesta suunnitelmien mukaisesti ja huolehtii siitä, että ohjelmisto toimii eri alustoilla (tietokoneella, tabletilla ja mobiilissa) sekä eri käyttöjärjestelmissä, kuten Windowsissa, macOS:ssä ja Androidissa. Front end -kehittäjä tekee tiivistä yhteistyötä back end -kehittäjän kanssa varmistaakseen, että käyttöliittymä ja taustapalvelut toimivat sujuvasti yhteen ja tarjoavat käyttäjälle selkeän ja toimivan kokonaisuuden.

Back end -kehittäjä
Back end -kehittäjä vastaa ohjelmiston palvelinpuolen toimintalogiikan ja tietokantojen rakentamisesta ja ylläpitämisestä ja toteuttaa rajapinnat, joiden kautta käyttöliittymä ja taustapalvelut kommunikoivat keskenään. Back end -kehittäjän vastuulla on myös varmistaa, että tallennustila on riittävä ja toteutettu oikein, tieto on suojattu asianmukaisesti ja että palvelusta voidaan hakea tietoa tehokkaasti ja tietoturvallisesti.

TIIVISTELMÄ OSAAMISTARPEISTA

- Ymmärtää ohjelmistolle asetetut tietoturvavaatimukset sekä uhkamallinnuksessa ja riskianalyysissä tunnistetut riskit ja niiden vaikutukset toteutukseen.
- Hallitsee turvallisen ohjelmistokehityksen periaatteet, kuten secure coding -käytännöt, käyttäjän syötteiden validoinnin, virheiden turvallisen käsittelyn ja tietoturvallisen rajapintakehityksen.
- Tuntee yleisimmät ohjelmistohaavoittuvuudet, osaa välttää niitä ja seuraa niiden kehitystä, esimerkiksi OWASP Top 10 -listojen avulla.
- Ymmärtää avoimen lähdekoodin ja kolmansien osapuolten komponentteihin liittyvät riskit sekä osaa käyttää ja päivittää riippuvuuksia turvallisesti.
- Osaa hyödyntää kehityksen aikaisia tietoturvaskannereita (esim. SAST, IaC-skannaus, salaisuusskannaus) ja tulkita niiden löydöksiä kriittisesti.
- Osaa käsitellä tunnistetietoja, avaimia ja muita salaisuuksia turvallisesti, rajata niiden käyttöä roolien mukaan ja huolehtia avainten elinkaaren hallinnasta.
- Osaa toteuttaa tietoturvaa tukevia yksikkö-, integraatio- ja E2E-testejä ja hyödyntää testauksen tuloksia korjausten ja parannusten suunnittelussa.
- Ymmärtää kehitys- ja julkaisuprosessien turvallisuuden, toimitusketjuriskit sekä ohjelmiston eheyden varmentamisen ennen tuotantoon siirtoa.
- Osaa tukea tietoturvapoikkeamien käsittelyä, toteuttaa päivityksiä ja konfiguraatiokovennuksia sekä arvioida muutosten vaikutuksia ohjelmistoon.
- Ymmärtää ohjelmiston hallitun alasajon, tietojen turvallisen poiston sekä loki- ja audit-tietojen säilytysvaatimukset elinkaaren lopussa.

VAATIMUSTEN MÄÄRITTELY

Vaatimustenmäärittelyvaiheessa ohjelmistokehittäjän tulee perehtyä ohjelmistolle asetettuihin vaatimuksiin ja ymmärtää niiden vaikutus toteutukseen. On osattava tulkita sekä toiminnallisia että tietoturvavaatimuksia ja huomioida ne omassa kehitystyössään. Ohjelmistokehittäjän tulee ymmärtää ohjelmiston turvalliseen toteutukseen liittyvät menetelmät ja osallistua uhkamallinnukseen, jotta tunnistetut uhat ja riskit heijastuvat konkreettisesti suunnitteluratkaisuihin ja toteutukseen.

Ymmärtää uhkamallinnuksen tuottamat tietoturvavaatimukset

Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on osattava tunnistaa uhkamallinnuksen tuottamat riskit ja muuttaa ne selkeiksi vaatimuksiksi. Tämä edellyttää ymmärrystä ohjelmiston tietovirroista, rajapinnoista, käyttäjäryhmistä ja ulkoisista riippuvuuksista sekä siitä, miten eri uhkavektorit voivat vaikuttaa toimintaan ja tiedon käsittelyyn.

Kehittäjän tulee arvioida ohjelmistoa myös tunnettujen tietoturvahaavoittuvuuksien näkökulmasta. OWASP Top 10 -haavoittuvuudet on käytävä läpi osana uhkamallinnusta, ja niihin liittyvät heikkoudet on kirjattava niin, että ne ohjaavat jatkosuunnittelua ja estävät yleisimpien tietoturvavirheiden syntymisen.

Kun riskit on tunnistettu, kehittäjän tehtävänä on varmistaa, että havainnot dokumentoidaan ja että niiden pohjalta voidaan määrittää oikeat suojaus- ja valvontamekanismit. Selkeästi määritellyt vaatimukset luovat perustan ratkaisuille, jotka kestävät tunnistetut uhat ja tukevat ohjelmiston turvallista toteutusta koko sen elinkaaren ajan.

Ymmärtää turvallisen pääsynhallinnan vaatimukset

Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on ymmärrettävä, miten pääsynhallinta rakennetaan tavalla, joka tukee ohjelmiston turvallista toimintaa sen koko elinkaaren ajan. Kehittäjän tehtävänä on varmistaa, että ratkaisut pohjautuvat vähimpien oikeuksien periaatteeseen ja että oletusarvoisesti kaikki pääsy on estetty. Tämä edellyttää käyttäjäryhmien ja roolien, sekä sen määrittämistä mihin tietoihin tai toimintoihin niillä on perusteltu tarve päästä. Pääsyoikeuksien tarkistamisen on tapahduttava jokaisella ohjelmiston tasolla, niin käyttöliittymällä, taustapalveluissa kuin rajapintakutsuissakin. Kehittäjän tulee huolehtia siitä, että vaatimukset sisältävät määrityksen siitä, ettei käyttäjä itse voi muuttaa tai korottaa omia oikeuksiaan. Lisäksi on määriteltävä, että pääsyoikeuksien tulee poistua käytöstä heti, kun niitä ei enää tarvita.

Kun nämä vaatimukset on tunnistettu ja kuvattu selkeästi jo määrittelyvaiheessa, ne antavat suunnittelulle selkeät lähtökohdat ja vähentävät riskejä myöhemmissä kehitysvaiheissa.

Ymmärtää tiedon suojaamisen vaatimukset

Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on tunnistettava, mitä tietoja ohjelmisto käsittelee ja millaisia velvoitteita niihin liittyy, kuten esimerkiksi tietosuojavelvoitteet. Kehittäjän tulee ymmärtää tiedoille asetettu kriittisyysaste ja määrittää, millä toimenpiteillä vaatimukset pystytään täyttämään.

Kehittäjän on ymmärrettävä salausmenetelmiä siten, että jo vaatimusmäärittelyä tehtäessä varmistetaan, että ohjelmiston tiedot suojataan riittävästi sekä siirron aikana että säilytettäessä. Lisäksi ohjelmistokehittäjän on huomioitava, miten salausavaimet tallennetaan ja säilytetään turvallisesti, jotta avaintenhallinta ei aiheuta uhkaa ohjelmiston turvallisuudelle.
Kun nämä vaatimukset määritellään jo alkuvaiheessa, tiedon suojaaminen voidaan toteuttaa johdonmukaisesti koko ohjelmiston elinkaaren ajan.

Turvallinen toteutus alkaa turvallisuuden ymmärtämisestä.

Ymmärtää arkkitehtuurin ja rajapintojen suojausvaatimukset
Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on tunnistettava arkkitehtuurin ja rajapintojen keskeiset suojaustarpeet ennen suunnittelua. Tämä edellyttää ymmärrystä ympäristön eriyttämisestä, kuten hiekkalaatikoista, kapseloinnista, konttiteknologioista ja verkkojen erottelusta, jotta hyökkäyspinta pysyy hallittuna. Rajapintojen osalta kehittäjän on määriteltävä vaatimukset tietoturvalliselle toiminnalle ja huomioitava OWASP API Top 10 -haavoittuvuudet osana vaatimusmäärittelyä. Lisäksi on kuvattava, mitä tietoja rajapinnat saavat palauttaa missäkin tilanteessa, jotta varmistetaan, ettei arkaluontoinen tieto paljastu oikeudettomille käyttäjille.

Kun nämä suojaustarpeet tunnistetaan ja kuvataan selkeästi jo määrittelyvaiheessa, arkkitehtuurille ja rajapinnoille voidaan myöhemmissä vaiheissa rakentaa ratkaisut, jotka tukevat turvallista ja hallittua kokonaisuutta.

Ymmärtää lähdekoodin suojauksen ja riippuvuuksienhallinnan vaatimukset
Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on määriteltävä lähdekoodin ja käytettävien komponenttien tietoturvavaatimukset ennen toteutusta. Tämä sisältää perustelut tietoturvakontrolleille sekä vaatimukset komponenttien valinnalle luotettavista, ylläpidetyistä ja yleisesti käytetyistä lähteistä.

Kehittäjän tulee kuvata käytettävät komponentit ja tunnistaa riskialttiit osat sekä vaaralliset toiminnallisuudet, jotka voivat mahdollistaa tietoturvauhkia. Lisäksi on määriteltävä haavoittuvuuksienhallinnan periaatteet, kuten löydösten priorisointi ja korjausaikataulut.

Kun nämä vaatimukset kirjataan selkeästi jo määrittelyvaiheessa, ne tukevat turvallista kehitystä ja hallittua riippuvuuksien käyttöä koko ohjelmiston elinkaaren ajan.

Ymmärtää infrastruktuurin ja toimintaympäristön suojausvaatimukset
Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on tunnistettava, millaisia suojausvaatimuksia ohjelmisto asettaa infrastruktuurille ja toimintaympäristölle. Tämä sisältää ympäristöön tarvittavat kovennukset ohjelmiston vaatimusten ja riskien perusteella. Kehittäjän tulee myös tunnistaa suojattavat omaisuudet ja järjestelmän osat, joihin vaatimukset kohdistuvat. Kun nämä asiat kuvataan selkeästi jo määrittelyvaiheessa, ne luovat pohjan turvalliselle ja hallitulle toteutukselle myöhemmissä kehitysvaiheissa.

Ymmärtää valvonnan ja reagoinnin vaatimukset
Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on määriteltävä, mitä lokitietoja ohjelmistosta kerätään, jotta valvonta ja reagointi on mahdollista vaaditussa laajuudessa. Tämä koskee tietoturva-, virhe- ja suorituskykylokeja sekä niiden käyttötarkoitusta.

Kehittäjän on huomioitava lainsäädäntö ja varmistettava, etteivät tietoturvalokit sisällä henkilötietoja, ellei laki sitä edellytä. Virhelokit eivät saa paljastaa ohjelmiston sisäisiä ratkaisuja ja suorituskykylokien sisällön tulee olla tarkoituksenmukaista. Lisäksi on tunnistettava lokien mahdollisesti sisältämät sensitiiviset tiedot ja määriteltävä niitä koskevat suojaustoimet jo määrittelyvaiheessa. Lokit eivät saa sisältää sellaisia sensitiivisiä tietoja, jotka voivat mahdollistaa esimerkiksi luvattoman pääsyn ohjelmistoon tai sen taustapalveluihin, kuten kirjautumistiedot tai salaus- ja pääsyavaimet.

Ymmärtää elinkaarenhallinnan ja käytöstä poiston vaatimukset
Vaatimusmäärittelyvaiheessa ohjelmistokehittäjän on huomioitava, että ohjelmiston elinkaari päättyy aikanaan ja että käytöstä poisto vaatii omat toimenpiteensä. Kehittäjän tulee tunnistaa jo tässä vaiheessa, millaisia vaatimuksia ohjelmiston alasajo, korvaaminen tai poistaminen asettavat, jotta nämä eivät muodostu riskeiksi myöhemmin.

Kehittäjän on tunnistettava ohjelmistoon toteutettavat toiminnallisuudet, käytettävät teknologiat ja käsiteltävä data, jotta käytöstä poistoon liittyvät tarpeet voidaan arvioida realistisesti. Tämä sisältää esimerkiksi sen, miten data käsitellään, siirretään tai poistetaan turvallisesti, kun ohjelmisto ei ole enää käytössä.

Lisäksi kehittäjän on huomioitava järjestelmän siirrettävyys toiseen ympäristöön osana vaatimusmäärittelyä. Kun nämä näkökulmat kirjataan vaatimuksiksi jo alkuvaiheessa, ne tukevat hallittua elinkaarenhallintaa ja vähentävät riskejä ohjelmiston käytön päättyessä.

Keskeiset toimet

- Tunnista uhkamallinnuksen tuottamat riskit ja kirjaa ne selkeiksi tietoturvavaatimuksiksi.
- Käy läpi OWASP Top 10- ja OWASP API Top 10 -haavoittuvuudet ja huomioi ne vaatimuksissa.
- Määrittele turvallisen pääsynhallinnan vaatimukset vähimmän oikeuden ja deny by default -periaatteiden mukaisesti.
- Tunnista ohjelmistossa käsiteltävät tiedot, niiden kriittisyys ja tiedon suojaamiseen liittyvät vaatimukset.
- Kuvaa arkkitehtuurin ja rajapintojen suojaustarpeet, mukaan lukien rajapintojen palauttaman tiedon rajaukset.
- Määrittele lähdekoodin ja riippuvuuksien hallinnan tietoturvavaatimukset, mukaan lukien haavoittuvuuksienhallinnan periaatteet.
- Tunnista infrastruktuurille ja toimintaympäristölle kohdistuvat suojausvaatimukset ohjelmiston tarpeiden pohjalta.
- Määrittele lokituksen, valvonnan ja reagoinnin vaatimukset lainsäädäntö ja tietoturva huomioiden.
- Huomioi elinkaarenhallinnan ja käytöstä poiston vaatimukset jo määrittelyvaiheessa, mukaan lukien datan käsittely ja siirrettävyys.

SUUNNITTELU

Suunnitteluvaiheessa ohjelmistokehittäjä vastaa siitä, että toteutusvaiheeseen siirtyessä ohjelmiston tavoitteet, toiminnallisuudet ja käytettävät tekniset ratkaisut ovat selvillä. Onnistuneen suunnittelun päätteeksi kehittäjällä on tiedossa mitä ohjelmiston tulee kyetä tekemään, miten ohjelmiston sisäiset ja ulkoiset komponentit kommunikoivat ja mitä teknologioita ohjelmistossa tullaan käyttämään. Suunnittelussa otetaan huomioon ohjelmistoon kohdistuvat vaatimukset, tunnistetut riskit ja tietoturvatoimia vaativat yksityiskohdat.

Osallistuu uhkamallinnukseen

Suunnitteluvaiheessa ohjelmistokehittäjän on osallistuttava uhkamallinnukseen ja tuotava siihen tekninen näkemyksensä. Kehittäjän tehtävänä on auttaa tunnistamaan, mihin ohjelmiston osiin riskit kohdistuvat ja mitkä kohteet vaativat suojausta. Tämä edellyttää kykyä tunnistaa suojattavat tiedot, infrastruktuuri, teknologiat, komponentit, konfiguraatiot ja lähdekoodi. Kun nämä huomioidaan suunnittelussa, voidaan ratkaisuja ohjata turvallisempaan suuntaan ennen toteutusta.

Suunnittelee ohjelmistoarkkitehtuurin ja toteutuksen tietoturvallisesti

Ohjelmistokehityksen suunnitteluvaiheessa ohjelmistokehittäjän on kyettävä suunnittelemaan arkkitehtuuri ja toteutus siten, että tietoturva, ylläpidettävyys ja selkeys tukevat toisiaan. Kehittäjän tulee varmistaa, että ratkaisut ovat mahdollisimman helposti ylläpidettäviä, selkeästi toteutettuja ja dokumentoituja, eikä suojausta rakenneta monimutkaisuuden varaan. Kehittäjän on suunniteltava suojaukset tunnistettuja hyökkäysvektoreita vastaan ja huomioitava, että hyökkäys voi tulla myös ohjelmiston sisältä esimerkiksi haavoittuvan komponentin kautta. Tämän vuoksi suunnittelussa on hyödynnettävä kerroksellista suojausta eikä luotettava yhden suojakerroksen riittävyyteen. Lisäksi kehittäjän tulee hyödyntää olemassa olevia, laajasti käytettyjä ja turvallisiksi todettuja kirjastoja ja viitekehyksiä omien ratkaisujen sijaan. Jo suunnitteluvaiheessa on myös varmistettava, että oletustunnusten vaihto niin kehitys-, testi- kuin tuotantoympäristössäkin on huomioitu osana toteutusta, jotta järjestelmä ei jää alttiiksi yleisille ja vältettävissä oleville riskeille.

Keskeiset toimet

- Osallistu uhkamallinnukseen ja tunnista, mihin kohtiin järjestelmää todelliset riskit kohdistuvat.
- Hyödynnä tunnettuja ja turvallisiksi todettuja kirjastoja ja rakenna suojaus kerroksittain.
- Varmista jo suunnittelussa, että yleiset ja vältettävissä olevat heikkoudet, kuten oletustunnukset, poistuvat toteutuksessa.

Suunnittelussa kehittäjä ohjaa toteutuksen turvalliselle polulle.

Hyvä koodi on samalla toimivaa ja turvallista.

TOTEUTUS

Toteutusvaiheessa ohjelmistokehittäjän päävastuulla on turvallisen ja laadukkaan lähdekoodin tuottaminen sekä ohjelmistokomponenttien yhteen toimimisen varmistaminen. Toteutuksen tulee noudattaa suunnitteluvaiheen dokumentaatiota ja hyviä tietoturvakäytäntöjä. Ohjelmistokehittäjän tulee ymmärtää luotettavien komponenttien ja laadukkaan lähdekoodin merkitys ohjelmiston tietoturvalle, sekä tiedostaa yleisimmät ohjelmistoissa ilmenevät tietoturvahaavoittuvuudet ja osata toteuttaa ohjelmisto siten, että näiden toteutumis- ja hyväksikäyttöriski toteutettavassa ohjelmistossa on mahdollisimman pieni.

Osaa soveltaa secure coding -käytäntöjä toteutusvaiheessa

Toteutusvaiheessa ohjelmistokehittäjän tulee osata noudattaa ja dokumentoida tietoturvallisen koodauksen periaatteet, joita kehitystyössä sovelletaan. Tämä edellyttää versionhallinnan järjestelmällistä käyttöä siten, että muutokset ovat jäljitettävissä ja koodin kehityshistoria on selkeästi dokumentoitu. Kehittäjän vastuulla on varmistaa, että ohjelmistokoodi tuotetaan ilman tunnettuja ohjelmointivirheitä, jotka voisivat altistaa sovelluksen tietoturvauhkille.

Käytännössä tämä tarkoittaa esimerkiksi sitä, että käyttäjän syötteet validoidaan huolellisesti eikä ohjelmisto mahdollista haitallisen koodin suorittamista, virheellisten arvojen käsittelyä tai sellaisten tietojen palauttamista, joihin käyttäjällä ei ole oikeutta. Ohjelmiston virheviestit ja palautekoodit on toteutettava siten, etteivät ne paljasta järjestelmän teknisiä yksityiskohtia, joita voitaisiin hyödyntää haavoittuvuuksien tunnistamiseen. Lisäksi lähdekoodiin ei tule sisällyttää selkokielisiä käyttäjätunnuksia, salasanoja tai muita salaisuuksia.

Ohjelmistokehittäjän tulee osata toteuttaa keskeiset tietoturvakontrollit osana ohjelmiston logiikkaa. Tähän kuuluu käyttäjän syöttämän ja tallentaman datan validointi sekä käyttöliittymässä että taustapalveluissa. Kehitystyössä hyödynnetään sallittujen ja estettyjen syötteiden listoja, joilla estetään virheellinen tai haitallinen syöte ja sallitaan vain ennalta määritellyn muotoinen data. Esimerkiksi numeeriseen kenttään hyväksytään vain numeerinen arvo, ja tämä varmistetaan aina myös palvelinpuolella riippumatta käyttöliittymän toteutuksesta.

Ymmärtää ja hyödyntää tietoturvaskannereita kehitystyössä

Toteutusvaiheessa ohjelmistokehittäjän tulee ymmärtää erilaisten tietoturvaskannereiden käyttötarkoitukset, hyödyt ja rajoitteet sekä osata hyödyntää niitä osana kehitystyötä. Kehittäjän on hahmotettava, miten käytettävät skannerit valitaan toteutusympäristön perusteella, esimerkiksi sen mukaan, onko kyseessä pilvipohjainen vai paikallinen ympäristö, ja ajetaanko skannauksia automaattisesti kehitysputkessa vai manuaalisesti.

Kehittäjän tulee ymmärtää myös skannereiden tuottamien havaintojen luonne. Kaikki löydökset eivät ole todellisia haavoittuvuuksia, joten väärien positiivisten tunnistaminen ja havaintojen manuaalinen varmentaminen on olennainen osa osaamista. Lisäksi havaintojen merkitys voi vaihdella ympäristön mukaan, eikä automaattinen skannaus poista manuaalisen testauksen tarvetta.

Ohjelmistokehittäjän on osattava ottaa käyttöön kehityksen aikaiset tietoturvaskannerit ja tulkita niiden tuottamaa tietoa ohjelmiston turvallisuuden näkökulmasta. Tyypillisiä käytettäviä skannereita ovat staattinen koodianalyysityökalu (SAST), joka tunnistaa tunnettuja haavoittuvuuksia lähdekoodista, infrastruktuurikoodin skannaustyökalu (IaC scanner), joka koh-

distuu esimerkiksi pilvi- ja konfiguraatiomäärittelyihin, sekä salaisuusskanneri, jonka avulla voidaan havaita lähdekoodiin vahingossa päätyneet salaisuudet.

Ymmärtää OWASP Top 10 -haavoittuvuudet ja osaa huomioida ne toteutuksessa

Toteutusvaiheessa ohjelmistokehittäjän tulee ymmärtää OWASP Top 10 -haavoittuvuuksien syntymekanismit ja osata toteuttaa ratkaisuja, joilla näiltä haavoittuvuuksilta suojaudutaan tai niiden vaikutusta pienennetään kyseisessä ohjelmistossa. Tämä edellyttää ymmärrystä siitä, miten tyypillisimmät haavoittuvuudet syntyvät käytännön kehitystyössä ja miten ne voidaan estää turvallisilla toteutustavoilla. Kehittäjän on lisäksi osattava huomioida vaatimustenmäärittely- ja suunnitteluvaiheessa tunnistetut sekä uhkamallinnuksessa esiin nousseet riskit toteutuksessa. Näin varmistetaan, että ohjelmiston kannalta olennaiset haavoittuvuudet käsitellään systemaattisesti ja että tietoturva rakentuu johdonmukaisesti läpi kehitystyön.

Ymmärtää turvallisen API-kehityksen periaatteet ja OWASP Top 10 API -riskit

Toteutusvaiheessa ohjelmistokehittäjän tulee ymmärtää rajapintahaavoittuvuuksien syntymekanismit ja osata toteuttaa rajapinnat siten, että niiltä suojaudutaan tai niiden vaikutusta pienennetään kyseisessä ohjelmistossa. Tämä edellyttää OWASP Top 10 API -riskien tuntemista ja niiden huomioimista käytännön kehitystyössä. Rajapintojen toteutuksessa on huomioitava vaatimus- ja suunnitteluvaiheessa tunnistetut sekä uhkamallinnuksessa esiin nousseet tyypillisimmät rajapintariskit. Kehittäjän tulee varmistaa, että rajapinnat käsittelevät vain tarvittavia tietoja, tarkistavat pääsyn jokaisessa pyynnössä ja estävät väärinkäytöt, jotka voisivat vaarantaa ohjelmiston tietoturvan.

Osaa toteuttaa salaisuuksien turvallisen hallinnan

Toteutusvaiheessa ohjelmistokehittäjän tulee tunnistaa, mitä salaisuuksia ohjelmistoon liittyy ja miten niitä käsitellään turvallisesti. Näihin kuuluvat esimerkiksi sertifikaatit, salasanat, SSH-avaimet ja salausavaimet, joiden suojaaminen on keskeistä ohjelmiston tietoturvalle. Kehittäjän on osattava toteuttaa salaisuuksien säilytys siten, etteivät ne ole luettavissa lähdekoodista tai muutoin ulkopuolisten hyödynnettävissä. Osaamiseen kuuluu myös salaisuuksien käyttöoikeuksien rajaaminen kehitystiimin sisällä roolien mukaan, jotta kaikilla ei ole pääsyä kaikkiin salaisuuksiin. Esimerkiksi tuotantoympäristön ja julkaisun yhteydessä käytettävät salaisuudet kuuluvat vain niille, jotka vastaavat julkaisusta. Lisäksi kehittäjän on huolehdittava siitä, että salausavaimia ja muita tunnistetietoja uusitaan säännöllisesti väärinkäytösriskin pienentämiseksi.

Osaa toteuttaa tietoturvaa tukevan yksikkö-, integraatio- ja E2E-testauksen

Toteutusvaiheessa ohjelmistokehittäjän tulee osata laatia kattavat yksikkötestit, joilla varmistetaan ohjelmiston yksittäisten komponenttien oikeanlainen toiminta. Yksikkötestauksen avulla varmistetaan, että komponentit toimivat asetettujen vaatimusten ja secure coding -periaatteiden mukaisesti myös tietoturvan näkökulmasta. Testauksen tulee kattaa sekä odotetut käyttötapaukset että tilanteet, joissa virheellinen tai ei-toivottu toiminta on estettävä.

Lisäksi kehittäjän tulee osata toteuttaa automaattisia integraatio- ja end-to-end (E2E) -testejä. Näillä varmistetaan, että eri komponenttien ja rajapintojen väliset integraatiot toimivat odotetusti ja että ohjelmisto toimii kokonaisuutena ottaen huomioon kaikki sen osa-alueet. Testauksen avulla tulee pystyä varmentamaan, ettei järjestelmä mahdollista sellaisia toimintoja tai käyttötilanteita, joita ei ole tarkoitettu sallittaviksi. Kehittäjän on ymmärrettävä ohjelmiston toiminnallisuudet ja riippuvuudet riittävän hyvin, jotta testitulosten perusteella voidaan arvioida ohjelmiston turvallisuus ja toimivuus kokonaisuutena. Näin systemaattinen testaus tukee varhaista virheiden ja haavoittuvuuksien havaitsemista ja muodostaa perustan turvalliselle ja luotettavalle toteutukselle.

Keskeiset toimet

- Varmista secure coding -periaatteiden noudattaminen sekä koodin jäljitettävyys versionhallinnassa.
- Varmista käyttäjän syötteiden validointi niin käyttöliittymällä kuin taustapalveluissakin ja estä haitalliset tai virheelliset syötteet.
- Varmista, etteivät virheilmoitukset ja palautekoodit paljasta teknisiä yksityiskohtia tai arkaluonteista tietoa.
- Hyödynnä tietoturvaskannereita (SAST, IaC, salaisuusskannaus) ja arvioi löydökset kriittisesti, mukaan lukien väärät positiiviset.
- Varmista OWASP Top 10- ja OWASP API Top 10 -riskien huomioiminen toteutuksessa ja käsittele uhkamallinnuksessa tunnistetut haavoittuvuudet.
- Toteuta rajapinnat siten, että ne käsittelevät vain tarvittavaa dataa ja tarkistavat pääsyn jokaisessa pyynnössä.
- Varmista salaisuuksien hallinnointi turvallisesti erillään lähdekoodista, rajaa käyttöoikeudet roolien mukaan ja uusi avaimet säännöllisesti.
- Varmista kattavien yksikkö-, integraatio- ja E2E-testitapausten toteuttaminen, joilla varmistetaan sekä tietoturva- että toiminnallisten vaatimusten toteutuminen.

RAKENTAMINEN

Rakentamisvaiheessa ohjelmistokehittäjän tehtävänä on varmistaa, että ohjelmistokehitysputki (CI/CD-putki) konfiguroidaan ja toteutetaan tietoturvallisesti. Prosessin tulee olla mahdollisimman pitkälle automatisoitu, dokumentoitu ja toistettavissa, jotta ohjelmisto rakennetaan aina samalla tavalla ja tuottaa tarvittavat paketit. Lisäksi kehittäjän on hallittava toimitusketjuun liittyvät riskit, esimerkiksi tuottamalla SBOM-listaus osana kehitysputkea.

Osaa varmistaa kehitysputken turvallisuuden

Rakentamisvaiheessa ohjelmistokehittäjän tulee osata varmistaa, että kehitysputki on mahdollisimman pitkälle automatisoitu ja toteutettu tietoturvallisesti. Kehittäjän on osattava konfiguroida käytettävät kehitystyökalut siten, että niissä on käytössä vain tarvittavat ominaisuudet ja että ne toimivat odotetulla tavalla. Tämä edellyttää ajantasaisten ja luotettavien työkalujen käyttöä sekä työkalujen aitouden ja eheyden varmentamista. Kehittäjän tulee myös huolehtia, että oma kehitysympäristö on turvallinen ja kehittämiseen käytetään hyvien tapojen mukaisesti vain työhön tarkoitettuja laitteita ja ohjelmistoja. Lisäksi on ymmärrettävä kehitys-, testi- ja tuotantoympäristöjen erilaiset kriittisyysasteet ja varmistettava, että tuotantoversion rakentaminen tapahtuu erillisessä, siihen osoitetussa ympäristössä.

Turvallinen toimitusketju syntyy hallitussa rakentamisessa.

Osaa hyödyntää rakentamisen aikaisia tietoturvaskannereita

Ohjelmistokehittäjän tulee osata ottaa käyttöön rakentamisen aikaiset tietoturvaskannerit ja ymmärtää niiden tuottaman tiedon merkitys ohjelmiston turvallisuudelle. Tämä sisältää dynaamisen tietoturvaskannauksen (DAST), jolla tunnistetaan ajonaikaisia haavoittuvuuksia, sekä ohjelmistokomponenttianalyysin (SCA), jonka avulla tunnistetaan käytetyt avoimen lähdekoodin ja kolmansien osapuolten komponentit sekä niihin liittyvät tunnetut haavoittuvuudet. Kehittäjän on osattava tulkita skannereiden tuottamia löydöksiä ja ymmärrettävä, että automaattinen skannaus ei poista manuaalisen arvioinnin tarvetta.

Osaa varmistaa komponenttien turvallisen käytön

Rakentamisvaiheessa ohjelmistokehittäjän tulee osata varmistaa, että ohjelmistossa käytettävät kolmansien osapuolten komponentit ovat tunnettuja, luotettavia ja turvallisiksi todettuja. Kehittäjän on arvioitava komponenttien ylläpidon aktiivisuus, dokumentaation laatu,

testauksen taso, lisenssien soveltuvuus sekä tietoturvan jatkuva ylläpito, jotta ohjelmiston kokonaisuus säilyy turvallisena ja luotettavana. Kehittäjän vastuulla on huolehtia ohjelmistossa käytettävien komponenttien säännöllisestä päivittämisestä ja ymmärtää päivityksiin liittyvät riskit, kuten uusien versioiden mahdolliset ongelmat. Automaattisten päivitysten yhteydessä kehittäjän tulee huomioida riittävä viive ennen käyttöönottoa, jotta mahdolliset virheet ehditään havaita ennen niiden päätymistä tuotantoon.

Osaa tuottaa ja hyödyntää SBOMia

Ohjelmistokehittäjän tulee osata luoda näkyvyys ohjelmiston komponentteihin ja riippuvuuksiin rakentamisvaiheessa. Tämä tarkoittaa Software Bill of Materials (SBOM) -tiedon tuottamista osana kehitysputkea joko erillisenä tiedostona tai SCA-työkalun avulla. Kehittäjän on ymmärrettävä, miten SBOM tukee haavoittuvuuksien tunnistamista, lisenssien hallintaa ja komponenttien alkuperän varmentamista sekä miten sitä hyödynnetään ohjelmiston turvallisuuden ylläpitämisessä.

Ymmärtää vaatimustenmukaisuuden varmentamisen rakentamisvaiheessa

Rakentamisvaiheessa ohjelmistokehittäjän tulee ymmärtää vaatimustenmäärittelyvaiheessa asetetut vaatimukset käytännön tasolla ja osata varmentaa niiden toteutuminen. Tämä edellyttää ymmärrystä siitä, miten vaatimukset näkyvät ohjelmiston rakenteessa, konfiguraatioissa ja tuotantopaketissa. Kehittäjän tulee osata käyttää vaatimuksia tukevia viitekehyksiä ja dokumentaatioita varmennuksen apuna sekä varmistaa, että ohjelmiston rakentaminen tukee asetettujen vaatimusten täyttymistä. Kehittäjän tulee tunnistaa tilanteet, joissa alkuperäiset suunnitelmat eivät ole tarkoituksenmukaisia tai toteutuskelpoisia, sekä esittää vaihtoehtoisia ratkaisuja ja päivittää vaatimusmäärittelyjä yhteistyössä muun kehitystiimin kanssa.

Osaa käsitellä ja hyödyntää testituloksia rakentamisvaiheessa

Ohjelmistokehittäjän tulee analysoida yksikkö-, integraatio- ja E2E-testien tulokset osana CI/CD-putkea sekä varmistaa, että havaitut virheet ja tietoturvalöydökset korjataan ennen julkaisua. Kehittäjän on arvioitava testitulosten vaikutus ohjelmiston toiminnallisuuteen, riippuvuuksiin ja turvallisuuteen, jotta kokonaisuus säilyy eheänä ja luotettavana.

Keskeiset toimet

- Konfiguroi CI/CD-putki tietoturvallisesti ja varmista, että rakennusprosessi on automatisoitu, dokumentoitu ja toistettavissa.
- Varmista, että kehitys-, testi- ja tuotantorakennukset tapahtuvat eriytetyissä ja tarkoitukseen sopivissa ympäristöissä.
- Hyödynnä rakentamisen aikaisia tietoturvaskannereita (esim. DAST, SCA) ja käsittele löydökset ennen julkaisua.
- Valvo ja päivitä kolmansien osapuolten komponentteja hallitusti ottaen huomioon päivityksiin liittyvät riskit.
- Tuota ja hyödynnä SBOM osana kehitysputkea toimitusketjuriskien hallitsemiseksi.
- Varmista, että rakennettu ohjelmistoversio täyttää määrittelyvaiheessa asetetut vaatimukset ennen tuotantoon siirtoa.

JULKAISU JA KÄYTTÖÖNOTTO

Älä vie tuotantoon oletuksia, vaan varmistettuja ratkaisuja.

Julkaisu- ja käyttöönottovaiheessa ohjelmistokehittäjän tehtävänä on huolehtia, että ohjelmisto toimii suunnitellusti, ei sisällä käyttöönottoa tai käyttöä estäviä haavoittuvuuksia ja että loki- ja virhetiedot tallentuvat oikein. Kehittäjä tukee julkaisusta vastaavia henkilöitä virheiden analysoinnissa ja niiden korjaamisessa. Kehittäjä myös osallistuu tarvittaessa käyttäjien koulutukseen ja ohjeistamiseen.

Osaa arvioida haavoittuvuuksien vaikutukset ja toteuttaa korjaukset

Julkaisuvaiheessa ohjelmistokehittäjän tulee ymmärtää ohjelmiston toiminnallisuudet ja rakenteet siinä määrin, että havaittujen tietoturvahaavoittuvuuksien vaikutus ohjelmistolle voidaan arvioida. Kehittäjän on osattava tunnistaa, mitkä haavoittuvuudet ovat ohjelmiston kannalta kriittisiä ja edellyttävät korjaamista ennen julkaisua. Osaamiseen kuuluu myös korjausten toteuttaminen havaittuihin tietoturvaongelmiin siten, että ohjelmiston toiminnallisuus ja tietoturva säilyvät suunnitellulla tasolla.

Osaa toteuttaa korjaukset koodikatselmoinneissa havaittuihin puutteisiin

Ohjelmistokehittäjän tulee osata hyödyntää koodikatselmointeja osana julkaisuvaiheen laadun ja tietoturvan varmistamista. Tämä edellyttää riittävää ymmärrystä ohjelmiston toiminnasta, jotta katselmoinneissa havaitut virheet ja puutteet voidaan korjata sekä lähdekoodissa että konfiguraatioissa. Kehittäjän vastuulla on varmistaa, että korjaukset toteutetaan huolellisesti ja että ne eivät aiheuta uusia ongelmia tai regressioita ennen ohjelmiston julkaisua.

Osaa hyödyntää julkaisuvaiheen tietoturvaskannereita

Julkaisu- ja käyttöönottovaiheessa ohjelmistokehittäjän tulee osata ottaa käyttöön tarvittavat tietoturvaskannerit ja ymmärtää niiden tuottamien havaintojen merkitys ohjelmiston turvallisuudelle. Konttien tietoturvaskannauksen avulla kehittäjän on varmistettava, että käytetyt konttiratkaisut eivät sisällä vanhentuneita kirjastoja, haavoittuvia käyttöjärjestelmäkomponentteja tai puutteellisia konfiguraatioita ennen tuotantoon siirtämistä. Kehittäjän osaamiseen kuuluu myös skannaustulosten tulkinta ja niiden perusteella tehtävät korjaavat toimenpiteet, jotta julkaisu täyttää asetetut tietoturvavaatimukset eikä tuo uusia riskejä tuotantoympäristöön.

Osaa varmistaa lokituksen ja monitoroinnin vaatimusten toteutumisen

Julkaisuvaiheessa ohjelmistokehittäjän tulee varmistaa, että ohjelmisto tuottaa tarvittavat lokitiedot asetettujen vaatimusten mukaisesti. Tämä sisältää sen varmistamisen, että lokit ovat sisällöllisesti riittäviä, säilytysajat on huomioitu ja lokit eivät sisällä salassa pidettävää tietoa, kuten selkokielisiä salasanoja, todentamisavaimia, API-avaimia tai muita salaisuuksia. Kehitysvaiheen lokit voivat sisältää enemmän tietoa, mutta ennen tuotantoon siirtymistä lokeista on poistettava kaikki sensitiivinen tieto. Kehittäjän on lisäksi osattava tuottaa lokitiedot muodossa, joka mahdollistaa niiden siirtämisen ja hyödyntämisen keskitetyssä lokienhallinta- ja valvontajärjestelmässä.

Osaa tukea käyttäjiä ohjeistuksella ja dokumentoida kehitystarpeet

Julkaisuvaiheessa ohjelmistokehittäjän tulee osata tarvittaessa tukea käyttäjiä ohjelmiston käyttöönotossa ja käytössä. Tämä voi tarkoittaa keskeisten toiminnallisuuksien esittelyä tai käyttöön liittyvien erityispiir-

teiden selventämistä. Lisäksi kehittäjän on osattava dokumentoida julkaisun yhteydessä havaitut kehitystarpeet ja jatkokehitystä vaativat havainnot, jotta ne voidaan huomioida ohjelmiston seuraavissa versioissa.

Keskeiset toimet

- Arvioi tietoturvaskannausten, koodikatselmointien ja tarkistusten löydökset ja toteuta tarvittavat korjaukset ennen julkaisua.
- Varmista, etteivät tehdyt korjaukset riko toiminnallisuuksia tai aiheuta uusia virheitä tai regressioita.
- Toteuta julkaisuvaiheen tietoturvaskannaukset ja käsittele löydökset ennen tuotantoon siirtämistä.
- Varmista, että lokitus ja virhetiedot ovat riittävät, eivät sisällä suojattavaa tietoa, ja tukevat monitorointia ja valvontaa.
- Tue julkaisusta vastaavia henkilöitä virhetilanteiden analysoinnissa.
- Kirjaa julkaisussa havaitut kehitystarpeet ja tietoturvahavainnot jatkokehitystä varten.

YLLÄPITO JA MONITOROINTI

Ylläpitovaiheessa ohjelmistokehittäjä vastaa ohjelmiston päivitysten toteuttamisesta sekä tietoturvaparannusten toteutuksesta. Kehittäjän tulee tuntea ohjelmistossa käytettävät ratkaisut siten, että julkaisu- ja käyttöönottovaiheessa ilmenevien haavoittuvuuksien merkitys kyseessä olevaan ohjelmistoon pystytään arvioimaan, kriittiset haavoittuvuudet priorisoimaan ja korjaamaan.

Osaa tukea tietoturvapoikkeamien käsittelyä ylläpitovaiheessa

Ylläpitovaiheessa ohjelmistokehittäjän tulee ymmärtää ohjelmistossa käytetyt ratkaisut ja niiden keskinäiset riippuvuudet siinä määrin, että hän pystyy tukemaan tietoturvapoikkeamien vaikutusten arviointia. Kehittäjän osaamiseen kuuluu poikkeamien yhteydessä tunnistettujen muutostarpeiden toteuttaminen, kuten korjausten, konfiguraatiomuutosten tai parannusten vieminen ohjelmistoon hallitusti. Tämän osaamisen avulla kehittäjä tukee poikkeamien tehokasta käsittelyä ja varmistaa, että ohjelmiston tietoturva palautuu ja säilyy suunnitellulla tasolla myös häiriötilanteiden jälkeen.

Osaa toteuttaa päivitykset ja konfiguraatiokovennukset

Ylläpitovaiheessa ohjelmistokehittäjän tulee osata toteuttaa ohjelmiston komponentteihin kohdistuvat korjaus- ja tietoturvapäivitykset hallitusti ja suunnitellun käytännön mukaisesti. Tämä edellyttää ymmärrystä siitä, miten päivitykset vaikuttavat ohjelmiston toimintaan ja yhteensopivuuteen sekä kykyä viedä muutokset tuotantoon ilman tarpeettomia häiriöitä. Lisäksi kehittäjän tulee osata toteuttaa konfiguraatiokovennuksia, joilla poistetaan tarpeettomia toimintoja, vahvistetaan oletusasetuksia ja pienennetään ohjelmiston hyökkäyspintaa. Näin ylläpidon aikana voidaan varmistaa, että ohjelmisto pysyy ajantasaisena ja tietoturvallisena myös muuttuvassa toimintaympäristössä.

Osaa toteuttaa uudet toiminnallisuudet sovittujen käytänteiden mukaisesti

Ohjelmistokehittäjän tulee toteuttaa uudet ja päivitettävät toiminnallisuudet sovittujen tietoturvakäytänteiden mukaisesti, sekä toteuttaa toiminnallisuuksille uhkamallinnus, jotta muutosten aiheuttamat mahdolliset uudet riskit tunnistetaan ennen toteutusta. Kehittäjän on varmistettava, etteivät muutokset heikennä olemassa olevia suojausmekanismeja, ja arvioitava niiden vaikutus ohjelmiston toimintaan, riippuvuuksiin, lokitukseen ja valvontaan, jotta järjestelmän turvallisuus ja havaittavuus säilyvät eheänä ja luotettavana.

Kehittäjä varmistaa, että turvallisuus säilyy myös muutoksissa.

Keskeiset toimet

- Tue tietoturvapoikkeamien käsittelyä toteuttamalla havaitut korjaus- ja muutostarpeet ohjelmistoon hallitusti.
- Arvioi poikkeamien vaikutukset ohjelmiston toimintaan ja varmista, että korjaukset palauttavat tietoturvan suunnitellulle tasolle.
- Toteuta komponenttien tietoturva- ja korjauspäivitykset säännöllisesti ja suunnitellun päivityskäytännön mukaisesti.
- Vahvista ohjelmiston suojaustasoa toteuttamalla tarvittavat konfiguraatiokovennukset.
- Toteuta uusille toiminnallisuuksille uhkamallinnus ja noudata toteutuksessa sovittuja tietoturvakäytänteitä
- Varmista, ettei muutokset vaikuta ohjelmiston tietoturvan tasoon heikentävästi

KÄYTÖSTÄ POISTO

Ohjelmiston elinkaaren päättyessä ohjelmistokehittäjä toimii yhdessä arkkitehdin ja ylläpidon kanssa käytöstä poiston toteuttajana. Ohjelmistokehittäjän on tunnettava ohjelmiston toiminnallisuudet ja ymmärrettävä poistamiseen liittyvät vaatimukset ja toimenpiteet. Vastuulla on toteuttaa ohjelmiston ja sen rajapintojen, mikropalveluiden ja integraatioiden sulkeminen, ulkoisten riippuvuuksien ja käyttöoikeuksien poistaminen, sekä näiden toimenpiteiden dokumentointi ja todentaminen.

Ymmärtää alasajoprosessin ja osaa toteuttaa sen turvallisesti

Käytöstä poiston vaiheessa ohjelmistokehittäjän tulee ymmärtää arkkitehdin ja kehitystiimin kanssa laadittu alasajosuunnitelma ja osata toimia sen mukaisesti. Kehittäjän on varmistettava, että alasajo kattaa ohjelmistoon liittyvien API-avainten, sertifikaattien ja muiden tunnistetietojen turvallisen hallinnan ja poistamisen. Osaamiseen kuuluu myös alas ajettavan järjestelmän riippuvuuksien tunnistaminen ja sen varmistaminen, että niihin liittyvät yhteydet suljetaan hallitusti ilman vaikutuksia muihin järjestelmiin.

Lisäksi ohjelmistokehittäjän tulee osata toteuttaa tietojen tietoturvallinen ja pysyvä tuhoaminen hyväksytyillä menetelmillä, kuten salausavainten tuhoamisella ja tietojen ylikirjoittamisella. Tarvittaessa on ymmärrettävä myös tilanteet, joissa tietojen fyysinen tuhoaminen on perusteltua. Kehittäjän vastuulla on toteuttaa ohjelmistokomponenttien turvallinen käytöstä poisto siten, ettei järjestelmään jää hyödynnettävissä olevia osia tai tietoja.

Ymmärtää loki- ja audit-tietojen säilytysvaatimukset ja osaa toteuttaa ne

Käytöstä poiston yhteydessä ohjelmistokehittäjän tulee ymmärtää loki- ja audit-tietoihin liittyvät lainsäädännölliset ja sopimukselliset säilytysvelvoitteet. Kehittäjän on osattava toteuttaa lokien turvallinen säilyttäminen siten, että tiedot säilyvät eheänä, luottamuksellisena ja saatavilla vaaditun ajan. Tämä varmistaa, että käytöstä poisto ei riko velvoitteita eikä aiheuta tietoturva- tai vaatimustenmukaisuusriskejä ohjelmiston elinkaaren päättyessä.

Turvallinen alasajo on osa vastuullista kehitystyötä.

Keskeiset toimet

- Noudata alasajosuunnitelmaa ja varmista, että API-avaimet, sertifikaatit ja muut tunnistetiedot poistetaan turvallisesti.
- Tunnista alas ajettavan järjestelmän riippuvuudet ja sulje niihin liittyvät yhteydet hallitusti ilman vaikutuksia muihin järjestelmiin.
- Toteuta tietojen pysyvä ja tietoturvallinen tuhoaminen hyväksytyillä menetelmillä, kuten salausavainten tuhoamisella ja tietojen ylikirjoittamisella.
- Poista ohjelmistokomponentit siten, ettei järjestelmään jää hyödynnettävissä olevia osia tai tietoja.
- Varmista, että loki- ja audit-tiedot säilytetään turvallisesti lainsäädännön ja sopimusten edellyttämän ajan.

Ilman testausta turvallisuus jää oletukseksi.

Testaaja

Tässä osiossa käsitellään testaajan tietoturvaan liittyviä osaamistarpeita ohjelmistokehityksen eri vaiheissa. Osiossa kuvataan, miten testaus tukee ohjelmiston turvallisuutta ja millaisia taitoja tarvitaan, jotta haavoittuvuudet havaitaan ajoissa ja kokonaisuus toimii luotettavasti.

MIKSI OHJELMISTOTURVALLISUUS KUULUU TESTAAJALLE?

Testaaja vastaa ohjelmiston laadunvarmistuksesta ja siitä, että ratkaisu täyttää suunnitteluvaiheessa asetetut toiminnalliset, laadulliset ja suorituskykyvaatimukset. Tehtäviin kuuluu testitapausten ja testaussuunnitelman laatiminen sekä testauksen suorittaminen tai ohjaaminen, riippuen siitä, onko testaus manuaalista vai automatisoitua. Testaaja vastaa myös erilaisten testien toteuttamisesta, kuten hyväksymistestauksista, kuormitustesteistä ja kehitysputkessa ajettavista tietoturvatesteistä.

Havaittujen virheiden ja puutteiden raportointi on olennainen osa testaajan työtä. Kehittäjien ja arkkitehtien kanssa tehtävä yhteistyö varmistaa, että löydetyt ongelmat ymmärretään ja korjaukset voidaan toteuttaa oikea-aikaisesti, jolloin testaus tukee ohjelmiston kokonaislaadun ja turvallisuuden parantamista.

Testaajan keskeiset tietoturvataidot rakentuvat kyvystä ymmärtää ohjelmistokehityksen elinkaarta ja tunnistaa siihen liittyvät riskit eri vaiheissa. Testauksen teknisten menetelmien ja tietoturvan perusperiaatteiden hallinta mahdollistaa testauksen suunnittelun ja toteutuksen siten, että haavoittuvuudet havaitaan ajoissa ja ohjelmiston turvallinen ja luotettava toiminta voidaan varmistaa.

Tiivistelmä

TIIVISTELMÄ OSAAMISTARPEISTA

- Ymmärtää ohjelmistolle asetetut tietoturva- ja vaatimustenmukaisuusvaatimukset ja osaa suunnitella testauksen niiden todentamiseksi.
- Osaa hyödyntää uhkamallinnusta ja riskianalyysiä testauksen kohdentamisessa ja priorisoinnissa.
- Ymmärtää tietoturvatestausten kokonaisuuden ohjelmiston elinkaaren eri vaiheissa ja osaa tukea niiden suunnittelua.
- Tuntee keskeiset testausmenetelmät ja työkalut, mukaan lukien automaattinen testaus, penetraatiotestaus, fuzz-testaus sekä white-, grey- ja black-box-testaus.
- Ymmärtää yleisimmät ohjelmisto- ja rajapintahaavoittuvuudet (OWASP Top 10 ja OWASP API Top 10) ja osaa huomioida ne testauksessa.
- Osaa toteuttaa manuaalisia tietoturvatestauksia ja tulkita testien tuloksia ohjelmiston turvallisuuden näkökulmasta.
- Osaa arvioida automatisoitujen tietoturvatestien ja -skannereiden tuloksia sekä erottaa merkitykselliset löydökset vääristä positiivisista.
- Osaa raportoida testitulokset ja tietoturvahavainnot kehitystiimille selkeästi ja korjattavassa muodossa.
- Ymmärtää testauksen roolin julkaisussa ja käyttöönotossa ja osaa varmentaa, että tuotantoversio täyttää sovitut tietoturvakontrollit.
- Osaa tukea jatkuvaa tietoturvatestausta ylläpitovaiheessa, mukaan lukien regressiotestaukset ja riippuvuuspäivitysten testaus.
- Ymmärtää tietoturvapoikkeamien käsittelyn testauksen näkökulmasta ja osaa tukea niiden tunnistamista ja raportointia.
- Osaa varmentaa lokituksen, hälytysten ja monitorointiratkaisujen toimivuuden sekä testata niitä säännöllisesti tuotantoympäristössä.

Hyvä testaus alkaa huolellisella suunnittelulla.

SUUNNITTELU

Suunnitteluvaiheessa testaaja osallistuu ohjelmiston testaussuunnitelman ja testausprosessien laatimiseen. Suunnitelman tulee perustua uhkamallinnukseen pohjautuvaan riskiarvioon. Testaaja vastaa testitapausten suunnittelusta ohjelmiston toiminnallisuuksien, arkkitehtuurin ja tietoturvavaatimusten perusteella. Testien on oltava kattavia ja relevantteja, ja niissä tulee huomioida myös harvinaisemmat toiminnot. Lisäksi testaajan on varmistettava, että testitapaukset pyrkivät tunnistamaan yleisimmät ohjelmistoissa esiintyvät haavoittuvuustyypit.

Ymmärtää vaatimustenmukaisuuden ja osaa suunnitella testauksen sen todentamiseksi

Suunnitteluvaiheessa testaajan tulee ymmärtää ohjelmistolle asetetut tietoturvavaatimukset ja osata suunnitella testitapaukset siten, että vaatimusten täyttyminen voidaan todentaa testauksen avulla. Tämä edellyttää yhteistyötä arkkitehdin ja ohjelmistokehittäjän kanssa, jotta testaaja ymmärtää ohjelmiston toiminnallisuudet, arkkitehtuurin ja keskeiset tekniset

ratkaisut. Näiden tietojen pohjalta testaaja laatii testaussuunnitelman, joka kohdistaa testauksen niihin osa-alueisiin, joissa vaatimustenmukaisuuden varmentaminen on kriittistä.

Ymmärtää riskianalyysin tulokset ja osaa hyödyntää niitä testauksen suunnittelussa
Suunnitteluvaiheessa testaajan tulee ymmärtää riskianalyysin tunnistamat ohjelmistoon kohdistuvat riskit ja niiden merkitys testauksen näkökulmasta. Testaajan osaamiseen kuuluu riskien huomioiminen testitapauksia suunniteltaessa siten, että testaus kohdistuu niihin toiminnallisuuksiin ja rakenteisiin, joissa riskienhallinnan tarve on suurin. Näin testaus tukee järjestelmällisesti riskien tunnistamista ja hallintaa jo ennen toteutusvaihetta.

Ymmärtää tietoturvatestausten kokonaisuuden ja osaa tukea niiden suunnittelua
Suunnitteluvaiheessa testaajan tulee ymmärtää, millaisia tietoturvatestauksia ohjelmiston elinkaaren eri vaiheissa tarvitaan ja miten ne tukevat riskienhallintaa ja vaatimustenmukaisuuden varmentamista. Testaajan rooliin kuuluu tietoturvatestausten suunnittelun tukeminen sekä osallistuminen siihen, mitä automaattisia ja manuaalisia testausmenetelmiä on tarkoituksenmukaista hyödyntää toteutus-, rakentamis-, testaus- ja julkaisuvaiheissa.

Testaajan tulee osata tukea sen määrittämistä, mitkä tietoturvaskannerit liitetään osaksi kehitystyötä ja testausputkea. Tämä voi sisältää esimerkiksi staattisen koodianalyysin (SAST) ja infrastruktuurikoodin skannauksen (IaC) toteutusvaiheessa, dynaamisen testauksen (DAST) ja riippuvuusskannauksen rakentamisvaiheessa sekä penetraatio- ja fuzz-testauksen testausvaiheessa. Lisäksi testaajan on ymmärrettävä konttien skannauksen rooli julkaisuvaiheessa.

Osaamiseen kuuluu myös erilaisten manuaalisten penetraatiotestausmenetelmien ymmärtäminen. Testaajan tulee tunnistaa, milloin white-box-, grey-box- tai black-box-testaus on tarkoituksenmukaista ja miten näitä menetelmiä hyödynnetään testauksen suunnittelussa ohjelmiston riskiprofiilin ja käytettävissä olevan tiedon perusteella.

Osaa suunnitella testauksen automatisointia osaksi kehitys- ja julkaisuprosessia
Suunnitteluvaiheessa testaajan tulee osata suunnitella testaus mahdollisimman pitkälle automatisoiduksi osaksi kehitys- ja julkaisuprosessia. Tämä edellyttää ymmärrystä siitä, mitkä testit soveltuvat automatisoitaviksi ja miten automaatio tukee testauksen toistettavuutta, kattavuutta ja oikea-aikaisuutta eri elinkaaren vaiheissa. Testaajan osaaminen auttaa varmistamaan, että testaus integroidaan luontevaksi osaksi kehitys- ja julkaisuputkea ilman, että se hidastaa kehitystyötä tai heikentää testauksen laatua.

Keskeiset toimet

- Laadi testitapaukset, joilla tietoturvavaatimusten täyttyminen voidaan todentaa.
- Kohdista testaus riskianalyysin perusteella niihin toiminnallisuuksiin ja rakenteisiin, joissa riskit ovat suurimmat.
- Laadi testaussuunnitelma yhdessä arkkitehdin ja kehittäjän kanssa varmistaaksesi riittävän kattavuuden.
- Määritä, mitä tietoturvatestauksia ja skannereita hyödynnetään eri elinkaaren vaiheissa ja missä kohtaa putkea niitä ajetaan.
- Valitse tarkoituksenmukaiset testausmenetelmät (automaattinen, penetraatio-, fuzz-, white/grey/black-box) ohjelmiston riskiprofiilin perusteella.
- Suunnittele testauksen automatisointi osaksi kehitys- ja julkaisuprosessia siten, että testaus on toistettavaa ja ajantasaista.

TOTEUTUS

Ohjelmistotestaajan vastuut toteutusvaiheessa liittyvät ohjelmiston toiminnallisuuksien tuntemiseen. Tietoturvatestauksen näkökulmasta testaajan on ymmärrettävä ohjelmiston toteutustapa, sillä eri toiminnallisuudet altistuvat erilaisille haavoittuvuuksille ja hyökkäysmenetelmille. Siksi toiminnallisuuksien ja yleisimpien ohjelmistohaavoittuvuuksien tunteminen on erityisen tärkeää.

Ymmärtää tietoturvaskannausten roolin ja osaa tukea niiden hyödyntämistä toteutusvaiheessa

Toteutusvaiheessa testaajan tulee ymmärtää yleisimmät ohjelmistohaavoittuvuudet ja niiden syntymekanismit, jotta voidaan arvioida, millaisille haavoittuvuuksille kehitettävä ohjelmisto voi olla altis. Tämä edellyttää tunnetuimpien haavoittuvuuksien tuntemista sekä kykyä suhteuttaa ne ohjelmiston toiminnallisuuksiin ja arkkitehtuuriin. Arvioinnissa apuna voidaan käyttää esimerkiksi OWASP Top 10- ja OWASP Top 10 API -haavoittuvuuslistauksia. Testaajan osaamiseen kuuluu varmistaa, että toteutusputkessa ajetaan tarkoituksenmukaiset tietoturvaskannaukset riskiarvion ja ohjelmiston ominaisuuksien perusteella, jotta haavoittuvuudet tunnistetaan mahdollisimman varhaisessa vaiheessa ja testaus tukee turvallista toteutusta.

Keskeiset toimet

- Kohdista tietoturvaskannaukset ohjelmiston riskiprofiilin perusteella ja varmista, että ne ajetaan toteutusputkessa oikeissa kohdissa.
- Hyödynnä OWASP Top 10- ja OWASP API Top 10 -listoja arvioidaksesi, millaisille haavoittuvuuksille toteutus voi olla altis.

Tietoturvatestaus onnistuu, kun testaaja ymmärtää sekä toiminnallisuudet että niihin liittyvät haavoittuvuudet.

TESTAUS

Testausvaiheessa ohjelmistotestaajan vastuulla on toteuttaa suunnitellut tietoturva- ja muut testit varmistaakseen, että toteutettu ohjelmisto on turvallinen ja toimiva. Testauksen tulee sisältää niin toiminnallisia, suorituskykyyn kuin tietoturvaankin liittyviä testejä. Näillä varmistetaan, että ominaisuudet toimivat odotetusti, ohjelmisto kestää suunnitellun kuormituksen ja ettei siinä ole hyväksikäytettäviä haavoittuvuuksia. Testaaja analysoi tulokset, raportoi ne kehitystiimille ja varmistaa korjausten toteutumisen uusintatesteillä.

Osaa toteuttaa tietoturvatestauksen ja tulkita sen tulokset

Testausvaiheessa testaajan tulee osata toteuttaa ohjelmistolle manuaalista tietoturvatestausta, mukaan lukien penetraatiotestaus, jolla pyritään tunnistamaan ohjelmiston yleisimmin esiintyvät ja kriittisimmät haavoittuvuudet. Testaajan osaamiseen kuuluu erityisesti OWASP Top 10- ja OWASP API Top 10 -haavoittuvuuksien varmentaminen sekä muiden ohjelmiston toiminnallisuuksien kannalta merkittävien heikkouksien tunnistaminen. Testaaja vastaa testauksen toteutuksesta, mutta ei korjausten suunnittelusta tai toteuttamisesta.

Osaamiseen kuuluu myös fuzz-testauksen hyödyntäminen osana testausvaihetta. Testaajan tulee osata konfiguroida fuzzaustyökalu tuottamaan virheellistä ja epänormaalia syötettä ohjelmistolle, jotta voidaan tunnistaa tilanteet, joissa ohjelmisto käyttäytyy odottamattomasti, kuormittuu hallitsemattomasti tai kaatuu. Testauksen tavoitteena on paljastaa virheet, jotka eivät ilmene tavanomaisissa käyttötapauksissa.

Osaa arvioida automatisoitujen tietoturvatestien tuloksia

Testausvaiheessa testaajan tulee osata arvioida kehitys- ja testausputkessa ajettavien automatisoitujen tietoturvatestien ja -skannereiden tuottamia tuloksia. Tämä edellyttää kykyä erottaa merkitykselliset löydökset vääristä positiivisista sekä ymmärtää havaintojen vaikutus ohjelmiston tietoturvaan ja toimintaan.

Osaa raportoida testitulokset ja kuvata havaitut haavoittuvuudet korjattavassa muodossa

Testausvaiheessa testaajan vastuulla on testitulosten ja havaintojen selkeä raportointi kehitystiimille. Tämä edellyttää kykyä analysoida havaittujen haavoittuvuuksien merkitys ohjelmiston tietoturvan ja toiminnallisuuden kannalta sekä erottaa olennaiset löydökset vähemmän merkityksellisistä havainnoista. Testaajan on osattava kuvata haavoittuvuuden hyväksikäyttötapa, vaikutukset ja toistettavuus riittävällä tarkkuudella, jotta kehittäjät pystyvät toteuttamaan tarvittavat korjaukset. Selkeä ja täsmällinen raportointi varmistaa, että testauksen havainnot johtavat konkreettisiin parannuksiin ohjelmiston turvallisuudessa.

Keskeiset toimet

- Toteuta manuaaliset tietoturvatestit, kuten penetraatio- ja fuzz-testaukset, kriittisten haavoittuvuuksien tunnistamiseksi.
- Arvioi automatisoitujen tietoturvatestien ja skannereiden tulokset ja erottele olennaiset löydökset vääristä positiivisista.
- Analysoi havaittujen haavoittuvuuksien vaikutukset ja priorisoi ne testauksen näkökulmasta.
- Raportoi testitulokset kehitystiimille selkeästi ja kuvaa hyväksikäyttötavat siten, että korjaukset voidaan toteuttaa.

Testaaja muuttaa havainnot kehitystä ohjaavaksi tiedoksi.

JULKAISU JA KÄYTTÖÖNOTTO

Julkaisu- ja käyttöönottovaiheessa testaaja varmistaa, että ohjelmisto voidaan ottaa käyttöön turvallisesti ja hallitusti. Testaajan vastuulla on toteuttaa käyttöönotto- ja hyväksymistestaukset, varmistaa, että toteutetut tietoturvakonfiguraatiot toimivat myös tuotantoympäristössä, ja todentaa että lokienkeruu ja valvonta toimivat suunnitellusti.

Osaa arvioida julkaisuvaiheen tietoturvaskannausten tuloksia

Julkaisu- ja käyttöönottovaiheessa testaajan tulee osata arvioida julkaisuprosessissa ajettavien tietoturvaskannereiden, kuten konttiskannaustyökalujen, tuottamia löydöksiä. Testaajan vastuulla on tunnistaa löydöksistä ohjelmiston turvallisuuden kannalta merkitykselliset havainnot ja erottaa ne vähemmän merkityksellisistä tai vääristä positiivisista. Lisäksi testaajan tulee raportoida havaitut tietoturvaongelmat kehitystiimille selkeästi ja oikea-aikaisesti, jotta mahdolliset korjaukset voidaan tehdä ennen tuotantoon siirtoa.

Osaa toteuttaa käyttöönottotestauksen tietoturvan näkökulmasta

Julkaisu- ja käyttöönottovaiheessa testaajan tulee osata varmistaa, että ohjelmiston tuotantoversio toteuttaa sille asetetut tietoturvakontrollit. Tämä edellyttää sen todentamista, että kehitysvaiheessa määritellyt tietoturvalliset konfiguraatiot, pääsynhallintakontrollit ja lokitussäännöt toimivat tuotantoympäristössä suunnitellulla tavalla. Testaajan vastuulla on lisäksi varmistaa, että aiemmin tunnistetut ja korjattavaksi sovitut haavoittuvuudet on tosiasiallisesti korjattu ennen käyttöönottoa. Näin käyttöönottotestaus tukee sitä, ettei tuotantoon päädy ratkaisuja, jotka heikentäisivät ohjelmiston tietoturvaa tai rikkoisivat sovittuja vaatimuksia.

Keskeiset toimet

- Arvioi julkaisuprosessissa ajettujen tietoturvaskannausten, kuten konttiskannauksen, löydökset ja tunnista niistä olennaiset riskit.
- Raportoi havaitut tietoturvaongelmat kehitystiimille ajoissa ja selkeästi ennen tuotantoon siirtoa.
- Varmista käyttöönottotestauksessa, että tuotantoversio toteuttaa sovitut tietoturvakontrollit.
- Todenna, että tietoturvalliset konfiguraatiot, pääsynhallintakontrollit ja lokitussäännöt toimivat tuotantoympäristössä suunnitellusti.
- Varmista, että aiemmin tunnistetut ja korjattavaksi sovitut haavoittuvuudet on aidosti korjattu ennen käyttöönottoa.

Testaus varmistaa, että tuotantoon siirtyy vain hyväksytty ja turvallinen kokonaisuus.

YLLÄPITO JA MONITOROINTI

Ohjelmiston käyttöönoton jälkeen testaaja osallistuu jatkuvaan laadun ja tietoturvan varmistamiseen toteuttamalla säännöllisiä tietoturvatestauksia sekä muutostilanteissa regressiotestauksia. Testaaja raportoi havaitut haavoittuvuudet ja varmentaa niiden korjauksen. Monitorointivaiheessa testaaja varmistaa, että ohjelmiston lokitus ja muut valvontamekanismit, kuten hälytykset, toimivat suunnitellusti ja tuottavat hallittavaa ja riittävää tietoa.

Osaa varmistaa säännöllisen tietoturvatestauksen ylläpitovaiheessa

Ylläpitovaiheessa testaajan tulee osata varmistaa, että ohjelmistolle suoritetaan säännöllisiä tietoturvatestauksia ja -skannauksia myös tuotantokäytön aikana.

Säännöllinen testaus pitää turvallisuuden ajan tasalla.

Testaajan vastuulla on todentaa, että kehitys- ja ylläpitoputkissa ajetaan automatisoidut tietoturvaskannaukset niille tarkastuksille, jotka on määritelty tarpeellisiksi ohjelmiston riskiprofiilin perusteella.

Osaamiseen kuuluu lisäksi kyky toteuttaa tarvittaessa manuaalisia tietoturvatestauksia silloin, kun automaattiset menetelmät eivät ole riittäviä tai kun ohjelmistoon tehdään merkittäviä muutoksia. Testaajan on myös varmistettava, että muutospäivitysten ja korjausten yhteydessä suoritetaan tarvittavat testaukset, jotta ohjelmiston tietoturva ja toimivuus säilyvät tuotantopäivityksen jälkeen. Näin testaus tukee ohjelmiston turvallista käyttöä koko sen elinkaaren ajan.

Osaa varmistaa muutos- ja riippuvuuspäivitysten turvallisuuden ylläpitovaiheessa
Ylläpitovaiheessa testaajan tulee osata varmistaa, etteivät ohjelmistoon tehtävät toiminnallisuus- tai riippuvuuspäivitykset heikennä ohjelmiston toiminnallisuutta tai tietoturvaa tuotantoympäristössä. Tämä edellyttää kykyä arvioida päivitysten vaikutuksia ohjelmiston kokonaisuuteen ja tunnistaa tilanteet, joissa riippuvuuspäivitykset vaativat erityistä huomiota testauksen näkökulmasta.

Testaajan osaamiseen kuuluu tarvittavien testien toteuttaminen päivitysten yhteydessä, jotta voidaan todentaa, että ohjelmisto toimii odotetusti ja että päivitykset eivät tuo mukanaan uusia virheitä, haavoittuvuuksia tai regressioita. Näin testaus tukee turvallista ja hallittua ohjelmiston ylläpitoa myös jatkuvien muutosten aikana.

Osaa tukea tietoturvapoikkeamien tunnistamista ja käsittelyä ylläpitovaiheessa
Ylläpitovaiheessa testaajan tulee osata varmistaa, että tuotantoympäristössä tehtävissä testauksissa esiin tulevat tietoturvahaavoittuvuudet tunnistetaan ja dokumentoidaan asianmukaisesti. Testaajan vastuulla on havaintojen analysointi testauksen näkökulmasta sekä niiden raportointi kehitystiimille riittävällä tarkkuudella, jotta poikkeamiin voidaan reagoida tehokkaasti. Testaajan rooli ei ole korjausten toteuttaminen, vaan sen varmistaminen, että löydökset ovat ymmärrettäviä, todennettavia ja ohjaavat oikea-aikaiseen korjaamiseen. Näin testaaja tukee tietoturvapoikkeamien hallittua käsittelyä ja ohjelmiston turvallista käyttöä myös tuotantovaiheessa.

Osaa varmentaa lokien riittävyyden monitorointivaiheessa

Monitorointivaiheessa testaajan tulee osata varmentaa, että ohjelmiston lokit kertyvät suunnitellulla tavalla ja että ne kattavat ne tapahtumat, joita poikkeamien ja tietoturvahäiriöiden havaitseminen edellyttää. Testaajan osaamiseen kuuluu lokien sisällön arviointi sen näkökulmasta, onko kerätty tieto riittävää, ajantasaista ja käyttökelpoista poikkeamien tunnistamiseen ja analysointiin.

Testaajan vastuulla ei ole lokituksen tekninen toteutus, vaan sen todentaminen, että lokitus tukee tehokasta monitorointia ja että mahdolliset puutteet tunnistetaan ja raportoidaan kehitystiimille tai muille vastuullisille tahoille. Näin testaaja tukee sitä, että ohjelmiston valvonta toimii luotettavasti myös tuotantokäytössä.

Osaa varmentaa hälytyssääntöjen toimivuuden monitorointivaiheessa

Monitorointivaiheessa testaajan tulee osata varmentaa, että ohjelmistolle määritellyt hälytyssäännöt toimivat suunnitellusti poikkeavien ja tietoturvan kannalta merkittävien tilanteiden havaitsemiseksi. Tämä edellyttää kykyä testata ja todentaa, että hälytykset laukeavat odotetuissa tilanteissa ja että ne perustuvat oikeisiin lokitapahtumiin ja kynnysarvoihin.

Testaajan vastuulla on lisäksi varmistaa, että hälytykset siirtyvät sovitusti käytössä olevaan hallinta- ja valvontajärjestelmään, kuten SIEM-ratkaisuun. Testaajan rooli ei ole hälytysjärjestelmän omistaminen, vaan sen varmentaminen, että hälytykset tukevat tehokasta monitorointia ja reagointia ja että mahdolliset puutteet tunnistetaan ja raportoidaan vastuullisille tahoille.

Osaa testata monitorointiratkaisujen toimivuuden säännöllisesti

Monitorointivaiheessa testaajan tulee osata toteuttaa säännöllistä testausta ohjelmiston monitorointi- ja valvontaratkaisuille. Tämä edellyttää sen varmentamista, että lokien kerääminen, käsittely ja välittyminen toimivat suunnitellusti myös tuotantoympäristössä. Testaajan osaamiseen kuuluu lisäksi poikkeavien tilanteiden simulointi sen todentamiseksi, että määritellyt hälytykset laukeavat oikein ja tukevat tehokasta reagointia.

Keskeiset toimet

- Varmista, että tietoturvatestaukset ja -skannaukset ajetaan säännöllisesti myös tuotantokäytön aikana.
- Todenna, että kehitys- ja ylläpitoputkissa suoritetaan automatisoidut tietoturvatarkastukset määritellyn riskitason mukaisesti.
- Toteuta tarvittaessa manuaaliset tietoturvatestaukset, kun automaattinen testaus ei ole riittävää.
- Varmista, että muutos- ja riippuvuuspäivitysten yhteydessä suoritetaan tarvittavat testaukset ennen tuotantoon siirtoa.
- Tunnista tuotantotestauksessa ilmenneet tietoturvahaavoittuvuudet ja raportoi ne kehitystiimille viipymättä.
- Arvioi haavoittuvuuksien vaikutus testauksen näkökulmasta ja varmista, että löydökset ohjaavat oikea-aikaisiin korjaustoimiin.
- Varmenna, että lokit kertyvät suunnitellusti ja sisältävät riittävää tietoa poikkeamien havaitsemiseksi.
- Arvioi lokien sisältö testauksen näkökulmasta ja tunnista puutteet, jotka heikentävät monitorointia tai reagointia.
- Testaa, että hälytyssäännöt laukeavat odotetuissa poikkeavissa tilanteissa.
- Varmista, että hälytykset välittyvät sovittuun valvonta- ja hallintajärjestelmään, kuten SIEM-ratkaisuun.
- Toteuta säännöllisiä monitorointiratkaisujen testauksia ja simuloi poikkeamatilanteita.
- Raportoi havaitut puutteet lokituksessa, monitoroinnissa ja hälytyksissä vastuullisille tahoille.

Ohjelmistoturvallisuuden osaamistarpeet rooleittain

Elinkaaren vaihe	Arkkitehti	Ohjelmistokehittäjä	Testaaja
Vaatimusten määrittely	Ymmärtää uhkamallinnuksen tuottamat riskit ja osaa muuntaa ne arkkitehtuuritasoisiksi tietoturvavaatimuksiksi. Hallitsee tietoturvavaatimusten rakenteen ja kattavuuden arvioinnin.	Ymmärtää tietoturvavaatimusten vaikutuksen toteutukseen ja tunnistaa toteutukseen liittyvät riskit. Osaa tulkita uhkamallinnuksen tuloksia ja toteuttaa niihin tarvittavat suojaukset.	Ymmärtää tietoturvavaatimukset ja osaa suunnitella testauksen niiden todentamiseksi.
Suunnittelu	Hallitsee turvallisen arkkitehtuurin suunnitteluperiaatteet ja osaa soveltaa Secure by Design -ajattelua. Ymmärtää käyttöoikeusmallien, rajapintojen ja suojausratkaisujen vaikutukset kokonaisuuteen.	Osaa suunnitella toteutuksen arkkitehtuurin ja tietoturvavaatimusten mukaisesti. Tunnistaa hyökkäyspinnat ja osaa hyödyntää turvallisia kirjastoja ja malleja.	Ymmärtää suunnitteluratkaisujen riskivaikutukset ja osaa kohdentaa testauksen kriittisiin rakenteisiin.
Toteutus	Ymmärtää secure coding -periaatteet ja kykenee arvioimaan niiden toteutumista. Osaa arvioida, toteutuvatko arkkitehtuurissa määritellyt tietoturvakontrollit oikein kooditasolla.	Hallitsee secure coding -käytännöt, syötteiden validoinnin ja salaisuuksien hallinnan. Osaa hyödyntää kehityksen aikaisia tietoturvaskannereita ja tulkita niiden löydöksiä.	Ymmärtää toteutuksen rakenteen ja tunnistaa, mille haavoittuvuuksille ohjelmisto voi altistua.
Rakentaminen	Ymmärtää CI/CD-putken ja rakentamisvaiheen vaikutuksen ohjelmiston turvallisuuteen.	Hallitsee rakentamisvaiheen ja CI/CD-putken tietoturvaperiaatteet, riippuvuuksien hallinnan sekä SBOMin merkityksen.	Ymmärtää rakentamisvaiheen skannausten tarkoituksen ja osaa arvioida niiden tulosten merkityksellisyyttä.
Testaus	Osaa arvioida testauksessa löytyneiden haavoittuvuukwsien vaikutusta arkkitehtuuriin ja tunnistaa rakenteelliset riskit.	Ymmärtää testitulosten teknisen merkityksen ja osaa tehdä tarvittavat korjaukset.	Hallitsee tietoturvatestauksen menetelmät, kuten penetraatio- ja fuzz-testauksen, ja osaa raportoida löydökset selkeästi.
Julkaisu ja käyttöönotto	Ymmärtää, miten arkkitehtuurissa määritellyt tietoturvakontrollit toteutuvat tuotantoympäristössä ja tunnistaa julkaisuun liittyvät riskit. Osaa arvioida, täyttääkö tuotantoon siirtyvä versio arkkitehtuurissa määritellyt tietoturvavaatimukset.	Ymmärtää tuotantoon siirtyvän version eheyteen, konfiguraatioihin, konttikuviin ja salaisuuksien hallintaan liittyvät tietoturvariskit. Osaa arvioida, täyttääkö tuotantoon siirtyvä versio sovitut tietoturvavaatimukset.	Osaa arvioida, täyttääkö tuotantoon siirtyvä versio sovitut tietoturvakontrollit ja onko aiemmin tunnistetut haavoittuvuudet korjattu.
Ylläpito ja monitorointi	Osaa arvioida muutosten ja haavoittuvuuksien vaikutuksen arkkitehtuuriin ja ohjata korjaustoimia. Ymmärtää lokituksen, telemetrian ja valvonnan merkityksen arkkitehtuuritasolla.	Hallitsee päivitysten ja konfiguraatiokovennusten tekniset periaatteet ja osaa tehdä tarvittavat korjaukset. Ymmärtää lokien, hälytysten ja monitorointimekanismien tekniset vaatimukset.	Osaa suunnitella ja toteuttaa säännöllisen tietoturva- ja regressiotestauksen ylläpitovaiheessa. Osaa arvioida lokien riittävyyttä ja testata hälytyssääntöjen toimivuutta.
Käytöstä poisto	Ymmärtää alasajon arkkitehtoniset ja riippuvuuksiin liittyvät riskit. Osaa suunnitella turvallisen alasajon.	Hallitsee teknisen alasajon, tunnistetietojen poistamisen ja datan hävittämisen periaatteet.	Ymmärtää käytöstä poiston tietoturvariskit ja osaa todentaa kontrollien toimivuuden.

Lisätietoja

Seuraavista lähteistä on saatavilla lisätietoja tässä oppaassa kuvatuista osaamistarpeista.

- **Cybersecurity and Infrastructure Security Agency (CISA)**
  Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Secure by Design Software
  https://www.cisa.gov/sites/default/files/2023-10/SecureByDesign_1025_508c.pdf

- **ISO/IEC**
  ISO/IEC 27034 Information technology – Security techniques – Application security
  https://www.iso.org/standard/44378.html

- **National Institute of Standards and Technology (NIST)**
  Secure Software Development Framework (SSDF), SP 800-218
  https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf

- **Open Worldwide Application Security Project (OWASP)**
  API Security Top 10
  https://owasp.org/www-project-api-security
  Application Security Verification Standard (ASVS)
  https://owasp.org/www-project-application-security-verification-standard
  Developer Guide
  https://devguide.owasp.org
  Software Assurance Maturity Model SAMM
  https://owasp.org/www-project-samm
  Top 10 Web Application Security Risks
  https://owasp.org/www-project-top-ten
  TOP 10 Proactive Controls
  https://top10proactive.owasp.org
  Web Security Testing Guide
  https://owasp.org/www-project-web-security-testing-guide

Liikenne- ja viestintävirasto Traficom
Kyberturvallisuuskeskus
Puhelin: 029 534 5000 (vaihde)
PL 313 (Erik Palménin aukio 1)
00561 Helsinki

The Finnish Transport and Communications Agency Traficom
National Cyber Security Centre Finland (NCSC-FI)
Phone: +358 29 534 5000 (switchboard)
P.O. Box 313 (Erik Palménin aukio 1)
FI-00561 Helsinki

Huoltovarmuuskeskus

TRAFICOM
Liikenne- ja viestintävirasto
Kyberturvallisuuskeskus