LAPIN AMK
Lapland University of Applied Sciences

OHJELMISTOTUOTANNON TIETOTURVA

Niko Karppinen

Opinnäytetyö
Tietotekniikan koulutusohjelma
Insinööri (AMK)

2014

LAPIN AMMATTIKORKEAKOULU

TEKNIIKKA JA LIIKENNE

Tietotekniikan koulutusohjelma

OHJELMISTOTUOTANNON TIETOTURVA

2014

Toimeksiantaja Lapin ammattikorkeakoulu

Tekijä Niko Karppinen

Hyväksytty 1.10.2014

Työ on luettavissa Theseus-verkkokirjastossa.

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön
tiivistelmä

Tekniikka ja liikenne
Tietotekniikan koulutusohjelma

Tekijä Niko Karppinen Vuosi 2014

Toimeksiantaja Lapin ammattikorkeakoulu
Työn nimi Ohjelmistotuotannon tietoturva
Sivu- ja liitemäärä 41 + 1

Opinnäytetyön aiheena oli tutkia tietoturvaa ohjelmistotuotannossa. Pääsiallisena ongelmana oli selvittää, miksi tietoturvaan ei kiinnitetä tarpeeksi huomiota ohjelmistotuotannossa. Tavoitteena oli selvittää, millaisia menetelmiä on tarjolla turvalliseen ohjelmistotuotantoon sekä löytää ratkaisukeinoja, joilla ohjelmistoyritykset saadaan huomioimaan tietoturva enemmän omassa ohjelmistotuotannossaan.

Työssä tarkasteltiin tietoturvan tarvetta, ongelmia ja haasteita sekä perehdyttiin Microsoftin turvallisen ohjelmistotuotannon elinkaareen, jonka avulla voidaan tuottaa turvallisia ohjelmistoja ja sovelluksia. Lopuksi pohdittiin tietoturvan tilaa sekä tietomurtojen kehitystä suhteessa siihen kuinka paljon tietoturva otetaan ohjelmistotuotannossa huomioon.

Tutkimukselle asetetut tavoitteet saavutettiin hyvin. Tutkimuksessa saatiin selville kustannusten ja ajattelumallien vaikutus tietoturvan laiminlyöntiin. Tutkimus tarjoaa myös yleiskatsauksen turvallisen ohjelmistotuotannon elinkaareen sekä kertoo, miksi tietoturvaa tarvitaan ja kuinka ohjelmistoyritykset saadaan huomioimaan se paremmin. Tutkimus antoi hyvän käsityksen tietoturvan tilasta sekä siitä, miksi siihen ei kiinnitetä tarpeeksi huomiota.

Avainsanat Microsoft, tietoturva, ohjelmistotuotanto, turvallinen ohjelmistotuotanto

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

School of Technology, Communication and Transport
Information Technology Programme

Author Niko Karppinen Year 2014

Commissioned by Lapland University of Applied Sciences
Subject of thesis Security in Software Development
Number of pages 41 + 1

The subject of this thesis was to research security in software development. The goal was to examine why software development companies do not pay more attention to security. The aim was to examine why security is needed and what kinds of problems and challenges it contains regarding software development. The aim was also to find out what kinds of methods are available to develop more secure software.

The research was done by examining Microsoft security development life cycle which can be used to develop more secure software and applications. The current state of information security and the evolution of security incidents were also examined. The theoretical knowledge was used to find solutions why software development companies do not pay more attention to security.

The result of the research was an overview of security development life cycle and the reasons why companies do not pay more attention to security and why security is needed. The result of the research helped to gain a better understanding of the information technology security.

Key words Microsoft, security, software development, security development life cycle

SISÄLTÖ

KUVIOLUETTELO

KÄSITELUETTELO

Autentikointi	Identiteetin varmennus
Denial of Service (DoS)	Palvelunestohyökkäys
PII	Henkilökohtaista tunnistettavaa tietoa (Personally identifiable information). Tietoa, jonka avulla henkilö voidaan tunnistaa, paikantaa tai häneen voidaan ottaa yhteyttä.
Secure feature	Järjestelmän ominaisuus, joka on turvallinen
Security feature	Järjestelmää tukeva tietoturvaominaisuus
STRIDE	Microsoftin kehittämä tietoturvauhkamalli, jota käytetään uhkien kartoituksessa ja mallinnuksessa.
Vesiputousmalli	Perinteinen ohjelmistotuotantoprosessin vaihejakomalli

1 JOHDANTO

Tietoturvan merkitys ohjelmistotuotannossa on noussut tärkeäksi puheenaiheeksi, koska tietomurtojen määrä on lisääntynyt viime vuosina. Median mielenkiinto on lisääntynyt tietoturvaan liittyvien tietomurtojen uutisoinnin takia. Aiemmin yritykset keskittyivät tuottamaan ohjelmistoja ja myymään ne asiakkaille, mutta tietoturvaan ei ole kiinnitetty suurta huomiota. Nykyään ohjelmistoyritykset tuottavat ohjelmistoja ja sovelluksia eri alustoille, jotka sisältävät käyttäjien arkaluontoisia tietoja. Tämän takia ohjelmistoyritysten tulisi tuottaa turvallisia ohjelmistoja. Tietomurrot tuottavat yrityksille taloudellisia tappioita ja aiheuttavat imagon tahraantumista. Tämän takia ohjelmistoyritysten olisi panostettava turvallisempaan ohjelmistokehitykseen.

Opinnäytteen pääasiallisena selvitettävänä ongelmana oli tutkia, miksi tietoturvaan ei kiinnitetä tarpeeksi huomiota ohjelmistotuotannossa. Tavoitteena oli myös selvittää, millaisia menetelmiä riittävän turvallisen ohjelmiston luomiseksi on tarjolla. Yritin löytää työssäni vastaukset näihin kysymyksiin.

Opinnäytteessä perehdyttiin tietoturvan tarpeeseen ohjelmistotuotannossa. Työssä käytiin läpi, millaisia haasteita ja ongelmia tietoturvan suunnittelussa ja toteuttamisessa on. Seuraavassa vaiheessa perehdyttiin Microsoftin turvallisen ohjelmistokehityksen elinkaareen, jossa käytiin läpi riittävän tietoturvan integrointi osaksi ohjelmistotuotantoa. Lopuksi pohdittiin tietoturvan nykytilaa ja tietomurtojen luonteen kehitystä viimeisen kymmenen vuoden aikana. Teorian pohjalta pyrittiin vastaamaan kysymyksiin, miksi tietoturvaan ei ole kiinnitetty tarpeeksi huomiota ja miten sitä voitaisiin parantaa.

Opinnäytetyön aiheen valinta perustui Itävallassa vaihto-oppilaana suorittamiini tietoturvaopintoihin, joiden pohjalta kiinnostus tietoturvaa kohtaan on kasvanut. Halusin yhdistää opinnäytetyössäni tietoturva- ja ohjelmistotekniikkaopinnoistani saamani tiedon sopivaksi kokonaisuudeksi, jossa molemmat osa-alueet otetaan huomioon ja yritetään selvittää tietoturvaan liittyvä ongelma. Ohjelmistotuotannon tietoturva ja etenkin

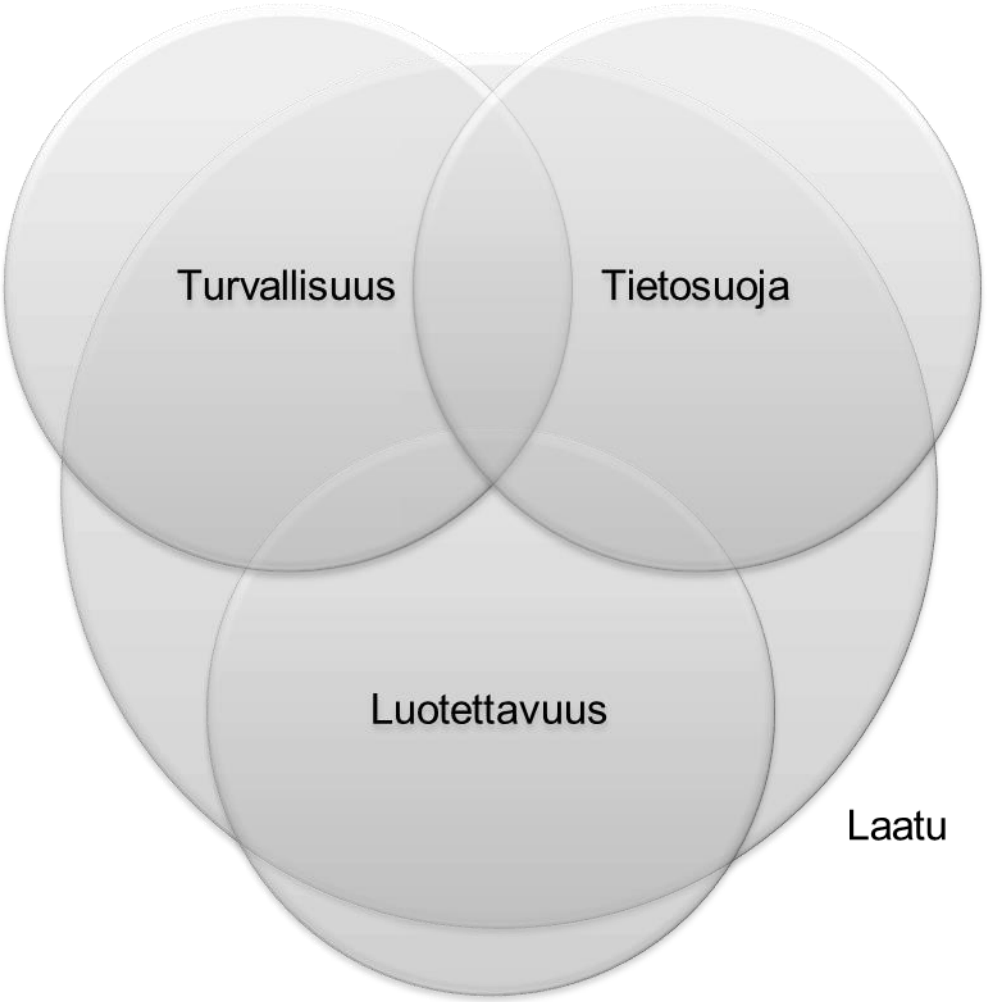
ohjelmistoyritysten asiakkailleen tarjoamat turvalliset ohjelmistot ovat ajankohtainen aihe, johon kaivataan vastauksia.

2 TIETOTURVA OHJELMISTOTUOTANNOSSA

2.1 TIETOTURVAN TARVE

Elämme aikaa, jossa erilaiset laitteet ovat yhä enemmän yhteydessä keskenään. Tietokoneet, palvelimet, taulutietokoneet, älypuhelimet ja erilaiset sulautetut järjestelmät luovat verkon, jossa ne voivat kommunikoida keskenään. Tämä luo mahdollisuuksia ohjelmistokehittäjille ja liiketoiminnan harjoittajille kehittää mitä innovatiivisimpia järjestelmiä. Kehitys tuo mukanaan myös uhkakuvia. Järjestelmät ovat haavoittuvia hyökkäyksille ja tämän takia järjestelmät tulisi suunnitella ottamalla tietoturva huomioon ohjelmistotuotantoprosessissa. (Howard–Leblanc 2004, 3.)

Suunniteltaessa turvallisia järjestelmiä on hyvä muistaa, että turvallisuus on osa laadunhallintaa. Ohjelmistotuotannossa saadaan aikaan vakaampaa koodia, kun suunnittelu ja toteutus on tehty turvallisuus ensisijaisena tavoitteena. Järjestelmään jälkikäteen lisätty turvallisuus ei ole yhtä kattavaa. Käyttäjien ja median arvostus turvallista tuotetta kohtaan on tärkeää. Media ei arvostele turvallista tuotetta. Käyttäjät ovat tyytyväisiä tuotteeseen, jossa heidän arkaluontoiset tietonsa ovat turvassa. Jos tuote on suunniteltu turvalliseksi, sitä on aiempaa edullisempaa korjata ja ylläpitää. Täydellistä ohjelmistoa ei kuitenkaan ole olemassa, eikä myöskään täysin turvallista ohjelmistoa. Turvallisin järjestelmä on sammutettu ja lukittu turvalliseen paikkaan, mutta silloinkaan se ei ole täydellisen turvallinen. Turvallinen ohjelmisto on tarpeeksi turvallinen siinä ympäristössä, missä sitä käytetään. Turvallisuus on osa laadukasta tuotetta. Turvallinen ohjelmisto on yritykselle etu muihin kilpailijoihin nähden. Turvallisuuden tarve ja turvallisuuteen panostaminen ohjelmistotuotannossa riippuu ympäristöstä ja ohjelmiston sisällöstä. (Howard–Leblanc 2004, 4–7.) Kuvio 1 osoittaa, millainen merkitys turvallisuudella ja tietosuojalla on laadukkaassa tuotteessa.



Kuvio 1. Suhde laadun, tietosuojan, turvallisuuden ja luotettavuuden välillä (Howard–Lipner 2006, 10)

2.2 TIETOTURVAN ONGELMAT JA HAASTEET

Asiakkaat tilaavat ohjelmistoyrityksiltä ohjelmistoja ja sovelluksia. Asiakkaat tahtovat sovelluksensa mahdollisimman nopeasti ja halpaan hintaan. Asiakas on valmis käyttämään kohtuullisen summan rahaa sovellukseen ja haluaa parasta laatua. Ohjelmistoyritykset käyttävät paljon resursseja toimivan sovelluksen rakentamiseen mahdollisimman nopeasti ja taloudellisesti, eivätkä kiinnitä tarpeeksi huomiota tietoturvaan. Tietoturvaan panostaminen luo turvallisia ohjelmistoja, mutta kasvattaa myös ohjelmiston rakentamiseen käytettyä budjettia huomattavasti.

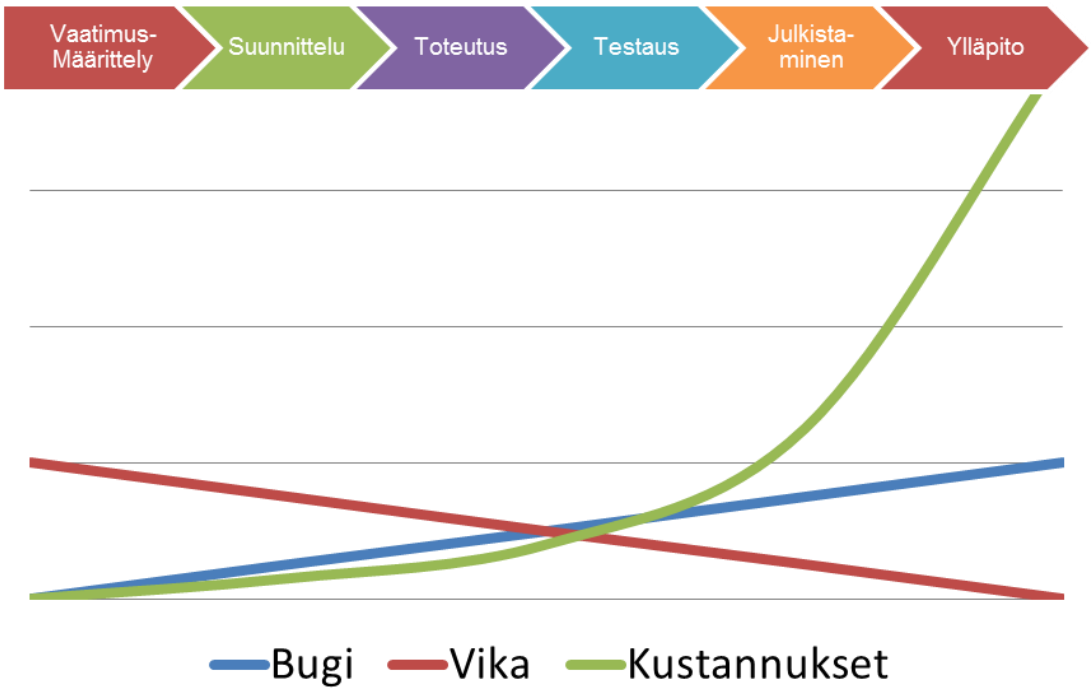
Ohjelmistoissa on nykyään paljon sisältöä ja ne ovat tekniseltä rakenteeltaan aiempaa monimutkaisempia kuin ne olivat ennen. Uusia teknologioita on

tullut lisää ja niitä käytetään paljon rinnakkain, mikä luo kompleksisuutta ohjelmistoihin. Ohjelmistot toimivat myös eri järjestelmissä. Ohjelmistojen kompleksisuus, uudet teknologiat ja erilaiset järjestelmät aiheuttavat paljon haasteita tietoturvalle. (Bart De Win 2013, 4.)

Kuvio 2 kuvaa ohjelmiston elinkaaren eri vaiheissa ilmeneviä virheiden (Bug), vikojen (Flaw) ja kustannuksien määrää suhteessa toisiinsa. Virhe tarkoittaa ohjelmiston virhettä, joka vaikuttaa haitallisesti ohjelmiston toimivuuteen. Virheet aiheutuvat yleisimmin ohjelmiston elinkaaren toteutusvaiheessa ohjelmointivirheen takia. Vika tarkoittaa suurempaa ohjelmiston loogiseen toimintaan vaikuttavaa virhettä. Vika on tapahtunut arkkitehtuuritasolla suunnittelussa ja se vaikuttaa ohjelmiston avulla suoritettavaan liiketoimintaan. Esimerkkinä voidaan pitää elokuvateatterin paikanvarausjärjestelmää. Asiakkaat voivat varata paikan elokuvateatterista, mutta heidän ei ole pakko maksaa paikkaa. Tämä haittaa liiketoimintaa, koska paikat voivat olla varattuna täyteen, mutta kukaan ei ole maksanut paikoista. Tällöin puhutaan viasta, joka vaikuttaa liiketoimintalogiikkaan. Kustannukset kuvaavat virheiden ja vikojen korjauksesta aiheutuneita kustannuksia ohjelmiston elinkaaren eri vaiheissa. (Bart De Win 2013, 5.)

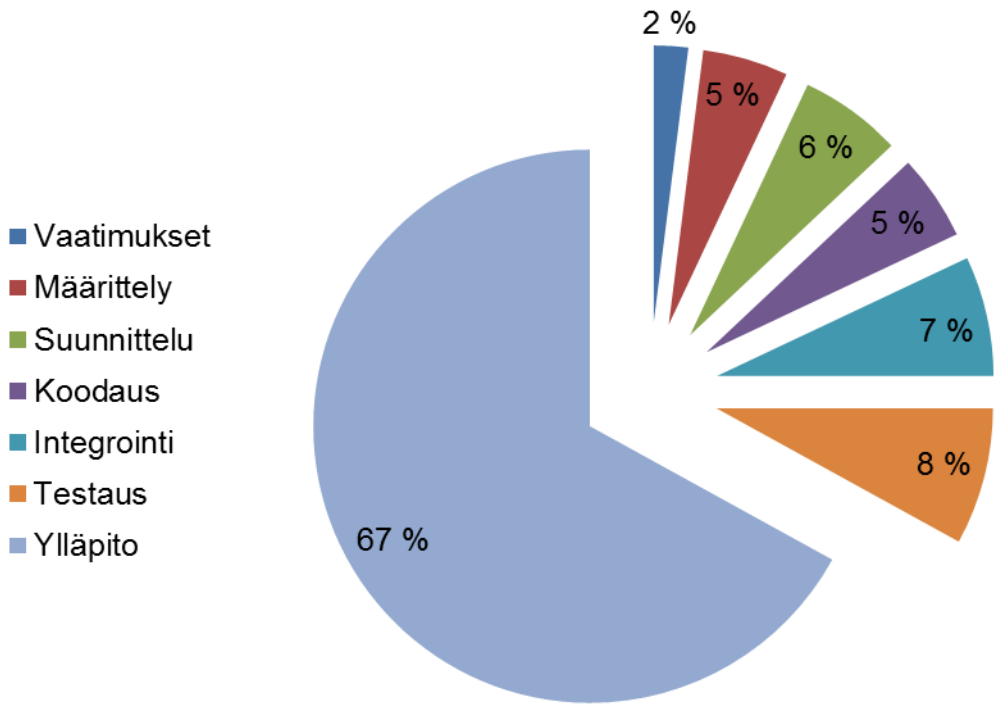
Riski ohjelmiston virheiden määrästä ohjelmiston elinkaaren alussa on pieni, koska ohjelmisto on suunnitteluvaiheessa eikä sitä vielä ole aloitettu toteuttamaan. Suunnittelussa täytyy kuitenkin kiinnittää huomiota teknologiavalintoihin ja ohjelmiston kompleksisuuteen, jotta virheitä voidaan ennaltaehkäistä. Toteutusvaiheen jälkeen virheiden riski kasvaa. Ohjelmistokehittäjien virheiden ja testauksen huolimattomuuden takia virheitä jää ohjelmistoon. Ohjelmiston kompleksisuus kasvattaa myös virheiden riskiä. Arkkitehtuuritasolla ohjelmistoa suunnitellessa on riski, että se suunnitellaan väärin. Looginen virhe ohjelmistossa on hankala korjata. Suunnitteluviat korostuvat ohjelmiston elinkaaren alussa ennen toteutusvaihetta, jonka jälkeen riski vian syntymiselle vähenee. Virheiden ja vikojen korjauksen aiheuttamat kustannukset kasvavat testausvaiheen jälkeen erittäin nopeasti. Virheiden korjauksissa joudutaan ohjelmiston elinkaaressa palaamaan takaisin toteutusvaiheeseen, jossa virhe korjataan. Tämän jälkeen uusi koodi testataan, ennen kuin se voidaan uudelleen

julkaista. Erityisesti virheiden korjaus jo julkaistuun ohjelmistoon voi aiheuttaa uusia virheitä. Vikojen korjaus tulee kustannusten osalta erittäin kalliiksi, koska silloin joudutaan palaamaan takaisin suunnitteluvaiheeseen. Liiketoimintalogiikkaa joudutaan suunnittelemaan uudestaan, jonka jälkeen se toteutetaan ja testataan ennen julkaisua. (Bart De Win 2013, 5.)



Kuvio 2. Ohjelmiston tietoturvan luonne (Bart De Win 2013, 5)

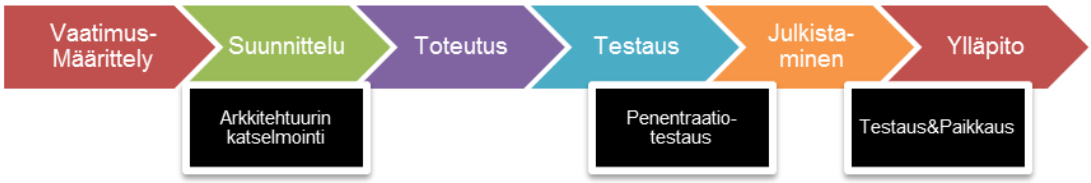
Ohjelmistotuotantoprosessin kustannusten jakautuminen on kuvattu tarkemmin kuviossa 3. Se kuvaa perinteisen vesiputousmallin kustannusten jakautumista sen eri vaiheissa. Tämä on tyypillinen esimerkkitapaus, koska ohjelmiston elinkaaren kustannukset ovat tapauskohtaisia (Haikala–Märijärvi 2004, 56). Kuten kuviosta 3 voidaan havaita, ylläpito kattaa suurimman osan prosessin kustannuksista. Muut vaiheet jakautuvat tasaisesti, mutta kustannusten määrä on prosessin edetessä nousujohteinen.



Kuvio 3. Kustannusten jakautuminen ohjelmiston elinkaaressa (Haikala–Märijärvi 2004, 57)

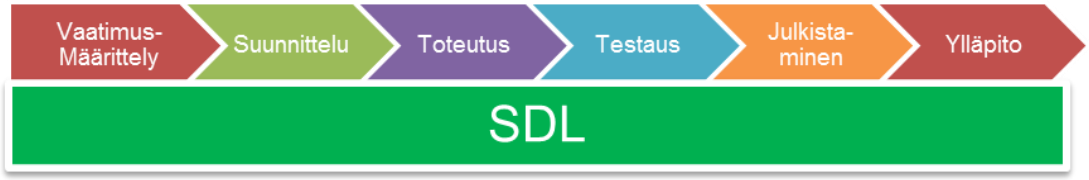
Kuvio 4 kuvaa nykyisin yleisimmin käytössä olevaa mallia, jolla pyritään varmistamaan ohjelmiston turvallisuus. Arkkitehtuurin katselmointiin (Architecture review) kiinnitetään huomiota jossain määrin, mutta tällainen katselmointi ei ole kovin yleistä. Arkkitehtuurin katselmoinnissa arvioidaan ohjelmiston arkkitehtuuria, ja hyvän arkkitehtuurin avulla parannetaan ohjelmiston laatua. Kuten kuvasta voidaan huomata, niin sanottu haavoittuvuus- ja murtotestaus (Penetration Testing) ajoittuu juuri ennen julkaisua, jolloin kustannusten näkökulmasta korjaus tulee olemaan kallis. Haavoittuvuus- ja murtotestauksessa testaaja pyrkii löytämään haavoittuvuuksia hyökkäämällä järjestelmään. Ylläpitovaiheessa ohjelmistoa testataan ja paikataan (Penetrate & Patch) tarvittaessa, kun ongelmia ilmenee. Tietoturvan huomioon ottaminen painottuu tässä mallissa todella myöhäiseen vaiheeseen ohjelmiston elinkaarta, jolloin haavoittuvuuksien riski kasvaa ja kustannukset ovat korkeat. Tietoturvan näkökulmasta ajateltuna ennen julkaisua ja ylläpitovaiheeseen painottuva testaus ja haavoittuvuuksien

korjaus on riski, koska julkaistu ohjelmisto on yleisessä käytössä. Tämä luo hakkereille ja tietojen kalastelijoille mahdollisuuden päästä haavoittuvuuksien kautta sisään järjestelmään. (Bart De Win 2013, 6.)



Kuvio 4. Nykyinen tietoturvan käytännön toteutus ohjelmistotuotannossa (Bart De Win 2013, 6)

Tietoturva ei ole ominaisuus, jonka voi lisätä ohjelmistoon jälkikäteen. Monet ohjelmistokehittäjät huomioivat tietoturvan vasta ohjelmiston julkaisun jälkeen, kun jotain on hajonnut tai tietoja on varastettu. Tietoturva täytyy sisällyttää ohjelmiston elinkaareen, jolloin tietoturva otetaan huomioon ohjelmistoprojektin jokaisessa vaiheessa. Näin voidaan arvioida, määrittää ja testata ohjelmistoon liittyvää tietoturvaa kehityksen eri vaiheissa. Tämä tuo ohjelmistotuotantoon lisää kustannuksia, mutta riski tietomurtoon ja sitä kautta aiheutuneisiin kustannuksiin pienenee olennaisesti. Tärkeintä on, että yritys luo turvallisia ohjelmistoja, jolloin asiakkaat ovat vakuuttuneita siitä, että heidän arkaluontoiset tiedot ovat turvassa. Tietoturva on ohjelmiston koko elinkaaren mittainen prosessi.



Kuvio 5. Tietoturvan sisällyttäminen ohjelmistokehitysprosessiin (Bart De Win 2013, 7)

3 MICROSOFT SECURITY DEVELOPMENT LIFECYCLE (SDL)

Turvallisen ohjelmistokehityksen elinkaari (SDL) on tietoturvaprosessi, jolla pyritään tuottamaan turvallinen ohjelmisto ohjelmistotuotantoprosessin rinnalla. Kokonaisvaltaisella ja käytännöllisellä lähestymistavalla SDL-prosessin tavoitteena on vähentää ohjelmistojen haavoittuvuuksien määrää ja vakavuutta. Prosessi ottaa tietoturvan ja tietosuojan huomioon ohjelmistokehityksen kaikissa vaiheissa. SDL-prosessin ydin koostuu kolmesta konseptista. Nämä ovat koulutus, jatkuva prosessin kehittäminen ja vastuullisuus. Jatkuva teknisen työn roolien koulutus ja valmennus ohjelmistokehitysryhmän sisällä ovat erittäin tärkeää. Organisaatioiden tulisi vaihtaa tietoja keskenään ja pysyä ajan tasalla teknologioista ja uusista uhkakuvista. Tietoturvauhat eivät ole staattisia, joten SDL-prosessissa painotetaan selkeästi ymmärtämään tietoturvaheikkouksien syyt ja seuraukset ja vaaditaan SDL-prosessien säännöllistä arviointia ja muutosten aikaansaamista vastauksena uudelle teknologiakehitykselle tai uusille uhille. (Microsoft 2014b, 4.)

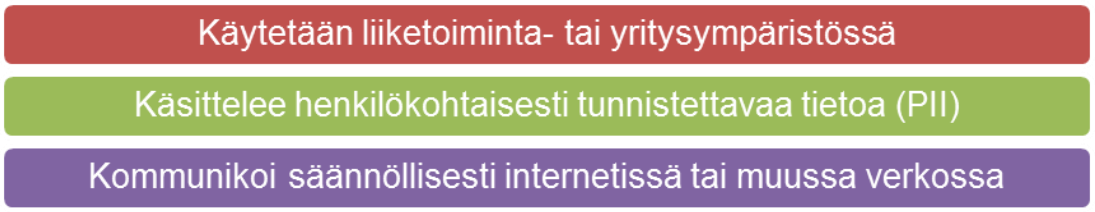
Turvallisen ohjelmistotuotantoprosessin integrointi olemassa olevaan ohjelmistotuotantoprosessiin voi olla haastavaa ja kallista, jos se tehdään väärin. Integroinnin onnistuminen riippuu monesta tekijästä, kuten organisaation koosta, resursseista ja johdon tuesta. Näitä tekijöitä voidaan ohjata ymmärtämällä hyvät turvallisen kehityksen toimintatavat ja täytäntöönpanon painopisteet. SDL-optimointimalli auttaa osoittamaan tällaiset asiat. (Microsoft 2014b, 4.)

SDL-optimointimalli koostuu viidestä vaiheesta, jotka muistuttavat karkeasti ohjelmistotuotantoprosessin vaiherakennetta. Nämä vaiheet ovat koulutus ja organisaation resurssit, määrittely ja suunnittelu, toteutus, vahvistus, julkistus ja vastatoimet. Vaiheet muistuttavat vesiputousmallin vaiherakennetta. SDL-optimointimalli määrittää toimintatapojen ja kapasiteetin tasoa neljällä eri alueella. Nämä tasot ovat perustaso, standardisoitu, edistynyt ja dynaaminen taso. Optimointimalli etenee perustasolta dynaamiselle tasolle. Perustasolla SDL-prosessi on otettu huomioon vähän tai ei ollenkaan. Prosessit, koulutus ja työkalujen käyttö on vähäistä tai työkalujen käyttöä ei ole ollenkaan.

Dynaamisella tasolla on edetty siihen pisteeseen, että SDL-prosessia noudatetaan täydellisesti koko organisaation tasolla. Täydelliseen SDL-prosessin noudattamiseen sisältyvät tehokkaat ja toimivat prosessit, korkeasti koulutettu henkilökunta, tehokas työkalujen käyttö ja voimakas vastuullisuus organisaation eri osa-alueilla. (Microsoft 2014b, 4.)

Kuvio 6. SDL-optimointimalli (Microsoft 2014b, 5)

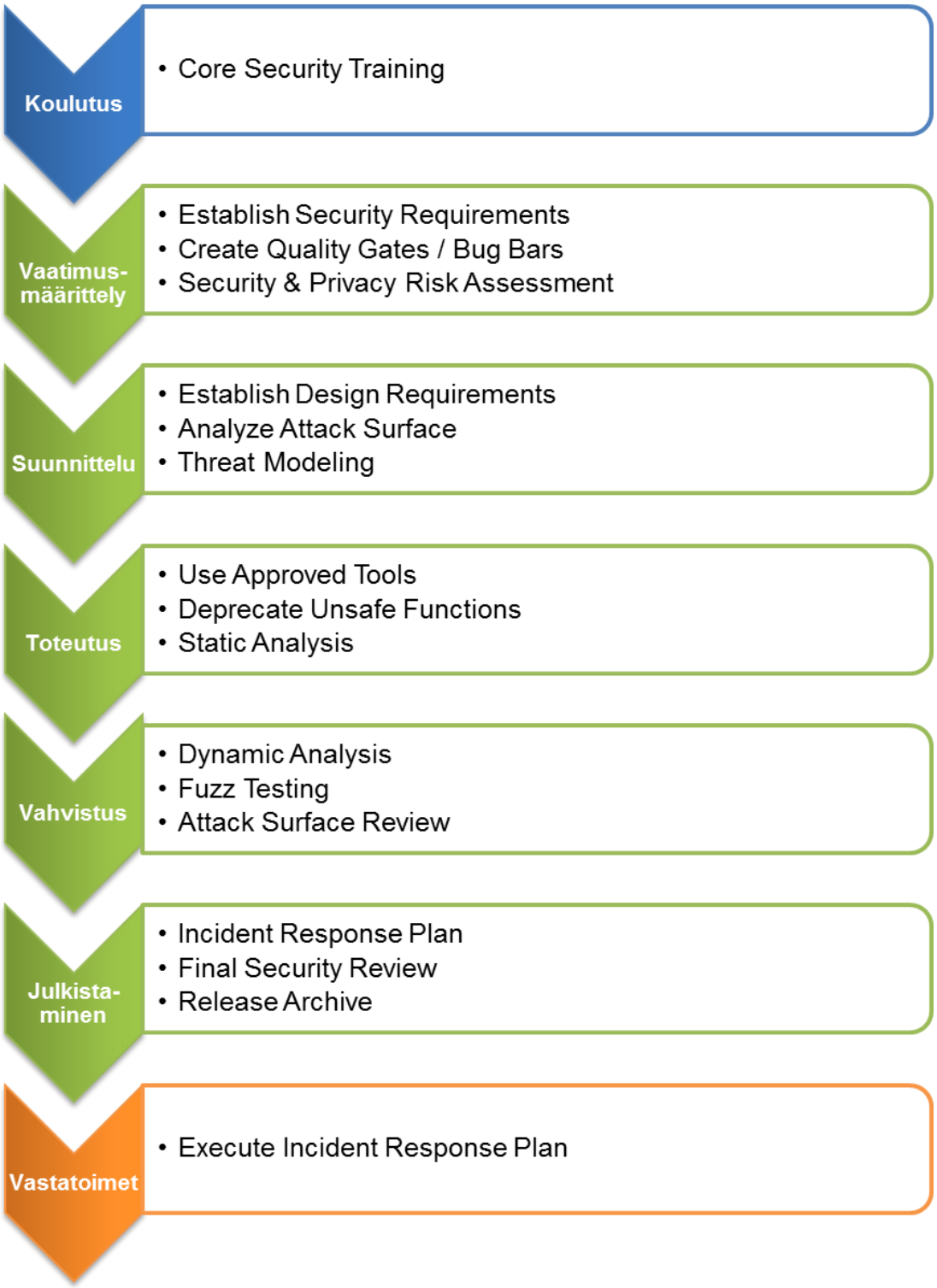
Organisaatioiden tulisi asettaa selkeät ohjeet, millaisille projekteille SDL-prosessia lähdetään toteuttamaan. SDL-prosessi tulisi toteuttaa, jos projekti tehdään liike- tai yritystoimintaympäristöön, se sisältää henkilökohtaisesti tunnistettavaa tietoa tai jotain muuta arkaluontoista tietoa tai se kommunikoi säännöllisesti internetissä tai jossain muussa verkossa. Kun huomioimme tietoteknisten järjestelmien ja tietoturvauhkien yleisyyden, on helpompi tunnistaa sellaiset ohjelmistokehitysprojektit, jotka eivät tarvitse SDL-prosessia kuin projektit, jotka tarvitsevat sitä. (Microsoft 2014b, 5.)



Kuvio 7. SDL-prosessin suorittamiseen vaaditut kriteerit (Microsoft 2014b, 5)

Yksinkertaisesti Microsoftin SDL-prosessi on kokoelma turvatoimenpiteitä, joilla saavutetaan turvallinen ohjelmisto. Toimenpiteet on järjestelty perinteisen ohjelmistotuotantoprosessin (vesiputousmalli) mukaan, jolloin SDL-prosessin käyttöönotto on helpompi toteuttaa. Toimenpiteet toimivat myös erillisenä prosesseina, mutta Microsoft on osoittanut, että osana ohjelmistotuotantoprosessia turvatoimet johtavat hyviin tuloksiin. SDL-prosessissa on myös vaihtoehtoisia turvatoimenpiteitä, joita voi lisätä osaksi prosessia projektin luonteen ja tarpeen vaatiessa. Organisaation tulisi keskittyä laatuun ja kattavuuteen SDL-prosessin kaikissa vaiheissa. Organisaatioilta odotetaan tietynlaista kehittyneisyyttä toimiessaan SDL-optimointimallin edistyneellä tai dynaamisella tasolla. Tämä tarkoittaa sitä, että organisaatio kehittää toimivat työskentelymallit. Ei ole tärkeää, onko esimerkiksi uhkamallinnus toteutettu erityistä työkalua käyttäen vai perinteisin projektityöskentelymenetelmin. Microsoftin SDL-prosessin oikea arvo on kattavissa ja tarkoissa tuloksissa. (Microsoft 2014b, 6–7.)

Jos ohjelmistoprojekti sisältää ominaisuuksia, jotka vaativat SDL-prosessin suorittamista, ohjelmistokehittäjien on käytävä läpi pakollinen 16-vaiheinen toimenpidesarja noudattaakseen Microsoftin SDL-prosessia. Tietoturva- ja tietosuoja-asiantuntijat ovat tunnustaneet pakolliset toimenpiteet tehokkaiksi ja niiden tehokkuutta tarkistetaan ja arvioidaan säännöllisesti osana vuosittaista arviointiprosessia. Ohjelmistokehittäjien tulee muistaa vaihtoehtoisien turvatoimenpiteiden olemassaolo, mutta 16 vaihetta tulee aina suorittaa, jotta vaatimustasoa noudatetaan. (Microsoft 2014b, 7.) Vaiheet voidaan kuvata myös havainnollisessa muodossa vuokaaviona (Liite 1).



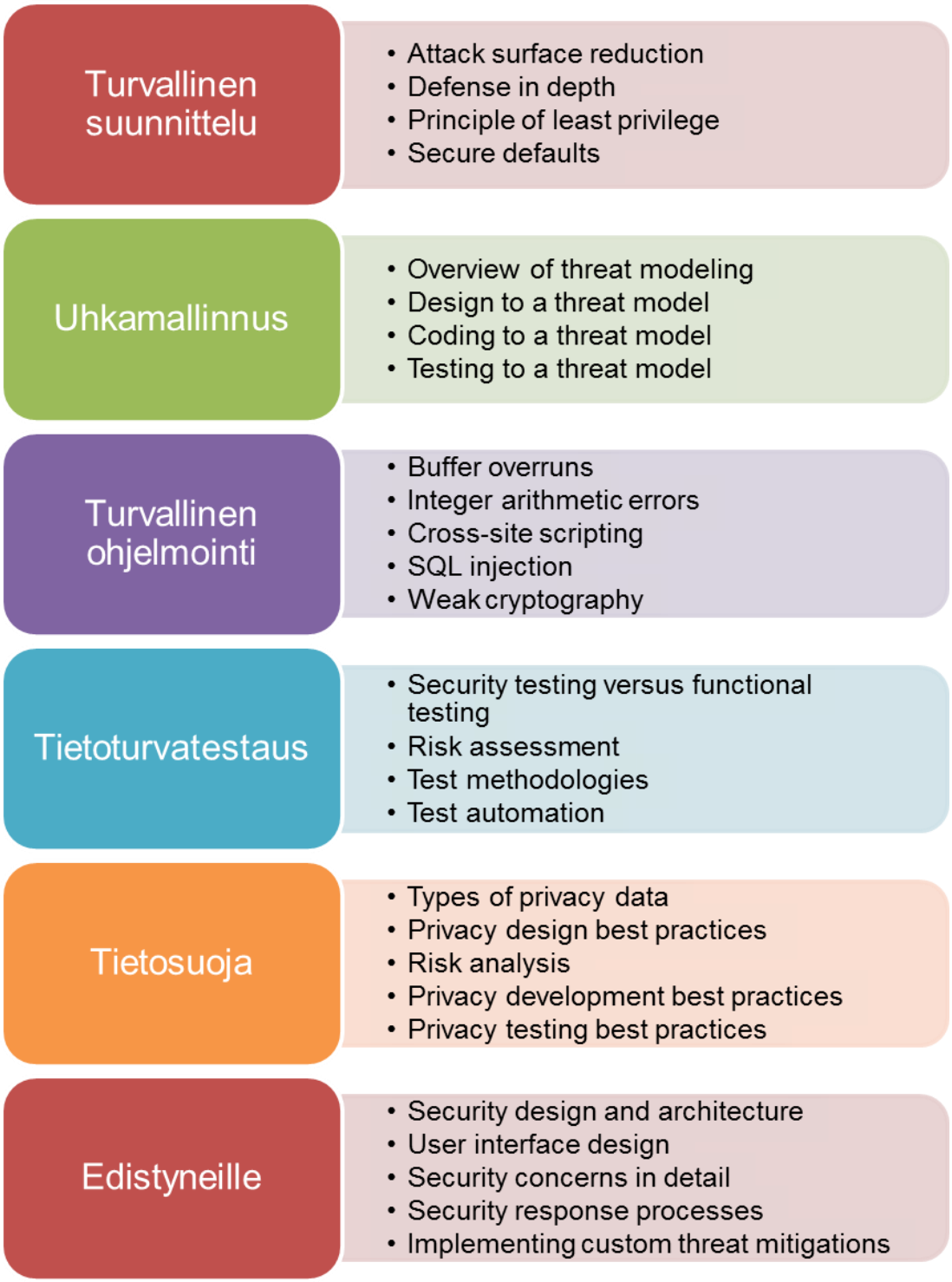
Kuvio 8. Microsoftin turvallisen ohjelmistokehitysprosessin malli (Microsoft 2014a, 11)

3.1 KOULUTUS

Ohjelmistokehittäjillä tulisi olla riittävästi tietoa tietoturvan perusteista, yleisistä tietoturvan vikatyypeistä, tietoturvasuunnittelusta tai

tietoturvatestauksesta, sillä ilman näitä ominaisuuksia ei ole mahdollisuuksia tuottaa turvallisia ohjelmistoja. Tavallisesti ohjelmistokehittäjät tietävät suhteellisen vähän ohjelmistojen tietoturvasta. Yritykset, jotka tahtovat tuottaa turvallisempia ohjelmistoja, on panostettava työntekijöidensä tietoturvakoulutukseen. (Howard–Lipner 2006, 55.)

Kaikkien ohjelmistotuotantoprosessissa mukana olevien jäsenten tulisi saada riittävä koulutus tietoturvan perusteista ja tietoa uusimmista tietoturvaan liittyvistä suuntauksista, haavoittuvuuksista ja yksityisyydensuojasta. Tietoturvakoulutuksen avulla pyritään takaamaan turvallinen ohjelmistotuotanto. Ohjelmistokehittäjien tulisi osallistua vähintään yhteen tietoturvakoulutukseen vuodessa, jotta riittävä, turvalliseen ohjelmistotuotantoon liittyvä ammattitaito pysyy yllä. Yrityksen on luotava oma ohjelmansa, jolla ylläpidetään työntekijöiden tietoturvaosaamista. Yritys voi tukea työntekijöitänsä esimerkiksi tietoturvasertifikaattien avulla. Työntekijöitten suorittamat sertifikaatit ovat samalla näyttö osaamisesta, työkokemuksesta ja siitä, että työntekijä kouluttaa itseään jatkuvasti. Yritys hyötyy tästä myös, sillä työntekijöiden tietoturvaosaaminen on ajan tasalla, ja yritys voi kehittää entistä turvallisempia ohjelmistoja. (Howard–Lipner 2006, 58–65.)



Kuvio 9. Tietoturvakoulutuksen kulmakivet (Microsoft 2014a, 14–15)

Tietoturvakoulutus ja tietoisuus tietoturvan suuntauksista ovat erittäin tärkeitä, kun tuotetaan tietoturvallisia ohjelmistoja. Ohjelmistoteollisuus on tänä päivänä vielä liikaa keskittynyt vain tuottamaan ohjelmistoja ja jättää tietoturvan vähälle huomiolle. Yliopistojen ja ammattikorkeakoulujen tulisi

tarjota koulutusohjelmissaan enemmän kursseja, jotka keskittyvät tietoturvaan ja tietosuojaan. (Howard–Lipner 2006, 65.)

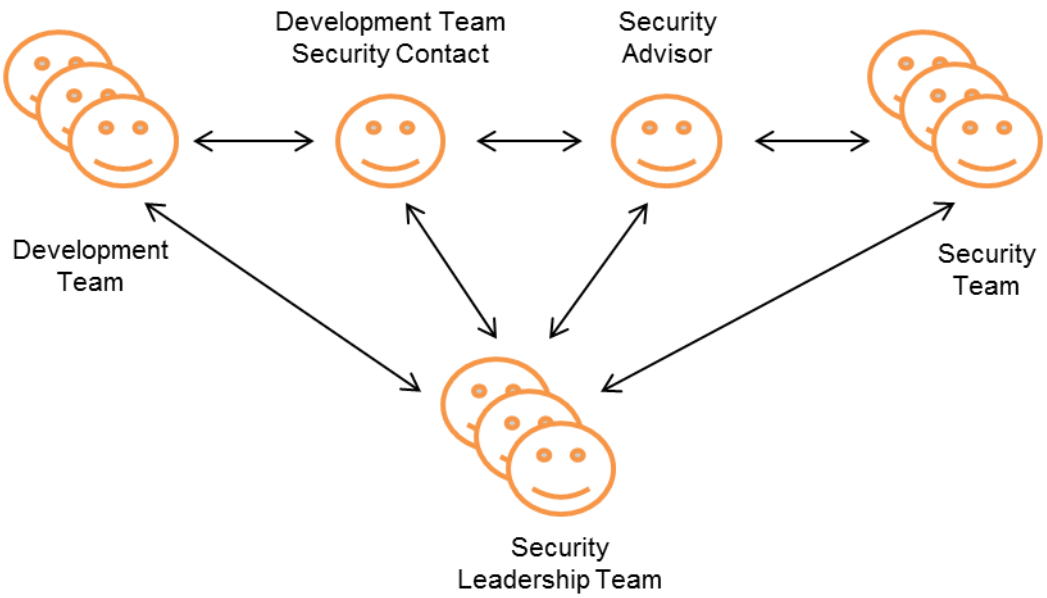
3.2 VAATIMUSMÄÄRITTELY

3.2.1 Tietoturvamääritykset

Tietoturvamäärityksien luomisen optimaalinen ajankohta on mahdollisimman varhaisessa ohjelmistoprojektin suunnitteluvaiheessa. Varhainen tietoturvan määrittäminen antaa ohjelmistokehittäjille avaimet tunnistaa tärkeät vaiheet ja toimituksen ajankohta. Tietoturvan ja tietosuojan integrointi mahdollistaa myös häiriöiden minimoimisen suunnitelmassa ja aikataulussa. Tietoturva- ja tietosuojamääritysanalyysi suoritetaan projektin aloitusvaiheessa (Project Inception). Siinä on määritetty järjestelmän tietoturvan minimivaatimukset siihen ympäristöön ja määrityksiin, mihin järjestelmää suunnitellaan sekä käyttöohjeet tai seurantajärjestelmä haavoittuvuuksien varalle. (Microsoft 2014b, 8.)

Tietoturvamääritykset on hyvä aloittaa heti projektin alussa, jotta projekti saadaan käyntiin tietoturvan osalta ja saadaan aikaiseksi turvallinen ohjelmisto. Ensimmäisenä harkitaan tietoturvan tarvetta tuotteessa. SDL-prosessin tarve määrittyy sen mukaan, mihin tarkoitukseen tuote on suunniteltu. Sisältyykö siihen arkaluontoista tietoa ja toimiiko tuote internetissä tai jossain muussa verkossa? Määritys on esitetty kuviossa 7.

Projektiin nimetään tietoturvaneuvonantaja (Security Advisor), joka ohjaa kehitysryhmän SDL-prosessin läpi. Tietoturvaneuvonantajan tehtäviin kuuluu pääasiassa kommunikoida kehitysryhmän ja tietoturvaryhmän välillä. Hän valvoo, että turvallisuus on ajan tasalla kehityksen eri vaiheissa. Hän määrittää ja arvioi kehityksen aikana tietoturvan toteutumista ja puuttuu tarvittaessa ongelmiin. Tietoturvaneuvonantaja voi toimia pienissä projekteissa yksin ilman erillistä tietoturvaryhmää, mutta isoissa yrityksissä ja projekteissa on usein mukana tietoturvaryhmä, jolloin tietoturvaneuvonantaja toimii tietoturvaryhmän johtajana. (Howard–Lipner 2006, 68–69.)



Kuvio 10. Projektin tiedonkulun hierarkia (Howard–Lipner 2006, 72)

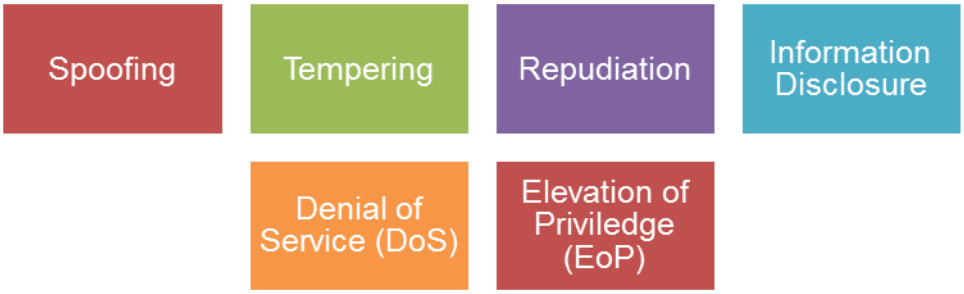
3.2.2 Laatutasot ja haavoittuvuuskartoitus

Laatutasoilla (Quality Gate) ja haavoittuvuuskartoituksella (Bug Bar) on totuttu määrittämään hyväksyttävät tietoturvan ja tietosuojan laadun minimivaatimukset. Näiden kriteereiden määrittämisellä projektin alussa pyritään parantamaan turvallisuuskysymyksiin liittyvien riskien ymmärtämistä. Laatutasot ja haavoittuvuuskartoitus mahdollistaa tietoturvahaavoittuvuuksien tunnistamisen ja korjaamisen kehityksen aikana. Projektiryhmän on neuvoteltava laatutasoista kaikissa kehitysvaiheissa. Sen jälkeen ryhmän täytyy hyväksyttää se tietoturvaneuvonantajalla, joka voi tarvittaessa lisätä projektikohtaisia selvennyksiä ja tiukentaa turvallisuusvaatimuksia. Projektiryhmän on myös havainnollistettava neuvoteltujen laatutasojen noudattamista, jotta lopullinen tietoturvakatselmointi (Final Security Review) saadaan päätökseen. (Microsoft 2014b, 8.)

Haavoittuvuuskartoitus koskee koko ohjelmistotuotantoprosessia. Sitä käytetään määrittämään tietoturvahaavoittuvuuksien vakavuuden kynnysarvoja. (Microsoft 2014b, 8.) Haavoittuvuuskartoituksen avulla määritetään uhkien erityispiirteet, järjestelmän rakenteiden riskialttius erilaisille uhille ja kuinka vakavasti niihin tulisi suhtautua. Ohjelmiston rakennetta tarkastellaan ja sen jälkeen määritetään eri osille riskitasot. Riskitasot voi luokitella 1–4 tasoon, jossa taso 1 on korkein riskitaso ja taso 4

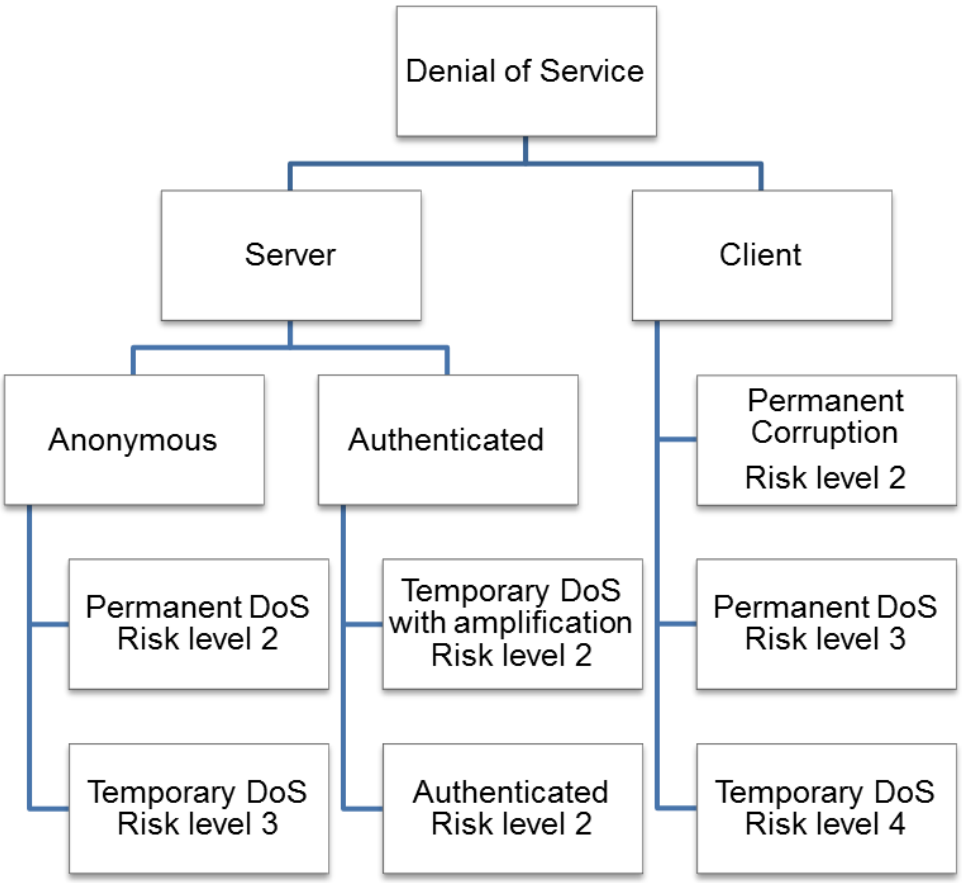
matalin. (Howard–Lipner 2006, 121.) Riskitaso kertoo kuinka rakenteen eri osiin tulee suhtautua. Tason 1–2 uhkiin tulee puuttua välittömästi niiden esiintyessä, tasoon 3 tulee puuttua viimeistään ennen tuotteen julkaisua ja tasoon 4, jos aikaa riittää. (Howard–Lipner 2006, 124.)

Haavoittuvuuskartoitus määritetään ottamalla järjestelmään kohdistuvat uhat huomioon. Uhkatyypit (STRIDE) ovat yleisimpiä tietoturvauhkia, joiden avulla haavoittuvuuskartoitus on suositeltavaa toteuttaa. Uhan ominaisuuksien avulla arvostellaan järjestelmää ja arvioidaan, millaista vahinkoa uhalla on mahdollista tehdä. Tämän jälkeen päätetään, mikä riskitaso järjestelmän osalle annetaan. Kuviossa 11 on esitetty, mitä ovat yleisimmät uhkatyypit.



Kuvio 11. Tietoturvauhkatyypit (STRIDE) (Howard–Lipner 2006, 120–121)

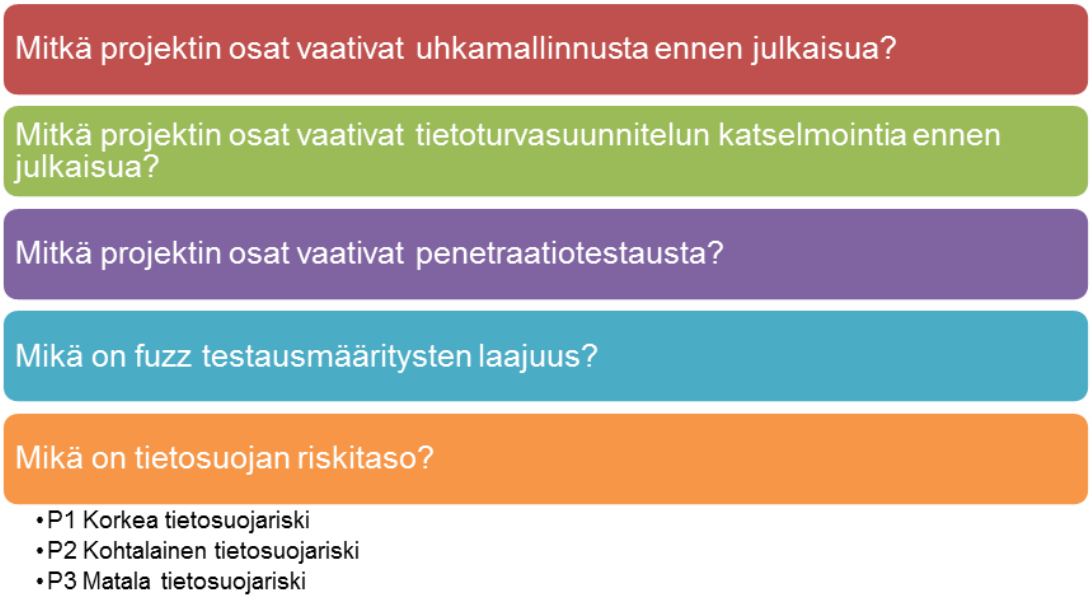
Kuviossa 12 on esimerkki haavoittuvuuskartoituksesta, jossa on määritetty palvelunestohyökkäyksen (DoS) erityispiirteet ja riskitasot jokaiselle uhalle.



Kuvio 12. Haavoittuvuuskartoitus palvelunestohyökkäyksestä (DoS) (Howard–Lipner 2006, 123)

3.2.3 Tietoturva- ja tietosuojariskien arviointi

Ennen kuin ohjelmiston suunnitteluun ja toteutukseen käytetään enemmän aikaa, on syytä selvittää, kuinka paljon turvallisen ohjelmiston rakentaminen nostaa kustannuksia. Riskit johtavat tarkempaan toteutukseen, joka puolestaan nostaa kustannuksia. Tietoturva ja tietosuoja ovat monimutkaisia ja niitä on vaikea erottaa toisistaan joissain tapauksissa, joten on tärkeää ymmärtää, kuinka ne vaikuttavat ohjelmistoon. Riskien arviointi on tärkeä osa turvallisen ohjelmiston rakennusta. (Howard–Lipner 2006, 93.) Tietoturva- ja tietosuojariskien arvioinnit ovat pakollisia prosesseja, jotka auttavat hajauttamaan resursseja, kun ohjelmistoa kehitetään (Howard–Lipner 2006, 99). Jokaisessa arvioinnissa tulisi vastata kuviossa 13 esitettyihin kysymyksiin.



Kuvio 13. Tietoturva- ja tietosuojariskien arviointikysymykset (Microsoft 2014b, 9)

Tietoturvariskien arvioinnin tarkoituksena on tarkastella ohjelmiston heikkoja kohtia, jotka ovat riskialttiita hyökkäyksille tai aiheuttaa eniten vahinkoa, jos järjestelmään hyökätään. Tietoturvariskien arviointi voidaan toteuttaa kyselynä, jonka avulla selvitetään, mihin kohtiin ohjelmistoa panostetaan, jotta vahinkoa ei pääsisi syntymään. (Howard–Lipner 2006, 94.) Tämän jälkeen kyselyn vastaukset analysoidaan. Analysoinnin hoitaa tietoturvaneuvonantaja tai isoissa projekteissa tietoturvaryhmä. Vastausten analysoinnin jälkeen osataan määrittää, kuinka varautua riskeihin. (Howard–Lipner 2006, 96.)

Tietoturvariskien lisäksi tulee arvioida tietosuojariskejä. Tietosuojariskien arvioinnissa on kolme tasoa, joiden avulla tiedon tärkeyttä arvioidaan. Nämä käytännöt ovat korkea, kohtalainen ja matala tietosuojariski. Korkea tietosuojariski määritetään, jos ohjelmisto sisältää henkilökohtaista tunnistettavaa tietoa (PII) tai siirtää sitä kolmannelle osapuolelle. Henkilökohtaista tunnistettavaa tietoa ovat esimerkiksi nimi, ikä ja osoite. Luottokorttinumerot ja henkilöturvatunnukset ovat henkilökohtaista tunnistettavaa tietoa, mutta niille on annettu oma kategoriansa (Sensitive PII). Korkea tietosuojariski kattaa jotain henkilökohtaista tunnistettavaa tietoa. Kohtalainen tietosuojariski määritetään, jos ohjelmisto sisältää anonyymia tietoa tai siirtää sitä kolmannen osapuolen palveluille. Matala tietosuojariski

määritetään, jos mikään korkean tai kohtalaisen tietosuojariskin määritelmistä ei toteudu. (Howard–Lipner 2006, 96–98.)

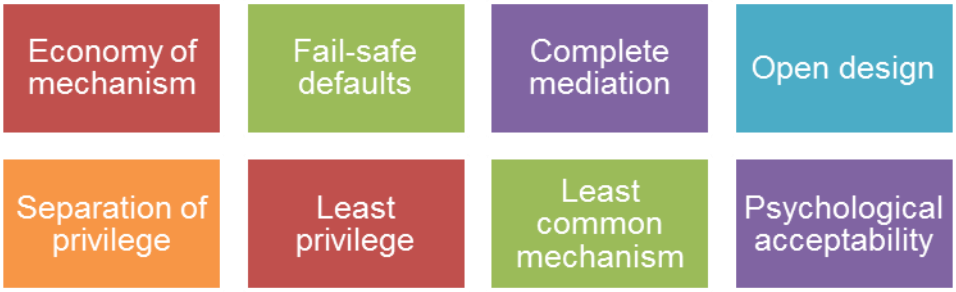
3.3 SUUNNITTELU

3.3.1 Suunnittelumääritykset

Suunnitteluvaiheessa rakennetaan suunnitelma, kuinka projekti viedään läpi SDL-prosessista aina toteutuksen ja testauksen kautta julkistamiseen asti. Suunnitteluvaiheen aikana noudatetaan suunnittelun parhaimpia käytäntöjä (Best Practices for Design). (Microsoft 2014a, 21.) Turvallinen suunnittelu on välttämätöntä kaikille ohjelmistoille (Howard–Lipner 2006, 75). Suunnittelun varhaisessa vaiheessa on tärkeää vaikuttaa suunnittelun luotettavuuteen. On hyödyllistä tarkastella huolellisesti tietoturvaan ja tietosuojaan liittyviä asioita suunnitteluvaiheen aikana, jolloin säästytään ylimääräisiltä kustannuksilta. Toiminnallisista määrityksistä tulisi kuvailla tietoturva- ja tietosuojaominaisuudet, jos ne kohdistuvat suoraan käyttäjiin. Tällaisia toiminnallisuuksia ovat esimerkiksi käyttäjätunnistus, ennen kuin käyttäjä pääsee käsiksi arkaluontoisia tietoja sisältäviin ominaisuuksiin. Suunnittelumääritysten pitäisi kuvailla, kuinka nämä ominaisuudet toteutetaan ja kuinka ne toteutetaan turvallisina ominaisuuksina (Secure feature). Turvallinen ominaisuus on huolella suunniteltu, jos se ottaa tietoturvan huomioon ja sisältää tiukan validoinnin ja salauksen, käsiteltäessä tietoja. Turvallisia ominaisuuksia ei tule sekoittaa turvaominaisuuksiin (Security Feature). (Microsoft 2014a, 22.)

On olemassa lukuisia turvallisen suunnittelun periaatteita, mutta klassisin ja eniten viitatuin malli on kuvattu kuviossa 14. Koodi ja suunnittelu tulisi pitää mahdollisimman yksinkertaisena ja pienenä. Monimutkaisesti rakennetussa ohjelmistossa on todennäköisesti enemmän virheitä. Ohjelmiston toiminnot tulisi pysyä turvallisina, vaikka ohjelmistossa tehty komento palauttaisi virheen. Pääsy arkaluontoisiin tietoihin tulisi tarkistaa. Ohjelmiston suunnittelu tulisi olla avointa. Vaikka suunnitelma joutuisi vääriin käsiin, sen ei pitäisi tuottaa vahinkoa ohjelmistolle. Oikeuksia tulisi hajauttaa. Esimerkiksi käyttäjän tunnistaminen (Autentikointi) pitäisi hajauttaa siten, että se perustuu kahteen edellytykseen eikä vain yhteen. Ohjelmiston täytyy toimia

mahdollisimman pienillä oikeuksilla. Prosessien tulisi toimia itsenäisesti eikä niiden tulisi jakaa resursseja keskenään. Esimerkiksi kahden prosessin ei tulisi muokata samaa tiedostoa tai ohjelmoitaessa tulisi käyttää mieluummin paikallisia kuin globaaleja muuttujia. Jos turvallinen ohjelmisto ei ole helppo käyttää, sitä ei haluta käyttää tai turvaominaisuuksia kytketään pois päältä. Turvallisuuden ja käyttöliittymäsuunnittelun tasapaino on tärkeää. (Howard–Lipner 2006, 76–77.)



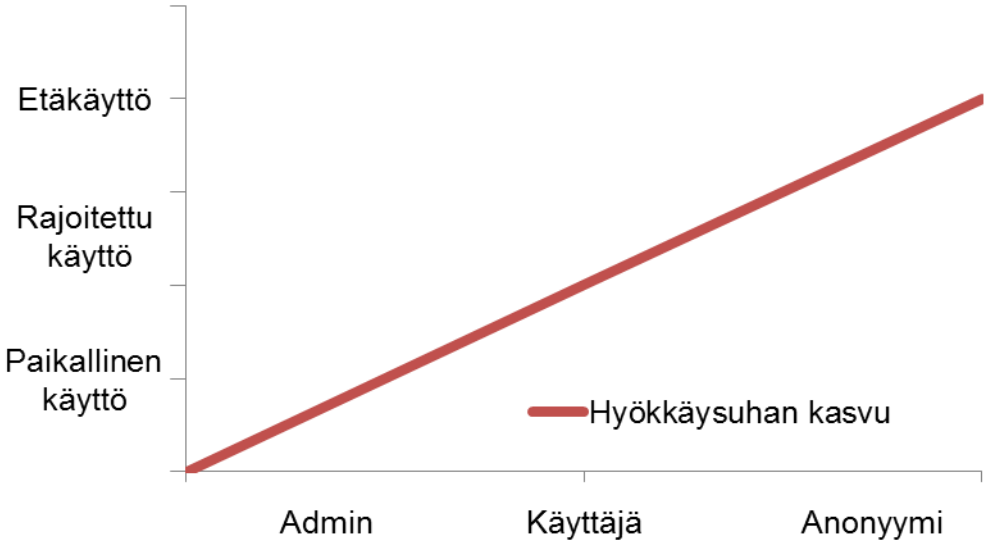
Kuvio 14. Yleiset turvallisen suunnittelun periaatteet (Howard–Lipner 2006, 76–77)

3.3.2 Hyökkäysuhan vähentäminen

Ohjelmiston käyttäjien ja ohjelmistoja vastaan hyökkääjien ei tulisi päästä käsiksi lähdekoodiin. Koodi sisältää aina virheitä ja osa niistä on tietoturvaan liittyviä, ja näitä hyökkäjät voivat käyttää hyväkseen. Hyökkäysuhan vähentämisen (Attack Surface Reduction) tarkoituksena on ymmärtää, mistä hyökkäysuhka sovelluksessa koostuu ja kuinka uhkaa voidaan vähentää. Ohjelmistoteollisuus keskittyy liian paljon koodin laatuun. Koodin laatu on ensiarvoisen tärkeää, mutta paraskin koodi voi olla haavoittuva hyökkäykselle. Uusia haavoittuvuuksia löytyy koko ajan lisää ja ne kehittyvät nopeasti, joten tänä päivänä tehty turvallinen koodi voi olla altis hyökkäyksille tulevaisuudessa. (Howard–Lipner 2006, 78.)

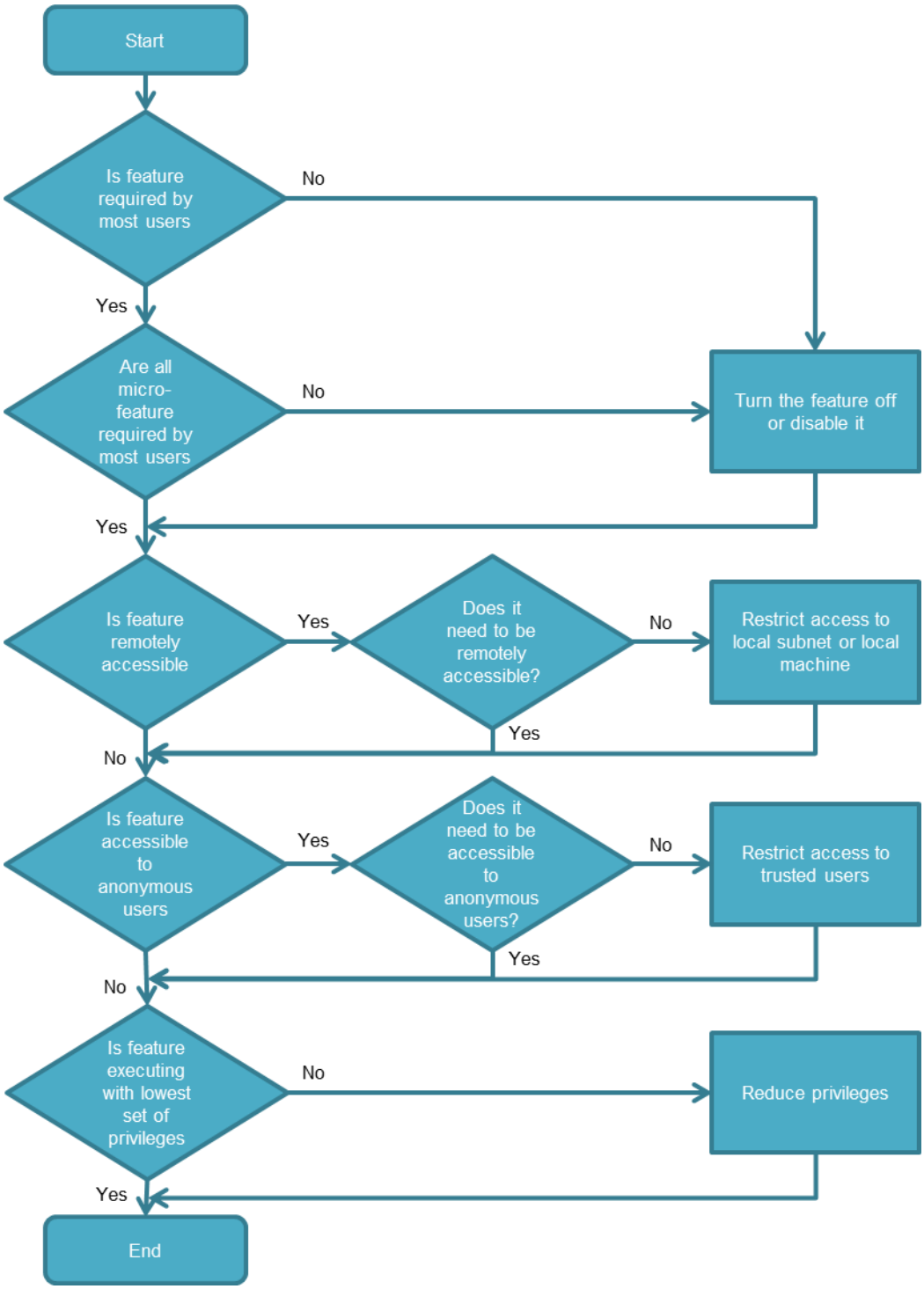
Hyökkäysuhan vähentäminen keskittyy kolmeen pääkohtaan (Howard–Lipner 2006, 79). Ensimmäisessä pääkohdassa pyritään vähentämään oletuksena suoritettavan koodin määrää. Jos ohjelmiston toimintoa tarvitsee alle 80 prosenttia kaikista käyttäjistä, toiminto tulisi oletuksena sulkea tai estää. (Howard–Lipner 2006, 81.) Toisena kohtana rajoitetaan kenellä on käyttöoikeudet koodiin ja mistä koodiin päästään käsiksi. Kuvio 15 kuvaa kuinka hyökkäysuhka kasvaa, kun siirrytään admin-käyttäjästä

tuntemattomaan käyttäjän ja lokaalista etäkäyttöön. Kolmantena kohtana pyritään vähentämään käyttäjien käyttöoikeudet sille tasolle, että käyttäjä saa tarvittavat työt tehtyä, mutta ei pääse käsiksi muihin tietoihin. Tämä tuo lisäturvallisuutta ja sillä estetään oikeuksien väärinkäytöt. (Howard–Lipner 2006, 82–83.)



Kuvio 15. Käyttöoikeudet kasvattavat hyökkäyksen uhkaa (Howard–Lipner 2006, 82)

Kuvio 16 esittää vuokaavion, jonka vaiheita seuraamalla hyökkäysuhkaa voidaan vähentää. Vaiheet seuraavat hyökkäysuhan vähentämisen kolmea pääkohtaa ja ne tarjoavat toimintamallit ohjelmiston suojaamiseen.



Kuvio 16. Hyökkäysuhan vähentämisen vuokaavio (Howard–Lipner 2006, 80)

3.3.3 Uhkamallinnus

Uhkamallinnus on yksi tärkeimmistä vaiheista turvallisen ohjelmiston kehityksessä. Projektin varhaisessa vaiheessa hyvin suoritettu ja oikein tehty uhkamallinnus auttaa löytämään tietoturvaongelmia ennen kuin yhtään riviä

koodia on kirjotettu. Tämä johtaa huomattaviin säästöihin kustannuksissa. (Howard–Lipner 2006, 101.)

Defect Introduction Point	Defects Found During Requirements	Defects Found During Architecture	Defects Found During Construction	Defects Found During Test	Defects Found After Release
Requirements	1	3	5-10	10	10-100
Architecture	None	1	10	15	25-100
Construction	None	None	1	10	10-25

Kuvio 17. Kustannusten kehittyminen ohjelmiston elinkaaren edetessä (Howard–Lipner 2006, 101)

Uhkamallinnuksen tarkoituksena on ymmärtää järjestelmän potentiaaliset tietoturvauhat, määritellä riskit ja luoda asianmukaiset lievennykset. Uhkamallinnus auttaa myös yrityksiä hallitsemaan ohjelmistojen riskejä, luo tietoisuutta tietoturvauhista ja tarjoaa kyvyn kääntää tekniset riskit liiketoiminnan vaikutukseksi (Business Impact). Uhkamallinnus ei ole staattinen prosessi, vaan sitä tulisi kehittää jatkuvasti helposti lähestyttäväksi ja käyttäjälleen hyödylliseksi. (Howard–Lipner 2006, 101–102.)

Uhkamallinnuksen hyödyt ovat moninaiset. Uhkamallinnus on osa riskienhallintaprosessia. Ohjelmiston ja infrastruktuurin uhat ovat riskejä käyttäjille ja ympäristölle. Uhkamallinnus paljastaa järjestelmän uhat ennen kuin koodia on kirjoitettu. Kehitysryhmä käy suunnittelmaa uudestaan läpi uhkamallinnuksen aikana, jolloin se tarkistaa uudelleen arkkitehtuuria ja suunnittelua. Uhkamallinnus pakottaa kehitysryhmän ajattelemaan suunnitelmaa tietoturvan ja tietosuojan näkökulmasta. Se selkeyttää ylläpitovaiheen vastatoimien valintaa ohjelmistolle ja ympäristölle. Uhkamallinnus on osallisena myös hyökkäysuhan vähentämisessä. Se auttaa ohjaamaan ohjelmiston koodin katselmointia sekä haavoittuvuus- ja murtotestausprosessia. (Howard–Lipner 2006, 102.)

Uhkamallinnusprosessin pääasiallisena tuloksena on dokumentti, joka kuvailee taustatiedot ohjelmistosta ja määrittää yleisen tason ohjelmistomalliin (Application Model). Ohjelmistomallin luontiin käytetään usein tietovirtakaaviota (Data Flow Diagram). Siinä on listattu osat, jotka tarvitsevat suojausta sekä järjestelmän uhat, jotka on listattu riskin vakavuuden mukaan ja vaihtoehtoisesti lievennykset. Asianmukaisten taustatietojen tulisi sisältää käyttöskenaariot (Use Scenarios), jotka sisältävät tuotteeseen kohdistuvat pääasialliset uhat, jotka voivat kohdata tuotetta. Ohjelmiston ulkoiset riippuvuudet (External Dependencies) toisiin palveluihin tulee kirjata ylös. Taustatietoihin tulee kirjata myös tietoturvaan liittyvät olettamukset (Security Assumptions) ja ulkoiset tietoturvamuistiinpanot (External Security Notes). Ulkoiset tietoturvamuistiinpanot sisältävät loppukäyttäjälle tai ylläpitäjälle tarpeellista tietoa turvalliseen käyttämiseen. (Howard–Lipner 2006, 103.)

Kuvio 18. Uhkamallinnusprosessi (Howard–Lipner 2006, 105)

3.4 TOTEUTUS

3.4.1 Varmistettujen työkalujen käyttö

Ohjelmistoteollisuus on täynnä turvallisen ohjelmoinnin parhaita käytäntöjä, mutta vain harvoja niistä käytetään. Turvallinen ohjelmistotuotannon elinkaari vaatii tarkkoja ohjelmointikäytäntöjä ja tukee testikäytäntöjen avulla sitä, että ohjelmointikäytäntöjä noudatetaan. (Howard–Lipner 2006, 143.)

Kehitysryhmän täytyy määrittää ja julkaista lista varmistetuista työkaluista ja niiden turvatarkastuksista. Varmistettujen työkalujen lista tulee hyväksyttää tietoturvaneuvonantajalla. Yleisesti ottaen kehitysryhmän pitäisi pyrkiä käyttämään varmistettujen työkalujen viimeisimpiä versioita ja hyödyntämään uusien tietoturva-analyysien toiminnallisuuksia ja suojauksia. (Microsoft 2014b, 10.)

3.4.2 Epäluotettavien funktioiden välttäminen

Monet yleisesti käytetyt funktiot ja rajapinnat eivät ole turvallisia nykyisessä uhkaympäristössä. Projektiryhmän täytyy analysoida kaikki funktiot ja rajapinnat, joita käytetään ohjelmistossa ja kieltää ne, jotka eivät ole turvallisia. Kun kielletyt funktiot ja rajapinnat on määritetty, projektiryhmän tulisi käyttää esimerkiksi koodintarkistustyökalua. Niillä tarkistetaan, ettei koodi sisällä kiellettyjä funktioita tai rajapintoja. Jos niitä löytyy, ne korvataan turvallisilla vaihtoehdoilla. (Microsoft 2014b, 10.)

Ohjelmointikieliä arvioidaan ja niistä etsitään jatkuvasti heikkouksia. Ohjelmointikielet sisältävät epäluotettavia funktioita, jotka voivat aiheuttaa tietoturvauhan. Epäluotettaville funktioille on yleensä turvallisempi vaihtoehto, jota tulisi käyttää, jotta tuotettava ohjelmisto on turvallinen. (Howard–Leblanc 2004, 241–242.)

3.4.3 Staattinen analyysi

Projektiryhmän tulee suorittaa lähdekoodille staattinen analyysi. Staattisessa analyysissä tarkastellaan lähdekoodia tarkastustyökalujen avulla sekä manuaalisesti. Lähdekoodista pyritään etsimään virheitä. (Howard–Lipner

2006, 145–146.) Staattinen analyysi tarjoaa valmiudet turvallisen koodin läpikäyntiä varten ja auttaa varmistamaan, että turvallisen koodin käytäntöjä noudatetaan. Staattinen analyysi ei itsessään riitä korvaamaan manuaalista koodin läpikäyntiä. (Microsoft 2014b, 11.)

On virhe luulla, että staattisen analyysin suorittavat työkalut korvaavat ihmisen tekemän tarkastuksen. Työkalut saattavat jättää huomioimatta vakavia virheitä ja tätä varten tarvitaan myös ihmisen suorittama tarkistus. Virheeksi voidaan luokitella myös väärät hälytykset. Työkalu saattaa ilmoittaa kielletystä funktiosta, joka todellisuudessa ei luo tietoturvauhkaa. Tällaisiin tapauksiin tarvitaan ihmisen harkintakykyä, mihin täytyy puuttua ja mihin ei. Lähdekoodin analysointityökalut keskittyvät yleensä rajattuun ryhmään ohjelmointikieliä. Jos lähdekoodi sisältää monia ohjelmointikieliä, täytyy analyysi suorittaa useammalla analysointityökalulla. Suurin osa työkaluista löytää ainoastaan lähdekoodin virheitä eivätkä ne kiinnitä huomiota suunnitteluvirheisiin. (Howard–Lipner 2006, 145–146.)

3.5 VAHVISTUS

3.5.1 Dynaaminen analyysi

Ohjelmiston ajonaikainen testaus (Run-Time Verification) on välttämätön toimenpide, jolla varmistetaan, että ohjelmisto toimii suunnitellulla tavalla. Tämä toimenpide määrittelee työkalut, jotka valvovat ohjelmiston käyttäytymistä muistin korruptoitumisen, käyttöoikeuteen liittyvien asioiden ja muiden vakavien tietoturvaongelmien varalta.

Ajonaikaista testausta käytetään rinnakkain muiden testausmenetelmien, kuten fuzz-testauksen (Fuzz Testing) kanssa, tavoiteltujen testaustulosten saavuttamiseksi. (Microsoft 2014b, 11.) Fuzz-testauksessa luodaan epämuodostunut tiedosto, joka syötetään järjestelmään ja tarkastellaan, kuinka järjestelmä reagoi siihen (Howard–Lipner 2006, 163). Ajonaikainen testaus voi löytää jopa vakavia arkkitehtuuritason tietoturvavirheitä testauksen ja analysoinnin aikana. Toimenpide tulisi suorittaa säännöllisesti, jotta varmistutaan, ettei ohjelmisto sisällä virheitä. (Howard–Lipner 2006, 165.)

3.5.2 Fuzz-testaus

Fuzz-testaus kehitettiin alun perin luotettavuusvirheiden havaitsemiseen. Fuzz-testaus on tehokas tapa löytää myös tietoturvavirheitä. Fuzz tarkoittaa epämuodostuneen tiedoston luomista, joka syötetään ohjelmiston luettavaksi. Tämän jälkeen nähdään, miten ohjelmisto reagoi epämuodostuneeseen tiedostoon. Jos se ei reagoi odotetusti, on oletettavasti löytynyt luotettavuusvirhe, joka samalla voi olla myös tietoturvavirhe. (Howard–Lipner 2006, 153–154.)

Fuzz-testaus on kohdistettu koodiin, joka analysoi tietorakenteita. Tällaisia tietorakenteiden analysoijia ovat parserit. Parserit voidaan jakaa karkeasti kolmeen luokkaan. Tiedostotyyppiparseri (File Format Parser) manipuloi esimerkiksi kuvan tiedostotyyppejä. (Howard–Lipner 2006, 154–155.) Verkkoprotokollaparseri (Network Protocol Parser) puolestaan lähettää epämuodostuneita paketteja prosessia kuuntelevalle verkkoportille, esimerkiksi TCP-protokollan (Transmission Control Protocol) tai UDP-protokollan (User Datagram Protocol) välityksellä (Howard–Lipner 2006, 160). Rajapintaparserit ja muut parserit (APIs and Miscellaneous Parsers) tarkoittavat kaikkia muita tietorakenteita analysoivia tai manipuloivia parsereita tai parsereita, jotka ovat peräisin epäluotettavista lähteistä (Howard–Lipner 2006, 163).

Kuvio 19. Fuzz-testauksen alueet (Howard–Lipner 2006, 154)

Fuzz-testauksen jälkeen ilmenneet ongelmat tutkitaan tarkemmin koodia läpikäymällä ja arvioidaan ongelmien riskit tietoturvan kannalta. Jos testauksen avulla ei löydetä yhtään virhettä, on syytä tarkistaa, että fuzz-testaustyökalu luo epämuodostuneita tiedostoja. (Howard–Lipner 2006, 164.)

3.5.3 Uhkamallin ja haavoittuvuuskartoituksen katselmointi

Uhkamallinnuksen dokumentit ovat korvaamattomia tietoturvatestauksen aikana. Toisinaan ohjelmiston toiminnallisuus ja toteutus saattavat muuttua

suunnitteluvaiheen jälkeen. Tämän takia uhkamallit tulisi läpikäydä uudelleen, jotta varmistutaan siitä, että ohjelmiston toiminnallisuudet ovat käsitelty tarkasti ja kattavasti. Uhkamalleja tulisi myös käyttää tietoturvatestauksen suunnittelun ohjaamisessa ja tiedottamisessa. Ohjelmiston osat, joilla on suurin hyökkäysuhka ja korkein riskitaso, tulisi testata perusteellisimmin. (Howard–Lipner 2006, 165.)

Projektiryhmän tulisi arvioida uudelleen ohjelmiston hyökkäysuhkaa testausvaiheessa. Se auttaa ymmärtämään, millä ohjelmiston osilla on suurin alttius hyökkäyksille ja suurin vahinkoriski, jos haavoittuvuus osuu kohdistettuun ohjelmiston osaan. Hyökkäysuhan arviointi auttaa ryhmää keskittymään testauksessa korkean riskin omaaviin alueisiin ja tekemään niihin tarvittavat korjaukset. Hyökkäysuhkien uudelleen arvioinnin jälkeen arviointi tulisi dokumentoida perusteluineen. (Howard–Lipner 2006, 166.)

3.6 JULKISTAMINEN

3.6.1 Vastatoimien suunnittelu

Jokaiseen turvalliseen ohjelmistoon on tehtävä suunnitelma (Incident Response Plan), jolla valmistaudutaan toteuttamaan vastatoimia, jos ohjelmistosta löytyy haavoittuvuus sen julkaisun jälkeen. Vaikka ohjelmistoa on suunniteltu ja toteutettu mahdollisimman turvalliseksi, ei voida olettaa, että se olisi täysin turvallinen. Jos ohjelmistossa ei ole haavoittuvuuksia julkistamisen aikaan, voi niitä esiintyä esimerkiksi vuoden päästä. (Howard–Lipner 2006, 187.)

Mikään projektiryhmä ei ole täydellinen. Jokainen tekee virheitä, joten täydellisen turvallista ohjelmistoa ei ole. SDL-prosessia käyttämällä haavoittuvuudet vähenevät enemmän kuin pelkästään turvallisen toteutuksen parhaita käytäntöjä käyttämällä. Se ei kuitenkaan takaa täysin turvallista ohjelmistoa, joten on hyödyllistä tehdä suunnitelma haavoittuvuuksien varalle. Uuden tyyppisiä haavoittuvuuksia löytyy jatkuvasti lisää. Haavoittuvuudet tekevät ohjelmiston turvattomaksi, joten niiden seuraaminen ja korjaaminen on tärkeää, jotta turvallisuutta voidaan ylläpitää. (Howard–Lipner 2006, 187189.)

Vastatoimiprosessi sisältää kaksi ryhmää. Ensimmäinen ryhmä reagoi asiakkaiden ja haavoittuvuuksia tarkkailevien palautteisiin. Toinen ryhmä valmistautuu tutkimaan ja paikkaamaan esiintyneet haavoittuvuudet. (Howard–Lipner 2006, 213.)

3.6.2 Lopullinen tietoturvakatselmointi

Lopullinen tietoturvakatselmointi (Final Security Review) on ennen julkaisua suoritettava prosessi, jossa käydään läpi kaikki ohjelmiston tietoturvaominaisuudet. Katselmoinnin suorittaa tietoturvaneuvonantaja kehitysryhmän sekä tietoturva- ja tietosuojaryhmän avustuksella. (Microsoft 2014b, 12.) Lopullisen tietoturvakatselmoinnin tarkoituksena on arvioida tuotteen julkaisukelpoisuus tietoturvan osalta. Lopullinen tietoturvakatselmointi on arviointi, jonka avulla nähdään onko projektiryhmä seurannut SDL-prosessia oikein koko ohjelmiston elinkaaren ajan. Jos sitä on seurattu oikein ja huolellisesti, katselmointi on lyhyt prosessi. (Howard–Lipner 2006, 181.) Katselmointi sisältää yleensä uhkamallien tarkastelua, poikkeusten käsittelyä, työkalujen tarkastusta ja toiminnan vertailua aiemmin määriteltyihin laatutasoihin. (Microsoft 2014b, 12.)

Katselmoinnin tulokset voidaan jakaa kolmeen tasoon. Ensimmäinen taso (Passed FSR) on tuloksiltaan paras. Kaikki katselmoinnissa tarkastetut tietoturva- ja tietosuojakäytännöt on korjattu tai niitä on lievennetty. Toisessa tasossa (Passed FSR with Exceptions) kaikki korjaukset ja lievennykset on tehty muutamia poikkeuksia lukuun ottamatta. Poikkeuksiin liittyvät ongelmat on ratkaistu ja ne on dokumentoitu ja korjattu seuraavassa julkaisussa. Kolmannessa tasossa (FSR with Escalation) kaikki vaatimukset eivät ole täyttyneet ja tietoturvaneuvonantaja ei ole päässyt yhteisymmärrykseen kehitysryhmän kanssa kompromisseista, joten tuotetta ei voi hyväksyä ja julkaista. (Microsoft 2014b, 12.)

Kuvio 20. Lopullisen tietoturvakatselmoinnin tulostasot (Microsoft 2014b, 12)

3.6.3 Tuotteen julkistaminen ja arkistointi

Lopullisen tietoturvakatselmoinnin jälkeen voidaan päättää, onko tuote julkaisukelpoinen. Tietoturvakatselmoinnin tulosten pohjalta voidaan nähdä, onko turvallisen tuotteen suunnittelussa ja toteutuksessa onnistuttu. Jos tuote ei ole läpäissyt lopullista tietoturvakatselmointia, tuotetta ei voida julkaista. Tämä tarkoittaa sitä, että joudutaan palamaan suunnitteluvaiheeseen korjaamaan tietoturvaan liittyviä ongelmia, jotta riittävä turvallisuus voidaan taata.

Hyväksytyn ja julkaistavan tuotteen kaikki projektiin liittyvät tiedot tulee arkistoida, jotta julkaisun jälkeen tiedot ovat helposti saatavilla ylläpitovaiheessa. Näihin tietoihin kuuluvat kaikki tekniset tiedot, lähdekoodit, uhkamallit, dokumentaatio projektin eri vaiheista, lisenssisopimukset ja kaikki kolmannen osapuolen palveluihin liittyvät huoltotoimenpiteet. (Microsoft 2014b, 12.)

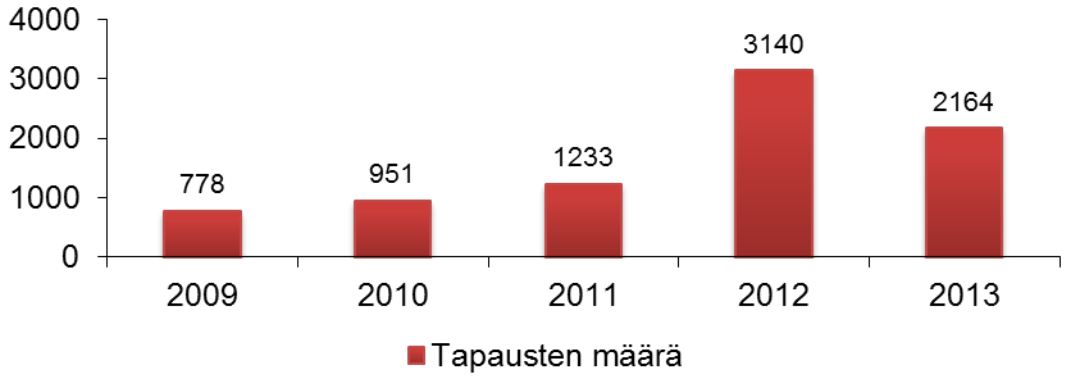
4 TIETOTURVAN KEHITTYMINEN OHJELMISTOTUOTANNOSSA

4.1 TIETOMURTOJEN KEHITTYMINEN

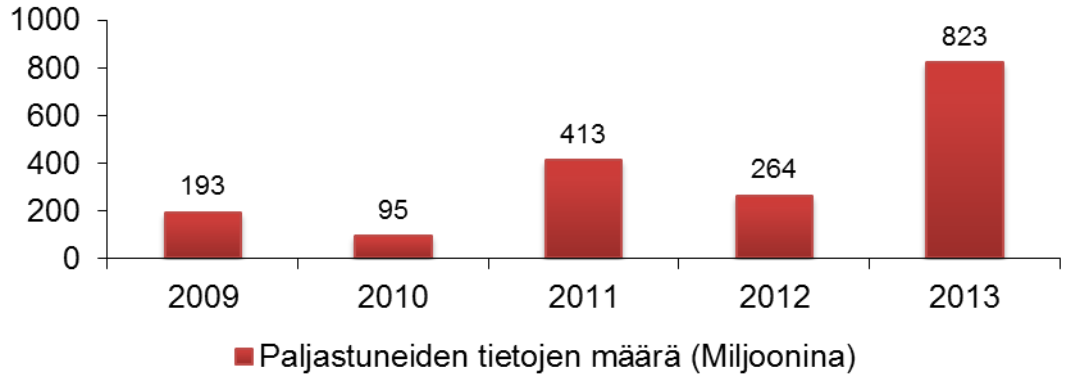
Tietomurtojen luonne on muuttunut merkittävästi viimeisen kymmenen vuoden aikana. Ennen erilaisten virusten ja haittaohjelmien tarkoituksena oli enemmänkin tehdä kiusaa käyttäjälle. 2000-luvun alkupuolella virukset ja haittaohjelmat pyrkivät leviämään mahdollisimman laajalle. Nykyään tietomurtojen tekijöihin kuuluu ammattirikollisia, jotka hakevat tietomurtojen, virusten ja haittaohjelmien avulla taloudellista hyötyä. Internet sisältää paljon palveluita, jotka sisältävät käyttäjien arkaluontoisia tietoja, kuten luottokorttitietoja. Rikolliset voivat myös kaapata tietokoneita käyttäjän huomaamatta osaksi Botnet-verkkoa ja käyttää sitä louhimaan Bitcoineja tai muita digitaalisia valuuttoja. Myös erilaiset tietokoneiden lukitsemiset ovat yleistyneet. Niissä haittaohjelma lukitsee tai korruptoi käyttäjän tiedostoja ja vaatii, että käyttäjä ostaa ohjelman, joka poistaa lukituksen tai korjaa korruptoituneet tiedostot. (F-Secure 2013.) Tietomurtojen uhka on todellinen ja ohjelmistoyritysten tulisi ottaa se huomioon omassa ohjelmistokehityksessään.

Kuviossa 21 on esitetty tietomurtotapausten määrä viimeisten viiden vuoden aikana ja kuviossa 22 paljastuneiden tietojen määrä samalla aikavälillä. Tietoturvauhat ovat todellisia ja niiltä tulisi suojautua. Taulukot kertovat tietoturvan tasosta. Siihen ei kiinnitetä tarpeeksi huomiota. Ohjelmistot ja sovellukset tallentavat käyttäjien arkaluontoisia tietoja tietokantoihin, jotka ovat helposti murrettavissa. Tämä luo käyttäjien keskuudessa epäluotettavuutta varsinkin, jos omat tiedot joutuvat anastetuiksi. Ohjelmistoyritys saa heikon tietoturvan vuoksi huonon maineen mediassa ja asiakkaat kaikkoavat. On erittäin tärkeää, että tietoturva on ensisijaisena tavoitteena, jos ohjelmiston tai sovelluksen tarkoituksena on käsitellä käyttäjien arkaluontoisia tietoja. Suurin osa tietomurroista tehdään järjestelmän ulkopuolelta, kuten kuvio 23 osoittaa. Myös sisäiset tietomurrot ja vahingossa tehdyt paljastukset ovat osa tietomurtotapauksia. Yritysten tulee turvallisessa ohjelmiston elinkaaressa ottaa huomioon sisäisten tietomurtojen mahdollisuus. Esimerkiksi projektin edetessä tulee olla selvät

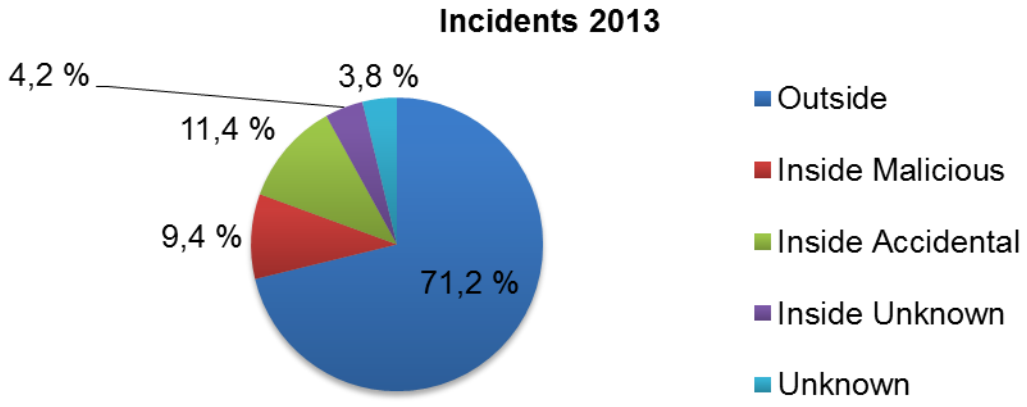
säännöt, kenellä on oikeudet tietojen muokkaukseen ja varmistua siitä, että tietoihin pääsevä henkilö on luotettava.



Kuvio 21. Rekisteröityjen tietomurtotapausten määrän jakautuminen (Risk Based Security 2013, 2)



Kuvio 22. Tietomurtotapauksissa paljastuneiden tietojen määrä miljoonina (Risk Based Security 2013, 2)



Kuvio 23. Tietomurtotapauksen aiheuttajan jakautuminen (Risk Based Security 2013, 3)

Syksyllä vuonna 2013 tehtiin Adobe Systems -järjestelmiin tietomurto, jossa anastettiin noin 150 miljoonan käyttäjän tietoja, kuten nimiä, sähköpostiosoitteita, salasanoja ja pankki- ja luottokorttien tietoja. Salasanat ja pankki- ja luottokorttien tiedot olivat salatussa muodossa. Anastuksen kohteeksi joutui myös Adobe Systems -tuotteiden lähdekoodeja. Tietomurtojen kohteet keskittyvät suuriin ohjelmistoyrityksiin ja niiden tuotteisiin, koska ohjelmistoyrityksillä on suuret käyttäjämäärät ja arkaluontoisia tietoja käyttäjistään. Adobe Systemsin kaltaisilla yrityksillä on suuri vastuu siitä, että tietoturva ja tietosuoja ovat ajan tasalla. Suuremmilta vahingoilta vältyttiin koska, salasanat ja pankki- ja luottokorttien tiedot olivat salattuja, eikä niitä ei päästy hyödyntämään.

4.2 TIETOTURVAN TILA

4.2.1 Kustannukset

Miksi tietoturvaa ei oteta tarpeeksi huomioon ohjelmistotuotannossa, vaikka tieto, välineet ja menetelmät ovat helposti saatavilla? Tähän kysymykseen liittyy oleellisesti kustannukset. Tietoturvaprosessin integroiminen omaan ohjelmistotuotantoprosessiin, ammattitaitoisen henkilökunnan hankkiminen tai sen kouluttaminen tuovat yritykselle lisää kustannuksia. Tietoturvan integroiminen on kallista, mutta sitä voidaan pitää sijoituksena tai vakuutuksena. Sijoituksena se toimii niin, että tietoturvan avulla yrityksen ohjelmistot saavat hyvän ja turvallisen maineen, jolloin on mahdollista kasvattaa yrityksen liiketoimintaa. Tietoturvan integrointia voidaan pitää enemmän vakuutuksena. Tietoturvaprosessia seuraamalla luodaan turvallisempia ohjelmistoja, jolloin kustannukset eivät hyökkäyksen sattuessa ole yhtä suuret kuin ne olisivat, jos niihin ei olisi varauduttu prosessin aikana.

Ohjelmistoyritykset luottavat liikaa, ettei heidän ohjelmistoihin tai palveluihin hyökätä ja tietoja kalastella. Kustannuksien takia ne ottavat riskin jättämällä tietoturvan liian vähälle huomiolle, joka voi hyökkäyksen sattuessa tulla erittäin kalliiksi. Asiakkaan roolia ei voida myöskään unohtaa. Asiakkaan, jolle ohjelmistoa tehdään, tulisi vaatia tietoturvan huomioimista. Tämä asettaa ohjelmistoyritykselle vaatimukset ottaa tietoturva huomioon ohjelmistoissaan. Kustannukset tietenkin suurenevat, mutta tietoturvan

täytyisi olla itseisarvo. Tietoturva ei saa olla lisäominaisuus, joka asennetaan lopuksi, jos budjetti sen sallii. Vaikutuksilta liiketoimintaan ei voida välttyä, kun puhutaan tietoturvasta. Turvallisuuden ja tietosuojan vaikutus on suoraan verrannollinen laadukkaaseen ohjelmistoon. Tämä tulisi jokaisen ohjelmistoyrityksen muistaa.

4.2.2 Tietoturvan kehittäminen

Ohjelmistoyrityksillä on ollut pitkään ensisijaisena tavoitteena luoda toimivia ohjelmistoja asiakkaille mahdollisimman nopeasti. Nopea aikataulu ja tiukka budjetti saavat aikaan sen, että tietoturvaa ei oteta huomioon tai se on liian vähäistä. Pitkään on ajateltu, että tietoturva on ohjelmistoon lisättävä ominaisuus tai pelkästään testausvaiheessa suoritettava tietoturvatestaus, jolla taataan turvallisuus. Sitä se ei kuitenkaan ole, vaan tietoturva tulisi olla osana ohjelmiston elinkaarta alusta loppuun saakka.

Uskon, että tietoturvaan suhtautumiseen on tulossa muutos, sillä tietoturva-asioitten käsittely mediassa on lisääntynyt huomattavasti viime vuosina. Tämä johtuu suurista tietomurroista ja niiden määrän lisääntymisestä. Tietomurroista rikolliset ovat saaneet haltuunsa suuren määrän käyttäjätunnuksia, salasanoja, sähköpostiosoitteita ja pankki- ja luottokorttitietoja. Median tuoma huomio saa varmasti ohjelmistoyritysten johdon pohtimaan tietoturvaan panostamista, jotta tietoturvavahinkoja voidaan ennaltaehkäistä.

Tietoturvan takaamiseen saatavia menetelmiä ja toimintamalleja on saatavilla vapaaseen käyttöön. Esimerkkinä tästä on Microsoftin tietoturvan kehittämisen elinkaari. Vesiputousmalliin kaltainen tietoturvaprosessi ei ole ainoa vaan niitä on useita. Ketteriä menetelmiä käytetään paljon ohjelmistotuotannossa ja perinteinen vesiputousmalli alkaa olla harvinaisempi. Microsoft on muokannut tietoturvaprosessin myös ketterille menetelmille sopivaksi. Ohjelmistoyritykset käyttävät erilaisia menetelmiä ja jokaisella on itselleen sopivat työskentelymenetelmät. Tietoturvaprosessin integroiminen yrityksen tuotantomenetelmiin on siis yksilöllistä. Menetelmät tietoturvan takaamiseksi ovat siis valmiina. Suurilla ohjelmistoyrityksillä on enemmän resursseja lähteä ratkaisemaan tietoturvaan liittyviä ongelmia,

mutta miten pienet ja keskisuuret yritykset pystyisivät samaan? Kyse ei kuitenkaan ole monimutkaisista asioista, joilla tietoturvaa voidaan kehittää omissa ohjelmistoissa.

Tietoturvan merkitys tulee kasvamaan tulevaisuudessa. Ihmiset käyttävät enemmän internetiä kuin aikaisemmin ja palvelut siirtyvät sinne. Sosiaalisessa mediassa ihmiset kertovat enemmän itsestään ja ihmisen jalanjälki netissä kasvaa. Palveluntarjoajien on otettava vastuuta, jotta ihmisten arkaluontoiset tiedot eivät joudu vääriin käsiin. Ohjelmistoyritysten on panostettava tietoturvakoulutukseen, jonka avulla ohjelmistokehittäjät ovat enemmän tietoisia tietoturvauhista ja osaavat varautua niihin kehityksen aikana. Jo koulutuksen ja tietoisuuden lisäämisellä päästään pitkälle. Huolellinen ohjelmiston suunnittelu on pääpaino turvallisen ohjelmistotuotannon kehittämisessä, jota toteutuksen ja testauksen aikana tarkastellaan ja arvioidaan.

Tietoturvaan keskittyvät konsultointiyritykset antavat koulutusta ja neuvontaa ohjelmistoyrityksille tietoturvaan liittyen. Tällaisten yritysten olemassaolo on ensiarvoisen tärkeää, jotta keskisuuret ja pienet ohjelmistoyritykset saavat tietoturvan integroimisen helposti käyttöönsä. Koulutuksen ja neuvonnan antaminen konsultointiyritysten toimesta palvelevat tietoturvan kehitystä ohjelmistoyrityksissä. Konsultointiyritysten palvelut helpottavat myös ohjelmistoyritysten taakkaa aloittaa tietoturvan integroiminen omaan ohjelmistotuotantoprosessiin. Ammattitaitoinen apu ja kerralla oikein tehty suunnittelu ja toteutus säästävät myös kustannuksissa.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli löytää syyt, miksi tietoturvaan ei nykyisessä ohjelmistotuotannossa kiinnitetä tarpeeksi huomiota. Tavoitteena oli perehtyä Microsoftin turvalliseen ohjelmistotuotantoprosessiin, jolla suunnitellaan ja toteutetaan turvallisia ohjelmistoja. Työssä pohdittiin myös tietoturvan tilaa ohjelmistotuotannossa, tietomurtojen luonteen kehittymistä sekä tuotiin esiin kehitysideoita, joilla tietoturvan huomioon ottamista voitaisiin parantaa.

Tavoitteet saavutettiin onnistuneesti. Kustannusten nousu tietoturvan integroinnissa sekä konservatiivinen ajattelu ovat osasyynä siihen, ettei tietoturvaan kiinnitetä tarpeeksi huomiota. Tietoturvasta ja sen käytöstä ohjelmistotuotannossa saatiin paljon uutta tietoa. Työ tarjoaa kattavan yleiskatsauksen Microsoftin turvallisen ohjelmiston elinkaarimallista, jonka avulla on helppo ymmärtää tietoturvaprosessia.

Haasteena työssä oli uuteen tietoturva-asiaan perehtyminen ja sen opetteleminen. Opinnäytetyön aiheen rajaaminen osoittautui ongelmaksi, koska tietoturva aihealueena on laaja. Aiheessa oli paljon uutta asiaa, johon en ennen ollut tutustunut. Aiheen rajaaminen sopivaksi kokonaisuudeksi osoittautui haastavaksi. Tämä vei opinnäytetyöprosessissa paljon aikaa. Tietoturva ja sen integrointi osaksi ohjelmistotuotantoprosessia oli erittäin kiehtova aihe. Tietoisuus tietoturvasta ja sen hyödyntämisestä ohjelmistotuotannossa karttui merkittävästi.

Opinnäytteessä keskityttiin tietoturvan kehittämisen elinkaaren teoreettiseen tietoon ja yleiseen tietoon tietoturvasta. Luonnollinen jatko opinnäytetyölle olisi soveltaa teoreettista tietoa käytäntöön. Tietoturvaprosessi integroitaisiin osaksi ohjelmistotuotantoprosessia ja menetelmien avulla tuotettaisiin turvallinen ohjelmisto tai sovellus. Ohjelmistoprojektin jälkeen arvioitaisiin ohjelmiston tietoturvaa ja tietoturvaprosessin mielekkyyttä.

LÄHTEET

Bart De Win 2013. Secure Development Lifecycles (SDLC). Osoitteessa http://secappdev.org/handouts/2013/Bart%20De%20Win/SecAppDev2013%20-%20SDLC%20Session%20Bart%20De%20Win%20v1.0.pdf. 18.06.2014.

F-Secure 2013. Mikko Hyppönen: Vuosi 2013–Kyperhyökkäysten huippuvuosi? Osoitteessa https://www.youtube.com/watch?v=lrmmEFq1IfA.

Haikala, I. – Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki: Talentum Media Oy.

Haikala, I. – Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Helsinki: Talentum Media Oy.

Howard, M. – Leblanc, D. 2004. Ohjelmoijan tietoturvaopas. Helsinki: Edita Prima Oy.

Howard, M. – Lipner, S. 2006. The Security Development Lifecycle. Redmond, Washington: Microsoft Press.

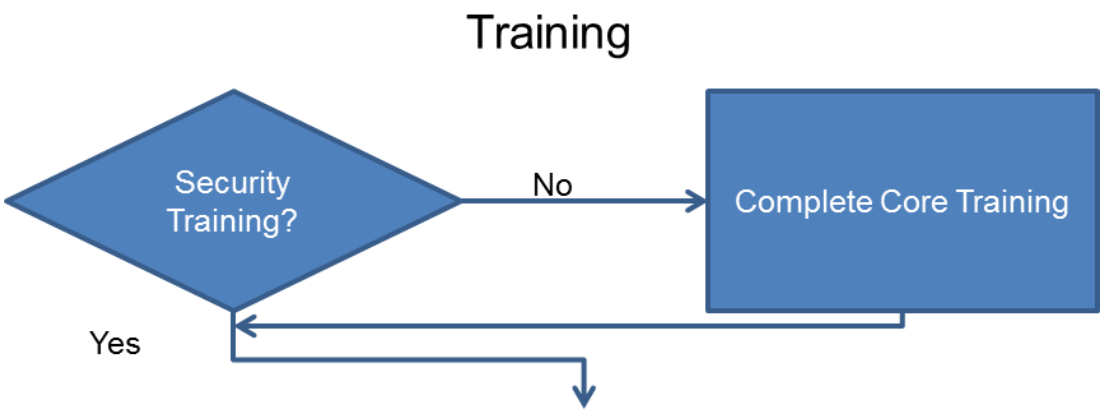
Microsoft 2014a. Microsoft Security Development Lifecycle (SDL) Process Guidance – Version 5.2. Osoitteessa http://www.microsoft.com/en-us/download/details.aspx?id=29884. 18.06.2014.

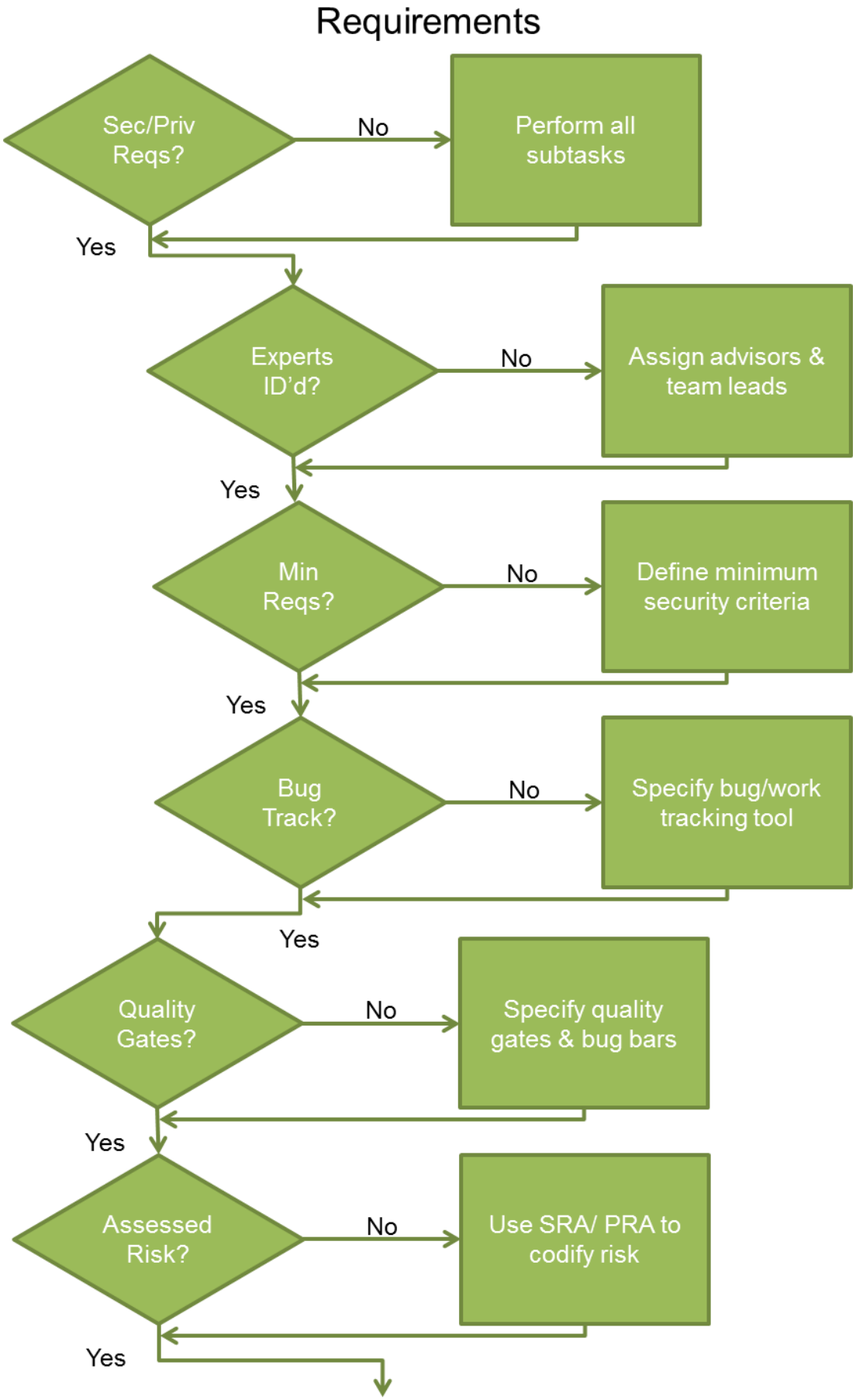
Microsoft 2014b. Simplified Implementation of the Microsoft SDL. Osoitteessa http://www.microsoft.com/en-us/download/details.aspx?id=12379. 04.07.2014.

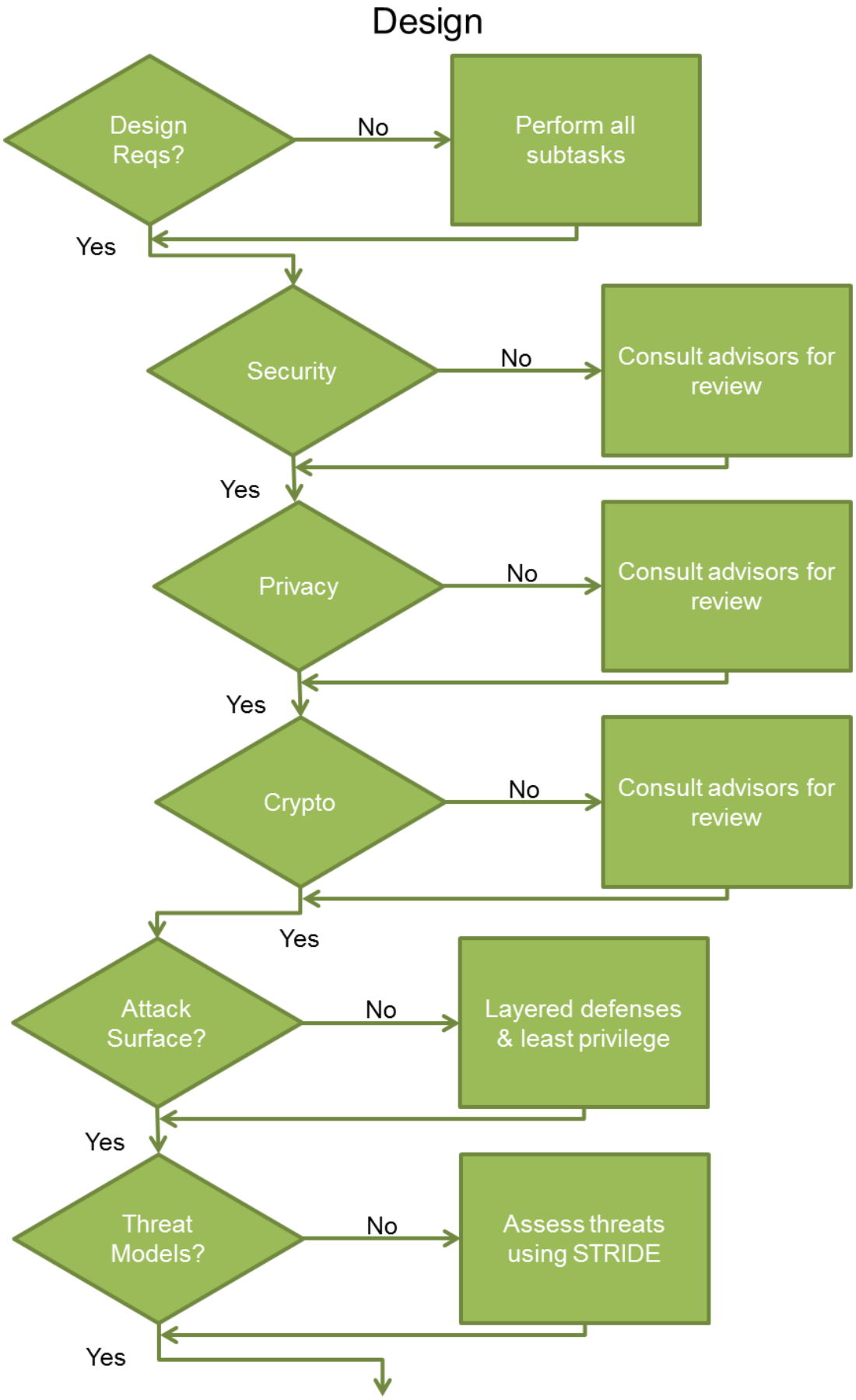
Risk Based Security 2013. Data Breach Quickview. Osoitteessa https://www.riskbasedsecurity.com/reports/2013-DataBreachQuickView.pdf.

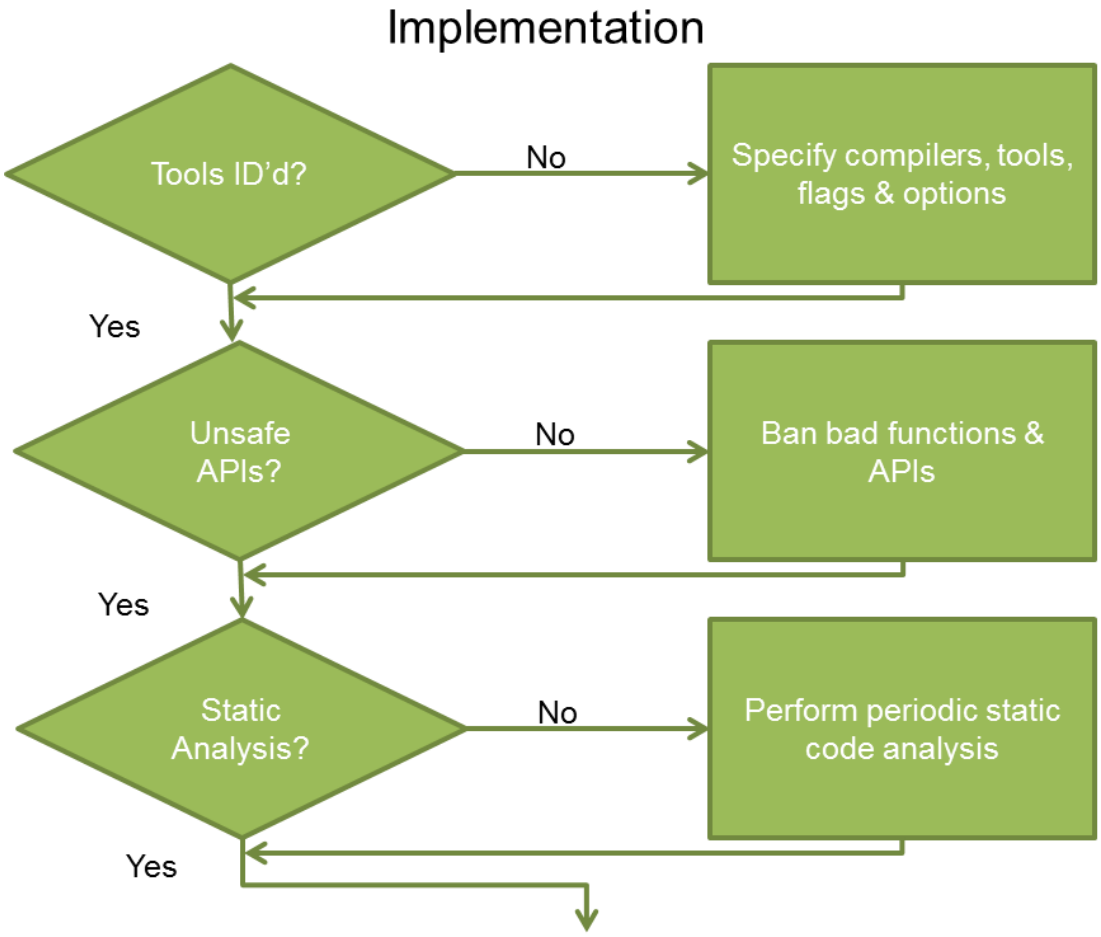
LIITTEET

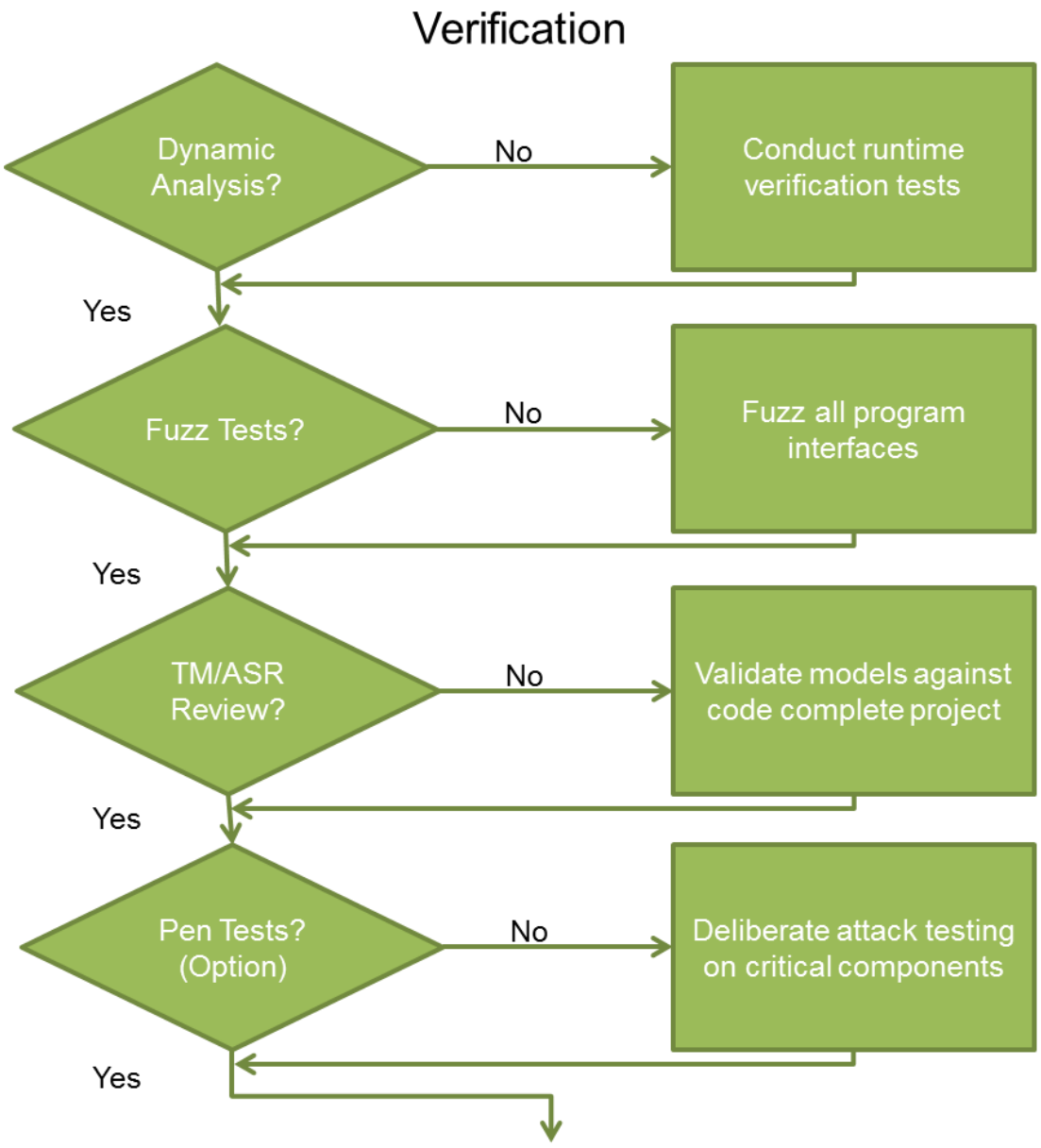
SDL-PROSESSIN VUOKAAVIO Liite 1

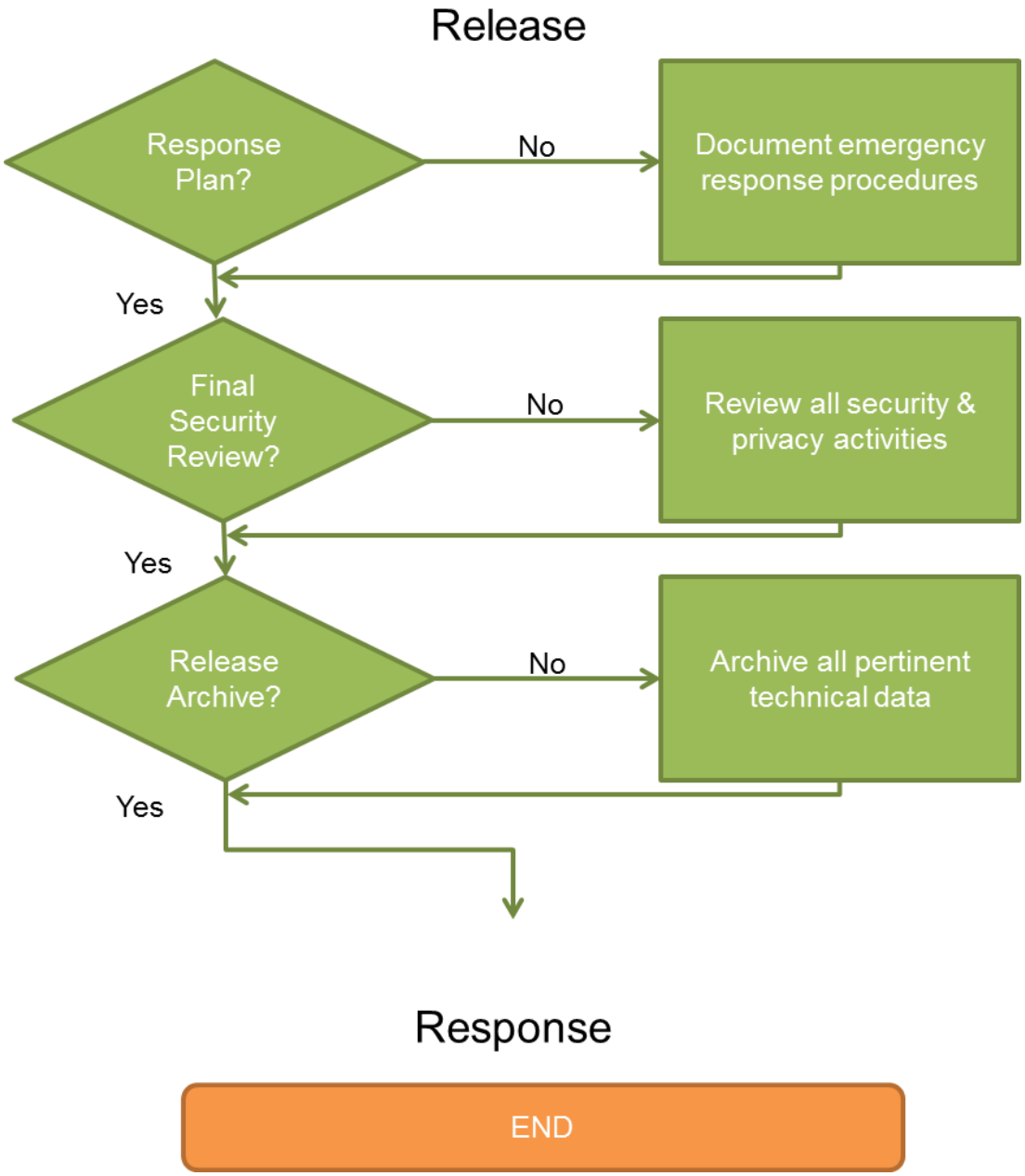



Requirements
Sec/Priv Reqs?
No
Perform all subtasks
Yes
Experts ID'd?
No
Assign advisors & team leads
Yes
Min Reqs?
No
Define minimum security criteria
Yes
Bug Track?
No
Specify bug/work tracking tool
Yes
Quality Gates?
No
Specify quality gates & bug bars
Yes
Assessed Risk?
No
Use SRA/ PRA to codify risk
Yes


Design
Design Reqs?
No
Perform all subtasks
Yes
Security
No
Consult advisors for review
Yes
Privacy
No
Consult advisors for review
Yes
Crypto
No
Consult advisors for review
Yes
Attack Surface?
No
Layered defenses & least privilege
Yes
Threat Models?
No
Assess threats using STRIDE
Yes


Implementation
Tools ID'd?
No
Specify compilers, tools, flags & options
Yes
Unsafe APIs?
No
Ban bad functions & APIs
Yes
Static Analysis?
No
Perform periodic static code analysis
Yes


Verification
Dynamic Analysis?
No
Conduct runtime verification tests
Yes
Fuzz Tests?
No
Fuzz all program interfaces
Yes
TM/ASR Review?
No
Validate models against code complete project
Yes
Pen Tests? (Option)
No
Deliberate attack testing on critical components
Yes



Liite 1. SDL-prosessin eteneminen vuokaaviona (Microsoft 2014b, 17)