Tampereen yliopisto

Tuomas Hyypiä

OHJELMISTON TURVALLISUUDEN KEHITYKSEN MITTAAMINEN

AUTOMATISOIDUN PENETRAATIOTESTAUKSEN AVULLA

Kandidaatintutkielma
Johtamisen ja talouden tiedekunta
Tarkastaja: Hannele Väyrynen
Joulukuu 2022

TIIVISTELMÄ

Tuomas Hyypiä: Ohjelmiston turvallisuuden kehityksen mittaaminen automatisoidun penetraatiotestauksen avulla
Measuring software security progress with automated penetration testing
Kandidaatintutkielma
Tampereen yliopisto
Johtamisen ja talouden tiedekunta
Teknis-taloudellinen kandidaattiohjelma
Joulukuu 2022

Ohjelmistoihin kohdistuvien turvallisuusuhkien määrä kasvaa kyberrikollisuuden yleistyessä. Ohjelmistoja tuottavat organisaatiot käyttävät paljon resursseja ohjelmiston turvallisuuden kehittämiseen estääkseen kyberuhkia. Ohjelmiston turvallisuuden parantamiseksi käytettyjen resurssien hyödyllisyyttä on alettu kyseenalaistamaan. Ohjelmiston turvallisuuden kehityksestä täytyy saada tietoa, jotta ohjelmiston turvallisuuden kehitystä voidaan verrata käytettyjen resurssien määrään. Ohjelmiston turvallisuuden kehityksestä voidaan kerätä lisää tietoa mittaamalla sitä. Ohjelmiston turvallisuuden kehityksen mittaamiselle ei ole kuitenkaan vakiintunut mitään selkeää tapaa. Automatisoitujen penetraatiotestien avulla saadaan tuotettua dataa ohjelmiston turvallisuudesta. Tutkimusongelmana tässä tutkimuksessa on automatisoidun penetraatiotestauksen soveltamismahdollisuudet ohjelmiston turvallisuuden kehityksen mittaamiseen. Tutkimusongelmasta on johdettu seuraava päätutkimuskysymys tälle tutkimukselle: Miten automatisoitua penetraatiotestausta voidaan soveltaa ohjelmiston turvallisuuden kehityksen mittaamisessa? Työssä keskeisinä käsitteinä ovat ohjelmiston turvallisuus, mittaaminen ja automatisoitu penetraatiotestaus.

Tämän työn tutkimus on toteutettu kirjallisuuskatsauksena systemaattista kirjallisuuskatsauksen mallia hyödyntäen. Lisäksi työssä on hyödynnetty lähdeaineiston hankinnassa erityisesti helmenkasvatusmenetelmää. Tutkimusaineisto koostuu tieteellisistä artikkeleista ja kirjoista erityisesti tietojenkäsittelytieteen ja tietotekniikan alalta, mutta erityisesti mittaamisen teoriaan on tuotu lähdemateriaalia myös muilta aloilta. Tutkimuksessa lähestyttiin tutkimusongelmaa mittaamisen ja automatisoidun penetraatiotestauksen näkökulmista. Tutkimuksessa havaittiin, että ohjelmiston turvallisuuteen liittyvää tutkimusta on tehty vain vähän, ja aiempaa tutkimusta automatisoidun penetraatiotestauksen hyödyntämisestä ohjelmiston turvallisuuden mittaamiseen ei löydetty ollenkaan tässä tutkimuksessa. Mittaamisen ja automatisoidun penetraatiotestauksen teorioiden perusteella automaattista penetraatiotestausta sovellettiin ohjelmiston turvallisuuden kehityksen mittaamiseen, mikä tuotti vastauksen tutkimuskysymykseen.

Tutkimuksessa osoitettiin, että ohjelmiston turvallisuus ja sen kehitys on mitattavissa sekä ohjelmiston turvallisuutta ja sen kehitystä on tarpeellista mitata. Lisäksi automatisoidun penetraatiotestauksen avulla mittaamisesta löydettiin erityisiä huomioitavia asioita. Huomioitavat asiat olivat niin mittaamiseen mahdollisuuksia tuovia kuin myös riskejä aiheuttavia tekijöitä. Tutkimuksen avulla pystyttiin osoittamaan, että automatisoidulla penetraatiotestauksella on hyödyntämismahdollisuuksia ohjelmiston turvallisuuden kehityksen mittaamisessa. Tutkimuksessa löydettyjen tulosten avulla on mahdollista lähteä kehittämään tiedolla johtamista esimerkiksi ohjelmistoa tuottavassa organisaatiossa, sillä tutkimuksessa löydettyjen tulosten avulla päätöksenteon tueksi on mahdollista tuoda lisää tietoa erityisesti ohjelmistokehityksessä. Automatisoidun penetraatiotestauksen hyödyntämismahdollisuudet ohjelmiston turvallisuuden kehityksen mittaamisessa riippuvat kuitenkin monesta tekijästä kuten ohjelmistosta ja organisaatiosta.

Avainsanat: Ohjelmiston turvallisuus, Turvallisuuden mittaaminen, Automatisoitu penetraatiotestaus, Tiedolla johtaminen, Päätöksenteko, Ohjelmistokehitys

ALKUSANAT

Tämä kandidaatintyö on tehty Tampereen yliopiston tietojohtamisen koulutusohjelman kandidaatintyön kurssilla syksyllä 2022. Työ on toteutettu kirjallisuuskatsauksena. Työssä tutkitaan ohjelmiston turvallisuuden kehityksen mittaamista automatisoidun penetraatiotestauksen avulla. Työn aihe kiinnosti itseäni, sillä sen avulla voidaan edistää tiedolla johtamista ohjelmistokehityksessä. Lisäksi aihe on ajankohtainen ohjelmistojen turvallisuuden korostuessa niihin kohdistuvien hyökkäyksien lisääntymisen vuoksi.

Haluan kiittää työni ohjaajaa Hannele Väyrystä, joka tuki tämän kandidaatintyön valmistumista auttamalla ja motivoimalla koko prosessin ajan. Lisäksi haluan kiittää kandidaatintyöryhmääni, joka on antanut palautetta, kommentteja ja kehitysideoita tämän työn tekemiseen. Lopuksi haluan kiittää myös perhettäni ja ystäviäni saamastani tuesta kandidaatintyön tekemisen aikana.

Tampereella, 26.11.2022

Tuomas Hyypiä

SISÄLLYSLUETTELO

.

1. JOHDANTO

1.1 Tutkimuksen tausta, perustelut ja tavoitteet

Ohjelmisto sisältää tietokoneelle tehtyjä ohjelmia, toimintamalleja ja dokumentaatiota. Ohjelmistotuote tarkoittaa valmista joukkoa tietokoneohjelmia, toimintamalleja ja dokumentaatiota. (IEEE Standard 1990) Tässä työssä käsitellään ohjelmistotuotetta ohjelmistona, sillä edellisen ohjelmiston ja ohjelmistotuotteen määritelmän perusteella ohjelmistotuote voidaan käsittää myös ohjelmistoksi.

Ohjelmiston turvallisuus tarkoittaa, että ohjelmistoa kehitetään siten, ettei ohjelmistoon olisi helppoa hyökätä (Jaatun 2018). Toisaalta ohjelmiston turvallisuudella voidaan tarkoittaa sitä, että ohjelmisto on tehty siten, että se toimii normaalisti, vaikka siihen kohdistuisi vahingoittamista yrittävä hyökkäys (McGraw 2004). Ohjelmiston turvallisuus voidaan Jaatunin (2018) ja McGrawn (2004) määritelmien perusteella määritellä niin hyökkäyksiä ennalta ehkäisevänä asiana kuin myös ohjelmiston kestävyytenä hyökkäyksiä vastaan. Tässä työssä ohjelmiston turvallisuutta käsitellään erityisesti ohjelmiston kestävyyden kannalta hyökkäyksiä vastaan. Jaatunin (2018) ja McGrawn (2004) määritelmien perusteella ohjelmiston turvallisuus on varsin laajalle ulottuva asia, mikä tekee siitä erityisen tärkeän ohjelmiston kannalta. Ohjelmiston turvallisuus ulottuu koko ohjelmiston laajuudelle (McGraw 2004), mikä lisää ohjelmiston turvallisuuden tärkeyttä entisestään.

Kyberrikollisuuden määrä on kasvanut viime vuosina merkittävästi muun muassa covid-19-pandemian takia, ja ohjelmiston turvallisuus on suuri ongelma nykyisissä ohjelmistoissa (Weir et al. 2021). Ohjelmiston turvallisuus on ajankohtainen aihe, sillä kyberrikollisuus on kasvanut merkittävästi viime vuosien aikana, ja siitä on tullut ongelma ohjelmistoille. Ohjelmistojen kehittäjät käyttävät resursseja kyberrikollisuuden ehkäisemiseksi, mutta ohjelmiston turvallisuutta edistävien toimien tehokkuutta on alettu kyseenalaistamaan yritysten johdossa (Weir et al. 2021). Ohjelmiston turvallisuus tuottaa ongelmia yritysten johdolle, sillä ohjelmiston turvallisuuteen kohdistettavat lisäresurssit eivät välttämättä tuota merkittäviä parannuksia ohjelmiston turvallisuuteen.

Ohjelmiston mittaamisella voidaan saada suuntaa antavia kuvia siitä, miten ohjelmisto on edistynyt sitä kehitettäessä (Chhillar & Gahlot 2017). Ohjelmiston turvallisuuden mittaamisen avulla voi seurata ohjelmiston turvallisuuteen käytettyjen resurssien kuten työ-

ajan tuottavuutta, kun verrataan ohjelmiston turvallisuuden kehittymistä ja siihen käytettyjen resurssien määrää. Ohjelmiston turvallisuuden mittaamisen tuottaman datan avulla saadaan tuotua lisää tietoa päätöksenteon tueksi, kun tämän datan perusteella rakennetaan mittareita. Ohjelmiston turvallisuus voi jäädä taka-alalle ohjelmistoa kehitettäessä, sillä ohjelmiston turvallisuus ei suurimmilta osin näy suoraan käyttäjälle, mutta esimerkiksi tunnistautuminen on käyttäjälle näkyvissä olevaa ohjelmiston turvallisuutta. Mittaaminen mahdollistaa organisaatiolle tärkeiden asioiden huomioimisen (Martinsuo et al. 2016, s. 133). Ohjelmiston turvallisuutta on mahdollista huomioida enemmän mittaamisen avulla ohjelmiston turvallisuuden ollessa ohjelmistoa tuottavalle organisaatiolle tärkeä asia.

Penetraatiotestaus on ohjelmistolle kehityksen loppuvaiheessa suoritettava testaus, jolla mitataan ohjelmiston turvallisuutta (McGraw 2004). Penetraatiotestauksessa testataan ohjelmiston turvallisuutta yrittämällä murtautua ohjelmistoon sisään. Nykypäivänä penetraatiotestausta on mahdollista automatisoida. Esimerkiksi Burp Suite on ohjelmisto, jonka avulla on mahdollista suorittaa automatisoitua penetraatiotestausta esimerkiksi pörrötyksessä (fuzzing), jossa ohjelmistoon kohdistetaan suuri kuorma haavoittuvuuksien etsimiseksi. Suuren kuorman kohdistaminen ohjelmistoon haavoittuvuuksien etsimiseksi on manuaalisesti työlästä ja aikaa vievää. Automatisoitu penetraatiotestaus ei tule ikinä korvaamaan ihmisen tarvetta penetraatiotestauksessa, mutta se voi tuoda siihen lisää hyötyjä, sillä sen avulla voidaan vähentää manuaalista työtä. (Automated penetration 2022) Automatisoitua penetraatiotestausta voisi mahdollisesti hyödyntää ohjelmiston turvallisuuden mittaamisessa, sillä penetraatiotestauksessa testataan ohjelmiston turvallisuutta.

Automatisoidun penetraatiotestauksen mahdollisuuksia ohjelmiston turvallisuuden kehityksen mittaamisessa on hyödyllistä selvittää. Kandidaatintyön tutkimuksen aiheeksi valikoitui tämän perusteella ohjelmiston turvallisuuden kehityksen mittaaminen automatisoitujen penetraatiotestien avulla. Aihe yhdistää tietojohtamisen opintoja monelta eri osa-alueelta ja yhdistää tietojohtamisen opinnot työelämän haasteisiin. Aihe on kiinnostava, sillä sen avulla saadaan tuotettua lisää tietoa päätöksenteon tueksi ohjelmistokehityksessä, jolloin ohjelmistokehityksen tiedolla johtamista voidaan kehittää. Mahdollisuus päätöksentekoa tukevan tiedon lisäämisestä korostaa tutkimuksen merkityksellisyyttä erityisesti tietojohtamisen alalla.

Tiedolla johtaminen on yksi keskeisimmistä asioista tietojohtamisen alalla. Laihonen et al. (2013, s.32) määrittelevät tiedolla johtamisen tiedon jalostamiseksi ja hyödyntämiseksi organisaation toiminnan johtamisessa. Tiedolla johtamisen avulla on mahdollista

tukea päätöksiä, sekä perustella tehtyjä päätöksiä, mikä lisää läpinäkyvyyttä organisaatiossa (Laihonen et al. 2013, s. 28). Tässä työssä tiedolla johtamisen rooli on erityisesti tutkimuksen tarpeellisuuden ja merkityksellisyyden perustelu. Tutkimuksen ollessa tarpeellinen ja merkityksellinen tietojohtamisen kannalta tutkimuksen tarkoitus on tuottaa myös uusia näkökulmia tiedolla johtamiseen erityisesti ohjelmistokehityksessä. Tutkimuksen tarpeellisuus ja merkityksellisyys tulee esiin tiedolla johtamisen kannalta päätöksenteon tueksi saatavan tiedon lisääntymisenä. ”Tiedon tehokkaalla hyödyntämisellä voidaan tukea päätöksentekoa tai vähentää arvoa tuottamatonta työtä” (Laihonen et al. 2013, s. 26). Työn tarkoituksena on tutkia ohjelmiston turvallisuuden kehityksen mittaamista automatisoitujen penetraatiotestien avulla. Ohjelmiston turvallisuuden kehityksen mittaamista voidaan soveltaa esimerkiksi käytettyjen resurssien tuottavuuden analysointiin, jonka avulla voidaan havaita esimerkiksi arvoa tuottamattomat resurssit.

Ohjelmiston turvallisuuden kehityksen mittaamisesta on verrattain vähän tietoa saatavilla, mikä osoittaa tutkimuksen tarpeellisuutta. Työn tarkasta aihealueesta ei löydy aikaisempaa tutkimusmateriaalia tutkimukseen valituista tietokannoista, mutta yleisemmin tutkimuksen aihealueesta, esimerkiksi ohjelmiston turvallisuudesta ja sen mittaamisesta, löytyy jonkin verran aikaisempaa tutkimusmateriaalia. Ohjelmiston turvallisuudesta ja sen kehityksen mittaamisesta tehty aiempi tutkimus on valtaosin tietojenkäsittelytieteen ja tietotekniikan alalla tehtyä tutkimusta. Aiempaa tutkimusta ohjelmiston turvallisuuden mittaamisesta löytyy myös eri tasoilta, muun muassa tieteellisistä artikkeleista ja opinnäytetöistä. Aiemman tutkimuksen vähyys ohjelmiston turvallisuuden selvittämisestä ohjelmistoa ulkoa päin testaamalla tukee tutkimuksen tarpeellisuutta ja selventää aiheesta tiedettäviä asioita tällä hetkellä.

Tutkimuksen tavoitteena on selvittää, miten automatisoitua penetraatiotestausta voidaan soveltaa ohjelmiston turvallisuuden mittaamiseen. Mittaaminen keskittyy erityisesti ohjelmiston turvallisuuden kehityksen mittaamiseen. Mittaamista on mahdollista käyttää ohjelmiston turvallisuuden kehittymisen selvittämiseksi, jotta ohjelmiston turvallisuuden tasosta ja siihen käytettyjen resurssien hyödyllisyydestä saadaan lisää tietoa päätöksenteon tueksi.

1.2 Tutkimuskysymys ja tutkimuksen rajaus

Tutkimuksen tavoitteena on selvittää automatisoidun penetraatiotestauksen soveltamismahdollisuuksia ohjelmiston turvallisuuden kehityksen mittaamiseen. Tutkimuksen tavoite on myös tutkimuksen keskeisin ongelma. Tutkimusongelman perusteella muodostettiin seuraavanlainen päätutkimuskysymys:

- Miten automatisoitua penetraatiotestausta voidaan soveltaa ohjelmiston turvallisuuden kehityksen mittaamisessa?

Päätutkimuskysymystä tukemaan on muodostettu kaksi alatutkimuskysymystä, joiden tavoitteena on auttaa ratkaisemaan päätutkimuskysymystä. Muodostetut alatutkimuskysymykset ovat seuraavat:

- Miten ohjelmiston turvallisuuden kehitystä voidaan mitata?
- Mitä asioita tulee huomioida automatisoitujen penetraatiotestien soveltamisessa ohjelmiston turvallisuuden mittaamiseen?

Ensimmäisen alatutkimuskysymyksen on tarkoitus auttaa tuomaan näkökulmia tutkimukseen erityisesti mittaamisen näkökulmasta. Toisen alatutkimuskysymyksen on tarkoitus tuoda tutkimukseen näkökulmia erityisesti automatisoidun penetraatiotestauksen käytöstä mittaamisessa. Toinen alatutkimuskysymys käsittelee automatisoidulle penetraatiotestaukselle ominaisia asioita mittaamisen teorian näkökulmasta. Alatutkimuskysymyksien tavoitteena on tuoda erilaiset näkökulmat päätutkimuskysymyksen ratkaisuun. Kahden eri näkökulman mukaan tuomisen tarkoituksena on tukea ratkaisun laadukkuutta. Alatutkimuskysymysten avulla päätutkimuskysymystä on lähestytty kahdesta eri näkökulmasta. Alatutkimuskysymysten avulla kahta eri näkökulmaa on tuotu lähemmäs päätutkimuskysymystä. Kun alatutkimuskysymyksiin on saatu vastaukset, lähestytään päätutkimuskysymystä alatutkimuskysymyksiin saatujen vastausten perusteella.

Ohjelmiston turvallisuutta voidaan tarkastella monesta eri näkökulmasta. Ohjelmiston turvallisuutta voidaan esimerkiksi tarkastella ohjelmistokehittäjän, johdon tai koko organisaation näkökulmasta. (Bishop 2003; Zalewski et al. 2011) Lisäksi ohjelmiston turvallisuuden mittaamista voi tehdä monilla eri tavoilla, joita ovat esimerkiksi standardien seuraaminen tai erilaiset analyysit kuten komponenttianalyysi (Saarela 2016). Tähän tutkimukseen on ohjelmiston turvallisuuden ja mittaamisen tavat ja näkökulmat valittu tukemaan työn aihetta ja tavoitetta.

Aiheen tarkastelunäkökulmaksi on valittu ohjelmistoa tuottava organisaatio, sillä tutkimuksen tavoitteena on tuottaa lisää tietoa päätöksenteon tueksi ohjelmistotuotannossa.

Mittaamisen kohteeksi on erityisesti valittu ohjelmiston turvallisuus, sillä ohjelmiston turvallisuuden mittaamisesta automatisoidun penetraatiotestauksen avulla ei löydy aiempaa tutkimusta tähän tutkimukseen valituista tietokannoista. Mittaaminen on rajattu keskittymään ohjelmiston turvallisuuden kehitykseen, jotta mittaaminen soveltuu erityisesti tiedolla johtamisen tueksi. Mittaamisen tarkentamiseksi mittaamisessa käsitellään vain automatisoidun penetraatiotestauksen avulla tehtävää mittaamista. Penetraatiotestaus on testauksen menetelmä, minkä takia ohjelmiston turvallisuuden käsittelyä rajataan vain ohjelmiston turvallisuuden testaamiseen. Rajaus automatisoituun penetraatiotestaukseen sulkee pois myös organisaation sisältä päin tehtävät tarkastelut ohjelmiston turvallisuuteen kuten esimerkiksi koodirivien määrän tarkastelun. Automatisoidun penetraatiotestauksen tueksi tutkimukseen on tuotu näkökulmia manuaalisesti suoritettavasta penetraatiotestauksesta. Automatisoitu penetraatiotestaus on valittu tutkimuksen kohteeksi, sillä automatisoitu penetraatiotestaus ei vaadi merkittäviä määriä henkilöresursseja. Automatisoidun penetraatiotestauksen avulla voidaan parantaa mittausten vertailukelpoisuutta, mikä on erityisesti kehityksen mittaamisen kannalta tärkeä asia. Tutkimuksesta on suljettu pois muut ohjelmiston mittaamiseen liittyvät näkökulmat, kuten ohjelmiston muiden osa-alueiden kuin turvallisuuden mittaaminen ja ohjelmiston turvallisuuden mittaaminen muilla menetelmillä.

Mittaukset antavat hetkellisiä tuloksia tietyistä asioita mittareiden antaessa aiempiin mittauksiin vertautuvaa analyysiä; mittaukset tuottavat raakadataa ja mittarit tuotetaan analysoimalla (Juneja et al. 2011). Tässä työssä käsitellään mittaamista ja mittareita Juneja et al. (2011) mukaan, eli suorituksen mittaamisella tarkoitetaan mittausdatan tuottamista ja mittareilla tarkoitetaan mittausten perusteella tehtyä analyysiä. Tutkimus on rajattu käsittelemään mittaamista, sillä työn tarkoituksena ei ole rakentaa mittaristoa, vaan perehtyä ohjelmiston turvallisuuden kehityksen mittaamisen mahdollisuuksiin automatisoitujen penetraatiotestien avulla. Työssä esitellään lyhyesti myös mittaristoa, sillä mittaamalla tuotetaan dataa mittaristolle (Juneja et al. 2011), minkä takia mittaamisessa on tärkeää ottaa huomioon asioita myös mittaristoista ja niiden rakentamisesta.

Rajaukset tukevat aiheeseen keskittymistä, jotta tutkimuksen tulokset vastaavat tutkimuskysymykseen mahdollisimman tarkasti. Rajausten tarkoitus on myös keskittää työtä alueeseen, josta tutkimusta löytyy erittäin rajallisesti. Lisäksi aihetta ja näkökulmia on rajattu työn laajuuden vuoksi.

1.3 Tutkimuksen rakenne

Tutkimus koostuu kuudesta eri luvusta. Ensimmäisen ja toisen luvun tarkoituksena on esitellä tutkimusta. Toisessa luvussa käsitellään tutkimusmenetelmää ja -aineistoa. Työn

teoriaosuus koostuu luvuista 3 ja 4. Kolmannessa luvussa perehdytään mittaamisen teoriaan ja mittaamisen haasteisiin. Kolmannessa luvussa mittaamisen teoriaa ja haasteita sovelletaan myös ohjelmiston turvallisuuden mittaamiseen. Neljännessä luvussa perehdytään automatisoituun penetraatiotestaukseen ja sen avulla mittaamiseen.

Viidennessä luvussa käsitellään automatisoidun penetraatiotestauksen soveltuvuutta ohjelmiston turvallisuuden mittaamiseen ja analysoidaan mahdollisia riskitekijöitä. Luvussa 5 esitellään vastaus tutkimusongelmaan, mikä tekee luvusta keskeisen tutkimuksen kannalta. Kuudennessa luvussa on tutkimuksen yhteenveto, jossa kerrotaan tutkimuksen keskeisimmät asiat, arvioidaan tutkimuksen tuloksia ja esitellään mahdollisia aiheita jatkotutkimukselle.

2. TUTKIMUSMENETELMÄN KUVAUS

2.1 Tutkimusmenetelmä

Kandidaatintyön tutkimusmenetelmäksi valittiin kirjallisuuskatsaus. Kirjallisuuskatsauksen vahvuudet tutkimusmenetelmänä ovat systemaattisuus, täsmällisyys ja toistettavuus (Fink 2014, s. 13–14). Kirjallisuuskatsaukseen käytettiin Finkin (2014, s. 3–5) seitsemän kohdan mallia, sillä sen avulla tutkimusprosessissa pystyttiin etenemään systemaattisesti. Malli koostuu seuraavista kohdista:

1. tutkimuskysymyksen asetus
2. kirjallisuuden ja tietokantojen valinta
3. hakusanojen ja -lauseiden valinta
4. käytännön hakukriteerien valinta
5. metodologinen rajaus
6. katsauksen tekeminen
7. tulosten syntetisointi Fink (2014, s.3–5).

Lisäksi kirjallisuuskatsauksen lähdemateriaalin hankkimiseksi hyödynnettiin helmenkasvatusmenetelmää, jossa uutta lähdemateriaalia etsitään jonkin aiheeseen soveltuvan julkaisun lähdeluettelosta (Schlosser et al. 2006; Tiedonhaun opas 2022).

Tutkimuksessa edettiin Finkin (2014, s. 3–5) mallin mukaisesti. Ensimmäisessä kohdassa tutkimuskysymykset asetettiin tutkimusongelman perusteella. Tutkimuskysymykset esiteltiin luvussa 1.2. Toisessa kohdassa pääasialliseksi lähdemateriaaliksi valittiin tieteelliset artikkelit ja vaihtoehtoiseksi lähdemateriaaliksi valittiin kirjat. Pääasialliseksi lähdemateriaaliksi valikoituivat tieteelliset artikkelit, sillä tieteellisten artikkelien saatavuus on hyvä ja niiden avulla saadaan tuotua tutkimukseen enemmän aineistoa erilaisista näkökulmista. Tutkimusaineiston hankintaan valittiin neljä eri hakupalvelua, jotka olivat Andor, Finna, Scopus ja ACM Digital Library. Hakupalveluita valittiin useita lähdemateriaalin riittävyyden ja laadukkuuden varmistamiseksi.

Kolmannessa kohdassa hakusanat ja -lausekkeet muodostuivat tutkimusongelman aihepiirin mukaisesti keskeisistä käsitteistä. Hakusanoiksi ja -lausekkeiksi määriteltiin seuraavat termit: ”software security”, ”metrics”, ”measuring”, ”automation” ja ”penetration testing”. Määritellyistä hakusanoista ja -lausekkeista muodostettiin erilaisia yhdis-

telmiä Boolen operaattoreiden avulla. Neljännessä kohdassa lähdeaineistoa rajattiin hakukriteerin avulla. Hakukriteeriksi määriteltiin lähdemateriaalin julkaisuajankohdaksi vuonna 2015 ja sen jälkeen julkaistu lähdemateriaali.

Viidennessä kohdassa määritettiin tutkimusaineistolle metodologinen rajaus. Lähdeaineiston heikon saatavuuden vuoksi laadullisia rajauksia ei viidennessä vaiheessa tehty, sillä lähdeaineiston laatu pyrittiin varmistamaan jo mallin toisessa ja neljännessä kohdassa hakupalveluiden, lähdemateriaalin julkaisutyypin ja aikarajauksen perusteella. Tutkimusaineiston metodologista rajausta pyrittiin kuitenkin toteuttamaan siten, että lähdeaineistossa termistö olisi määritelty mahdollisimman samankaltaisesti. Kuudennessa kohdassa arvioitiin hakutuloksia ja rajattiin hakutuloksista epärelevantteja lähteitä pois. Rajaus kohdistui erityisesti sellaiseen tutkimusaineistoon, jossa tämän työn tutkimusaihetta käsiteltiin eri näkökulmista kuin tässä tutkimuksessa. Finkin (2014, s. 5) mallin mukaisesti lopuksi seitsemännessä kohdassa luotiin synteesi kerätyn tutkimusaineiston perusteella.

2.2 Tutkimusaineisto

Tutkimusaineiston hankintaan käytettiin neljää eri hakupalvelua, jotka olivat Andor, Finna, Scopus ja ACM Digital Library. Hakupalveluita valittiin useampi kappale, jotta lähdemateriaali olisi laadukasta ja sitä olisi riittävästi. Eri hakupalvelut valittiin niiden monimuotoisuuden vuoksi, ja juuri kyseiset hakupalvelut valikoituivat yliopiston käyttöoikeuksien perusteella. Esimerkiksi ACM Digital Library sisältää enemmän materiaalia tietotekniikan ja tietojenkäsittelytieteen aloilta ja Finna sisältää erityisesti opinnäytetöitä.

Lähdemateriaalin julkaisuaika rajattiin siten, että lähdemateriaali on julkaistu vuonna 2015 tai sen jälkeen lähdemateriaalin ajantasaisuuden varmistamiseksi. Osa tutkimuksen lähdeaineistosta oli myös ennen vuotta 2015 julkaistua, sillä kyseisistä aiheista ei ollut saatavilla uudempaa aineistoa. Ennen vuotta 2015 julkaistua lähdeaineistoa on löydetty pääsääntöisesti helmenkasvatusmenetelmän avulla aikarajaukseen sopivista lähteistä, mikä tukee ennen vuotta 2015 julkaistun lähdemateriaalin luotettavuutta. Lähdeaineiston aikaväliksi valikoitui kyseinen aikaväli, sillä lähdemateriaalia löytyi aikarajauksella 2015–2022 sopivasti, ja aineistoa perusteorioista haettiin helmenkasvatusmenetelmän avulla aikavälille 2015–2022 sijoittuvasta aineistoista. Lähdemateriaalin laadukkuuden ja riittävyyden tarkoitus oli mahdollistaa työn hyvä lopputulos.

Taulukossa 1 on esitelty Scopus- ja ACM Digital Library -tietokantoihin syötettyjä tutkimuksen keskeisistä käsitteistä muodostettuja hakulausekkeita ja niiden tuottamia hakutulosten määriä.

Taulukko 1. *Hakulausekkeet ja niiden tuottamien tuloksien määrät*

Hakulauseke	Scopus	ACM DL
"software" AND "security" AND "penetration testing"	256	647
"software security" AND "metrics"	133	580
"software security" AND ("penetration testing" OR "metrics")	249	893
"penetration testing" AND "metrics"	36	217
"software security" AND "metrics" AND "penetration testing"	2	48

Taulukon 1 hakutulosten määrästä voidaan nähdä, että mitä tarkemmin hakua rajataan tutkimusongelmaan, sitä vähemmän siihen liittyvää materiaalia on saatavilla. Taulukon 1 tarkoitus on esitellä työn aihepiirin aineiston rajallisuutta. Tämän tutkimuksen tarkasti rajatusta aiheesta ei löytynyt aiempaa tutkimusta ollenkaan valituista tietokannoista.

Tutkimuksessa käytetty aineisto sisälsi niin teoreettista kuin empiiristäkin tutkimusta. Tutkimus perustui kuitenkin vahvasti teoriaan, joten teoreettista lähdemateriaalia hyödynnettiin enemmän. Myös joistakin empiirisistä tutkimusaineistoista hyödynnettiin vain teoreettista osuutta. Lähdemateriaali sijoittui valtaosin tietojenkäsittelytieteen ja tietotekniikan alalle. Myös muiden alojen julkaisuista hyödynnettiin lähdeaineistoa erityisesti mittaamisen aihepiiristä.

Valtaosa tutkimuksen lähdeaineistosta löydettiin määriteltyjen hakulausekkeiden perusteella. Lähdeaineiston hankinnassa käytettiin yleisesti määritettyjen hakusanojen ja -lauseiden lisäksi joihinkin asioihin tarkennettuja hakusanoja ja -lauseita. Esimerkiksi mittaamisen perusteorian hankinnassa käytettiin seuraavia hakusanoja ja -lausekkeita: ”laskentatoimi”, ”päätöksenteko” ja ”suorituksen mittaaminen”. Tutkimuksessa hyödynnettiin paljon helmenkasvatusmenetelmää, sillä sen avulla löytyi paljon tutkimuksen aihepiiriin soveltuvaa lähdeaineistoa.

Tutkimusaineiston valitsemisessa hyödynnettiin erityisesti metodologista rajausta, jossa määriteltiin, että tutkimusaineistossa termit tulisi olla määritelty mahdollisimman samankaltaisesti. Lisäksi tutkimusaineiston valitsemisessa tarkasteltiin tutkimusaineiston olennaisuutta tämän tutkimuksen kontekstissa. Tähän tutkimukseen sopivaa ja oleellista tutkimusaineistoa pyrittiin valitsemaan mahdollisimman paljon, jotta tutkimukseen saataisiin mahdollisimman paljon erilaisia näkökulmia. Tutkimusaineiston määrä pysyi sopivana tutkimukseen laajuuteen verrattuna, vaikka tutkimusaineistoa valittiin monesta eri näkökulmasta, sillä aiheesta saatavilla olleen tutkimusaineiston määrä oli varsin rajalli-

nen. Tutkimukseen otettiin mukaan laajasti lähdeaineistoa, sillä tutkimuksen tarkasta aiheesta ei löytynyt aiempaa tutkimusta, jolloin tutkimuksen tarkkaan aiheeseen ja eri näkökulmiin jouduttiin tuomaan sisältöä useista eri lähteistä.

Taulukkoon 2 on listattu työn kannalta keskeisimmät lähteet sekä tietokanta ja hakusana tai -lauseke, jonka avulla lähde on löytynyt.

Taulukko 2. *Tutkimuksen keskeisimmät lähteet*

Tietokanta	Hakulauseke	Kirjoittaja(t)	Teoksen nimi
ACM Digital Library	”software security”	Rotella (2018)	Software Security Vulnerabilities: Baselining and Benchmarking
ACM Digital Library	"security" AND "metrics" AND "automation"	Zaber & Nair (2020)	A framework for automated evaluation of security metrics
Finna	”measuring” AND "software security"	Saarela (2016)	Measuring software security from the design of software
Andor	"laskentatoimi" AND "päätöksenteko"	Suomala et al. (2011)	Laskentatoimi johtamisen tukena
Scopus	"performance measurement" AND "problems"	Lewis (2015)	The politics and consequences of performance measurement

Taulukosta 2 voidaan havaita, että työssä on ollut useita keskeisiä lähteitä. Tutkimuksessa yhdisteltiin useaa asiaa, minkä vuoksi usea lähde oli keskeinen tutkimuksen kannalta. Rotellan (2018) teoksesta *Software Security Vulnerabilities: Baselining and Benchmarking* tuotiin työhön erityisesti teoriaa vertailukohdista ohjelmiston turvallisuuden kehityksen mittaamisessa, sillä vertailukohdat ovat tärkeä osa kehityksen mittaamista. Zaberin & Nairin (2020) teoksesta *A framework for automated evaluation of security metrics* tuotiin työhön erityisesti näkökulmia automaatiosta mittaamisessa, sillä työn keskeisenä asiana oli automatisoidut penetraatiotestit. Automatisoitujen penetraatiotestien käytöstä mittaamisessa ei löytynyt lähdeaineistoa, joten lähdeaineistoa piti soveltaa automatisoidun mittaamisen puolelta. Saarelan (2016) teoksesta *Measuring software security from the design of software* tuotiin vain vähän näkemyksiä tähän tutkimukseen, sillä teos on diplomityö. Teos oli kuitenkin yksi tutkimuksen keskeisimpiä lähteitä, sillä helmenkasvatusmenetelmän avulla kyseisestä teoksesta saatiin merkittävä määrä lähdemateriaalia tähän tutkimukseen erityisesti ohjelmiston turvallisuudesta. Suomalan et al. (2011) *Laskentatoimi johtamisen tukena* teoksesta tuotiin tähän tutkimukseen suori-

tuksen mittaamisen perusteoriaa sekä mahdollisuuksia ja riskejä. Lewisin (2015) teoksesta *The politics and consequences of performance measurement* tuotiin tähän tutkimukseen erityisesti näkemyksiä suorituksen mittaamiseen liittyvistä ongelmista.

3. OHJELMISTON TURVALLISUUDEN MITTAAMINEN

Ohjelmiston turvallisuus tarkoittaa sitä, että ohjelmisto on tehty siten, että se toimii normaalisti, vaikka siihen kohdistuisi vahingoittamista yrittävä hyökkäys. Ohjelmiston turvallisuus on koko ohjelmiston laajuudelle ulottuva asia, joka kattaa sekä ohjelmiston turvallisuusmekanismit kuten pääsyoikeudet että ohjelmiston suunnittelun turvallisuusnäkökulman kuten ohjelmiston suunnitellun vahvuuden hyökkäyksiä vastaan. (McGraw 2004) Ohjelmiston turvallisuudesta erityisen haasteellista tekee nykypäivänä se, että uusia ominaisuuksia on alettu julkaisemaan yhä nopeampaa tahtia (Rotella 2018). Ohjelmiston turvallisuus saatetaan myös sekoittaa turvallisuusohjelmistoon, mutta nämä kaksi ovat täysin eri asioita, eikä näitä kahta tule sekoittaa keskenään (McGraw 2004).

Ohjelmiston turvallisuuden määrittely riippuu tarkasteltavasta näkökulmasta, joka voi olla esimerkiksi ohjelmiston kehittäjä tai päätöksentekijä (Bishop 2003; Zalewski et al. 2011). Päätöksenteon näkökulmasta ohjelmiston turvallisuus tarkoittaa usein riskien toteutuessa aiheutuvan haitan ja riskien ehkäisemisen kustannusten vertailua (Jaquith 2007). Tässä työssä ohjelmiston turvallisuutta ja sen kehittymisen mittaamista käsitellään päätöksentekijän näkökulmasta organisaatiotasolla.

3.1 Suorituksen mittaaminen ja haasteet

Asioita ei ole aina perusteellista ilmaista suoraan rahamääräisinä kustannuksina tai tuottoina. Tuotekehityksessä tehtäviä valintoja voi perustella esimerkiksi suorituksen mittaamisen tuottamien tulosten perusteella. Suorituksella tarkoitetaan mitattavan asian hetkellistä suoritusta tietyssä tilanteessa. Suorituksen mittaamisessa on usein keskeisinä kohteina asiat, joiden uskotaan olevan merkittäviä pidemmällä tarkastelujaksolla, mutta niiden muutokset eivät ole välttämättä huomattavissa lyhyemmällä aikavälillä. Suorituksen mittaamisessa usein keskeisinä kohteina olevilla asioilla on myös usein vain välillinen vaikutus taloudelliseen tulokseen. Esimerkiksi suorituksen mittaamisessa usein keskeisenä kohteena oleva asia on laatu tai kokemus laadusta. (Martinsuo et al. 2016, s. 132–133)

Suorituksen mittaamisen avulla on mahdollista tuoda niin organisaation sisäisesti kuin ulkoisesti näkyvämmäksi asioita, joita ei ole esimerkiksi helppo muuntaa numeroiksi tai jotka eivät ole selkeästi näkyviä toiminnassa. Suorituksen mittaamisen avulla voidaan mahdollistaa myös seuraavia asioita:

- motivointi
- palkitseminen ja tunnustusten jakaminen
- ei-toivotun toiminnan karsiminen ja toivotun toiminnan lisääminen
- tehtävän arvon ja merkityksen viestiminen.

Tehtävän arvon ja merkityksen viestiminen voi sisältää viestintää molempiin suuntiin. Organisaatiolle tärkeitä asioita voidaan tuoda näkyvämmiksi mittaamisen avulla, ja työntekijät voivat perustella merkitystään ja suorituksen tasoaan mittaamisen avulla. (Suomala et al. 2011, s. 133–136)

Suorituksen mittaamisen avulla on mahdollisuus määrittää erilaisia mittaristoja (Juneja et al. 2011). Mittari on työkalu, jonka avulla voidaan helpottaa päätöksentekoa datan keräämisen, analysoinnin ja raportoinnin avulla (Ahmed 2016). Hyvä mittari on muun muassa tarkka, mitattavissa ja saavutettavissa oleva, toistettava ja aikariippuvainen (Juneja et al. 2011). Mittaristoa rakennettaessa tulee ensiksi määrittää mitattava asia selkeästi. Mitattavan asian selkeän määrityksen jälkeen tulee määrittää tapa kuvailla mitattavaa asiaa laadun näkökulmasta. Kun tapa kuvailla mitattavaa asiaa laadun näkökulmasta on määritetty, tulee määrittää itse mittari, joka voi olla pelkästään mitattava asia kuvailtuna laadun näkökulmasta, mutta toisaalta se voi myös olla useamman laadun näkökulmasta mitattavan asian yhdistelmä. Mittarin määrittämisen jälkeen tulee vielä määrittää mittausprosessi. (Zalewski et al. 2014) Mittaamisen tulee olla myös toistettavissa (Juneja et al. 2011). Mittaamisen toistettavuuden tärkeyden vuoksi mittaamisen toistettavuus tulee ottaa huomioon mittaamisprosessia rakentaessa.

Suorituksen mittaaminen sisältää haasteita, joihin tulisi kiinnittää huomiota. Eräänä mittaamisen yleisenä haasteena on sen aiheuttamat kustannukset. Lisäksi suorituksen mittaamisella voi olla kielteisiä vaikutuksia. Esimerkiksi vain etukäteen johdon määrittämiin mitattaviin asioihin keskittyminen voi olla organisaatiolle aiheutuva kielteinen vaikutus mittaamisesta. Vain mitattavaan suoritukseen keskityttäessä esimerkiksi asioiden tekeminen hyvin ja hyvässä yhteishengessä voi jäädä huomioimatta. (Suomala et al. 2011, s. 136)

Lewisin (2015) mukaan suorituksen mittaamisessa on tyypillisesti 5 yleistä ongelmaa, jotka ovat seuraavat:

- mittaaminen ei tavoita haluttuja asioita
- mittaaminen on puutteellista
- mittaamista ei käytetä oikein tai olleenkaan
- mittaaminen aiheuttaa ei toivottua toimintaa
- mittaaminen aiheuttaa ei toivottuja muutoksia suorituksessa.

Lewisin (2015) esittämät ongelmat mittaamisen aiheuttamasta ei toivotusta toiminnasta ja ei toivotuista muutoksista suorituksessa kuuluvat Suomalan et al. (2011) esittämään ongelmaan mittaamisen kielteisistä vaikutuksista. Ei toivotun toiminnan ja ei toivottujen muutosten huomioiminen on erityisen tärkeää ottaa huomioon, kun mittaamista tarkastellaan johdon näkökulmasta, sillä muihin suorituksen mittaamisen tyypillisiin ongelmiin johto voi omalla toiminnallaan vaikuttaa vahvasti. Mittaamisen aiheuttama ei toivottu toiminta voi näkyä esimerkiksi vain mitattaviin asioihin keskittymisenä, mikä aiheuttaa mittaamisen tyypillisistä ongelmista löytyviä ei toivottuja muutoksia suorituksessa. Mittaamisen mahdollisesti aiheuttama ei toivottu toiminta ja ei toivotut muutokset suorituksessa ovat siis yleensä työntekijälähtöisiä, joten johdon on erityisen tärkeää kiinnittää niihin huomiota mittaamista suunnitellessa.

3.2 Suorituksen mittaamisen soveltaminen ohjelmiston turvallisuuteen

Ohjelmiston turvallisuuteen liittyviä valintoja mietittäessä perusteluita voi pohjata mitattuihin suorituksiin, sillä ohjelmiston turvallisuuden vaikutusten havaitseminen voi olla vaikeaa, sillä ohjelmiston turvallisuuden muutoksen taloudellinen vaikutus voi olla haastavaa arvioida. Taloudellisen vaikutuksen arvioinnin haastavuutta tukee ohjelmiston turvallisuudessa löytyvät piirteet, joiden perustella sen suoritusta olisi kannattavaa mitata; esimerkiksi ohjelmiston turvallisuuden heikkenemisen vaikutukset eivät välttämättä näy heti asiakkailla. Lisäksi ohjelmiston turvallisuuden heikkeneminen ei näy välttämättä suoraan taloudellisessa tuloksessa, mutta ohjelmiston turvallisuuden heikkeneminen voidaan selkeästi havaita taloudellisessa tuloksessa esimerkiksi asiakasmäärän vähenemisen aiheuttamina tulojen vähenemisinä.

Suomala et al. (2011, s. 133) määrittelevät kriittiseksi menestystekijäksi asian, joka on menestymisen kannalta välttämätön tai hyvin tärkeä. Kriittisten menestystekijöiden mittaaminen on erityisen tärkeää (Suomala et al. 2011, s. 137). Ohjelmiston turvallisuus

takaa ohjelmistolle mahdollisuuden menestyä, sillä oletusarvona ohjelmistoa hankkiessa on sen turvallisuus, joten ohjelmiston turvallisuus voidaan käsittää kriittiseksi menestystekijäksi. Ohjelmiston turvallisuus on siis asia, jota olisi tärkeä mitata. Joidenkin näkemyksien mukaan turvallisuus ei ole kuitenkaan ollenkaan mitattava asia (Torgersen 2007; Stolfo et al. 2011). Zalewski et al. (2011) toteavat kuitenkin ohjelmiston turvallisuuden mittaamisen sisältävän mahdollisuuksia, ja antavat esimerkkejä ohjelmiston turvallisuuden määrittämisestä. Ohjelmiston turvallisuuden määrittämisen voi toteuttaa esimerkiksi teoreettisesti, kokeellisesti tai simuloimalla (Zalewski et al. 2011). Tässä tutkimuksessa keskitytään ohjelmiston turvallisuuden mittaamiseen automatisoidun penetraatiotestauksen avulla, minkä takia ohjelmiston turvallisuuden tarkastelunäkökulmana on kokeellinen lähestymistapa. Ohjelmiston turvallisuuden määrittäminen kokeellisella mittaustavalla keskittyy erityisesti mittaamiseen ja mittaristoihin (Zalewski et al. 2011), mikä tukee kokeellisen näkökulman valitsemista tässä tutkimuksessa.

Turvallisuuden mittaamista käytetään terminä nykyään usein, mutta sille ei ole tarkkaa määrittelyä (Ahmed 2016; Juneja et al. 2011), mikä tukee sitä, että turvallisuuden mittaamiseen on monia eri näkemyksiä. Eri näkemykset voivat olla esimerkiksi ohjelmistokehittäjän tai johdon näkemykset turvallisuuden mittaamiseen. Toisaalta erilaiset näkemykset voivat liittyä myös ohjelmiston turvallisuuden mitattavuuteen. Tässä työssä ohjelmiston turvallisuuden mittaamista käsitellään johdon näkökulmasta ja tutkitaan ohjelmiston turvallisuuden mittaamisen mahdollisuutta erityisesti automatisoidun penetraatiotestauksen avulla.

Zalewskin et al. (2014) mukaan yksi tärkeimmistä asioista ohjelmiston turvallisuutta mitattaessa on määrittää, miksi jotakin mitataan. Kuten Martinsuo et al. (2016, s. 132) esittävät, suoritusta voidaan mitata esimerkiksi tukemaan valintojen perusteluja päätöksenteossa. Myös Zalewskin et al. (2014) mukaan mittaamisen avulla voidaan parantaa päätöksentekoa. Mittariston avulla on mahdollista saada selville tietoa muun muassa turvallisuuden kehittymisestä ja turvallisuustason riittävyydestä (Juneja et al. 2011), minkä avulla on mahdollista parantaa päätöksentekoa. Tässä työssä mittaamista käsitellään näkökulmasta, jossa mittaamisen tarkoitus on tuottaa lisää tietoa päätöksenteon tueksi ohjelmistokehityksessä.

Rotellan (2018) mukaan ohjelmiston turvallisuuden kehityksen mittaaminen on mahdotonta ilman vertailukohtia ja suorituskykymittausta. Vertailukohtien ja suorituskykymittauksen avulla voidaan tehdä seuraavia asioita:

- vähentää merkittäviä haavoittuvuuksia ohjelmistossa
- vähentää haavoittuvuuksien kokonaismäärää ohjelmistossa
- määrittää tehokkaita tapoja ehkäistä haavoittuvuuksia
- määrittää kannattamattomia toimintatapoja ohjelmiston turvallisuuden lisäämiseen
- levittää tehokkaammin parhaita tapoja ohjelmiston turvallisuuden kehittämiseksi (Rotella 2018).

Ohjelmiston turvallisuutta mitatessa voidaan pitää erityisen tärkeinä vertailukohtia, joita on mahdollista tuottaa muun muassa toteuttamalla suorituskykymittauksia toistuvasti tietyn ajan välein. Aiemmat vertailukohdat ovatkin erityisen tärkeitä haavoittuvuuksien hallitsemisen seuraamiseksi (Rotella 2018). Vertailukohtia voidaan pitää tärkeinä myös ohjelmiston turvallisuuden kehitystä mitattaessa, sillä vertailukohtien avulla on mahdollista seurata ohjelmiston turvallisuuden kehitystä. Esimerkiksi Rotellan (2018) esille tuoma kannattamattomien toimintatapojen määritys ohjelmiston turvallisuuden lisäämisessä vertailukohtien avulla on erityisesti ohjelmiston turvallisuuden kehityksen seuraamista siihen käytettyjen resurssien suhteen.

Mittaamisen yleisiä haasteita voidaan pitää sovellettavina myös ohjelmiston turvallisuuden mittaamisessa, sillä mittaamisen yleiset haasteet pätevät kaikkeen mittaamiseen. Ohjelmiston turvallisuuden mittaamisessa erityisenä haasteena on se, ettei edes ohjelmistoa tuottava taho voi tietää haavoittuvuuksien kokonaismäärää (Zalewski et al. 2014). Tietämättömyys haavoittuvuuksien kokonaismäärästä ohjelmiston turvallisuutta mitattaessa on erityinen haaste ohjelmiston turvallisuutta mitattaessa, minkä vuoksi tietämättömyys haavoittuvuuksien kokonaismäärästä tulee ottaa huomioon ohjelmiston turvallisuutta mitattaessa.

4. AUTOMATISOITU PENETRAATIOTESTAUS MITTAAMISESSA

Penetraatiotestaus on ohjelmistolle kehityksen loppuvaiheessa suoritettava testaus, jolla testataan ohjelmiston turvallisuutta (McGraw 2004). Penetraatiotestauksessa ohjelmiston turvallisuuden testaaminen toteutetaan yrittämällä tunkeutua ohjelmistoon sisälle (Epling et al. 2015; Automated penetration 2022). McGrawn (2004) mukaan penetraatiotestauksen etuna on, että sen avulla saadaan ymmärrys ohjelmiston turvallisuudesta oikeassa ympäristössä. Toisaalta penetraatiotestauksen avulla ei todennäköisesti saavuteta kiinnostavaa tietoa ohjelmiston haavoittuvuuksista syvemmällä tasolla. Mikäli penetraatiotestit epäonnistuvat, tarkoittaa se yleensä suuria puutoksia ohjelmiston turvallisuudessa. (McGraw 2004) Penetraatiotestien luonteen perusteella penetraatiotestejä voi pitää yleisen tason testeinä ohjelmiston turvallisuudelle.

Penetraatiotestauksen mallille on kehitetty omaa standardia, mutta standardin kehitys on vielä kesken. Standardista on kuitenkin julkaistu jo ensimmäinen versio, jonka mukaan penetraatiotestaus voidaan jakaa seitsemään eri vaiheeseen seuraavasti:

1. määrittely
2. tiedonkeruu
3. riskienmallinnus
4. haavoittuvuuksien analysointi
5. hyökkäys
6. jälkihyökkäys
7. raportointi.

Penetraatiotestauksen mallin vaiheet suoritetaan aina listatun järjestyksen mukaisesti. (Penetration Testing Execution Standard 2014)

4.1 Automatisoitu penetraatiotestaus

Manuaalisten penetraatiotestien suorittaminen kestää yleensä viikkoja tai kuukausia, mikä tuottaa myös paljon kustannuksia. Automaatio on tehokas työkalu tietotekniikan toistettavissa tehtävissä. Jonkin toiminnon automatisointi vie yleensä paljon aikaa, mutta kun kyseinen asia on saatu automatisoitua, on mahdollista säästää paljon resursseja.

Automaation kanssa on oltava myös varovainen. Automaation avulla tehdyt penetraatiotestit saattavat aiheuttaa esimerkiksi palvelinten kaatumisia, mikäli testiympäristö ei sovellu automatisoiduille penetraatiotesteille. Automaation avulla on kuitenkin mahdollista säästää resursseja, kun sitä käytetään oikeaan tarkoitukseen. (Epling et al. 2015)

Penetraatiotestauksen avulla voidaan kartoittaa koko organisaation IT-infrastruktuurin turvallisuutta. Koko organisaation IT-infrastruktuurin turvallisuuteen kuuluu myös muun muassa fyysinen tiedustelu ja tietojenkalastelusähköpostit. Fyysinen tiedustelu ja tietojenkalastelusähköpostit eivät ole helppoja asioita automatisoida. Toisaalta joidenkin penetraatiotestien suorittamisessa kone on ihmistä parempi. Esimerkiksi penetraatiotestauksessa on mahdollista tehdä testejä, joissa ohjelmistoon kohdistetaan suuria kuormia haavoittuvuuksien löytämiseksi. 99,9 % kohdistetusta haavoittuvuuksia etsivästä kuormasta ei löydä ohjelmistosta haavoittuvuuksia, mutta 0,01 % kohdistetusta haavoittuvuuksia etsivästä kuormasta löytämät haavoittuvuudet voivat olla merkittäviä löydöksiä. Koska ohjelmistoa pitää kuormittaa suuria määriä haavoittuvuuksien löytämiseksi, ei sitä ole järkevää tehdä ihmisvoimin, jolloin suuria määriä kuormitusta vaativiin penetraatiotesteihin kannattaa käyttää automaatiota manuaalisen suorituksen sijaan. (Automated penetration 2022) Tässä työssä keskitytään erityisesti automatisoituihin penetraatiotesteihin, joissa ohjelmistoon kohdistetaan suuria kuormia haavoittuvuuksien löytämiseksi.

Esimerkiksi PortSwigger tekee työkalua automatisoituun penetraatiotestaukseen. Tämän työkalun nimi on Burp Scanner ja se on osa Burp Suite -ohjelmistoa. Kuvassa 1 on esitelty Burp Suiten Burp Scannerin esimerkkiraportin tiivistelmän osa automatisoidusta penetraatiotestistä.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	12	4	1	17
	Medium	0	2	0	2
	Low	4	0	0	4
	Information	9	2	0	11

Kuva 1. *Burp Scannerin esimerkkiraportin tiivistelmän osa (Burp Scanner 2013)*

Kuvassa 1 on esitelty Burp Scannerin löytämiä haavoittuvuuksia ohjelmistosta, ja haavoittuvuudet on jaoteltu niiden vakavuuden (Severity) ja luotettavuuden (Confidence) pe-

rusteella. Vakavuus kuvastaa haavoittuvuuden vakavuutta tavanomaiselle organisaatiolle, ja luotettavuus kuvastaa löydöksen luotettavuutta löytämiseen käytetyn tekniikan perusteella. (Burp Scanner 2013)

Burp Scannerin (2013) muodostama esimerkkiraportti sisältää myös tarkempaa tietoa suoritetun penetraatiotestin eri osa-alueista, joita esimerkkiraportissa on yhteensä 22. Eräs esimerkki tällaisesta osa-alueesta on tietokannat. Burp Scannerin (2013) muodostama esimerkkiraportti vastaa osittain Penetration Execution Standardin (2014) määrittelemistä penetraatiotestauksen vaiheista seitsemättä, eli raportointia. Burp Scannerin (2013) muodostamasta esimerkkiraportista puuttuu kuitenkin esimerkiksi Penetration Execution Standardissa (2013) määritelty jatkotoimenpidesuositus-osio. Tietoa jatkojalostamalla Burp Scannerin (2013) muodostaman raportin perusteella on mahdollista suorittaa Penetration Execution Standardin (2013) määrittämän penetraatiotestauksen standardin seitsemäs vaihe eli raportointi.

4.2 Mittaaminen automatisoidun penetraatiotestauksen avulla

Penetraatiotestien automatisoinnin avulla ohjelmiston turvallisuudesta pystytään tuottamaan paljon dataa, sillä automatisoitujen penetraatiotestien avulla on mahdollista kohdistaa ohjelmistoon suuria määriä haavoittuvuuksia etsivää kuormaa. Vastaavia määriä dataa on vaikeaa hankkia manuaalisena ihmistyönä toteutettuna, minkä takia automatisoitu penetraatiotesti on hyvä tapa hankkia dataa ohjelmiston turvallisuuteen liittyen, sillä automaattisen penetraatiotestin avulla dataa ohjelmiston turvallisuudesta saadaan laajemmin ja vaivattomammin kuin manuaalisesti testattuna. Automaattisten penetraatiotestien suorittaminen ei myöskään vaadi organisaatiolta lisäresurssia henkilöstöpuolella.

Penetraatiotestauksen avulla mitatessa on mahdollista vertailla kahden eri ohjelmiston turvallisuutta keskenään (Pendelton et al. 2016). Toisaalta penetraatiotestauksen mahdollistaman vertailun perusteella penetraatiotestausta on mahdollista soveltaa yhden ohjelmiston kahden eri version mittaamiseen, sillä yhden ohjelmiston kaksi eri versiota voidaan tulkita kahdeksi eri ohjelmistoksi IEEE Standardin (1990) ohjelmiston määritelmän mukaisesti. Kahden eri penetraatiotestikerran vertailukelpoisuuden takia penetraatiotestaus on soveltuva ohjelmiston turvallisuuden kehityksen mittaamiseen.

Myös Zaber & Nair (2020) tukevat ajatusta, että muutoksia turvallisuudessa voidaan analysoida mittaamalla turvallisuutta ennen ja jälkeen muutoksen. Tätä ajattelutapaa on mahdollista hyödyntää myös pidempijaksoisessa turvallisuuden kehityksen seuraamisessa. Turvallisuuden pidempijaksoisen kehityksen seuraamisessa on kuitenkin lähes

välttämättömänä edellytyksenä, että testaus on täysin automatisoitua, sillä tällöin voidaan olla varmoja testauksen toistettavuudesta ja johdonmukaisuudesta. (Zaber & Nair 2020) Automatisoitu penetraatiotestaus täyttää vaatimukset testauksen täydelle automatisoinnille.

Penetraatiotestaukseen perustuvassa mittaamisessa keskitytään erityisesti Penetration Testing Execution Standardin (2014) seitsemänteen vaiheeseen, eli raportointiin. Penetration Testing Execution Standardin (2014) seitsemän vaiheen mallin mukaan vaiheista viimeinen, eli seitsemäs, sisältää raportin penetraatiotestauksesta. Tämä raportti sisältää tietoa muun muassa penetraatiotestauksen aikana löydetyistä haavoittuvuuksista ja niiden vakavuudesta (Penetration Testing Execution Standard 2014).

Automatisoidun penetraatiotestauksen käyttö mittaamisessa aiheuttaa kuitenkin myös lisää huomioon otettavia asioita. Erityisesti automatisoitujen penetraatiotestien käytössä pitää huomioida niiden sopivuus tehtävään ja ympäristöön, sillä esimerkiksi testiympäristön ollessa epäsoveltuva automatisoidun penetraatiotestauksen käyttöön voi ilmetä ongelmia kuten palvelinten kaatumisia (Epling et al. 2015). Toisaalta automatisoitujen penetraatiotestien soveltumisella tehtävään voidaan tarkoittaa myös sitä, että mittaamisella saadaan tuotettua haluttua asioita. Tämän lisäksi automatisoidun penetraatiotestauksen käyttämisessä ohjelmiston turvallisuuden mittaamiseen tulee muistaa ottaa huomioon myös ohjelmiston turvallisuuden mittaamiselle ominaiset riskit ja yleisesti mittaamiselle ominaiset riskit.

5. AUTOMATISOIDUN PENETRAATIOTESTAUKSEN SOVELTUVUUS OHJELMISTON TURVALLISUUDEN KEHITYKSEN MITTAAMISEEN

Rotellan (2018) mukaan ohjelmiston turvallisuuden kehityksen mittaamisessa erityisen tärkeä asia on vertailukohdat. Automatisoidun penetraatiotestauksen avulla todettiin pystyvän tuottamaan useita mittauksia, jotka ovat vertailukelpoisia keskenään. Lisäksi penetraatiotestaus testaa ohjelmiston turvallisuutta (McGraw 2004). Automatisoitujen penetraatiotestien avulla on siis mahdollista mitata ohjelmiston turvallisuuden kehitystä, sillä automatisoitujen penetraatiotestien avulla on mahdollista tuottaa vertailukelpoista dataa, mikä mahdollistaa ohjelmiston turvallisuuden kehityksen mittaamisen. Automatisoidun penetraatiotestauksen käyttö ohjelmiston turvallisuuden kehityksen mittaamisessa sisältää niin mahdollisuuksia kuin mahdollisia riskitekijöitä.

5.1 Automatisoidun penetraatiotestauksen mahdollisuudet ohjelmiston turvallisuuden kehityksen mittaamisessa

Automatisoitu penetraatiotestaus tuo monia mahdollisuuksia ohjelmiston turvallisuuden kehityksen mittaamiseen. Automatisoidun penetraatiotestauksen tuomia mahdollisuuksia tulisi tarkastella ohjelmiston turvallisuuden kehityksen mittaamista suunnitellessa, sillä automatisoidun penetraatiotestauksen tuomat mahdollisuudet ohjelmiston turvallisuuden kehityksen mittaamiseen voivat esimerkiksi säästää organisaation resursseja ja edistää tiedolla johtamista. Toisaalta taas automatisoidun penetraatiotestauksen tuomat mahdollisuudet ohjelmiston turvallisuuden kehityksen mittaamisessa eivät välttämättä tue mittaamistilannetta ja -tarkoitusta, jolloin on tarpeellista tiedostaa, että automatisoidut penetraatiotestit eivät ole aina paras tapa mitata ohjelmiston turvallisuuden kehitystä.

Rotellan (2018) mukaan vertailukohdat ovat tärkeitä ohjelmiston turvallisuuden mittaamisessa. Automaation avulla manuaalista työmäärää on mahdollista vähentää merkittävästi (Epling et al. 2015). Erityisesti automatisoitujen penetraatiotestien manuaalisen työmäärän vähyyden vuoksi automatisoituja penetraatiotestejä on mahdollista tehdä useammin kuin esimerkiksi manuaalisesti toteutettuja penetraatiotestejä. Useammin toteutetut automatisoidut penetraatiotestit antavat enemmän ja tiheämmällä välillä olevia vertailukohtia. Vertailukohtien suurempi lukumäärä antaa mahdollisuuden seurata ohjelmiston turvallisuuden kehitystä tarkemmin.

Juneja et al. (2011) esittävät toistettavuuden olevan tärkeä asia mittaamisessa. Zaber & Nair (2020) määrittävät automaation lähes välttämättömäksi asiaksi jatkuvassa turvallisuuden mittaamisessa toistettavuuden takia. Erityinen hyöty automatisoitujen penetraatiotestien käytössä ohjelmiston turvallisuuden kehityksen mittaamiseksi on automaation mahdollistama tarkka toistettavuus, sillä ohjelmiston turvallisuuden kehityksen mittaaminen on jatkuvaa mittaamista.

Rotellan (2018) mukaan vertailukohtien ja suorituskykymittauksen avulla pystytään muun muassa määrittämään kannattamattomia toimintatapoja ohjelmiston turvallisuuden lisäämiseen ja levittämään tehokkaammin parhaita tapoja ohjelmiston turvallisuuden kehittämiseksi. Automatisoitujen penetraatiotestien avulla on mahdollista toteuttaa suorituskykymittausta ja automatisoiduista penetraatiotesteistä saadaan vertailukohtia tuleville mittauksille. Kannattamattomien toimintatapojen tunnistamisella ja kannattavien toimintatapojen levittämisellä ohjelmiston turvallisuutta olisi mahdollista parantaa myös tulevaisuudessa. Näkökulma proaktiivisesta ohjelmiston turvallisuuden parantamisesta mittauksien perusteella on myös tärkeää ottaa huomioon, sillä se mahdollistaa ohjelmiston turvallisuuteen käytettävien resurssien tehokkaamman kohdistamisen oikeisiin asioihin.

Automaatio on pidempijaksoisessa turvallisuuden mittaamisessa helppouden ja toistettavuuden kannalta lähes välttämätöntä (Zaber & Nair 2020). Automaation tuoma helppous mittauksessa mahdollistaa laajojen mittausdatamäärien tuottamisen, sillä mittausdatan tuottaminen ei vaadi merkittäviä henkilöresursseja. Lisäksi automatisoitujen penetraatiotestien avulla ohjelmistoon pystytään kohdistamaan paljon kuormaa (Automated penetration 2022). Suuri määrä ohjelmistoon kohdistettua testikuormaa tuottaa myös suuren määrän dataa ohjelmiston turvallisuudesta. Ohjelmiston turvallisuudesta saatavaa dataa voidaan mahdollisuuksien mukaan hyödyntää mittaamisessa, ja mittaamisen tarkkuutta on mahdollista parantaa suuremman mittausdatamäärän avulla.

Automatisoitu penetraatiotestaus on myös Suomalan et al. (2011, s. 136) esittämään haasteeseen mittaamisen kustannuksista varsin tehokas ratkaisu. Eplingin et al. (2015) mukaan automaatio on kustannustehokas työkalu tietotekniikan toistettavissa tehtävissä. Automatisoidun penetraatiotestauksen käyttö ohjelmiston turvallisuuden kehityksen mittaamisessa on kustannustehokkaampi ratkaisu, kuin esimerkiksi täysin manuaalityönä suoritettava penetraatiotestaus, joka vaatii Eplingin et al. (2015) mukaan paljon resursseja, mikä aiheuttaa paljon kustannuksia. Automatisoidun penetraatiotestauksen kustannustehokkuuden vuoksi mittaamisen kustannushaaste ei ole yhtä suuri kuin

useissa muissa mittausmenetelmissä. Ohjelmiston turvallisuuden kehityksen mittaamista automatisoidun penetraatiotestauksen avulla voi perustella muun muassa sen kustannustehokkuuden avulla.

5.2 Mahdolliset riskitekijät automatisoidun penetraatiotestauksen soveltamisessa ohjelmiston turvallisuuden kehityksen mittaamiseen

Mahdollisten riskitekijöiden toteutuessa ohjelmiston turvallisuuden kehityksen mittaamisessa penetraatiotestauksen avulla voi päätöksenteon tukena käytettävä tieto olla väärää, ja täten päätöksenteossa tulee ottaa mahdolliset riskitekijät huomioon, mikäli ohjelmiston turvallisuuden kehitystä on mitattu automatisoitujen penetraatiotestien avulla. Mahdollisten riskitekijöiden huomiointi on myös tarpeellista ohjelmiston turvallisuuden kehitystä mitattaessa automatisoidun penetraatiotestauksen avulla, sillä mahdollisten riskitekijöiden huomioinnilla ohjelmiston turvallisuuden kehitystä penetraatiotestauksen avulla mitattaessa on mahdollista vähentää mahdollisten riskitekijöiden toteutumista.

Penetration Testing Standardin (2014) mukaan penetraatiotestauksesta saadaan laaja raportti ohjelmiston turvallisuudesta, joka sisältää tietoa esimerkiksi haavoittuvuuksista ja niiden vakavuudesta. Kaikkea raportin sisältöä ei välttämättä kannata käyttää ohjelmiston turvallisuuden mittaamiseen, sillä kaikki raportin sisältö ei välttämättä ole relevanttia ohjelmiston turvallisuuden kehityksen mittaamisen näkökulmasta. Raportin oleellisiin asioihin keskittymällä ennalta ehkäistään Lewiksen (2015) esittelemää mittaamisen tavallista ongelmaa, jossa mittaaminen ei tavoita haluttuja asioita. On siis tärkeää osata karsia penetraatiotestauksen tuottaman raportin mittaamisen kannalta epärelevantti tieto pois mittaamisesta, jotta mittaustulokset keskittyvät haluttuihin asioihin. Zalewski et al. (2014) toteavat ohjelmiston turvallisuuden mittaamisessa tärkeäksi asiaksi määritellä, miksi jotakin mitataan. Selkeä tavoite mittaamisessa auttaa siinä, että mittaaminen tavoittaisi haluttuja asioita.

Ohjelmiston turvallisuuden ja automatisoidun penetraatiotestauksen avulla mittaamisen erityiset haasteet tulee ottaa huomioon ohjelmiston turvallisuuden kehityksen mittaamisessa automatisoidun penetraatiotestauksen avulla. Zalewski et al. (2014) määrittelivät ohjelmiston turvallisuuden mittaamisen sisältävän erityisesti haasteen ohjelmiston haavoittuvuuksien kokonaismäärän tietämättömyydestä. Mahdollinen riski ohjelmiston haavoittuvuuksien kokonaismäärän tietämättömyydestä tulee ottaa huomioon myös ohjelmiston turvallisuuden kehitystä automatisoitujen penetraatiotestien avulla mitattaessa, mutta toisaalta Rotella (2018) painottaa vertailukohtien tärkeyttä ohjelmiston turvallisuu-

den mittaamisessa. Kokonaismäärien tietämättömyyden haasteen aiheuttama mahdollinen riski vähenee, kun ohjelmiston turvallisuuden mittaamisen kehityksessä käytetään Rotellan (2018) mukaisia vertailukohtia, sillä löytyneiden haavoittuvuuksien määrää voidaan vertailla aiempiin mittauksiin ja vertailun avulla ohjelmiston turvallisuuden kehitystä on mahdollista seurata.

Automatisoidun penetraatiotestauksen riskinä on Epling et al. (2015) mukaan mahdollinen palvelinten kaatuminen, mikäli testausympäristö ei ole sopiva automatisoiduille penetraatiotesteille. Tämän mahdollisen riskitekijän huomiointi on tärkeää erityisesti ohjelmiston turvallisuuden mittausympäristöä suunnitellessa, sillä ympäristö tulisi pitää Junejan et al. (2011) määrittämän mittaamisen toistettavuuden perustella samanlaisena joka mittauskerralla, jolloin testiympäristöä ei tulisi rakentaa uudelleen jokaista mittauskertaa varten.

Eplingin et al. (2015) mukaan automatisointi vie paljon resursseja ja Suomalan et al. (2011, s. 136) mukaan kustannukset ovat haaste mittaamisessa. Mittaamista aloitettaessa kustannukset saattavat olla siis korkeat, sillä automatisointi vie paljon resursseja. Suuret kustannukset mittaamisen alussa saattavat johtaa siihen, että mittaaminen lopetetaan. Toisaalta erilaiset valmiit työkalut, kuten esimerkiksi PortSwiggerin tarjoama Burp Suite voivat helpottaa penetraatiotestauksen automatisointia (Automated penetration 2022). Mahdollista riskiä mittaamisen lopettamisesta mittaamisen alkuvaiheessa syntyvien automatisoinnin aiheuttamien korkeiden kustannusten vuoksi voi olla mahdollista pienentää erilaisten valmiiden työkalujen avulla. Riski korkeista kustannuksista mittaamisen alkuvaiheessa automatisoinnin vuoksi on kuitenkin otettava huomioon mittaamisen kustannuksia tarkastellessa.

6. YHTEENVETO

Johdannossa määriteltiin kaksi alatutkimuskysymystä tutkimukselle. Ensimmäisen alatutkimuskysymyksen avulla pyrittiin selvittämään ohjelmiston turvallisuuden kehittymisen mittaamista erityisesti mittaamisen näkökulmasta. Ensimmäiseen alatutkimuskysymykseen saatu vastaus ei ollut täysin yksiselitteinen, sillä ohjelmiston turvallisuuden mittaamiseen löytyi useita eri näkökulmia. Ohjelmiston turvallisuuden mittaamiseen löytyi kuitenkin paljon tukevaa materiaalia, joten työssä käsiteltiin ohjelmiston turvallisuutta mittaamiskelpoisena asiana. Ohjelmiston turvallisuuden kehitys pystyttiin myös määrittämään asiaksi, jota olisi tärkeä mitata.

Toisen alatutkimuskysymyksen avulla pyrittiin selvittään ohjelmiston turvallisuuden mittaamista automatisoitujen penetraatiotestien avulla erityisesti automatisoituen penetraatiotestien näkökulmasta. Toisen alatutkimuskysymyksen avulla saatiin kerättyä automatisoidulle penetraatiotestaukselle ja automatisoidun penetraatiotestauksen avulla mittaamiselle erityisiä huomioitavia asioita eri näkökulmista.

Päätutkimuskysymykseen muodostettiin ratkaisu kahden alatutkimuskysymyksen avulla. Alatutkimuskysymysten avulla mittaamista ja penetraatiotestausta vietiin lähemmäksi päätutkimuskysymystä, ja lopulta alatutkimuskysymysten ratkaisut yhdistettiin selvittäessä ratkaisua päätutkimuskysymykseen. Päätutkimuskysymyksen avulla pyrittiin selvittämään automatisoidun penetraatiotestauksen mahdollisuuksia ohjelmiston turvallisuuden kehityksen mittaamisessa. Päätutkimuskysymyksen avulla automatisoidun penetraatiotestauksen käytöstä ohjelmiston turvallisuuden kehityksen mittaamisessa löytyi mahdollisuuksia ja riskitekijöitä.

6.1 Tutkimuksen keskeiset tulokset

Zalewskin et al. (2014) mukaan mittaamisessa on erityisen tärkeää määrittää, miksi jotakin mitataan. Tässä työssä mittauksen syyksi on määritelty tiedon määrän lisääminen päätöksenteossa, mikä edistää tiedolla johtamista. Tutkimuksen keskeisiä tuloksia tarkastellaan myös tästä näkökulmasta.

Ensimmäiseen alatutkimuskysymykseen vastaamalla saatiin tutkimuksen kannalta merkittävä tulos, joka vahvisti, että ohjelmiston turvallisuuden kehitystä voidaan mitata. Lisäksi ensimmäiseen alatutkimuskysymyksen avulla todettiin, että ohjelmiston turvallisuuden kehitystä kuuluisi mitata. Toisen alatutkimuksen avulla saatiin selvitettyä automati-

soidun penetraatiotestauksen tärkeimmät huomioitavat asiat mittaamisen kannalta. Tärkeimmiksi huomioitaviksi asioiksi automatisoidussa penetraatiotestauksessa ohjelmiston turvallisuuden kehityksen mittaamisen kannalta muodostuivat toistettavuus ja vertailukelpoisuus.

Tutkimuksen keskeisimpänä löytönä oli se, että automatisoitu penetraatiotestaus sisältää ominaisuuksia, joiden vuoksi automatisoidusta penetraatiotestauksesta voi olla hyötyä ohjelmiston turvallisuuden kehityksen mittaamisessa. Automatisoidun penetraatiotestauksen käytön mahdollisuuksia ja riskejä ohjelmiston turvallisuuden kehityksen mittaamisessa käsiteltiin luvussa 5. Taulukossa 3 on esitelty tutkimuksen keskeisimpinä tuloksina automatisoidun penetraatiotestauksen mahdollisuudet ja riskit ohjelmiston turvallisuuden kehityksen mittaamisessa.

Taulukko 3. *Automatisoidun penetraatiotestauksen mahdollisuudet ja riskit ohjelmiston turvallisuuden kehityksen mittaamisessa*

Mahdollisuudet	**Riskit**
Vaatii vähän henkilöresursseja	Merkityksellisen tiedon erottaminen
Helppo toistettavuus	Ympäristön soveltuvuus
Tuottaa paljon dataa	Alun korkeammat kustannukset
Vertailtavuus keskenään	
Kustannustehokas ratkaisu	

Taulukon 3 perusteella automatisoidun penetraatiotestauksen voidaan todeta sisältävän enemmän mahdollisuuksia kuin riskejä ohjelmiston turvallisuuden mittaamisessa tämän tutkimuksen perusteella. Mahdollisuuksien ja riskien suuruudet ovat kuitenkin organisaatiokohtaisia, jolloin automatisoidun penetraatiotestauksen sovellettavuutta ohjelmiston turvallisuuden kehityksen mittaamiseen tulee harkita organisaatiokohtaisesti. Automatisoidun penetraatiotestauksen löydettiin myös vähentävän joitakin mittauksen yleisistä ongelmista aiheutuvia riskejä kuten mittaamisesta aiheutuvia kustannuksia. Tämän tutkimuksen perusteella automatisoitua penetraatiotestausta tulisi harkita ainakin yhdeksi datankeruumenetelmäksi, kun mitataan ohjelmiston turvallisuuden kehitystä.

6.2 Tulosten arviointi

Tutkimuksen tulosten avulla pystytään tuomaan lisää tietoa päätöksenteon tueksi. Tutkimuksen tulokset auttavat erityisesti ohjelmiston turvallisuuden kehityksen mittaustavan päättämisessä, mikä edistää tiedolla johtamista ohjelmistotuotannossa. Tutkimuksen tuloksilla on mahdollisesti myös välillinen vaikutus tiedolla johtamisen lisääntymiseen ohjelmistotuotannossa, sillä ohjelmistotuotannossa saadaan lisää tietoa päätöksenteon tu-

eksi, jos ohjelmiston turvallisuuden kehitystä aletaan mittaamaan automatisoidun penetraatiotestauksen avulla. Erityisesti tiedolla johtamisen mahdollistaminen entistä paremmin ohjelmistotuotannossa tekee tutkimuksen tuloksista merkittäviä.

Tutkimusmenetelmänä tutkimuksessa toimi kirjallisuuskatsaus. Kirjallisuuskatsauksessa lähdeaineistolla on merkittävä rooli tutkimuksen kannalta. Tutkimusmateriaalia onnistuttiin etsimään erilaisista lähteistä erilaisten näkökulmien esilletuomiseksi. Lähdeaineistoista onnistuttiin myös erittelemään erityisesti tämän tutkimuksen kannalta merkittävät asiat, mikä oli erityisen tärkeää, sillä tämän tutkimuksen tarkasta aiheesta ei löytynyt aiempaa tutkimusta. Lähdeaineistoista onnistuttiin löytämään toisiaan tukevia näkemyksiä useassa kohtaa joko samanlaisella tai eriävällä näkemyksellä. Tutkimuksen avulla onnistuttiin vastaamaan tutkimuskysymyksiin, ja tutkimuksen avulla saatiin tuotettua halutunlaista tietoa automatisoidun penetraatiotestauksen mahdollisuuksista ohjelmiston turvallisuuden kehityksen mittaamisessa.

Tutkimusaihe oli verrattain vähän tutkittu, minkä vuoksi tähän tutkimukseen soveltamiskelpoisen tutkimusmateriaalin määrä oli varsin rajallinen. Tutkimukseen onnistuttiin kuitenkin löytämään riittävä määrä tutkimusmateriaalia sen toteuttamiseksi. Tutkimusmateriaaleissa oli osittain näkökulmallisia eroja toisiinsa nähden, mikä saattaa heikentää tulosten paikkansapitävyyttä. Tutkimuksessa tukeuduttiin kuitenkin näkökulmiin, joita enemmistö valitusta lähdeaineistosta tuki.

Tutkimus ei sisältänyt ollenkaan empiiristä osuutta. Empiirisen osuuden avulla olisi ollut mahdollista vahvistaa teorian perusteella tehdyt johtopäätökset. Tutkimuksen rajallinen pituus rajoitti tutkimuksen lähestymisen tutkimusaiheeseen vain kahdesta eri näkökulmasta, jotka olivat mittaaminen yleisellä tasolla ja automatisoitu penetraatiotestaus. Pidempi tutkimus olisi mahdollistanut useamman näkökulman tuomisen tutkimukseen, minkä avulla tutkimuksen tuloksista olisi mitä luultavimmin saatu vielä tarkempia ja luotettavampia.

Tutkimusaiheesta ei löytynyt aiempaa tutkimusta, mikä tekee tulosten vertailun aiempaan tutkimukseen vaikeaksi. Osia tuloksista voi kuitenkin vertailla aiempiin tuloksiin. Esimerkiksi ohjelmiston turvallisuuden mittauskelpoisuudesta löytyy tämän tutkimuksen tulosta tukevia tuloksia. Vertailukelpoiset tulosten osat olivat samankaltaisia kuin enemmistössä aiemmista aihetta koskevista tutkimuksista.

6.3 Jatkotutkimusehdotukset

Tämä tutkimus liittyi ohjelmiston turvallisuuteen ja se keskittyi erityisesti mittaamiseen penetraatiotestien avulla. Mittaamisen avulla on mahdollista rakentaa erilaisia mittaristoja. Toisaalta myös johonkin muuhun kuin automatisoituihin penetraatiotesteihin pohjautuvaa ohjelmiston turvallisuuden kehityksen mittaamista voisi tutkia vielä lisää. Mahdollisia jatkotutkimusaiheita tämän tutkimuksen perusteella ovat esimerkiksi automatisoidun penetraatiotestaukseen perustuva ohjelmiston turvallisuuden mittaristo, mittariston rakentaminen ohjelmiston turvallisuuden kehityksen seuraamiseksi ja resurssien vaikutukset ohjelmiston turvallisuuden kehitykseen.

Automatisoituun penetraatiotestaukseen perustuvasta ohjelmiston turvallisuuden mittaristosta voisi tutkia, mitä kaikkea mittariston tulisi sisältää, jotta ohjelmiston turvallisuudesta saataisiin mahdollisimman realistinen kuva. Lisäksi pelkän automatisoidun penetraatiotestauksen riittävyyttä ohjelmiston turvallisuuden kehityksen mittaamisessa voisi selvittää. Mittariston rakentamisessa ohjelmiston turvallisuuden kehityksen seuraamiseksi voisi tutkia erityisesti mittariston rakentamista mittaamisen perusteella ohjelmiston turvallisuuden kehityksen näkökulmasta. Lisäksi ohjelmiston turvallisuuden jatkuvan kehityksen mittaristoon voisi soveltaa automatisoidun penetraatiotestauksen käyttöä ohjelmiston turvallisuuden kehityksen mittaamisessa. Resurssien vaikutusta ohjelmiston turvallisuuden kehitykseen voisi tutkia erityisesti empiirisesti rakennetun mittariston perusteella. Tutkimus resurssien vaikutuksesta ohjelmiston turvallisuuden kehitykseen mahdollistaisi tiedon tuomisen päätöksenteon tueksi, minkä avulla voitaisiin edistää tietojohtamista ohjelmistotuotannossa.

LÄHTEET

Ahmed, R. K. (2016). Overview of Security Metrics. Software Engineering. Vol. 4(4), s. 59–64.

Automated penetration testing software. (2022). PortSwigger. Saatavissa (30.10.2022): https://portswigger.net/solutions/penetration-testing/automated-penetration-testing.

Bishop, M. (2003). What is computer security?. IEEE Security & Privacy. Vol. 1(1), s. 67–69.

Burp Scanner Sample Report. (2013). PortSwigger. Saatavissa (24.11.2022): https://portswigger.net/burp/samplereport/burpscannersamplereport.

Chhillar, R. S. & Gahlot, S. (2017). An Evolution of Software Metrics: A Review. In Proceedings of the International Conference on Advances in Image Processing (ICAIP 2017). Association for Computing Machinery, New York, NY, USA. s. 139–143.

Epling, L., Hinkel, B. & Hu, Y. (2015). Penetration testing in a box. In Proceedings of the 2015 Information Security Curriculum Development Conference (InfoSec '15). Association for Computing Machinery, New York, NY, USA. Article 6, s. 1–4.

Fink, A. (2014). Conducting research literature reviews: from the Internet to paper. Thousand Oaks (Calif.): Sage.

IEEE Standard Glossary of Software Engineering Terminology. (1990). IEEE Std 610.12-1990. s. 1–84.

Jaatun, M. G. (2018). Software Security Activities that Support Incident Management in Secure DevOps. In Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018). Association for Computing Machinery, New York, NY, USA. Article 8, s. 1–6.

Jaquith, A. (2007). Security metrics. Upper Saddle River: Pearson Education Inc.

Juneja, D., Arora, K. & Duggal, S. (2011). Developing Security Metrics For Information Security Measurement System. International Journal of Enterprise Computing and Business Systems. Vol. 1(2), s. 1–10.

Laihonen, H., Hannula, M., Helander, N., Ilvonen, I., Jussila, J., Kukko, M., Kärkkäinen, H., Lönnqvist, A., Myllärniemi, J., Pekkola, S., Virtanen, P., Vuori, V. & Yliniemi, T. (2013) Tietojohtaminen. Tampereen teknillinen yliopisto, Tiedonhallinnan ja logistiikan laitos.

Lewis, J. (2015). The politics and consequences of performance measurement. Policy and Society. Vol. 34(1), s. 1–12.

Martinsuo, M., Mäkinen, S., Suomala, P. & Lyly-Yrjänäinen, J. (2016). Teollisuustalous kehittyvässä liiketoiminnassa. Helsinki: Edita.

McGraw, G. (2004). Software security. IEEE Security & Privacy. Vol. 2(2), s. 80–83.

Pendelton, M., Garcia-Lebron, R., Cho, J. & Xu S. (2016). A Survey on Systems Security Metrics. ACM Computing Surveys. Vol. 49(4), Article 62, s. 1–35.

Penetration Testing Execution Standard. (2014). Saatavissa (19.11.2022): http://www.pentest-standard.org/.

Rotella, P. (2018). Software security vulnerabilities: baselining and benchmarking. In Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment (SEAD '18). Association for Computing Machinery, New York, NY, USA. s. 3–10.

Saarela, M. (2016). Measuring software security from the design of software. University of Turku, Department of IT.

Schlosser, R.W., Wendt, O., Bhavnani, S. & Nail-Chiwetalu, B. (2006). Use of information-seeking strategies for developing systematic reviews and engaging in evidence-based practice: the application of traditional and comprehensive Pearl Growing. A review, International Journal of Language & Communication Disorders. Vol. 41(5), s. 567–582.

Stolfo, S., Bellovin, S.M. and Evans, D. (2011). Measuring Security. IEEE Security & Privacy. Vol. 9(3), s. 60–65.

Suomala, P., Manninen, O. & Lyly-Yrjänäinen, J. (2011). Laskentatoimi johtamisen tukena. 1. painos. Helsinki: Edita.

Tiedonhaun opas: Tiedonhaun suunnittelu. (2022). Tampereen yliopiston kirjasto. Saatavissa (5.11.2022): https://libguides.tuni.fi/tiedonhaun-opas/haun-suunnittelu.

Torgersen, M.D. (2007). Security Metrics for Communication Systems, In Proc. ICCRTS'07, Intern. Command and Control Research and Technology Symposium (Newport, RI, June 19–21).

Weir, C., Migues, S., Ware, M. & Williams, L. (2021). Infiltrating Security into Development: Exploring the World's Largest Software Security Study. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on

the Foundations of Software Engineering (ESEC/FSE 2021). Association for Computing Machinery, New York, NY, USA. s. 1326–1336.

Zaber, M. & Nair, S. (2020). A framework for automated evaluation of security metrics. In Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20). Association for Computing Machinery, New York, NY, USA. Article 89, s. 1–11.

Zalewski, J., Drager S.& Kornecki, A. (2011). Can We Measure Security and How?. In Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW '11). Association for Computing Machinery, New York, NY, USA. Article 46, s.1–4.

Zalewski, J., Drager S.& Kornecki, A. (2014). Measuring Security: A Challenge for the Generation. Position Papers of the 2014 Federated Conference on Computer Science and Information Systems. s. 131–140.