Aaron Bondfolk

TIETOTURVAHAASTEET JA -RATKAISUT DEVOPS-KEHITYKSESSÄ

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2022

TIIVISTELMÄ

Bondfolk, Aaron
Tietoturvahaasteet ja -ratkaisut DevOps-ympäristössä
Jyväskylä: Jyväskylän yliopisto, 2022, 30 s.
Tietojärjestelmätiede, kandidaatintutkielma
Ohjaaja(t): Marttiin, Pentti

Tässä kandidaatintutkielmassa syvennytään DevOps-menetelmään, sekä siihen liitettyihin tietoturvahaasteisiin, sekä niihin kehitettyihin ratkaisuihin. DevOps on jatkuvasti suosiotaan kasvattava menetelmä, jossa kehitys- ja toimintatiimi yhdistetään toisiinsa saumattomalla tavalla. Siihen sisältyy muun muassa sen nopean luonteen vuoksi joitakin tietoturvahaasteita ja tässä tutkielmassa esitellään sekä niitä, että tietoturvahaasteisiin kehitettyjä ratkaisuja. Tarkoituksena oli kartoittaa tietoturvan huomiointia DevOps-kehityksessä ja potentiaalisten ratkaisujen vastaavuutta haasteisiin. Tutkielma toteutettiin kirjallisuuskatsauksena. Tutkielmassa todetaan, että DevOps-menetelmään liittyy useita erityyppisiä tietoturvahaasteita, mutta niihin on kehitetty melko kattavia ratkaisuja.

Asiasanat: DevOps, SecDevOps, tietoturva,

ABSTRACT

Bondfolk, Aaron
Security issues and solutions in DevOps
Jyväskylä: University of Jyväskylä, 2022, 30 pp.
Information systems, Bachelor's thesis
Supervisor(s): Marttiin, Pentti

This bachelor's thesis delves into the DevOps method, as well as the security challenges associated with it, and the solutions developed for them. DevOps is a method that is constantly gaining popularity, where the development and operation teams are seamlessly integrated. Among other things, due to its rapid nature, it includes some security challenges, and this tutorial presents both them and solutions developed for security challenges. The purpose was to map the consideration of information security in DevOps development and the equivalence of potential solutions to the challenges. The dissertation was carried out as a literature review. The dissertation states that the DevOps method involves several different types of security challenges, but comprehensive solutions have been developed for them.

Keywords: DevOps, SecDevOps, data security

KUVIOT

SISÄLLYS

1 JOHDANTO

Ohjelmistotuotannon ominaispiirre on nykypäivänä järjestelmien monimutkaisuus. Tämä on luonut niin mahdollisuuksia kuin haasteitakin. Yksi suurimmista haasteista on ollut vastata ketterien ja iteratiivisten kehitysmetodien lisääntyneeseen muutosvauhtiin (IEEE, 2021). Näihin ongelmiin on kehitetty vastaamaan kehitysmetodi nimeltään DevOps

DevOps on toimintamalli, jossa ohjelmistokehitysprosessia pyritään tehostamaan siten, että kehitystiimi (engl. development team) ja tuotantotiimi (engl. operations team) kootaan yhteen prosessiin. DevOps tarkoittaa siis kehittämisen (Dev) ja toiminnan (Ops) yhdistämistä. DevOpsin avulla voidaan muun muassa parantaa kehittäjien ja tuotantotiimien välistä yhteistyötä, lisätä käyttöönottojen tiheyttä ja helppoutta, lisätä joustavuutta asiakkaiden tarpeiden huomioinnissa ja parantaa koodin laatua kehittäjäyhteistyötä (Mohan, & Othmane, 2016).

DevOpsiin liittyy kuitenkin myös muutamia ongelmia, jotka valitettavasti koskevat hyvin usein turvallisuutta. Turvallisuushaasteet liittyvät usein siihen, että kehitys- ja tuotantotiimi eivät tue toistensa motiiveja ja tarkoituksia (Jaatun ym, 2017). Tällaisia ongelmia syntyy esimerkiksi silloin, kun halutaan kehittää ohjelmistoa nopeassa tahdissa, eikä oteta huomioon tuotantotiimin motiiveja turvallisuuteen. Nykyajan ilmapiirissä turvallisuuteen tähtääviä toimenpiteitä on alettu pitää velvoittavuuden lisäksi myös hyödyllisinä ja tarpeellisina osina järjestelmää. Myös DevOpsin turvallisuuteen on haluttu alkaa käyttää enemmän aikaa ja resursseja, ja sen yhteydessä on alettu myös puhua termeistä SecDevOps ja DevSecOps, jossa turvallisuus on nostettu osaksi koko kehitysprosessin kulkua. ”Sec” tarkoittaa tässä turvallisuutta (security).

Joillakin teollisuuden aloilla DevOpsissa on alettu käyttää työkaluja, jotka on yksinomaan tarkoitettu turvaamaan ohjelmistotuotannon turvallisuutta. Tällaiset työkalut voivat perustua esimerkiksi automaatioon tai toimintatapoihin, esimerkiksi valvonnan lisäämiseen tai pääsyoikeuksien rajaamiseen. Monet työ-

kalut eivät kuitenkaan ole yleistettävissä tuotannonaloihin kohdistuvien uhkien erilaisuuden vuoksi (Dullmann, Paule, & van Hoorn, 2018). Tässä tutkielmassa onkin tarkoitus selvittää

- Minkälaisiin haasteisiin DevOps-järjestelmäkehitysmetodia sovellettaessa yleisesti törmätään, ja millaisia ratkaisuja tietoturvahaasteisiin on kehitetty

Kandidaatintutkielmani tutkimusmenetelmä on kirjallisuuskatsaus, jossa tutkimusaineistoa on kerätty eri tietokannoista hakusanoilla, kuten ”DevOps”, ”SecDevOps”, ”DevSecOps”, ”Secure DevOps”, sekä näiden hakusanojen yhdistelmiä. Pääasiallisia tietokantoina ovat olleet Jykdok ja IEEE. Aineiston valintaan on vaikuttanut sitatointien määrä, sekä tehdyt vertaisarvioinnit, mutta koska tutkimusmateriaali on vielä verrattain uutta, on valintaan saattanut vaikuttaa myös muut tekijät. Lisäksi lähteitä on kerätty tietokannoista löytyneiden, hyväksi havaittujen artikkeleiden lähteistä, ja tätä toimintamallia on jatkettu aina niin kauan, kun hyviä lähteitä on löytynyt. Yhtenä lähteenä on käytetty internetistä löytynyttä blogia, koska paras määritelmä mikropalveluille löytyi sieltä. Lopullinen lähteiden valinta on tapahtunut lähteiden tarkastelun jälkeen sillä perusteella, mitkä vastasivat parhaiten tutkimuskysymyksiin. Lähteitä on ollut arvioitavana parhaimmillaan noin 60 ja tutkielman ollessa valmis, on käytetty yhteensä 31 lähdettä.

Aiheen tutkimus on oleellista siksi, että DevOps on ollut joillakin mittareilla arvioituna jopa räjähdysmäisessä suosiossa viime vuosina, mutta sen turvallisuuteen ei nähdäkseni ole kiinnitetty tarpeellista huomiota. Kokoava tutkimus DevOpsiin liitetyistä riskeistä lisää tietoisuutta ja vaikuttaa siten myös siihen, että asiaan kiinnitetään huomiota. Tutkimuksen tulosten voidaan olettaa kiinnostavan yrityksiä, joissa DevOps on käytössä tai jossa se aiotaan ottaa käyttöön.

Tutkielman ensimmäisessä sisältöluvussa määritellään DevOps ja sen ominaisuudet, hyödyt sekä haasteet. Toisessa luvussa käsitellään tietoturvallisuutta yleisesti, sekä sitä, minkälaisia uhkia käsitetään liittyvän DevOps-ympäristöön. Kolmannessa kappaleessa esitellään eri työkaluja, menetelmiä ja kulttuureja, joita on kehitetty vastaamaan tietoturvahaasteisiin DevOps-ympäristössä. Tutkielma päättyy yhteenvetoon.

2 DEVOPS

Tämän luvun tarkoituksena on selvittää, mitä DevOps tarkoittaa käsitteenä ja mitkä ovat sen ominaisuudet, hyödyt ja haasteet eri toimijoille. Tarkoitus on antaa tarpeeksi lukijalle tarpeeksi laaja kuva siitä, mitä kaikkea DevOpsiin kuuluu ja miten se sijoittuu ohjelmistokehityksen kentällä.

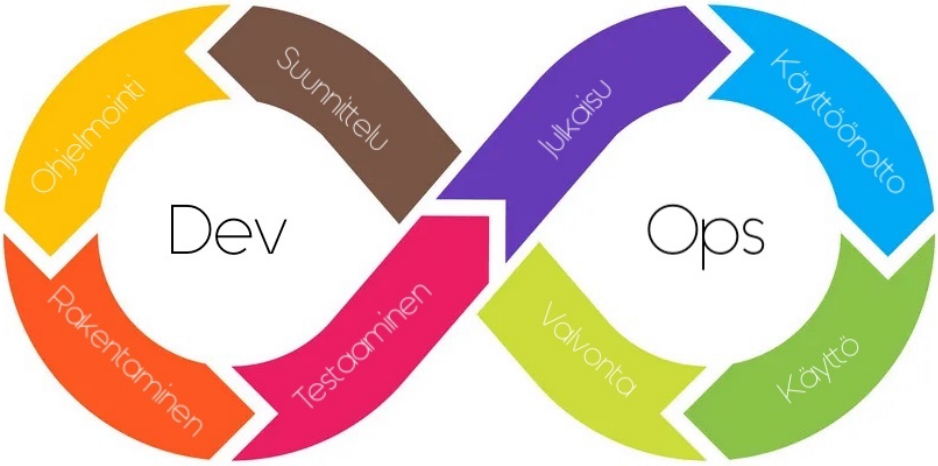
2.1 DevOpsin määritelmä

Ohjelmisto- ja IT-teollisuudessa on monesti kyse nopeudesta ja tehokkuudesta. Toisin sanoen tuotteita ja päivityksiä halutaan saada jatkuvasti nopeammin markkinoille. Toisaalta on kuitenkin nähtävissä, että kehittäjät ja toiminnoista vastaavat toimijat ovat usein ristiriidassa. Kehittäjät haluavat julkaista ohjelmistoja nopeammin ja useammin, kun taas toiminnoista vastaavat henkilöt haluavat turvata infrastruktuurin vakauden (Farroha & Farroha, 2014). Lisäksi Rami Juvosen (2018) mukaan etenkin vesiputousmalliin pohjautuvilla menetelmillä on yleisesti tapana päätyä 90 prosenttisesti valmiiksi määräaikaan mennessä. Jäljelle jäävä 10 prosenttia on tyypillisesti integrointia ja virheenkorjausta, johon kuluu huomattavasti enemmän aikaa ja työtä kuin oli ajateltu (Juvonen, 2018).

Yhdeksi paradigmaksi tuotteiden tuomiseksi nopeammin markkinoille, mutta toisaalta ottamaan huomioon ohjelmiston elinkaaren ja käyttäjien kaikki tarpeet, on kehitetty toimintamalli nimeltä DevOps. DevOps-termi muodostuu englanninkielisistä sanoista *development* (suom. kehitys) ja *operations* (toiminnot). *Development*, eli kehitys, kuvaa tässä yhteydessä kaikkia niitä toimia, joiden käsitämme kuuluvan ohjelmistokehityksen tekniseen toteutukseen. Tämä tarkoittaa siis suunnittelua (plan), ohjelmointia (engl. code), rakentamista (build) ja

testaamista (test). Termi *operations*, eli toiminnot, kuvaa kaikkia ohjelmistotuotannon toiminnallisia osia, joiden ei käsitetä varsinaisesti kuuluvan ohjelmiston kehitysvaiheeseen. Käytännössä tämä tarkoittaa julkaisua (engl. release), käyttöönottoa (deploy), käyttöä (operate) ja valvontaa (monitor) (Farroha & Farroha, 2014) (kuvio 1). DevOps on erään määritelmän mukaan joukko käytäntöjä, joiden tarkoituksena on lyhentää järjestelmän muutoksen tekemisen ja muutoksen normaaliin tuotantoon siirtymisen välistä aikaa ja samalla varmistaa korkea laatu (Zhu, Bass, Champlin-Scharff, 2016). Lwakataren, Kuvajan ja Oivon (2016) mukaan ensimmäisen tieteellisen määritelmän Dev-Opsille antoi Penners ja Dyck vuonna 2015. Vuonna 2016 Lwakatare, Kuvaja ja Oivo ehdottivat annetun määritelmän pohjalta uudeksi määritelmäksi seuraavaa:

> ” **[**DevOps on] ajattelutapa, joka perustuu käytäntöihin, jotka rohkaisevat ohjelmistokehitysorganisaation tiimien, erityisesti kehitys- ja IT-toimintojen tiimien, välistä poikkitoiminnallista yhteistyötä, jotta voidaan käyttää kestäviä järjestelmiä ja nopeuttaa muutoksen toteutumista” (Lwakatare, Kuvaja & Oivo, 2016).

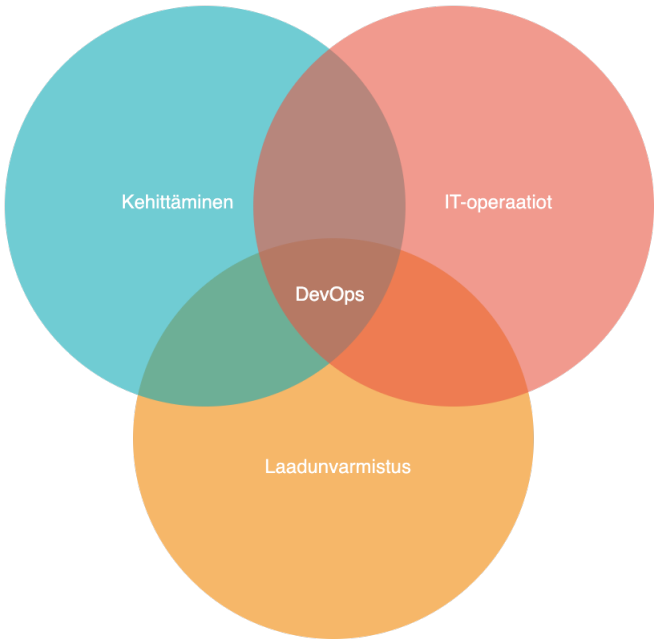


KUVIO 1 DevOps-kehä, suomennettu (Farroha & Farroha, 2014)

2.2 DevOpsin ominaisuudet ja tavoitteet

DevOps on siis joustavaa ja nopeaa liiketoimintaprosessien kehittämistä ja provisiointia. Provisiointi tarkoittaa tässä yhteydessä sitä, että julkaisut kyetään ottamaan käyttöön nopeammin ja mahdollisesti automaatiota käyttäen. DevOpsin tarkoituksena on yhdistää kehitys- ja tuotantomaailma toisiinsa käyttämällä automatisoitua kehitystä, käyttöönottoa ja infrastruktuurin valvontaa. Kyseessä on eräänlainen organisaatiomuutos tai kulttuurin muutos, jossa eri tiimit työskentelevät yhdessä jatkuvan operatiivisen toimituksen parissa sen sijaan, että hajautetut ryhmät suorittaisivat toimintoja erikseen. Organisaatiomuutos tarkoittaa myös muutosta kehitysyhteistyön, laadunvarmistuksen ja toiminnan välillä. Sen sijaan, että organisaatiot käyttäisivät prosessikonsepteja, jotka eivät koskaan toteudu, organisaatiot toimittavat jatkuvasti pieniä päivityksiä. Tällainen toimintatapa auttaa toimittamaan arvoa nopeammin ja jatkuvasti, vähentäen tiimin jäsenten välisestä väärinymmärryksestä johtuvia ongelmia ja nopeuttaen ongelmien ratkaisua (Ebert, Gallardo, Hernantes & Serrano, 2016).

DevOpsin tärkein tavoite on yhdistää kehittämiseen, IT-operaatioihin ja laadunvarmistukseen liittyvät toiminnot toisiinsa jouhevasti (kuvio 2). Käytännössä tämä tarkoittaa sitä, että ohjelmistoprosessin tehottomuutta korjataan DevOpsiin kuuluvilla viestintä- ja yhteistyötavoitteilla. Tosin tätä voidaan pitää myös DevOpsissa haasteena, sillä DevOpsin yksityiskohtaisia käyttöönotto-ohjeita ei ole laajasti hyväksytty tai sovittu (Perera & Bandara, 2016).



Kuvio 2 DevOpsin osa-alueet

Perusongelmia, joita DevOps on kehitetty ratkaisemaan, ovat sopeutumiskyky muutokseen, ohjelmistotuotteen saaminen nopeammin markkinoille ja korkean laadun varmistaminen siten, että kokonaiskustannukset säilyvät mahdollisimman alhaisina. DevOps-periaatteet eivät ole sidottuja mihinkään tietyn tyyppiseen tuotteeseen tai palveluun, vaan sitä voidaan soveltaa kaikenlaisissa ohjelmistoprojekteissa, monimutkaisuudesta tai organisaation tai projektin koosta riippumatta (Virmani, 2015). Virmani myös toteaa, että tavoitteellista ei ole siirtyminen nopeasti ”ei-DevOps”-tilasta suoraan DevOpsin täysimittaiseen integrointiin, vaan organisaatioissa voidaan tehdä rakennemuutoksia myös hitaammalla aikavälillä.

DevOps liittyy läheisesti jatkuvan toimituksen (engl. continuous delivery) ja julkaisemisen (deployment) käsitteeseen, jonka virtuaalinen arkkitehtuuri mahdollistaa IaC:n (infrastructure as code), jolloin suurin osa käyttöönotosta ja konfiguroinnista tehdään käyttämällä esikonfiguroituja komentosarjoja. (Abrahamsson ym., 2020). DevOpsin arkkitehtuuri rakennetaan useimmiten mikropalvelun keinoin (Mishra & Otaiwi, 2020). Mikropalvelu tarkoittaa hajautettua sovelluksen arkkitehtonista mallia, jossa erikseen käyttöönotettavat palvelut suorittavat annettuja tehtäviä ja kommunikoivat verkkorajapinnan kautta. Mikropalveluarkkitehtuuri on pilvipohjainen ja sen tavoitteena on toteuttaa ohjelmistojärjestelmiä pienten palveluiden kokonaisuutena (Balalaie, Heydarnoori & Jamshidi, 2016). DevOpsissa tämä tarkoittaa sitä, että tiimit liittävät toiminnallisuuksia mikropalveluihin ja rakentavat suurempia kokonaisuuksia liittämällä mikropalvelut yhteen (Microsoft, 2021). Juuri tällaisen infrastruktuurin ansiosta DevOps-menetelmää hyödyntävät organisaatiot voivat tuottaa uusia päivityksiä jopa useita kertoja päivässä.

2.3 DevOpsin hyödyt ja haasteet

Kolmelle eri suomalaiselle ohjelmistoyritykselle toteutetussa kyselytutkimuksessa (Riungu-Kalliosaari, Mäkinen, Lwakatare, Tiihonen & Männistö, 2016) havaittiin, että DevOpsin käyttöönotosta on seurannut lukuisia eri hyötyjä kohdeyrityksissä. DevOpsin havaittiin vaikuttavan erityisesti ohjelmistotuotteiden toimitustahtiin, mikä auttoi toteuttamaan enemmän toiminnallisuuksia pienemmällä toimitusvälillä. Myös tuotteiden laadun koettiin parantuvan muun muassa

siksi, että menetelmän avulla voitiin tehostaa palautesyklejä. Haastateltavien yritysten näkemysten mukaan DevOps auttaa yrityksiä testaamaan erilaisia ideoita nopeasti ja palautteen perusteella tekemään päätöksiä ketterästi.

Lyhyt toimitustahti koettiin merkittäväksi hyödyksi erityisesti yksittäiselle asiakkaalle, jolle voidaan toimittaa uusia ominaisuuksia nopeammin. Uusien ominaisuuksien tuominen pienemmissä ja nopeammissa erissä myös vaikuttaa siihen, että käyttäjällä on mahdollisuus havaita ja oppia ominaisuus tehokkaammin. Luonnollisesti yksittäisistä toiminnoista saatu palaute on tällöin hyödyllisempää kehittäjälle (Riungu-Kalliosaari ym., 2016).

Kyselytutkimuksessa todettiin, että automaattiset rakennus-, testaus-, ja käyttöönottoprosessit ovat merkittävä DevOpsin etu. Automatisointi vaikuttaa muun muassa siihen, että organisaatiot voivat kanavoida enemmän ominaisuuksia tuotanto- ja toimitusputkiin. Toimitusputkisto (engl. delivery pipeline) kuvaa ohjelmistoprosessin järjestelmällistä läpikäyntiä ja kehitystyökalujen kohdistamista ohjelmistotuotteen- tai päivityksen vapauttamiseksi saumattomalla tavalla (Moyón, Soares, Pinto-Albuquerque, Mendez, & Beckers, 2020). Ilmiselvä vaikutus automaatiossa on se, että se vähentää ihmisen tekemää työtä ja sen vuoksi vähentää mekaanisen työn vaivaa. Automaation on kyselytutkimuksen perusteella myös todettu parantavan laadunvarmistusta. Voidaan todeta, että varsinaista tuotantoa tehostavaa hyötyä ajaa prosessien tavoitteellinen automatisointi yksi kerrallaan siten, että ohjelmiston matka kehittäjältä käyttäjälle on mahdollisimman suoraviivainen. Esimerkiksi ohjelmistopäivityksen kestäessä manuaalisen testauksen ja lukuisten toimitusten vuoksi hyvin pitkään, voidaan näitä vaiheita automatisoida ja siten niiden kestoa ja luotettavuutta parantaa oleellisesti (Juvonen, 2018).

Merkittäväksi DevOpsin vaikutukseksi todettiin se, että kehitys- ja tuotantotiimit pakotetaan toimimaan entistä tiiviimmässä vuorovaikutuksessa, minkä todettiin vaikuttavan erityisesti yhteistyön ja viestinnän tehostumiseen. Käytännössä tämä voi tarkoittaa esimerkiksi sitä, että tiimien välisen tiedon halutaan vaihtuvan ja tiimit ovat valmiimpia jakamaan omia kokemuksiaan. Mielenkiintoinen työyhteisöön vaikuttava merkitys DevOpsilla on havaittu olevan myös työhyvinvoinnin kannalta. Kyselytutkimuksessa haastateltu työntekijä kertoi säännöllisten julkaisujen auttavan vähentämään stressitasoja, koska suurista muutoksista johtuvat epämukavuustekijät, kuten ahdistus on eliminoitu (Riungu-Kalliosaari ym., 2016).

Bou Ghantousin ja Gillin (2017) mukaan DevOpsiin siirtymisellä on lukuisten hyötyjen lisäksi myös pääasiassa tunnistettu neljää erilaista haastetta. Ensinnäkin DevOpsiin siirtymisessä voi olla hankalaa päästä yli ajatuksesta, että kehitys- ja tuotantotiimi ovat samaa tiimiä, toisin sanoen voi olla hankalaa mentaalitasolla päästä yli ”Dev vs. Ops” -asettelusta. Toisena haasteena koettiin perinteisestä ohjelmistoarkkitehtuurista siirtyminen uudempaan

mikropalveluarkkitehtuuriin. Kolmantena haasteena pidettiin sitä, että DevOps on metodina työkalukeskeinen ja siten kalujen toimimattomuus voi aiheuttaa ongelmia. Lisäksi näiden työkalujen integrointi ja ylläpito voi osoittautua ongelmalliseksi ja vaikeaksi. Samoin koettiin haasteellisena se, että kehittäjä- ja tuotantotiimillä saattaa olla käytössään erityppisiä työkaluja, joiden sovittaminen keskenään on hankalaa (Bou Ghantous & Gill, 2017). DevOpsiin liittyvät haasteet ovat kenties selitettävissä ja yleistettävissä muutosvastarintailmiöllä, jota kohdataan lähes kaikissa organisaatiomuutoksissa.

3 TIETOTURVA OHJELMISTOKEHITYKSESSÄ

Itsestään selvä ja yleisesti hyväksytty tosiasia on se, että tietoturvaa halutaan yhä enenevissä määrin vaalia ja lisätä. Järjestelmien turvallisuuden merkityksen voidaan nähdäkseni olettaa kasvavan samassa tahdissa digitalisaation etenemisen kanssa. Sen lisäksi, että jokaisella ihmisellä on henkilökohtaiset motiivit suojella omia henkilötietojaan, tietoturva on nykyisin myös varsin säännelty informaatioteknologian osa-alue. Merkittävin tietosuojaa koskeva asetus on EU:n yleinen tietosuoja-asetus (GDPR).

Tietoturvaan keskittyminen järjestelmäkehityksen yhteydessä lisääntyneestä ei mielenkiinnosta huolimatta ole kovinkaan uusi asia. Jo yli 40 vuotta sitten tietoturvaan on osattu kiinnittää huomiota. Muun muassa vuonna 1979 ilmestyneessä tutkimuksessa (Denning & Denning, 1979) selvitetään, että turvallisuuden hallinta tietojen käsittelyssä koostuu lukuisista eri osa-alueista, jotka kaikki tulee ottaa huomioon tietojen turvallisen säilyvyyden varmistamiseksi. Tietoturvan varmistaminen on ohjelmistoyritykselle monelta kannalta hyvin tärkeää. Tietosuojalainsäädännön näkökulmasta on tietysti oleellista se, että palveluntarjoaja noudattaa lakia, joiden rikkomisesta aiheutuvat sanktiot ovat verrattain suuria, mutta myös rikkomuksista tai huolimattomuuksista johtuvat tietomurrot saattavat aiheuttaa pysyvää mainehaittaa yritykselle ja pahimmillaan yritys saattaa joutua konkurssiin. Tällaisia tapauksia on viime vuosien aikana tapahtunut useampiakin, joista Suomessa eniten esillä lienee ollut Psykoterapiakeskus Vastaamon tietomurto.

Tässä luvussa käsitellään tietoturvaa ja tietoturvahaasteita yleisesti ja DevOpsin näkökulmasta ja sitä, minkälaisia tietoturvahaasteita käsitetään esiintyvän DevOps-ympäristössä. Tavoitteena on vastata tutkimuskysymyksen ensimmäiseen osaan ”minkälaisiin haasteisiin DevOps-järjestelmäkehitysmetodia sovellettaessa yleisesti törmätään”.

3.1 Turvallisuus ohjelmistokehityksessä yleisesti

Science of Security (SoS) on tutkimusalue, jonka tarkoituksena on pyrkiä soveltamaan tieteellistä lähestymistapaa turvallisten ja luotettavien tietojärjestelmien tutkimukseen ja suunnitteluun (Jaatun, Cruzes & Luna, 2017). Tutkimusalan tavoitteena on kehittää peruslakeja, joiden avulla voidaan tehdä ennusteita ohjelmiston turvallisuudesta. Ohjelmistoturvallisuuden alalla kuitenkin ainoa ennuste, joka voidaan täysin luotettavasti tehdä, on se, että järjestelmän turvallisuus pettää lopulta aina, kun löytyy tarpeeksi motivoitunut hyökkääjä (Jaatun ym., 2017). Jaatun kuitenkin toteaa, että ohjelmistokehityksen turvallisuuden tutkiminen on tärkeää ja tarpeellista, koska on tärkeää tutkia vaihtoehtoisia toimintatapoja hyökkäyksen sattuessa ja ymmärtää sitä, mitkä vähimmäisominaisuudet hyökkääjällä tulee olla, mikäli se mielii hyökätä ohjelmistoa vastaan. Tieteenala kuitenkin kärsii edelleen empiirisen aineiston puutteesta erityisesti ketterässä kontekstissa, minkä vuoksi sitä ei voi aina pitää tieteellisesti täysin uskottavana (Jaatun ym., 2017).

Ohjelmistokehityksessä turvallisuuden voi käsittää useammalla eri tavalla. Useimmiten ohjelmistoturvallisuus käsitetään nimenomaan sovellusten kykynä kestää sen haavoittuvuuksiin kohdistuvia hyökkäyksiä. Digitalisaation edetessä turvallisuusuhkia on kuitenkin käsillä entistä enemmän ja uhkien lisääntyessä tietoturvan huomioiminen ohjelmistokehityksen elinkaaressa on äärimmäisen tärkeää kaiken kokoisille yrityksille (Assal & Chiasson, 2018).

Organisaatioissa suhtaudutaan tietoturvaan hyvin eri tavalla. Assalin & Chiassonin (2018) mukaan joissakin organisaatioissa turvallisuus huomioidaan alusta asti tuotannon jokaisessa työvaiheessa, kun taas toisaalla saatetaan välinpitämättömyyden vuoksi turvallisuus sivuuttaa kokonaan tai tehdä minimaalinen turvallisuusintegraatio. Suurimmassa osassa organisaatioissa totuus lienee jossakin näiden kahden skenaarion välimaastossa. Assalin ja Chiassonin (2018) haastattelujen perusteella he totesivat, että turvallisuuskäytännöt organisaatioissa eroavat hyvin oleellisesti kirjallisuudessa tunnistetuista parhaista käytännöistä. Nämä käytännöt jätetään usein huomiotta, koska niiden noudattaminen lisäisi työntekijöiden työtaakkaa liiallisesti, jolloin kustannustehokkuus kärsii. Heidän mukaansa myös kehittäjät ovat suurin yksittäinen riskitekijä turvallisuudelle, mutta syy turvallisuuden laiminlyönneistä ei ainakaan yksiselitteisesti ole heidän, vaan ongelma ulottuu yrityshierarkiassa ylemmille tasoille. Assal ja Chiasson (2018) myös mainitsevat, että lisäturvallisuuden tarpeen korostuessa tilannetta voisi korjata esimerkiksi automaatiolla, joka on oleellinen DevOps-kehityksen turvallisuustekijä.

3.2 Turvallisuus DevOps-ympäristössä

Kasvavasta suosiostaan huolimatta, DevOpsin käyttöönottavia ohjelmistokehitysorganisaatioita huolettaa sen turvallisuuteen liittyvät epävarmuudet. Siinä missä Zhu (ym., 2016) esittää, että Devops mahdollistaa ohjelmistotuotteen vapauttamisen markkinoille nopeasti siten, että laatu ei kärsi, Jaatun ym. (2017) pohtivat sitä, kattaako laatu tässä tapauksessa myös tietoturvan laadun. Toisaalta Katal (ym., 2019) pitävät DevOpsiin kiinteästi yhdistettyä laatua ratkaisuna tietoturvaongelmiin, sillä heidän näkemyksensä mukaan ohjelmiston laatu voi vähentää tietoturva-aukkoja. Katalin mukaan DevOpsin nopean tuotantoputken ominaisuus mahdollistaa sen, että myös virheet voidaan korjata nopeammin. Samalla kuitenkin herää kysymys, onko turvallista korjata jo käyttöön päästettyä koodia vasta jälkeenpäin. Lähdekirjallisuus ei siis anna kovin selkeää lähtökohtaa sille, onko DevOpsia syytä pitää luotettavampana vai vähemmän luotettavana suhteessa muihin ohjelmistokehitysparadigmoihin. Ainoa tosiasia tässä vaiheessa on se, että myös DevOps-kehitykseen kohdistuu monia erilaisia tietoturvahaasteita.

Moyonin ym. (2020) mukaan ohjelmistokentältä puuttuu vielä menetelmiä, joilla voidaan osoittaa turvallisuusstandardien noudattaminen DevOpsia käytettäessä ja aihetta koskevia julkaisuja on vielä verrattain vähän.

3.3 Tietoturvahaasteet DevOps-ympäristössä

Tietoturvahaasteiksi on DevOps-kehityksessä tunnistettu lähdekirjallisuuden perusteella monenlaisia eri tekijöitä. Useimmin esiintyvät tuotantoputken turvallisuuden varmistaminen sekä nopean julkaisun ja turvallisuuden välinen suhde. Lisäksi haasteiksi tunnistettiin myös lisääntyneiden sisäpiiriläisten määrä kehitysprosessissa, automaattisten ja ei-automaattisten ratkaisujen välillä tasapainottelu, oikein tehty turvallinen vaatimusmäärittely, kehittäjien riittävä taitotaso sekä oikeiden työkalujen löytäminen julkaisuputkessa.

Yrityksissä, joissa DevOps-käytänteitä on alettu käyttää, voidaan siis ottaa ohjelmistomuutokset käyttöön nopeasti automaattisen putkilinjan avulla. Tästä syystä saattaa käydä niin, että ohjelmistomuutokset eivät käy läpi tarpeeksi kattavia tietoturvatarkastuksia, mikä saattaa johtaa haavoittuviin ohjelmistoihin (Rahman & Williams, 2016). Saman huolen käyttöönottoputken turvallisuudesta esittää myös Bass ym. (2015). Bassin (ym.) mukaan putkilinjan turvaamista

vaikeuttaa putkistossa olevien työkalujen suuri määrä, joista jokaisella voi olla täysin erilaiset turvaamismenetelmät.

Huolimatta siitä, että automaattisten käyttöönottoputkien avulla organisaatiot kykenevät toimittamaan ohjelmistomuutoksia nopeasti, niin samalla tämä automaatiokehitys on yksi suurimmista Dev-Opsiin liittyvistä tietoturvariskeistä. Sen voidaan tietyissä tapauksissa havaita kannustavan kompromissiin prosessia hidastavien tietoturvakäytäntöjen ja lyhyen toimitusvälin välillä, mikä ennen pitkää johtaa järjestelmien haavoittuvuuteen. Tietoturvan noudattamista koskevat toiminnot, kuten dokumentointi, saatetaan kokea DevOps-tiimin ylikuormittamisena, joka pidentää ohjelmiston tuottamiseen kuluvaa aikaa. Tietoturvatoiminnot eivät myöskään tuota asiakkaalle lisäarvoa, minkä takia se nähdään rasitteena sekä tiimille, että asiakkaalle. Automatosoinnissa riskinä on aina myös se, että tietoturvan hallitsemiseen tarkoitetut automaattiset työkalut saatetaan valita väärin. Nopean toimituksen ja turvallisuuden välillä tasapainoilu onkin yksi useimmin esiintyvä kokonaisuus, jota käsitellään DevOpsin turvallisuuteen liittyvissä artikkeleissa (Mansfield-Devine, 2018).

Joissakin lähteissä mainitaan tietoturvahaasteeksi myös DevOpsin nopean filosofian vuoksi syntynyt lisätarve kehittäjien tietoturvaosaamiselle. Koodin tekijän työtä saattaa alkaa ohjata tarve tietää, mitkä koodin osat tulee tarkistaa perusteellisesti. Tämä lisää oleellisesti lisäkoulutuksen tarvetta (Ferry ym., 2018).

Tietoturvahaasteena nähdään myös automaattisten ja ei-automaattisten ratkaisujen välinen tasapainottelu. Jaatun (ym. 2017) pohtii sitä, että vaikka iso osa tietoturvatoiminnoista olisi automatisoitavissa, niin ei välttämättä kannata toimia, vaan oleellisempaa on lisätä kehittäjien motivaatiota tuottaa turvallista tuotetta. Jaatun näkee, että ei-automaattiset ratkaisut ovat tässä yhteydessä tärkeitä, esimerkiksi manuaalisen koodintarkistuksen ja vaatimusmäärittelyn muodossa. Vaatimusmäärittelyn osalta onkin todettavissa, että sen ollessa puutteellinen turvallisuuden osalta, voi se heijastua koko prosessin turvallisuuteen. Vaatimusmäärittelyn seuraamisessa saatetaan joskus myös valita vääränlaisia työkaluja ja seurata myöhemmin vääriä mittareita, jolloin tietoturvan eheys kärsii (Dullmann, Paule, & van Hoorn, 2018)

Ahmadvand (ym. 2018) näkee haasteena kehitystiimien välisen yhteistyön myötä syntyneen ohjelmistoprosessin sisäpiiriläisten määrän. DevOpsin jakamisperiaate ja yhteisökulttuuri voivat ajaa tiimit tilanteeseen, jossa sisäpiiriläisten pääsy järjestelmiin on laajempi kuin ennen. Voidaan siis todeta, että yhteistyö myös lisää tietovuodon riskiä.

4 TIETOTURVAHAASTEISIIN KEHITETYT RATKAISUT

Tässä luvussa esitellään keskeisimpiä ja useimmin esiintyviä ratkaisuja tietoturvahaasteisiin vastaamiseen DevOps-ympäristössä menetelmien, käytäntöjen, työkalujen ja kulttuurin muodossa.

4.1 Neljä tietoturvakäytäntöä

Rahman ja Williams (2019) havaitsivat neljää erilaista tietoturvatoimintoa tutkiessaan turvallisuuskäytäntöjä DevOps-organisaatioissa. Yhdeksässä yrityksessä (muun muassa Facebook, Netflix, SAS, Mozilla Firefox) käytettiin kaikissa vähintään kolmea seuraavista neljästä toiminnosta; automaattitoimintojen käyttö, yhteistyö eri osastojen välillä, turvallisuuskoulutuksen tarjoaminen kehitystiimin jäsenille ja ei-automaattisten suojauskäytäntöjen käyttö

4.1.1 Automaatiotoimintojen käyttö

Automaatioon perustuvien työkalujen käyttö on kiinteä osa DevOps-kehitystä. Siksi tämä koskee luonnollisesti myös automaattisten tietoturvatoimintojen käyttöä. Rahmanin ja Williamsin (2016) mukaan automaattisten toimintojen käyttö sisältää viisi osaa; ohjelmiston valvonnan, koodintarkistuksen, järjestelmätestauksen, palomuurin ja lisensoinnin automatisoinnin. Näistä yleisin keino on valvonnan ja testauksen automatisointi. Harvinaisempaa kohdeyrityksissä selvästi oli lisensoinnin automatisointi, joka oli harvemmin käytössä kuin ei-käytössä. Automaattiset turvallisuudenhallintatyökalut säästävät aikaa, kustannuksia ja

poistavat inhimillisen erehdyksen mahdollisuuden turvallisuusprotokollasta (Battina, 2017).

Ohjelmiston valvonnan automatisointi tarkoittaa sitä, että järjestelmä kerää ja tallentaa automaattisesti dataa, joka liittyy järjestelmän toimintaan. Käytännössä tämä tarkoittaa dataa muun muassa prosessorien toiminnasta, muistin hyödyntämistä lisäanalyysia varten. Koodintarkistuksen automatisointi tarkoittaa sellaisia automaattisia toimintoja, jossa koodin tarkistus ja asianmukaisen palautteen antaminen ohjelmistokehittäjille tapahtuu käyttämällä esimerkiksi avoimen lähdekoodin automaattisia analyysityökaluja. Järjestelmätestauksen automatisointi tarkoittaa toimintaa, jossa automaattiset työkalut suorittavat testaustehtäviä eri testejä varten. Testata voidaan esimerkiksi integrointia tai toiminnallisuutta (Rahman & Williams, 2016).

Palomuurin automatisointi tarkoittaa organisaation käyttämien palomuurien johdonmukaiseen hallitsemiseen käytettäviä automaattisia työkaluja. Ohjelmiston lisensointi tarkoittaa toimintaa, jossa käyttäjät voivat ostaa, asentaa ja käyttää ohjelmistoja ohjelmistotoimittajan asettamien ehtojen mukaisesti. Lisensoinnin automatisointi tarkoittaa tämän toiminnan valvontaa automaattisten työkalujen avulla.

Voidaan siis todeta, että yleisesti käytössä olevat käytännöt, kuten automatisointi, voivat osaltaan myös lisätä nimenomaan turvallisuutta DevOpsissa. Rahman ja Williams (2016) kuitenkin mainitsevat myös, että väärien automaatiotyökalujen valinta saattaa johtaa tietoturvan heikkenemiseen.

4.1.2 Tiimien välinen yhteistyö

Lisääntynyt yhteistyö voidaan nähdä turvallisuustekijänä erityisesti siksi, että kaikissa yhdeksässä kohdeyrityksessä on erilliset turvallisuus- ja kehitystiimit. Tämä tarkoittaa sitä, että turvallisuus kyetään ottamaan kenties mittavammin huomioon. Rahmanin ja Williamsin (2016) kyselyssä käy ilmi, että turvallisuustiimin yhteistyö sekä kehitys- että tuotantotiimin kanssa on tavallisesti vähintään kohtalaista, mutta merkille pantavaa on se, että kehitys- ja tuotantotiimin välinen yhteistyö vaikuttaisi kuitenkin olevan vielä suositumpaa. Turvallisuustiimin yhteistyö kehitys- ja tuotantotiimin kanssa on kuitenkin aktiivista vakiintuneissa DevOps-organisaatioissa.

Rahman & Williams (2016) uskovat kuitenkin myös, että tiimien välinen tiivis yhteistyö voi johtaa myös tietoturvan heikkenemiseen. Riski toteutuisi silloin, jos yhteistyötä ei valvota ja vaihdettavaa tietoa ei rajoiteta. Tällöin olisi hyödyllistä ottaa myös yhteistyössä käyttöön käytäntöjä, jotka estävät liiallisen tietojen vaihdon ja siten tietojen vuotamisen. Myös Farroha & Farroha (2014) näkevät

tiedon määrän, monimuotoisuuden ja saatavuuden kasvu on hälyttävää, jolloin tiedonhallintakäytänteitä on kriittistä määrittää.

Farroha & Farroha (2014) laajentavat turvallisen ohjelmistotuotannon käsitteen koskemaan myös muita ohjelmistoalan sidosryhmiä. He korostavat yhteistyön merkitystä myös sidosryhmien kanssa esimerkiksi siten, että on hyödyllistä kommunikoida sidosryhmien kanssa ja että riskienhallintastrategiaa on syytä noudattaa yksimielisesti. Sidosryhmien hyödyntämisestä on myös hyötyä liiketoiminnan, lain noudattamisen ja vaatimustenmukaisuuden hallinnan kannalta ja yhteistyö myös tasapainottaa liiketoimintaan liittyviä riskejä.

Myös Gartnerin raportissa (Ennaciri & Spafford, 2021) korostetaan, että sääntelystä vastuussa olevien johtajien tulisi optimoida ja yksinkertaistaa valvontatoimenpiteitä tekemällä yhteistyötä keskeisten kehitys-, tietoturva-, laki-, vaatimustenmukaisuus-, ja sisäisen tarkastuksen sidosryhmien kanssa. Ahmadvand, Pretschner, Ball & Eyring (2018) toteavat, että on tärkeää tietää, kuinka määritellä pääsyoikeudet ja asianmukainen dokumentaatio, joilla varmistetaan, että tieto ei ajaudu käyttäjille, joille se ei kuulu.

Turvallisuus-, kehitys- ja turvallisuustiimin välisestä yhteistyöstä on alettu käyttää myös termiä SecDevOps tai DevSecOps. Tämä tarkoittaa sitä, että turvallisuus (sec) mielletään yhtä merkittäväksi osaksi paradigmaa kuin kehitys ja toiminnot. SecDevOpsia käsitellään lisää myöhemmin tässä tutkielmassa.

4.1.3 Turvallisuuskoulutuksen tarjoaminen

Tietoturvallisuuden suurin tunnettu riski on käyttäjä itse. Siksi on tärkeää, että ohjelmiston käyttäjät koulutetaan käyttämään ohjelmistoa turvallisesti inhimillisten virheiden minimoimiseksi. Rahmanin ja Williamsin (2019) tutkimuksessa yhdeksästä kohdeorganisaatiosta seitsemällä on käytäntönä tarjota koulutusta kehitystiimille.

Turvallisuuskoulutuksen tarjoaminen on hyödyllistä esimerkiksi siksi, että tiimeissä lisääntyvä tietoisuus turvallisuudesta auttaa välttämään pullonkauloja, jotka johtuvat turvallisuusasiantuntijoiden puutteesta (Moyón ym., 2020). Lisäksi on yleisesti tiedossa, että kaikissa ohjelmiston elinkaaren vaiheissa inhimillinen erehdys on yksi suurimmista tietoturvan vaarantumisen aiheuttajista.

4.1.4 Ei-automaattisten suojaustoimintojen käyttö

Rahmanin ja Williamsin (2019) tutkimuksessa havaittiin 10 erilaista ei-automaattisia käytäntöä, jotka olivat kohdeorganisaatioissa käytössä. Näistä tärkeimpiä olivat turvallisuusvaatimusanalyysi, suojausasetusten konfigurointi, suojauskäytäntöjen suorittaminen, manuaalisten turvatestien suorittaminen, vaatimusten noudattaminen ja suunnittelun arviointi. Myös sellaisia toimintoja, kuten epäluotettavien syötteiden eristämistä, uhkamallinnusta ja riskianalyysiä esiintyi. Yhdeksästä organisaatiosta yli puolet käyttää kaikkia näitä toimintoja turvallisuuden edistämiseksi.

Moyónin ym. (2020) mukaan siinä missä IEC 62443 -standardiperheen osan, IEC 62443-4-1:n, noudattaminen on mahdollista lisäämällä edelleen automaatioon perustuvia työkaluja, niin kokonaan se ei niillä ole vielä standardin oikeellisen noudattamisen kannalta mahdollista. Heidän mukaansa 38 % toiminnoista on edelleen mahdotonta automatisoida siten, että standardien noudattaminen toteutuisi. Tästä syystä ei-automaattiset suojatoiminnot ovat myös sääntelyn vuoksi tarpeellisia DevOpsin tietoturvan hallinnassa.

Kuitenkin Vriesin (2014) mukaan perinteinen tietoturvalähestymistapa, joka keskittyy dokumentaatioon, manuaalisiin prosesseihin ja työkaluihin, ei sovellu jatkuvaan käyttöönottoon ja sellaiset työkalut tulisi korvata nykyaikaisemmilla työkaluilla, kuten automaatiolla. Vriesin näkemys ei siis Moyónin artikkelin perusteella ole mahdollista, mutta nähdäkseni Vriesin esitys koskeekin enemmän tulevaisuuden ihannetilaa, ja Moyónin artikkeli tarkastelee aihetta nykyisen sääntelyn valossa.

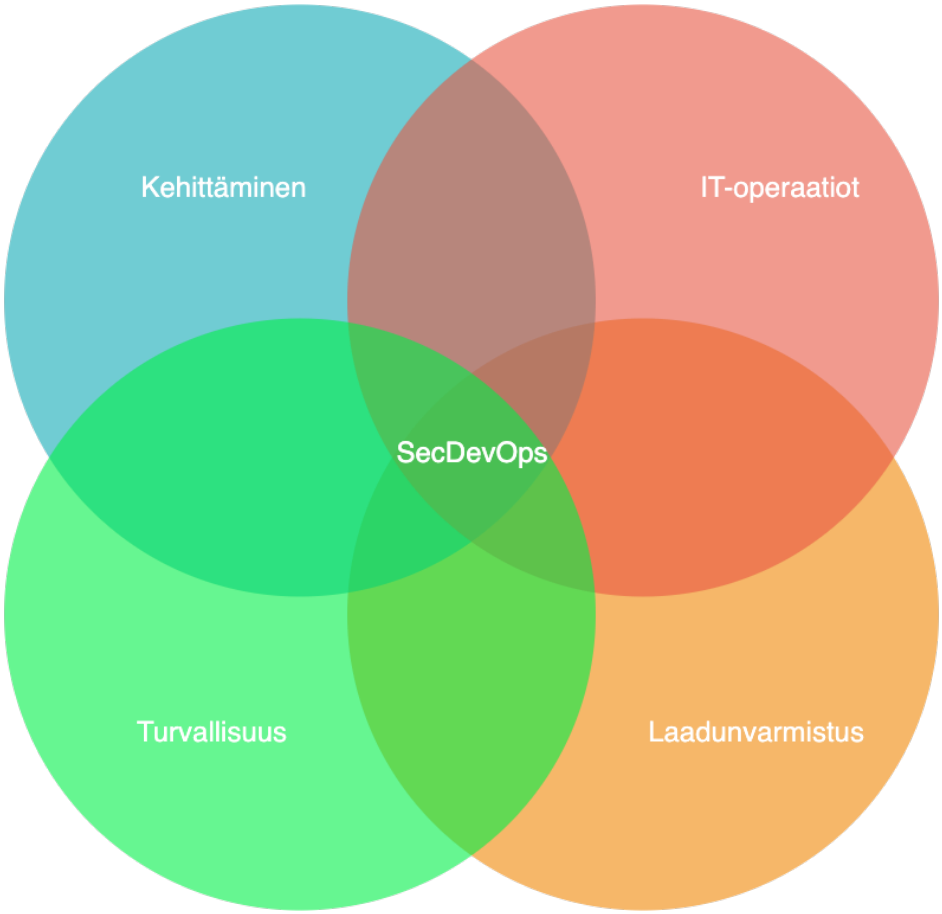
4.2 SecDevOps

SecDevOps esiintyy lukuisissa eri lähteissä vähintään mainintana, erityisesti silloin kun puhutaan DevOps-metodiin liittyvästä turvallisuudesta. SecDevOps tarkoittaa tietoturvakäytäntöjen integrointia DevOps-prosessiin käyttäen laadukkaita tietoturvatekniikoita ja lisäämällä turvallisuustiimien, kehitystiimien ja tuotantotiimien välistä yhteistyötä (Mohan & Othmane, 2016). SecDevOpsista käytetään usein myös termejä DevSecOps ja Secure DevOps, mutta tässä tutkielmassa pitäydytään termissä SecDevOps.

SecDevOps on metodi, joka vastaa suoraan DevOpsin tietoturvatarpeeseen. Sen konsepti on yrittää luoda ja sisällyttää nykyaikaisia tietoturvakäytäntöjä.

Kuten jo aiemmin tässä tutkielmassa on todettu, SecDevOpsin tarkoitus laajentaa kehittäjien ja toimintatiimin välistä yhteistyötä siten, että tietoturva-asiantuntijat otetaan mukaan kehitystyöhön prosessin alusta alkaen (Myrbakken & Colomo-Palacios, 2017) (kuvio 3). Myrbakken ja Colomon tekemän kirjallisuuskatsauksen mukaan SecDevOps pitäisi nähdä välttämättömänä laajennuksena DevOpsiin, ja siten sisällyttää tietoturvavalvonta ja -prosessit ohjelmistokehityssykliin. Tähän paras keino Myrbakken ja Colomo-Palacioksen mukaan on nimenomaan turvallisuustiimin, toimintatiimin ja kehitystiimin välisen yhteistyön kehittäminen ja sujuvoittaminen (Myrbakken & Colomo-Palacios, 2017).

SecDevOpsin toteutuminen ei ole kovin yksiselitteistä eikä suoraan voida löydetyn aineiston perusteella sanoa, että milloin organisaation käytössä oleva on DevOps ja milloin SecDevOps. Mohanin ja Othmanen (2018) mukaan SecDevOps on kuitenkin ”kovennettu” DevOps-menetelmä, joka edellyttää turvallisuuskäytäntöjen noudattamista. SecDevOpsin taustalla pätee hyvin saman tyyppiset periaatteet kuin DevOpsissa. Myrbakken ja Colomo-Palacioksen (2017) mukaan SecDevOpsin periaatteet perustuvat DevOpsiin ja CAMS-periaatteisiin, eli kulttuuriin (culture), automaatioon (automation), mittaukseen (measurement) ja jakamiseen (sharing) ja sen lisäksi turvallisuus on osa prosessia alusta asti. Tässä yhteydessä kulttuuri tarkoittaa lähes samaa asiaa kuin aiemmin kuvailtu tiimien välinen yhteistyö. SecDevOps-kulttuuri eroaa luonnollisesti DevOps-kulttuurista siten, että yhteistyössä on tasavertaisena osapuolena mukana turvallisuustiimi ja yhtä lailla sitä, että kaikki ovat vastuussa turvallisuudesta. Jakamisperiaate on nähdäkseni hyvin lähellä kulttuuria ainakin sillä perusteella, että jakamisperiaatteeseen kuuluu tiedonvaihto tiimien välillä. Turvallisen DevOpsin kannalta oleellista on vielä se, että tietoa vaihdetaan nimenomaan esiin tulleista haasteista ja uhkista. SecDevOpsin automaatioperiaate tarkoittaa luonnollisesti sitä, että myös turvallisuustyökalut olisi tarkoitus automatisoida niin pitkälle, kuin se vain on mahdollista. Tavoitteena automaatiossa on se, että prosessi ei hidastu, vaikka suoritetaankin tarvittavat turvatoimet.



Kuvio 3: SecDevOpsin osa-alueet

Siinä missä mittaustyökaluja käytetään DevOps-kehitykessä liiketoiminnan keskeisten tekijöiden mittaamiseen, SecDevOps edistää sellaisten mittareiden käyttöä, jotka seuraavat uhkia ja haavoittuvuuksia koko ohjelmistoprosessin ajan. Automaattiset seurantajärjestelmät mahdollistavat sen, että organisaatiot voivat tarkistaa aina reaaliajassa, kuinka hyvä sovellus on turvallisuuden kannalta (Myrbakken & Colomo-Palacios, 2017).

Devopsin laajentaminen SecDevOpsiin tuo mukanaan paljon oleellisia hyötyjä. Ensinnäkin, jos turvallisuustiimi integroidaan prosessiin heti alusta alkaen, eikä perinteisesti vain loppuun, on paljon helpompi suunnitella ja toteuttaa turvallisuustoimenpiteet siten, että se ei aiheuta viivästyksiä, tai että liikkeelle päästettyä ohjelmistoa tarvitsisi korjata jälkikäteen. Tietoturvavirhe tai -rikkomus voi käydä myös erittäin kalliiksi organisaatiolle monessakin mielessä. Erityisesti sanktiot rikkomuksista ovat tunnetusti isoja ja myöskin mainehaitta saattaa vaikuttaa oleellisesti organisaation menestymiseen.

4.3 Ratkaisujen vastaavuus haasteisiin

Tutkielmassa esitettyjen seikkojen perusteella voidaan todeta, että DevOpsin turvallisuushaasteisiin kyetään vastaamaan hyvin, mutta ei tyhjentävästi. Yleisimpänä käytäntönä tarjotaan automaatioon perustuvia ratkaisuja, minkä voidaan katsoa vastaavan ainakin osittain niihin haasteisiin, joilla julkaisuputkesta voidaan saada turvallinen. Tässä yhteydessä on kuitenkin syytä huomioida, että kaikkia toimintoja ei voida, ei ole syytä, eikä kannata automatisoida. Ei-automaattiset ratkaisut ovat tarpeellista ainakin sääntelyn ja vaatimusmäärittelyn vuoksi. Lisäksi on otettava huomioon, että työkalujen valitseminen oikein on äärimmäisen kriittistä. Automaatio vastaa myös nopeutensa puolesta siihen ongelmaan, joita esiintyy, kun tietoturvamenetelmät hidastavat prosessia.

Kokoavasti voidaan myös todeta, että tiimien välisen yhteistyön lisääntyminen on turvallisuuden kannalta hyvä asia niin kauan, kun vaihdettava tieto ei ole vaarassa joutua sellaisten osapuolten saavutettaviin, joille se ei kuulu. Samoin on tärkeää lisätä kehittäjien tietoturvaymmärrystä, vaikka he eivät varsinaisesti joutuisikaan arvioimaan itsenäisesti tuotoksensa turvallisuutta.

SecDevOps on tavoitetila, johon nähdäkseni jokaisen DevOps-organisaation tulisi pyrkiä yleisen hyödyn vuoksi, vaikka se tuottaisikin lisäkustannuksia.

5 YHTEENVETO

Tässä kirjallisuuskatsauksessa syvennyttiin DevOpsiin, sekä siihen liitettyihin tietoturvahaasteisiin, -ongelmiin ja -uhkiin. Ensimmäisessä luvussa käsiteltiin DevOpsia paradigmana, menetelmänä ja kulttuurina, sekä pohdittiin sen etuja, ominaisuuksia, hyötyjä ja haasteita. Toisessa luvussa käsiteltiin tietoturvallisuutta yleisesti sekä sitä, minkälaisia tietoturvahaasteita kohdataan DevOps-kehityksessä ja minkälaisia ongelmia saattaa seurata, jos DevOps-kehityksessä ei oteta huomioon tietoturvallisuutta. Kolmannessa luvussa tuotiin esiin muutamia eri työkaluja, menetelmiä sekä kulttuureja, jotka on kehitetty tietoturvan lisäämiseksi DevOps-ympäristössä.

DevOps on paradigma, joka on kehitetty ohjelmistotuotteiden tuomiseksi nopeammin markkinoille, mutta samalla ottamaan huomioon ohjelmiston elinkaaren ja käyttäjien kaikki tarpeet. Termi muodostuu englanninkielisistä sanoista development ja operations. Ydinajatuksena DevOpsissa on ajattelutapa, jossa ohjelmistokehitysorganisaation tiimien välistä poikkitieteellistä yhteistyötä tiivistetään, jotta voidaan käyttää kestäviä järjestelmiä ja nopeuttaa muutoksen toteutumista. DevOps on joustavaa ja nopeaa liiketoimintaprosessin kehittämistä ja sen voi kehitysmetodin lisäksi ymmärtää myös organisaatiomuutoksena tai -kulttuurina. Tärkein tavoite on yhdistää kehittämistiimi, toimintatiimi ja laadunvarmistus toisiinsa jouhevalla tavalla siten, että jokaiseen ohjelmistokehitysprosessin vaiheeseen kuluisi vähemmän aikaa.

Tietoturvan katsotaan tänä päivänä olevan oleellinen osa ohjelmistokehitystä ja sen historia ulottuu melko pitkälle. Nykypäivänä digitalisaation edetessä myös tietomurtojen mahdollisuus kasvaa. Ohjelmistokehitysalalla on tehty johtopäätös, että ainoa asia, jonka voidaan tietoturva osalta sanoa olevan varmaa, on se, että kaikki ohjelmistot ovat murrettavissa, kun vain löytyy tarpeeksi motivoitunut hyökkääjä. Tämän ei voi kuitenkaan olla este tietoturvan kehittämiselle. Ohjelmistoturvallisuus tarkoittaa ohjelmiston kykyä kestää sen kohtaamia hyökkäyksiä. Tämän tutkielman tutkimuskysymys oli

- Minkälaisiin haasteisiin DevOps-järjestelmäkehitysmetodia sovellettaessa yleisesti törmätään, ja millaisia työkaluja näiden uhkien torjumiseksi käytetään

DevOpsin nopeasta luonteesta ja innokkaasta tavasta tuottaa ohjelmistoja nopeasti markkinoille, sen turvallisuus saattaa ajoittain kärsiä tai jopa puuttua kokonaan. DevOps-organisaatioilla on hyvin erilaisia motivaatioita harjoittaa tietoturvaa tai sivuuttaa se kokonaan. Yleisimpiä haasteita tietoturvan saralla ovat nopean tuotannon halukkuudesta johtuva tietoturvallisuuden laiminlyönti ja DevOps-tuotantoputken turvallisuuden varmistaminen, lisääntyneiden sisäpiiriläisten määrä kehitysprosessissa, automaattisten ja ei-automaattisten ratkaisujen välillä tasapainottelu, oikein tehty turvallinen vaatimusmäärittely, kehittäjien riittävä taitotaso sekä oikeiden työkalujen löytäminen julkaisuputkessa.

Tietoturvahaasteisiin kyetään vastaamaan DevOps-ympäristössä monellakin eri tavalla. Rahman ja Williams (2019) esittelee neljä eri tietoturvakäytäntöä, jotka he kehottavat DevOps-organisaatioita omaksumaan osaksi organisaation sisäisiä käytäntöjä. Nämä käytännöt ovat automaatiotoimintojen käyttö, tiimien välinen yhteistyö, turvallisuuskoulutuksen tarjoaminen ja ei-automaattisten suojaustoimintojen käyttö. DevOps-menetelmästä, jossa turvallisuus huomioidaan kiinteäksi osaksi ohjelmistokehitysprosessia, on alettu käyttää myös nimitystä SecDevOps. SecDevOpsissa on kyse siitä, että DevOps-kehityksessä vastataan suoraan sen omiin tietoturva tarpeisiin. Toisin sanoen voidaan sanoa, että kehitys- ja tuotantotiimin yhteyteen lisätään tasavertaiseksi osapuoleksi myös turvallisuustiimi.

Tässä tutkielmassa onnistuttiin löytämään DevOpsiin liitettyjä tietoturvahaasteita sekä onnistuttiin löytämään ratkaisuja lähinnä käytäntöjen ja menetelmien tasolla. Työkalujen tarkastelu tarkemmin jouduttiin rajaamaan ulos tutkielmasta, sillä niiden analysointi olisi tehnyt tutkielmasta liian laajan. Tutkielman kirjoittamista hankaloitti kelvollisen lähdemateriaalin vähäisyys ja osin taas lähteiden samankaltaisuus, jolloin niiden vertailu on verrattain hankalaa. Yhtä kaikki, tutkielmassa onnistuttiin löytämään kattava vastaus tutkimuskysymykseen.

DevOpsin turvallisuudesta tulisi tehdä tutkimusta jatkossakin ja siinä olisi mielekästä kiinnittää huomiota uusien, luotettavien turvallisuusstandardien ja vaikkapa automaattisten työkalujen luomiseen. Huomiota voisi kiinnittää esimerkiksi nopeuteen ja kustannuksiin siitä näkökulmasta, että tietoturva ei olisi ensimmäinen uhri.

LÄHTEET

Abrahamsson, P., Botterweck, G., Ghanbari, H., Jaatun, M. G., Kettunen, P., Mikkonen, T. J., ... & Wang, X. (2020). Towards a secure devops approach for cyber-physical systems: An industrial perspective. *International Journal of Systems and Software Security and Protection (IJSSSP)*, 11(2), 38-57.

Ahmadvand, M., Pretschner, A., Ball, K., & Eyring, D. (2018, June). Integrity protection against insiders in microservice-based infrastructures: From threats to a security framework. In *Federation of International Conferences on Software Technologies: Applications and Foundations* (pp. 573-588). Springer, Cham.

Assal, H., & Chiasson, S. (2018). Security in the software development lifecycle. In *Fourteenth symposium on usable privacy and security* (SOUPS 2018) (pp. 281-296).

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices architecture enables devops: Migration to a cloud-native architecture. *Ieee Software*, 33(3), 42-52.

Bass, L., Holz, R., Rimba, P., Tran, A. B., & Zhu, L. (2015, May). Securing a deployment pipeline. In *2015 IEEE/ACM 3rd International Workshop on Release Engineering* (pp. 4-7). IEEE.

Battina, D. S. (2017). Best practices for ensuring security in DevOps: A case study approach. *International Journal of Innovations in Engineering Research and Technology*, 4(11), 38-45.

Bou Ghantous, G., & Gill, A. (2017). DevOps: Concepts, practices, tools, benefits and challenges. *PACIS2017*.

Denning, D. E., & Denning, P. J. (1979). Data security. *ACM Computing Surveys (CSUR)*, 11(3), 227-249.

Düllmann, T. F., Paule, C., & van Hoorn, A. (2018, May). Exploiting devops practices for dependable and secure continuous delivery pipelines. In *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)* (pp. 27-30). IEEE.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *Ieee Software*, 33(3), 94-100.

Ennaciri, H., Spafford, G. (2021). 3 Steps to Esure Compliance and Audit Success With Devops. *Gartner*. https://www.gartner.com/en/documents/3970276

Farroha, B. S., & Farroha, D. L. (2014, October). A framework for managing mission needs, compliance, and trust in the DevOps environment. In *2014 IEEE Military Communications Conference* (pp. 288-293). IEEE.

Ferry, N., Solberg, A., Song, H., Lavirotte, S., Tigli, J. Y., Winter, T., ... & Castelruiz Aguirre, A. (2018, March). Enact: Development, operation, and quality assurance of trustworthy smart iot systems. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment* (pp. 112-127). Springer, Cham.

IEEE Standards Committee. (2021). IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: *IEEE Standard* 2675-2021.

Katal, A., Bajoria, V., & Dahiya, S. (2019, March). DevOps: Bridging the gap between Development and Operations. In *2019 3rd International Conference on Computing Methodologies and Communication* (ICCMC) (pp. 1-7). IEEE.

Jaatun, M. G., Cruzes, D. S., & Luna, J. (2017, August). Devops for better software security in the cloud invited paper. In *Proceedings of the 12th International Conference on Availability, Reliability and Security* (pp. 1-6).

Juvonen, R. (2018). *Ohjelmistoprojektin sudenkuopat ja miten ne vältetään*. BoD-Books on Demand.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016, November). Relationship of DevOps to agile, lean and continuous deployment. In *International conference on product-focused software process improvement* (pp. 399-415). Springer, Cham.

Mansfield-Devine, S. (2018). DevOps: finding room for security. *Network Security*, 2018(7), 15-20.

Microsoft. (14.6.2021). What are Microservices? https://docs.microsoft.com/en-us/devops/deliver/what-are-microservices

Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. *Computer Science Review*, 38, 100308.

Mohan, V., & Othmane, L. B. (2016, August). Secdevops: Is it a marketing buzzword?-mapping research on security in devops. In *2016 11th international conference on availability, reliability and security (ARES)* (pp. 542-547). IEEE.

Mohan, V., ben Othmane, L., & Kres, A. (2018, September). BP: Security concerns and best practices for automation of software deployment processes: An industrial case study. In *2018 IEEE Cybersecurity Development (SecDev)* (pp. 21-28). IEEE.

Moyón, F., Soares, R., Pinto-Albuquerque, M., Mendez, D., & Beckers, K. (2020, November). Integration of security standards in devops pipelines: An industry case study. In *International Conference on Product-Focused Software Process Improvement* (pp. 434-452). Springer, Cham.

Myrbakken, H., & Colomo-Palacios, R. (2017, October). DevSecOps: a multivocal literature review. In *International Conference on Software Process Improvement and Capability Determination* (pp. 17-29). Springer, Cham.

Perera, P., Bandara, M., & Perera, I. (2016, September). Evaluating the impact of DevOps practice in Sri Lankan software development organizations. In *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)* (pp. 281-287). IEEE.

Rahman, A. A., & Williams, L. (2016, April). Security practices in DevOps. In *Proceedings of the Symposium and Bootcamp on the Science of Security* (pp. 109-111).

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L. E., Tiihonen, J., & Männistö, T. (2016, November). DevOps adoption benefits and challenges in practice: A case study. In *International conference on product-focused software process improvement* (pp. 590-597). Springer, Cham.

Virmani, M. (2015, May). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In *Fifth international conference on the innovative computing technology* (intech 2015) (pp. 78-82). IEEE.

de Vries, S. (2014). Continuous security testing in a DevOps world. *OWASP AppSec Europe*.

Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and its practices. *IEEE Software*, 33(3), 32-34.