Mikko Rämö

DevSecOps

Turvallisempaa ketterää sovelluskehitystä

Tekniikka
2019

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Mikko Rämö
Opinnäytetyön nimi	DevSecOps: Turvallisempaa ketterää sovelluskehitystä
Vuosi	2019
Kieli	suomi
Sivumäärä	26 + 3 Liitettä
Ohjaaja	Jukka Matila

Työ toteutettiin CGI Suomi Oy:lle ja tarkoitus oli tuoda esiin sovelluskehityksen hyviä tapoja ja tekniikoita, jotka edistäisivät sovelluskehityksen turvallisuutta esittelemällä DevSecOpsin teemoja sovelluskehityksestä CGI:n työntekijöille.

DevSecOps tulee sanoista Development Security Operations ja tämän aiheen tutkimisessa käydään läpi useita käsitteitä ja kolmen erillaisen aiheen yhteinäisyyttä sovelluskehityksessä. Ongelmana tutkimuksessa on mielipiteiden suuri määrä, jonka pohjalta joudun vertailemaan parhaimmat ratkaisut ja kehityksen ongelmakohdat.

Tutkimuksen lopputuloksena on parempi ymmärrys DevSecOpsista ja sen yhteneväisyyksistä muihin frameworkeihin, kuten Scaled Agile Frameworkiin eli SAFeen, Scrumiin tai Toyotan tehtaalta tulevaan Kanbaniin. Tutkimuksessa myös tarkistellaan, kuinka nämä frameworkit ovat muodostettu tasapainottamaan ja tuomaan tietynlaista toistuvuutta rakentamalla prosessi kaiken tekemisen ympärille, jotta työ olisi näkyvämpää, toistettavampaa ja helpompaa tehdä ketterämmin.

DevSecOps on periaatteellinen silta kehityksen ja operatiivisten tekijöiden välillä, jossa kohtaa enemmän kokeileva kehitys, joka on valmis tekemään runsaasti muutoksia ja operatiiviset käsitteitä rakenteiden rakentamisesta niin, että ne kestävät tulevan muutoksen. Tämän kaiken sisään rakennetaan laadukkaalla koodilla ja ”future proof” -tekniikalla varmuutta ja turvallisuutta sovelluksiin.

Avainsanat ketterä kehittäminen, DevSecOps, turvallisuus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology

ABSTRACT

Author	Mikko Rämö
Title	DevSecOps: Safer Agile Software Development
Year	2019
Language	Finnish
Pages	26 + 3 Appendices
Name of Supervisor	Jukka Matila

The work was developed to CGI Suomi Oy and purpose was to bring the best practices of software development and techniques from software development which would progress security of software by presenting DevSecOps and its different themes to developers in CGI.

DevSecOps comes from words Development Security Operations and in my research of these subjects I will process many concepts including software development in many fields and integrity of these themes in software development. The problem in research is multiple opinions where I have to do comparing between each and decide which are the best practices and which are not.

The result of this research is better understanding of DevSecOps and its similarities to other frameworks like Scaled Agile Framework which is known as SAFe, SCRUM and from factory of Toyota the Kanban. These frameworks are formed to balance and produce certain kind of routine to development by building processes to around everything we do so that the work could be more visible, frequent and more agile than before which this work presents.

DevSecOps is a principle bridge between makers of development, operations and security where they meet as a more experiencing development which is ready to make a change and operative part which wants to build structures so that they will stand coming changes from development. Inside of all this will be built by high-quality code and “future proof” certain techniques and security to our applications

Keywords Agile development, DevSecOps, safety

SISÄLLYS

KÄSITELUETTELO

DevSecOps	Tapa yhdistää ketterä sovelluskehitys ja operatiiviset tekijät tuomalla mukaan turvallisuus,
Scrum	Nopeaa, kulttuurin omaista sovelluskehitystä, joka sisältää omia rituaaleja,
CI	Continuous Integration, jolla tarkoitetaan kehitysputkea jolla varmistetaan sovelluksen eheys,
Peer review	Kun tarkistetaan manuaalisesti toisen kirjoittamaa koodia.
Jenkins	Tekniikka automatisoida testejä koodille suoraan versionhallinnasta.
Docker	Virtualisointitekniikka pyörittää sovelluksia omissa pienissä virtuaalikoneissa.
Substanssi	Tietoa asiakkaan alasta.
SAFe	Scaled Agile Framework, jolla kyetään roolittamaan projekteja niide koosta riippumatta.
Lean ajattelu	Mahdollisimman pienellä määrällä muutoksia, paljon arvoa.
AngularJs	Kuolemassa oleva javascript framework.
Product Owner	Tuoteomistaja, vastaa asiakkaalle tuotettavasta sovelluksesta tai tuotteesta.

1 JOHDANTO

1.1 Työn tausta

CGI on auttanut tämän DevSecOps ketterän kehityksen mallin purkamisessa antamalla minulle useita haastatteluita erinäisiltä alan asiantuntijoilta, jotka edustavat sovelluskehityksen eri osa-alueita. Työn sivutuotteena syntyy opinnäytetyön lisäksi CGI:n sisäiseen käyttöön niin sanottu lyhyt esitys, joka kertoo CGI:n tavan tehdä DevSecOpsia. Työssä on tarkoitus myös vertailla alalla oleviin muihin tapoihin toteuttaa DevSecOpsia projktien rakenteesta riippuen.

1.2 DevSecOps tausta

Tulevaisuudessa vaaditaan yhä ketterämpää tekemistä sovelluskehittämisen saralla. Kehitys on edennyt jo niin pitkälle, että ei riitä sovelluksien toteutus tehtävän vain parhaimmilla menetelmillä ja ketterimmillä tavoilla. Sen ympärille on kehitettävä prosesseja, joilla saadaan paras ja turvallisin mahdollinen tulos pienimmällä mahdollisella työmäärällä. /1/

Sovelluskehityksen tekemisen malli täytyy ketteröityä niin, että yritykset eivät voi enää vesiputousmallin mukaisesti ostaa kokonaisia projekteja ja olettavansa saavan parhaan mahdollisen tuotteen, vaan heidän on omistauduttava kehitykselle olemalla kehityksessä mukana. Yrityksissä kehittämisen täytyy lähteä ylimmästä johtoportaasta, joka mahdollistaisi kehitykselle vapaat kädet ja mahdollisimman nopean palautteen sovelluksen kehitykselle. /2/

1.3 GGI Suomi Oy

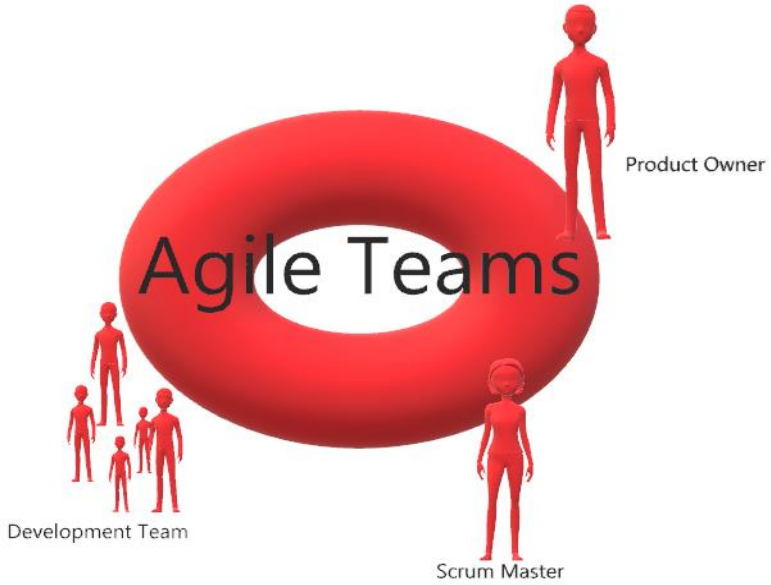
CGI Suomi Oy on kanadalainen yritys, joka tuottaa sovelluskehitystä ja konsultoinnin palveluita ympäri maailman. CGI on myös mukana monessa suuressa suomalaisessa IT- projektissa niin julkishallinnollisella kuin yksityisellä sektorilla. CGI on suuressa roolissa koko Suomen tietotekniikan alalla ja kasvattunut osuuttaan jatkuvasti. CGI:n tavoite on brändätä turvallisuutta kaikessa tekemisessään. /9/

2 PROJEKTI

2.1 Tarkoitus

CGI:llä on tarve toteuttaa projekteja yhä ketterämmin tulevaisuudessa ja tätä päämäärää edesauttamaan tarvitaan yhä ketterämpiä metodeja. CGI tarvitsee myös tietoa siitä, miten muualla toteutetaan ja mitkä ovat tämän hetken tiedon mukaan parhaimpia tapoja toteuttaa. Tässä tutkimuksessa tarkoitukseni on vertailla nykyisessä kirjallisuudessa ja verrata projekteissa DevOpsin erilaisia tapoja CGI:n tapoihin.

Tässä työssä on myös tarkoitus avata rooleja, mitkä kuuluvat nykypäivän korkean tason suunniteltuun sovelluskehitykseen ja mitä kehittäjät, operatiiviset henkilöt ja turvallisuuden parhaat tavat voivat saada korkealla tasolla aikaiseksi integroituessaan kehitysprosesiin. Kuvassa 1 esitetään nykypäiväinen kehitystiimi, jossa on esitettynä kaikki moderniin ketterään sovelluskehitykseen tarvittavat henkilöt.



Kuva 1. Moderni kehitystiimin rakenne /8/

2.2 Nykypäivän DevOps-kirjallisuus

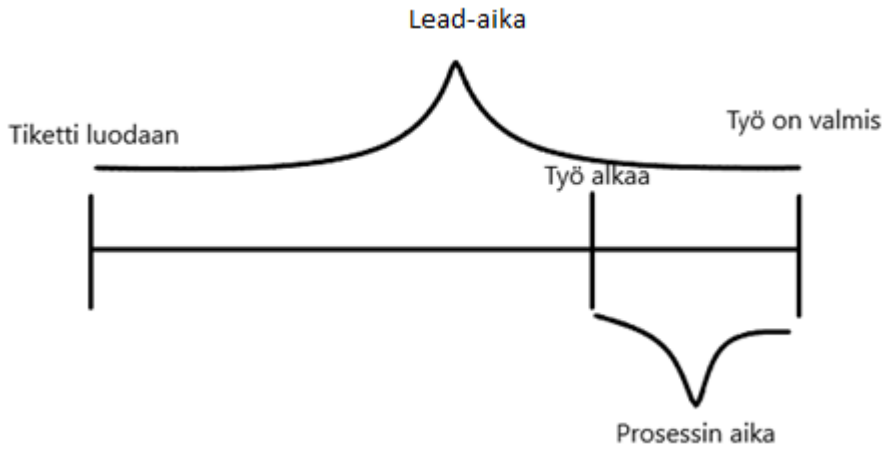
DevOpsista löytyi useita lähteitä, joissa käytettiin esimerkkinä oikeita henkilöitä, jotka kertovat IT-alan projektista, jossa tekeminen oli kaukana hallitusta ja suunnitellusta.Tämä johti siihen, että prosessit olivat kalliita ja tarvittavia resursseja ei saatu jaettua yrityksen sisällä sinne missä niitä tarvittiin eniten.

The Phoenix Project kertoo projektista missä kehitystä yritetään saada aikaiseksi liian suurilla työmäärillä, liian lyhyissä aikatauluissa, joissa projektin kehitysosastot eivät ole tietoisia operatiivisestä toiminnasta ja päinvastoin, johtaen useimmiten tulipalotilanteisiin, joita voidaan korjata mahdollisimman tehokkaasti ja nopeasti. Siitä huolimatta suuret kalliit projektit myöhästyvät. /1/

DevOps Handbook – How to create world - class agility, reliability and security määrittelee kuinka teknologian arvovirrat DevSecOpsissa tai DevOpsissa tyypillisesti määritellään tarvittavaksi prosessiksi, joka vaaditaan muuttamaan liiketalouden hypoteesit teknologian tarjoamaksi palveluksi, joka voidaan mahdollisimman nopeasti muuttaa arvoksi asiakkaalle.

Handbook tiivistää tavoiteltavaksi pitkään kestäneiden projektien tilaksi tavoitteen, jossa testaus ja operaatiot tapahtuvat samanaikaisesti suunnittelun ja kehityksen kanssa, lisäämällä nopeampaa virtausta, tarkoittaen pienempiä syklejä palautteen saamiseen tuotettavasta tuotteesta. Tämä tapa onnistuu ainoastaan silloin kun työ saadaan mahdollisimman pieniin osuuksiin ja rakentamalla laatua arvovirran keskelle kaikilla osa-alueilla. Kirjassa puhutaan tekniikasta, kuten testauspohjainen kehittäminen, jossa testaus tehdään ennen kuin yhtäkään riviä koodia on kirjoitettu. /2/

Kirjat antoivat todella paljon samanlaista viestiä epäselvyyksistä prosesseissa, joista pitäisi saada paljon läpinäkyvämpiä. Tähän DevSecOpsin tarjoama kulttuurin muutos voisi olla vastaus.



Lead-aika on se minkä asiakas kokee olevan hetki, kun pyyntö uudesta ominaisuudesta tehdään ja loppuu hetkeen kunnes työ on valmis. Prosessiaika on se missä oikeasti konkreettisesti työ tehdään sovelluskehityksessä. /2/

Kuva 2 kertoo sen kuinka ”Lead-aika” todella usein jää asiakkaalta osittain piiloon, kun kehityksen prosessit eivät ole tiedossa eikä näkyvillä.

3 KEHITYS

3.1 Ostamisen malli ja yksi tiimi

Ostamisen mallilla tarkoitetaan tiimiä generalisteja, jotka pystyvät pätevästi hallitsemaan perusasioita, joihin kuuluu esimerkiksi ohjelmointia palvelimen ja selaimen puolella. Tavoitteena on välittää asiakkaalle tiimi, joka on mahdollisimman autonominen.

Tämän mallin ideana on projekteissa tuoda substanssiosaaminen asiakkaalta ja vaatia asiakkaalta ketterämpää liiketoimintalogiikkaa. Toisin sanoen, asiakkaan on hyväksyttävä toteutuksen hallinnasta luopuminen tuotteen tekijöille ja osallistuttava intensiivisemmin osaksi kehitystä.

3.2 Ketterät menetelmät ja Lean-ajattelu

Sovelluskehityksessä on viimeisten vuosien aikana yleistynyt ketterä kehittäminen ja määrä on edelleen kasvussa. Tällaisiä ketterän kehittämisen frameworkeja ovat muun muassa Scrum, Kanban ja näiden kahden tavan erilaiset välimuodot, esimerkiksi ”Scrumban”, jota ihmiset useimmiten käyttävät. Ketterä kehittäminen on tavanomainen osa DevSecOpsia tuomalla kehitykseen läpinäkyvyyttä ja kykyä tehdä muutoksia aikaisempaan määrittelyyn mahdollisimman lyhyessä ajassa.

Ketterät menetelmät eivät enään rajoitu pelkästään kehitystiimin tarkoituksia tukemaan vaan ne tulevat tukemaan myös liiketoimintaa. Asiantuntija CGI:llä totesi: ”Yritys ei voi olla ketterä ellei se ole itse valmis olemaan ketterä”. Ongelmana on, että ketteryyttä ei pysty myymään yritykselle vaan sen täytyy lähteä yrityksen sisältä ja omaksua se osaksi liiketoiminta logiikkaa. Tähän ongelmaan on nykypäivänä joissakin yrityksissä otettu käyttöön Scaled Agile Framework eli SAFe, joka roolittaa projektit Enterprise -tasolla niin, että kehityksessä on rooli ylimmästä johtajasta aina kehittäjiin asti. /8/

Kehittämistä ja suunnittelua varten on olemassa ajatus Leanistä toteuttajana, jonka perustana on yksinkertainen idea pienemmistä julkaisemiseristä nopeammilla sykleillä niin, että saadaan tuotettua asiakkaalle mahdollisimman nopeasti arvoa.

Tätä tarvitaan osaksi esimerkiksi useiden operatiivisten, turvallisuuden ja kehityksen toimintojen automatisoimista käyttämällä muun muassa sieto- eli regressiotestejä, joilla testataan ohjelman sietokykyä rasituksen alla.

3.3 Muutoksen suunnittelu

Projekteissa ainoa varma asia on muutos, johon täytyy kyetä varaamaan aikaa, jotta voidaan ennakoida tulevat ongelmat teknologiassa, kehityksen kulttuurissa ja tavoissa.

3.3.1 Teknologiseen muutokseen

Java, Angular ja moni muu kehityskieli kehittyvät jatkuvalla tahdilla ja käytännössä teknologioiden ylläpitäminen on jatkuva haaste suurille sovelluskehitysprojekteille, koska tehtyä työtä ja koodia on hirvittäviä määriä. Konkreettisia ongelmia ovat teknologioiden versiot, joista tulee ongelmia kun niiden ylläpitäminen mahdollisesti lopetetaan ajan myötä ja nämä vanhentuneet versiot tuottavat mahdollisia tietoturvallisuusaukkoja sovellukseen.

Kehitykseen on varattava aikaa, jotta mahdollisesti vanhentuvat kirjastot kyetään korvaamaan uudemmilla, tietoturvallisemmilla ja toimivimmilla ratkaisuilla. Myös mahdollista refatktorointia ja teknistä velkaa pyritään lyhentämään suunnitellulla aikataululla ja tekemällä PoCeja eli ”Proof of Concepteja”, joilla voidaan todentaa teknologian toimivuus liiketoiminnan kannalta.

Ongelmaksi on myös muodostunut frameworkien suunnaton määrä, joka saattaa tuottaa vaikeuksia, kun yritetään mahdollisimman kattavasti vastata asiakkaan tarpeisiin ja tehdä mahdollisimman turvalllista sovelluskehitystä käyttämällä vain taatusti turvallisia kirjastoja, jotka eivät esimerkiksi lähetä tietoja kolmansille osapuolille.

3.3.2 Kulttuurin muutokseen

Paras tulos yrityksen kulttuurin muutoksesta saadaan, kun kehitys lähtee yrityksen ylätasolta. Tällä tarkoitetaan, että vanhoista isoista vesiputousmallisista batcheistä

ja kehittämisestä on päästävä irti ja siirryttävä liiketoiminnan ja projektin hallinnan johdolla lähemmäksi asiakasta. Muutosta pystytään edesauttamaan tiukentamalla yhteistyötä viemällä suunnittelu ja mahdollisesti konkreettinen tekeminen ja kehittäminen lähemmäksi asiakasta ja jos mahdollista niin asiakkaan työympäristöön. Tätä tehdään monessa suuressa IT -yrityksessä, kuten Tieto ja Accenture.

Tällaista kehittämistä tukemaan projektin johdon on suunniteltava projektit niin, että asiakkaalla on mahdollisimman matala kynnys lähteä toimintaan mukaan tuomaan omaa substanssiosaamistaan. Paras mahdollinen tulos tällaisessa tilanteessa saadaan, kun siirretään kehityksen ja liiketoiminnan tavoitteet mahdollisimman lähelle toisiaan.

SAFe-malli tukee vastaavaa aatetta tuomalla kehitystiimiin Product Ownerin, jonka tehtävänä on taata tuotteen haluttuun pisteeseen saattaminen tuomalla DevSecOps -tiimille osaamisensa ja tietoutensa liiketoiminnasta. Tämä tarkoittaa myös sitä, että liiketoiminnan tarpeet voivat radikaalisti muuttua, jos asiakkaan kanssa yhteistyö ei ole tarpeeksi tiivistä. Yhteistyö mahdollistaa kehittäjälle mahdollisuuden ketterämmin validoimaan oikeellisuuden tekemässään koodin laadussa ja sovelluksen toteutuksen lopputuloksessa. /8/

3.3.3 Leani muutos

Kehitys on mahdollistettava nopeille sykleille käyttäen jatkuvia julkaisuja projektin tyypistä riippuen, esimerkiksi 2 -3 viikkoon. Toisin sanoen, suositellaan mahdollisimman lyhyitä iteraatioita, joista nähdään kehityksen suunta, jota voidaan sitten korjata tarvittaessa.

Korjaukset tapahtuvat toistuvissa suunnittelutilaisuuksissa johdonmukaisesti sovitun ajanjakson välein.

Suunnitteluaihioissa yritetään suunnitella työmäärät mahdollisimman pieniksi eli niin kuin Lean-ajattelumalli sanoo: ”keep batch size small” ja niin, että työmäärä on toteutettavissa seuraavan suunnitellun sprintin aikana. /3/

4 TURVALLISUUS

Asiakkaan tietojen turvallinen, oikea ja asianmukainen käsittely on viime vuosina noussut monella tavalla osaksi liiketoimintamalleja ja nämä ovat myös suuressa roolissa osana DevSecOpsia.

4.1 Tunnista tarpeet ja suunnittele ympäristö

DevSecOps on kehityksen tukena, kun on tarve tunnistaa asiakkaan tarpeet turvallisuudesta niin yksityisellä kuin julkisella sektorilla. Muun muassa GDPR (General Data Protection Regulation) on tullut kaikissa käyttäjien ja asiakkaiden datan käsittelyssä selvästi esille. Sen on tarkoitus parantaa henkilötietojen suojaa ja tietosuojaoikeuksia. Myös yritykset kouluttavat työntekijöitään, jotta työntekijät ovat tietoisia tästä minimistandardista tiedon käsittelyssä. /4/

4.1.1 Varmat kirjastot

Kehittämisessä turvallisuutta pyritään arvioimaan tekemällä kartoitus käytettävistä kirjastoista, tukipalveluista, rajapinnoista ja ohjelmointikielistä. Hyvänä esimerkkinä voisi mainita AngularJs aikaisemmat versiot, joiden tuki on lopetettu ja jos selaimeen tulee päivitys, on mahdollista, että koko ohjelmointikieli lopettaa toimintansa.

Kirjastot ja muut tekniset vaatimukset ovat myös isossa roolissa ympäristön vaatimusten suunnittelun suhteen, joista projekteissa yleensä vastaa tekninen arkkitehti, jonka vastuulla on määrittää käytettävät teknologiat autentikaatiosta käyttäjien hallintaan. Arkkitehdin täytyy pitää huolta, että kaikki ymmärtävät mitä ja miten tekniikkaa pitää käyttää kehityksessä.

Usein täytyy myös pitää huolta asiakkaan mielipiteestä, jos aikoo käyttää kolmannen osapuolen kirjastoja, jotta ei tule riskejä mahdollisista yllättävistä kuluista ja pakollisista riippuvuuksista, joita lisenssit tai muut luvat ulkopuolisista ei-open source-kirjastot saattavat tuottaa.

4.1.2 Turvallinen logitus

Osana kehitysympäristön suunnittelua on tieto ohjelman toiminnoista syntyvistä logeista. Logien tuottaminen täytyy olla ohjattuna sellaiseen paikkaan, että siihen ei ulkopuolinen pääse mitenkään käsiksi. Syynä tähän on mahdollisuus paljastaa käyttäjän tietoja tai ohjelman toiminnan kannalta kriittisiä tietoja, jotka saattaisivat tuottaa haavoittuvuuksia.

Kuitenkin logitus täytyy tehdä sovittuun paikkaan, josta sovelluksen ylläpitäjät sen pystyvät löytämään aina tarvittaessa.

4.2 Kehityksen turvallisuus ja varmentaminen

Ohjelman kehityksen turvallisuus lähtee aina ohjelman kirjoittajasta eli koodarista. Kehittäjää on aika-ajoin valistettava turvallisuuden tärkeydestä ja tätä kautta annettava mahdollisuus asennoitua ja osoittaa kykynsä mahdollisen turvallisen koodin tuottamiseksi.

Äärimmäisen laadukkaan ja turvallisen koodin varmistukseen voidaan käyttää staattisia ja dynaamisia työvälineitä. Hyvänä esimerkkinä tslint, jolla voidaan dynaamisesti tarkistaa, että kirjoitettu Typescript on asetetun standardin mukainen. Staattisilla tarkkailuvälineilleä voidaan tarkistaa, että kirjoitettu koodi on pysynyt muuttumattomana.

Hyvään kehitystyöhön kuuluu tietyt tavat ja varmenteet, jotka pitää suorittaa ennen kuin koodin voi laittaa versionhallintaan. Turvallisuuden takaamiseksi koodille pitää tehdä ”Peer review”, jolla tarkoitetaan yhden tai useamma muun kehittäjän tekemää koodin lukemista ja läpikäyntiä. Tällä tavalla muut kehittäjät oppivat ja takaavat toisen kehittäjän tekemän koodin laadun.

Kehityksen tavoitteena on aina SAFen termein ”Build-in quality”, mikä tarkoittaa kaikin puolin laadukasta koodia, joka saadaan aikaiseksi hyvällä kehitysprosessilla käyttäen parhaimpia tapoja, yleensä seuraten hyväksi havaittuja Desing Guidelineseja, jotka arkkitehti määrittää. /8/

Ymmärrettävä ja helposti toistettavissa oleva koodi on turvallista ja varmaa, koska se on ymmärrettävää eli tiedetään mitä ohjelma tekee pelkästään koodin perusteella. Koodin yhdeksi tavoitteeksi voisikin sanoa, että kirjoitetaan koodia ihmisille, eikä koneille eli toisin sanoen ymmärretään aina varmasti mitä kirjoitettu koodi tekee.

Kehitys saadaan myös mahdollisimman turvalliseksi jos pystytään takaamaan ympäristö, joka vastaa mahdollisimman paljon tuotantoympäristöä. Tällaisia pystytään mahdollistamaan muun muassa virtualisaatiolla tuottamalla virtuaalinen kone, joka on täydellinen palvelin image-tuotannossa pyörivästä palvelimesta.

Muun muassa OWASP määrittää ja tunnistaa mahdollisia turvallisuusuhkia ja koostaa näistä 4 vuoden välein suurimmat web-applikaatioriskit ja tekee niistä TOP 10-listan, jolla vuonna 2017 kahtena ensimmäisenä olivat muun muassa injectiot ja rikkinäinen autentikaatio. /5/

4.3 Elinkaari

Elinkaari tarkoittaa koko kehityksen virtaa, joka projekteissa tapahtuu eli elinkaari on konkreettisesti kehityksen viemistä määrityksistä ja vaatimuksista koodaukseen. Tehdystä työstä eli koodaamisesta tuote viedään versionhallintaan, josta mahdollisimman nopesti julkaistavaksi ympäristöön, jossa voidaan automaattisesti todeta työn olevan laadukasta, määritysten mukaista ja vastata ennalta määrättyjä turvallisuusstandardeja, jotka voidaan määrittää erinäisillä työkaluilla. Sonar Qube on työkalu koodin analysoimiseen ja mahdollisten tietoturvauhkien ehkäisemiseen koodin puolesta.

4.3.1 Automaatio osana elinkaarta

DevSecOpsissa on hyvin tärkeää, että automatisoidaan projektin kiertokulusta suuri osa. Yleensä asioita, joita voidaan ja kannattaa automatisoida ovat pienet mekaaniset usein toistetut tehtävät, esimerkiksi buildit, testit ja julkaiseminen. Julkaiseminen tapahtuu yleensä joko kehitys- tai testiympäristöihin. Toistuvasti toitotetaan käytettävän Jenkins pipeline nimistä tekniikkaa, joka pystytään

configuroimaan tekemään testejä, buildaamaan ja kommunikoimaan viestien ja sähköpostien välityksellä.

Palvelinten ja sovelluksien toimintaa pitää pystyä tarkkailemaan, jotta pysytään ajan tasalla mahdollisista riskeistä ja turvallisuusongelmista. Tavoiteltavaa olisi, että nämä statistiikat ja data olisi mahdollisimman näkyvää ja niihin olisi pääsy kaikilla kehittäjillä, jotta saadaan nopeasti tieto ohjelman tai palvelimen hajoamisesta tai virhetilanteesta. Hyvänä työkaluna voisi mainita WAFin, jolla kyetään monitoroimaan HTTP-liikennettä web applikaation ja internetin välillä. /6/

Koodi on myös hyvä automaattisesti analysoida käyttämällä nykyaikaisia teknologioita, kuten Sonar Qube tai Nexus code scanner. Tällä tavoin pyritään vähentämään ihmisen peer review-virheitä turvallisen koodin tuottamisessa.

5 OPERATIIVISET TOIMINNOT

Perimmäinen tehtävä operatiivisilla toiminnoilla on ylläpitää ja tuottaa tasapainoisia ympäristöjä, jotka tukevat kehitystä sekä tehdä ja parantaa nykyisiä valmiita sovelluskehityksen tuottamia palveluita.

5.1 Rajapinnat, alustat ja julkaiseminen

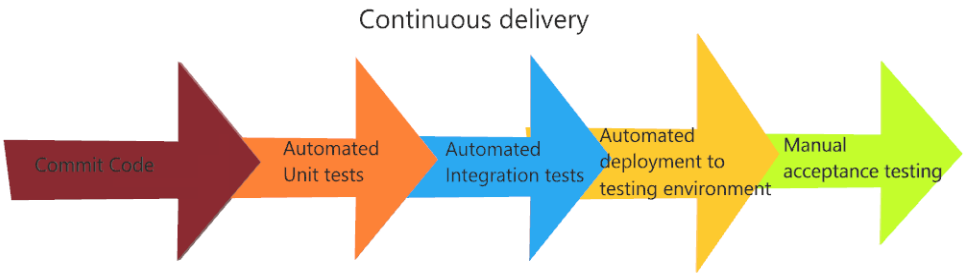
Kehitykselle operatiiviset rajapinnat, alustat ja julkaisuputki on tehtävä mahdollisimman helpoiksi. Kehittäjille ja kehitetyn koodin ylläpitäjille täytyy sallia mahdollisuudet ja oikeudet rajatuissa määrin operatiiviisiin ympäristöihin, joihin kehitetty koodi rakennetaan, jotta DevSecOps-ajattelu saisi mahdollisimman paljon tukea ja kykenisi onnistumaan.

Tällaisia ympäristöjä kyetään toteuttamaan monella eri tavalla ja tästä esimerkkinä kehittäjän koneella oleva virtuaalinen ympäristö, johon kehittäjän on mahdollisuus päästä todella matalalla kynnyksellä. Kehittäjä voi toteuttaa, testata ja ajaa koodia turvallisesti rikkomatta muita ympäristöjä.

5.1.1 Puppet

Tapoja ja tekniikoita toteuttaa rajapintoja ja alustoja on monia, mutta mainittavan arvoisina on Puppet, jolla voidaan mahdollistaa isoissa projekteissa infrasturktuureihin jatkuvan julkaisemisen ja tuotantoon jatkuvan toimituksen tuottamalla tehtäväpohjaisen konfiguroinnin haluttuihin ympäristöihin tarpeen mukaan.

Jatkuva julkaiseminen ja jatkuva toimitus tunnetaan maailmalla ”Continuous deployment” ja ”Continuous delivery”. Nämä kaksi sisältävät konfiguraatiot kaiken tarvittavan koodin julkaisemisesta mahdollisimman turvallisesti automatisoimalla esimerkiksi testit ja koodin laadun varmistuksen.



Kuva 3. Tässä esimerkki kuinka jatkuva julkaisuputki koostuu suunnitelluista vaiheista joilla taataan koodin ja tuoteen hyvä laatu.

Tarkoituksena kuitenkin on, että tämä edellä mainittu ”deployment pipelin” eli toimitusputki, olisi täysin automatisoitu, jotta sitä päästäisiin julkaisua varten hyväksymistestaamaan ja varmentamaan, että tuote on juuri tarkoituksenmukainen.

5.1.2 Jenkins

Jenkins on yksi monista työkaluista, joita DevSecOps-tiimit käyttävät tuottaakseen julkaisemisen ja automaattisen testauksen haluttuihin ympäristöihin.

Jenkins on open-source, joka tarkoittaa. että sen käyttäminen on ilmaista kehittäjälle. Tämä on tämän tekniikan yksi hyvistä puolista pitkille sovelluskehitysprojekteille, koska se antaa matalamman kynnyksen teknologian käyttöönottoon kilpailijoihinsa nähden. /7/

Tällä hetkellä tekniikoiden kirjo on yhtä laaja kuin tekijöidensä.

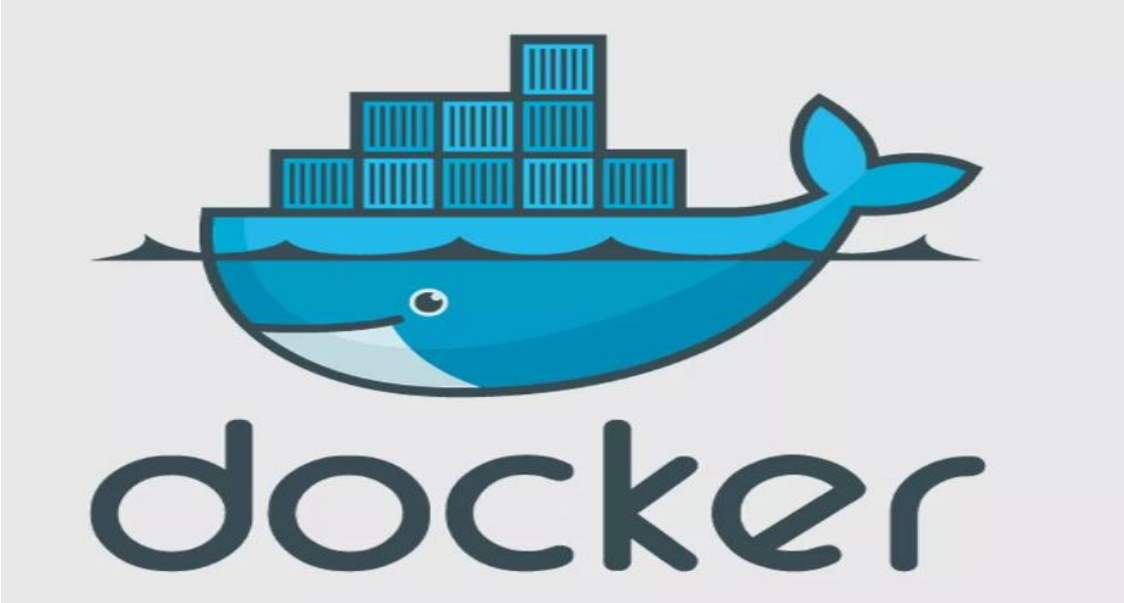
Taulukko 1. CGI:n DevOps-ohjeistus materiaalista kuvaamaan teknologioiden kirjoa, jolla voidaan toteuttaa DevSecOpsia.

DEV			PROD			
Dev	Build	Test	Deploy	Provision	Monitor	Operate
Ant	Ant	Bamboo	Ansible	Ansible	Elasticsearch	HipChat
Eclipse	AWS CodePipeline	Blazemeter	AWS CodeDeploy	AWS CodeDeploy	SPLUNK	OpsGenie
Git	Bamboo	Gatling	Bamboo	Bamboo		PagerDuty
GitHub	Concourse	Jenkins	Chef	Chef		ServiceNow
Gradle	Electric Cloud	JMeter	Cloud Foundry	Cloud Foundry		Slack
IDE	Gradle	LoadRunner	Deis	Deis		VictorOps
IntelliJ	Jenkins	MSTest	Docker	Docker		
JIRA	Maven	Selenium	Electric Cloud	Electric Cloud		
JUnit	MSBuild	SilkPerformer	Jenkins	Jenkins		
Maven	Nant	SoapUI	OpenShift	OpenShift		
MSBuild	TeamCity	TeamCity	Puppet Labs	Puppet Labs		
Nant	Visual Studio	Visual Studio	Salt	Salt		
NUnit	Visual Studio TFS	Visual Studio TFS	TeamCity	TeamCity		
Subversion			Wercker	Wercker		
TeamCity						
Visual Studio						

5.2 Ylläpito operatiivisissa toiminnoissa

Kun olemme saaneet sovelluksen tuotantoon ja jatkuva kehitys lopetetaan, niin ongelmaksi koituu kuinka voisimme ylläpitää kaiken kehittämisen vaadituissa ympäristöissä sovelluksen hengissä. Tähän vaaditaan tietynlaista konfiguraatiomahdollisuutta ympäristöille, esimerkiksi versionhallinnalla Gitillä ja käyttämällä artifaktoreita, joilla kykenee julkaisemaan valmiin sovelluksen tarvittaessa, esimerksiksi puppetilla. Nämä ovat vain muutamia esimerkkejä valtavasta tekniikan kirjosta, mutta nämä vaikuttavat olevan eniten käytössä.

Nykypäivänä puhutaan myös ylläpidon saralla sovelluksien kontittamisesta käyttäen Dockeria, joka on kevyt palvelin pelkästään sovelluksen halutulle osa-alueelle, jonka voi konfiguroida tarpeen mukaan. Erinomaisena puolena siinä on se, että se on halpa, kevyt ja nopea nostamaan halutut palvelut takaisin pystyyn. Kuvassa 5 Docker yrityksen logo, josta kuvastuu mahdollisuus kontittamisen määrälle valaan kokoisissa eli suurissa projekteissa.



Kuva 5. Docker on nousevassa suosiossa ollut jo useamman vuoden ajan ja useat yritykset suosivat sen käyttöönottoa.

5.3 Tee datasta näkyvää

Informaation kulku ja kommunikaation puutos projekteissa koetaan aina isoksi uhkaksi kehitykselle, kun ei tiedetä mitä muut tekevät, niin on mahdollisuus tehdä jotain väärin tai samaa kuin on jo tehty.

Sovelluksen virhetilanteet kehittäjälle ja käyttäjille, logitus haluttuun paikkaan, story tiketteinä tehtävien jakaminen, statistiikat testien tuloksista ja ympäristöjen ongelmista. Statistiikka on hyvä myös näkyä suoraan asiakkaalle tai taholle, jolle tuotetaan arvoa. Tätä voidaan tuottaa esimerkiksi hyväksikäyttäen sovelluskehityksen työkaluja, joilla voidaan HTML-muotoisena selaimeen saada tulostettua virhetilaraportteja.

Asiakkaalle datan näkyväksi tekemisen lisäksi läpinäkyvyyttä tuomaan käytössä ovat paperiset kanban boardit tai selaimessa Atlassianin JIRA kanban boardit ja kyseisen yrityksen muut työkalut, joilla tehdään selväksi projektin toteuttajille ja vastaaville mahdollisimman nopeasti missä tilanteessa ollaan ja tarvitseeko ruveta

priorisoimaaan uudelleen työtä. Tälläisiä ovat tilanteet, joissa on monta kehitystiimiä, jotka pyrkivät välttämään konflikteja versionhallinnassa sekä esteitä kehitykselle. Tällöin kehitys tarvitsee toiselta tiimiltä komponenttia tai ominaisuutta, jotta kykenisi jatkamaan omaa kehitystä ja on tässä tilanteessa nähtävä mahdollisimman nopeasti, koska ominaisuus tai komponentti valmistuu.

5.4 Elinkaaren palvelut

Koko elinkaaren perimmäinen tarkoitus on ottaa kiinni tuotteen vahingolliset toiminnot ja löytää tuotannolle mahdolliset uhkat ennen kuin ne tapahtuvat kaikissa mahdollisissa kehityksen vaiheissa, joissa voidaan käyttää esimerkiksi logituksia, automaatiota ja muita validaatioita.

5.4.1 Toiminnalliset ja ei-toiminnalliset määritykset

Sovelluskehityksessä pyritään aina täyttämään asiakkaan vaatimukset, jotka täytyy selvittää aina ennen kuin edes ajatellaan ryhtyä toteuttamaan sovellusta paitsi, kun asiakkaalle toteutetan esimerkkitapaus eli POC:kki, jolla pyritään todistamaan konsepti.

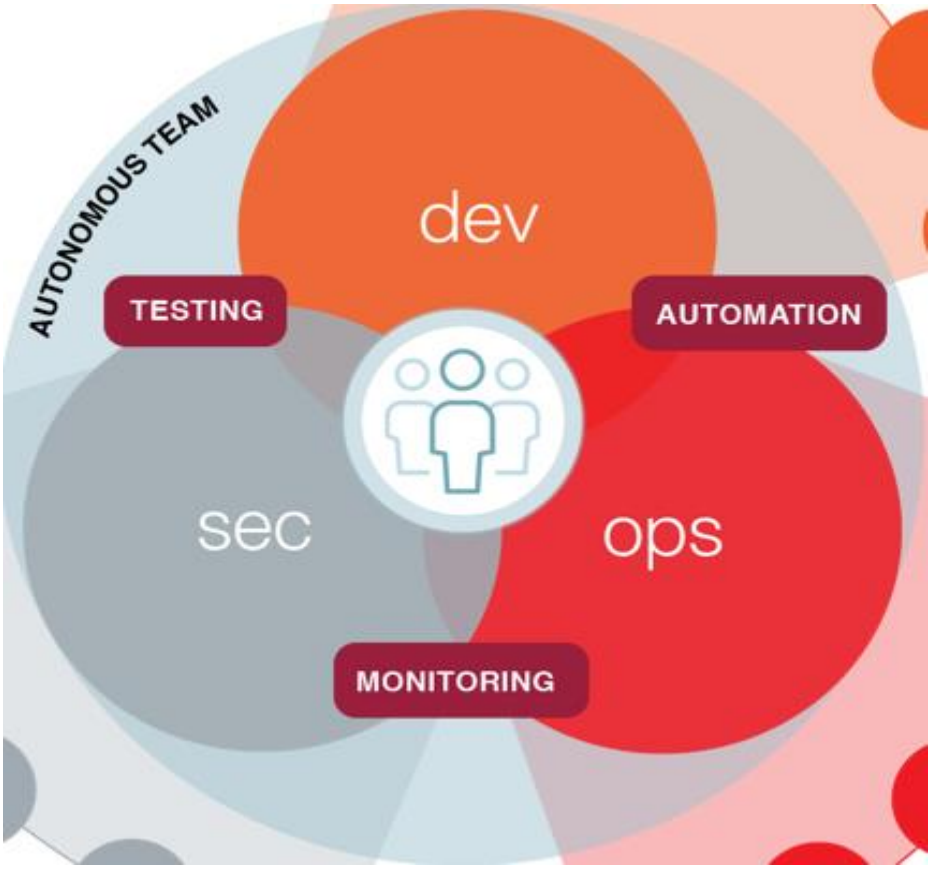
Yleensä aivan sovelluksen elinkaaren alussa, kun sovellusta lähdetään kehittämään ja yritetään suurin piirtein määrittää toiminnalliset ja ei -toiminnalliset vaatimukset. Nämä vaatimukset joista toiminnalliset määritykset määrittävät mitä sovelluksen pitää tehdä auttavat kehittäjään ymmärtämään mikä on sen liiketoimintalogiikka, joka vaaditaan asiakkaalle.

Toiminnallisten vaatimuksien jälkeen on ei-toiminnalliset vaatimukset, jotka sisältävät tekniikat, kirjastot, validaatiot ja autentikaatiot eli toisin sanoen kaiken, jotta sovellus saataisiin toimimaan mahdollisimman turvallisesti ja tehokkaasti. Ei-toiminnallisilla vaatimuksilla ei ole mitään tekemistä sen kanssa kuinka sovelluksen tulisi toimia tai tuoda arvoa asiakkaan liiketoimintalogiikkaan. Nämä vaatimukset tulevat kehityksen aikana muuttumaan, joten niitä täytyy olla mahdollista ajan myötä muokata.

Sovelluksen elinkaaren kehitykseen pitää myös suunnitella versionhallinnan turvalliset varmuuskopiot, jotta mitään ei menetetä, jos tapahtuu jotain odottamatonta, kuten ympäristökatastrofi, joka tuhoaa kaikki verkkopalvelimet koko kaupungista tai Suomesta.

6 DEVSECOPS YHTEISET TEEMAT

DevSecOps syvin olemus on tiimin koostumus. Tiimin täytyy kyetä suoriutumaan kaikilla sovelluskehityksen osa-alueilla ihan niin kuin SAFe-mallissa, jossa tiimit koostuuvat PO:sta, Scrum masterista ja kehittäjistä. Määrittelijöille ja erikoistuneille testaajille ei ole nimetty erikseen paikkaa kehityksessä vaan tiimin täytyy kyetä tuottamaan autonomisesti kehitystä palveluna.



Kuva 2. DevSecOpsin yhteiset teemat.

Kehitystä, turvallisuutta ja operatiivisiä toimintoja tukevat ja mahdollistavat kehityksen yleinen kattava testaus eli toisin sanoen varmistus siitä, että on mahdollisimman selvät liiketoiminnan tarpeet, hyvät työkalut tuottaa analytiikkaa työn laadusta, esimerkiksi ”Burb” -web applikaatio scannerilla ja integraatiotestaus tarkistamaan, että data entityt tuottavat juuri haluttua dataa. DevSecOpsissa tavoitteena on, että kaikilla osa-alueilla on kyettävä automatisoimaan mahdollisimman paljon pientä työtä.

7 PROJEKTIN LÄHESTYMISTAPA JA TOTEUTUS

Tämä työ pohjautuu omiin kokemuksiin, kirjallisuuteen ja työelämässä CGI:n tarjoamiin koulutuksiin SAFe frameworkistä.

Toteutuksen lähestymistapa oli aluksi lähteä CGI:llä kyselemään alan asiantuntijoilta ajatuksia, mielipiteitä ja tapoja toteuttaa DevSecOpsia. Tämän jälkeen keräsin kokemusta lukemalla DevOpsiin liittyvää kirjallisuutta, joka tuotti kaikenlaisia filosofisia ja vähemmän konkreettisia ajatuksia tehdä ja toteuttaa työtä, jonka mukana tuli erilaisia tapoja luoda kulttuuria projektien tiimeihin.

Työn kirjoittaminen tuki mainiosti myös tekemistäni scrum masterina ja samalla valisti nykypäivän hyvistä tavoista toteuttaa sovelluskehitystä.

Opinnäytetyöni aikana tutustuin moniin erilaisiin projektirakenteisiin, joissa DevSecOps saattoi näkyä sisäänleivottuna erillisenä tiiminä, joka ylläpiti kehityksen ja operatiivisten järjestelmien erinäisiä palveluita. Tutustuin myös ympäristöön, jossa toteutus tehtiin koodarin osaamista vaativana tekemisenä, jossa kehittäjä piti myös huolta kehitysympäristön ylläpidosta kehityksen ohessa, jossa oma toteutus oli mennyt vahingoittamaan operatiivisia toimintoja.

Jokaisesta työskentelemässäni projektissa turvallisuus oli aina korkeassa prioriteetissa. Turvallisuuden ylläpitämiseksi yritettiin aina ylläpitää projektin ihmisten tietous tekemisen tilanteesta, työn konkretiasta eli varmistetaan, että tekijä tietää mitä pitää tehdä ja automatisoimalla aina mahdolliset pienet tehtävät.

8 JOHTOPÄÄTÖKSET JA POHDINTA

Tämä työ toteutettiin CGI Suomi Oy:lle työskentelyn ohessa työstämällä ja tekemällä huomioita projektityöstä ja nykypäivän kirjallisuudesta, jotka tukevat ajatuksia prosesseista, jotka tulevat tukemaan turvallisuutta, varmuutta ja tulevaisuuden sovelluskehitystä tavoitteena parhaat tavat ja keinot tehdä sovelluksia erilaisilla ohjelmointikielillä.

Liiketoiminta ja kehitys täytyy saada mahdollisimman lähelle toisiaan tekemällä datasta ja tiedonsiirrosta mahdollisimman näkyvää. Työ on pyrittävä pilkkomaan niin pieneksi ja konkreettiseksi, jotta uuden toiminnallisuuden toimittamiseen menee ainoastaan korkeintaan päivä.

Näiden lisäksi suuri osa työstä kuluu myös siihen, että saadaan asiakas ja kehittäjä puhumaan ns. ”samaa kieltä”. Välttämättä liiketoiminnan ja tekniikan slangisanat eivät suoraan osu yksi yhteen, joten täytyy kehittää hieman yhteistä kieltä ja kartoitetaan käsitteistöä näiden kahden ryhmän välille, jotta kyetään mahdollisimman tehokkaasti saamaan palautetta kehitykseen ja kehitykseltä liiketoiminnalle. Kirjoissa ja työmaalla on eritoten painotettu lisäämään kommunikaatiota, jotta epäselvyydet ja epäkohdat saadaan esille asiakkaan kanssa ja kehitystiimin sisällä. Tähän on syynä se, että aina ei välttämättä kaikkia asioita, jotka pitää tarkistaa, ole automatisoitu.

DevSecOps ei ole sovelluskehityksessä välttämättömyys vaan se on yksi tapa ja kulttuuri tehdä sovelluksia. DevSecOpsilla on paljon yhteneväisyyksiä enterprisetason Scaled Agile Frameworkkiin eli SAFeen. Tällaiset DecSecOpsin kaltaiset kehityskehykset yleensä täydentävän puutteita projekteissa tuomalla mukanaan ideologian turvallisuudesta ja automaatiosta SAFe:n kaltaiseen frameworkkiin, jolla yritetään roolittaa projekteja.

Lopuksi tämän opinnäytetyön päätteeksi voisin todeta, että sovelluskehitys ei ole enää tiivistynyt frontin ja backendin osaamiseen, jotka ainoastaan ymmärtävät jotain nykypäväisistä uudesta tekniikasta kuten Angular 7, Spring Boot ja Java ,vaan on ruvettava kouluttamaan ja vastuullistamaan yleisempiä generalisteja

vastaamaan projektien tarpeisiin tuottamalla projekteja, joissa on osaamista kaikilta osa-aluilta, mutta vastuuta jaettava niin, että jokainen kehittäjä joutuu olemaan osana automaation, sovelluksen ja operatiivisten tarpeiden kehitystä. Tiimeihin on luotava kulttuuri, jossa vastuuta ei kaihdeta vaan se otetaan mieluummin ilolla vastaan ja niin, että vastuuta pilkotaan mahdollisimman paljon, jotta se olisi helpompi omaksua, ymmärtää ja toteuttaa.

9 TYÖN KONKREETTINEN TULOS

Alkuperäinen tavoitteeni tässä tutkimustyössä oli saada valmiiksi CGI:n sisäiseen käyttöön materiaali DevSecOpsin tekemisen parhaimmista tavoista työssä olevien erinäisten teemojen kautta ja niille asetettavia tietynlaisia haasteita. Mielestäni kuitenkin työ laajeni DevSecOpsin ulkopuolelle. Syynä ovat aiheet, jotka ovat myös isossa roolissa useissa muissa prosessia kuvaavissa frameworkeissä, kuten SAFe, Scrum, DevOps tai Kanban, joka toi tietynlaista abstraktiota työn tekemiseen ja vaikeutti itse DevSecOpsiin opinnäytetyön tiivistämisen.

Toisaalta DevSecOps on antanut tämän opinnäytetyön kirjoittajalle suuria määriä yleissivistystä työn prosesseista, sovelluskehityksestä, johtamisesta ja sovelluksen rakenteista. Lisääntyneen sivistyksen kautta olen pystynyt tuottamaan arvoa niin asiakkaalle kuin CGI Suomi Oy:lle täyttämällä minuun kohdistuneet vaatimukset erinäisissä projekteissa.

Projekteissa asemani on tavanomaisesti ollut suurimman osan ajasta sovelluskehittäjä tai Scrum master, koska tunnen, että prosesseissa lipsutaan yllättävän useinkin pois, mikä johtaa ajan käyttämisen asiaan kuullumattomiin asioihin ja kuluttaa tehokasta työaikaa, syö tehtyä työtä ja maksaa satoja tai jopa tuhansia euroja niin CGI Suomelle kuin asiakkaalle. Tämän työn ansiosta koen, että olen pystynyt ottamaan suurempaa roolia ja pystynyt sopeutumaan aina uusiin tilanteisiin ja haasteisiin.

Työni vaikutukset näkyvät parhaiten sovituissa tapaamisissa, joissa nivoutuu yhteen kehitys, turvallisuus ja operatiivisten toimintojen tekninen suunnittelu asiakaan rajapinnassa. Tavoitteena on aina pyrkiä toteuttamaan sovelluskehitystä parhampia tapoja käyttäen.

LÄHTEET

/1/ Kim, G., Humble, J. & Debols, P. 7.10.2015. The DevOps Handbook How to Create World-Class Agility, Reliability & Security in technology organizations Viitattu: 5.7.2018

/2/ Kim, G., Behr, K. & George Spafford. 2013. The Phoenix Project A Novel About IT, DevOps, and Helping Business Win. Viitattu: 9.8.2018

/3/ 12 Principles Behind the Agile Manifesto Viitattu: 27.1.2019

https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

/4/ Mikä on GDPR? Viitattu: 31.1.2019 https://tietosuoja.fi/gdpr

/5/ OWASP Top 10 Most Critical Web Application Security Risks Viitattu: 6.2.2019 https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

/6/ Web Application Firewall Viitattu: 6.2.2019

https://www.owasp.org/index.php/Web_Application_Firewall

/7/ Ansible vs Jenkins Viitattu 17.2.2019 https://stackshare.io/stackups/ansible-vs-jenkins

/8/ Scaled Agile Framework Viitattu 26.3.2019

https://www.scaledagileframework.com/

/9/ CGI Suomi Oy Viitattu 27.4.2019 https://www.cgi.fi/fi