Opinnäytetyö (AMK)

Tietojenkäsittely

2023

Markus Ijäs

Sovelluslokien keräys ja analysointi suuressa verkkopohjaisessa oppimisympäristössä

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tietojenkäsittely

2023 | 46 sivua

Markus Ijäs

Sovelluslokien keräys ja analysointi suuressa verkkopohjaisessa oppimisympäristössä

Verkkosovellusten toiminnan seuranta lokitietoja keräämällä ja analysoimalla on entistä tärkeämpää tietoturvan varmistamiseksi. Tämän opinnäytetyön tarkoituksena on selvittää, millaisia vaatimuksia lokitietojen keräämiseksi ja analysoimiseksi on löydettävissä, ja millaisia työkaluja vaatimusten toteuttamiseksi on tarjolla.

Tärkeimpiä yleisiä vaatimuksia sovelluslokien keräykselle ja analysoinnille ovat lokikirjausten tallennus niiden tuottajista erilliselle palvelimelle ja kirjausten riittävä kattavuus. Kattavuudella tarkoitetaan sitä, että lokikirjauksilla kyetään vastaamaan muun muassa kysymyksiin mitä, miksi ja miten jokin asia tapahtui. Lokikirjausten keräys ja tallennus lokitietojen tuottajista erilliselle palvelimelle on perusteltua helpomman analysoinnin, käytettävyyden ja jäljitettävyyden vuoksi. Myös EU:n yleinen tietosuoja-asetus asettaa usein vaatimuksia lokikirjausten keräämisen toteuttamiselle.

Tässä opinnäytetyössä vertaillaan useita lokikirjausten keräykseen, analysointiin ja visualisointiin soveltuvia työkaluja julkisten lähteiden ja saatavilla olleiden demotuotteiden perusteella. Toimeksiantajan käyttöön valittaviksi työkaluiksi suositellaan Grafana Loki, Grafana, Tempo ja Mimir -työkaluista koostuvaa LGTM-työkalupinoa. Lokitietojen keräysprojektin jatkamiseksi suositellaan näiden tuotteiden asennusta testipalvelimelle ja yhdistämistä oppimisympäristön kehitysversioon.

Asiasanat:

Lokitiedosto, monitorointi, verkkopalvelu, vaatimusmäärittely.

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Business Information Technology

2023 | 46 pages

Markus Ijäs

Gathering and Analyzing Application Logs in a Large Web-based Learning Platform

Countless cyber security threats constantly increase the importance of monitoring the log data web applications produce. The purpose of this thesis was to identify the requirements for collecting and analyzing log data from the client’s web learning platform, research the available tools for log data collection and analysis, and providing recommendations to the client.

Two of the more common requirements for gathering and analyzing log data are the ability to store the logs in a separate log collection server and the logs collected being sufficiently comprehensive. Analyzing these logs should enable answers to the ‘what’, ‘why’ and ‘how’ of any incident. Storing the logs to a separate server allows easier analysis, better usability, and traceability of the logs. The EU’s General Data Protection Regulation might also set requirements for monitoring activities.

This thesis compared multiple tools and services for collecting, monitoring, and analyzing the log data based on several public sources and available demo versions of the tools. The tools were compared on whether and how they meet each of the requirements gathered. The LGTM software stack consisting of the Grafana Loki, Grafana, Tempo and Mimir tools was recommended for the client. The client was advised to install these tools on a development server and integrate their web learning platform with these systems.

Keywords:

Log files, monitoring, online services, requirements specification.

Sisältö

Liitteet

Kuvat

Taulukot

Käytetyt lyhenteet

GDPR	EU:n yleinen tietosuoja-asetus (*General Data Protection Regulation*) (EU:n yleinen tietosuoja-asetus 2016/679.)
HTTP	Hypertekstin siirtoprotokolla (*Hypertext Transfer Protocol*) (MDN Contributors, 2023a)
IETF	Internet-standardeja kehittävä organisaatio (*Internet Engineering Task Force*) (Internet Engineering Task Force, ei pvm.)
ISO	Kansainvälinen standardointijärjestö (*International Organization for Standardization*) (International Organization for Standardization, ei pvm.)
LGTM	Grafana Labsin tarjoama työkalukokoelma, joka koostuu työkaluista Grafana Loki, Grafana, Tempo ja Mimir (Bonadies, 2023)
NoSQL	Ei-relationaalinen tietokanta (*Not only SQL*) (Microsoft, ei pvm.)
SNMP	Yksinkertainen verkonhallintaprotokolla (*Simple Network Management Protocol*) (Case ym., 1988)
SQL	Rakenteinen kyselykieli relaatiotietokantoihin (*Structured Query Language*) (Laine, 2002)
TCP	Tavujen kuljetusprotokolla tietoverkoissa (*Transmission Control Protocol*) (Tarkoma, 2010)
TLS	Internet-sovellusten tietoliikenteen suojausprotokolla (*Transport Layer Security*) (MDN Contributors, 2023b)
UDP	Sanomien kuljetusprotokolla tietoverkoissa (*User Datagram Protocol*) (Tarkoma, 2010)

1 Johdanto

Verkkosovellusten lokitietojen kerääminen on entistä tärkeämpää, sillä verkkohyökkäykset kohdistetaan usein yksittäisiin verkkosovelluksiin palvelinten sijaan. Verkkosovellusten tietoturvan varmentamista ja uhkiin reagointia vaikeuttavat usein sovelluslokien puutteellisuus tai epäyhtenäinen muotoilu, ja kerättyjen lokitietojen suuntautuminen sovelluskehityksen virheenhallintaan. Lokitietojen keräämisen ja analysoinnin tärkeyttä osoittaa myös lokitietojen nouseminen OWASP Top 10:2021 -listan sijalle yhdeksän. (Chuvakin & Peterson, 2010; OWASP Top 10, 2021.)

Turun yliopiston oppimisanalytiikan tutkimusinstituutin kehittämä Java-pohjainen ViLLE-verkko-oppimisympäristö (myöhemmin *toimeksiantajan oppimisympäristö*) tuottaa suuren määrän lokitietoja. Tuotettujen lokitietojen kerääminen ja analysointi koetaan tärkeäksi luvun alussa mainittujen syiden vuoksi, oppimisympäristön parantamiseksi ja sovelluskehitystoiminnan kehittämiseksi. Lisäksi oppimisympäristössä suoritettava tieteellinen tutkimus hyödyntää lokitietojen keräystä ja analysointia. Tämän opinnäytetyön tavoitteena on selvittää vaatimuksia toimeksiantajan kehittämän oppimisympäristön tuottaman lokitiedon keräämiselle, kerätyn tiedon analysoinnille ja visualisoinnille ja mitä työkaluja näiden vaatimusten toteuttamiseksi on tarjolla.

Opinnäytetyön aluksi selvitetään, mitä tarkoitetaan vaatimusmäärittelyllä, lokitiedoilla ja lokitietojen keräämisellä. Toimeksiantajan oppimisympäristön lokitietojen keräämiselle luodaan vaatimusmäärittely, minkä avulla kartoitetaan soveltuviksi katsottavat lokitietojen keräyksen, analysoinnin ja visualisoinnin työkalut. Lopuksi esitellään päätelmät ja suositukset kerättyjen vaatimusten toteuttamiseksi käytettävistä työkaluista.

2 Tavoitteiden ja vaatimusten määrittely

2.1 Mitä tarkoitetaan vaatimuksilla ja niiden määrittelyllä?

Jokaiseen projektiin kohdistuu vaatimuksia, oletuksia ja toiveita, joilla pyritään määrittelemään projektin lopputulosta. Vaatimuksia, oletuksia ja toiveita kohdistui niin ikään toimeksiantajan oppimisympäristön lokitietojen keräämiseen, ja niitä selvitetään tässä opinnäytetyössä tavoitteellisesti. Vaatimusten määrittelyn tärkeydestä puhuvat Dick ja muut (2017, s. 2) todeten, että vaatimukset antavat järjestelmäkehitykselle suunnan ja toimivat kehitettävän järjestelmän hyväksymiskriteereinä. Kyberturvallisuuskeskus (Traficom, 2019) taas kehottaa luomaan lokienhallintapolitiikan lokitietojen keräämisen ja hallinnan yhteydessä.

Vaatimuksella tarkoitetaan Youngin (2003, ss. 1–2) mukaan järjestelmään liittyvää tarpeellista määrettä, joka määrittelee käyttäjälle arvoa tuottavaa järjestelmän ominaisuutta, piirrettä tai laadullista tekijää. Vastaavasti Dick ja muut (2017, s. 2) kuvaavat vaatimusten määrittelevän järjestelmän käyttäjien, kehittäjien ja muiden osallisten tarpeita järjestelmälle ja sen toiminnalle. Sommerville (2016, s. 102) puolestaan määrittelee vaatimuksen olevan joko käyttäjävaatimus (*user requirement*), eli abstrakti korkean tason kuvaus siitä palvelusta, jonka järjestelmä tarjoaa, tai toisessa ääripäässään järjestelmävaatimus (*system requirement*) eli tarkka, muodollinen kuvaus järjestelmän toiminnasta.

Tässä opinnäytetyössä vaatimuksella tarkoitetaan mitä tahansa järjestelmälle määriteltyä tarvetta, piirrettä tai laatutekijää, jonka perusteella järjestelmää kehitetään ja arvioidaan.

Vaatimusten keräämisellä, määrittelyllä ja ylläpidolla tarkoitetaan Dickiä ja muita (2017, s. 9) mukaillen kaikkia niitä toimia, joiden avulla järjestelmään kohdistuvat vaatimukset kyetään tekemään näkyviksi, analysoimaan ja hallitsemaan koko järjestelmäprojektin ajan. Vastaavasti Sommervillen (2016, s.

102) mukaan tällä tarkoitetaan vaatimusten löytämistä, analysointia, dokumentointia ja varmentamista. Haikala ja Märijärvi (2004, s. 78) puolestaan määrittelevät vaatimusmäärittelyn olevan ”projektin tarpeellisuuden ja toteuttamiskelpoisuuden selvittämistä”, ”tavoitteiden ja vaatimusten asettamista” ja ”ratkaisumallin laatimista”. Voidaan siis todeta, että vaatimusmäärittelylle ei ole yhtä yksiselitteistä määritelmää, mutta vaatimusmäärittelyn voidaan karkeasti ottaen mieltää vastaavan etenkin kysymyksiin mitä ja miksi. Tässä opinnäytetyössä vaatimusten määrittelyksi kutsutaan järjestelmän vaatimukset määrittävää kokonaisuutta, ja vaatimusmäärittelyksi dokumentoituja vaatimuksia kokonaisuudessaan.

Vaatimusten määrittelyä on syytä jatkaa, yleisestä virheellisestä käsityksestä poiketen, koko projektin ajan. Vaatimukset voivat muuttua koko projektin ajan, kun toteutettavasta tuotteesta saadaan käyttökokemuksia tai kun esimerkiksi jokin vaatimus on jäänyt projektin alkuvaiheessa huomaamatta. On totta, että kehitysprosessin aikaisia vaatimusten muutoksia tulisi pyrkiä välttämään, mutta samalla tulee muistaa, että kaikkia vaatimuksia ei voida ymmärtää etukäteen kunnolla. Ketteriä sovelluskehityksen menetelmiä käytettäessä sanotaankin usein sarkastisesti, että vaatimukset muuttuvat niin nopeasti, että vaatimusmäärittelydokumentti on vanhentunut heti, kun se on kirjoitettu. Tämä ei kuitenkaan tarkoita, etteikö vaatimuksia kannattaisi kerätä, vaan että ne kerätään usein esimerkiksi niin kutsuttuina käyttäjätarinoina ohjelmistoprojektin aikana muodollisen vaatimusmäärittelyn luomisen sijaan. (Dick ym., 2017, ss. 11–12; Haikala & Märijärvi, 2004, ss. 92–99.)

Sen lisäksi, että huolella määriteltyjä vaatimuksia voidaan käyttää projektissa tuotetun järjestelmän toteutuksen ja testauksen perustana, voidaan vaatimusmäärittelyä käyttää kommunikaatioon projektien välillä. Hyvin määritellyt vaatimukset toimivat myös ei-teknisenä kuvauksena siitä, mitä projektissa pyritään kehittämään, ja toisaalta ne samalla saattavat olla hyvä tekninen yhteenveto projektista. (Dick ym., 2017, s. 13.)

2.2 Kuinka vaatimuksia kerätään?

Koko vaatimusmäärittelyprosessi voidaan ajatella alkavan projektin kuvauksen ja tarkoituksen määrittelystä ja päättyvän joukkoon kerättyjä vaatimuksia. Lopullisen vaatimusmäärittelyn tulisi olla yksiselitteinen kuvaus siitä, mitä järjestelmän tai sen osan on tarkoitus todellisuudessa tehdä. Vaatimusmäärittelyn tulisi yksiselitteisyyden lisäksi olla joustava ja modulaarinen, jolloin vaatimuksia voidaan tarvittaessa muuttaa ja täsmentää projektin edetessä. Lopullinen vaatimusmäärittely ei kuitenkaan tarkoita, että järjestelmän kaikki vaatimukset tulisi aina kerätä ennen sovelluskehitystyön aloittamista. Se tarkoittaa ennemminkin, että koko projektin lopputuloksena on onnistuttu havaitsemaan ja määrittelemään todelliset järjestelmään kohdistuvat vaatimukset ja toteuttamaan järjestelmä näitä todellisia vaatimuksia vastaavaksi. (Leach, 2016, ss. 90–94; Sommerville, 2016, s. 126.)

Vaatimusten keräämiseksi annetaan kirjallisuudessa erilaisia menetelmiä. Youngin (2003, s. 96) mukaan menetelmiä voivat olla esimerkiksi haastattelut, olemassa olevan dokumentaation analyysi, aivoriihet, työpajat, prototypointi, käyttötapaukset, tarinataulut (*storyboards*), aiempien toteutusten erilaiset analyysit, mallinnus ja skenaariot. Sommerville (2016, s. 115) taas esittää vaatimusten keräämiseksi kaksi pääläheStymistapaa: haastattelun, jossa keskustellaan siitä, mitä järjestelmän tulevat käyttäjät tekevät ja mitä heidän pitäisi saada aikaan sekä havainnoinnin, jossa tulevia käyttäjiä tarkkaillaan heidän työskennellessään. Sommervillen mukaan näitä kahta menetelmää tulisi käyttää yhdessä tiedon keräämiseksi, ja kerätyn tiedon pohjalta määriteltäisiin vaatimukset.

Yksittäisiä menetelmiä tärkeämpää on kuitenkin Goldsmithin (2004, ss. 46–47) mukaan yhdistää riittävä liiketoimintatietoisuus analyyttiseen ajatteluun ja objektiivisuuteen, teknisten ihmisten arvostus liiketoimintaa ja liiketoiminnan ihmisiä kohtaan sekä ymmärrys siitä, että liiketoimintavaatimukset ovat todellisia ja tärkeitä.

Vaatimusmäärittelyprosessia ei voida tarkastella huomioimatta eri ohjelmistokehityksen menetelmiä ja niiden vaikutusta vaatimusten määrittelyyn. Sommerville (2016, ss. 45–46) luokittelee ohjelmistokehityksen menetelmät korkealla tasolla vesiputousmalliin, inkrementaalisiin menetelmiin sekä määrittelyyn ja konfigurointiin. Haikala ja Märijärvi (2004, ss. 36–48) puolestaan luokittelevat ohjelmistokehityksen menetelmät vesiputousmalliin, prototyyppilähestymistapaan, inkrementaalisiin menetelmiin ja ketteriin menetelmiin.

Vesiputousmallia Sommerville (2016, ss. 47–49) kuvaa viisiportaiseksi suoraviivaisesti eteneväksi menetelmäksi, jonka askeleet ovat vaatimusten määrittely, järjestelmän suunnittelu, toteutus ja yksikkötestaus, integraatio- ja järjestelmätestaus sekä järjestelmän käyttöönotto ja ylläpito. Periaatteessa vesiputousmallissa tulisi hänen mukaansa edetä seuraavaan vaiheeseen vasta, kun edellinen vaihe on saatu valmiiksi, mikä tekee siitä niin kutsutun suunnitelmalähtöisen mallin. Edellisiin vaiheisiin palaaminen on Sommervillen mukaan tarvittaessa mahdollista, mutta tämä usein viivästyttää koko kehitysprosessia. Haikala ja Märijärvi (2004, ss. 38–41) kuvaavat vesiputousmallia hyvin pitkälti samoin, mutta lisäävät siihen tarvittaessa toteutettavan esitutkimusvaiheen määrittelyvaiheen edelle. Protoilumallilla Haikala ja Märijärvi tarkoittavat käytännössä sarjaa pitkähköjä vesiputouksia, jossa tuotteen jotain piirrettä kokeillaan ennen varsinaisen toteutuksen rakentamista. Toisin sanoen tuotteesta tai sen osasta rakennetaan ensin kevyempi prototyyppi, jonka perusteella varsinainen tuote lopulta toteutetaan.

Inkrementaalisella kehityksellä Sommerville (2016, ss. 49–50) tarkoittaa lyhyesti kuvattuna menetelmiä, joissa järjestelmästä kehitetään useita epätäydellisiä versioita lyhyemmissä kehityskierroksissa ja joissa määrittelyn, kehityksen ja validoinnin vaiheet limittyvät vesiputousmallia nopeammin eteneväksi kokonaisuudeksi. Haikala ja Märijärvi (2004, s. 45) erottavat ketterät ja inkrementaaliset menetelmät toisistaan kuvaten, että inkrementaalisissa menetelmissä tehdään muutama kehityskierros, joista jokaisessa syntyy toimiva järjestelmä. Samassa yhteydessä Haikala ja Märijärvi esittelevät kuitenkin Ivar

Jacobsonin kehittämän RUP-nimisen menetelmän (*Rational Unified Process*), jossa jokaisen inkrementin jälkeen ei välttämättä synny täysin toimivaa järjestelmää, vaan pikemminkin niin sanottu beta-versio rajatun asiakaskunnan käyttöön.

Ketterillä menetelmillä Haikala ja Märijärvi (2004, s. 47) tarkoittavat tyypillisesti hyvin lyhyistä iteraatioista koostuvia menetelmiä, joissa suunnittelun, kehityksen ja toteutuksen validoinnin voidaan ajatella olevan käytännössä jatkuvaa. Samankaltaisesti ketteriä menetelmiä kuvaa myös Sommerville (2016, ss. 73–74), jonka mukaan määrittely, suunnittelu ja kehitys limittyvät toisiinsa ja tapahtuvat lyhyissä usein toistuvissa iteraatioissa. Sommervillen (2016, ss. 49–50) mukaan ketterät menetelmät ovat osa inkrementaalisten menetelmien kokonaisuutta yhdessä suunnitelmalähtöisemmän lähestymistavan kanssa, joskin usein inkrementaalisissa ohjelmistoprojekteissa yhdistellään hänen mukaansa ketteriä menetelmiä ja suunnitelmalähtöisyyttä. Asiakaspalautetta kuunnellaan Sommervillen mukaan näiden iteraatioiden aikana jatkuvasti muuttuvien vaatimusten selvittämiseksi. Hänen mukaansa ketterissä menetelmissä suunnitteludokumentaation määrä on minimaalinen, ja vaatimuksetkin määritellään lähinnä kuvailemalla toteutettavan järjestelmän tärkeimmät ominaisuudet.

Sommervillen (2016, ss. 52–53) määrittelemänä kolmantena ohjelmistonkehitysmenetelmänä on integrointi ja konfigurointi, joilla hän tarkoittaa uudelleenkäytettävien komponenttien ja järjestelmien konfigurointia ja integrointia uuteen projektiin ja ympäristöön. Sommerville kuvaa tällaista prosessia lineaariseksi kuluksi vaatimusten määrittelystä ohjelmistokomponenttien etsimiseen ja evaluointiin, minkä perusteella vaatimuksia täsmennetään. Tämän jälkeen valitut ohjelmistot konfiguroidaan, tarvittaessa muokataan ja integroidaan käyttöön.

Kuten erilaisia ohjelmistokehitysmenetelmiä tarkasteltaessa voidaan havaita, kerätään vaatimuksia näissä joko suunnitelmalähtöisesti ja laajasti ennen toteutuksen aloitusta tai kevyemmin lyhyiden kehitysiteraatioiden sisällä. Suunnitelmalähtöisissä menetelmissä vaatimuksia kerätään usein muodollisesti,

kun taas ketterissä menetelmissä mahdollisesti hyvinkin epämuodollisesti asiakkailta saadun välittömän palautteen perusteella.

2.3 Miten vaatimukset jaotellaan ja dokumentoidaan?

Vaatimukset voidaan jakaa lähteestä ja tarpeesta riippuen usealla eri tavalla. ISO-standardi 12207:2017(E) (2017) jakaa vaatimukset karkeasti ja limittäin sidosryhmävaatimuksiin pohjautuviin järjestelmän ja ohjelmiston vaatimuksiin. Standardin mukaan sidosryhmien vaatimukset voivat olla vapaamuotoisempia tai formaaleja. Järjestelmän ja ohjelmiston vaatimukset voivat olla standardin mukaan toiminnallisia vaatimuksia, tehokkuuteen liittyviä vaatimuksia, prosessivaatimuksia, ei-toiminnallisia vaatimuksia tai rajapintojen vaatimuksia.

Youngin (2003, s. 46) mukaan vaatimukset voidaan jaotella laitteisto- ja ohjelmistovaatimuksiin, joista ohjelmistovaatimukset jaotellaan toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Youngin käyttämässä jaottelussa laitteistovaatimukset jaetaan Gradyn (2006, s. 54) esittelemän luokituksen mukaisesti suorituskykyvaatimuksiin ja rajoitteisiin. Youngin kirjassaan esittelemä jaottelu kokonaisuudessaan:

- Laitteistovaatimukset
    - Suorituskykyvaatimukset
    - Rajoitteet
        - Rajapintojen vaatimukset
        - Tekniset erityisvaatimukset
        - Ympäristövaatimukset
- Ohjelmistovaatimukset
    - Toiminnalliset vaatimukset
    - Ei-toiminnalliset vaatimukset

Hieman Youngin kanssa samoilla linjoilla on myös Sommerville (2016, ss. 105–108) esittäessään vaatimusten jakautumisen toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Hänen mukaansa toiminnallisella vaatimuksella tarkoitetaan sitä, millaista palvelua järjestelmän tulisi tarjota, miten järjestelmän tulisi reagoida

syötteisiin ja kuinka järjestelmän tulisi käyttäytyä määritellyissä tilanteissa. Toisin sanoen toiminnallinen vaatimus vastaa hänen mukaansa kysymykseen siitä, mitä järjestelmän tulisi tehdä. Ei-toiminnallisella vaatimuksella Sommerville tarkoittaa järjestelmän palveluihin tai toimintoihin asetettuja rajoitteita, kuten aikarajoitteita, ohjelmistokehitysprosessin rajoitteita ja standardien asettamia rajoitteita. Hänen mukaansa ne ovat siis sellaisia vaatimuksia, jotka eivät suoraan koske tiettyjä järjestelmän käyttäjilleen tarjoamia toimintoja.

Yksittäisen vaatimuksen dokumentointiin ei ole löydettävissä yksiselitteistä kaikkialla toimivaa mallia, vaan dokumentaation muoto ja syvyys riippuvat paitsi projektin tarpeista myös osallistujista. Vaatimuksen dokumentaatiossa tulisi Dickin ja muiden (2017, s. 95) mukaan olla mahdollisuus luokitella vaatimuksia tärkeyden, tyypin ja kiireellisyyden mukaan. Heidän mukaansa vaatimuksen määrittelyn tila, eli onko vaatimus vahvistettu, sisällöltään tyydyttävä ja hyväksytty, tulisi löytyä jokaisen vaatimuksen dokumentaatiosta. Dickin ja muiden mukaan sieltä tulisi löytyä myös esimerkiksi tehokkuus- ja testauskriteerit, perustelut ja kommentit. Lisäksi jokainen vaatimus tulisi heidän mukaansa olla yksilöllisesti tunnistettavissa.

Toisaalta taas esimerkiksi Young (2003) ja Goldsmith (2004) eivät määrittele yksittäisen vaatimuksen sisältöä näin suoraan. Etenkin Goldsmith keskittyy kuvaamaan vaatimuksia liiketoiminnan näkökulmasta, ja pitää korkean tason liiketoimintatarpeiden ymmärtämistä tärkeänä. Goldsmithin (2004, s. 41) mukaan liiketoimintavaatimusten ja järjestelmävaatimusten välinen ero on siinä, että liiketoimintavaatimukset kuvaavat **mitä** tarvitaan, kun taas järjestelmävaatimukset **miten** tarve toteutetaan.

3 Verkkopalvelun lokitietojen keräys ja analysointi

Lokitiedolla tarkoitetaan, tietojenkäsittelyn yhteydessä, aikajärjestyksessä automaattisesti kirjattua tallennetta tapahtumista ja niiden aiheuttajista. Lokiin voidaan kirjata (ts. lokittaa) esimerkiksi tietojärjestelmissä, sovelluksissa, tietoverkoissa ja tietosisällöissä tapahtuneita tapahtumia ja muutoksia. Erilaisia lokeja ovat auditointiin käytetyt lokit, tapahtuma-, virhe-, viesti-, turvallisuus- ja transaktiolokit. Esimerkkinä virhelokin kirjauksesta toimii tieto sovelluksessa tapahtuneesta virheestä, joka estää sovelluksen käytön. (Chuvakin ym., 2013, s. 24; Darrington, 2023; Traficom, 2019.)

Lokitieto voi olla Darringtonin (2023) mukaan ihmisen luettavaksi tarkoitettua tai ensisijaisesti koneellisesti luettavaksi muotoiltua. Ihmisen luettavaksi muotoiltu lokitieto on hänen mukaansa usein avattavissa tekstinkäsittelyohjelmalla, kun taas koneellisesti luettavaksi muotoillun lokin avaaminen vaatii usein siihen tarkoitetun sovelluksen. Lokikirjauksen sisältötiedon muodolle ei ole olemassa Chuvakinin ja muiden (2013, s. 28) mukaan yksiselitteistä standardia, mutta tarpeen mukaan kirjauksia voidaan muotoilla rakenteisesti tai ei-rakenteisesti esimerkiksi RFC 5424 -standardia mukaillen.

Yhtenä käyttökelpoisen oloisena esimerkkinä lokiviestin muotoilusta voidaan pitää Chuvakinin ja Petersonin (2010, s. 83) esittelemää ihmisen luettavaksi tarkoitettua muotoa. Tässä muodossa lokiviestin sisältö koostuu pilkuin toisistaan erotetuista yhtäsuuruusmerkillä ilmaistuista otsikko-arvo-pareista, joiden lukeminen koneellisesti onnistuu suhteellisen pienellä vaivalla. Esimerkkinä lokiviestin sisällöstä ja muotoilusta Chuvakin ja Peterson antavat viestin, jossa kirjauksen tehnyt järjestelmä on nimeltään ”mainserver” ja virheen syyksi ilmoitetaan salasanan oleminen väärin: ”*system=mainserver, reason=”password incorrect”*”.

Lokitiedon ollessa RFC 5424 -standardin mukaista voi siinä olla sekä rakenteisia että vapaamuotoisempia tietoja. Standardissa esitellään useita esimerkkejä rakenteista ja ei-rakenteista tietoa sisältävistä lokikirjauksista, joista yksi esitetään lainauksena alla. Standardinmukaisesti kirjaus alkaa tiedolla

tärkeydestä ja käytetystä Syslog-versiosta (<165>1) jatkuen aikaleimalla ja tiedolla viestin lähettäjästä. Kirjauksen rakenteinen osuus on sijoitettu hakasulkeisiin, minkä jälkeen esitetään lokikirjauksen vapaamuotoisempi viestiosuus (standardissa kohta 6.4 MSG).

> <165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47 [exampleSDID@32473 iut="3" eventSource="Application" eventID="1011"] BOMAn application event log entry... (Gerhards, 2009, s. 20)

Yksittäinen lokikirjaus koostuu Chuvakinin ja muiden (2013, s. 28) mukaan tyypillisesti aikaleimasta, lähdetiedosta (mistä järjestelmästä lokikirjaus on tullut) ja sisältötiedosta (lokiviestistä). Chuvakin ja Peterson (2010, s. 83) esittävät, että lokikirjaukseen tulisi merkitä tiedot siitä, mistä kirjaus on peräisin, mitä tapahtui, missä tapahtui, milloin tapahtui, miksi tapahtui ja miten tapahtui. Kyberturvallisuuskeskuksen (Traficom, 2019) mukaan lokiin kannattaisi puolestaan kirjata aikaleima, tapahtuma ja toimija, käyttöoikeus, tapahtuman lähde ja tapahtuman tila. Yleisellä tasolla voidaan todeta, että lokiin kirjattava tietosisältö riippuu lokin käyttötarkoituksesta, tietoturvan ja tietosuojan asettamista vaatimuksista ja siitä, millaista tietoa kirjattavaksi on saatavilla. Esimerkiksi tieteellisen tutkimuksen tueksi voidaan tarvita erilaista tietoa kuin tietoturvaongelmien ratkaisemiseksi.

Luokittelemalla lokitietoja esimerkiksi kriittisyyden perusteella voidaan suuresta määrästä lokitietoa saada nostettua esille tärkeitä ja nopeaa reagointia vaativia tilanteita, joita ei muuten välttämättä tietomassasta havaittaisi. Tällainen kriittisyyteen perustuva luokittelu on verkkopalvelujen osalta nähdäkseni varsin yleistä, käyttäen esimerkiksi Chuvakinin ja muiden (2013, ss. 24–25) esittelemää jaottelua informatiivisiin lokikirjauksiin, virheiden etsimiseen tarkoitettuihin kirjauksiin, varoituksiin, virheviesteihin ja hälytyksiin. Vastaavasti IETF:n RFC 5424 -standardissa ”The Syslog Protocol” (2009, s. 11) lokikirjaukset jaotellaan Chuvakinin ja muiden esittämää luokittelua yksityiskohtaisemmin, esitettynä kriittisyyden mukaan kriittisimmästä alkaen:

- Hätätilanne (emergency): järjestelmää ei voi käyttää.
- Hälytys (alert): lokikirjaus vaatii välitöntä reagointia.

- Kriittinen (critical): kriittinen tilanne.
- Virhe (error): virhetilanne.
- Varoitus (warning): varoitus (ts. ei vielä virhe, mutta vaatii huomiota).
- Huomio (notice): normaalitilanne, joka kuitenkin vaatii erityishuomiota.
- Informatiivinen (informational): lokiin kirjoitettava asia, joka ei vaadi erityishuomiota.
- Virheenetsintätieto (debug): virheiden poistamiseksi tarpeellinen tieto.

Kriittisyyden lisäksi lokikirjauksia voidaan luokitella esimerkiksi kirjauksen lähteen perusteella, kuten RFC 5424 -standardissa (Gerhards, 2009, s. 10), tai ryhmitellä jonkin muun lokikirjaukseen liittyvän tiedon perusteella. Lokitietoihin voidaan myös kirjata tapahtuman tunnistetieto (ID), jolloin eri järjestelmistä tuleva tiettyyn asiaan liittyvä lokitieto on helposti löydettävissä ja yhdistettävissä toisiin lokeihin (Chuvakin & Peterson, 2010, s. 84). RFC 5424 -standardin määrittelemää teknologiaa kutsutaan Chuvakinin ja muiden (2013, s. 52) mukaan Syslog-teknologiaksi, joksi sitä tässä opinnäytetyössäkin jatkossa kutsutaan.

Edellä käsiteltyjen tavanomaisten lokitietojen lisäksi sovelluksista ja laitteista voidaan usein kerätä mittaustietoa ja jäljitystietoja. Mittaustiedolla tarkoitetaan mitä tahansa sellaista sovelluksen ominaisuutta tai toimintaa, josta saadaan numeerisesti mitattavia tai laskettavia tuloksia (Bieman & Fenton, 2014, ss. 4–7; JavaTpoint, ei pvm.). Esimerkiksi mittaustiedosta Bieman ja Fenton antavat verkkosivuston vastausnopeuden jollakin tietyllä syötteellä. Jäljitys (*tracing*) taas tarkoittaa Kiddin (2019) mukaan sovelluksen tai järjestelmän sisäisten tietovirtojen ja toiminnan seurantaa esimerkiksi ongelmatapausten selvittämiseksi ja voi sisältää huomattavasti tavanomaista lokitietoa enemmän informaatiota.

Tässä opinnäytetyössä lokitiedolla tarkoitetaan kaikkea lokiin kirjattua ja sen avulla kerättyä tietoa mittaus- ja jäljitystiedot mukaan lukien. Lokikirjauksella tarkoitetaan yksittäistä tapahtumaa kuvaavaa tietueriviä jossakin lokissa.

3.1 Lokitietojen keräys paikallisesti

Lokitietojen keräämiseen ja tallennukseen on useita erilaisiin tarpeisiin kehitettyjä menetelmiä. Tyypillinen tapa lokitietojen keräämiselle on tallentaa ne tekstitiedostoon paikallisesti ja kierrättää näiden lokitiedostojen sisältöä arkistoimalla ja poistamalla vanhoja lokimerkintöjä. Näin lokitiedostojen koko ei kasva liian suureksi, eikä niiden käsittely muutu tarpeettoman hitaaksi. Lokitietojen kirjausta ihmisen luettaviin tiedostoihin on käytetty tyypillisesti Unixin kaltaisissa järjestelmissä, kun taas Windows-pohjaisissa järjestelmissä lokitiedot kirjataan usein Windows Event Log -lokienhallintajärjestelmään. Windows Event Log -järjestelmässä lokitiedot tallennetaan Microsoftin omassa suljetussa tiedostomuodossa. (Chuvakin ym., 2013, ss. 26–27, 85–86; Red Hat, ei pvm.)

Unixin kaltaisissa järjestelmissä lokitietoja hallinnoidaan tyypillisesti rsyslogd- tai journald-palveluilla, joista rsyslogd on nimensä mukaisesti RFC 5424 -standardiin pohjautuva lokien hallintajärjestelmä. Journald puolestaan on etenkin ongelmien ratkaisun tueksi tarkoitettu rsyslogd-perustaista tallennusta nopeampi rakenteistettu indeksoitu lokitietojen hallintajärjestelmä. (Chuvakin ym., 2013, ss. 27, 86; Red Hat, ei pvm.)

Lokikirjauksia voidaan Williamsin mukaan tehdä myös paikallisiin tietokantoihin. Tietokantaan kirjaaminen voi auttaa lokikirjausten analysoinnissa, sillä tietokannan käyttö voi mahdollistaa tietojen nopean etsimisen, kyselyjen tekemisen lokikirjauksiin ja erilaisten analysointityökalujen käytön. Esimerkiksi Syslog-palvelimen voi asettaa kirjoittamaan vastaanotetut lokikirjaukset paikalliseen tai etätietokantaan. Lokitietojen kirjaaminen etenkin relaatiotietokantaan saattaa aiheuttaa suorituskykyongelmia, kuten kirjoitusten hidastumista, minkä lisäksi relaatiotietokannoilla on usein rajoituksia siinä, paljonko tietoa niihin voi tallentaa. Tämä saattaa osoittautua ongelmaksi suurten lokitietomäärien tallennuksessa. Niin kutsutut NoSQL-tietokannat (ts. perinteisestä relaatiomallista poikkeavat tietokannat) ratkaisevat monia relaatiotietokantojen ongelmia, esimerkiksi MongoDB-tietokannan tapauksessa

tiedon lukemisen suorituskyvyn kustannuksella. Lokitietojen kirjaaminen on kuitenkin huomattavasti niiden lukemista suorituskykykriittisempää, joten NoSQL-tietokanta saattaakin olla hyvä vaihtoehto lokitietojen säilytyspaikaksi. (Chuvakin ym., 2013, ss. 89–90, 101; Microsoft, ei pvm.; Williams, 2014, s. 266.)

Osaa nykyaikaisten verkkosovellusten ohjelmakoodista suoritetaan usein palvelimella (esimerkiksi Java-ohjelmointikielellä toteutettuna), osaa käyttäjän selaimessa (esimerkiksi JavaScript-sovelluksena). Tällöin käyttäjän selaimessa tapahtuvien lokiin kirjattaviksi tarkoitettujen tapahtumien kirjaaminen suoraan palvelimen lokiin ei ole mahdollista, vaan nämä viestit tulee välittää palvelimella suoritettavalle kirjauksia vastaanottavalle sovellukselle tai sen osalle erikseen. Samaan tapahtumaan liittyviä lokitietoja voi syntyä palvelimella ja selaimessa, ja jotta tilanteesta saadaan luotua kokonaiskuva jälkikäteen, tulee lokikirjauksia voida tehdä myös selaimessa ajettavasta sovelluksesta käsin. (Choudhary & Orso, 2009, s. 303; Wang ym., 2024, ss. 2–5.)

Sovellusten sisäisiä tapahtumia on tyypillisesti kirjattu Williamsin (2014, s. 265) mukaan standarditulostevirtaan. Tämä on usein erittäin hyödyllistä ohjelmistokehityksen aikana, mutta kuten Williams toteaa, ei konsoliin kirjattuja merkintöjä usein tallenneta minnekään.

Tiedon välitykseen palvelimen ja selaimen välillä käytetään usein HTTP-pyyntöjä vastaanottavaa rajapintaa, jollainen voidaan toteuttaa myös lokikirjauksia varten. Tällöin selaimessa ajettava sovellus voi lähettää lokikirjauksia palvelimelle tarvittaessa reaaliaikaisesti HTTP-pyynnöin, tai tarvittaessa kerätä isomman määrän lokikirjauksia lähetettäväksi kerralla. Lokikirjausten lähettäminen erissä auttaa usean HTTP-pyynnön palvelimelle aiheuttaman rasituksen välttämisessä. Nimenomaan lokitietojen välitykseen käyttäjän selaimessa ajettavasta sovelluksen osasta palvelimelle ei ole yksinkertaista ja helppokäyttöistä valmista ratkaisua, vaan sellainen on usein kehitettävä kussakin projektissa itse. (Marty, 2011, ss. 182–183; Wang ym., 2024, ss. 4–5.)

3.2 Lokitietojen tallennus keskitetylle lokipalvelimelle

Lokitietojen tallentamisessa keskitetylle lokitietoja keräävälle palvelimelle (myöhemmin lokipalvelin) on useita hyötyjä, kuten Anusooya ja muut (2015, s. 2245) listaavat:

1. Lokitiedot ovat saatavilla yhdestä paikasta, eikä lokitietoja tarkastelevan ole tarpeellista kirjautua jokaiseen tietoja tuottavaan järjestelmään erikseen.
2. Lokien analysointityökaluja voidaan ajaa lokipalvelimella aiheuttamatta ylimääräistä kuormaa yksittäisille tietoja tuottaville järjestelmille.
3. Lokitietoja ei tarvitse säilyttää yksittäisillä tiedon tuottavilla palvelimilla, jolloin niiden levytilan tarve pienenee.
4. Lokitiedot ovat luettavissa, vaikka tiedon tuottava palvelin ei olisi käynnissä.
5. Lokitietoja on helpompi seurata ennakoiden esimerkiksi tietoturvaongelmien varalta.

Lokipalvelimen käytön hyödyllisyydestä kirjoittaa myös Rinnan (2005, s. 43) päättötyössään, etenkin lokeja analysoivien käyttäjien ajankäytön näkökulmasta. Hän myös toteaa, ettei lokien keskittäminen lokipalvelimelle yksin riitä, vaan analysointi- ja visualisointityökalujen yhdistäminen lokipalvelimeen on tärkeää.

Lokipalvelimen käyttö ei mainituista hyvistä puolista huolimatta ole vailla ongelmia. Lokitietoja saatetaan haluta kirjata todella suuria määriä, mikä vaatii runsaasti tallennus- ja tiedonkäsittelykapasiteettia (Chuvakin ym., 2013, s. 83). Lokitietojen välittämiseen lokipalvelimelle liittyy myös tietosuojan ja -turvan haasteita esimerkiksi tietojen luottamuksellisuuden ja eheyden osalta, kuten osiossa 3.3 kerrotaan.

Lokitietojen välitys lokipalvelimelle ei sekään ole aivan yksinkertaista. Lokitietojen välitykseen tietoja tuottavista järjestelmistä lokipalvelimelle on useita menetelmiä, jotka jaetaan Chuvakinin ja muiden (2013, s. 67) mukaan

niiden toimintaperiaatteen mukaisesti ”kysyntäperiaatteella” (*pull*) ja ”tarjontaperiaatteella” (*push*) toimiviksi. Kysyntämenetelmä tarkoittaa heidän mukaansa, että lokitietoja vastaanottava palvelin kysyy lokitietojen tuottajilta säännöllisesti, onko niillä uusia kirjauksia lokiin merkittäväksi. Tarjontamenetelmässä puolestaan lokitiedon tuottaja välittää kirjaukset suoraan niiden vastaanottajalle, jonka on oltava jatkuvasti valmis vastaanottamaan näitä viestejä.

Lokitietojen välitysmenetelmistä Unixin kaltaisessa ympäristössä yleisin on Chuvakinin ja muiden (2013, s. 68) mukaan käyttää Syslog-teknologiaa, jossa lokikirjaukset välitetään erityisinä Syslog-viesteinä niitä vastaan ottaville yhdelle tai useammalle palvelimelle tarvittaessa Syslog-viestinvälittäjien kautta. Syslog-viestejä on heidän mukaansa perinteisesti välitetty keskitetylle lokipalvelimelle UDP-protokollaa käyttäen, joskin TCP-protokollaa tuetaan moderneissa Syslog-toteutuksissa UDP:n lisäksi. TCP-protokollan käyttäminen mahdollistaa Chuvakinin ja muiden (2013, s. 101) mukaan TLS-suojauksen käytön lokitietojen välityksen salaamiseksi ja tietojen lähettäjän varmistamiseksi, mikä onkin usein tarpeellista esimerkiksi lakiteknisten ja tietoturvavaatimusten toteuttamiseksi. Tavanomaisesti Syslog-viestit on heidän mukaansa lähetetty selväkielisinä, eikä niiden aitoutta ole voitu helposti varmistaa.

Toinen Syslogin ohella merkittävä lokitietojen kaltaisten tapahtuma- ja tilatietojen välittäjä on SNMP. SNMP on alun perin kehitetty protokollaksi verkkolaitteiden konfigurointiin ja tilatietojen kyselyyn, ja sen ensimmäinen versio on perinteisen Syslog-viestin tapaan salaamaton UDP-protokollan yli lähetettävä merkkijono. SNMP:n myöhemmissä versioissa tietoturvaa on parannettu. SNMP:llä voidaan välittää tietoja kysyntäperiaatteella, tai erityisiä SNMP-ansoja (*traps*) käyttäen tarjontaperiaatteella, mahdollistaen näin tavanomaisempienkin lokitietojen välityksen. Vaikka SNMP onkin tärkeä osa verkkolaitteiden tila- ja lokitietojen välitystä, sen ei katsota soveltuvan verkkosovelluksen lokitietojen välitysmenetelmäksi kovinkaan hyvin, eikä SNMP:tä ole tätä varten kehitettykään. (Chuvakin ym., 2013, ss. 76–78.)

Tässä käsiteltyjen yleisempien lokitietojen välitysmenetelmien lisäksi lokitietoja voidaan välittää itse tehdyin ratkaisuin ja käyttäen ns. kolmannen osapuolen suljettuja tai avoimen lähdekoodin järjestelmiä. Esimerkkejä kolmannen osapuolen järjestelmistä ovat Logstash, Prometheus ja Apache Flume. Lokitietojen välitykseen soveltuvia järjestelmiä tarkastellaan tarkemmin osiossa viisi osana järjestelmäkokonaisuuksien vertailua.

3.3 Yksityisyyden suoja ja tietoturva

Lokitietojen pysyminen urkinnalta, muokkaus- ja poistoyrityksiltä suojassa ja vain luvan saaneiden saatavissa on tärkeää, kuten sekä Chuvakin ja muut (2013, s. 279) että Kyberturvallisuuskeskus (Traficom, 2019) osoittavat. Lokit voivat sisältää henkilötietoja, vaikkei niitä tarkoituksellisesti tallennettaisi, kuten Kyberturvallisuuskeskus osoittaa antaessaan esimerkin käyttäjän verkkolomakkeen väärään kenttään syöttämästä salasanasta ja siten sen tallentumisesta lokiin selväkielisenä. Kyberturvallisuuskeskuksen mukaan henkilötiedoiksi katsotaan esimerkiksi nimi tai sähköpostiosoite lokitietojen yhteydessä, minkä lisäksi IP-osoite voidaan joissain tapauksissa katsoa henkilötiedoksi. Lokin sisältäessä henkilötietoja muodostuu siitä henkilörekisteri, jolloin on huomioitava EU:n yleisen tietosuoja-asetuksen (ns. GDPR) asettamat velvoitteet. Kyberturvallisuuskeskus antaakin hyvän muistilistan lokitietojen keräämiselle (lista on lainattu muuttamattomana Kyberturvallisuuskeskuksen verkkosivustolta (Traficom, 2019)):

- Mieti, mikä lokituksen tarkoitus on.
- Ovatko lokiin tallennettavat tiedot tarpeellisia käyttötarkoituksen kannalta?
- Muista lainsäädännön velvoitteet erityyppisille lokeille.
- Käsittele lokitietoja ennalta määriteltyjen järjestelmien ja toimintatapojen mukaisesti.
- Kun tietoa ei enää tarvita, poista se.
- Määrittele käyttöoikeudet tietotarpeen perusteella.

- Rekisteröityjen ja ylläpidon tietosuojasta ja oikeusturvasta on huolehdittava.
- Informoi käyttäjiä riittävästi etenkin, jos kyse on teknisestä valvonnasta. Muista myös YT-menettely.

Kuten Kyberturvallisuuskeskuksen yllä olevasta listauksesta ja muustakin teoriatiedosta käy ilmi, kannattaa joidenkin tietojen kirjaamista lokiin välttää pääsääntöisesti kokonaan. Vältettävää tietoa ovat Kyberturvallisuuskeskuksen (Traficom, 2019) sekä Chuvakinin ja muiden (2013, s. 313) listaamia tietoja mukaillen esimerkiksi:

- henkilötunnukset
- EU:n yleisen tietosuoja-asetuksen tarkoittamat arkaluonteiset tiedot
- luottokorttinumerot
- salasanat ja valtuutustiedot
- järjestelmien väliset käyttöavaimet ja salasanat
- henkilöiden välisen viestiliikenteen sisältö.

Konkretiaa lainsäädännön vaatimusten ja muiden, kuten aiemmin käsittelemieni Kyberturvallisuuskeskuksen ohjeistusten, tueksi saadaan erilaisia lokijärjestelmiin kohdistuvia uhkia tarkasteltaessa. Näitä uhkia ovat esimerkiksi Chuvakinin ja muiden (2013, s. 279) mainitsemat loki- tai muihin järjestelmiin tehtyjen hyökkäysten jälkien piilottaminen ja hyökkääjän piiloutuminen sekä aiemmin mainitsemani tietojen urkinta ja muokkausyritykset. Hyökkäykset voidaan heidän mukaansa kohdistaa lokitietojen keräämiseen eri vaiheisiin aina tietojen lähteestä sen välitysmekanismeihin ja lopulta lokitietojen tallennukseen ja analysointiin asti. Tämä tekee lokitietojen suojaamisesta tärkeää jokaisessa lokitietoja välittävän ja käsittelevän ketjun vaiheessa.

Lokitietoja suojattaessa on tärkeää huomioida niiden luottamuksellisuus, eheys ja saatavuus. Luottamuksellisuudella tarkoitetaan, ettei mahdollinen hyökkääjä pääse käsiksi eri lokitietoihin. Niiden avulla tämä voi esimerkiksi saada tietoonsa sen, mitä sovelluksia palvelimella ajetaan (ts. mitä vastaan voisi kannattaa hyökätä) ja mitä lokikirjauksia tehdään (ts. miten piiloutua, tai mistä

löytää esimerkiksi pääsyavaimia ja käyttäjätunnuksia). Eheys tarkoittaa, ettei lokitietoja olla asiattomasti muokattu, lisätty tai poistettu esimerkiksi hyökkäyksen jälkien piilottamiseksi. Saatavuudella puolestaan tarkoitetaan sekä asiallisten käyttäjien pääsyä lokitietoon, että lokitietoon pääsyn estämistä esimerkiksi palvelunestohyökkäyksin. (Chuvakin ym., 2013, ss. 279–293.)

3.4 Lokitietojen analysointi ja visualisointi

Lokitietojen analysoinnilla tarkoitetaan kerätyn lokitiedon tarkastelua ja siitä päätelmien tekoa joko automaattisesti tai ihmisen tekemänä. Analysoinnin avulla voidaan muun muassa etsiä epäilyttäviä tapahtumia, saada erilaista käyttötietoa järjestelmistä, auttaa ohjelmistojen virheiden korjauksessa ja tehdä automatisoituja hälytyksiä. Ihmisen tekemää analysointia voidaan auttaa visualisoimalla kerätystä tiedosta esimerkiksi järjestelmien käyttötrendejä, ja toisaalta analysoinnin tuloksia voidaan esittää visualisoimalla niitä lokitietojen analyysijärjestelmissä. (Anusooya ym., 2015; Chuvakin ym., 2013, ss. 121–133; Wang ym., 2024.)

Ihmisen käsin tekemä lokitietojen analysointi on erittäin aikaa vievää, ja kun lokitietoa saattaa kertyä useita gigatavuja (miljoonia rivejä) suhteellisen pienestäkin järjestelmästä, osoittautuu käsin tehty lokitietojen analysointi nopeasti käytännössä mahdottomaksi. Automatisoidulla lokien analysointijärjestelmällä voidaan lokipalvelimelle tulevaa viestivirtaa tarkastella reaaliaikaisesti. Tarkkailemalla viestien sisältöä ja niiden välisiä suhteita voidaan tehdä hälytyksiä ja toimenpiteitä tarvittaessa viipymättä. Lisäksi analysointijärjestelmä auttaa löytämään vanhojakin toisiinsa liittyviä lokikirjauksia, joita muutoin jouduttaisi hakemaan käsin. (Chuvakin ym., 2013, ss. 134–145.)

Lokipalvelimelle kerättävää tietoa voidaan analysoinnin lisäksi visualisoida piirtämällä esimerkiksi kuvaajia lokimerkintöjen saapumisesta eri lähteistä, niiden kirjaamisesta suhteessa kuluvaan aikaan ja lokikirjausten välisistä suhteista. Merkittävän oloisia kirjauksia voidaan esittää analyysijärjestelmässä

ja -raporteissa eri värein tai muutoin nostaa paremmin näkyville. Visualisointien avulla saatetaan havaita sellaisia mielenkiintoisia tapahtumia, joita pelkästään tekstimuotoista lokitietoa tarkastelemalla ei niin helposti havaittaisi. (Chuvakin ym., 2013, s. 216.)

Useista eri lähteistä lokitietoa kerättäessä havaitaan usein, että nämä tiedot eivät ole yhtenäisessä muodossa tai, että jokin lähde tuottaa lokitietoja tarpeettoman paljon, tehden niiden jatkokäytöstä hankalaa. Näitä ongelmia voidaan ratkaista suodattamalla vastaan otettavaa lokitietoa ja normalisoimalla suodatuksen jälkeen säilytettäviksi valikoituneet tiedot. Normalisoinnilla tarkoitetaan lokikirjauksen muokkausta ja järjestelyä yhtenäiseen muotoon esimerkiksi järjestelemällä, muuttamalla ja poistamalla kirjauksen sisältöä. Tämä tekee lokitietojen analysoinnista ja visualisoinnista helpompaa poistaen virheellisiä tietoja, nopeuttaen tietojen hakua ja parantaen tietojen ristiin vertailua. (Chuvakin ym., 2013, ss. 150–152; Kidd, 2022.)

4 Oppimisympäristön lokitietojen keräämiselle kohdistuvat vaatimukset

Vaatimusten määrittelylle luotiin lähdemateriaaleja mukaillen toimeksiantajan sisäinen vaatimusmäärittelydokumentti, jonka avulla järjestelmää lokitietojen keräämiseksi ja analysoinniksi lähdettiin kartoittamaan. Vaatimusmäärittely pidettiin kevyenä ja helposti muutettavana koko lokitietojen keräämisen kehitysprojektin ajan jalostuvana kirjallisena dokumentaationa projektin tavoitteista ja tarpeista, jotta tiedon kulku projektin osallisten välillä olisi yksiselitteistä ja perustelut valituille ratkaisuille säilyisivät kirjallisina.

Toimeksiantajalle luodussa vaatimusmäärittelyssä esitetään lyhyesti lokien keräysprojektin kuvaus ja käytetyt numeroinnit ja luokittelut. Vaatimukset itsessään jaettiin toiminnallisiin ja ei-toiminnallisiin vaatimuksiin, sillä tarkemmalle jaolle ei nähty tarvetta. Yksittäisestä vaatimuksesta kirjattiin otsikon lisäksi tärkeys asteikolla yhdestä viiteen (viiden tarkoittaessa suurinta tärkeyttä), tila asteikolla alustava–vahvistettu–poistettu sekä vaatimuksen tarkempi kuvaus ja lisätiedot, mikäli näille oli tarvetta. Vaatimusmäärittelyä ei esitetä salassapitovelvoitteen vuoksi, mutta vastaava kuvitteelliset tiedot sisältävän dokumentin osa esitetään liitteessä 1 dokumentin rakenteen ja vaatimusten luokittelun havainnollistamiseksi.

Käsitellyn teoriatiedon pohjalta voidaan koostaa joitakin yleisiä vaatimuksia, jotka koskevat myös toimeksiantajan lokitietojen keräystä. Erillisen lokipalvelimen käyttöä lokitietojen tallennukseen voidaan pitää varsin tärkeänä sen mahdollistaessa lokitietoihin kohdistuvan pääsynhallinnan, lokitietojen säilyvyyden ja saatavuuden varmistamisen, lokitietojen tallennuksen vähäisen vaikutuksen tietojen lähteisiin ja tietojen yksinkertaisemman keskitetyn analysoinnin ja raportoinnin. Keskitettyä lokipalvelinta käyttämällä voidaan minimoida lähteisiin jäävä lokitiedon määrä tarvittaessa. Lisäksi EU:n yleisen tietosuoja-asetuksen mahdollisesti asettamat vaatimukset on nähdäkseni helpointa toteuttaa käyttäen keskitettyä lokipalvelinta. Toimeksiantajan

mahdollisesti suorittama tieteellinen tutkimus saattaa myös tarvita pääsyä lokitietoihin, jolloin keskitetyn lokipalvelimen käyttö on perusteltua.

Lokitietojen luottamuksellisuuden ja eheyden varmistamiseksi tietojen siirron osalta on tietojen siirto lähdepalvelimilta lokipalvelimelle voitava salata tarvittaessa, kuten osiossa 3.3 havaittiin. Tämän ja tietojen saatavuuden varmistamiseksi on lokipalvelimella lisäksi voitava rajoittaa sinne lähetettyjä lokiviestejä vain sallituilta lokitietojen tuottajilta tuleviin kirjauksiin. Lisäksi lokipalvelimen on toimeksiantajan tapauksessa tuettava lokitietojen kirjausta oppimisympäristön verkkoselaimessa ajettavista osista. Lokipalvelimen olisi hyvä myös tukea yleistä Syslog-menetelmää ja mahdollistaa Javan *java.util.logging*-paketin avulla tehtyjen lokikirjausten vastaanottaminen lokipalvelimelle välitettynä.

Lokitietojen analysointijärjestelmä voi teoriatiedon perusteella sijaita joko lokipalvelimella tai erillisellä palvelimella. Ohjelmiston sijaintia tärkeämpää näyttää olevan mahdollisuus automatisoida hälytyksiä ja luoda organisaatiossa tarvittavia raportteja, mitkä ovat myös toimeksiantajan oppimisympäristön lokien keräämiselle asetettavia vaatimuksia. Lisäksi lokitietojen keräys- ja analysointikokonaisuutta suunniteltaessa tulisi selvittää tarve mittaustietojen ja jäljitystietojen keräämiselle.

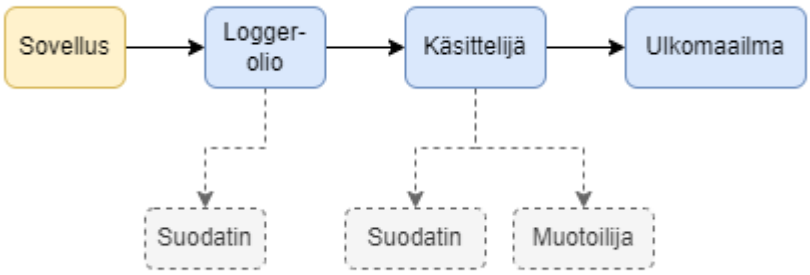
5 Lokitietojen keräys-, analysointi- ja visualisointityökalut

Opinnäytetyöni keskittyessä nimenomaisesti verkkosovelluksen lokitietojen keräämiseen ja analysointiin käsittelen tässä osiossa lähinnä tähän tarkoitukseen kehitettyjä menetelmiä. Vähemmälle huomiolle jäävät siten esimerkiksi palvelinsovellusten lokit, verkkolaitteistojen tuottamat lokit sekä käyttöjärjestelmien lokitiedot. Verkkosovellusten lokitietojen keräys- ja analysointityökaluihin tutustuttiin varsin empiirisesti tarkastelemalla työkalujen valmistajien verkkosivustoja ja tutustumalla demotuotteisiin, mikäli sellaisia oli saatavilla. Työkalujen tärkeimmät ominaisuudet on koottu osiossa 5.3 esitettyihin taulukoihin, joiden perusteella suositukset toimeksiantajalle käyttöön valittavista tuotteista tehtiin.

5.1 Tavanomaiset menetelmät

Eri ohjelmointikielet tarjoavat usein oletusmenetelmän lokitietojen välittämiseen sovellusten sisäisiltä komponenteilta eteenpäin. Näin toimii myös Java tarjotessaan *java.util.logging*-paketin (tunnetaan myös lyhenteestä *JUL*). Javan logging-paketti tarjoaa työkalut lokiviestien luomiseen, niiden muotoiluun ja välittämiseen eteenpäin standarditulostevirtaan, konsoliin, tiedostoihin, TCP-vastakkeeseen ja keskusmuistiin. Lisäksi tarjotaan mahdollisuus kolmannen osapuolen komponenttien liittämiseen lokitietoja kuuntelemaan erillisten lokinkäsittelijöiden avulla. (Oracle, 2023.)

Javan logging-paketin tarjoamassa menetelmässä lokiin kirjataan viestejä kutsumalla Javan Logger-olion tarjoamia palveluita. Logger-olio kutsuu yhtä tai useampaa lokikirjauksen käsittelijää, minkä jälkeen lokikirjaus voidaan välittää sovelluksesta ulos. Logger-olioon voidaan liittää suodattimia ja lokin käsittelijöihin suodattimia ja muotoilijoita. Esimerkinomainen kuvaus logging-paketin tarjoamasta kontrollivuosta esitetään kuvassa 1. (Oracle, 2023.)



Kuva 1. Javan logging-paketin kontrollivuo (Oracle, 2023).

Toimeksiantajan oppimisympäristön kehityksessä käytettävässä Vaadin-sovelluskehyksessä lokiviestien kirjaus on tyypillisesti tehty käyttäen Javan tavanomaista logging-pakettia, kuten Vaadin-sovelluskehyksen version kahdeksan dokumentaatiossa esitetään. Dokumentaatiossa annettu esimerkki esitetään ohjelmana 1.

Ohjelma 1. Vaadin-sovelluskehyksen dokumentaatiossa annettu esimerkki Logger-olion käytöstä (Vaadin Ltd, 2021).

```
public class MyClass {
  private final static Logger logger =
          Logger.getLogger(MyClass.class.getName());

  public void myMethod() {
    try {
      // do something that might fail
    } catch (Exception e) {
      logger.log(Level.SEVERE, "FAILED CATASTROPHICALLY!", e);
    }
  }
}
```

Vaadin-sovelluskehyksen uusimmassa versiossa (V24) käytetään edelleen samoja Javan logging-paketin tarjoamia perustyökaluja, joskin dokumentaation antamissa esimerkeissä Logger-olio haetaan tehdasluokan avulla. Ote dokumentaation esimerkistä esitetään ohjelmana 2.

Ohjelma 2. Vaadin-sovelluskehyksen dokumentaatiossa annettu esimerkki tehdasluokan käytöstä Logger-olion hakuun (Vaadin Ltd, 2023).

```
private static final Logger logger =
LoggerFactory.getLogger(CustomErrorHandler.class);
```

Javan logging-paketin käyttö Vaadin-sovelluskehyksessä mahdollistaa osaltaan Java-ohjelmistokehittäjälle tutun työkalun käytön ja yhdistämisen kolmannen osapuolen lokityökaluihin, kuten osiossa 5.2 havaitaan.

Javan logging-paketti ei ole ainoa työkalu sovelluksen sisäisten lokikirjausten tekoon, vaan sen voi korvata esimerkiksi Apache Log4j:llä tai Logback-työkalulla. Log4j:n hyviä puolia ovat Apachen (2023) omien materiaalien mukaan esimerkiksi tuki auditoinnille, asynkroninen lokien kirjaus, tuki lisäosille, yksinkertaisuus ja lambdalausekkeiden tuki. Logback puolestaan tarjoaa esimerkiksi määritystiedostojen automaattisen uudelleenlatauksen, lokitiedostojen automaattisen poiston ja pakkauksen, suodattimet ja kattavan dokumentaation (QOS.ch Sarl, ei pvm.). Esittämäni listaukset eivät ole kattavia kuvauksia näiden työkalujen eroista, vaan lähinnä havainnollistavat tarvetta huomioida Java-sovelluksessa sisäisesti käytettyjen työkalujen ominaisuudet ja ulkoinen yhdistettävyys lokityökalujen valinnan yhteydessä.

5.2 Kolmannen osapuolen työkalut

Kolmannen osapuolen työkalujen osalta on huomioitava, että osa niistä koostuu useammasta erillisestä lokitietojen keräys-, analysointi- ja visualisointityökalusta muodostaen työkalupinon, kun taas joissain työkaluissa nämä ominaisuudet on rakennettu yhteen sovellukseen. Toimeksiantajan sisäisten keskustelujen ja tekemieni verkkohakujen perusteella valittiin vertailtaviksi niin kutsutut LGTM- ja Elastic-työkalupinot sekä yksittäisistä työkaluista Prometheus, InfluxDB, Splunk, Graylog, Loggly, Sematext, Datadog ja Nagios Core. Seuraavaksi esittelen listatut tuotteet lyhyesti.

LGTM-työkalupino koostuu työkaluista Grafana Loki, Grafana, Tempo ja Mimir, jotka ovat Grafana Labsin tarjoamia avoimen lähdekoodin ilmaisia tuotteita.

Grafana Loki on nimenomaan lokitietojen keräykseen ja säilytykseen tarkoitettu Prometheus-järjestelmästä inspiraationsa saanut työkalu, johon on mahdollista välittää lokitietoja useista keräystyökaluista, kuten Promtail ja Grafana Agent -keräimistä. Grafana Loki ei indeksoi saamiensa lokitietojen tekstisisältöjä sellaisenaan, vaan käyttää indeksointiin erityisiä tunnisteita. Grafana puolestaan on esimerkiksi Loki tai Prometheus-työkaluilla kerätyn tiedon analysointi-, visualisointi-, raportointi- ja hälytystyökalu, jonka toiminnallisuutta voidaan täydentää lisäosien avulla. Grafana Tempo on vuonna 2020 julkaistu jäljitystyökalu, joka mahdollistaa esimerkiksi virhetilanteiden jäljitystietojen tuonnin ja tallennuksen muun muassa OpenTelemetry, Jaeger ja Zipkin -protokollia käyttäen ja niiden integroimisen muihin Grafana Labsin tuotteisiin. Grafana Mimir on tarkastelemistani Grafana Labsin tuotteista uusin, vuonna 2022 julkaistu etenkin Prometheus-työkaluun kerättyjen mittaustietojen pitkäaikaissäilytykseen kehitetty muuhun Grafana Labsin tuoteportfolioon integroituva työkalu. Kaikki tarkastelemani Grafana Labsin tuotteet ovat avointa lähdekoodia ja asennettavissa tarpeen mukaan yhdelle tai useammalle fyysiselle palvelimelle tai pilvipalvelinympäristöön. (Grafana Labs, ei pvm.-c, ei pvm.-d, ei pvm.-a, ei pvm.-b.)

Elastic-pino koostuu Elasticsearch, Kibana, Beats ja Logstash -tuotteista, jotka ovat Elasticsearch B.V.:n kehittämiä niin ikään ilmaisia avoimen lähdekoodin työkaluja. Toisin kuin Grafana Labsin tuotteissa, Elastic-pinossa kaikki kerätty mittaus-, jäljitys- ja lokitieto tallennetaan Elasticsearch-työkalun avulla. Elasticsearch onkin Elastic-pinon ydin, joka on tarkoitettu kerätyn loki-, mittaus- ja jäljitystiedon tallennukseen ja analysointiin. Kuten useaan muuhunkin tarkasteltuun työkaluun, Elasticsearch-työkaluun voidaan syöttää tietoja lukuisia työkaluja ja protokollia käyttäen. Elastic-pinon tuotteista Kibana puolestaan vertautuu Grafanaan tarjoten välineet kerätyn tiedon visualisointiin, analysointiin, raportointiin ja hälytyksien tekoon. Kibanan voi toisin sanoen mieltää käyttöliittymäksi Elastic-pinon päivittäiskäyttöön. Tietojen syöttämiseksi Elasticsearch-työkaluun tarjotaan Elastic-pinossa erillisinä tuotteina Beats ja Logstash, joista Beatsit ovat yksittäisiä tiettyyn tarkoitukseen luotuja tiedonvälitystyökaluja ja Logstash palvelinpuolen työkalu lokitietojen

prosessointiin, muotoiluun ja tallennusta varten eteenpäin välitykseen. Esimerkinomaisesti Filebeat-työkalulla (yksi tarjotuista Beats-työkaluista) voidaan välittää jonkin lokitiedoston sisältöä Logstash-työkaluun, jossa sitä voidaan muokata ja analysoida, minkä jälkeen se voidaan välittää Elasticsearch-työkaluun tallennusta varten. (Elasticsearch B.V., ei pvm.-d, ei pvm.-a, ei pvm.-c, ei pvm.-b.)

Yksittäisistä työkaluista Prometheus tarjoaa mahdollisuuden aikasarjatyyppisen tiedon keräämiseen ja tallennukseen. Tietojen keräämiseen käytetään, kuten tavanomaista on, erillisiä tähän tarkoitukseen luotuja kirjastoja ja työkaluja, joiden tuottama tieto tallennetaan Prometheuksen tarjoamaan aikasarjatietokantaan. Prometheus voi lähettää kerättyihin tietoihin perustuvia hälytyksiä niin kutsutun Alertmanager-työkalun avulla, ja kerättyjä tietoja voidaan visualisoida esimerkiksi Grafanan avulla. (Prometheus Authors & The Linux Foundation, ei pvm.). InfluxDB on Prometheuksen ohella toinen erityisesti aikasarjatyyppisen tiedon tallennukseen, analysointiin ja visualisointiin kehitetty työkalu, johon kerättyä tietoa voidaan jatkokäsitellä ja analysoida muilla työkaluilla (InfluxData Inc, ei pvm.). Prometheus tai InfluxDB yksin eivät nähdäkseni ole useinkaan riittäviä, vaan niiden käyttöä kannattaa harkita osana isompaa kokonaisuutta.

Splunk on Splunc Incin tarjoama valmistuote lokitietojen, mittaustietojen ja jäljitystietojen keräämiseen, tallennukseen, käsittelyyn ja analysointiin. Se tarjoaa lisäksi mahdollisuuden hälytysten luontiin ja raporttien tekoon. Splunkin ohella vastaavat perusominaisuudet (tietojen keräys, tallennus ja analysointi, hälytykset ja raportit) tarjoavia tuotteita ovat Graylog Incin Graylog, SolarWinds Worldwide LLC:n Loggly, Sematext Group Incin Sematext ja Datadog Incin Datadog. Kaikki nämä ovat käytännössä maksullisia tuotteita, vaikka tarjoavatkin eri tavoin rajoitettuja lähinnä kokeilukäyttöön soveltuvia versioitaan. (Datadog Inc, ei pvm.; Graylog Inc, ei pvm.; Sematext Group Inc, ei pvm.; SolarWinds Worldwide LLC, ei pvm.; Splunk Inc, ei pvm.)

Nagios Enterprises LLC puolestaan poikkeaa muista maksullisia tuotteita tarjoavista yrityksistä tarjoten Nagios Core -työkalun verkkoinfrastruktuurin

monitorointiin. Lisäksi he tarjoavat maksulliset Nagios XI:n monitorointiin ja hälytyksiin, Nagios Log Server -tuotteen lokitietojen käsittelyyn, Nagios Network Analyzer -tuotteen verkkoinfrastruktuurin monitorointiin ja Nagios Fusion -työkalun yleisempään IT-infrastruktuurin visualisointiin ja monitorointiin. (Nagios Enterprises LLC, ei pvm.)

5.3 Työkalujen vertailu

Tehtäessä valintaa käyttöön otettavasta lokijärjestelmästä tavanomaisen *java.util.logging*-paketin ja kolmannen osapuolen ratkaisujen välillä on Williamsin (2014, ss. 270–272) mukaan tärkeää huomioida käyttöönoton vaatima työmäärä ja sen vaatimat muutokset lokitietoja tuottaviin järjestelmiin. Lisäksi hänen mukaansa on huomioitava lokikirjausten teon mahdollisesti aiheuttama lähdejärjestelmien hidastuminen. Kuten työkaluja tarkasteltaessa havaittiin, ei Javan logging-paketin ja kolmannen osapuolen työkalujen välillä ole välttämätöntä tehdä valintaa, vaan vertailluista työkaluista usean todettiin mahdollistavan logging-paketin käsittelemän lokitiedon välityksen eteenpäin näihin järjestelmiin. Tämä osoittautui erääksi tärkeäksi toimeksiantajan vaatimukseksi.

Lokityökaluille asetettiin vertailua varten viisi perustoiminnallisuutta, joiden toteutumista tarkasteltiin ensimmäiseksi. Näitä perustoiminnallisuuksia ovat lokitietojen keräys ja tallennus, analysointi, visualisointi, hälytykset ja analysoinnin tulosten raportointi. Taulukossa 1 esitetään perustoiminnallisuuksien toteutuminen kunkin työkalun osalta. Kaikki työkalut, Prometheusta ja InfluxDB:tä lukuun ottamatta, tukivat määriteltyjä perustoiminnallisuuksia. Prometheuksen ei katsottu toteuttavan analysoinnin, visualisoinnin ja raportoinnin vaatimuksia riittävästi, kun taas InfluxDB:n osalta raportoinnin toteutusta ei saatu selvitettyä riittävästi.

Taulukko 1. Työkalujen perustoiminnallisuudet.

	Keräys / tallennus	Analysointi	Visualisointi	Hälytykset	Raportit
LGTM-pino	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Elastic-pino	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Prometheus	Kyllä	Ei	Ei	Kyllä	Ei
InfluxDB	Kyllä	Kyllä	Kyllä	Kyllä	Ei tiedossa
Splunk	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Graylog	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Loggly	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Sematext	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Datadog	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Nagios	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä

Työkalujen integrointi toimeksiantajan oppimisympäristöön ja muuhun järjestelmään on olennaista, joten työkaluja vertailtiin useaa kriteeriä käyttäen. Teoriatiedon perusteella järjestelmien tulisi tukea Syslog-menetelmää, mitä kaikki työkalut Prometheusta lukuun ottamatta riittävästi tukivat. Kaikki tarkastellut työkalut tarjosivat tuen Javan logging-paketille jollakin tavalla, joskin Datadogin osalta tätä ei saatu luotettavasti selvitettyä. Salassa pidettävät vaatimukset toteutuivat muiden kuin Logglyn, Sematextin, Datadogin ja Nagioksen osalta. Integraatiovaatimusten toteutuminen esitetään taulukossa 2.

Taulukko 2. Integraatiovaatimusten toteutuminen.

	Syslog	Java.util.logging	Salassa pidettävät
LGTM-pino	Kyllä	Kyllä	Kyllä
Elastic-pino	Kyllä	Kyllä	Kyllä
Prometheus	Ei	Kyllä	Kyllä
InfluxDB	Kyllä	Kyllä	Kyllä
Splunk	Kyllä	3. osapuolen	Kyllä
Graylog	Kyllä	3. osapuolen	Kyllä
Loggly	Kyllä	Kyllä	Ei
Sematext	Kyllä	Kyllä	Ei
Datadog	Kyllä	Ei tiedossa	Ei
Nagios	Kyllä	Kyllä	Ei

Tietoturvan ja tietosuojan vaatimuksia ovat teoriatiedon perusteella vähintään lokikirjausten lähettäjän rajoitusmahdollisuus, pääsyn hallintaominaisuudet ja mahdollisuus tietojen ajastettuun poistoon. Jotkin työkaluista luottivat lokikirjausten lähettäjiä rajoittaessaan lähinnä käänteisen välityspalvelimen ja palomuurin käyttöön, siinä missä osa toteutti rajoitusta myös sisäisesti. Vertailua tehtäessä voitiin todeta, että lokikirjausten lähettäjän rajoitus oli riittävissä määrin toteutettavissa kaikkiin työkaluihin. Pääsyn hallinta (käyttäjien kirjautuminen) toteutui Prometheusta lukuun ottamatta kaikissa työkaluissa, joskin Nagioksen osalta sitä ei saatu luotettavasti selvitettyä. Salassa pidettävät vaatimukset toteutuivat muiden kuin InfluxDB:n, Logglyn, Sematextin ja Nagioksen osalta, joista kolmen viimeisen kohdalla kaikkien vaatimusten toteutumista ei voitu riittävästi selvittää. Tietoturvan ja tietosuojan vaatimusten toteutuminen kunkin työkalun osalta esitetään yksinkertaistettuna taulukossa 3.

Taulukko 3. Tietoturvan ja tietosuojan vaatimusten toteutuminen.

	Lokikirjausten lähettäjän rajoitus	**Pääsyn hallinta**	**Tietojen ajastettu poisto**	**Salassa pidettävät**
LGTM-pino	Kyllä	Kyllä	Kyllä	Kyllä
Elastic-pino	Kyllä	Kyllä	Kyllä	Kyllä
Prometheus	Kyllä	Ei	Kyllä	Kyllä
InfluxDB	Kyllä	Kyllä	Kyllä	Ei
Splunk	Kyllä	Kyllä	Kyllä	Kyllä
Graylog	Kyllä	Kyllä	Kyllä	Kyllä
Loggly	Kyllä	Kyllä	Kyllä	Ei tiedossa
Sematext	Kyllä	Kyllä	Ei	Ei tiedossa
Datadog	Kyllä	Kyllä	Kyllä	Kyllä
Nagios	Kyllä	Ei tiedossa	Ei tiedossa	Ei tiedossa

Muita huomionarvoisia ominaisuuksia olivat tieto siitä, onko työkalu avointa lähdekoodia, ja onko se asennettavissa erilliselle palvelimelle (tai palvelimille). Kuten osiossa 3.2 havaittiin, on erilliselle palvelimelle asennettavuus teoriatiedon perusteella mielekäs vaatimus asettaa. Muita toimeksiantajan

vaatimuksia ei esitetä tässä työssä, vaan ne yhdistettiin salassa pidettävien tietojen sarakkeeseen. Näiden ominaisuuksien ja salassa pidettävien vaatimusten toteutuminen esitetään taulukossa 4.

Työkaluista LGTM- ja Elastic-pinojen ja Prometheuksen havaittiin olevan avointa lähdekoodia. InfluxDB:n versio kaksi (V2 OSS) on myös avointa lähdekoodia, mutta uudempi versio kolme ei. Nagioksen osalta vain Nagios Core -työkalu on avointa lähdekoodia. Kaikki työkalut olivat ainakin periaatteessa asennettavissa erilliselle palvelimelle, mutta Logglyn, InfluxDB V3:n ja Datadogin tapauksessa niistä ei ollut saatavilla maksullistakaan paikalliselle palvelimelle asennettavaa versiota. Toisin sanoen nämä osoittautuivat vain kunkin palveluntarjoajan omilla palvelimilla ajettaviksi työkaluiksi.

Kaikki salassa pidettävät muut vaatimukset toteutuivat LGTM- ja Elastic-pinojen ja Prometheuksen kohdalla muiden jäädessä vajaaksi vähintään yhden vaatimuksen osalta.

Taulukko 4. Muiden vaatimusten toteutuminen.

	Avointa koodia	**Asennettavissa erilliselle palvelimelle**	**Salassa pidettävät**
LGTM-pino	Kyllä	Kyllä	Kyllä
Elastic-pino	Kyllä	Kyllä	Kyllä
Prometheus	Kyllä	Kyllä	Kyllä
InfluxDB	V2 OSS	V2 OSS	Ei
Splunk	Ei	Kyllä	Ei
Graylog	Ei	Kyllä	Ei
Loggly	Ei	Ei	Ei
Sematext	Ei	Kyllä	Ei
Datadog	Ei	Ei	Ei
Nagios	Osittain	Kyllä	Ei

6 Päätelmät ja suositukset

Opinnäytetyöni tavoitteena oli selvittää vaatimuksia toimeksiantajan oppimisympäristön lokitietojen keräämiselle, analysoinnille ja visualisoinnille sekä kartoittaa toimeksiantajan käyttöön soveltuvat työkalut määriteltyjen vaatimusten toteuttamiseksi.

Vaatimusten keräämiseksi ja dokumentoimiseksi on useita menetelmiä. Tiukan yksittäisiin menetelmiin ja standardeihin sitoutumisen sijaan kannattaneekin yleensä pyrkiä hahmottamaan, miksi vaatimuksia kerätään ja miten vaatimuksia käytetään kulloinkin työstettävässä projektissa ketterästi. Vaatimusten keräämisen tarkoitusta voisikin kuvailla menetelmäsuuntautuneen lähestymistavan sijaan todellista arvoa tuottavien tarpeiden ja ominaisuuksien löytämiseksi, näkyväksi tekemiseksi ja toteuttamiseksi koko ohjelmistoprojektin ajan.

Vaatimusten määrittelyn havaittiin olevan tärkeää, tehtiin se sitten suunnitelmalähtöisesti ennen toteutuksen aloitusta tai ketterän kehityksen menetelmässä osana kutakin iteraatiota. Todellista arvoa tuottavien tarpeiden ja ominaisuuksien löytäminen, näkyväksi tekeminen ja toteuttaminen ovatkin olennaisessa osassa onnistunutta ohjelmistokehitysprosessia. Huolella määritellyt vaatimukset toimivat myös järjestelmän testauksen perustana ja saattavat olla hyvä tekninen yhteenveto projektista.

Työn aikana havaittiin, että verkkosovellusten lokitietojen kerääminen riittävän kattavasti ja tallentaminen ulkoiselle lokipalvelimelle ovat tärkeitä lokitietojen käytettävyyden, eheyden ja tietosuojan varmistamiseksi sekä tiedoista tehtävän analysoinnin ja niiden pohjalta tehtävien hälytysten mahdollistamiseksi. Verkkosovellusten ja -palvelinten, lokien käyttäjien sekä lainsäädännön asettamien vaatimusten huomiointi on tärkeää ja vaikuttaa merkittävästi valittavien ratkaisujen toteutukseen, minkä vuoksi yhtä jokaiseen ympäristöön sopivaa ratkaisua sovelluslokien keräämiseksi ja analysoimiseksi ei voida antaa. Yleisesti voidaan kuitenkin todeta, että tarkastellut työkalut olivat varsin valmiin oloisia ja laajoja lokitietojen hallinnan kokonaisuuksia.

Kaikki yleiset ja toimeksiantajan sisäiset vaatimukset toteuttavia työkaluja olivat LGTM- ja Elastic-pinojen työkalukokonaisuudet. Myös Graylog ja Datadog vaikuttivat hyvin viimeistellyiltä työkaluilta, vaikka eivät täyttäneetkään kaikkia toimeksiantajan vaatimuksia. Suosittelen toimeksiantajalle LGTM-pinon tuotteita mahdollisesti yhdistettynä Prometheus-aikasarjatietokantaan, mikäli sellaista erikseen tarvitaan. Toimeksiantajan kannattaisi mielestäni jatkaa lokitietojen keräysprojektiaan asentamalla LGTM-pinon tuotteet testipalvelimelle aloittaen Loki- ja Grafana-työkalujen käyttöönotosta, minkä jälkeen oppimisympäristön kehitysversio voidaan yhdistää tuottamaan tietoja näihin työkaluihin.

Lähteet

Anusooya, R., Rajan, J., & Satya Murty, S. A. V. (2015). Importance of centralized log server and log analyzer software for an organization. *International Research Journal of Engineering and Technology (IRJET)*, *2*(3), 2244–2250.

Bieman, J., & Fenton, N. E. (2014). *Software metrics: a rigorous and practical approach* (Third edition). CRC Press.

Bonadies, E. (2023). *Getting started with the Grafana LGTM stack*. https://grafana.com/go/webinar/getting-started-with-grafana-lgtm-stack/

Case, J., Davin, J., Fedor, M., & Schoffstall, M. (1988). *RFC 1067 - A Simple Network Management Protocol*. https://datatracker.ietf.org/doc/html/rfc1067

Choudhary, S. R., & Orso, A. (2009). Automated Client-Side Monitoring for Web Applications. *2009 International Conference on Software Testing, Verification, and Validation Workshops*, 303–306. https://doi.org/10.1109/ICSTW.2009.44

Chuvakin, A., & Peterson, G. (2010). How to Do Application Logging Right. *IEEE Security & Privacy Magazine*, *8*(4), 82–85. https://doi.org/10.1109/MSP.2010.127

Chuvakin, A., Schmidt, K., & Phillips, C. (2013). *Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management*. Syngress.

Darrington, J. (2023). *What is Log Management? A Complete Logging Guide*. Graylog, Inc. The Graylog Blog. https://graylog.org/post/what-is-log-management-a-complete-logging-guide/

Datadog Inc. (ei pvm.). *Datadog Product*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.datadoghq.com/product/

Dick, J., Hull, E., & Jackson, K. (2017). *Requirements Engineering*. Springer International Publishing. https://doi.org/10.1007/978-3-319-61073-3

Elasticsearch B.V. (ei pvm.-a). *Beats*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.elastic.co/beats

Elasticsearch B.V. (ei pvm.-b). *Elasticsearch*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.elastic.co/elasticsearch

Elasticsearch B.V. (ei pvm.-c). *Kibana*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.elastic.co/kibana

Elasticsearch B.V. (ei pvm.-d). *Logstash*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.elastic.co/logstash

EU:n yleinen tietosuoja-asetus 2016/679.

Gerhards, R. (2009). *RFC 5424 - The Syslog Protocol*. IETF Trust. https://datatracker.ietf.org/doc/rfc5424/

Goldsmith, R. (2004). *Discovering Real Business Requirements for Software Project Success*. Artech House.

Grady, J. O. (2006). *System Requirements Analysis*. Elsevier Inc.

Grafana Labs. (ei pvm.-a). *Grafana*. Noudettu 22. marraskuuta 2023, osoitteesta https://grafana.com/oss/grafana/

Grafana Labs. (ei pvm.-b). *Grafana Loki*. Noudettu 22. marraskuuta 2023, osoitteesta https://grafana.com/oss/loki

Grafana Labs. (ei pvm.-c). *Grafana Mimir*. Noudettu 22. marraskuuta 2023, osoitteesta https://grafana.com/oss/mimir/

Grafana Labs. (ei pvm.-d). *Grafana Tempo*. Noudettu 22. marraskuuta 2023, osoitteesta https://grafana.com/oss/tempo/

Graylog Inc. (ei pvm.). *Graylog*. Noudettu 22. marraskuuta 2023, osoitteesta https://graylog.org

Haikala, I., & Märijärvi, J. (2004). *Ohjelmistotuotanto* (Kymmenes painos). Talentum Media Oy.

IEEE STD 12207-2017. (2017). *ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes*.

InfluxData Inc. (ei pvm.). *Get Started with InfluxDB*. Noudettu 22. marraskuuta 2023, osoitteesta https://docs.influxdata.com/influxdb/v2/get-started/

International Organization for Standardization. (ei pvm.). *ISO name and logo*. Noudettu 23. marraskuuta 2023, osoitteesta https://www.iso.org/iso-name-and-logo.html

Internet Engineering Task Force. (ei pvm.). *Introduction to the IETF*. Noudettu 13. marraskuuta 2023, osoitteesta https://www.ietf.org/about/introduction/

JavaTpoint. (ei pvm.). *Software Metrics*. Noudettu 23. marraskuuta 2023, osoitteesta https://www.javatpoint.com/software-engineering-software-metrics

Kidd, C. (2019). *Tracing vs Logging vs Monitoring: What's the Difference?* https://www.bmc.com/blogs/monitoring-logging-tracing/

Kidd, C. (2022). *Data Normalization Explained: How To Normalize Data*. https://www.splunk.com/en_us/blog/learn/data-normalization.html

Laine, H. (2002). *History of SQL*. https://www.cs.helsinki.fi/u/laine/tuelip/sql_material/sql_history.html

Leach, R. J. (2016). *Introduction to software engineering* (Second edition). CRC Press, Taylor & Francis Group.

Marty, R. (2011). Cloud application logging for forensics. *Proceedings of the 2011 ACM Symposium on Applied Computing*, 178–184. https://doi.org/10.1145/1982185.1982226

MDN Contributors. (2023a). *MDN web docs: HTTP*. https://developer.mozilla.org/en-US/docs/Web/HTTP

MDN Contributors. (2023b). *MDN web docs: Transport Layer Security*. https://developer.mozilla.org/en-US/docs/Web/Security/Transport_Layer_Security

Microsoft. (ei pvm.). *NoSQL Database - What is NoSQL?* Noudettu 7. marraskuuta 2023, osoitteesta https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-nosql-database

Nagios Enterprises LLC. (ei pvm.). *Nagios*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.nagios.org

Oracle. (2023, lokakuuta). *Java Logging Overview*. Java Core Libraries Developer Guide, Release 21. https://docs.oracle.com/en/java/javase/21/core/java-logging-overview.html

OWASP Top 10. (2021). *A09:2021 – Security Logging and Monitoring Failures*. https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/

Prometheus Authors, & The Linux Foundation. (ei pvm.). *Prometheus Overview*. Noudettu 22. marraskuuta 2023, osoitteesta https://prometheus.io/docs/introduction/overview/

QOS.ch Sarl. (ei pvm.). *Reasons to prefer logback over log4j 1.x*. Noudettu 23. marraskuuta 2023, osoitteesta https://logback.qos.ch/reasonsToSwitch.html

Red Hat. (ei pvm.). *System Administrator's Guide - Chapter 23. Viewing and Managing Log Files*. Noudettu 24. lokakuuta 2023, osoitteesta https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-viewing_and_managing_log_files#doc-wrapper

Rinnan, R. (2005). *Benefits of centralized log file correlation* [Master's Thesis, Gjøvik University College]. https://ntnuopen.ntnu.no/ntnu-

xmlui/bitstream/handle/11250/143787/Rinnan%20-%20Benefits%20of%20centralized%20log%20file%20correlation.pdf

Sematext Group Inc. (ei pvm.). *Cloud Monitoring Tools & Services*. Noudettu 22. marraskuuta 2023, osoitteesta https://sematext.com/cloud/

SolarWinds Worldwide LLC. (ei pvm.). *Loggly Product Overview*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.loggly.com/product/

Sommerville, I. (2016). *Software Engineering, Global Edition* (Tenth Edition). Pearson Education Limited.

Splunk Inc. (ei pvm.). *Splunk Products*. Noudettu 22. marraskuuta 2023, osoitteesta https://www.splunk.com/en_us/products.html

Tarkoma, S. (2010). *Tietoliikenteen perusteet - Kuljetuskerros*. https://www.cs.helsinki.fi/u/starkoma/tilpe/luku3

The Apache Software Foundation. (2023). *Log4j 2 Manual: Introduction*. https://logging.apache.org/log4j/2.x/manual/index.html

Traficom. (2019). *Näin keräät ja käytät lokitietoja*. https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/ohjeet-ja-oppaat/nain-keraat-ja-kaytat-lokitietoja

Vaadin Ltd. (2021). *Vaadin V8 Docs: Logging*. https://vaadin.com/docs/v8/framework/advanced/advanced-logging

Vaadin Ltd. (2023). *Vaadin V24 Documentation: Custom Error Handling*. https://vaadin.com/docs/latest/advanced/custom-error-handler

Wang, D., Galster, M., & Morales-Trujillo, M. (2024). Application Monitoring for bug reproduction in web-based applications. *Journal of Systems and Software*, *207*. https://doi.org/10.1016/j.jss.2023.111834

Williams, N. S. (2014). *Professional Java for Web Applications* (1. p.). John Wiley & Sons, Incorporated.

Young, R. (2003). *Requirements Engineering Handbook* (1. p.). Artech House.

Esimerkinomainen vaatimusmäärittelydokumentti

Kuvaus

Tämä vaatimusmäärittely ja siinä esitetyt vaatimukset ovat täysin kuvitteellisia. Niitä käytetään vain esimerkinomaisesti kuvaamaan sitä, minkälainen vaatimusmäärittelydokumentti opinnäytetyön toimeksiantajalle toteutettiin.

Käytettävä luokittelu ja numerointi

Vaatimukset jaetaan toiminnallisiin [FR] ja ei-toiminnallisiin [NFR] vaatimuksiin.

Vaatimukset numeroidaan kunkin tyypin sisällä juoksevasti, esimerkiksi toiminnallinen vaatimus 21 saa koodin [FR-21] otsikkonsa eteen.

Vaatimukselle merkitään tärkeys asteikolla 1–5, jossa 5 on suurin tärkeys.

Vaatimuksen tila: alustava / vahvistettu / poistettu.

Esimerkkivaatimus

[NFR-1] Ei-toiminnallinen vaatimus yksi

Tärkeys (1–5): 2
Tila: vahvistettu

Vaatimuksen kuvaus.

Lisätiedot alaotsikoin eroteltuina, jos tarvitaan. Esimerkiksi tarkemmat perustelut, kommentit, testauskriteerit jne.

Toiminnalliset vaatimukset

[FR-1] Järjestelmän tulee vaatia turvallisia salasanoja

Tärkeys (1–5): 5
Tila: vahvistettu

Järjestelmän käytettävien salasanojen tulee noudattaa NIST SP 800-63b:n kohdan 5.1.1.2 Memorized Secret Verifiers määräyksiä ja ohjeistuksia. Järjestelmän tulee valvoa tämän noudattamista. Linkki määräykseen: https://doi.org/10.6028/NIST.SP.800-63b.

[FR-2] Järjestelmään tulee voida kirjautua sisäverkon tietokoneilta

Tärkeys (1–5): 3
Tila: alustava

Sisäverkosta kirjautumisen on onnistuttava käyttäen yksilöllistä käyttäjätunnus-salasanaparia.

Ei-toiminnalliset vaatimukset

[NFR-1] Järjestelmän ei tule aiheuttaa hidastumista järjestelmään tietoja syöttävissä sovelluksissa

Tärkeys (1–5): 5
Tila: vahvistettu

Järjestelmä on toteutettava siihen tietoja syöttävistä sovelluksista eriytetysti siten, että se ei aiheuta asiakassovelluksissa näkyvää hidastumista. Huomio: todellisessa vaatimusmäärittelyssä tähänkin vaatimukseen kannattaisi kirjata tarkemmin mitattavia kriteereitä ja testitapauksia mainitulle ”hidastumiselle”.