Ilona Nurminen

Syslog-protokollan viestien analysointi järjestelmän vianetsinnän apuna

Tietotekniikan kandidaatintutkielma

29. huhtikuuta 2017

Jyväskylän yliopisto

Tietotekniikka

Tekijä: Ilona Nurminen

Yhteystiedot: k.ilona.nurminen@student.jyu.fi

Ohjaaja: Marjaana Nokka

Työn nimi: Syslog-protokollan viestien analysointi järjestelmän vianetsinnän apuna

Title in English: Analysis of Syslog messages for system troubleshooting

Työ: Kandidaatintutkielma

Sivumäärä: 25+0

Tiivistelmä: Tietojärjestelmät keräävät toiminnastaan jatkuvasti lokitietoja. Vikatilanteissa lokitietoja voidaan hyödyntää virheen paikantamisen apuna. Lokinkirjoittamisen teollisuusstandardiksi on noussut Syslog-protokolla. Tarve protokollaa varten kehitetyille lokianalyysitekniikoille ja -työkaluille on näin ollen noussut. Tässä kandidaatintutkielmassa pyritään esittelemään näitä analyysitekniikoita ja -työkaluja, ja pohtimaan näiden hyödyllisyyttä järjestelmän vianetsintää suorittavan järjestelmänvalvojan näkökulmasta.

Avainsanat: Syslog, lokianalyysi, järjestelmänvalvonta, vianetsintä

Abstract: Information systems constantly gather operational log data. In case of system failure, this log data can be used as a troubleshooting tool in locating the problem source. The Syslog protocol has become an industry standard in logging. For this reason, a need for log analysis techniques and tools for the protocol has arisen. This bachelor's thesis attempts to introduce these techniques and tools, as well as consider their effectiveness in system troubleshooting from a system administrator's viewpoint.

Keywords: Syslog, log analysis, system administration, troubleshooting

Kuviot

Sisältö

1 Johdanto

Moderni tietojärjestelmä on moniosainen kokonaisuus, jota voidaan käyttää ajasta ja paikasta riippumatta. Maantieteelliset etäisyydet merkitsevät yhä vähemmän tietoverkkojen yhteen sitomassa maailmassa. Nämä tietoverkot myös kytkevät yhteen järjestelmiä, jotka huolehtivat tietoyhteiskunnan kriittisistä toiminnoista. Poikkeustilanteessa ongelman lähde tulee selvittää viipymättä ja mahdollisimman tehokkaasti. Tämä voi kuitenkin olla haastavaa, sillä ongelman ydin saattaa piillä järjestelmän useassa osassa. Vikaa jäljittävän tahon tulee siis saada järjestelmältä tietoa vikatilanteeseen johtaneesta toiminnasta. Tätä tarkoitusta varten tietojärjestelmät kirjaavat ylös lokitietoja järjestelmän tapahtumista.

Lokitiedoilla on järjestelmän kannalta useita tärkeitä rooleja: ne paitsi avustavat ongelmien diagnosoinnissa, myös kuvaavat järjestelmän tyypillistä toimintaa, antavat tietoa järjestelmän käyttäjien toimintatavoista ja saattavat jopa toimia todisteena oikeudenkäyntitilanteessa. Sovelluskehittäjät käyttävät lokeja kehitystyössään osana debug-prosessia. Joskus tarve lokien keräämiseen voi myös tulla organisaation ulkopuolelta esimerkiksi teollisuusalan vaatimien standardien muodossa.. On myös huomioitava, että useat modernit IT-hallintoviitekehykset, kuten COBIT ja ISO 27002 asettavat lokien keräämiselle ja säilyttämiselle tarkat säännöt (Nemeth ym. 2011).

Lokien kirjaamisen standardiksi on noussut Eric Allmanin 1980-luvulla kehittämä Syslog-protokolla. Tässä tutkielmassa selvitetään, miten Syslog-protokolla toimii, millaisia keinoja protokollan tulosteen analysointiin on olemassa ja miten ne toteutuvat käytännössä. Tutkielma huomioi erityisesti järjestelmänvalvojaksi kutsutun henkilön tai henkilöryhmän näkökulman. Kuten Xu ym. (2009) esittävät, järjestelmänvalvoja on henkilö, joka hyödyntää lokien tulostetta järjestelmän ylläpidossa ja poikkeustilanteissa. Järjestelmänvalvojia voi olla yksi tai useampia, ja he voivat keskittyä tietojärjestelmän eri osiin.

Tutkimus toteutetaan kirjallisuuskatsauksena. Keräämällä yhteen eri tekniikoiden hyötyjä ja heikkouksia voidaan muodostaa läpileikkaus aihealueen moninaisista lähestymistavoista. Kun tiedetään, miten aihetta on aiemmin tutkittu, on mahdollista pohtia omaa lähestymiskulmaa. Tutkimuksen teoriapuolta on myös tarkoitus havainnollistaa käytännön esimerkkien ja

pohdinnan kautta ja kytkeä työtä näin lähemmäs järjestelmänvalvojan arkea. Kirjallisuuskatsaus on siis luonteva lähestymistapa nämä lähtökohdat huomioiden. Lokianalyysi aihepiirinä elää ja muuttaa muotoaan jatkuvasti. Kehittyvä teknologia luo puitteet analyysitekniikoille, joiden toteuttaminen ei olisi aiemmin ollut mahdollista lainkaan. Parhaan mahdollisen toteutusratkaisun valitsemiseksi järjestelmän kehittäjän tulee olla tietoinen alan menneistä ja nykyisistä käytännöistä, eikä tulevaisuuteenkaan katsominen ole pahitteeksi.

Tutkielman luvussa 2 käydään ensin lyhyesti läpi Syslog-protokollan tausta ja perehdytään protokollan standardisoituun määritelmään. Lisäksi luvussa esitellään joitakin määritelmän mukaisia protokollaimplementaatioita. Luvussa 3 pohditaan, miksi lokien analysointi voi olla tärkeää ja käydään läpi keskeisimpiä lokianalyysin taustalla vaikuttavia teorioita. Luvussa 4 selvitetään, miten lokianalyysia on tutkittu nimenomaan järjestelmän vianetsinnän kannalta. Lopuksi luvussa 5 pyritään tuomaan esille aikaisempien lukujen olennaisimmat tulokset ja näiden kautta syntyneet johtopäätelmät.

2 Syslog-protokolla

Tässä luvussa käsitellään aluksi lyhyesti syslog-protokollan historiaa. Sen jälkeen tarkastellaan syslog-protokollan rakennetta ja esitellään joitakin sen implementaatioita.

Syslog-protokollan juuret ovat BSD (Berkeley Software Distribution)-käyttöjärjestelmäprojektissa, johon Eric Allman kehitti Sendmail-ohjelmaprojektia varten syslog-nimisen lokinkirjoitusjärjestelmän (Vixie ja Avolio 2002). Se muodostui ajan myötä teollisuusstandardiksi, josta oli käytössä useita poikkeavia versioita eri ohjelmistoprojekteissa. Vuonna 2001 julkaistiin RFC 3164-dokumentti, jossa esiteltiin protokolla sen erilaisten käytössä olevien implementaatioiden kautta. Näin ollen se ei ole varsinainen protokollan määrittelevä standardi, vaan kokoelma protokollan havaittuja toteutustapoja. RFC 3164:n mukaista syslogia kutsutaan yleisesti BSD-syslogiksi kyseisen dokumentin nimen mukaan (Lonvick 2001). Syslog-protokolla standardisoitiin virallisesti IETF:n toimesta vuonna 2009 julkaistussa RFC 5424-dokumentissa. Standardisoinnin tavoitteena oli yhdenmukaistaa protokollan määrittely implementaatioiden ja protokollalaajennusten yhteensopivuuden parantamiseksi (Gerhards 2009).

Tässä tutkielmassa käsitellään pääasiassa RFC 5424:n mukaista modernia syslog-protokollaa, jota tutkielmassa nimitetään IETF-syslogiksi. Vaikka RFC 3164:ssä esitellyn mukaisia protokollatoteutuksia on edelleen käytössä, tutkielmassa on tarkoitus perehtyä työkaluihin, joiden toiminnallisuus saattaa edellyttää RFC 5424:n mukaisesti toteutettua syslog-implementaatiota.

2.1 Protokollan määrittely

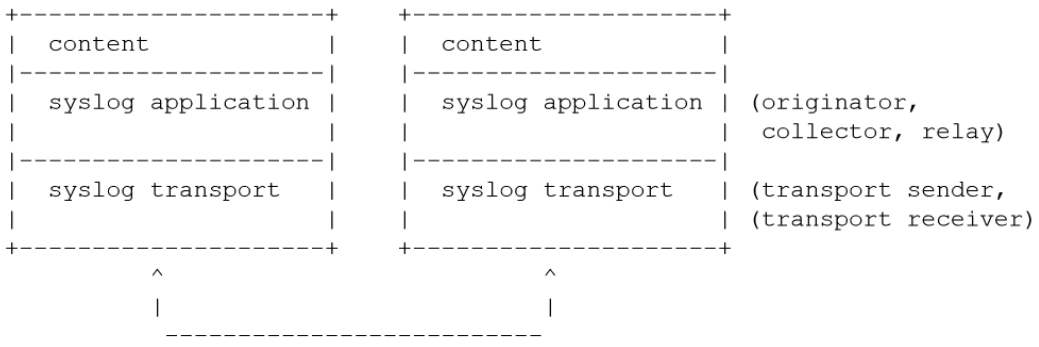
Tässä luvussa määritellään Syslog-protokollan perusspesifikaatio Gerhards (2009) laatimaa RFC 5424-dokumenttia mukaillen. Protokollasta käsitellään tutkielman aiheen kannalta oleellisimmat seikat.

Syslog on järjestelmän tapahtumien raportointiin tarkoitettu protokolla. Standardisoidussa nykymuodossaan se hyödyntää kolmikerroksista suunnittelumallia, jossa viestit pidetään

erillään niitä kuljettavasta ja käsittelevästä osasta. Tämä kolmijako mahdollistaa kerrosten toisistaan erillisen laajentamisen.

Syslog-protokollan kolme kerrosta ovat kuvion 1 mukaan

- **Syslog-sisältökerros** (Syslog content) pitää sisällään Syslog-viestin sisältämän varsinaisen tiedon.
- **Syslog-ohjelmakerros** (Syslog application) hallitsee syslog-viestien luomista, tulkintaa, reititystä ja säilytystä.
- **Syslog-kuljetuskerros** (Syslog transport) huolehtii viestien lähettämisestä ja vastaanottamisesta.



Kuvio 1. Syslog-protokollan kerrosmalli. (Gerhards 2009)

Kerrosten osilla on tiettyjä rooleja. Syslog-viestin sisällön luovaa osaa sanotaan luojaksi *(originator)*. Kerääjä *(collector)* puolestaan kerää syslog-viestejä lajittelua varten. Välittäjä *(relay)* ottaa vastaan viestejä luojilta ja muilta välittäjiltä ja välittää niitä eteenpäin kerääjille tai muille välittäjille. Lähettävä ja vastaanottava kuljettaja *(transport sender/transport receiver)* puolestaan huolehtivat viestien siirtämisestä kuljetusprotokollalle.

Perusmäärittelyltään Syslog on yksisuuntainen protokolla, eli siihen ei sisälly luotettavaa tiedonvälitystä. Näin ollen viestien perille saattaminen on järjestelmänvalvojan vastuulla. RFC 5424 ei määrittele syslogin kanssa käytettävää kuljetuskerroksen protokollaa, mutta asettaa vähimmäisvaatimukseksi tuen UDP- ja TLS-protokollille. Kuljetusprotokolla ei myöskään saa muokata syslog-viestin sisältöä, ellei se palauta viestiä ennalleen siirtäessään viestiä vas-

taanottavalle kuljettajalle.

Syslog-viestit ovat itsessään salaamatonta raakatekstiä. Viesteille on määritelty tarkka koko ja rakenne yhteensopivuuden takaamiseksi. Viestin minimikoko on 480 oktettia, suurin mahdollinen koko puolestaan 2048 oktettia. Maksimikokoa suuremmat viestit voidaan joko hylätä tai typistää lyhyemmiksi; vastuu tästä jätetään implementaatiolle.

```
<34>1 2003-10-11T22:14:15.003Z mymachine.example.com su - ID47
- BOM'su root' failed for lonvick on /dev/pts/8
```

Kuvio 2. Esimerkki Syslog-protokollan mukaisesta viestistä. Huomioi, ettei su-prosessilla ole prosessi-ID:tä. (Gerhards 2009)

Järjestelmänvalvojan kannalta Syslog-viestin määrittävät luvut ovat vakavuus *(severity)* ja laitos *(facility)*. Nämä löytyvät kuviossa 2 alun hakasuluista nk. *PRI*-kentästä. Vakavuus määrittelee, kuinka tärkeästä viestistä on kyse. Vakavuutta mitataan asteikolla 0-7, jossa 0 tarkoittaa kriittistä virhettä ja 7 debug-tasoista tiedotetta. Laitos puolestaan kertoo, mitä järjestelmän osaa viesti koskee. Laitosarvo voi olla väliltä 0-23. *PRI*-kenttää varten vakavuudesta ja laitoksesta lasketaan yhteisarvo kertomalla laitosarvo kahdeksalla ja lisäämällä siihen vakavuusarvo.

Hakasulkujen jälkeen tuleva luku määrittää viestin käyttämän protokollaspesifikaation version. Aikaleimaa seuraa viestin lähettäneen koneen identifioiva isäntänimi *(hostname)*. Seuraavat kohdat määrittelevät viestin lähettäneen prosessin nimen ja prosessi-ID:n, jonka jälkeen määritellään viestin tyyppi *MSGID*-kentässä. Näitä määrittelyjä seuraa varsinainen viestikenttä, johon viestin lähettänyt ohjelma voi kirjata asiansa. Joskus viestissä voi olla mukana myös *STRUCTURED-DATA*-kenttä, johon voidaan kirjata viestiin liittyvää metatietoa tai ohjelmakohtaisia lisätietoja.

Syslog määritelmän mukaan on hyvin riisuttu, mutta tehokas viestinvälitysprotokolla. Se ei tarjoa luotettavaa tiedonvälitystä eikä salausta, mutta korvaa sen keveydellään. Seuraavassa luvussa tarkastellaan määritelmän pohjalta rakennettuja protokollaimplementaatioita, jotka pyrkivät parantamaan protokollan eri osa-alueita.

2.2 Protokollan implementaatioita

Protokollamäärittelyn tavoitteena on asettaa minimivaatimukset implementaatioita varten, eli määrittää ehdot, jotka implementaation on täytettävä, jotta sitä voi kutsua protokollaa noudattavaksi. Implementaatioon voidaan kuitenkin lisätä ominaisuuksia ja sitä voidaan muokata käyttötarkoitukseen sopivaksi protokollamäärittelyn sallimissa rajoissa. Tässä kappaleessa tarkastellaan lyhyesti muutamaa yleisesti käytössä olevaa Syslog-implementaatiota ja käydään läpi niiden erityisominaisuudet IETF-syslogin määritelmään verrattuna.

2.2.1 sysklogd

Alkuperäistä BSD-syslogia muistuttaa eniten syslogkd, joka noudattaa RFC 3164:ssa muodostettua BSD-syslogin määritelmää. Sysklogd-paketti koostuu syslogd- ja klogd-daemoneista. *Syslogd* huolehtii ohjelmien lähettämien lokien kirjaamisesta paikallishostilla ja etähosteilla, *klogd* puolestaan käsittelee kerneliltä tulevia järjestelmäviestejä. Viimeinen varsinainen versio sysklogd:sta julkaistiin vuonna 2007, jota seurasi tietoturvapäivitys vuonna 2014 (Schulze 2014).

Koska sysklogd ei tue IETF-syslogin standardisoitua viestimuotoa sellaisenaan, sitä ei välttämättä voida käyttää modernien lokianalyysityökalujen kanssa. Se on silti syytä mainita yleisesti käytössä olevien syslog-implementaatioiden joukossa historiallisen asemansa takia.

2.2.2 syslog-ng

Syslog-ng on eräs lokinkirjoitusohjelmisto. Se on saatavilla usealle eri järjestelmäarkkitehtuurille ja käyttöjärjestelmälle. Erityisen syslog-ng:stä tekee sen julkaisutapa. Ohjelmisto on saatavilla kahtena eri versiona: avoimen lähdekoodin OSE-versiona (Open Source Edition) ja maksullisena, suljettuna Premium-versiona (Premium Edition). Premium-versio tarjoaa tiettyjä lisäominaisuuksia, kuten salatun viestinvälityksen ja tuen Windows-käyttöjärjestelmille. Syslog-ng kehitettiin alunperin BSD-syslogia varten, mutta versiossa 3.0 lisättiin tuki IETF-syslogille. RFC 5424:n määrittelemien kuljetusprotokollien lisäksi syslog-ng tukee TCP-pohjaista luotettavaa viestinvälitystä. Lisäksi se tarjoaa vuonhallintaominaisuudet viestivirran säätelyyn. Syslog-ng kykenee suodattamaan, muotoilemaan ja pilkkomaan viestejä esi-

merkiksi säännöllisiä lausekkeita hyödyntäen (*The syslog-ng Open Source Edition 3.9 Administrator Guide* 2017).

2.2.3 rsyslog

Rsyslog on vaihtoehdoksi syslog-ng:lle kehitetty lokinkirjoitusohjelmisto. Rsyslog-projektin alkuunpanija Rainer Gerhards (2007) ilmaisi blogikirjoituksessaan: ”Uusi tekijä markkinoilla ehkäisee monokulttuurien muodostumista ja tarjoaa valinnanvapautta.” Rsyslog tarjoaa syslog-ng:n tapaan monipuoliset suodatusominaisuudet: viestejä voidaan suodattaa viestin osien tai säännöllisillä lausekkeilla määriteltyjen suodattimien avulla. Ohjelmistolle on saatavilla sysklogd:n klogd-osan toteuttava *imklog*-laajennus, joka tekee rsyslog:sta täysin sysklogd-yhteensopivan. Rsyslog tukee kuljetusprotokollista UDP:n ja TLS:n lisäksi RELP-protokollaa *(Reliable Event Logging Protocol)*. RELP tarjoaa syslog-viesteille luotettavan tiedonvälityksen, joka toteutetaan TCP-protokollan avulla (*Rsyslog 8.26.0 documentation* 2017).

2.3 Syslog ja muut lokinkirjoitusmenetelmät

On tärkeää huomioida, etteivät kaikki laajemmalti käytössä olevat lokinkirjoitusratkaisut perustu syslog-protokollaan. Syslogin perinteinen raakatekstiformaatti voidaan nähdä sekä etuna että haittana. Raakateksti on helposti käsiteltävissä monilla standardiohjelmilla, kuten UNIX-kaltaisten käyttöjärjestelmien *grep*-hakutyökalulla. Toisaalta erityisen suurten tekstitiedostojen käsittely ei ole kovinkaan nopeaa, ja tekstitiedostojen nopea kasvu vaatii säännöllistä vanhan tiedon arkistointia. Moniriviset lokiviestit voivat myös aiheuttaa ongelmia käsittelytyökaluille (Schäfer 2016). Jos nämä seikat tuottavat järjestelmässä ongelmia, saattaa olla tarpeen tutustua johonkin vaihtoehtoiseen lokinkirjoitusratkaisuun.

Eräs näistä ratkaisuista on systemd-järjestelmäpakettiin kuuluva *journald*. Se käyttää perinteisten raakatekstilokien sijaan omaa tiedostotyyppiään. Tiedostotyyppi mahdollistaa lokitietojen tehokkaamman käsittelyn ja selauksen, mutta raakatekstin käsittelyä varten kehitetyt työkaluohjelmat eivät ole yhteensopivia journald:n kanssa. Journald tukee viestien välitystä syslog-protokollaimplementaatioille. On huomioitava, että journald ei tarjoa syslogin kal-

taista client-server-arkkitehtuuria keskitetyn lokipalvelimen käyttöä varten, ja sellaista kaivattaessa on syytä käyttää syslogia yhdessä journald:n kanssa (Schäfer 2016).

Nykyään monet yleisesti saatavilla olevat lokianalyysityökalut tukevat syslog-versioiden rinnalla myös muita lokinkirjoitusjärjestelmiä. On järjestelmänvalvojan vastuulla päättää, millainen lokinkirjoitusratkaisu sopii parhaiten omaan järjestelmään ja sopii yhteen mahdollisten olemassaolevien ohjelmistojen kanssa.

3 Syslog-tulosteen analysointi

Tässä luvussa käsitellään syslog-protokollan antaman tulosteen analysointia eli lokianalyysia. Ensiksi pohditaan, miksi analyysista voisi olla hyötyä ja mitä analyysilla tarkoitetaan. Sen jälkeen esitellään joitakin aiheeseen liittyviä keskeisiä teorioita ja lähestymistapoja.

3.1 Miksi analysoida?

Kuvitellaan tilanne, jossa yrityksen, viraston tai intohimoisen harrastajan verkkoon on onnistuneesti pystytetty syslog-palvelin, joka ottaa vastaan viestejä siihen kytketyistä laitteista. Verkossa voi olla lukuisia erilaisiin tehtäviin erikoistuneita laitteita - palvelimia, työasemia, kytkimiä, palomuureja, levypakkoja ja kahvinkeittimiä. Näillä kaikki luonnollisesti raportoivat syslogin välityksellä oman toimintansa kannalta olennaisista asioista. Lokit voivat pitää sisällään kaikkea aina normaaliin toimintaan liittyvistä tilatiedoista kriittisiin virheisiin. Jotkut ohjelmat haluavat kirjata ylös jokaisen liikkeensä ja välittää ne lokipalvelimelle.

Ongelmatilanteessa lokien merkitys korostuu. Verkon ruuhkautumista tutkittaessa voi olla tarpeen perehtyä esimerkiksi kytkimien ja palomuurin lokeihin, kun taas verkkolevyjen hidastelu saattaa johtaa levypakan lokien jäljille. Aina vika ei kuitenkaan selkeästi paikannu yhteen laitteeseen tai verkon osaan. Kokenut järjestelmänvalvoja saattaa edetä vaistonsa varassa tutkimaan tiettyjä lokeja, mutta useimmiten seassa on paljon ongelman kannalta hyödytöntä tietoa. Ongelma on myös saattanut syntyä pitemmällä aikavälillä, eikä lokien kuukausia taaksenpäin kelaamisessa ole välttämättä mieltä. Joskus lokiviestien sisältö itsessään voi olla hankalasti ymmärrettävää ilman syvempää perehtymistä kyseisen järjestelmän osan lokinkirjoitusmetodologiaan. Miten järjestelmänvalvoja voisi vastata näihin tarpeisiin lokien manuaalista silmäilyä tehokkaammin?

Jiang ym. (2009) esittävät järjestelmäongelmien koostuvan oireista *(problem symptom)* ja ongelman aiheuttaneista syistä *(problem root cause)*. Järjestelmänvalvojan tehtävänä on siis lääkärin tavoin pyrkiä selvittämään oireiden perusteella niitä aiheuttava syy. Kun ongelma on paikannettu, voidaan siihen etsiä ratkaisu. Lokien tapauksessa oireiden havaitseminen tarkoittaa lokeista löytyneiden poikkeuksellisten tapahtumien ja mahdollisesti niiden välisten

suhteiden tarkastelua. Tähän voidaan ottaa avuksi analyysityökalu, joka karsii lokivirrasta aiheeseen liittymättömän tiedon ja korostaa normaalista toiminnasta poikkeavat tapahtumat.

Lokianalyysi on siis

- lokien tulkitsemista
- mahdollisten ongelmien havaitsemista lokeista
- virheisiin johtaneen toiminnan tarkastelua
- ongelmien ennaltaehkäisyä.

Syslog määritelmän mukaan ainoastaan kerää, välittää ja vastaanottaa lokitietoja, ei analysoi niitä. Analyysi on siis jätetty implementaatioiden ja niitä hyödyntävien erillisten analyysityökalujen huoleksi. Seuraavaksi tarkastelemme joitakin lähestymistapoja lokianalyysiin.

3.2 Lähestymistapoja lokianalyysiin

Tässä luvussa tarkastellaan muutamia lokianalyysiin liittyviä keskeisiä teorioita ja teoreettisia malleja. Nämä mallit on jaettu käsittelyn tueksi kolmeen pääkategoriaan. Ensiksi käsitellään watchdog-mallia ja avainsanapohjaisia skriptejä, jonka jälkeen siirrytään tiedonlouhintamenetelmiin. Lopuksi tarkastellaan sääntöpohjaista menetelmää.

3.2.1 Watchdog-malli ja avainsanapohjaiset skriptit

Lokien monitorointi voidaan toteuttaa varsin seikkaperäisesti nk. watchdog-ajastin-suunnittelumallia hyödyntäen.

Watchdog on järjestelmän toimintaa valvova osa, jonka toiminta perustuu ajastimeen ja sen hallintaan. Normaalitilanteessa järjestelmä laittaa ajastimen käyntiin laskemaan kohti nollaa. Tämän jälkeen järjestelmä voi suorittaa erilaisia testejä järjestelmän eri osa-alueisiin liittyen. Jos kaikki testit menevät onnistuneesti läpi, watchdog käynnistää ajastimen uudelleen (eng. *kick the watchdog*). Jos taas osa testeistä ei mene läpi, niiden suorituksessa ilmenee ongelmia tai järjestelmä ei vastaa, ajastin laskee nollaan ja aiheuttaa poikkeustilanteen. Tällöin järjestelmä voi suorittaa korjaavia toimenpiteitä, kuten käynnistää itsensä uudelleen (Crawford 2016).

Lokianalyysin yhteydessä watchdogilla tarkoitetaan useimmiten skriptiä, joka tarkkailee tiettyjä lokitiedostoja. Todd Atkinsin *swatchdog* on eräs tälläinen yleisessä käytössä oleva skripti (Atkins 2017). Jos watchdog havaitsee jotakin normaalista poikkeavaa, se suorittaa määritellyn toiminnon, esimerkiksi lähettää järjestelmänvalvojalle sähköpostia. Tyypillinen lokeja monitoroiva watchdog etsii lokivirrasta tiettyjä avainsanoja tai kielellisiä rakenteita. Apuna voidaan käyttää mm. säännöllisiä lausekkeita. Watchdog voi myös itse tarkkailla järjestelmän tila- tai sensoritietoja, kuten levyjärjestelmien täyttymistä tai komponenttien lämpötilaa ja ilmoittaa näistä järjestelmänvalvojalle.

Watchdog-skriptin toimintaperiaate on helppo ymmärtää käsitteellisellä tasolla, eikä ohjelmallinen toteutus ole useimmiten erityisen monimutkaista. Hyvän ja kattavan watchdog-skriptin luominen vaatii kuitenkin perehtymistä paitsi itse järjestelmän, myös sen komponenttien ja siinä ajettavien ohjelmistojen lokinkirjoitusmetodologiaan. Järjestelmänvalvojan tulee siis ymmärtää, millaisia viestejä järjestelmä tyypillisesti lähettää ja mitä niistä voidaan tulkita.

Ongelmallista on myös lokinkirjoitusjärjestelmien tyypillinen monisanaisuus. Monet järjestelmän virheet saattavat aiheuttaa lokeihin suuren määrän viestejä lyhyessä ajassa. Jotkin protokollaimplementaatiot, kuten syslog-ng voivat tarjota tähän älykkäämpiä suodatusominaisuuksia (*The syslog-ng Open Source Edition 3.9 Administrator Guide* 2017). Yksinkertaisemmat versioinnit saattavat kuitenkin päästää paljonkin viestejä läpi watchdog-skriptin vaivaksi.

Lokianalyysista puhuttaessa on huomioitava, että watchdog-mallia noudattavat työkalut eivät aina tarjoa viestien tulkintaan liittyviä ominaisuuksia. Watchdog ilmoittaa järjestelmän tapahtumista, muttei välttämättä pyri selvittämään niiden merkitystä tai analysoimaan tapahtumien välisiä korrelaatioita. Yksinkertainen watchdog-skripti ei siis aina ole varsinainen lokianalyysityökalu. Monet lokianalyysityökalut kuitenkin hyödyntävät watchdog-mallia jossakin muodossa. Watchdogin keräämät ja suodattamat viestit voidaan esimerkiksi käsitellä edelleen tarkempaa korrelaatioanalyysia ja poikkeamien havainnointia tarjoavilla työkaluilla.

3.2.2 Tiedonlouhinta

Tiedonlouhinta eli *data mining* on suurten tietomassojen käsittelyä halutun tiedon löytämiseksi. Tiedonlouhinnan keinoin voidaan esimerkiksi havaita tietomassassa ilmeneviä tyypillisiä käyttäytymismalleja, havaita poikkeavuuksia ja tarkastella tapahtumien suhteita toisiinsa. Suurille tietomäärille kehitellyt käsittely- ja analyysitekniikat ovat erinomaisesti sovellettavissa lokianalyysin parissa tyypillisesti ilmenevien ongelmien ratkaisemiseen.

Tietojärjestelmät ovat perusluonteeltaan jatkuvasti muuttuvia ja olosuhteiden mukaan eläviä. Siksi järjestelmän kehittämishetkellä luotu tyypillisen käyttäytymismalli ei välttämättä päde enää tulevaisuudessa. Tehokas lokianalyysimenetelmä voidaan toteuttaa dynaamisesti järjestelmän käyttäytymistä oppivana. Yamanishi ja Maruyama (2005) esittelevät verkkoongelmien havaitsemiseen suunnitellun dynaamisen tiedonlouhintatekniikan, joka pyrkii mallintamaan syslogin käyttäytymistä. Menetelmä perustuu Markovin piilomallien (HMM-mallit) yhdistelmään. Oppivan algoritmin avulla malli mukautuu järjestelmän käyttäytymiseen. Kerätystä lokitiedostosta laskettua poikkeusarvoa verrataan mukautuvaan rajapyykkiin. Rajapyykin ylittävät sessiot tulkitaan poikkeuksellisiksi. Menetelmä siis kykenee mukautumaan dynaamisesti muuttuvaan järjestelmään ja havaitsemaan mahdollisia uusia toimintamalleja.

Jotkin lokianalyysimenetelmät yhdistävät lokianalyysiin muita tarkentavia tekniikoita. Xu ym. (2009) ovat kehittäneet lähdekoodin analyysia lokianalyysin osana hyödyntävän menetelmän. Lähdekoodia tutkimalla muodostetaan rakenteellinen malli kaikista mahdollisista lokiviesteistä, jonka perusteella ohjelman kirjoittamista oikeista viesteistä voidaan poimia olennaiset viestitiedot ja muuttujat. Näistä tiedoista muodostetaan ominaisuusvektorit, joita voidaan käsitellä poikkeamien havaitsemiseksi itseoppivaan algoritmiin perustuvalla PCA-metodilla *(Principal Component Analysis)*. Lähdekoodia analysoimalla pyritään löytämään juuri kyseiselle järjestelmälle sopiva lokimalli, jonka pohjalta oppiva algoritmi voi työskennellä.

Kuten aiemmin on todettu, tietojärjestelmissä ilmenevät ongelmat voivat olla kytköksissä useisiin järjestelmän osiin. Jossakin järjestelmän laitteessa ilmennyt kriittinen virhe saattaa aiheuttaa ongelmia muissa laitteissa, tai poikkeustilanne saattaa syntyä useamman laitteen virheellisen toiminnan seurauksena. Yamanishi ja Maruyama (2005) esittelemä dynaaminen

malli ottaa kantaa myös tällaisten laitteiden välisten korrelaatioiden löytämiseen. Verkon eri laitteista kerätyt tiedot käsitellään aiemmin kuvatulla tavalla HMM-mallien yhdistelmäksi, ja näistä havaitut poikkeukselliset sessiot synknoroidaan yhdeksi suureksi lokitiedostoksi. Yhdistettyä lokitiedostoa voidaan nyt käsitellä samalla tavalla kuin yksittäisen laitteen lokitiedostoa, ja siitä voidaan nähdä verkon laitteiden välille muodostuneita korrelaatioita.

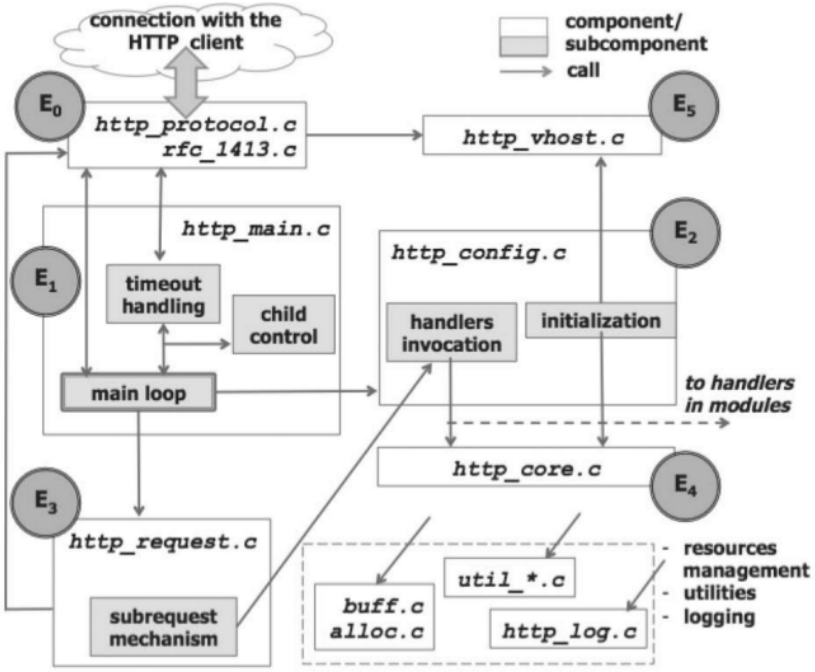
Tiedonlouhintaan pohjautuvat lokianalyysimenetelmät pystyvät parhaimmillaan havaitsemaan suuresta tietomäärästä tapahtumia, niiden välisiä korrelaatioita ja erilaisia poikkeamia hyvinkin tehokkaasti. Ne eivät kuitenkaan välttämättä kykene selittämään, mitä löydetyille ongelmille voisi tehdä (Oliner, Ganapathi ja Xu 2011). Tulosten lopullinen tulkitseminen ja niiden perusteella suoritettavien korjaustoimenpiteiden päättäminen jää ammattitaitoisen järjestelmänvalvojan tehtäväksi. Tiedonlouhintamenetelmien vaatiman laskentatehon ja siitä muodostuvien suoritusaikarajoitteiden vaikutus tulee myös ottaa huomioon.

3.2.3 Sääntöpohjainen menetelmä

Tähän mennessä tässä luvussa käsitellyt menetelmät ovat keskittyneet lokien analysointiin viestin kirjoitushetkellä tai sen jälkeen. Tällaisella lähestymistavalla haetaan useimmiten yhteensopivuutta olemassaolevien järjestelmien kanssa. Kyse voi olla myös alustariippumattomuuden tavoittelusta. Kaupallista lokianalyysiohjelmistoa suunniteltaessa on olennaista vastata asiakkaiden tarpeisiin tukemalla näiden käyttämiä käyttöjärjestelmiä ja protokollaimplementaatioita. Olemassaolevien ratkaisujen päälle rakentamisessa on kuitenkin riskinsä. Lokianalyysin toimivuus järjestelmänvalvonnan apuvälineenä perustuu olettamukseen, että järjestelmän kirjaamat lokit pystyvät luotettavasti kuvaamaan järjestelmän tilaa. Todellisuudessa näin ei kuitenkaan aina ole.

Cinque ym. (2010) havaitsivat tapaustutkimuksia vertailemalla, että noin 6 virhettä 10:stä ei jätä lainkaan jälkiä järjestelmän lokeihin. On siis tarpeen tarkastella ohjelmiston lokinkirjoitusmetodien toimivuutta jo järjestelmän suunnitteluvaiheessa. Ratkaisuna tähän Cinque, Cotroneo ja Pecchia (2013) ehdottavat sääntöpohjaiseksi lokinkirjoitusmenetelmäksi *(rule-based logging)* kutsuttua menetelmää. Sääntöpohjaisessa menetelmässä kehitettävästä tietojärjestelmästä muodostetaan korkean tason malli, josta ilmenevät järjestelmän kohteet (*en-*

tity) ja niiden keskinäinen vuorovaikutus. Kohteille määritellään mallin perusteella lokinkirjoitussäännöt *(logging rules)*, joiden perusteella ohjelmakoodiin voidaan automaattisesti lisätä tiettyjä virhetyyppejä huomioivaa lokinkirjoituskoodia. Säännöt siis määrittelevät, milloin ja mitä lokiin kirjataan.



Kuvio 3. Apache Web Server-ohjelmiston moduuleista ja vuorovaikutuksesta muodostettu kohdemalli. (Cinque, Cotroneo ja Pecchia 2013)

Kuviossa 3 E_0 - E_1 merkitsevät kyseisen järjestelmän kohteita. Kohteet tarjoavat muille kohteille ja järjestelmän ulkoisille osille palveluita (*service*). Palvelut voivat sisältää kohteen sisäisiä toimintoja, joiden tulokset luovutetaan kohdetta kutsuneelle taholle. Palvelu voi myös siirtää tarvittaessa kontrollin toiselle kohteelle tai ulkoiselle osalle, jolloin tapahtuu vuorovaikutus *(interaction)*. Sääntöpohjaisessa menetelmässä pyritään havaitsemaan palveluissa ja vuorovaikutuksissa tapahtuvat virheet systemaattisesti muodostetun virhemallin kautta, jossa on neljä eri virhekategoriaa:

- Palveluvirhe *(SER, service error)*. Palvelua ei saada suoritettua loppuun lainkaan.
- Palveluvalitus *(CMP, service complaint)*. Palvelu on päättynyt epäonnistuneesti.

- Vuorovaikutusvirhe *(IER, interaction error)*. Kohteen aloittamaa vuorovaikutusta ei saada päätettyä onnistuneesti, eli kohteen kutsuma toinen kohde tai järjestelmän osa ei palauta kontrollia kohteelle.
- Kaatumisvirhe *(CER, crash error)*. Kohteessa on tapahtunut odottamaton virhe ja sen suoritus keskeytyy.

Virhetyyppien lisäksi sääntöpohjainen menetelmä kirjaa kohteiden käynnistys- ja sammutustapahtumat, joiden perusteella voidaan havaita esimerkiksi järjestelmän odottamattomia uudelleenkäynnistyksiä. Tämä muodostettu sääntömalli voidaan lisätä automatisoidusti koodiparserin avulla kohteiden lähdekoodiin. (Cinque, Cotroneo ja Pecchia 2013)

Cinque, Cotroneo ja Pecchia (2013) esittelemä sääntöpohjainen menetelmä ottaa järjestelmän rakenteen huomioon, mikä lisää lokien luotettavuutta. Se kykenee näkemään järjestelmän kokonaisuutena ja havainnoimaan ohjelmiston kohteiden välisessä vuorovaikutuksessa tapahtuvia virheitä, mikä tuottaa vaikeuksia perinteisille lokinkirjoitusjärjestelmille. Menetelmä kuitenkin vaatii toimiakseen järjestelmästä muodostetun kohdemallin. Jos tällaista mallia ei ole olemassa tai se ei ole ajan tasalla, joudutaan turvautumaan takaisinmallintamiseen. Tämä saattaa hankaloittaa sääntöpohjaisen menetelmän sovittamista jo olemassaolevaan järjestelmään.

4 Lokianalyysi järjestelmän vianetsinnän kannalta

Vaikka lokien tarkastelu kuuluu järjestelmänvalvojan arkeen normaalitilanteessakin, poikkeustilan saapuessa niiden merkitys korostuu entisestään. Lokien sisältämä informaatio muuttuu silloin eräänlaiseksi todisteiden ketjuksi, jota seuraamalla pyritään ongelman jäljille. Tässä luvussa pohditaan järjestelmän vianetsintäprosessia kokonaisuutena ja pyritään löytämään lokianalyysin rooli osana sitä.

Lokianalyysin käyttöönottoa suunniteltaessa on tarpeen hahmottaa, millaisia ongelmia järjestelmät tyypillisesti kohtaavat. Jiang ym. (2009) keräämässä aineistossa tyypillisimpiä asiakasongelmia ovat laitteistoviat (40% tutkituista tapauksista) ja väärin konfiguroiduista järjestelmistä johtuvat ongelmat (21% tutkituista tapauksista). Konfiguraatio-ongelmien verrattain suuri prosentuaalinen osuus on kiinnostava ja jo itsessään tutkimisen arvoinen. Mitkä seikat vaikuttavat konfiguraatio-ongelmien syntyyn?

Osa ongelmista osoittautuu inhimillisistä virheistä johtuviksi. Joskus kyse voi taas olla järjestelmäpäivitysten yhteydessä syntyneistä yhteensopivuusongelmista, esimerkiksi uuteen ohjelmistoversioon siirryttäessä aiemmat konfiguraatiotiedostot saattavat lakata toimimasta. Puutteelliset laitteiston tai ohjelmiston toimittajalta saadut käyttöohjeet voivat myöskin aiheuttaa merkittäviä väärinkäsityksiä. Monet edellä kuvatun kaltaisista tilanteista juontavat samankaltaisille juurille: kommunikaatio järjestelmänvalvojan ja muiden osapuolien välillä ei ole syystä tai toisesta toiminut halutulla tavalla.

On huomioitava, että lokeihin kirjattu tieto kertoo järjestelmän tilanteesta nimenomaan teknisen laitteiston näkökulmasta. Pelkästään sen perusteella ei kuitenkaan voida arvioida järjestelmää käyttävien ja huoltavien ihmisten ajatusketjuja. Lokeista voidaan mahdollisesti nähdä, kuka järjestelmää on käyttänyt ja mitä hän on siellä tehnyt, mutta nämä tiedot vaativat yleensä tuekseen lisäselvitystä käyttäjältä itseltään. Järjestelmänvalvojan tapauksessa tämä voi tarkoittaa esimerkiksi henkilökohtaista päiväkirjaa tai muuta dokumenttia, johon kirjataan vaihe vaiheelta, miksi ja miten järjestelmälle on tehty jotakin. Jos järjestelmänvalvojia on useampia, yhteisen järjestelmän muutoksista kertovan *changelog*-dokumentin ylläpitäminen on entistä tärkeämpää. Laajoissa järjestelmissä järjestelmänvalvojilla voi olla eriäviä

vastuualueita, eivätkä kaikki välttämättä ole toistensa vastuualueiden asiantuntijoita.

Käyttäjän ilmoittaessa järjestelmänvalvojalle ongelmista on myös hyvä toivoa vastaavanlaista selvitystä ongelmaan johtaneista toimenpiteistä. Kun tiedetään, miten ongelmatilanteeseen on päädytty, muodostuu tulkinnallinen viitekehys, jonka puitteissa lokit ja lokianalyysityökalut pääsevät todella näyttämään kyntensä. Työkalujen löytämiä poikkeavuuksia voidaan nyt tulkita ongelman luonne huomioiden. Kommunikaatio muodostaa siis monessa mielessä vianetsintäprosessin pohjan. Monet lokianalyysityökalupaketit, kuten Splunk tarjoavat visualisointiominaisuuksia, joiden avulla on mahdollista kuvata järjestelmän eri osien tilaa graafisessa muodossa (Carasso 2012). Järjestelmänvalvoja voi käyttää näitä visualisointeja apuna esimerkiksi ei-teknisen henkilökunnan kanssa kommunikoidessa. Kun järjestelmän käyttämistä resursseista voidaan automaattisesti muodostaa seikkaperäinen kaavio, voidaan sitä käyttää apuna esimerkiksi järjestelmäuudistusten suunnittelussa tai tulevien muutosten budjetoinnissa.

Kiireellisessä tilanteessa lokianalyysi joutuu tulikokeeseen. Yamanishi ja Maruyama (2005) havaitsivat verkkolaitteiden vianetsintää käsittelevässä tutkimuksessa, että samassa verkossa erillisissä solmuissa sijaitsevat palvelimet korreloivat voimakkaasti keskenään poikkeustilanteissa. Tästä voidaan päätellä, että yhdessä palvelimessa tai solmussa tapahtunut virhe saattaa levitä suurella todennäköisyydellä verkon muihin osiin. Näin ollen järjestelmän kriittisten osien ongelmien pikainen paikantaminen on ensisijaisen tärkeää. Jiang ym. (2009) havaitsivat lokitiedoilla varustettujen asiakasvikailmoitusten selviävän huomattavasti nopeammin (16% - 88%) kuin ilman lokeja lähetettyjen vikailmoitusten, mikä puhuu vahvasti lokien puolesta.

Lokianalyysista voidaan tehdä silmäpari, joka katseellaan etsii avainkohtia tietovirrasta. Järjestelmänvalvoja tarttuu näihin avainkohtiin ja selvittää kokemuksensa perusteella, mitä ne varsinaisesti tarkoittavat. Tulevaisuudessa analyysimenetelmät voivat mahdollisesti toimia nykytilannetta itsenäisemmin ja tehdä päätelmiä järjestelmän vaatimista huoltotoimenpiteistä. Vielä on kuitenkin aikaista sanoa, miten nopeasti tähän vaiheeseen päästään.

5 Yhteenveto

Tutkielman tavoitteena oli tarkastella Syslog-protokollan toimintaa ja sen tulosteen analysointia lokianalyysiksi kutsuttujen menetelmien avulla. Lisäksi pyrittiin pohtimaan lokianalyysin roolia osana järjestelmän vianetsintäprosessia.

Syslog on perusmäärittelyltään riisuttu, mutta tehokas ja monikäyttöinen järjestelmän tapahtumien kirjaamiseen ja käsittelyyn tarkoitettu protokolla. Gerhards (2009) muodostamaa protokollamäärittelyä on laajennettu implementaatiotasolla monipuolisemmaksi. Syslogin käyttämä raakatekstiformaatti on käsiteltävissä helposti standardityökaluilla, mutta suurten tiedostojen käsittely voi olla ajoittain hidasta ja ongelmallista. Tästä syystä joskus voi olla tarpeen käyttää jotakin vaihtoehtoista lokinkirjoitusratkaisua, kuten systemd-järjestelmäpaketin osana toimitettavaa *journald*:tä (Schäfer 2016).

Lokianalyysin teoreettisesta taustasta pyrittiin muodostamaan kolme erilaisiin asioihin painottavaa suuntausta. Watchdog-malli ja avainsanapohjaiset skriptit tarjoavat seikkaperäisen tavan monitoroida järjestelmän tilaa poikkeavuuksien varalta, mutta vaativat samalla syvällistä perehtymistä käsiteltävään järjestelmään. Järjestelmänvalvojan tulee siis ymmärtää, millaisilla viesteillä järjestelmä tyypillisesti kommunikoi. Tiedonlouhintaan perustuvat menetelmät, kuten Yamanishi ja Maruyama (2005) ja Xu ym. (2009) puolestaan tarjoavat suurten tietomäärien käsittelyyn sopivia, järjestelmän käyttäytymiseen mukautuvia toimintatapoja. Ne eivät kuitenkaan välttämättä pysty selittämään, mitä havaituille ongelmille voisi tehdä (Oliner, Ganapathi ja Xu 2011). Cinque, Cotroneo ja Pecchia (2013) esittelemä sääntöpohjainen menetelmä puolestaan pyrki parantamaan lokinkirjoitusta jo ohjelmiston suunnittelutasolla luomalla sääntömallin, jonka perusteella ohjelmakoodiin voidaan lisätä oikeisiin kohtiin lokinkirjoituskoodia. Menetelmä vaatii kuitenkin tuekseen järjestelmästä muodostetun korkean tason mallin, joka saatetaan olemassaolevaa järjestelmää käsiteltäessä joutua luomaan takaisinmallintamisen keinoin.

Lokianalyysityökalut oikein käytettynä ja konfiguroituna tukevat vianetsintäprosessia. Järjestelmänvalvojalta vaaditaan silti laajaa kokemuspohjaa ja perehtyneisyyttä oman järjestelmän tarpeisiin. Lokien kirjoittamista ei myöskään kannata jättää ainoastaan järjestelmän

huoleksi, vaan järjestelmänvalvojan kannattaa pitää yllä omaa ylläpitodokumentaatiota, josta järjestelmään tehdyt muutokset löytyvät kätevästi. Näin on mahdollista saada selville, milloin, miksi ja miten jotakin on tehty. Lokianalyysista eivät hyödy ainoastaan järjestelmänvalvojat, vaan sen menetelmillä voidaan parhaimmillaan tukea koko organisaatiota. Lokianalyysityökalut voivat esimerkiksi visualisoida monimutkaisia asiakokonaisuuksia helposti sisäistettävään muotoon, jota voidaan käyttää apuna vaikkapa muun kuin teknisen henkilökunnan kanssa kommunikoidessa.

Tämä tutkielma on käsitellyt Syslog-protokollaa ja lokianalyysin teoriaa käytännön kannalta havainnollistaen. Lokianalyysi aihepiirinä on runsaasti tutkittu ja jatkuvasti kehittyvä, mutta tutkimattomampia tulokulmia on mahdollista löytää. Luvussa 4 käsiteltyä kommunikaation roolia järjestelmänvalvonnassa on alan tutkimuksissa tarkasteltu melko vähäisesti, ja siihen perehtyminen voisi tarjota kiehtovaa materiaalia useille eri tieteenaloille. Lokianalyysia tarkastelevia laajoja aineistotutkimuksia, kuten Jiang ym. (2009) toteuttivat tallennusjärjestelmien kohdalla, on niin ikään tieteen kentällä verrattain vähän. On kiinnostavaa seurata, millaisia lähestymistapoja tulevaisuus tuo tullessaan.

Lähteet

Atkins, Todd. 2017. *Simple Log Watcher.* `https://sourceforge.net/projects/swatch/`.

Carasso, David. 2012. "Exploring splunk". *published by CITO Research, New York, USA:* 978–.

Cinque, Marcello, Domenico Cotroneo, Roberto Natella ja Antonio Pecchia. 2010. "Assessing and improving the effectiveness of logs for the analysis of software faults". Teoksessa *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on,* 457–466. IEEE.

Cinque, Marcello, Domenico Cotroneo ja Antonio Pecchia. 2013. "Event logs for the analysis of software failures: A rule-based approach". *IEEE Transactions on Software Engineering* 39 (6): 806–821.

Crawford, Paul S. 2016. *Linux Watchdog Daemon - Overview.* `http://www.sat.dundee.ac.uk/psc/watchdog/watchdog-background.html`.

Gerhards, Rainer. 2007. *why does the world need another syslogd? (aka rsyslog vs. syslog-ng).* `http://blog.gerhards.net/2007/08/why-does-world-need-another-syslogd.html`.

———. 2009. "The syslog protocol", Request for Comments.

Jiang, Weihang, Chongfeng Hu, Shankar Pasupathy, Arkady Kanevsky, Zhenmin Li ja Yuanyuan Zhou. 2009. "Understanding Customer Problem Troubleshooting from Storage System Logs." Teoksessa *FAST,* 9:43–56.

Lonvick, Chris. 2001. "The BSD syslog protocol", Request for Comments.

Nemeth, Evi, Garth Snyder, Trent R Hein ja Ben Whaley. 2011. *Unix and Linux System Administration Handbook.* Neljäs painos. Prentice Hall.

Oliner, Adam, Archana Ganapathi ja Wei Xu. 2011. "Advances and challenges in log analysis". *Queue* 9 (12): 30.

Rsyslog 8.26.0 documentation. 2017. http://www.rsyslog.com/doc/v8-stable/.

Schulze, Joey. 2014. *sysklogd.* http://www.infodrom.org/projects/sysklogd/.

Schäfer, Jorgen. 2016. *Why Journald?* https://www.loggly.com/blog/why-journald/.

The syslog-ng Open Source Edition 3.9 Administrator Guide. 2017. https://www.balabit.com/documents/syslog-ng-ose-latest-guides/en/syslog-ng-ose-guide-admin/html/index.html.

Vixie, Paul A, ja Frederick M Avolio. 2002. *Sendmail: theory and practice.* Elsevier.

Xu, Wei, Ling Huang, Armando Fox, David Patterson ja Michael I Jordan. 2009. "Detecting large-scale system problems by mining console logs". Teoksessa *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles,* 117–132. ACM.

Yamanishi, Kenji, ja Yuko Maruyama. 2005. "Dynamic syslog mining for network failure monitoring". Teoksessa *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining,* 499–508. ACM.