

Apache Spot havaitsemassa verkkohyökkäyksiä

Osmo Suomalainen

Opinnäytetyö

Toukokuu 2019

Tekniikan ja liikenteen ala

Insinööri (AMK), Tieto- ja viestintätekniikan tutkinto-ohjelma

Tekijä(t) Suomalainen, Osmo	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Toukokuu 2019
	Sivumäärä 61	Julkaisun kieli Suomi
		Verkojulkaisulupa myönnetty: Kyllä
Työn nimi Apache Spot havaitsemassa verkkohyökkäyksiä		
Tutkinto-ohjelma Tieto- ja viestintätekniikka, Tietoverkkotekniikka		
Työn ohjaaja(t) Tero Kokkonen, Mika Rantonen		
Toimeksiantaja(t) Nixu Oyj ja JYVSECTEC		
<p>Tiivistelmä</p> <p>Opinnäytetyön toimeksiantajina toimi Nixu Oyj sekä JYVSECTEC. Tavoitteena oli saada näille tahoille tietoa Apache Spotin toimivuudesta kyberturvallisuuden osana. Kyberturvallisuus on nykypäivänä erittäin tärkeä konsepti yrityksille ja julkisen vallan toiminnalle, sillä melkein kaikki mitä teemme on Internetissä. Datamäärän kasvaessa kovaa vauhtia täytyy hyökkäysten havainnoinnin sekä estämisen olla nopeaa ja tehokasta. Opinnäytetyössä tutkittiin, miten Apache Spot:a voidaan käyttää netflow’n poikkeavuuksien havainnointiin.</p> <p>Opinnäytetyön teoriaosuudessa käydään läpi Apache Spotin infrastruktuuria, sen eri palveluita, joita tämän tuotteen pystyttämiseen tarvittiin, sekä kyberturvallisuuteen liittyviä käsitteitä.</p> <p>Käytännön vaiheessa Apache Spot asennettiin Cloudera klusterille RGCE-ympäristöön. Klusteri koostui viidestä Centos-palvelimesta, joilla kaikilla oli oma erillinen roolinsa tuotteen toiminnassa. Tuote saatiin asennettua, mutta johtuen tuotteen kehitysvaiheesta, näitä toiminnallisuuksia ei saatu toimimaan.</p> <p>Apache Spot on vielä kehitysvaiheessa, sillä tuote oli hankala asentaa. Tuotteen versiohallinta ja dokumentaatio olivat osin puutteellisia. Kun nämä asiat saadaan korjattua, tuotteella on varmasti käyttöä kyberturvallisuuden osana.</p>		
Avainsanat (asiasanat) Apache Spot, kyberturvallisuus, verkon poikkeavuudet, Hadoop		
Muut tiedot (Salassa pidettävät liitteet)		

Author(s) Suomalainen, Osmo	Type of publication Bachelor's thesis	Date May 2019
		Language of publication: Finnish
	Number of pages 61	Permission for web publication: Yes
Title of publication Using Apache Spot for detecting network attacks		
Degree programme Information and Communication Technology, Data Network Engineering		
Supervisor(s) Kokkonen Tero, Rantonen Mika		
Assigned by Nixu Oyj and JYVSECTEC		
<p>Abstract</p> <p>The bachelor's thesis was assigned by Nixu Oyj and JYVSECTEC. The main goal was to examine if Apache Spot could be of use in cyber defense. Cyber security is an important topic to companies and the public now days, since almost everything people do is in the Internet. The amount of data that on sees means that there must be efficient and fast ways to digest it as well as to detect and prevent cyber-attacks. The main point was to figure out, how Apache Spot can be used with netflow to detect network anomalies.</p> <p>The theory part describes the architecture of Apache Spot including the services that are needed for it to be installed. It also covers the terminology used in cyber security.</p> <p>Apache Spot was installed on a Cloudera cluster in the RGCE environment. The cluster consisted of five Centos servers, which all had their own role. The product was installed however, due to it still being in development stage, the functionalities were not working.</p> <p>For the time being Apache Spot is still under development, which could be seen with the difficulties in its installation. Version control and documentation are now still lacking, however when these are fixed, Apache Spot will certainly be used in cyber defense.</p>		
Keywords/tags (subjects) Apache Spot, Cyber security, Network anomaly, Hadoop		
Miscellaneous (Confidential information)		

Sisältö

1	Työn lähtökohdat	7
1.1	Yleistä	7
1.2	Toimeksiantaja	7
1.3	Työn tavoitteet ja vaatimusmäärittely	8
2	Kyberhyökkäys	8
2.1	Yleistä	8
2.2	Kyberhyökkäysten kategorisointi	9
3	Kyberturvallisuus.....	10
3.1	Yleistä	10
3.2	Passiivinen kyberturvallisuus	11
3.3	Aktiivinen kyberturvallisuus	13
4	SIEM/Flowdata.....	14
4.1	SIEM.....	14
4.2	Flow data	15
5	Apache Spot	17
5.1	Yleistä	17
5.2	Hadoop	18
5.2.1	Yleistä.....	18
5.2.2	Hadoop Distributed Filsesystem.....	19
5.2.3	HIVE	20
5.2.4	IMPALA	21
5.2.5	KAFKA.....	21
5.2.6	SPARK.....	24
5.2.7	YARN	26
5.2.8	Zookeeper	28
5.3	Apache-Spot rakenne	30
5.3.1	Yleistä.....	30
5.3.2	Apache Spot-ingest.....	31
5.3.3	Apache Spot-ML	33
5.3.4	Apache Spot-OA.....	34

5.3.5	Apache Spot-setup	34
6	Suunnitelma	35
6.1	Yleistä	35
6.2	Ympäristö.....	35
6.3	Hyökkäykset.....	35
7	Toteutus.....	36
7.1	Hadoopin pystyttäminen.....	36
7.2	Apache-Spotin pystyttäminen.....	38
7.2.1	Ingestin konfiguraatio.....	41
7.2.2	ML:n konfiguraatio	43
7.2.3	OA:n konfiguratio	43
8	Testaus ja tulokset.....	45
8.1	Testaus.....	45
8.2	Tulokset	45
9	Pohdinta ja johtopäätökset.....	47
9.1	Johtopäätökset	47
9.2	Pohdinta	47
	Lähteet	50
	Liitteet.....	54
	Liite 1. Flow'n matriisi (Supsicious Connects Analysis n.d.).....	54
	Liite 2 DNS-matriisi (Supsicious Connects Analysis n.d.)	56
	Liite 3. Proxyn matriisi (Supsicious Connects Analysis n.d.)	58

Kuviot

Kuvio 1. HIDS ja NIDS	12
Kuvio 2. Tietoturva yrityksissä	14
Kuvio 3. ICMP Netflow	16
Kuvio 4. UDP Netflow	16
Kuvio 5. TCP Netflow	17
Kuvio 6. MapReduce	19
Kuvio 7. Kafka.....	23
Kuvio 8. Kafka topic	23
Kuvio 9. Spark	26
Kuvio 10. YARN	27
Kuvio 11. Zookeeper hierarkkia	29
Kuvio 12. Zookeeper	30
Kuvio 13. Apache Spot Service Layout.....	31
Kuvio 15. Apache Spot ingest	33
Kuvio 16. SELinux ja firewallld	36
Kuvio 17. /etc/hosts.....	37
Kuvio 18. Cloudera roles	37
Kuvio 19. Toteutettu service layout	38
Kuvio 20. ID RSA public	39
Kuvio 21. Spot.conf	40
Kuvio 22. HDFS-setup varmistus.....	41
Kuvio 23. Ingest riippuvuuksien versiot.....	42
Kuvio 24. Ingestion konfiguraatiotiedosto	43
Kuvio 25. engine impala.....	44
Kuvio 26. Web-palvelin käynnistetty	44
Kuvio 27. Apache Spot lopputulos.....	45
Kuvio 28. ML-dokumentaatio	46

Lyhenteet

ACL	Access Control List
AM	AppMaster
API	Application Programming Interface
CIRT	Computer Incident Responder Team
DBMS	Database Management Systems
DNS	Domain Name System
HDFS	Hadoop Distributed Filsesystem
HQL	Hive Query Language
ICMP	Internet Control Message Protocol
IP	Internet Protocol
JDBC	Java Database Connection
LDA	Latent Dirichlet Allocation
LDAP	Lightweight Directory Access Protocol
LTS	Long Term Support
ML	Machine learning
NPM	Node Package Manager

NSM	Network Security Monitoring
NixuCDC	Nixu Cyber Defense Center
OA	Operational Analytics
ODBC	Open Database Connection
PCAP	Packet capture
RDD	Resilient Distributed Dataset
RGCE	Realistic Global Cyber Environment
Sbt	Simple build tool
SIEM	Security Information and Event Management
SOC	Security Operations Center
SQL	Structured Query Language
SYN	Synchronization
SYN-ACK	Synchronization-Acknowledgement
Sudo	Super User Privileges
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface

URI

Uniform Resource Identifier

YARN

Yet Another Resource Negotiator

1 Työn lähtökohdat

1.1 Yleistä

Opinnäytetyössä keskityttiin ajankohtaiseen kyberturvallisuuden osa-alueeseen eli puolustuksen sekä havainnoinnin parantamiseen. Työn tarkoituksena oli tutkia, miten netflow'n sekä pakettien analysoinnista voidaan saada tehokas työkalu tietoverkkoja puolustaville tahoille. Palvelu Apache Spot otettiin työnkeskipisteeksi. Tavoitteena oli tutkia, miten kyseinen palvelu parantaa sekä edistää kyberturvallisuutta yrityksille. Tehtävänä oli asentaa kyseinen palvelu sekä testata sen toiminnallisuuksia, kun suojattavaan kohteeseen kohdistuu hyökkäys, sekä miten palvelu toimii normaaliolosuhteissa. Opinnäytetyö on tehty laadullisen tutkimusmenetelmän pohjalta.

1.2 Toimeksiantaja

Työn toimeksiantajina toimi Nixu Oyj ja JYVSECTEC. Nixu Oyj on tietoturvakonsultointiin erikoistunut asiantuntijayritys, joka aloitti Suomessa ja on sittemmin levinnyt Pohjoismaihin sekä muihin maihin. Nixu Oyj hoitaa asiakkaidensa tietoturvaa niin sisäisessä tietohallinnossa, sähköisessä liiketoiminnassa kuin teollisen internetin ratkaisualueilla. Esimerkki heidän tuotteistaan on Nixu Cyber Defense Center (NixuCDC). Tämä palvelu on Security Operations Center:in (SOC) tapainen, mutta laajennetumpi versio. (Historia n.d.)

JYVSECTEC on Jyväskylän ammattikorkeakoulun IT-instituuttiin kuuluva kyberturvallisuuden tutkimus-, koulutus- ja kehityskeskus. JYVSECTEC sai alkunsa projektina, joka käynnistettiin vastaamaan ajankohtaisia haasteita vastaan vuonna 2011. Projektin päämääränä oli saada johtava tutkimus-, kehitys- ja koulutuskeskus Suomeen, sekä tuoda yhteen kansallisia, että kansainvälisiä turvallisuusalan toimijoita sekä yrityksiä. JYVSECTEC kehitti ympäristön nimeltä Realistic Global Cyber Environment (RGCE), joka pyrkii mallintamaan oikeaa globaalia verkkoa palveluineen. RGCE:ssä voidaan tutkia sekä harjoitella kyberturvallisuuden parantamista. (Tietoa meistä n.d.)

1.3 Työn tavoitteet ja vaatimusmäärittely

Opinnäytetyön tavoitteena oli tarkastella ja tutkia Apache Spotin toimintaa ja kuinka tehokkaasti se voi netflow'n avulla havaita poikkeamia keskisuurten ja isojen yritysten verkkoliikenteessä.

Keskeiset kysymykset, joiden avulla tähän koitettiin vastata, olivat seuraavat:

- Onko tämä tehokas tapa havaita verkkohyökkäyksiä keskisuurissa ja isoissa yrityksissä?
- Pärjääkö Apache Spot maksullisille vaihtoehtoille?
- Mitä heikkouksia Apache Spotissa on?
- Kuinka Apache Spot voisi parantaa tietoturvaa?
- Mitä kehittämitarpeita Apache Spotissa on?
- Apache Spotin rooli kyberturvallisuudessa?
- Mitä olemassa olevia vaihtoehtoja Apache Spot voi korvata?
- Mikä on Apache Spotin skaalautuvuus isoihin ympäristöihin?
- Deployment arkkitehtuurit

Alustavien arvioiden mukaan Apache Spot tarvitsee paljon netflow'ta normaalitilanteessa, jotta se toimisi. Tuote on vielä kehitysvaiheessa. Tämän hetkiset maksulliset tuotteet ovat edellä, mutta Apache Spotin toimintaperiaatteet ovat monipuoliset ja toiminnot ovat riittävän nopeat, joten sillä hyvät mahdollisuudet kuroa tuo etumatka umpeen, kunhan sen kehittämiseen panostetaan.

2 Kyberhyökkäys

2.1 Yleistä

Kyberhyökkäyksillä tarkoitetaan erilaisia metodeja, joissa hyökkääjä pyrkii aiheuttamaan tuhoa kohteen infrastruktuuriin (What Are the Most Common Cyberattacks?

n.d). Nykyään hyökkääjät tekevät näillä töillään myös tuottoa, esim. Ransomwarella, eli kohteen laitteisiin asennetaan ohjelmia, jotka kryptaavat kohteen tiedostot, ja vain maksamalla hyökkääjälle nämä tiedostot saadaan auki.

2.2 Kyberhyökkäysten kategorisointi

Kyberhyökkäykset voidaan jakaa neljään pääkategoriaan, jotka ovat Denial Of Service (DoS), Remote to User Attacks (R2L), User to Root (U2R) ja Probing. Probingissa hyökkääjän tarkoituksena on tutkia kohteen verkkoa tai konetta löytääkseen niistä joko heikkouksia tai suoraan haavoittuvuuksia, joita pystyy käyttämään tulevilla hyökkäyksissä. (An implementation of intrusion - - 2012.)

Probingissa hyökkääjä ei kerää kaikkea mahdollista tietoa, vain tarpeellisen, koska hän pyrkii pitämään toimensa salassa. Mitä enemmän tietoa hän pyrkii hankkimaan, sitä suurempi todennäköisyys on, että hänen toimensa havaitaan. Hyökkääjä kerää tietoa mm. seuraavista asioista: (1) mitä kohteita verkosta on saavutettavissa, (2) mitä käyttöjärjestelmiä näissä on sekä (3) mitä palveluita näillä kohteilla on. Mitä kattavamman käsityksen hän saa kohteesta, sitä helpommaksi hänen hyökkäyksensä muuttu. (Cole 2009, 789.) Työkaluna probingissa voidaan käyttää esim. Network Mapperia (Nmap). Nmap on avoimen lähdekoodin työkalu, jonka avulla voidaan skannata kohdeverkko. Nmap:n avulla voidaan selvittää mitä koneita kyseisessä verkossa on, mitä palveluita laitteilla ylläpidetään sekä näiden käyttöjärjestelmät. Nmap sopii paitsi tietoturva-asiantuntijoille, myös verkonylläpitäjille, koska sen avulla voidaan pysyä ajantasalla omasta verkosta. (Chapter 15. Nmap Reference Guide n.d.)

DoS-hyökkäyksellä estetään tietyn palvelun saavuttaminen joko lähettämällä kyseiselle palvelulle todella paljon kyselyitä niin, että palvelun resurssit ehtyvät, tai tukkimalla kyseisen palvelun kaista. Molemmissa tapauksissa palvelun asiakkaat eivät saa omia kyselyitään läpi ja kyseinen palvelu on joko todella hidas tai saavuttamattomissa. Kaistan tukkimisessa hyödynnetään yleensä hajautettua bottiverkkoa. Tästä käytetään silloin nimitystä Distributed Denial of Service (DDoS) hyökkäys. (SebastianZ 2013.)

R2L-hyökkäyksessä pyritään esimerkiksi hyödyntämään huonosti konfiguroitua tietokoneen tietoturvasääntöjä, joiden avulla hyökkääjä pääsee kohteeseen käsiksi. Esimerkkejä hyökkäysvektoreista ovat ftp-write sekä guest (Effective Analysis on Remote to User - - 2014). Hyökkääjä voi hyödyntää R2L:ää, jonka avulla hän pääsee kohteen perustason käyttäjän tilille, josta hän voi aloittaa U2R hyökkäyksen.

U2R:ssa hyökkääjän tavoitteena on saada kohteen pääkäyttäjän oikeudet. Ensimmäinen askel on saada kohteesta perustason käyttäjän oikeudet ja tämän jälkeen pyrkiä eskaloimaan nämä haavoittuvuuksien avulla pääkäyttäjän tasolle. (An implementation of intrusion - - 2012.)

Käyttäjän eskalointi pääkäyttäjäksi voi tapahtua hyödyntäen huonosti konfiguroituja tietoturvasääntöjä, esim. koneelta voi löytyä tiedostoja, jotka sisältävät pääkäyttäjän tunnukset, huono salasanapolitiikka tai tietoturvan kannalta väärin konfiguroidut palvelut. Mikäli kuitenkin nämä asiat ovat kunnossa kyseisellä koneella, hyökkääjä jatkaa hyökkäystään käyttöjärjestelmän kerneliä vastaan. Mikäli hyökkääjä pystyy suorittamaan koodiaan kernel-tasolla hyökkäyksessään, hän pystyy ohittamaan kohteen tietoturvan. Hyvin laadittu ja tietoturva hankaloittaa hyökkääjän toimintaa, ellei hänellä ole arsenaalissaan bugia, joka ei ole yleisessä tiedossa. Nämä bugit kulkevat nimellä nollapäivähaavoittuvuus ja ne ovat arvokkaita hyökkääjille. (Privilege escalation n.d.)

3 Kyberturvallisuus

3.1 Yleistä

Kyberturvallisuudella tarkoitetaan suojautumista kyberhyökkäyksiltä, jotka kehittyvät joka päivä. Kyberturvallisuuden pyrkimys on estää ja havaita hyökkäyksiä, jotka kohdistuvat infrastruktuuriin tai muuhun suojattavaan kohteeseen. Kyberturvallisuuden on ajateltu olevan tarkoitettu enimmäkseen yrityksille, mutta se kohdistuu myös yksityishenkilöihin. Yksityishenkilöiden tulee olla valveutuneita siitä, että heitä voidaan käyttää kyberhyökkäyksissä hyökkäysvektoreina. Yhteiskunta siirtyy yhä riippuvaisemmaksi tietoverkoista. Tämän takia on yhä tärkeämpää, että käyttäjät ymmärtävät

perustason siitä, miten heidän tulisi suojella itseään. Kyberturvallisuudessa tulee ymmärtää molempien osapuolten pyrkimykset, tiedot sekä taidot. Kyberturvallisuutta voidaan ajatella kahtena osana: aktiivinen kyberturvallisuus ja passiivinen kyberturvallisuus.

3.2 Passiivinen kyberturvallisuus

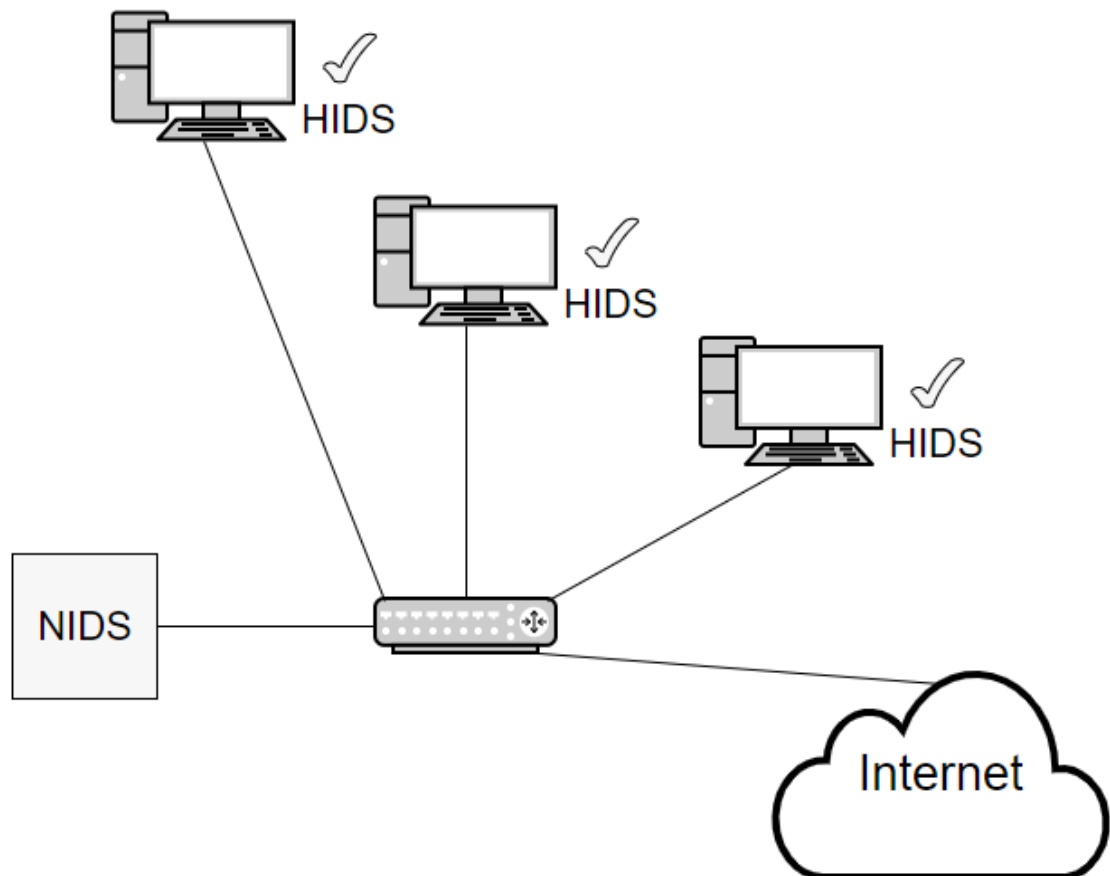
SANS organisaation mukaan passiivinen kyberturvallisuus on parhaimmillaan silloin kun infrastruktuuri on toteutettu järkevästi. Passiivinen puolustus keskittyy käyttämään hyökkääjien resursseja, jotta hyökkäys muuttuisi järjettömäksi edes toteuttaa. Tämä voidaan saavuttaa esimerkiksi pitämällä tietoturva päivitettyinä. Tällöin normaalit haittaohjelmat eivät välttämättä tee tuhoa ja hyökkääjä tuhlaa hänelle tärkeää resurssia eli aikaa. Työkaluja, joita passiivisessa kyberturvallisuudessa käytetään, ovat palomuurit, automaattiset virusskannaukset, Intrusion Detection System (IDS) sekä muut työkalut, jotka edistävät tietoturvaa eivätkä tarvitse kokoaikaista seuranta, vaan ne toimivat taustalla. (The Sliding Scale of Cyber Security 2015.)

IDS:n tarkoituksena on nostaa tietoturva-asiantuntijoille ilmoitukset, kun heidän suojattavaan ympäristöön on hyökätty tai edes yritetty. Ilmoituksen saatuaan he voivat käynnistää prosessit tämän hyökkäyksen pysäyttämiseksi (An Introduction to IDS 2001). IDS on hyvä esimerkki passiivisen kyberturvallisuuden komponentista, sillä tämä voidaan jakaa kahteen eri kategoriaan: Network-Based Intrusion Detection System (NIDS) ja Host-Based Intrusion Detection System (HIDS).

HIDS-tuote asennetaan paikallisesti koneille, josta ne keräävät tietoa kyseisen koneen käyttäytymisestä. Tieto, joka näiltä koneilta kerätään, voidaan analysoida joko paikallisesti tai lähettää keskitetylle lokienhallintajärjestelmälle. HIDS:llä voidaan pysyä perillä käyttäjän toiminnasta kyseisellä koneella. Tällä voidaan havaita, mikäli käyttäjä pyrkii tekemään toimenpiteitä mitä hänen ei kuuluisi tehdä. Tämä voi tarkoittaa esimerkiksi tiedostojen muokkausyritystä, joihin heidän ei pitäisi edes koskea. Kun tällainen tapahtuma havaitaan, niin siitä lähtee ilmoitus tai hälytys tietoturvatilille. HIDS:n yksi heikkous on käyttäjien runsas määrä. Silloin dataa voi kertyä to-

della paljon ja sen analysointi voi kestää liian kauan. Toisaalta mikäli hyökkääjä havaitsee, että koneelle on asennettu HIDS, hän yksinkertaisesti ajaa tämän palvelun alas. (An Introduction to IDS 2001.)

Suojattavan kohteen kasvaessa tarpeeksi isoksi on parempi käyttää NIDS:iä. NIDS:t toimivat "packet-sniffereinä" ja pyrkivät täten tutkimaan kohdeverkon läpi menevää dataa ja päättämään, onko kyseinen liikenne haitallista vai ei. NIDS:ien isoimpina haasteina on kehitys, jossa siirrytään yhä nopeampiin verkkoihin ja tietoliikenne suojataan kryptaamalla. (Cole 2009, 550-551.) Kryptattu liikenne aiheuttaa sen, että kyseinen NIDS ei välttämättä pysty tulkitsemaan tätä liikennettä. Kuvio 1 on havainnollistettu esimerkki HIDS:n ja NIDS:n paikoista tietoverkossa.



Kuvio 1. HIDS ja NIDS

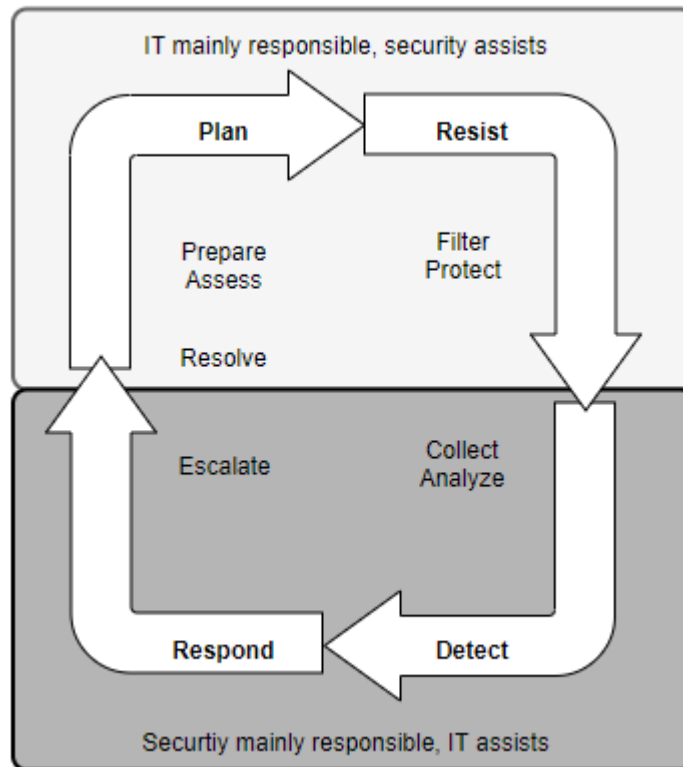
3.3 Aktiivinen kyberturvallisuus

Aktiivinen kyberturvallisuus nojaa ihmisten reagointiin. Tämä vaihe on otettu huomioon tilanteessa, kun vastassa ei ole enää normaali haittaohjelma ja passiivisen puolustuksen työkalut eivät sitä pysty havaitsemaan. Aktiivisessa kyberturvallisuudessa on tärkeää tunnistaa, mihin hyökkäys on kohdistunut ja miten tämä hyökkäys saataisiin pidettyä hallinnassa. Ryhmiä, joita aktiivisesta puolustuksesta löytyy, ovat muun muassa SOC-analyttikot ja haittaohjelmien takaisinmallintajat. Aktiivinen puolustus on niin hyvä, kuin mitä nämä henkilöt ovat koulutukseltaan/ammattitaidoltaan. Näiden henkilöiden tulee pysyä tekniikan mukana, muuten tämä vaihe pettää infrastruktuurin, jota suojellaan. (The Sliding Scale of Cyber Security 2015.)

Aktiivisen kyberturvallisuuden yhtenä mallina on Network Security Monitoring (NSM). NSM tarkoittaa tapaa etsiä poikkeavuuksia, esim. hälytyksiä murrosta, ja sulkea haittatekijät niin, että hyökkäys aiheuttaa minimaalisesti vahinkoa yritykselle. Henkilöitä, jotka työskentelevät murtoja vastaan, kutsutaan Computer Incident Responder Teamiksi (CIRT). Näiden henkilöiden tarkoituksena on tehdä hyökkääjien työ mahdollisimman vaikeaksi. CIRT:t, jotka työskentelevät NSM-periaatteiden mukaan toimivat seuraavasti: He keräävät mahdollisimman paljon dataa hyökkäyksestä, analysoivat kerätyn datan, saadakseen selville mitkä laitteet ovat joutuneet hyökkäyksen kohteeksi. Kun kohteet on löydetty he työskentelevät laitteiden omistajien kanssa, jotta hyökkäys saadaan loppumaan. Kerätyn datan perusteella he selvittävät kuinka paljon vahinkoa aiheutui sekä mistä hyökkäys sai alkunsa. (Bejtlich 2013, 4-5.)

Kuvio 2 käydään läpi CIRT:n toiminta yrityksessä. Ensimmäisessä osuudessa IT-henkilöstö käyttää passiivista kyberturvallisuutta hyödyksi ja toisessa osuudessa CIRT:t hyödyntävät aktiivista kyberturvallisuutta. Toiminta aloitetaan suunnittelusta, miten infra saadaan suojattua ja mihin passiivinen kyberturvallisuus sijoitetaan. Tämän jälkeen annetaan passiivisen puolustuksen toimia ja pyrkii estämään hyökkääjien toiminnan sekä pienentämään hyökkäysvektoria. Koska mikään kyberturvallisuus ei ole täysin varma, kun hyökkääjä pääsee passiivisen puolustuksen läpi, täytyy CIRT:ien alkaa toimia. He havaitsevat hyökkäyksen, keräävät datan sekä analysoivat

sen. Kun on saatu käsitys hyökkäyksestä, täytyy siihen vastata ja toimia asianmukaisesti eli saada hyökkääjä pois kohteesta ja minimoida vahingot. Kun hyökkäys on esitetty, täytyy tehdä riskilaskentaa ja arvioida hyökkäyksen suuruus ja päivittää suunnitelma sekä passiivinen puolustus. On hyvä olla tietoinen siitä, että yli puolta tietomurroista ei huomata kuin vasta kuukausien kuluttua murrosta. (Bejtlich 2013, 4-5.)



Kuvio 2. Tietoturva yrityksissä (Bejtlich 2013, 5.)

4 SIEM/Flowdata

4.1 SIEM

Security Information and Event Management (SIEM) on tuote, joka koostuu useasta eri teknologian komponentista. Tämän avulla voidaan luoda kuva ympäristöstä, jonka kanssa ollaan tekemisissä tekniikan kanssa. (What Is a SIEM? 2016.)

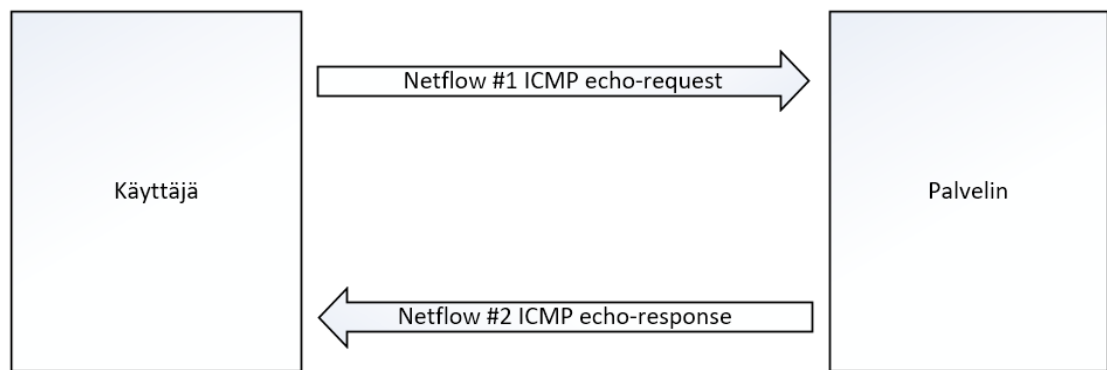
SIEM:ssä tärkeimpiä asioita ovat:

- Lokien ja tapahtumien kerääminen eri laitteilta, kuten palomuurilta, kytkimiltä, reitittimiltä. Sekä lokien hallinnointi.
- Hälyttäminen, tämä on tärkeä osuus esim. SOC:in toiminnalle, jotta he pystyvät reagoimaan tapahtumiin reaaliajassa.
- Kokonaiskuva ja datan esitys, antaa hallinnoivalle tiimille kerätystä datasta kokonais kuvan tapahtumista ja niiden yhtäläisyyksistä. (What Is a SIEM? 2016.)

4.2 Flow data

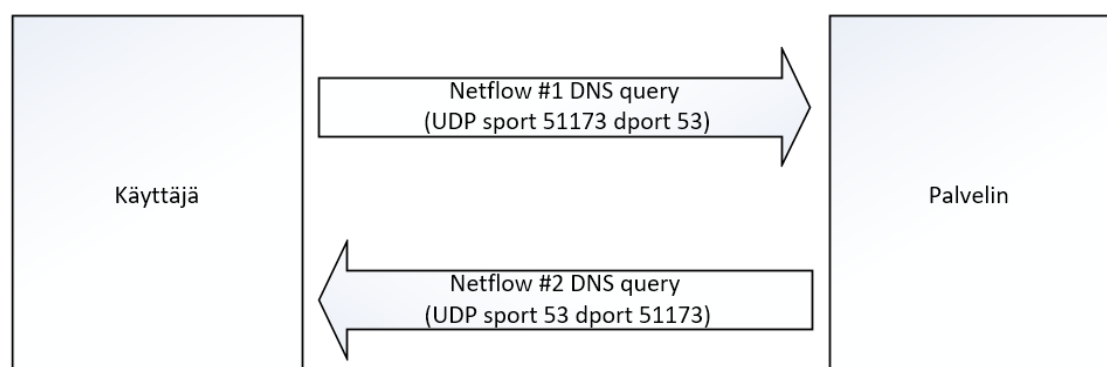
Flow datalla tarkoitetaan kahden laitteen välistä keskustelua, jossa ei tarkastella niiden asiasisältöä, vaan näiden keskustelua tarkastellaan ylimalkaisesti. (Traffic Analysis for Network Security - - 2018). Uniikki flow määritellään niin, että paketissa on sama lähde- ja kohdeosoite porttia myöten. Flowssa ei myöskään saa muuttua Type of Service (ToS), IP-protokolla tai mistä rajapinnasta kyseinen liikenne tulee. Yhdenkin arvon muuttuessa luodaan uusi flow. (Why You Should Start Leveraging Network Flow - - 2018.) Flow data on arvokasta, koska sen avulla voidaan analysoida verkon toimintaa sekä tämä tieto saadaan tallennettua huomattavasti pienempään tilaan, kun se ei sisällä paketin dataa. Suojaava taho saa jo flow datan sisältämästä tiedosta paljon informaatiota, jopa havaitsemaan poikkeamia verkostaan ja tämän perusteella tekemään suojaavia toimenpiteitä verkossaan. (Traffic Analysis for Network Security - - 2018.) Seuraavissa esimerkeissä oletetaan, että ToS, IP-protokolla sekä rajapinta pysyvät samana.

Kuvio 3 esitetään helpoin käsite flow datasta eli Internet Control Message Protocol (ICMP) flow. Tämä käydään läpi tarkastelemalla pingaamista. Pingissä on kaksi eri flowta yhdistettynä käyttäjän näkyviin. Käyttäjältä lähtevä echo-message kohteeseen on ensimmäinen osa tätä flowta ja echo-reply kohteelta käyttäjälle on toinen osa. Jos käyttäjä jatkaa kohteen pingaamista, niin nämä kaikki echo-messaget kuuluvat samaan flowhun, samoin kuten echo-replyt, koska flow periaatteen mukaan näissä ei muutu mikään. (Lucas 2010, 14-15.)



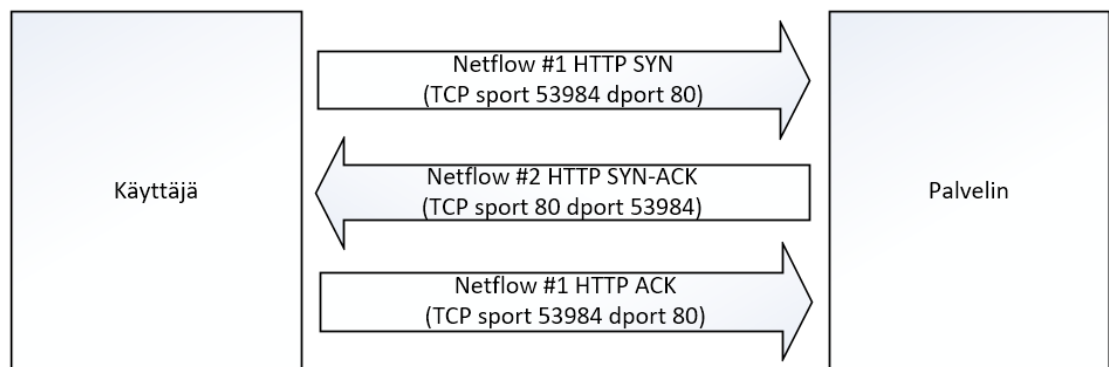
Kuvio 3. ICMP Netflow

User Datagram Protocol (UDP) flowssa käsitellään enemmän tietoa, sillä UDP paketti pitää sisällään porttitiedon. UDP on yhteydetön protokolla ja toimii, ammu ja unohda periaatteella. Domain Name System (DNS) kyselyä tarkastellaan helppona esimerkkinä UDP flowsta Kuvio 4. Käyttäjä aloittaa kyselyn lähettämällä DNS-palvelimelle kyselyn kohteesta, jolle se haluaa IP-osoitteen. Käyttäjä tarvitsee tämän voidakseen, aloittaa keskustelun kyseisen laitteen kanssa. Kysely lähtee portista, joka ei ole käytössä palvelimen porttiin 53. Tämän jälkeen se saa paluuviestin samaan porttiin, josta kysely lähti. Molemmat UDP ja ICMP ovat samanlaisia siinä, koska kummallakaan ei ole tapaa ilmoittaa yhteyden päättymisestä, vaan flow datan kerääjä pitää kerätyn flow datan tietyn aikaa muistissaan, jonka jälkeen se poistaa sen. (Lucas 2010, 15-16.)



Kuvio 4. UDP Netflow (Lucas 2010, 15.)

Transmission Control Protocol (TCP) flowssa tuodaan vielä enemmän tietoa tarkasteltavaksi, koska tässä tulevat niin sanotut liput mukaan. TCP yhteydessä liput kertovat yhteydentilan. Kun tarkastellaan TCP yhteyttä esim. HTTP-yhteyden aloitusta flowmielessä, alkaa yhteys samalla tavalla kuin UDP:ssä eli lähde valitsee käyttämättömän portin käyttöönsä. Kohteen porttina on 80, koska tämä on de facto-portti http-yhteydelle. Yhteys alkaa lipulla Synchronization (SYN), millä lähde ilmoittaa kohteelle, että se haluaa muodostaa yhteyden sen kanssa. Kohde vastaa Synchronization-Acknowledgement (SYN-ACK) viestillä, jos se on kykenevä muodostamaan yhteyden sillä hetkellä. Lähteen tarvitsee vastata tähän vielä Acknowledgement (ACK) viestillä. Tämä yksi TCP yhteys on muodostunut kahdesta flowsta. Ensimmäinen flow on lähteeltä tulevat viestit, koska näissä kohteen ja lähteen osoitteet sekä portit pysyvät samoina. Toisena flowna on palvelimen vastaus (ks. Kuvio 5). (Lucas 2010, 16-17.)



Kuvio 5. TCP Netflow (Lucas 2010, 17.)

5 Apache Spot

5.1 Yleistä

Apache Spot on avoimen lähdekoodin alusta, jonka avulla kerätään ja analysoidaan flow dataa sekä paketteja. Sen tarkoitus on olla niin yritysten kuin palvelun tarjoajien apuna estämässä tietoturvaaukia. Spot parantaa yritysten näkyvyyttä heidän verkkoonsa ja nostaa poikkeamia keräämällään tiedolla. Se hyödyntää Machine Learning (ML) tutkiessaan poikkeamia verkossa. Alusta on suunniteltu kasvavaan ympäristöön,

jopa pilvipalvelualusta ympäristöihin. (Apache Spot (Incubating) 2015.) Apache Spotissa puhutaan käsitteestä netflow, joka tarkoittaa samaa asiaa kuin flow data.

5.2 Hadoop

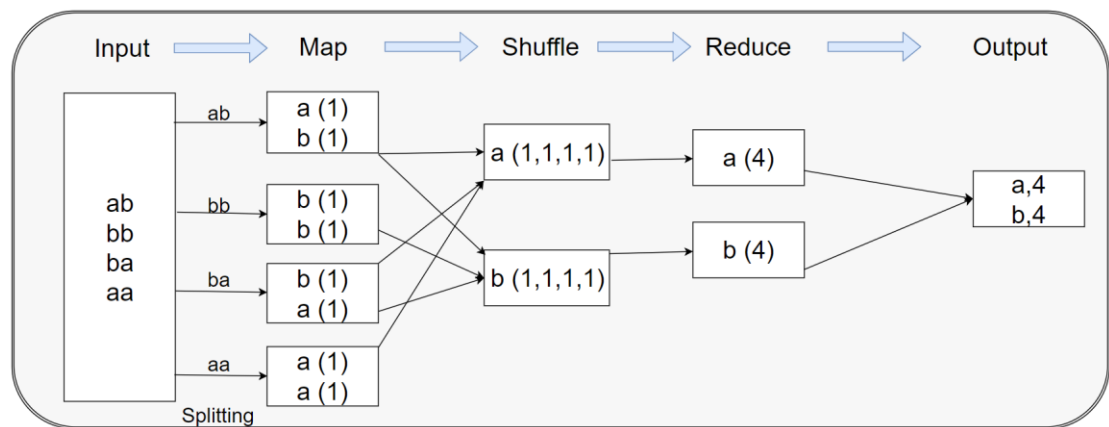
5.2.1 Yleistä

Hadoop on ympäristö, jonka avulla on mahdollista käsitellä isoa määrää dataa ryhmällä koneita, eli klusterilla. Hadoopin tarkoituksena on mahdollistaa korkea palvelusaatavuus. Korkea saatavuus on tarkoitus saavuttaa ohjelmistolla eikä laitteistolla. Hadoopin tarkoitus on havaita ongelmatilanteet ja pyrkiä lieventämään vahingot, jotka ovat jo sattuneet. Parhaatkin koneet ovat vika-alttiita ja vikatilanteet pitää ratkoa nopeasti, jotta korkea palvelun saatavuus voidaan saavuttaa. (Apache Hadoop n.d.)

MapReduce on Hadoopissa toimiva keskeinen toiminto, jonka avulla pystytään käsittelemään suuria tietomääriä. Kyseinen prosessi jaetaan kahteen eri päävaiheeseen, Map- ja Reduce-vaiheeseen. Map-vaiheessa muodostetaan käyttäjän antaman tiedoston pohjalta avainpareja. Tämän jälkeen alkaa Reduce-vaihe, joka on jaettu kahteen vaiheeseen: Shuffle- ja Reduce-vaihe. Shuffle-vaiheessa Map:sta saatu tieto käsitellään uudestaan, ja luodaan uudet avainparit, jotka syötetään Reduce-vaiheeseen. Reduce-vaiheessa käydään nämä uudet avainparit läpi ja tulostetaan redusoidumpi avainparilistaus alkuperäisestä tiedosta, joka kyseiselle prosessille syötettiin. (Hadoop - Mapreduce n.d.) Prosessin toiminta nähdään yksinkertaisuudessaan Kuvio 6. MapReducessa työn käsittelyä tehostetaan käsittelemällä tieto useammalla eri palvelimella, eli nodella. Itse tietoa ei tässä siirretä, vaan kyseinen prosessi vie tietoa luokse. Tällä ehkäistään verkon tukkeutumista ja nopeutetaan prosessia. (MapReduce Tutorial 2018.)

Kuvio 6 käyttäjä haluaa tietää, montako a- ja b-kirjainta tiedostossa on. Käyttäjä antaa tiedoston, jonka hän haluaa käsiteltävän. Tehtävä jaetaan neljälle eri nodelle, jotka käsittelevät oman osuutensa tiedosta. Tämän jälkeen tapahtuu shuffle-vaihe eli tietoa yhdistetään niin, että samat avainparit ovat samalla nodella. Tämän jälkeen

siirrytään itse Reduce-vaiheeseen eli shufflesta saatu tieto redusoidaan ja yhdistetään avainparien arvoja. Reducen tulos annetaan käyttäjille tulosteena.



Kuvio 6. MapReduce

5.2.2 Hadoop Distributed Filesystem

Hadoop Distributed File System (HDFS) on hajautettu levyjärjestelmä, jonka tarkoitus on pyöriä jokapäiväisellä raudalla. HDFS on suunniteltu olemaan vikasietoinen, se voidaan siirtää yhdeltä alustalta toiselle helposti. Tämä levyjärjestelmä on suunniteltu ottamaan vastaan isoja tiedostoja, joiden koot voivat vaihdella gigoista jopa teroihin. Tiedostojen ollessa näin isoja on paljon helpompaa siirtää laskenta tiedon luokse kuin toisin päin. Tämän avulla säästetään infrastruktuurin kapasiteettia, jossa HDFS toimii. Vaikka HDFS mahdollistaa suoratoiston ohjelmille, jotka käyttävät sen sisältämää dataa, vasteajat eivät ole pääpiste vaan suuri kapasiteetti, joka pystytään tarjoamaan. (HDFS Architecture Guide 2018.)

HDFS on jaettu kahteen eri tyyppiin: master ja worker. Master-tyyppiä kutsutaan nimellä namenode ja worker-tyyppiä nimellä datanode. Namenode toimii johtoroolissa, eli se tietää tiedon sisällön sekä sen paikan datanodeissa. Namenoden tieto on jaettuna kahteen osioon: edit logiin ja namespace imageen. Namenode tietää, missä tiedostojen lohkot sijaitsevat siihen asti, kunnes palvelu käynnistetään uudelleen, jolloin tämä tieto rakennetaan uudestaan. (White 2015, 46.)

Hadoopissa tieto ei ole jaettu tiedostoihin vaan lohkoihin. Yksi Hadoopin lohko on oletuksena 128 MB. Tämä lohkon oletuskoko on valittu, taklaamaan hakujen minimointia. Mikäli tiedosto, joka tuodaan, on pienempi kuin mitä lohkon koko on, niin se vie vain oman kokonsa verran levytilaa. Mikäli tiedosto on suurempi kuin mikä lohkon koko on, nämä lohkot tallennetaan eri datanodeille. Harvoissa tapauksissa on myös mahdollista tallentaa nämä saman tiedoston lohkot yhdelle datanodelle, mutta tämä ei ole suositeltavaa. Suojausmekanismina tiedon tuhoutumista vastaan tallennetaan yksi lohko kolmelle eri datanodelle. Vaikka tieto tuhoutuisi yhdestä paikasta, on tieto kuitenkin vielä saatavilla. Hadoop huolehtii automaattisesti siitä, että tämä tuhoutunut lohko korvataan uudella kopiolla. Tällä varmistetaan tiedon tallessa pysyminen. (White 2015, 45.)

Namenodet on myös todella tärkeitä suojata tuhoutumisen varalta, sillä vain ne tietävät mihin tieto on tallennettu. Mikäli kaikki namenodet tuhoutuvat, on käytännössä samalla datakin tuhoutunut, koska kukaan ei tiedä enää tiedon paikkaa. Yhtenä mahdollisuutena voidaan konfiguroida namenoden muuttumaton tieto toiseen järjestelmään, josta se voidaan palauttaa, mikäli namenode sattuisi hajoamaan. Näin voitaisiin rakentaa tiedostot uudestaan lohkoista. (White 2015, 47.)

5.2.3 HIVE

Apache Hive on tietovarastotyökalu, jonka tarkoituksena on tehdä kyselyitä sekä analysoida Big Dataa (Hive-Tutorial 2018). Big Data on yksinkertaisuudessaan käsite, jolla tarkoitetaan tietoa, jota tulee useasta lähteestä, todella suuria määriä ja erittäin nopealla tahdilla (What is Big Data? n.d). Hive käyttää kyselyitä tehdessään sekä analysoidessaan Structured Query Language (SQL) pohjautuvaa kieltä nimeltä Hive Query Language (HQL). Hiven tarkoituksena on helpottaa käyttäjää tiedonkäsittelyssä, niin että hänen ei tarvitse tehdä monimutkaisia MapReduce-ohjelmia. (Hive-Tutorial 2018.)

5.2.4 IMPALA

Apache Impalan tarkoitus on nopeuttaa Hadoopissa tapahtuvia SQL kyselyitä, oli tietokantatoteutuksena sitten HDFS tai Apachen oma HBASE. Impala käyttää Apache Hivessä olevia vakiotuja ominaisuuksia, kuten samanlaista metadataa sekä samanlaisia syntakseja kuin HQL:ssä. (Overview n.d.)

Impala laskee viivettä, joka normaalisti aiheutuisi MapReducesta tai tiedostoista, jotka eivät ole tallennettuina lokaalisti. Tämä viive käsitellään Impalan palveluprosessien avulla, jotka suoritetaan Hadoopin infrastruktuuria ylläpitävillä palvelimilla. Nopeutensa ansiosta Impala pystyy kilpailemaan jopa maksullisten järjestelmien kanssa. (Impala: A Modern, Open-Source SQL Engine for Hadoop n.d.)

Impala on jokseenkin riippuvainen tietokantatoteutuksesta. Impalasta saadaan maksimaalinen teho ulos, kun tietokantatoteutuksena käytetään HDFS:ää. HDFS:n avulla voidaan saada nopeampia hakuajoja kuin maksullisista ohjelmista. Impala käyttää hyödykseen Hadoopista jo valmiiksi löytyviä osia, kuten Yet Another Resource Negotiator:a (YARN) ja HDFS:ää, jotta tämä voisi tuottaa relaatiotietokannan tapaisen kokemuksen. Jotta käyttäjät pääsevät tarkastelemaan tietoa mitä Impalasta löytyy, on heillä kaksi mahdollista yhdistäytymistapaa: Open Database Connection (ODBC) ja Java Database Connection (JDBC). Nämä molemmat ovat Application Programming Interface:ja (API), joiden avulla käyttäjä pystyy yhdistäytymään Database Management Systems:n (DBMS). Tunnistautumisessakin on kaksi eri tapaa: Lightweight Directory Access Protocol (LDAP) ja Kerberos. (Impala: A Modern, Open-Source SQL Engine for Hadoop n.d.)

5.2.5 KAFKA

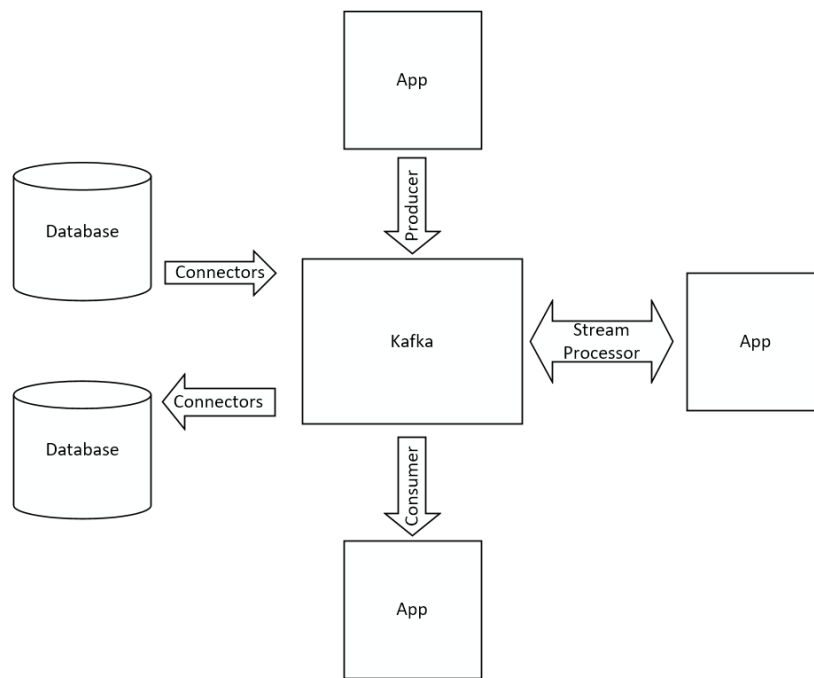
Apache Kafka on hajautettu suoratoistoalusta, joka toimii yhdellä tai useammalla palvelimella. Suoratoistajärjestelmillä on kolme pääominaisuutta: (1) Toimittaa suoratoista reaaliajassa, (2) pystyä pitämään näitä suoratoistoja tallessa, (3) vikatilanteen sattuessa toiminta pystyy jatkumaan ennallaan sekä julkaisemaan ja tilaamaan suoratoistoa. (Introduction n.d.)

Kafka alustalla on kaksi päätarkoitusta. Sillä rakennetaan reaaliajassa toimivia suoratoistojakoja mahdollistaen datan välityksen kahden palvelun välillä tai sillä rakennetaan reaaliajassa toimivia ohjelmia, jotka käsittelevät niiden saaman datan. (Introduction n.d.)

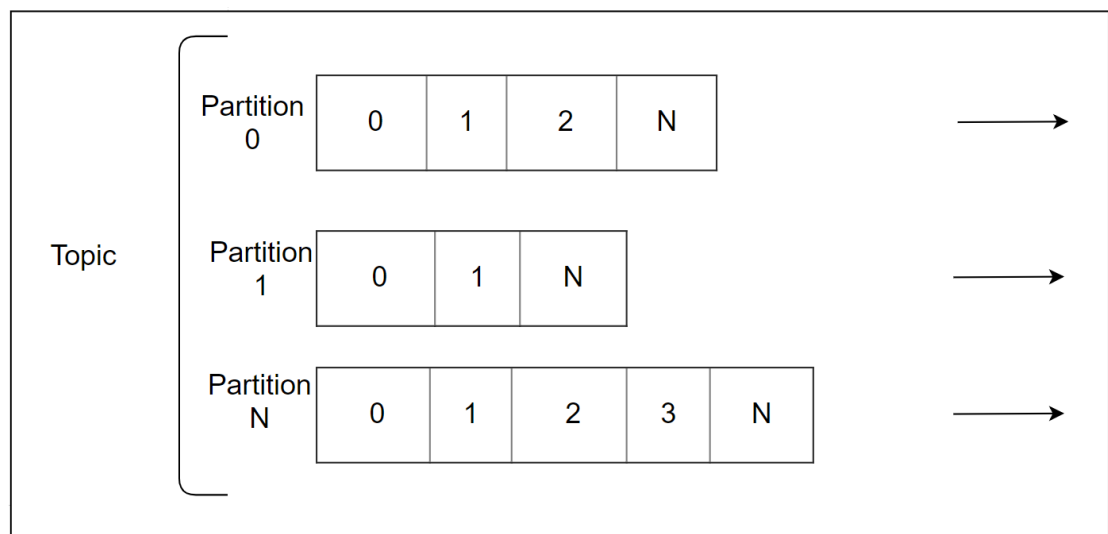
Kuvio 7 käydään läpi Kafkan neljä erilaista API eli ohjelmointirajapintaa. Kahden näistä voidaan ajatella olevan toistensa vastakohtia, ne ovat Producer ja Consumer API. Producer mahdollistaa ohjelmien tuovan tietoa Kafkaan, kun taas Consumer jakaa tätä tietoa Kafkasta eteenpäin. Connector API antaa mahdollisuuden yhdistää esim. tietojärjestelmiä, joihin voidaan tallentaa dataa, myöhemmin uudelleen käytettäväksi jollain toisella ohjelmalla. Streams API mahdollistaa, että ohjelmat pystyvät keräämään useamman topicin yhteen tai useampaan topiciin. Näitä topiceja voidaan sitten suoratoistaa tilaajille. (Introduction n.d.)

Kafkassa topic tarkoittaa kategoriaa suoratoiston tallenteesta. Topicit jaetaan osioihin, jotta näiden käsitteleminen olisi nopeampaan. Kun topicin osioon lisätään tietoa, lisätään se aina sen loppuun ja se saa järjestysnumeron. Tämä järjestysnumero on osio kohtainen ja se kertoo tiedon sijainnin kyseisen osion sisällä, tätä numeroa kutsutaan offsetiksi (ks. Kuvio 8). Kerran kun tieto on kirjoitettu topiciin niin sitä kirjoitettua tietoa ei voi muuttaa. Topicien osioita voidaan kirjoittaa useammalle eri palvelimelle, mutta osion täytyy kokonaisuudessaan mahtua sinne. Topicilla ei tarvitsella olla yhtään tilaajaa tai sillä voi olla useampia tilaajia. (Apache Kafka Topic – Architecture & Partitions 2018.)

Topiceille voidaan määrittää aika, kuinka kauan niitä säilötään, riippumatta siitä käytettiinkö kyseistä tietoa hyödyksi vai ei. Vaikka topiceihin asetettaisiin pidempi säilytysaika, ei tällä ole juuri minkäänlaista vaikutusta Kafkan nopeuteen. (Introduction n.d.)



Kuvio 7. Kafka (Introduction n.d.)



Kuvio 8. Kafka topic

5.2.6 SPARK

Spark on avoimen lähdekoodin klusterialusta, jonka tarkoitus on nopeuttaa Hadoopissa tapahtuvaa laskentaa. Spark jatkaa MapReducen idean päälle ja parantaa tätä, esim. mahdollistamalla interaktiiviset kyselyt. Sparkin nopeus tulee siitä, että se ajaa laskentansa muistissa eikä levyllä, kuten MapReduce. Tämän avulla voidaan saavuttaa jopa satakertainen ero. Sparkissa on useampi ohjelmointirajapinta ja se tukee kieliä kuten Python sekä Java. (Apache Spark - Introduction n.d.)

Spark kokonaisuutena sisältää monta komponenttia, jotka on integroitu tiiviisti toimimaan yhdessä. Pohjimmiltaan Spark on laskennallinen moottori, joka toimii hallinnoijana muille laskenta prosesseille. Sparkin ytimen tehokkuutensa ansiosta pystyy se valjastamaan vaativampia komponentteja kuten SQL:n omalla tehollaan erinäisiin työtehtäviin. Tiiviin integraation myötä mahdollistetaan helppo tuotteen ylläpito sekä jos tuote saa uuden lisäosan tarvitsee tuote vain päivittää, jonka jälkeen lisäosa on jo käytettävissä. (Karau, Konwinski, Wendell & Zaharia 2015, 2.)

Spark Core

Sparkin ytimestä löytyy Sparkin perusominaisuudet, kuten tehtävien ja muistin hallinta. Sparkin ytimessä on myös ohjelmointirajapinta Resilient Distributed Data-
seteille (RDD). RDD tarkoittaa kokoelmaa tiedosta, jotka ovat hajautettu useammalle eri nodelle. Tätä tietoa pystytään käyttämään hyödyksi samanaikaisesti. Spark Co-
resta löytyy myös ohjelmointirajapinta, jota kautta tietoa voidaan muokata. (Karau, Konwinski, Wendell & Zaharia 2015, 3.)

Spark SQL

Spark SQL:n tarkoitus on käsitellä jäsenneltyä tietoa. Tämä kertoo Sparkille minkälainen on tiedon sekä laskennan rakenne, jota aiotaan käsitellä. Spark SQL voi käyttää SQL-kyselyillä, ohjelmointirajapinnalta tulevalla tiedolla tai molemmilla. Spark SQL on monipuolinen, koska sillä on useampi eri ohjelmointirajapinta. Käyttäjä voi vaihtaa useamman eri ohjelmointirajapinnan välillä, koska suoritusmoottori, joka ajaa nämä kyselyt ei ota kantaa niiden alkuperään. (Apache Spark n.d.)

Spark Streaming

Spark Streamingin mahdollistaa tiedon suoratoiston. Tietoa suoratoistoon voidaan kerätä useammasta eri lähteestä sekä tätä voidaan käsitellä toiminnoilla kuten MapReducella. Sparkin Machine Learning-kirjasto (MLib) voi hyödyntää kyseistä tietoa. Kerätty tieto voidaan tulostaa mm. tietokantaan tai suoraan käyttäjän näkyville. (Apache Spark n.d.)

MLib

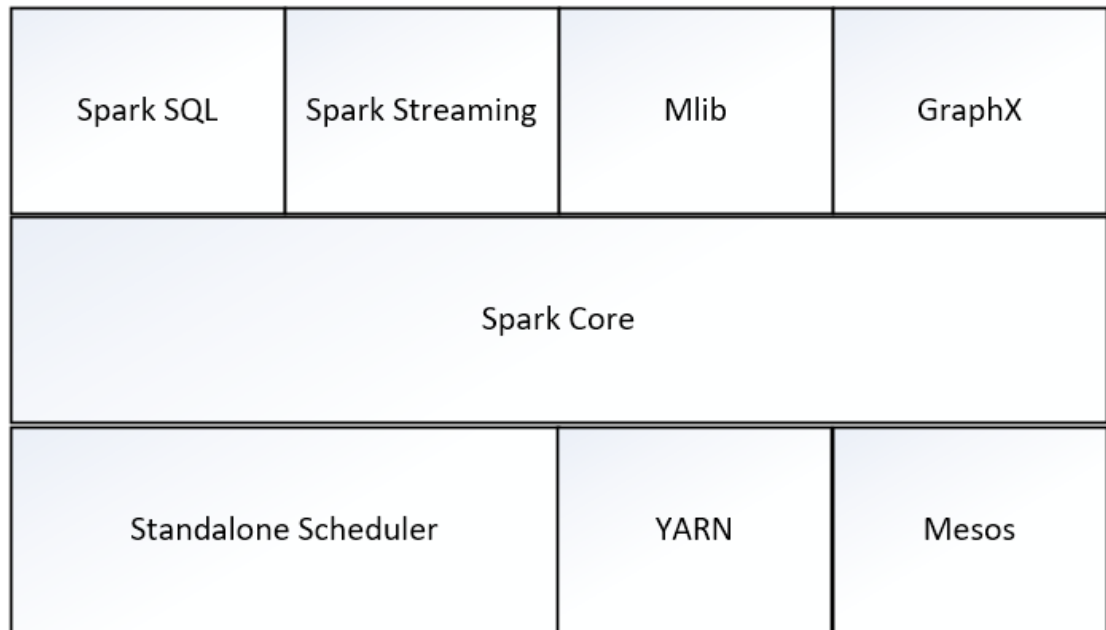
Sparkin mukana tulee Machine Learning (ML)-kirjasto, josta löytyy yleisiä toiminnallisuuksia mitä voidaan olettaa ML:stä. Kirjasto sisältää mm. eri tyyppisiä ML algoritmeja, lokerointi ja tiedon vastaanottamista. Kaikki mitä kyseisestä kirjastosta löytyy, on suunniteltu jaettavaksi koko klusterille. (Karau, Konwinski, Wendell & Zaharia 2015, 4.)

GraphX

GraphX on osio kuvaajille sekä kaavio-rinnakkaislaskennalle. GraphX jatkaa siitä mihin Sparkin RDD jäi. GraphX:llä voi muodostaa esim. kuvaajia, joissa janan molemmissa päissä on ominaisuus ja näiden kahden ominaisuuden välinen yhteys kuvataan janan sivulla. Kyseistä kuvaajaa kutsutaan nimellä property graph. GraphX sisältää laajan koelman erilaisia kaavioalgoritmeja sekä rakentajia, jota laajennetaan koko ajan. (GraphX Programming guide n.d.)

Cluster Managers

Sparkin suunnittelussa otettiin huomioon se, että tämän tulee olla todella hyvin skaalautuva palvelu. Jotta Sparkia pystytään ajamaan useammalla palvelimella yhtä aikaa, täytyy näiden välillä toimia koordinaattori, kuten YARN tai Apache Mesos, mikäli nämä ovat asennettuja klusteriin. Jos kuitenkin on niin, että näitä ei ole, niin Sparkista löytyy oma hallinnointi järjestelmä nimeltä Standalone Scheduler. Kyseisellä hallinnoinnilla pääsee alkuun (ks. Kuvio 9). (Karau, Konwinski, Wendell & Zaharia 2015, 4.)



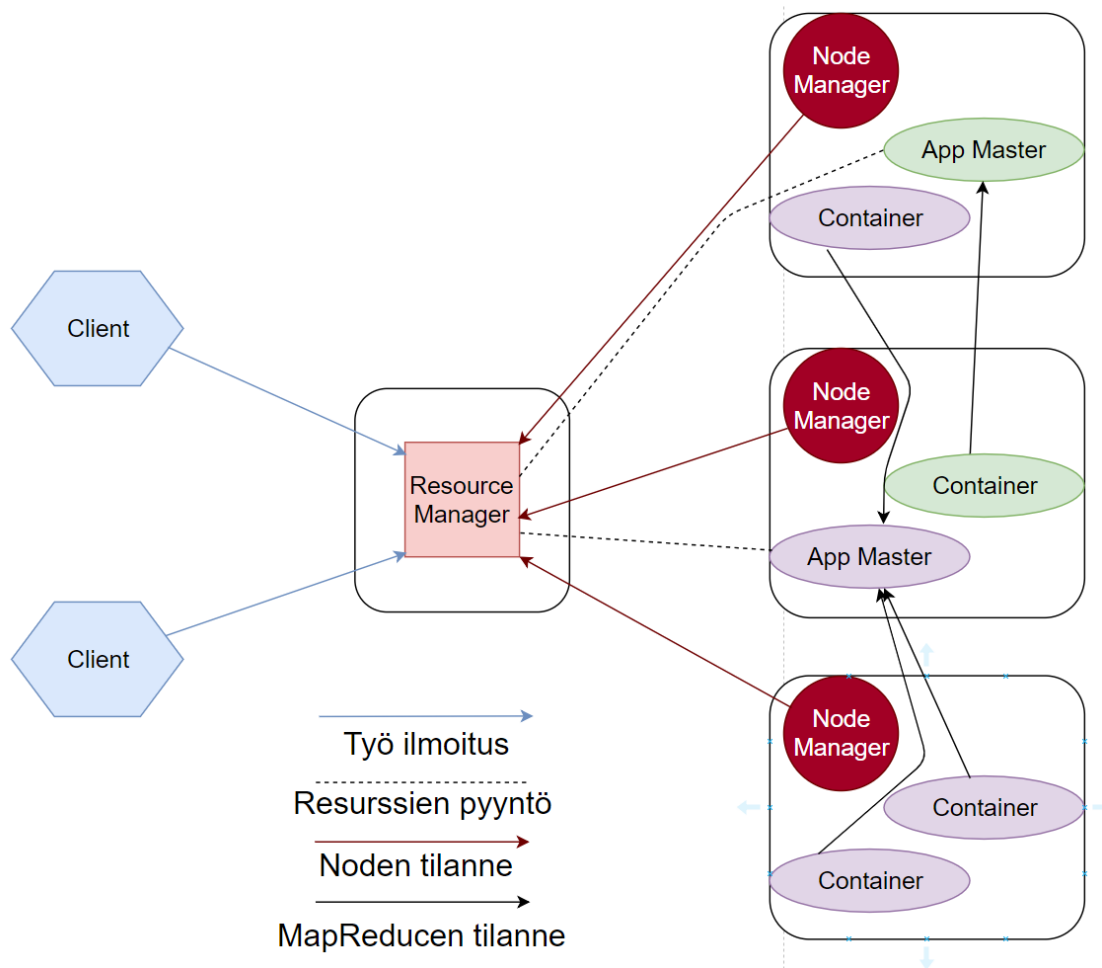
Kuvio 9. Spark (Karau, Konwinski, Wendell & Zaharia 2015, 3.)

5.2.7 YARN

YARN:n tarkoitus on parantaa resurssienhallintaa Hadoopissa, jakamalla toiminnot kahteen eri palveluprosessiin resurssienhallintaan ja monitorointiin (Apache YARN, n.d). YARN on jaettu neljään pääkomponenttiin. Ne ovat Resurssienhallinta, Node Manager, Application Manager sekä Container. Resurssienhallinta on globaali palveluprosessi, eli näitä löytyy vain yksi klusterista. Se vastaa nimensä veroisesti resurssienhallinnasta sekä näiden resurssien jakamisesta. Tämä on jaettu kahteen eri osaan, Scheduleriin ja Application Masteriin. Scheduler on vastuussa vain ja ainoastaan resurssien jakamisesta eri töille, scheduler ei ota vastuuta työn uudelleenkäynnistämisestä, jos joku työ sattuu kaatumaan. Toinen osa on Application Manager (AM), tämä vastaa töiden hyväksymisestä sekä resurssien myöntämisestä resurssienhallinnan kanssa Application Masterille. Toisin kuin Scheduler niin Application Manager käynnistää uudestaan Application Mastereita, jos ne sattuvat kaatumaan ennen aikojaan. Node Manager monitoroi koko ajan noden kuntoa, koska se on vastuussa siitä sekä sen sisällä tapahtuvista töistä. Node Manager on myös vastuussa Containereiden luomisesta. Container viittaa resursseihin mitä kyseiselle työlle on suotu. (Hadoop YARN Tutorial - - 2018.)

Jokaisella työllä mitä YARN vastaanottaa on oma Application Master, joka valvoo tämän kyseisen työn toimintaa alusta loppuun. Application Master on vastuussa resurssien varaamisesta resurssienhallinnasta sekä yhteistyöstä Node Managerin kanssa jonka alueelta se saa nämä containerin resurssit käyttöönsä. (YARN's application master in Hadoop n.d.)

Kuvio 10 nähdään miten yksinkertaisesti YARN toimii. Tilaaja ilmoittaa työstä resurssienhallintaan, App Master varaa tarvittavat resurssit näille töille. Node Manager luo containerit ja lainaa niiden sisältämiä resursseja, kunnes kyseinen valmistuu ja käyttäjä saa työn lopputuloksen.



Kuvio 10. YARN (Apache YARN, n.d)

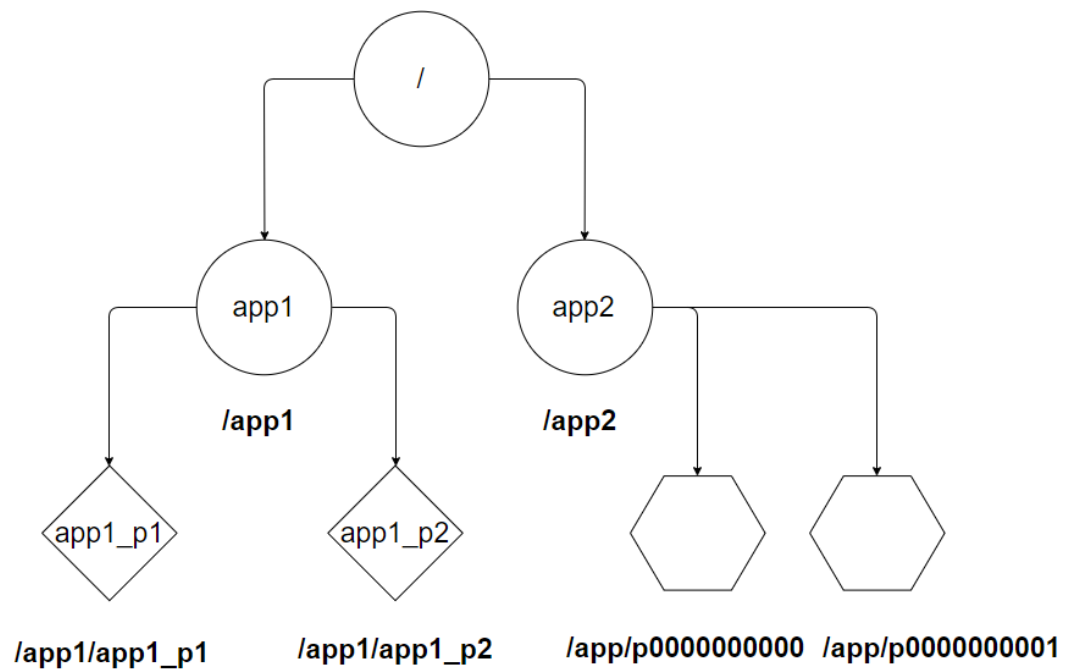
5.2.8 Zookeeper

Zookeeper on avoimen lähdekoodin koordinaattori palvelu muille palveluille, joiden pitää olla synkronissa keskenään. Zookeeperin tarkoitus on olla nopea sekä aina valmiina vastaanottamaan pyyntöjä tilaajilta. Nopeus Zookeeperissa saadaan siitä, että se tekee työnsä muistissa eikä levyllä. (Apache Zookeeper n.d.)

Zookeeper on replkoituna useammalle eri palvelimelle, jotka ovat tietoisia toistensa tiloista. Toimivuus voidaan varmistaa tämän tiedon avulla, sekä toimimattomat palvelimet voidaan eliminoida tämän perusteella. Tilaajat ottavat yhteyden TCP:llä näihin palvelimiin. Yhteyden katketessa tilaaja ottaa uuden yhteyden uuteen palvelimeen. (Apache Zookeeper n.d.)

Zookeeperin nodeja kutsutaan nimellä znode. Znodejen rakenne on hierarkkinen hakemistopolku (ks. Kuvio 11). Poikkeavaa Zookeeperissa on se, että znodet voivat sisältää tietoa sekä aliprosesseja. Tieto, jota znodet pitävät sisällään ovat mm. versio numero, aikaleima sekä tieto oikeuksista Access Control List (ACL). Versio numero kasvaa aina kun tapahtuu muutoksia, tämä lähetetään tiedon mukana tilaajalle. (Apache Zookeeper n.d.)

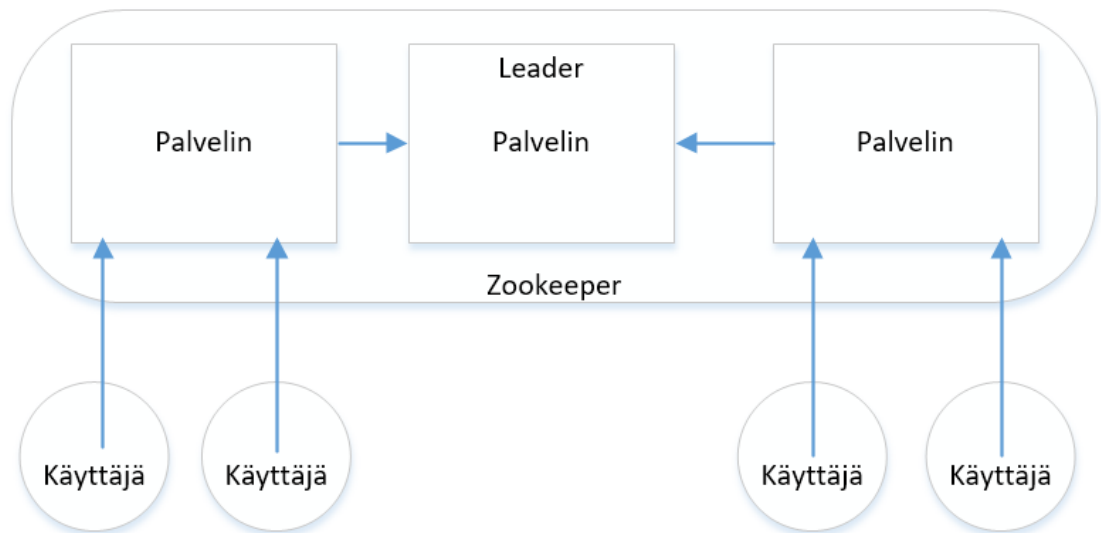
Znodeja on kolmea eri tyyppiä: (1) pysyviä znodeja, (2) ephemeral znodeja eli hetkellisiä znodeja sekä (3) järjestysnumerolla varustettuja znodeja. Pysyvät znodet ovat olemassa vaikka kukaan ei olisi yhdistettynä niihin. Hetkelliset znodet ovat olemassa vain niin kauan kuin niihin ollaan yhteydessä. Viimeisen yhteyden poistuessaa tämä kyseinen hetkellinen node poistetaan, koska znodejen rakenne on hierarkkinen tämän takia hetkellisellä znodella ei voi olla alinodeja. Järjestysnumerolla varustetut znodet voivat olla joko pysyviä tai hetkellisiä. Tämä znode sisältää nimessään kymmennumeroisen järjestysnumeron, tämän znoden nimi pohjautuu hierarkiassa ylempänä olevaan nimeen, jonka perään lisätään tämä järjestysnumero. (Zookeeper - Fundamentals n.d.)



Kuvio 11. Zookeeper hierarkkia

Zookeeperissä pidetään yllä tietoa siitä, että missä mennään watchesien avulla. Watchit ovat käytännössä trappeja, joita käyttäjä voi asettaa. Kun watchi on asetettu znodelle saa käyttä ilmoituksen siitä, kun kyseisellä znodella muuttuu jotain. Yhteyden katketessa palvelimelle kyseinen ilmoitus ei lähde käyttäjälle vaan se jää paikalliseksi. (Apache Zookeeper n.d.)

Zookeeperissä on yksinkertainen API, jonka avulla on mahdollista tehdä muutoksia. Kaikki palvelimet voivat palvella käyttäjiä ja jos käyttäjä haluaa jotain tietoa, niin silloin tämä tieto toimitetaan paikallisesta järjestelmästä käyttäjälle. Zookeeper toimii johtaja seuraaja periaatteella, eli kaikki muutokset tehdään johtajaan, joka on määriteltynä (ks. Kuvio 12). Mikäli johtava palvelin vikaantuu, niin silloin Zookeeperin viestintä taso huolehtii uuden johtajan valitsemisesta. (Apache Zookeeper n.d.)



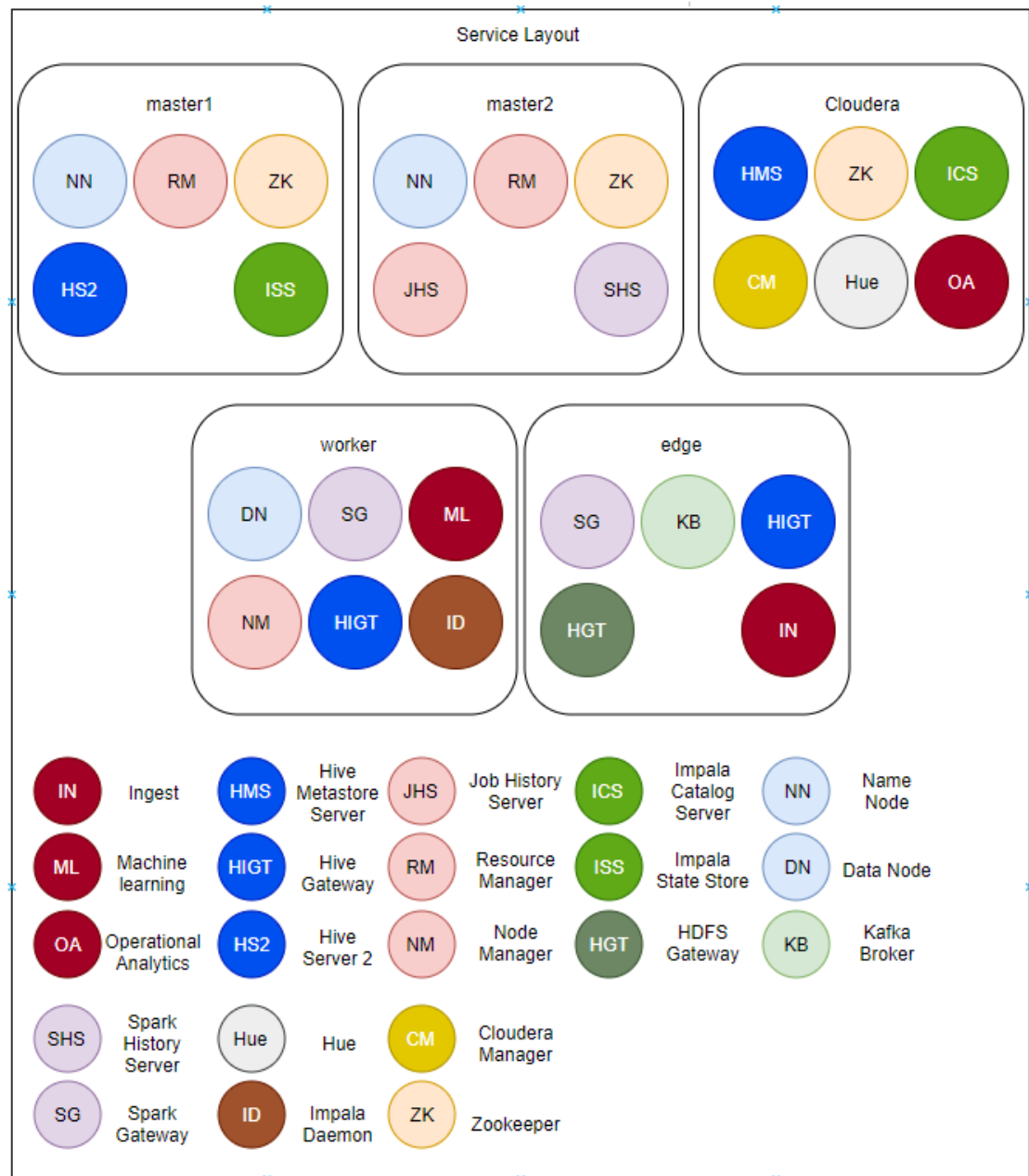
Kuvio 12. Zookeeper (Apache Zookeeper n.d.)

5.3 Apache-Spot rakenne

5.3.1 Yleistä

Apache Spot kokonaisuutena tarvitsee kaikki edellä mainitut osat sekä vielä neljä muuta osaa, jotka yhdistävät tämän kokonaisuuden. Se on avoimen lähdekoodin tuote ja tällä hetkellä uusin versio siitä on 1.0. Kuvio 13 nähdään Apache Spotin palvelut sekä miten ne ovat sijoitettu eri palvelimille.

Apache Spotin rakenne koostuu useasta Hadoopin osasta ja tämän päälle on asennettu Spotin omia osia. Kuvio 13 on rakennettu puhdas Hadoop-ympäristö, johon on asennettu Apache Spot. Ympäristö on jaettu viiteen eri nodeen, joista löytyy kaksi master nodea (name node), Cloudera Manager, worker (data node) ja edge node. Jokaisella näistä on oma rooli, ainoastaan master nodet ovat melkein samat. Apache Spotia voidaan ajaa myös ympäristössä, joka ei ole puhtaasti Hadoop-ympäristö, mutta tässä työssä keskitytään pelkästään tähän puhtaaseen ympäristöön. Kuvio 13 nähdään tummanpunaisella merkittynä Spotin omat palvelut, ne ovat ingest, ML sekä Operational Analytics (OA). Nämä palvelut asennetaan Cloudera Manager-, edge- sekä worker-nodelle.



Kuvio 13. Apache Spot Service Layout (Environment n.d.)

5.3.2 Apache Spot-ingest

Spot-ingestin tarkoituksena on siirtää tieto verkkotyökaluilta Hiveen, jotta se saadaan käyttäjälle tulkittavaksi. Kerääjiltä saatu tieto käsitellään työkaluilla kuten nfdump ja tshark, koska se ei ole helposti luettavissa. Tshark on wiresharkin versio, joka toimii komentoriviltä. Tieto tallennetaan kahteen eri paikkaan, Hiveen sekä HDFS:n. HDFS:ssä tieto on alkuperäisessä muodossaan, kun taas Hiveen siirrettäessä se

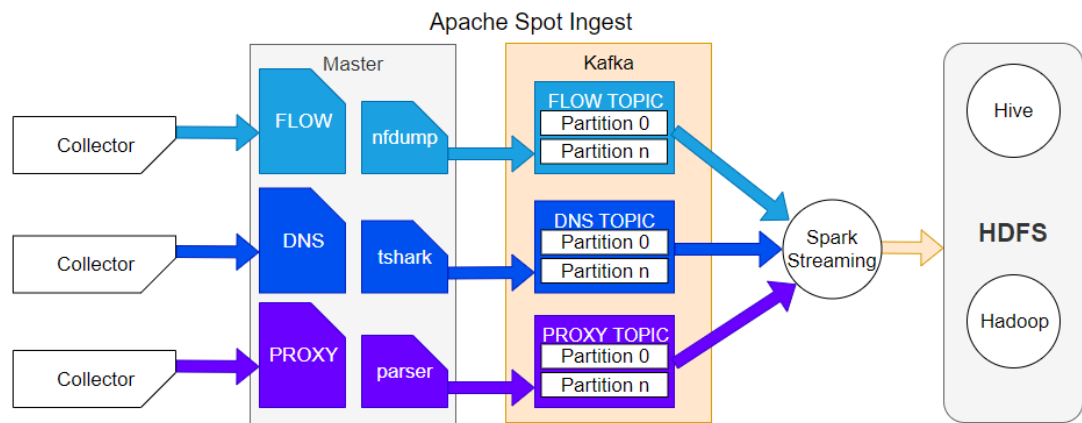
muunnetaan Avro-parquet muotoon. Tieto muunnetaan sen takia Hiveen siirrettävässä, että siihen voitaisiin tehdä SQL-kyselyitä. (Apache Spot Ingestion n.d.)

Tallennetulle tiedolle on kaksi eri mahdollisuutta ja se riippuu tiedon koosta. Yli 1MB kokoisista tiedostoista, lähetetään Kafkalle vain tiedon nimi sekä HDFS-sijainti. Mikäli tallennettua tietoa on alle 1 MB, niin kyseinen informaatio lähetetään suoraan Kafkalle ja tämä sen Sparkilla. (Apache Spot Ingestion n.d.)

Kafka tekee oman topicin jokaisesta uudesta tiedosta, jonka se saa. Kafka toimii niin sanotusti välikätenä eli kaikki tieto mitä Kafka saa kerääjiltä kerätään Kafkaan, jotta Spotista löytyvät nk. workerit pystyvät käsittelemään tämän datan. Kafka tekee partitiointia sen perusteella, että kuinka monta workeriä löytyy. (Apache Spot Ingestion n.d.)

Spotin workkerit käsittelevät datansa perustuen siihen mitä topicia ne käsittelevät ja mihin partitioon ne kuuluvat. Workerit käsittelevät tiedon ja lopulta tallentavat tämän Hiven taulukkoihin, josta ML algoritmit hakevat tiedon omaan käsittelyynsä. Spotin workereitä on kahta eri tyyliä: Pythoniin pohjautuvia workereitä sekä Spark-streaming workereitä. Pythonin workereiden toimenkuvaan kuuluu tiedon jäsentäminen hyödyntäen useampaa threadiä, kun taas Spark-streamingin workerit hyödyntävät Sparkia, lukemaan tieto Kafkalta. (Apache Spot Ingestion n.d.)

Kuvio 14 nähdään Spotin toiminta kuvattuna prosessikaaviona. Toiminnassa nähdään eri kerääjät, jotka toimittavat tietonsa masterille, josta tämä tieto välitetään Kafkalle ja siitä eteenpäin tallentaville järjestelmille.



Kuvio 14. Apache Spot ingest (Apache Spot Ingestion n.d.)

5.3.3 Apache Spot-ML

Spot-ml:n avulla tarkastellaan netflow-, proxy- sekä DNS-lokeja erilaisia ML ruutiineja vastaan. Tämän avulla pyritään löytämään viitteitä poikkeavuuksiin verkon toiminnassa. Käsitellystä tiedosta tehdään listat, jotka jaetaan kahteen eri kategoriaan. Toinen lista sisältää poikkeamat verkontoiminnassa, kun taas toinen sisältää normaalin verkkoliikenteen. Tämä jako perustuu todennäköisyyksiin verkkoliikenteessä. (Apache Spot Machine Learning n.d.)

Spot-ml käyttää analysointiin ingestin kautta saatua dataa. Analysointiin käytetään topic modeling mallia (Apache Spot Machine Learning n.d). Topic modelingillä tarkoitetaan dokumenttien analysointia, jossa pyritään löytämään analysoitavasta tiedosta saman tyyppisiä sanoja, jotka niputetaan eri otsikoiden alle (LDA Topic Modeling: An Explanation 2018). Spot-ml:ssä dokumentit muodostetaan IP-osoitteiden perusteella, ne sisältävät logit kyseisen IP-osoitteen toiminnasta verkossa. ML käyttää Latent Dirichlet Allocation (LDA) päättämään onko tämä toiminta epänormaalia kyseiselle IP:lle. Otsikot, joiden alle LDA sijoittaa IP-osoitteen toiminnan, pohjautuu normaaliin verkon toimintaan. LDA on todennäköisyyspohjainen malli, jota käytetään tiedon käsittelyyn ML:ssä. (Apache Spot Machine Learning n.d.) Spotin poikkeamien analy-

sointi tapahtuu ML:ssä valvomattomana, mutta käyttäjä voi vaikuttaa tähän analysointiin antamalla ML:lle palautetta sen tekemistä päätöksistä (Suspicious Connects Analysis n.d).

Jokainen logi tarkastellaan käyttäen eri parametrejä. ML käyttää eri taulukoita, riippuen mistä lähteestä kyseinen informaatio on saatu. Käytössä olevat taulukot ovat DNS-, netflow- sekä Proxy-tilukko (ks.Liite 1) (ks. Liite 2) (ks. Liite 3). (Suspicious Connects Analysis n.d.)

5.3.4 Apache Spot-OA

Spot-OA:n tarkoitus on muotoilla tieto niin että se voidaan näyttää käyttäjälle User Interfacessa (UI). Muokkaus tapahtuu työkalujen avulla, jotka tulevat OA:n mukana. Työkalut ovat Python pohjaisia, ne sisältävät toimenpiteet, joiden avulla ML:stä saatu tieto käsitellään käyttäjälle luettavaan muotoon. (Apache Spot (incubating) Operational Analytics 2015.)

Toimiakseen Spot-OA tarvitsee seuraavat komponentit, jotka pitää olla asennettuna kyseiseen palvelimeen:

- Python 2.7
- Spot-OA:n liitännäisten konfiguraatitiedosto täytyy olla muokattuna oikein.
- OA:n täytyy saada ML:n tulokset, mutta käyttäjä voi konfiguroida oman ML:n tuottamaan tämän tuloksen, hänen ei tarvitse käyttää Spot-ML:ää tähän.
- Spot-setup:in täytyy olla ajettuna, jotta Hivessä olisi oikeanlainen kanta. (Apache Spot (incubating) Operational Analytics 2015.)

5.3.5 Apache Spot-setup

Spot-setup osuus sisältää itsessään HDFS:n skriptin ja tämän avulla Spotin tietokannan perusta tehdään. Tämä kyseinen osuus ajetaan vain kerran, jotta kyseinen tietojärjestelmä saadaan pystyyn. Kyseinen skripti on riippuvainen Spot-conf-tiedostosta, josta se hakee tarvittavat tiedot asennusprosessiin. (Spot-setup 2016.)

6 Suunnitelma

6.1 Yleistä

Työn tutkimusmetodina käytettiin kvalitatiivista tutkimusotetta. Kvalitatiivinen tutkimusote tarkoittaa tutkimusmenetelmää, jossa keskitytään tutkimaan tietyn kohteen ominaisuuksia sekä laatua. (Laadullinen tutkimus 2015.) Apache Spotin tutkinnassa keskityttiin asennuksen helppouden sekä itse tuotteen toiminnan selvittämiseen. Kriteereinä tuotteen asennuksen helppoudelle olivat: (1) kuinka paljon aikaa kyseiseen asennukseen jouduttiin käyttämään, (2) onko tuotteen asennus intuitiivinen, (3) vika-tilanteen sattuessa kuinka helppo se on ratkaista. Tuotteen toimivuuden kriteereinä olivat: (1) kuinka paljon tuote nostaa vääriä hälytyksiä, (2) kuinka hyvin tuote havaitsee oikeata hättaliikennettä, (3) onko käyttäjän nähtävillä tarpeelliset tiedot.

6.2 Ympäristö

Hadoop pystytettiin RGCE:hen, jossa oli tarkoituksena testata miten Apache Spot havaitsee verkossa olevan haitallisen liikenteen niin livenä kuin sille syötetystä PCAP-tiedostosta. Apache Spotin oli tarkoitus antaa ensin livessä oppia verkon normaali toiminta. Tämän jälkeen oli tarkoitus generoida sinne hättaliikennettä sekä tarkastella miten tämä palvelu nostaa ne tarkastelijalle nähtäville.

Kaikki laitteet, joita työssä käytettiin olivat samassa verkossa. Näin pyrittiin vähentämään riskiä, että tuloksista puuttuisi tutkimukselle tärkeää tietoa ja näin tutkimuksen tulokset vääristyisivät. Käytetty verkko oli 10.110.3.0/24. Palvelinalustana oli tarkoitus toimia Ubuntu 16.04 Long Term Support (LTS), mutta ongelmien ilmettyä käyttöjärjestelmä vaihdettiin Centos 7.

6.3 Hyökkäykset

Työssä oli tarkoituksena ajaa kahta erilaista dataa Apache Spotille, joista toinen olisi sisältänyt hyökkäyksen ja toinen olisi sisältänyt puhdasta liikennettä. Molemmat oli tarkoitus ajaa livenä sekä PCAP-tiedostosta. Hyökkäykset oli tarkoitus ajaa kaksi kertaa siksi, että tavoitteena oli saada selville, kuinka hyvin tuote tunnistaa normaalista

liikenteestä haitallisen liikenteen. Tavoitteena oli myös selvittää kuinka paljon tämä nostaisi vääriä positiivisia tuloksia.

Tämän jälkeen oli tarkoitus syöttää Apache Spottiin normaalia liikennettä, joka ei olisi sisältänyt haittaliikennettä. Tämä olisi tehty, jotta olisi voitu tarkastella nostaako kyseinen palvelu vääriä hälytyksiä ja jouduttaisiinko näihin käyttämään turhia resursseja.

7 Toteutus

7.1 Hadoopin pystyttäminen

Työssä seurattiin Apache Spotin omaa dokumentaatiota palvelun asentamiseen sekä Clouderan 60 minuutin AWS-asennusohjeita Spotille, kun tarvittiin toista lähdettä vastaamaan heränneisiin kysymyksiin. Tässä työssä käytettiin käyttöjärjestelmänä Centos 7, koska uusin Clouderan managerin käyttötuki oli tälle versiolle Centosista. Kuvio 15 oli otettu SELinux sekä firewallit pois käytöstä, tämä tehtiin siksi, että asennuksesta saataisiin yksinkertaisempi. Tämä toistettiin kaikille palvelimille.

```
not running
[soluser@edge src]$ sestatus
SELinux status: disabled
[soluser@edge src]$ firewall-cmd --state
not running
[soluser@edge src]$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
[soluser@edge src]$
```

Kuvio 15. SELinux ja firewalld

Hadoopin asennus tapahtui asentamalla Cloudera Manager yhdelle näistä koneista, joka on nimetty Clouderaksi. Clouderan asennus tapahtui hakemalla järjestelmän sivulta binääritiedosto, jonka jälkeen tämä ajettiin. Cloudera Managerista käytettiin versiota 5.16.1. Jokainen kone tehtiin tietoisiksi toistaan laittamalla niiden IP-osoitteet hosts-tiedostoon (ks. Kuvio 16).

```

127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.110.3.191 master1
10.110.3.114 master2
10.110.3.184 Cloudera
10.110.3.138 worker
10.110.3.194 edge

```

Kuvio 16. /etc/hosts

Kun Cloudera oli asennettu, niin lisättiin kaikki koneet ylläpitoon ja määriteltiin niille roolit. Otettiin mallia dokumentaation suunnittelijoiden luomasta kuvasta. Kuvassa nähtiin, miten he olivat asettaneet palvelut eri nodeille. Kuvio 17 nähdään kaikki palvelut, jotka asennettiin eri palvelimille. Kuvio 18 nähdään, että täysin samanlaista asennusta kuin dokumentaatioissa oli, ei saatu, sillä Cloudera vaati tiettyjen palveluiden olevan samalla palvelimella. Se vaati myös osan palvelimista sisältävän muita palveluita, jotta kokonaisuus saatiin asennettua.

cloudera MANAGER Clusters ▾ Hosts ▾ Diagnostics ▾ Audits Charts ▾ Administration ▾

Roles

Clusters

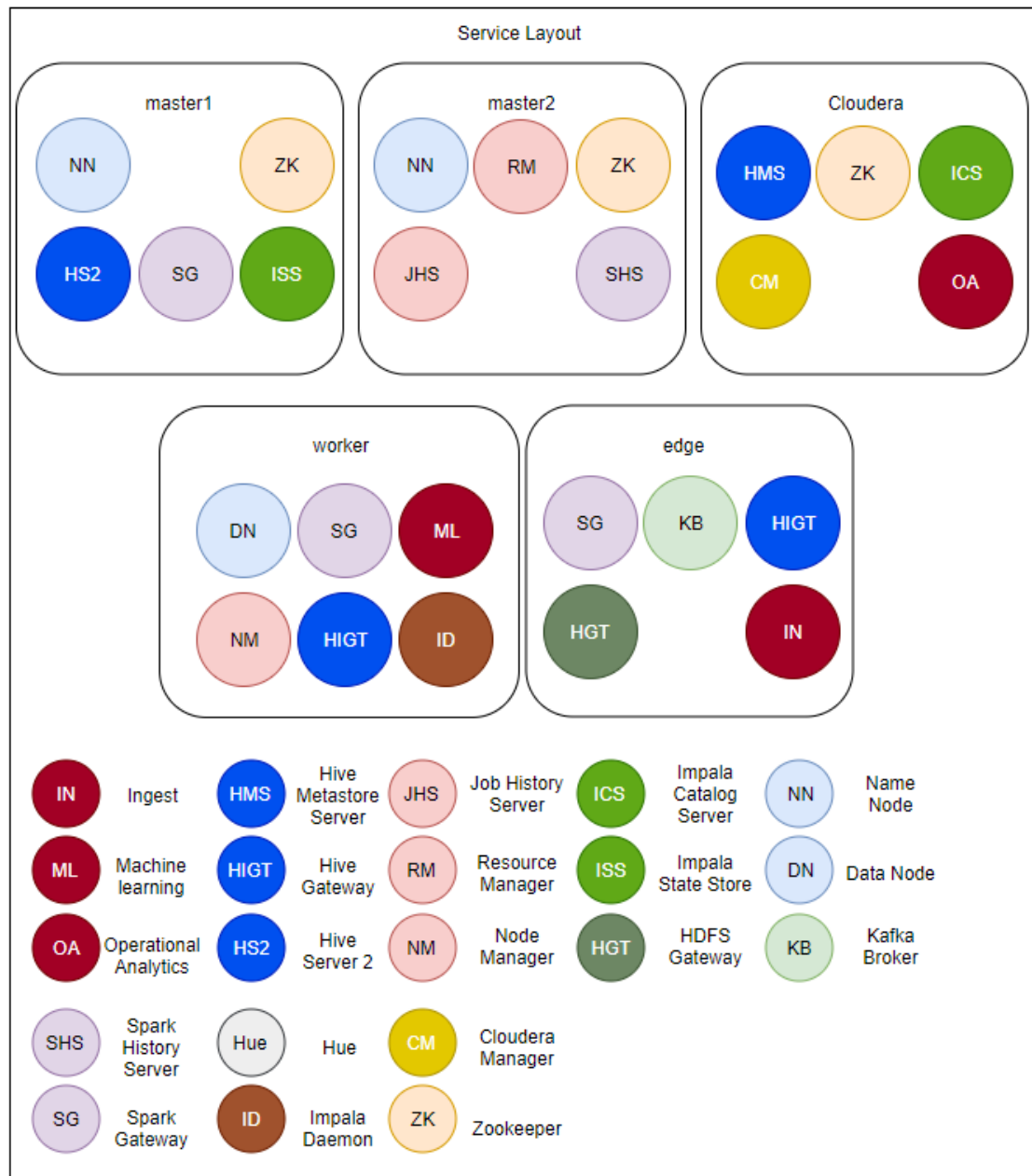
Cluster 1

Cluster 1

Hosts	Count	Roles
Cloudera	1	HMS ICS AP ES HM RM SM S
edge	1	NF... G KB G
master1	1	NN HS2 ISS G S
master2	1	SNN HS JHS RM S
worker	1	DN G ID G NM

This table is grouped by hosts having the same roles assigned to them.

Kuvio 17. Cloudera roles



Kuvio 18. Toteutettu service layout

7.2 Apache-Spotin pystyttäminen

Hadoop alustan pystyttämisen jälkeen lähdettiin pystyttämään Spottia. Jokaiselle nodelle tehtiin sama yksi käyttäjä, jolla oli Super User Priviledges (Sudo) sekä pääsy HDFS:lle. Edge-palvelimelle eli ML-nodelle tehtiin käyttäjälle salainen avain ja sen jälkeen lisättiin julkinen avain kaikille muille nodeille. Tämä oli vaatimuksena, että ML

pystyi toimimaan oikein. Tämä sama tehtiin myös Cloudera-palvelimelle eli UI-nodelle (ks. Kuvio 19).

```

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA5bIk08Cg5u8m9SsLZBwSe8PI13eHna9nm0yA9M/+kr1R/2HV
KJVbculv/y/hr+CIJtWtltp6beBU2IOt1xpTuKC7jbW6A475uT7BT9nQ0qPrMyTA
muOgBaJwl13N00YznwT8kys77Nf6u10rHgWbyuvAFff50XIlSRMABeydxQTaZpI
sKZD06Ev+qosjq41FV/abaMroL5ZhVg3zhL6auG2Vq3EHZ7jH38AIzSIlrVENrPE
OmfvhPI0IPNWn5/Us+GEXLZegRfTABGfBbxxWLzsS+v0bV1nY4Eu//pEiJ7Rt3mM
CnX8mLjAP03rR45eeuhy83QuzWnBWMoPegv4DQIDAQABAoIBAALHggq7DPdeY3FK
AQxIDq0ytEZvKtPsoGtISVMtBR6PbeOSMnsY05ALZRCIasmvBNa7dStDC1+y/DPW
fA8DQYoVX8+aKFMMstbb/Ijt00QLmbf3t0Dq6i/HRGeAehBCjDNbYxcAyevWvTDx
MqlDvGJBkgRyCLD6QZwENws23jd0plwmWNlStGkMZlGexlImX8220uqTJzILwikip
uJ/8wLso05ULFmkOqLnifYwBU9/V+HwCLB0DX57eQIPvebu3W8XJGdfm4Gajj04
WzVajuV/ZQP8h4ky8WHKsP8YzcUDPSzfIQHXIOFWVZ9Xt+n0ATKmpXwvLGy+JRbN
3ULN1aECgYEA+5s791daIokNXVxK8wseRHc57QXGzeLRCqJs3N8yBMCYJK/R8gEf
3rSDjb8DMJ32rZWeomEd0vmUZEzujGtoGkJ92ZZxHkMxKhHfxtf0detTMErPQtSN
Cfi9SNM9IphG9E8YJQBbTCMVfydyjyP+ZW0AFcpsjjghaeenDJ2zkkUCgYEA6bT1
uRUArNT0x/WBI+i0byMkHEC7rHJmn54yH039dYhnB7dt6fQe4qsnIUUMfOUaLODM
-----
~/.ssh/id_rsa
ssh-keygen -t rsa
ssh-copy-id soluser@master1
ssh-copy-id soluser@master2
ssh-copy-id soluser@worker
ssh-copy-id soluser@edge
ssh-copy-id soluser@cloudera

```

Kuvio 19. ID RSA public

Ladattiin Githubista Spotin konfiguraatitiedostot, joista sitten tämä kokonaisuus kasattiin. Tehtiin tarvittavat muokkaukset Spotin konfiguraatitiedostoon. Spotin sivuilla löytyi tieto niistä arvoista, jotka piti muokata. Muokkauksen jälkeen siirrettiin tämä tiedosto etc-kansioon sekä se kopioitiin ML- ja UI-nodelle, koska nämä tarvitsivat tämän saman tiedon. Spotin omassa dokumentaatiossa oli määritetty mitkä parametrit piti muokata. Kuvio 20 nähdään muutetut parametrit. Konfiguraatio tiedostoon täytyi määritellä nodet sekä mille käyttäjälle oli asennettu tämä kyseinen ohjelmisto. Nodejen konfiguraatioon käytettiin palvelinten nimiä ja käyttäjän poluksi määriteltiin soluserin kotihakemiston polku. Kyseiseen tiedostoon muokattiin vielä Sparkille annetut resurssit. Arvot piti laskea, sillä ne olivat riippuvaisia siitä kuinka paljon resursseja kyseisellä Spot ympäristöllä oli käytettävissä.

```
[soluser@worker ~]$ diff -y /etc/spot.conf /home/soluser/apache-spot-1.0-incubating/spot-setup/spot.conf
```

```
# Licensed to the Apache Software Foundation (ASF) under one
# contributor license agreements. See the NOTICE file distri
# this work for additional information regarding copyright ow
# The ASF licenses this file to You under the Apache License,
# (the "License"); you may not use this file except in compli
# the License. You may obtain a copy of the License at
```

```
# http://www.apache.org/licenses/LICENSE-2.0
```

```
# Unless required by applicable law or agreed to in writing,
# distributed under the License is distributed on an "AS IS"
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expres
# See the License for the specific language governing permiss
# limitations under the License.
```

```
#node configuration
UINODE='Cloudera'
MLNODE='worker'
GWNODE='edge'
DBNAME='scm'
```

```
#hdfs - base user and data source config
HUSER='/user/soluser'
NAME_NODE=''
WEB_PORT=50070
DNS_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
PROXY_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
FLOW_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
HPATH=${HUSER}/${DSOURCE}/scored_results/${FDATE}
```

```
#impala config
IMPALA_DEM=worker
IMPALA_PORT=25000
```

```
#local fs base user and data source config
LUSER='/home/soluser'
LPATH=${LUSER}/ml/${DSOURCE}/${FDATE}
RPATH=${LUSER}/ipython/user/${FDATE}
LIPATH=${LUSER}/ingest
```

```
#dns suspicious connects config
USER_DOMAIN=''
```

```
SPK_EXEC='6'
SPK_EXEC_MEM='8192m'
SPK_DRIVER_MEM='8192m'
SPK_DRIVER_MAX_RESULTS='1g'
SPK_EXEC_CORES='1'
SPK_DRIVER_MEM_OVERHEAD='819'
SPK_EXEC_MEM_OVERHEAD='819'
SPK_AUTO_BRDCAST_JOIN_THR='10485760'
```

```
LDA_OPTIMIZER=''
LDA_ALPHA=''
LDA_BETA=''
```

```
PRECISION='64'
TOL='1e-6'
TOPIC_COUNT=20
DUPFACTOR=1000
```

```
# Licensed to the Apache Software Foundation (ASF) under one
# contributor license agreements. See the NOTICE file distri
# this work for additional information regarding copyright ow
# The ASF licenses this file to You under the Apache License,
# (the "License"); you may not use this file except in compli
# the License. You may obtain a copy of the License at
```

```
# http://www.apache.org/licenses/LICENSE-2.0
```

```
# Unless required by applicable law or agreed to in writing,
# distributed under the License is distributed on an "AS IS"
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either expres
# See the License for the specific language governing permiss
# limitations under the License.
```

```
#node configuration
UINODE='node03'
MLNODE='node04'
GWNODE='node16'
DBNAME='spot'
```

```
#hdfs - base user and data source config
HUSER='/user/spot'
NAME_NODE=''
WEB_PORT=50070
DNS_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
PROXY_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
FLOW_PATH=${HUSER}/${DSOURCE}/hive/y=${YR}/m=${MH}/d=${DY}/
HPATH=${HUSER}/${DSOURCE}/scored_results/${FDATE}
```

```
#impala config
IMPALA_DEM=node04
IMPALA_PORT=21050
```

```
#local fs base user and data source config
LUSER='/home/spot'
LPATH=${LUSER}/ml/${DSOURCE}/${FDATE}
RPATH=${LUSER}/ipython/user/${FDATE}
LIPATH=${LUSER}/ingest
```

```
#dns suspicious connects config
USER_DOMAIN=''
```

```
SPK_EXEC=''
SPK_EXEC_MEM=''
SPK_DRIVER_MEM=''
SPK_DRIVER_MAX_RESULTS=''
SPK_EXEC_CORES=''
SPK_DRIVER_MEM_OVERHEAD=''
SPK_EXEC_MEM_OVERHEAD=''
SPK_AUTO_BRDCAST_JOIN_THR='10485760'
```

```
LDA_OPTIMIZER=''
LDA_ALPHA=''
LDA_BETA=''
```

```
PRECISION='64'
TOL='1e-6'
TOPIC_COUNT=20
DUPFACTOR=1000
```

Kuvio 20. Spot.conf

Käyttäjälle piti lisätä supergroup ryhmä, jotta se pystyi käsittelemään kyseisiä tiedostoja, jotka tallennettiin HDFS:lle. Tämä kyseinen ryhmä piti ensiksi luoda. Kun Spot.conf oli muokattu sekä supergroup-ryhmä oli lisätty kyseiselle käyttäjälle, niin sen jälkeen ajettiin skriptitiedosto nimeltä hdfs_setup, joka löytyi Spot-setup-kansioista. Kyseinen skripti loi Hiveen tietokannan sekä kansiot DNS:lle, proxyille sekä flow'ille. Varmistettiin että palvelimelta master1 löytyi kyseiset kansiot sekä että tietokanta löytyi worker-palvelimelta (ks. Kuvio 21).

The screenshot shows the Hadoop web interface at `master1:50070/explorer.html#/user/soluser`. The 'Browse Directory' view displays a table of files and directories in the HDFS. The table has columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The files listed are 'dns', 'flow', and 'proxy'. On the left, a terminal window shows the output of a 'SHOW TABLES' query in a Hive shell, listing various tables like 'dns', 'flow', 'proxy', etc. On the right, a small terminal window shows the command 'id soluser' and its output, confirming the user's identity and group membership.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxr-x	soluser	supergroup	0 B	Sat Dec 29 16:59:52 +0200 2018	0	0 B	dns
drwxrwxr-x	soluser	supergroup	0 B	Sat Dec 29 16:59:22 +0200 2018	0	0 B	flow
drwxrwxr-x	soluser	supergroup	0 B	Sat Dec 29 17:00:22 +0200 2018	0	0 B	proxy

Kuvio 21. HDFS-setup varmistus

Näiden konfiguraatioiden jälkeen siirryttiin asentamaan Spotin omia palveluita tämän rungon päälle.

7.2.1 Ingestin konfiguraatio

Ingestin asennus oli suoraviivainen. Tehtiin kansio, johon haettiin tuotteiden lähteet, joita ei saatu suoraan pakettienhallinnointityökaluilta, kuten pip:ltä. Lähteiden hake-
misen jälkeen asennettiin ne. Päivitetyimmän ohjeen mukaan automake, joka latautui
nfdumpin-skriptin avulla, oli vanha. Tämän takia, ennen sen ajamista, ladattiin päivi-
tetympi versio siitä. Alkuperäisessä dokumentaatioissa oleva linkki tsharkille ei toimi-
nut enää, joten uusi linkki otettiin päivitetyimmästä ohjeesta. Kuvio 22 todennettiin,
että ingestiltä löytyi nyt tarvittavat työkalut.


```
[soluser@edge src]$ tshark --version
TShark (Wireshark) 2.2.9 (wireshark-2.2.9)

Copyright 1998-2017 Gerald Combs <gerald@wireshark.org> and contributors.
License GPLv2+: GNU GPL version 2 or later <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled (64-bit) with libpcap, without POSIX capabilities, without libnl, with
Glib 2.56.1, with zlib 1.2.7, without SMI, without c-ares, without Lua, without
GnuTLS, without Gcrypt, with MIT Kerberos, without GeoIP.

Running on Linux 3.10.0-327.el7.x86_64, with locale LC_CTYPE=en_US.UTF-8,
LC_NUMERIC=fi_FI.UTF-8, LC_TIME=fi_FI.UTF-8, LC_COLLATE=en_US.UTF-8,
LC_MONETARY=fi_FI.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=fi_FI.UTF-8,
LC_NAME=fi_FI.UTF-8, LC_ADDRESS=fi_FI.UTF-8, LC_TELEPHONE=fi_FI.UTF-8,
LC_MEASUREMENT=fi_FI.UTF-8, LC_IDENTIFICATION=fi_FI.UTF-8, with libpcap version
1.5.3, with zlib 1.2.7.
      Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz (with SSE4.2)

Built using gcc 4.8.5 20150623 (Red Hat 4.8.5-36).
[soluser@edge src]$ automake --version
automake (GNU automake) 1.14
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv2+: GNU GPL version 2 or later <http://gnu.org/licenses/gpl-2.0.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Tom Tromey <tromey@redhat.com>
      and Alexandre Duret-Lutz <adl@gnu.org>.
[soluser@edge src]$ pip --version
pip 18.1 from /usr/lib/python2.7/site-packages/pip (python 2.7)
[soluser@edge src]$
```

Kuvio 22. Ingest riippuvuuksien versiot

Seuraavaksi muokattiin ingestin omaa konfiguraatiotiedostoa, jonka avulla se käynnistää nämä prosessit. Tarkoituksemme oli tässä keskittyä pelkästään flow'n sekä pcap-tiedoston tutkimiseen. Joten ainoat asiat mitä tähän tiedostoon merkittiin, oli tietokannan nimi, HDFS:n polku, prosessien määrä sekä intervalli ja kafkan tarvitsemat tiedot. (ks. Kuvio 23)

```

{
  "dbname" : "scm",
  "hdfs_app_path" : "/user/soluser",
  "collector_processes":5,
  "ingestion_interval":1,
  "spark-streaming":{
    "driver_memory":"",
    "spark_exec":"",
    "spark_executor_memory":"",
    "spark_executor_cores":"",
    "spark_batch_size":""
  },
  "kafka":{
    "kafka_server":"10.110.3[REDACTED]",
    "kafka_port":"9092",
    "zookeeper_server":"10.110.3[REDACTED]",
    "zookeeper_port":"2181",
    "message_size":900000
  },

```

Kuvio 23. Ingestion konfiguraatiotiedosto

7.2.2 ML:n konfiguraatio

ML:n konfiguraatio tapahtui seuraavasti. Githubista ladusta kansioista kopioitiin ML:n oma osuus worker-palvelimelle ja sinne tehtiin oma lähdekansio. Tänne ladattiin riippuvuudet, joita kyseinen alusta tarvitsi. Ladattiin ja asennettiin Scalan Simple build tool (sbt). Spotin oma dokumentaatio viittasi linkkiin, jonka takana oli useampia eri versioita kyseiselle työkalulle. Kokeiltiin useampaa eri versiota. Lopulta saatiin version, jolla Scala saatiin rakennettua.

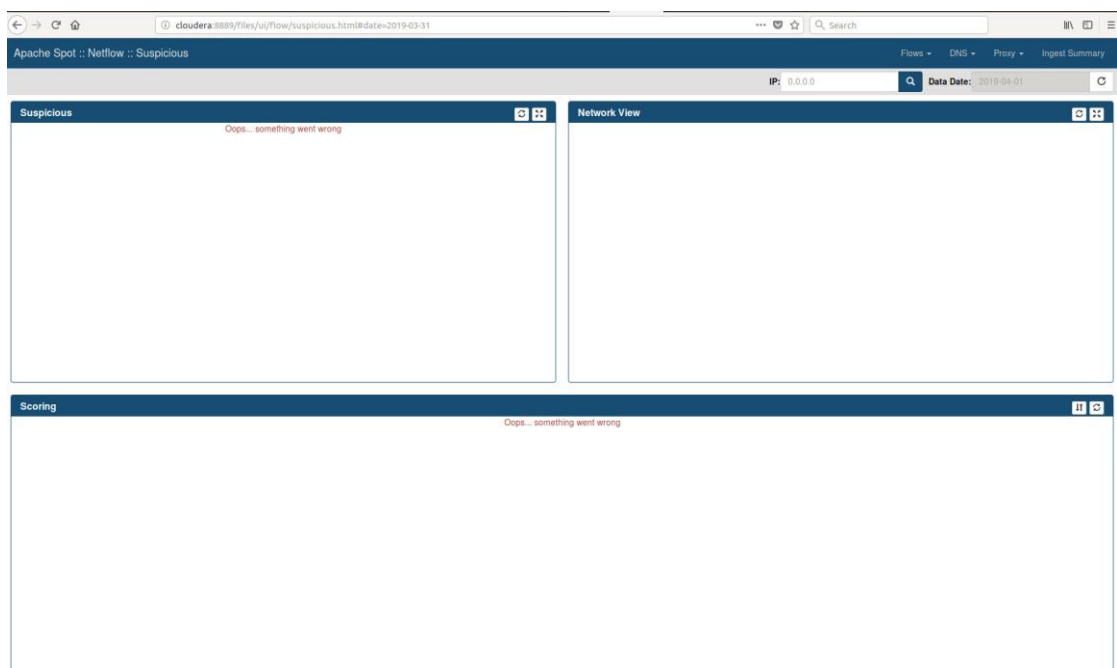
7.2.3 OA:n konfiguraatio

Samalla tavalla kuin ML:ssä niin OA:n asennuksessakin kopioitiin OA:n oma kansio vastuulliselle nodelle eli Clouderalle. Tämän jälkeen tehtiin konfiguraatiotiedostoihin tarvittavat muutokset, jotta saatiin ML:n tuottamat lopputulokset OA:lle. Kuvio 24 nähdään, että käyttöön valittiin impalan tiedon käsittelijäksi, sekä OA:lle kerrottiin, että missä kyseinen prosessi sijaitsee (ks. Kuvio 24).

```
[root@cloudera spot-oa]# cat oa/components/data/engine.json
{
  "oa_data_engine": "impala",
  "impala": {
    "impala_daemon": "worker"
  },
  "hive": {}
}
```

Kuvio 24. engine impala

Jäljellä oli vielä kaksi riippuvuutta, jotka täytyi asentaa, browserify sekä uglify-js. Nämä asennettiin Node.js:n paketinhallintajärjestelmällä ja tämän jälkeen vielä ajettiin Node Package Manager (npm) install, joka asensi loput riippuvuuksista. Lopulta kun kaikki oli asennettu, käynnistettiin web-palvelin. (ks. Kuvio 25)



Kuvio 25. Web-palvelin käynnistetty

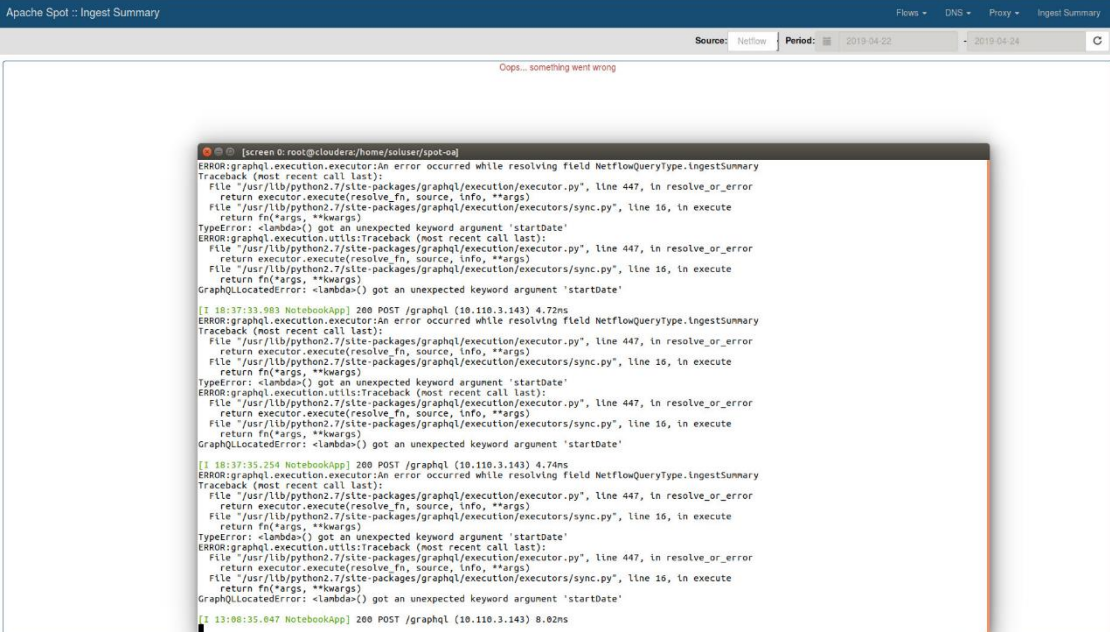
8 Testaus ja tulokset

8.1 Testaus

Itse tuotteen testaukseen asti ei päästy, koska vaikka tuotteen dokumentaatiota seurattiin ja yritettiin korjata vastaan tulleita virheitä, niin siltikään tuotetta ei saatu toiminta kuntoon. Dokumentaation puutteellisuus sekä versionhallinta olivat suurimmat tekijät, siihen että kyseistä tuotetta ei saatu asennettua toimivaksi.

8.2 Tulokset

Alkuperäinen suunnitelma oli asentaa kyseinen tuote Ubuntuille, mutta asennuksen jälkeen huomattiin, että tuote ei ollut asentunut oikein. Tuotteen debuggaus oli todella hankalaa, koska se koostuu niin monesta komponentista. Tämän jälkeen siirryttiin kokeilemaan asennusta Centosille, koska dokumentaatio näytti olevan Centosille tehty. Käyttöjärjestelmän vaihtaminen ei edesauttanut työtä. Asennus oli yhtä vaikeaa, kuin Ubuntuillakin. Nähtiin, että web-palvelin saa kyllä kyselyt, jotka tehtiin käyttöliittymän kautta, mutta näitä HTTP POST-paketteja tutkittaessa huomattiin, että osa koodista ei toimi oikein (ks. Kuvio 26).



```

Apache Spot :: Ingest Summary
Source: Netflow Period: 2019-04-22 2019-04-24

Oops... something went wrong

[screen 0: root@clouders:/home/soluser/spot-qa]
ERROR:graphql.execution.executor:An error occurred while resolving field NetFlowQueryType.IngestSummary
Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
TypeError: <lambda>() got an unexpected keyword argument 'startDate'
ERROR:graphql.execution.utils:Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
GraphQLLocatedError: <lambda>() got an unexpected keyword argument 'startDate'

[18:37:33.983 NotebookApp] 200 POST /graphql (10.110.3.143) 4.72ms
ERROR:graphql.execution.executor:An error occurred while resolving field NetFlowQueryType.IngestSummary
Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
TypeError: <lambda>() got an unexpected keyword argument 'startDate'
ERROR:graphql.execution.utils:Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
GraphQLLocatedError: <lambda>() got an unexpected keyword argument 'startDate'

[18:37:35.254 NotebookApp] 200 POST /graphql (10.110.3.143) 4.74ms
Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
TypeError: <lambda>() got an unexpected keyword argument 'startDate'
ERROR:graphql.execution.utils:Traceback (most recent call last):
  File "/usr/lib/python2.7/site-packages/graphql/execution/executor.py", line 447, in resolve_or_error
    return executor.execute(resolve_fn, source, info, **args)
  File "/usr/lib/python2.7/site-packages/graphql/execution/executors/sync.py", line 16, in execute
    return fn(*args, **kwargs)
GraphQLLocatedError: <lambda>() got an unexpected keyword argument 'startDate'

[13:08:35.047 NotebookApp] 200 POST /graphql (10.110.3.143) 8.02ms

```

Kuvio 26. Apache Spot lopputulos

Tuotteen asennuksessa tuli todella paljon ongelmia, koska dokumentaation oli todella epäselvä. Heti dokumentaation alussa ilmoitettiin että, näiden Hadoopin lisäosien täytyy olla vähintään asennettuna, mutta missään tilanteessa ei otettu kantaa muiden lisäosien versioihin paitsi Sparkin. Versioiden kontrollointi koko dokumentaation osalta oli lähestulkoon olematonta. Yksi isoimmista ongelma tilanteista tuli, kun ML-nodelle täytyi asentaa sbt. Sbt:ssä oli kaksi eri haaraa, versiot 1.x ja 0.x. Missään vaiheessa ei kerrottu, että kumpi näistä tulisi asentaa ja mikä näistä voisi toimia. Versioita oli yhteensä yli 20. Tapa, miten toimiva versio saatiin, oli kokeilemalla molempia eri versiohaaroja. Apuna tämän selvittämisessä käytettiin webarchivea, jonka avulla koitettiin eliminoida, mitä eri versiota dokumentaation aikana oli julkaistu.

Kun Spotin konfiguraatiodokumentit ja lähdettiin editoimaan, niin dokumentaatioissa ei välttämättä kerrottu polkua tiedostoon, jota piti editoida, vaan se piti yksinkertaisesti grepata. Spotin dokumentaatioissa oli Kuvio 27 löytyvä komento. Mikäli tuo scp-komento kopioitiin suoraan päätteeseen, muuttaen arvot vastaamaan oikeata palvelinta sekä käyttäjää se ei toiminut. Tämä johtui siitä, että siinä oli neljä eri komentoa putkeen, mutta niitä ei ole eroteltu toisistaan.

```
[root@edge apache-spot-1.0-incubating]# scp -r spot-ml worker:/home/soluser/.
ssh worker mv spot-ml ml cd /home/soluser /ml
/ml: No such file or directory
[root@edge apache-spot-1.0-incubating]# ls
DISCLAIMER          LICENSE              README.md           spot-oa
docs                 LICENSE-topojson.txt spot-ingest         spot-setup
flow.collector.out  NOTICE              _                  spot-ml
```

Kuvio 27. ML-dokumentaatio

Toisena ohjelähteenä käytettiin Cloudera-sivuilta löytyvää ohjetta Spotille, jossa tuote oli asennettu AWS:n. Ongelmana oli, että tämä palveluiden sijoittelu ei vassannut Spotin omilta sivuilta löytyvää dokumentaatiota.

Alkuperäisen dokumentaation palveluiden sijoittelussa nähdään HUE-palvelu, mutta missään vaiheessa sen asentamisesta ei mainittu mitään. Kaikki Spotin Githubin tiedostot grepattiin eikä, löytynyt mitään referenssiä kyseiseen palveluun. Tuotteen tutkiminen loppui tähän, koska tämän palvelun debuggaaminen oli aivan liian iso haaste.

9 Pohdinta ja johtopäätökset

9.1 Johtopäätökset

Apache Spot on avoimenlähdekoodin projekti, jonka tarkoituksena on edistää yritysten tietoturvaa. Projektina tämä tuote oli todella mielenkiintoinen, mutta tällä hetkellä tuote ei ole samalla tasolla kuin maksulliset tuotteet. Henkilöt, jotka ovat vastuussa tästä projektista, eivät olleet ylläpitäneet omaa dokumentaatiotaan. Siellä oli useampia virheitä. Asennusyrityksen jälkeen olen vakuuttunut siitä, että he eivät ole testanneet asennusta itse seuraamalla heidän omaa dokumentaatiotaan. Muutamat linkit tuotteen dokumentaatioissa olivat rikkiäisiä tai sitten linkkien takana olevat tuotteet olivat päivittyneet niin, että ne eivät olleet suoraan yhteensopivia Spotin kanssa.

Henkilö, joka asentaa Spotia, joutuu debugaamaan tuotetta heti lähtötilanteessa, sillä hänen täytyy selvittää, mitkä eri versiot tuotteista ovat yhteensopivia. Dokumentaatioissa oli todella paljon kohtia, joissa käytettiin eri nimiä ja tämä aiheuttaa turhaa hämmennystä asentajalle. Esimerkkinä heidän omassa service layout-kuvassa he puhuvat palvelimesta, jolle asennetaan ML:ä worker-palvelimena, mutta myöhemmin sitä kutsutaan nimellä ML-node.

9.2 Pohdinta

Työn tarkoituksena oli tutkia miten netflow sekä pakettien analysoinnista voidaan saada tehokas työkalu tietoverkkoja puolustavalle taholle. Tämä pyrittiin selvittämään Apache Spotin avulla. Miten kyseinen palvelu edistää yritysten kyberturvallisuutta. Lopputulos, johon päästiin oli se, että tuotetta ei saatu asennettua. Apache Spot oli tällä hetkellä versiossa 1.0, eikä ole saanut päivitystä

noin kahteen vuoteen. Kun käy tarkistamassa kyseisen palvelun JIRA, nähtiin että siellä kehitettiin kokoajan kyseistä palvelua, mutta vielä ei ollut mitään konkreettista päivitysaikataulua. Tämän hetkinen tuote oli aivan liian raskas asennettavaksi, mikäli tuotteen asennusta saadaan helpotettua on sillä lupaava tulevaisuus kyberturvallisuudessa. Tällä hetkellä Hortonworksiltä löytyy samanlainen tuote kuin mitä Spot on. Nopeasti selaamalla se vaikutti huomattavasti yksinkertaisemmalta asentaa.

Tämän opinnäytetyön pohjalta opin, kuinka tärkeää on luoda hyvä dokumentaatio käyttäjilleen sekä huolehtia versionhallinnasta, jotta tuotteet saadaan asennettua niin kuin ne pitäisikin saada. Työ opetti kyberturvallisuudesta todella paljon ja myös siitä, että kun dokumentaation on kirjoittanut, täytyy se testata toimivaksi ja ymmärrettäväksi, juuri niin kuin se on kirjoitettu. Näin voidaan varmistua onnistuneesta lopputuloksesta. On todella vaikea sanoa että missä vaiheessa työtä menttiin pieleen, koska tämä muodostui niin monesta kokonaisuudesta. Voi olla, että vikaan menttiin vasta viime metreillä tai heti alussa. Tiedonhakua olisi voinut parantaa. Sillä tietoa joutui hakemaan jälkeinpäin. Mikäli tuotteen asennus ei olisi ollut näin hankala olisi aikaa kulunut vähemmän tämän tuotteen kasaamiseen ja enemmän hyökkäysten sekä tulosten analysointiin. Jälkeinpäin ajateltuna olisi ollut varmasti paljon helpompaa selvittää sbt oikea version greppaamalla tiedostoja, mutta silloin ei ollut tiedossa että missä formaatissa kyseinen tieto on. Tuotetta yritettiin asentaa useamman henkilön voimin (ohjelmoija, palvelinasiantuntija) mutta tällöinkään tuotteen asennus ei onnistunut. Tuotteen asennuksessa haettiin apua Apache Spotin omalta slack-kanavalta, mutta sieltä ei saatu vastausta. Taidot, jotka otan tästä opinnäytetyöstäni mukaan ovat solveltaminen, debuggaaminen sekä pitkäjänteisyys.

Työni oli siinä mielessä onnistunut että saatiin tämän hetkinen vastaus siihen että onko Apache Spotista tällä hetkellä suojaamaan yrityksiä kyberhyökkäyksiltä. Vastaus on ei. Syy on yksinkertaisuudessaan se, että jos pelkkä tuotteen asentaminen vie näin paljon resursseja, niin mitä tapahtuu siinä vaiheessa, kun alustaa tai itse tuotetta pitää lähteä päivittämään. Toisaalta työ epäonnistui siinä, että itse tuotteen

ominaisuuksia ei päästy tarkastelemaan oikeassa ympäristössä ja varsinkaan hyökkäysten kera.

Lähteet

An Implementation of Intrusion Detection System Using Genetic Algorithm. 2012. Artikkel Arxiv-sivustolla. Viitattu 12.4.2019.
<https://arxiv.org/ftp/arxiv/papers/1204/1204.1336.pdf>

An Introduction to IDS. 2001. Artikkel Symantec-sivustolla. Viitattu 18.4.2019.
<https://www.symantec.com/connect/articles/introduction-ids>

Apache Hadoop. N.d. Artikkel Apache Hadoop-sivustolla. Viitattu 10.9.2017.
<https://hadoop.apache.org/>

Apache Kafka Topic – Architecture & Partitions. 2018. Artikkel Data-flair-sivustolla. Viitattu 2.5.2019. <https://data-flair.training/blogs/kafka-topic-architecture/>

Apache Spark. N.d. Dokumentaatio Gigaspaces-sivustolla. Viitattu 22.4.2019.
<https://docs.gigaspaces.com/latest/overview/apache-spark.html>

Apache Spark - Introduction N.d. Tutorialspoint-sivusto. Viitattu 27.4.2019.
https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

Apache Spot (incubating) Operational Analytics. 2015. Dokumentaatio Github-sivustolla. Viitattu 27.10.2018. <https://github.com/apache/incubator-spot/blob/master/spot-oa/oa/INSTALL.md>

Apache Spot (Incubating). 2015. Dokumentaatio Github-sivustolla. Viitattu 2.9.2017.
<https://github.com/apache/incubator-spot>

Apache Spot Ingestion. N.d. Artikkel Apache spot-sivustolla. Viitattu 27.10.2018.
<https://spot.incubator.apache.org/project-components/ingestion/>

Apache Spot Machine Learning. N.d. Artikkel Apache spot-sivustolla. Viitattu 27.10.2018. <https://spot.incubator.apache.org/project-components/machine-learning/>

Apache YARN. N.d. Artikkel Apache YARN-sivustolla. Viitattu. 5.9.2018.
<https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html>

Apache Zookeeper. N.d. Artikkel Apache Zookeeper-sivustolla. Viitattu 18.9.2018.
<https://zookeeper.apache.org/doc/current/zookeeperOver.html>

Bejtlich, R. 2013. The practice of network security monitoring : understanding incident detection and response. 1. p. No Starch Press.

Chapter 15. Nmap Reference Guide. N.d. Dokumentaatio Nmap-sivustolla. Viitattu 2.5.2019. <https://nmap.org/book/man.html>

Cole, E. 2009. Network Security Bible, 2. p. Wiley Publishing, Inc.

Effective Analysis on Remote to User (R2L) Attacks Using Random Forest Algorithm. 2014 Artikkel Academia-sivustolla. Viitattu 21.4.2019.

https://www.academia.edu/7314391/Effective_Analysis_on_Remote_to_User_R2L_Attacks_Using_Random_Forest_Algorithm

Environment. N.d. Dokumentaatio Apache Spott-sivustolla. Viitattu 2.9.2017
<https://spot.incubator.apache.org/doc/>

GraphX Programming guide. N.d. Artikkelin Apache Spark-sivustolla. Viitattu 22.4.2019. <https://spark.apache.org/docs/latest/graphx-programming-guide.html#vertex-and-edge-rdds>

Hadoop - MapReduce. N.d. Tutorialspoint-sivustolla. Viitattu 20.4.2019.
https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

Hadoop YARN Tutorial – Learn the Fundamentals of YARN Architecture. 2018. Viitattu 20.4.2019. <https://www.edureka.co/blog/hadoop-yarn-tutorial/#Introduction%20to%20Hadoop%20YARN>

HDFS Architecture Guide. 2018. Dokumentaatio Apache Hadoop-sivustolla. Viitattu 24.4.2019. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Historia. N.d. Artikkelin Nixu-sivustolla. Viitattu 2.9.2017.
<https://www.nixu.com/fi/nixu-oyj/historia>

Hive-Tutorial. 2018. Artikkelin Edureka-sivustolla. Viitattu 23.4.2019.
<https://www.edureka.co/blog/hive-tutorial/>

Impala: A Modern, Open-Source SQL Engine for Hadoop. N.d. Artikkelin CIDR-sivustolla. Viitattu 18.9.2018.
http://cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf

Introduction. N.d. Artikkelin Apache Kafka-sivustolla. Viitattu 18.9.2018.
<https://kafka.apache.org/intro>

Karau, H., Konwinski, A., Wendell, P. & Zaharia, M. 2015. Learning Spark Lightning-Fast Data Analysis. 1. p. O'Reilly

Laadullinen tutkimus, 2015. Artikkelin Jyväskylän yliopiston sivulla, Viitattu 13.4.2019.
<https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/laadullinen-tutkimus>

LDA Topic Modeling: An Explanation. 2018. Artikkelin Towards Data Science-sivustolla. Viitattu 3.5.2019. <https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd>

Lucas, M. 2010. Network Flow Analysis. 1. p. No Starch Press.

MapReduce Tutorial. 2018. Artikkelin Edureka-sivustolla. Viitattu 21.4.2019.
<https://www.edureka.co/blog/mapreduce-tutorial/>

Overview. N.d. Artikkelin Apache Impala-sivustolla. Viitattu 18.9.2018.
<https://impala.apache.org/overview.html>

Privilege escalation. N.d. Artikkelin Azeria-sivustolla. Viitattu 21.4.2019. <https://azeria-labs.com/privilege-escalation/>

SebastianZ. 2013. Security 1:1 - Part 3 - Various types of network attacks. Artikkelin Symantec-sivustolla. Viitattu 17.4.2019. <https://www.symantec.com/connect/articles/security-11-part-3-various-types-network-attacks>

Spot-setup. 2016. Dokumentaatio Github-sivustolla. Viitattu 10.11.2018. <https://github.com/apache/incubator-spot/tree/master/spot-setup>

Suspicious Connects Analysis. N.d. Artikkelin Apache spot-sivustolla. Viitattu 27.10.2018. <https://spot.incubator.apache.org/project-components/suspicious-connects-analysis/>

The Sliding Scale of Cyber Security. 2015. Artikkelin SANS-sivustolla. Viitattu 8.1.2018. <https://web.archive.org/web/20180607155315/https://www.sans.org/reading-room/whitepapers/analyst/sliding-scale-cyber-security-36240>

Tietoa meistä. N.d. Artikkelin Jyvsectec-sivustolla. Viitattu 2.9.2017. <https://web.archive.org/web/20160814042337/http://jyvsectec.fi/fi/tietoa-meista/>

Traffic Analysis for Network Security: Two Approaches for Going Beyond Network Flow Data. 2016. Artikkelin Carnegie Mellon University-sivustolla. Viitattu 22.4.2019. https://insights.sei.cmu.edu/sei_blog/2016/09/traffic-analysis-for-network-security-two-approaches-for-going-beyond-network-flow-data.html

What Are the Most Common Cyberattacks? N.d. Artikkelin Cisco-sivustolla. Viitattu 21.4.2019. <https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html>

What Is a SIEM? 2016. Artikkelin Tripwire-sivustolla. Viitattu 2.9.2017. <https://www.tripwire.com/state-of-security/incident-detection/log-management-siem/what-is-a-siem/>

What is Big Data? N.d. Artikkelin Oracle-sivustolla. Viitattu 23.4.2019. <https://www.oracle.com/big-data/guide/what-is-big-data.html>

White, T. 2015. Hadoop The Definitive Guide Storage and Analysis At Internet Scale. 4. p. O'Reilly.

Why You Should Start Leveraging Network Flow Data Before the Next Big Breach. 2018. Artikkelin SecurityIntelligence-sivustolla. Viitattu 24.4.2019. <https://securityintelligence.com/why-you-should-start-leveraging-network-flow-data-before-the-next-big-breach/>

YARN's application master in Hadoop N.d. Dummies-sivustolla. Viitattu 27.4.2019 <https://www.dummies.com/programming/big-data/hadoop/yarns-application-master-in-hadoop/>

Zookeeper - Fundamentals. N.d. Artikkelin Tutorial-sivustolla. Viitattu 21.4.2019. https://www.tutorialspoint.com/zookeeper/zookeeper_fundamentals.htm

Liitteet

Liite 1. Flow'n matriisi (Supsicious Connects Analysis n.d.)

Flown suunta	<p>Lähde- ja kohdeporttien ollessa nolla. Tätä ominaisuutta ei oteta huomioon kumpaankaan dokumenttiin. Sama päätee myös silloin, kun molemmat tai ei kumpinkaan omaa portille arvoa, joka on alle 1025.</p> <p>Jos vain toinen porteista on nolla, niin tätä ei oteta huomioon sen puolen dokumentissa, mutta toisessa dokumentissa merkitään tähän kohtaan arvo -1. Tämä arvo annetaan myös silloin, kun vain toinen porteista on alle 1025, mutta ei kuitenkaan nolla. Arvo merkitään tämän dokumenttiin joka on alle 1025 ja toiseen dokumenttiin tätä ei merkitä.</p>
Portit	<p>Molempien porttien ollessa nolla. Annetaan nolla arvoksi molempiin lähde- ja kohdedokumentteihin.</p> <p>Vain toisen portin ollessa suurempi kuin nolla. Annetaan tämän portin arvo molempiin dokumentteihin.</p> <p>Vain toisen porteista ollessa suurempi kuin nolla ja pienempi kuin</p>

	<p>1025. Annetaan kyseisen portin arvo molempiin dokumentteihin.</p> <p>Molempien porttien ollessa pienempiä kuin 1025, mutta suurempia kuin nolla. Annetaan molempiin dokumentteihin arvo "111111".</p> <p>Molempien porttien ollessa isompia kuin 1025. Annetaan molempiin dokumentteihin arvo "333333".</p>
Protokolla	Käytetään log-tiedostosta löytyvää jonoa muokkaamattomana.
Kello	Merkitään tunteina
Koko	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot jaoteltu 0, 1, 2, 4, 8...
Pakettien määrä	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot jaoteltu 0, 1, 2, 4, 8...

Liite 2 DNS-matriisi (Supsicious Connects Analysis n.d.)

DNS nimi kysely	<p>Jos DNS-nimi kuuluu Alexan ensimmäisen miljoonan joukkoon käytä numeroa 1</p> <p>DNS-nimen kuullessa käyttäjän omaan domainiin, käytä numeroa 2</p> <p>Muuten käytä numeroa 0</p>
Framen pituus	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot jaoteltu 0, 1, 2, 4, 8...
Kello	Merkitään vain tunnit
Alidomainin pituus	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot jaoteltu 0, 1, 2, 4, 8...
Alidomainin nimen entropia	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot on jaoteltu: 0.0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3.0, 3.3, 3.6, 3.9, 4.2, 4.5, 4.8, 5.1, 5.4, 20
Kuinka monta pistettä alidomainissa on?	Käytä sarjan binääristä arvoa mihin kyseinen framen pituus osuu. Arvot jaoteltu: 0, 1, 2, 4, 8...

DNS kyselyn tyyppi	Käytetään log-tiedostosta löytyvää jonoa muokkaamattomana.
DNS kyselyn vastaus	Käytetään log-tiedostosta löytyvää jonoa muokkaamattomana.

Liite 3. Proxy-matriisi (Suspicious Connects Analysis n.d.)

DNS-nimi kysely	<p>Jos DNS-nimi kuuluu Alexan ensimmäisen miljoonan joukkoon käytä numeroa 1</p> <p>DNS-nimen kuullessa käyttäjän omaan domainiin, käytä numeroa 2</p> <p>Muuten käytä numeroa 0</p>
Kello	Merkittään vain tunnit
Pyyntötapa	Käytetään log-tiedostosta löytyvää pyyntötapaa (Get, Request, jne.)
Uniform Resource Identifier (URI) entropia	<p>Käytä numeroa 0-18. Numerot vastaavat seuraavia entropian arvoja:</p> <p>0.0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3.0, 3.3, 3.6, 3.9, 4.2, 4.5, 4.8, 5.1, 5.4, 20</p>
Haun sisällön tyyppi	Merkitse haun sisällön tyyppi log-tiedostosta (oliko kyseessä esim. kuva tai binääri)
Käyttäjän agentin tyyppien yleisyys harjoitus tiedossa	Kuinka usein käyttäjä esiintyi: 0-∞ käytä esityksenä 0, 1, 2, 4, 8, 16...
Vastauskoodi	Käytetään log-tiedostosta löytyvää jonoa muokkaamattomana.