jamk.fi

Asiakasliikenteen salaus SDN-verkossa

Tuure Valo

Opinnäytetyö
Huhtikuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), tietotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Valo, Tuure	Opinnäytetyö, AMK	22.4.2016
	Sivumäärä 85	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Asiakasliikenteen salaus SDN-verkossa

Tutkinto-ohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Jarmo Viinikanoja, Mika Rantonen

Toimeksiantaja(t)
Cyber Trust –projekti, Jyväskylän ammattikorkeakoulu, Janne Alatalo

Tiivistelmä

Opinnäytetyön tilaajana toimi Jyväskylän ammattikorkeakoulun Cyber Trust -projekti, joka on osallisena Digilen Cyber Trust -tutkimusohjelmaa. Tutkimusohjelman tarkoituksena on nostaa Suomi johtavaksi kyberturvallisuuden asiantuntijaksi ja edelläkävijäksi luomalla tietoturvasta digitaalisen luottamuksen perustan.

Toimeksiantona opinnäytetyölle oli tutkia SDN-verkon välitystasolla kulkevan asiakasliikenteen salausmahdollisuuksia hyödyntäen verkon kontrollerin liikenteenohjausominaisuuksia ja avoimen lähdekoodin IPsec-ohjelmistoja.

Toteutuksen perustana toimi neljän kytkimen muodostama verkkotopologia, jossa asiakaslaitteiden välistä liikennöintiä salattiin käyttämällä strongSwan-ohjelmistoa Docker-kontin sisällä. Liikenne ohjattiin ONOS-kontrollerilla salauskomponenteille, joista verkkoliikenne laitettiin salauskomponenttien välisiin IPsec-tunneleihin. Verkon skaalautuvuuden ja hallinnan parantamiseksi Ansiblella luotiin playbookkeja, joilla verkkoon kyettiin lisäämään uusia Open vSwitch-kytkimiä ja hallinnoimaan salauskomponenttien konfiguraatioita keskitetysti.

Työn tavoitteena oli salata asiakasliikennettä ja siinä onnistuttiin kiitettävällä tasolla. Myös salauskomponenttien konfiguraatiohallinta onnistuttiin täysin automatisoimaan. Uuden kytkimen implementointi verkkoon onnistui hyvin, sillä prosessi automatisoitiin niin pitkälle kuin mahdollista. Sen sijaan koko salausprosessin automatisointia ei onnistuttu toteuttamaan, sillä verkon kontrollerilta ei haettu tarvittavia kytkimien tilatietoja liikenteenohjauksen toteuttamiseen automatisoidusti.

Avainsanat (asiasanat)
SDN, IPsec, Docker, NFV, OpenFlow, OVS, ONOS, Ansible

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Valo, Tuure	Bachelor's thesis	22.4.2016
		Language of publication: finnish
	Number of pages 85	Permission for web publication: x

Title of publication
Data encryption in SDN-networks

Degree programme
Information Technology

Supervisor(s)
Jarmo Viinikanoja, Mika Rantonen

Assigned by
Cyber Trust-project, Jyväskylä University of Applied Sciences, Janne Alatalo

Abstract

The thesis was assigned by a project called Cyber Trust within JAMK University of Applied sciences as a part of Digile's Cyber Trust research program. The goal of the research program is to lift Finland to expert level in cyber security and to be a forerunner of digital trust and information security.

The Assignment was to search for possibilities to encrypt customer data on the infrastructure layer of SDN-networks by using traffic forwarding and open source IPsec-softwares.

The base of the network topology consisted of four Open vSwitches where the traffic between customers were encrypted by IPsec-software called strongSwan running inside Docker containers. Traffic between customers was forwarded to encryption components by using ONOS's forwarding capabilities, where the traffic was then encapsulated to IPsec tunnels. To improve network scalability and configuration management Ansible was used to managing configuration changes for each encryption component. Also another Ansible playbook was created to introduce new switches to the existing network.

As a result, it was possible to encrypt traffic in the network and the process of configuration management supported the implementation satisfyingly. Also the automation of introducing new switches to the network was done to a point where minimal amount of manual configuration was needed. The automation of the whole process of introduction a fully functional entirety of IPsec tunnel and traffic forwarding was not carried out due to not getting the needed information of the network from the network controller.

Keywords/tags (subjectshttp://vesa.lib.helsinki.fi/)
SDN, IPsec, Docker, NFV, OpenFlow, OVS, ONOS, Ansible

Miscellaneous

Sisältö

Lyhenteet 7
1 Lähtökohdat 9
1.1 Toimeksiantaja 9
1.2 Toimeksianto 9
2 Software Defined Networking 10
2.1 Muutoksen tarve 10
2.2 Nykyiset rajoitteet 10
2.3 Arkkitehtuuri 11
2.4 SDN-Kontrollerit 12
2.5 OpenFlow 13
2.5.1 Yleistä 13
2.5.2 OpenFlow-kytkin 14
2.5.3 Vuo ja vuotaulu 15
2.6 NFV 16
2.6.1 Yleistä 16
2.6.2 Hyödyt 17
2.6.3 Haasteet 18
3 Internet Protocol Security 19
3.1 Yleistä 19
3.2 Protokollat 19
3.2.1 Authentication Header-protokolla 19
3.2.2 Encapsulating Security Payload -protokolla 21
3.2.3 Internet Key Exchange-protokolla 23
4 Käytetyt komponentit 25
4.1 Open Network Operating System 25
4.2 Open vSwitch 27

4.3 Docker 29
4.4 strongSwan 31
4.5 Ansible 31
4.5.1 Yleistä 31
4.5.2 Playbookit ja roolit 32
5 Käytännön toteutus 33
5.1 Ympäristön ja topologia 33
5.2 ONOS asennus 36
5.3 OVS asennus 39
5.4 Docker asennus 40
5.5 Kytkinten ja Dockerin liittäminen ympäristöön 40
6 Salauksen implementointi verkkoon 45
6.1 Salauskomponentin luonti 45
6.2 Salauskontin liittäminen kytkimiin 47
6.3 Salauksen konfigurointi asiakkaiden välille 51
7 Verkon automatisointi 62
7.1 Konfiguraatiohallinta 62
7.2 Uuden kytkimen luonti 70
8 Johtopäätökset 73
8.1 Työn lopputulos 73
8.2 Jatkokehitys 74
8.3 Pohdinta 75
Lähteet 76
Liitteet 78
Liite 1. Dockerfile salaajakonttia varten 78
Liite 2. roles/ipsec/vars/main.yml-tiedosto 79
Liite 3. Konfiguraatiohallinnan suorittaminen 80
Liite 4. Cipher-OVS1-kontin ipsec.conf-tiedosto 81

Liite 5. Cipher-OVS1-kontin ipsec.secrets-tiedosto.....82

Liite 6. /etc/ansible/roles/ovs/tasks/main.yml-tiedosto.....82

Liite 7. /etc/ansible/roles/docker/tasks/main.yml.....83

Liite 8. Uuden kytkimen asennus Ansiblella.....84

Kuviot

Kuvio 1. Nykyisen verkkotekniikan yksitasoinen rakenne 11
Kuvio 2. SDN-verkon keskitetty hallinta käyttämällä kontrolleria 12
Kuvio 3. SDN-verkon kolmitasoinen arkkitehtuuri 13
Kuvio 4. OpenFlow-kytkimen rakenne 15
Kuvio 5. Vuon rakenne 15
Kuvio 6. Perinteisen verkkolaitteiston ero verrattuna NFV-toteutukseen 17
Kuvio 7. Autentikointi-otsikon paikka IP-paketissa kuljetustilassa 20
Kuvio 8. Autentikointi-otsikko 21
Kuvio 9. ESP-otsikon sijoitus kuljetustilassa 22
Kuvio 10. ESP-otsikon sijoitus tunnelitilassa 22
Kuvio 11. ESP-otsikon rakenne 23
Kuvio 12. IPsec-skenaariot 24
Kuvio 13. Suojaussidosten muodostus osapuolten välillä 25
Kuvio 14. ONOSin graafinen käyttöliittymä 27
Kuvio 15. OVS-kytkimen porttien tilatietojen hakeminen 28
Kuvio 16. OVS-kytkimen vuotaulujen tulostus 29
Kuvio 17. Ero virtualisoinnin ja Dockerin välillä 29
Kuvio 18. Docker-kontin luominen 31
Kuvio 19. Toteutuksen verkkotopologia 34
Kuvio 20. Virtuaalikoneiden hallintayhteys NAT Network –asetuksen kautta 35
Kuvio 21. Virtuaalikoneiden verkotus käyttäen Internal Network –asetusta 36
Kuvio 22. ONOSin käyttöliittymä kirjautumisen jälkeen 38
Kuvio 23. OVS-kytkimen luomisen todennus 42
Kuvio 24. OVS1-virtuaalikoneen rajapintojen osoitteet 43
Kuvio 25. Kytkimen kontrolleriyhteys ja geneve-rajapinnat 44
Kuvio 26 ONOSin webbikäyttöliittymä 45
Kuvio 27. Uuden kansion hakemiston Dockerfile-tiedostossa 46
Kuvio 28. Asennuskansion määritteleminen 46
Kuvio 29. Tiedostojen lisäys uuteen hakemistoon 46
Kuvio 30. Consul-koneen levykuvat 47

Kuvio 31. Docker-konttien rajapinnat OVS1-kytkimellä 49
Kuvio 32. Cipher-OVS1-kontin verkkoasetukset 50
Kuvio 33. Host1-kontin verkkoasetukset 50
Kuvio 34. Salauskonttiin kiinnitetyn hakemiston sisältö 52
Kuvio 35. strongSwanin konfiguraatiotiedosto 52
Kuvio 36. Cipher-OVS1-kontin esijaetut avaimet 52
Kuvio 37. Cipher-OVS2 -kontin esijaetut avaimet 52
Kuvio 38. Cipher-OVS1 -kontin IPsec-konfiguraatiot 53
Kuvio 39. Cipher-OVS2 -kontin IPsec-konfiguraatiot 54
Kuvio 40. Graafinen esitys konfiguroidusta IPsec-tunnelista 54
Kuvio 41. Suojaussidosten neuvottelu konttien välillä 55
Kuvio 42. Muodostunut suojaussidos 56
Kuvio 43. ICMP-ping host1 ja host2 välillä 56
Kuvio 44. Havainnollistettu esitys liikenteen kulusta 57
Kuvio 45. OVS1-kytkimen porttinumerointi ja dpid-arvo 58
Kuvio 46. Docker inspect host1 58
Kuvio 47. Docker inspect cipher-OVS1 59
Kuvio 48. Intenttien luomat vuot 60
Kuvio 49. IPsec-tunneliin menevä ping 60
Kuvio 50. Muutokset suojaussidoksen tulosteessa 61
Kuvio 51. Salatun liikenteen kaappaus 61
Kuvio 52. Graafinen esitys salatusta liikenteestä 62
Kuvio 53. Ansiblen hosts-tiedosto 64
Kuvio 54. Ansiblen hakemistorakenne 64
Kuvio 55. Konfiguraatiohallinnan playbook 65
Kuvio 56. roles/ipsec/tasks/main.yml-tiedosto 66
Kuvio 57. Consulilla olevat konfiguraatiot 66
Kuvio 58. roles/ipsc/templates/salaaja.j2-tiedosto 67
Kuvio 59. osa roles/ipsec/vars/main.yml-tiedostoa 68
Kuvio 60. roles/ipsec/templates/secrets.j2-tiedosto 68
Kuvio 61. roles/ipsec/templates/strongswan.j2-tiedosto 69
Kuvio 62. Automatisoinnilla luodut suojaussidokset 70
Kuvio 63. Uusi hosts-tiedosto 71

Kuvio 64. new_machines-ryhmän muuttujat....71
Kuvio 65. Playbook uuden kytkimen luomiseen 71
Kuvio 66. Docker-roolin shell-moduuli....73
Kuvio 67. Dockerin ja OVS:n asennus....73

Taulukot

Taulukko 1. Ympäristön laitteisto 34
Taulukko 2. Konttien MAC- ja IP-osoitteet....49

Lyhenteet

AAA	Authentication, Authorization and Accounting
AH	Authentication Header
API	Application Programmable Interface
ESP	Encapsulating Security Payload
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IKE	The Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
NAT	Network Address Translation
NE	Network Element
NFV	Network Function Virtualisation
ONOS	Open Network Operating System
OVS	Open vSwitch
QoS	Quality of Service
RGCE	Realistic Global Cyber Environment
SaaS	Software-as-a-Service
SSH	Secure Shell
SDN	Software Defined Networking

TTL	Time-To-Live
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
YAML	Yet Another Markdown Language

1 Lähtökohdat

1.1 Toimeksiantaja

Jyväskylän ammattikorkeakoulu on yksi monista tutkimusorganisaatioista, jotka osallistuvat DIGILEn Cyber Trust -tutkimusohjelmaan. Cyber Trust -tutkimusohjelman tärkeimmät tavoitteet ovat luottamuksen ja yksityisyyden palauttamisessa digitaalisessa maailmassa havainnoimalla uhkia ja heikkouksia uusissa tekniikoissa, kehittämällä työkaluja riskien hallitsemiseen ja muokkaamalla kansan mielikuvaa kyberturvallisuuden integroimisesta jokapäiväiseen elämään. Cyber Trust -ohjelma on jaettu neljään työryhmitykseen, joihin yritykset ja tutkimusorganisaatiot on jaettu. Jyväskylän ammattikorkeakoulu on osa työryhmitystä, jonka tarkoituksena on tutkia uusia verkkotekniikoita ja niiden uhkia. (Savola 2014)

Opinnäytetyön toimeksiantajana toimi Cyber Trust -projekti Jyväskylän ammattikorkeakoulussa. Syksyn 2015 aikana projekti sai aikaan virtualisoidun testiympäristön, jolla voidaan simuloida SDN-pohjaista verkko-operaattoria Jyvsectecin RGCE-ympäristössä.

1.2 Toimeksianto

Opinnäytetyön tarkoituksena oli tutkia erilaisia salausvaihtoehtoja SDN-pohjaisen verkko-operaattorin asiakkaille. Työn takana oli ajatus verkko-operaattorin palvelukaupasta, josta verkon asiakas voisi ostaa haluamiaan palveluita ilman, että se tuottaisi asiakkaalle laitehankintoja tai konfigurointia. Yksi verkko-operaattorin tarjoamista palveluista oli toimipisteiden välinen salaus ohjaamalla asiakkaan liikenne sitä salaavalle NFV-elementille. Työn painopisteenä oli tutkia salauskomponentin sijoittamista verkon eri kohtiin ja tutkia mahdollisia ohjelmistoja liikenteen salaamiseen.

2 Software Defined Networking

2.1 Muutoksen tarve

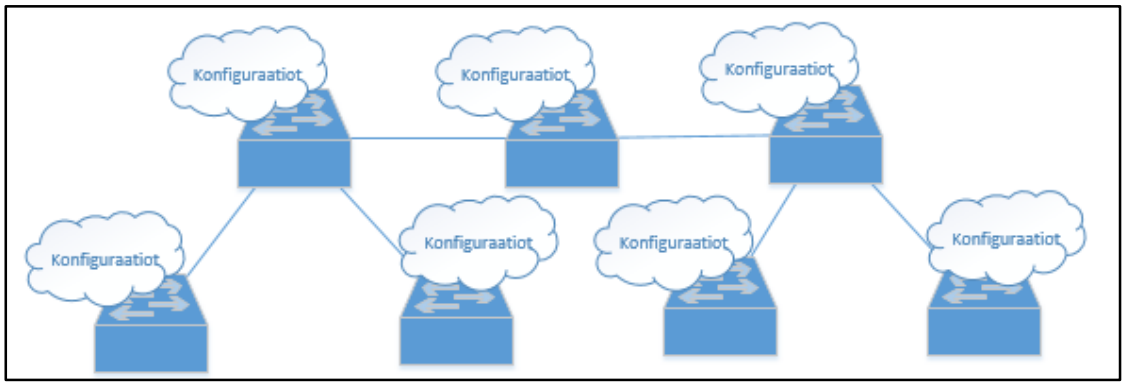
Nykyiset tietoverkot ovat murrosvaiheessa. Verkkojen kuormitus kasvaa jatkuvasti ja nykyisen infrastruktuurin kapasiteettivaatimukset kasvavat uhkaavasti. Suuret laitekompleksit ja lukuisat protokollat ovat ajaneet tietoverkot tilanteeseen, jossa niiden ylläpitäminen on jatkuvasti vaikeampaa ja samalla verkkoon tehtävät muutokset muuttuvat vaikeammin toteutettavaksi. Tämä vaikuttaa myös tietoverkkojen tutkimusprojekteihin, sillä kynnys innovaatioille ja uusille ideoille kasvaa jatkuvasti nykyisen verkkotekniikan laajetessa. Seurauksena tutkijat ovat nimittäneet tämän ilmiön verkon ”kangistumiseksi”. (Anderson, Balakrishnan, McKeown, Parulkar, Peterson, Rexford, Shenker & Turner 2008, 1.)

2.2 Nykyiset rajoitteet

Nykyinen verkkoliikenteen lähes räjähdysmäinen kasvu on kuitenkin pakottanut tutkimusorganisaatiot etsimään ratkaisua ongelmaan. Nykyisiä kasvua rajoittavia ongelmia ovat mm. hyvin staattisiksi määritellyt verkkoratkaisut kuten verkkolaitteilla olevat pääsylistat, VLAN- ja QoS-asetukset, joiden pitää reagoida verkossa tapahtuviin muutoksiin. Usein se tarkoittaa konfiguraatioiden muutoksia laitekohtaisesti. Toinen merkittävä rajoittava tekijä on verkkojen huono skaalautuvuus, sillä verkkolaitteiden määrä lisääntyy jatkuvasti, mikä johtaa monimutkaisempiin verkkoratkaisuihin. Operaattorit ovatkin jo pitkän aikaa nojautuneet linkkivälien yliallokointiin perustuen ennustettaviin liikennemääriin, mutta allokointi muuttuu jatkuvasti vaikeammaksi, koska verkkoon liittyvien laitteiden kuten erilaisten mobiililaitteiden muodostama liikenne tekee liikenteen ennustamisesta haastavampaa. Yritykset ja operaattorit etsivätkin ratkaisuja kasvattaakseen verkkojen ja palveluiden kapasiteettia vastatakseen asiakkaiden tarpeisiin, mutta laitteiston toimittajien haluttomuus yhteistyöhön muiden valmistajien kanssa vaikeuttaa verkottamismahdollisuuksia, kun standardoitua avointa rajapintaa ei saada tehtyä laitteiden välille. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

2.3 Arkkitehtuuri

Nykyhetken verkkoratkaisuissa verkon kontrolli- sekä asiakasliikenne kulkevat samassa tasossa ja jokaista verkkoelementtiä hallitaan itsenäisenä osana verkkoa. Tätä on havainnollistettu kuviossa 1. Verkkoelementit kommunikoivat keskenään, levittäen hallintainformaatiota ja siten verkon on mahdollista reagoida siinä tapahtuviin muutoksiin. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

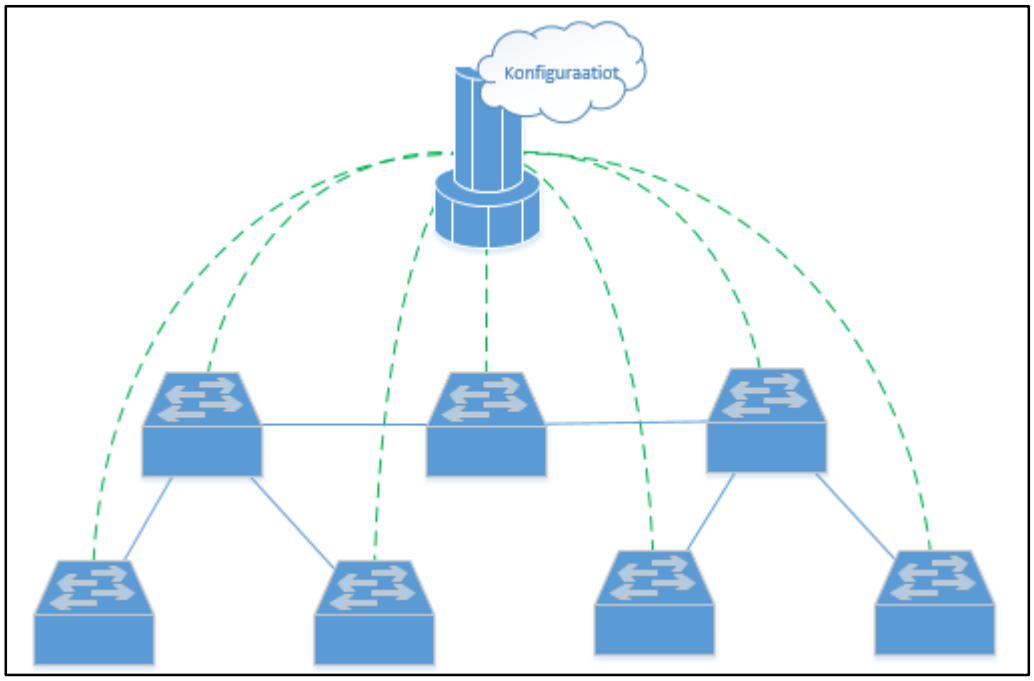


Kuvio 1. Nykyisen verkkotekniikan yksitasoinen rakenne

SDN-verkko eroaa tästä erottamalla asiakasliikenteen ja kontrolliliikenteen eri tasoille, jolloin ylemmällä tasolla kulkevaa liikennettä voidaan hallinnoida keskitetyltä ohjelmistolta. Tätä verkon osaa kutsutaan SDN:ssä verkon kontrolleriksi. Siten verkkoelementtien ei tarvitse vaihtaa tilatietoja toistensa kanssa, vaan kontrolliliikenne liikkuu ainoastaan laitteen ja kontrollerin välillä mahdollistaen fyysisen infrastruktuurin erottamisen loogisiksi tai virtuaalisiksi kokonaisuuksiksi. (Software-Defined Networking: The New Norm for Networks 2012, 7.)

Monikerrosarkkitehtuuri mahdollistaa yritysten ja operaattorien valita haluamansa laitekannan riippumatta välityskerroksella olevien laitteiden mahdollisesta yhteensopimattomuudesta. Kun verkkoa voidaan hallita yhden pisteen kautta, se yksinkertaistaa myös verkon suunnittelua ja sen operointia. Myöskin itse laitteet muuttuvat yksinkertaisemmiksi, sillä niiden ei tarvitse ymmärtää tuhansia eri protokollia, sillä laite voi kysyä ohjeita pakettien käsittelyyn kontrollerilta. Kontrollerin ja verkkoelement-

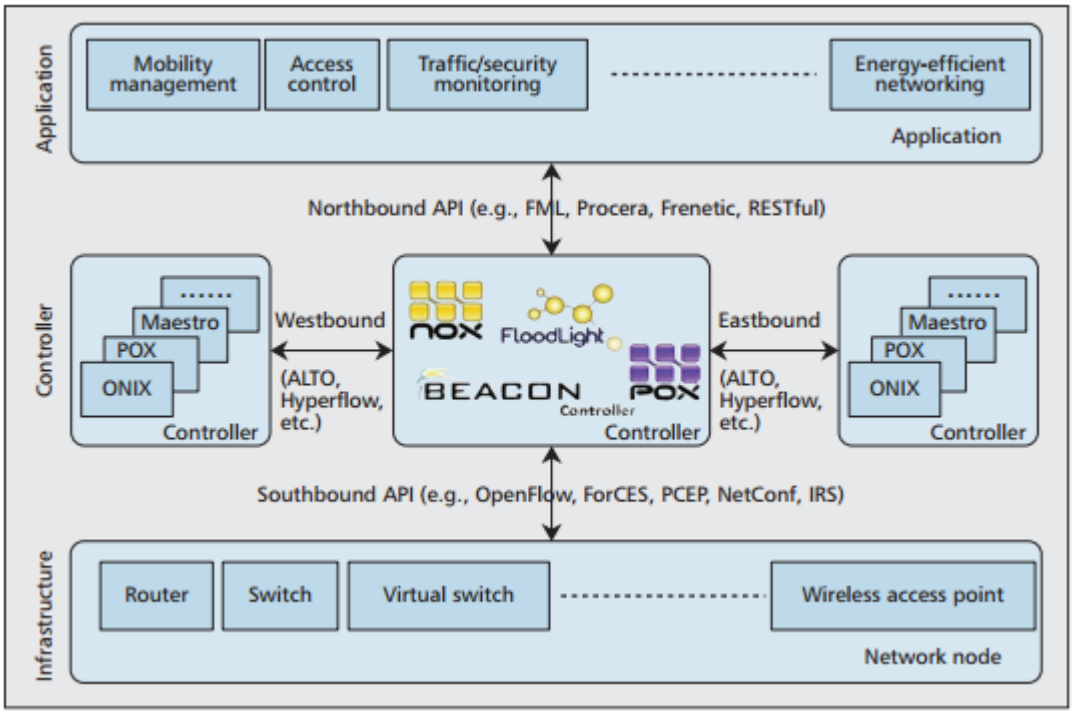
tien väliseen kommunikaatioon käytetään OpenFlow-protokollaa (kts. luku 2.4.2). Kuviossa 2 on havainnollistettu, kuinka vihreällä kuvatut kontrolliyhteydet on kokonaan eristetty linkeistä, jossa tuotantoliikennettä kuljetetaan. (Software-Defined Networking: The New Norm for Networks 2012, 7.)



Kuvio 2. SDN-verkon keskitetty hallinta käyttämällä kontrolleria

2.4 SDN-Kontrollerit

Monikerroksisessa SDN-verkossa verkon kontrollerilla on merkittävä rooli. Se toimii informaation välittäjänä verkon välitystasolla olevien laitteiden ja sovellustasolla olevien applikaatioiden välillä. Kontrollerin ja välitystasolla olevien laitteiden välistä rajapintaa kutsutaan southbound-rajapinnaksi, ja northbound-rajapinta on kontrollerin ja sen applikaatioiden välissä. Tyypillisimpiä protokollia southbound-rajapinnalla ovat OpenFlow ja OVSDB. Northbound-rajapinta hyödyntää useimmiten RESTful-, FML-, Procera- tai Frenetic-ohjelmointirajapintaa. Kuviossa 3 on havainnollistettu kolmitasoisen arkkitehtuurin rakennetta. (What are SDN Controllers (or SDN Controllers Platforms)? N.d)



Kuvio 3. SDN-verkon kolmitasoinen arkkitehtuuri (Chouhan, Finnegan, Fraser, Lake, Miller, Rao, Scott-Hayward, Sezer & Viljoen 2013, 36.)

SDN-kontrollerit sisältävät sovellustasolla applikaatioita, joilla voidaan suorittaa erilaisia tehtäviä verkossa. Joidenkin applikaatioiden tehtävä voi olla hyvinkin selvä: esimerkiksi tukea verkon välitystasolle pakettien L2-edelleenlähetystä. Kontrollerilla on myös jatkuvasti käsitys koko verkon tilasta, jolloin sillä on mahdollisuus kerätä statistiikkaa mm. laitteista ja verkon kapasiteetista. Laajennettavissa olevilla applikaatioilla voidaan parantaa verkon toiminnallisuutta ja optimoida verkon kapasiteettia analysoimalla suorituskykyä ja orkestroimalla uusia sääntöjä verkkoon. (What are SDN Controllers (or SDN Controllers Platforms)? N.d)

2.5 OpenFlow

2.5.1 Yleistä

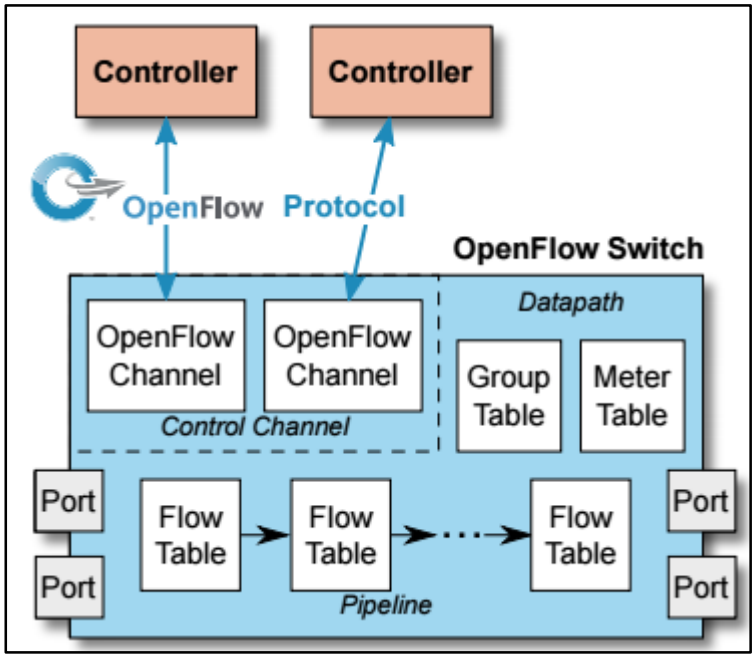
OpenFlow on ensimmäinen avoin standardi rajapinnalle, joka mahdollistaa kommunikaation SDN-verkon välitystasolla olevien verkkoelementtien ja kontrollitasolla ole-

vien kontrollerien välillä. Näin verkon kontrolleri voi suoraan manipuloida välitystasolla olevia laitteita ja sitä kautta vaikuttaa liikenteen välitykseen. OpenFlow-protokollan perustana ovat vuot ja vuotaulut (flow, flowtable), jotka mahdollistavat verkon erittäin hienojakoisen jaottelun vastaamalla reaaliajassa sovellus-, käyttäjä- ja sessiotason muutoksiin. Näin hienojakoista hallintaa nykyinen IP-perustainen reititys ei tarjoa, sillä liikenne kahden päätepisteen välissä menee aina samaa polkua pitkin ottamatta huomioon erityyppisten liikenteiden vaatimuksia. (Software-Defined Networking: The New Norm for Networks 2012, 8)

Alun perin OpenFlow-kytkentä sovitettiin Ethernet-perustaisiin verkkoihin, mutta nykyään OpenFlow-protokollaa voidaan hyödyntää yhä useammissa käyttötapauksissa, sillä OpenFlow-yhteensopivuutta voidaan tuoda jo olemassa oleviin virtuaalisiin ja fyysisiin verkkoihin. Laitevalmistajien OpenFlow-tuella varustetut verkkolaitteet toimivat myös perinteisien verkkolaitteiden tapaan, mikä helpottaa yritysten ja operaattorien SDN-pohjaisten ratkaisujen tuomista vaiheittain olemassa olevan infrastruktuurin päälle mahdollistaen useiden laitevalmistajien laitteiden käytön samassa verkossa. Valmistajat ovat myös päivittäneet vanhempiin laitteisiinsa OpenFlow-tuen yksinkertaisesti tuomalla sen laitteen sulautetun ohjelmistopäivityksen yhteydessä. (Software-Defined Networking: The New Norm for Networks 2012, 10.)

2.5.2 OpenFlow-kytkin

OpenFlow-kytkimen rakenne on varsin yksinkertainen, sillä sen voi jakaa kahteen loogiseen osioon. Kytkimellä on salattuja kanavia kontrollerien kanssa, jota pitkin kontrollerit hallitsevat kytkintä OpenFlow-protokollalla. Toinen osio on vuotaulut, jonka perusteella paketteja välitetään välitystasolla. Pakettien käydään läpi tarkemmin luvussa 2.5.3. Paketin käsittely on riippuvainen kontrollerilla olevien applikaatioiden tekemistä päätöksistä. Kuviossa 4 on vielä visualisoitu vuotaulun, salatun kanavan ja kontrollerin suhde. (Appenzeller, Balland, Barker, Beckmann, Casado, Cohn, Crabbe, Curtis, Das, dHeureuse, Ding, Dunbar, Erickson, Gandham, Gibb, Heller, Kis, Kobayashi, Lantz, Madabushi, Malek, McDysan, McKeown, Mizrahi, Moses, Nygren, Orr, Pettit, Pfaff, Poutievski, Ramanathan, Price, Sherwood, Schneider, Takahashi, Talayco, Tonsing, Tourrilhes, Vicisano, Ward, Yabe, Yadav, Yap & Yiakoumis 2014, 11.)



Kuvio 4. OpenFlow-kytkimen rakenne (Appenzeller ym. 2014, 11.)

2.5.3 Vuo ja vuotaulu

OpenFlow-kytkin toimii hieman kuten prosessilinjasto, sillä jokaisella kytkimellä on sisään- ja ulostulo-portin välissä tulee olla vähintään yksi vuotaulu, kuten kuviossa 4 on ilmaistu. Vuon rakenne on esitetty kuviossa 5.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Kuvio 5. Vuon rakenne

Vuossa on seitsemän kenttää:

- Match field: Kenttä, johon paketin otsikkotietoja sekä sisääntuloporttia verrataan.
- Priority: Vuot käydään läpi prioriteettikentän mukaisessa järjestyksessä aloittaen suurimmasta prioriteetista
- Counter: Vuolla on laskuri, joka ilmoittaa siihen sovitettujen pakettien määrän.
- Instuctions: Kenttä, jolla määritellään paketille tehtävät toiminnot.

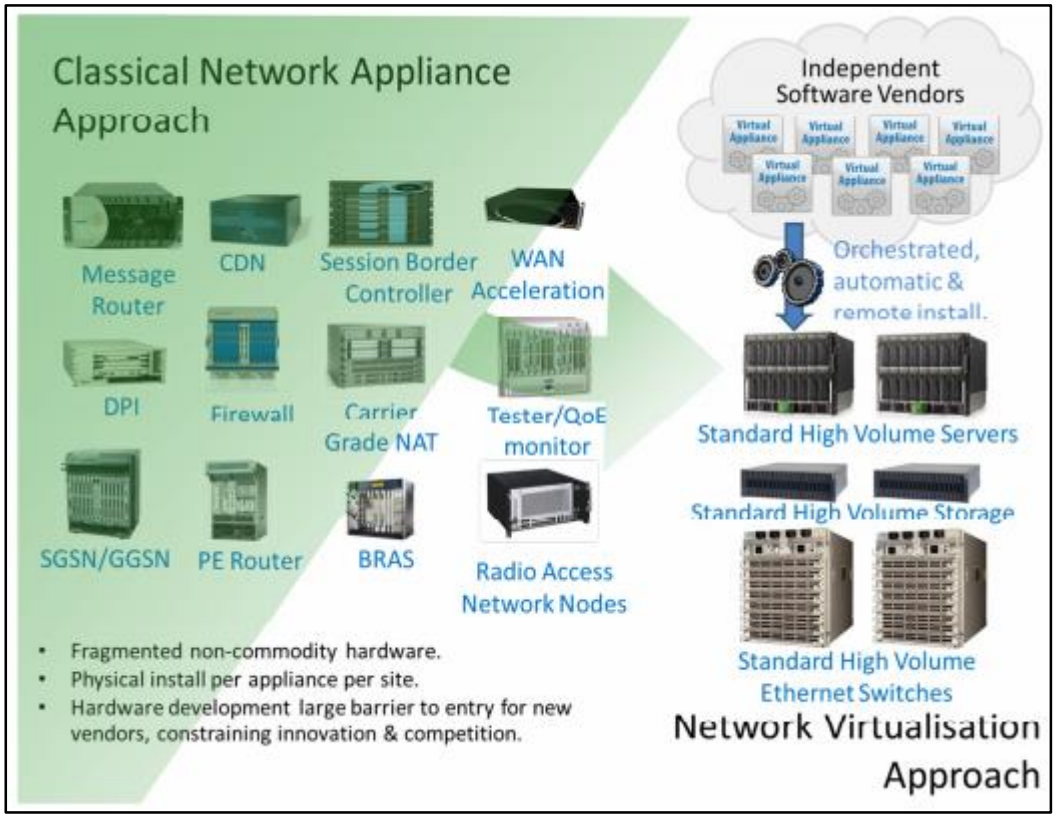
- Timeout: Suurin mahdollinen aika vuolle olla passiivisessa tilassa, ajan tullessa täyteen kytkin poistaa vuon.
- Cookie: Kontrollerin antama läpinäkymätön data-arvo, jota kontrolleri käyttää yksilöllisten voiden tunnistamiseen.
- Flags: Lipuilla voidaan muuttaa tapaa, joilla voita hallitaan.

Kuvion 4 mukaisesti paketin tullessa kytkimelle kytkin poimii paketin sisääntuloportin ja otsikkokenttien tiedot. Tätä dataa verrataan tauluissa oleviin voihin prioriteettikentän mukaisessa järjestyksessä aloittaen suurimmasta prioriteetista. Jos paketti sopii vuohon, sille tehdään instructions-kentän mukaiset operaatiot. Jos paketti ei osu taulun mihinkään vuohon, kytkin siirtyy vertaamaan pakettia seuraavaan tauluun. Jos paketti ei viimeisenkään taulun jälkeen ole sopinut yhteenkään vuohon tilanteesta riippuen paketti pudotetaan tai laitetaan kontrollerin käsiteltäväksi. (Appenzeller ym. 2014, 22-23.)

2.6 NFV

2.6.1 Yleistä

Operaattorien verkot muuttuvat jatkuvasti suuremmiksi ja monimutkaisemmiksi varsinkin laitteiston kannalta. Uuden verkkopalvelun käynnistäminen vaatiikin usein lisää fyysistä tilaa nykyisestä infrastruktuurista. Kun yhtälöön lisätään vielä tarve uuden laitteiston hallintaan vaativalle ammattitaidolle, kuluu huomattavasti rahaa palvelun käyttöönottoon. Lisäksi laitteistoon perustuvilla palveluilla on harmillisen lyhyt elinkaari. Teknologian kehityksen kiihtyessä jatkuvasti, palvelusta saatava tuotto voi jäädä hyvinkin pieneksi. NFV eli verkkotoimintojen virtualisointi tähtää verkkoarkkitehtuurien yksinkertaistamiseen muuttamalla laitteistolla tuotettuja palveluita kuten palomuurit ja muut yksilölliset verkkolaitteet standardoiduiksi korkean suorituskyvyn palvelimilla suoritettaviksi ohjelmiksi. Kuviossa 6 tuodaan esille, kuinka rautatason erilaiset laitteet voidaan korvata fyysisillä palvelimilla. (Benitez, Bugenhagen, Chiosi, Clarke, Cui, Damker, Delisle, Demaria, Deng, Fargano, Feger, Fukui, Guardini, Khan, Kolias, Lopez, Loudier, Manzalini, Matsuzaki, Michel, Minerva, Ogaki, Prodip, Reid, Ruhl, Salguero, Shimano & Willis 2012, 3.)



Kuvio 6. Perinteisen verkkolaitteiston ero verrattuna NFV-toteutukseen (Benitez ym. 2012, 5.)

Tämä mahdollistaa virtuaalisen laitteiston siirtämisen verkon sisällä ilman uuden fyysisen laitteiston asentamista. NFV on mahdollista toteuttaa ilman SDN-tekniikkaa, mutta SDN:n suurimmat edut saadaan esille virtualisoimalla SDN-verkon tarvitsema infrastruktuuri palvelinympäristöissä. NFV-tekniikalle jo kartoitettuja käyttötapoja ovat mm. reitittimet, liikenneanalysaattorit, salaustunnelointi, AAA-palvelimet, kuormantasaajat ja tietoturvakomponentit kuten palomuurit, virusskannerit ja IDS. Vielä ei ole kuitenkaan tutkittu tarpeeksi, mistä käyttötapauksista on suurin hyöty perinteisiin verkkolaitteisiin verrattuna. (Benitez ym. 2012, 3-7.)

2.6.2 Hyödyt

Virtualisoitujen laitteiden etuja on useita. Fyysiset verkkolaitteet tulee tuotantoverkossa ylimitoittaa, jotta ruuhkatilanteessa verkko ei ylikuormitu. Tämä johtaa siihen, että laitteisto kuluttaa enemmän sähköä, kuin sille on tarvetta ns. normaalitilan-

teessa. Sen sijaan virtualisoitujen laitteiden resursseja voidaan säätää lennosta vastaamaan verkon tarpeita. Uusia tuotteita voidaan tuoda markkinoille nopeammin, sillä rautatason investoinnit eivät enää ole osa tuotekehitystä. Tällöin laitteistoa on myös helpompi päivittää. NFV mahdollistaa myös tuotanto- ja testiympäristön pyörittämisen samassa infrastruktuurissa, mikä vähentää verkon kehitys- ja testauskustannuksia. Palveluita voidaan myös kohdistaa perustuen maantieteellisiin sijaintiin tai asiakaskuntiin, sillä palveluita voidaan provisioida keskitetysti ohjelmistolla, ilman asentajan menemistä paikalle asentamaan uutta laitteistoa. NFV:llä voidaan eristää ja luoda räätälöityjä palveluita asiakkaille perustuen heidän tarpeisiinsa saman fyysisen tuotantoympäristön sisällä. Myös verkonhallinnan tehokkuus kasvaa, sillä infrastruktuurin muuttuessa yhdenmukaiseksi verkko muuttuu skaalautuvammaksi ja sitä on helpompi orkestroida. (Benitez ym. 2012, 8-9.)

2.6.3 Haasteet

Verkkolaitteiden virtualisoinnissa on kuitenkin omat haasteensa. Ohjelmistojen tulee tukea useiden palvelinvalmistajien laitteistoja, jotta virtuaalisia verkkolaitteita voidaan käyttää verkon eri osissa. Varmaa on myös, että virtuaalikoneiden suorituskyky ei ole yhtä hyvä kuin fyysisen laitteella, mutta tekniikan kehittyessä haasteena on suorituskyvyn heikkenemisen minimoiminen virtualisointikerroksella. Operaattorit haluavat myös implementoida NFV-komponentteja verkkoon vaiheittain, joten perinteisten ja virtuaalikoneiden tulee toimia yhdessä ilman, että se vaikuttaa operaattorin palveluiden laatuun. Virtuaalikoneille on myös elintärkeää, että skaalautuvuus ja muutoksiin reagointi on automatisoitu, jotta luvussa 2.6.2 mainittuihin energiansäästöhyötyihin päästään. Turvallisuus, kestävyys ja vakaus ovat virtualisoidun verkon kulmakiviä. Operaattorien tulee taata verkon NFV-komponenttien toimivuus mahdollisissa häiriötilanteissa. Viimeisenä haasteena on verkkoympäristön yksinkertaistaminen kaikilla osa-alueilla. Uusien laitteiden implementointi ja verkonhallinta tulee olla yksinkertaisempaa verrattuna nykyiseen verkkoarkkitehtuuriin. On tärkeää välttää perinteisen verkkoarkkitehtuurin haasteiden vaihtamista uuden arkkitehtuurin tuomiin haasteisiin ilman, että haasteiden vakavuus tai lukumäärä ei vähene. (Benitez ym. 2012, 10-12.)

3 Internet Protocol Security

3.1 Yleistä

Internetin alkuaikoina sitä käytettiin tutkimusorganisaatioita ja akatemioita yhdistävänä toimialueena. Sen tarkoituksena oli mahdollistaa kommunikaatio ja yhteistyö organisaatioiden välillä käyttämällä etäyhteyttä. Kuitenkin jo 1980-luvun lopussa oli selvää, että erinäiset tahot olivat kiinnostuneita avoimen yhteyden väärinkäytöstä, joten liikennettä toimipisteiden välillä piti salata erilaisin keinoin. Oli kuitenkin selvää, että paikalliset ratkaisut kuten salauksen implementointi käytettävään sovellukseen olisi raskasta toteuttaa, joten haluttiin kokonaisvaltaisempi toteutus, joka salaisi kaiken liikenteen toimipisteiden välillä. Vuonna 1992 IETF aloitti IPsec-salauksen implementoinnin IP-protokollaan. (An Introduction to IPsec (INTERNET PROTOCOL SECURITY) 2001, 1.)

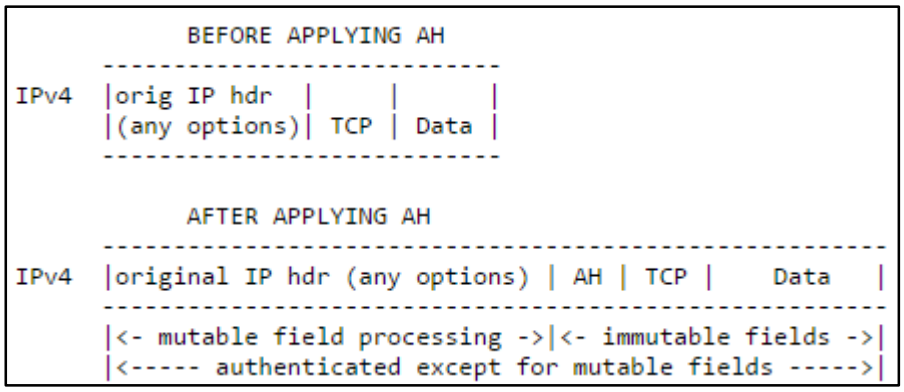
IPsec tarjoaa liikenteelle eheyden, joka varmistaa, että liikenteen vastaanottaja saa datan ilman peukalointia. Eheys toteutetaan yhteydettömästi, sillä IPsec ei vahvista lähettäjälle tietoa, onko vastaanottaja saanut paketin. Datan alkuperän autentikointi varmistetaan autentikointiotsikolla, jotta vastaanottaja voi olla varma, että lähettäjä on oikea, eikä tuntematon käyttäjä naamioituneena lähettäjäksi. Lähettäjän ja vastaanottajan välinen liikenne salataan salausalgoritmein, jotta välissä olevat verkon käyttäjät eivät näe dataa luettavassa muodossa. Data suojataan myös siten, että ulkopuoliset käyttäjät eivät voi määritellä myöskään liikenteen laatua tai määrää. (An Introduction to IPsec (INTERNET PROTOCOL SECURITY) 2001, 2-3.)

3.2 Protokollat

3.2.1 Authentication Header-protokolla

Autentikointiotsikko eli AH on IPsec:n lisäämä kenttä IP-otsikon L3- ja L4-otsikoiden väliin (kts. kuvio 7). AH mahdollistaa datan alkuperän autentikoinnin ja suojaa dataa toistolta, jos sille on tarvetta. Ylempien kerrosten data on autentikoitua, mutta IP-otsikon kentät voivat muuttua ja niitä AH-otsikko ei voi suojata. AH ei kuitenkaan sa-

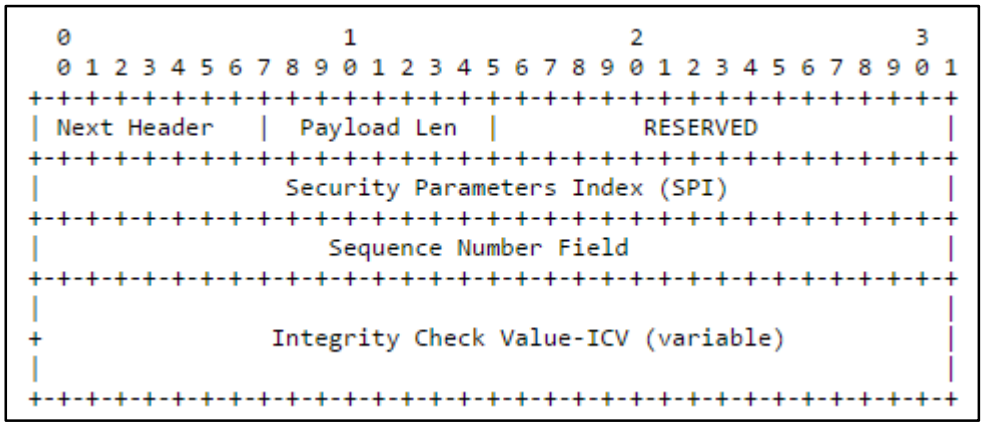
laa liikennettä, sillä se toteutetaan ESP-protokollalla. Salaus on eriytetty omaksi protokollaksi joidenkin maiden lainsäädännöllisten syiden vuoksi koskien Internet-liikenteen salausta. (An Introduction to IPsec (INTERNET PROTOCOL SECURITY) 2001, 2-3.)



Kuvio 7. Autentikointi-otsikon paikka IP-paketissa kuljetustilassa (Kent 2005a, 10.)

IP-otsikon protokollakenttä arvolla 51 määrittelee autentikointiotsikon käytön. AH-otsikon koko on 96 bittiä, joka on jaettu neljään kenttään, joista jokainen kenttä on pakollinen. Ensimmäinen kenttä otsikossa on Next Header (8 bittiä), joka määrittelee seuraavan otsikon AH:n jälkeen perustuen IANA:n antamaan protokollanumerointiin. Payload Length -kenttä määrittelee otsikon pituuden 32-bittisinä ”sanoina”. Reserved-kenttä on 16-bittinen kenttä, jolla ei ole tällä hetkellä käyttötarkoitusta. Kentän arvo tulee olla aina nolla, mutta se kuitenkin lasketaan osaksi otsikkoa, vaikka vastaanottaja jättääkin sen noteeraamatta. SPI on mielivaltainen 32-bittinen arvo, joka on sidoksissa kohteen IP-osoitteeseen ja AH-otsikkoon. Sen tarkoituksena on tunnistaa suojaussidos (security association) kyseiselle L4-datagrammille. Sequence Number on ylöspäin kasvava laskuri, jota lähettäjän tulee aina lisätä lähettäessä paketteja. Suojaussidoksesta riippuen vastaanottajan ei kuitenkaan välttämättä tarvitse käsitellä kyseistä kenttää. Mikäli suojaussidoksessa on määritelty käytettäväksi paketin kierrätyksen kielto, tulee lähettäjän ja vastaanottajan Sequence Number -laskurit olla aina samat. Jos luvut eivät täsmää, osapuolten tulee neuvotella uusi suojaussidos ja nollata laskurit. Viimeisenä kenttänä on IVC eli Integrity Check Value, joka lasketaan IP-otsikosta, AH-otsikosta ja AH-otsikon jälkeen olevasta muuttumattomasta datasta.

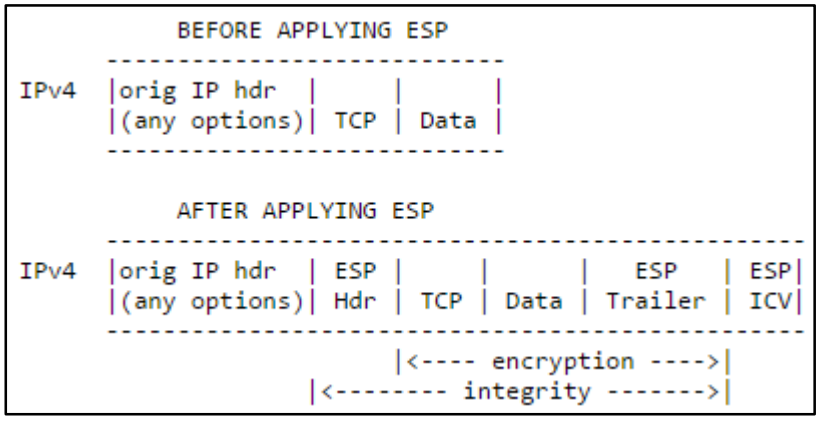
Kentän tarkoituksena on taata datan eheys. Kuviossa 8 on graafinen esitys autentikointiotsikosta. (Kent 2005a, 5-9.)



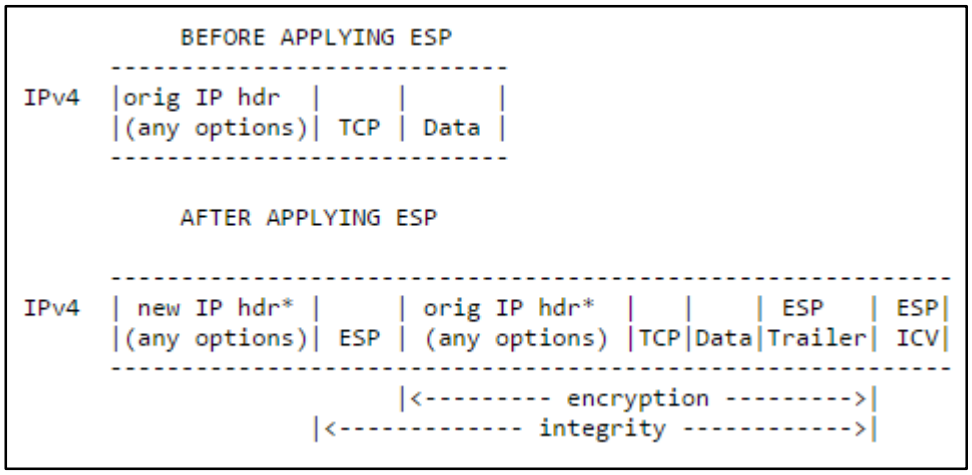
Kuvio 8. Autentikointi-otsikko (Kent 2005a, 4.)

3.2.2 Encapsulating Security Payload -protokolla

ESP-protokolla tarjoaa myös datalle eheyttä, luottamuksellisuutta ja autentikointia, sekä pakettien kierrätyksen eston, aivan kuten AH-protokolla. Sen lisäksi ESP salaa liikenteen hyödyntäen useita salausalgoritmeja. Kuljetustilassa ESP-otsikko on sijoitettu IP-otsikon jälkeen, mutta tunnelitilassa alkuperäinen IP-otsikko ja ylemmän kerroksen data kapseloidaan, jolloin paketille kirjoitetaan uusi IP-otsikko. Usein uusi IP-otsikko sisältää IPsec-tunnelin yhdyskäytävien tiedot. Kuvioissa 9 ja 10 on vertailtu otsikointia kummassakin tapauksessa. (Kent 2005b, 2.)

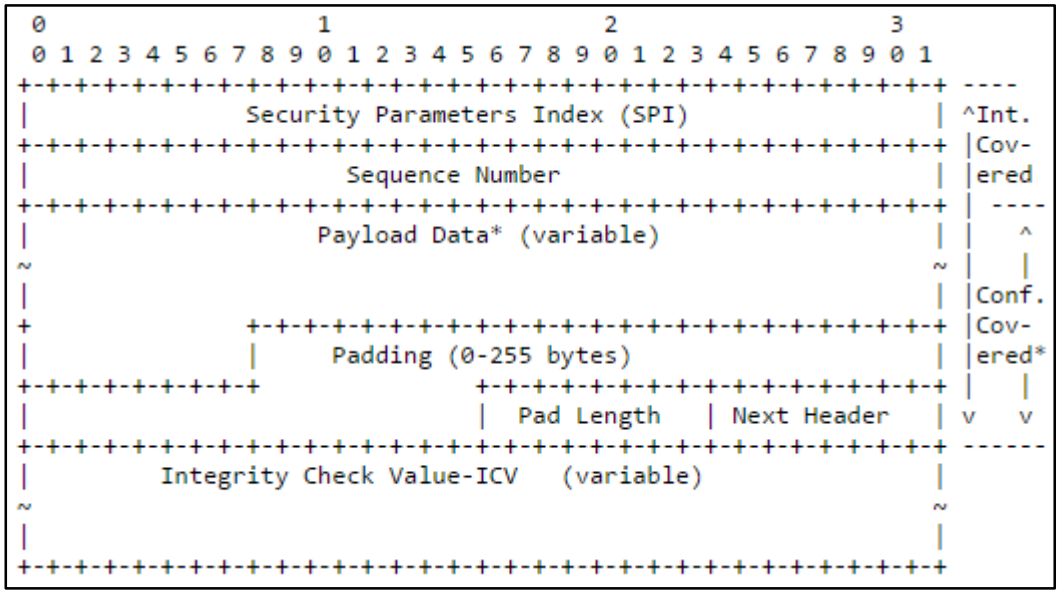


Kuvio 9. ESP-otsikon sijoitus kuljetustilassa (Kent 2005b, 17.)



Kuvio 10. ESP-otsikon sijoitus tunnelitilassa (Kent 2005b, 19.)

ESP-paketti määritellään IP-protokollalla 50. Paketti alkaa SPI-kentällä ja jatkuu Sequence Number -kentällä. Sen jälkeen tulee hyötykuorma, mutta sen rakenne on sidoksessa salausalgoritmeihin ja niiden tiloihin. ESP-traileri tulee hyötykuorman jälkeen ja tarvittaessa pakettiin lisätään täytettä (padding), jota seuraa Next header -kenttä. Valinnainen ICV-arvo päättää ESP-kapseloidun paketin (kts. kuvio 11). (Kent 2005b, 5.)

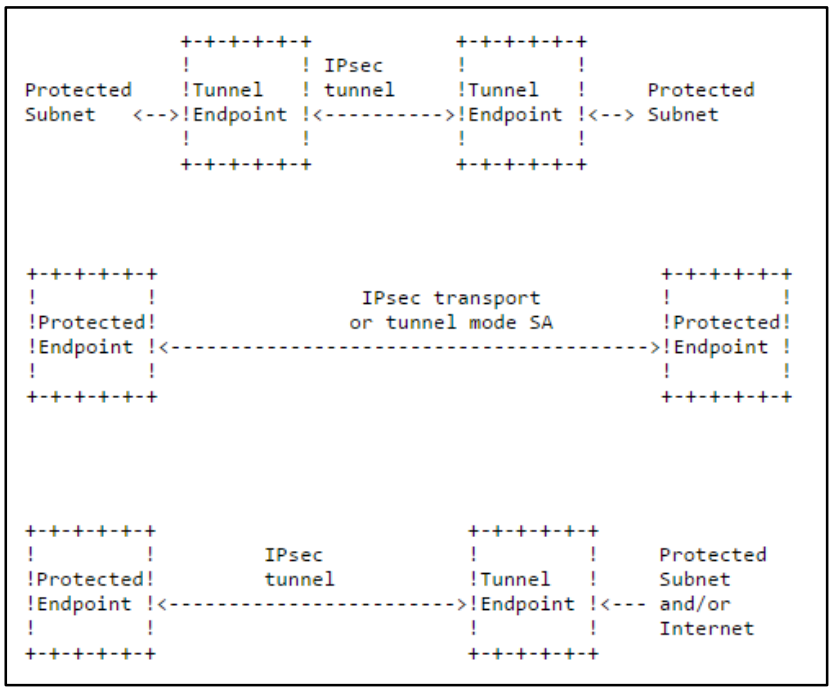


Kuvio 11. ESP-otsikon rakenne (Kent 2005b, 4.)

3.2.3 Internet Key Exchange-protokolla

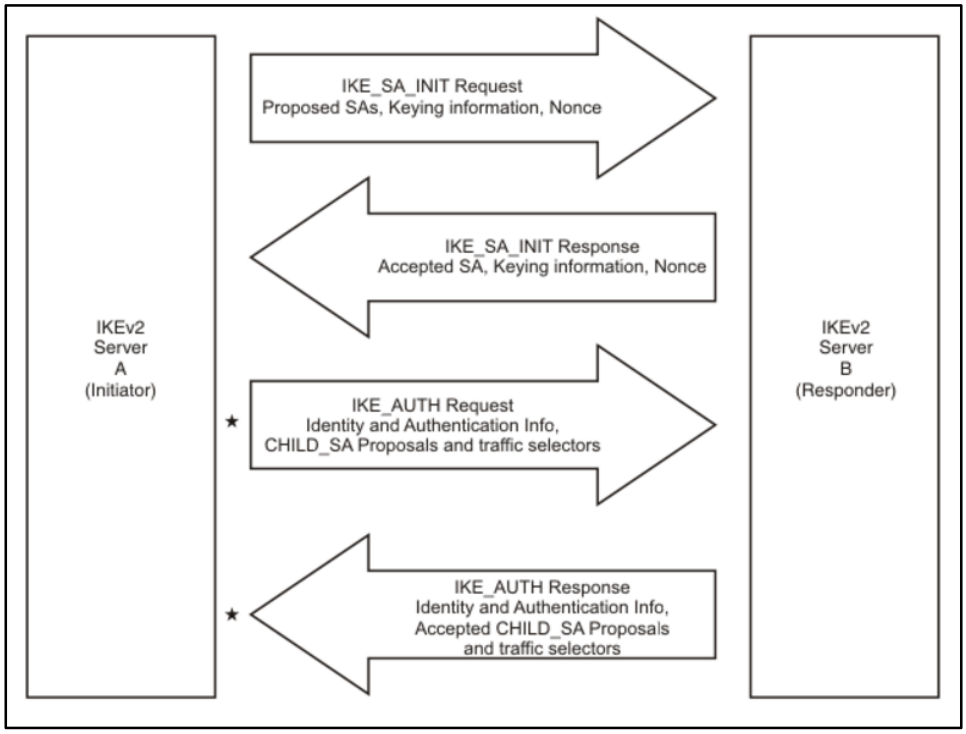
IPsec:n tarjotessa luottamuksellisuutta, datan eheyttä, autentikointia ja salausta, manuaalisesti IPsec:n toiminnalle tarvittavia parametreja on erittäin työlästä konfiguraoida. IKE-protokolla tarjoaakin yhtenäisen osallisten välisen autentikoinnin, jota kutsutaan suojaussidokseksi. Suojaussidos sisältää jaetun salaisuuden mm. (shared secret), jota käytetään ESP:n, AH:n ja kryptograafisen salauksen vakiinnuttamiseen. (Kaufman 2005, 3.)

IPsec:llä on kolme tyypillistä toimintaskenaariota (kts. kuvio 12): tunneli kahden IPsec-yhdyskäytävän välille, jolloin päätelaitteiden ei tarvitse tietää yhdyskäytävien olemassaolosta. Tällöin alkuperäinen IP-otsikko kapseloidaan ja ESP-otsikon eteen asetetaan uusi IP-otsikko vastaamaan IPsec-yhdyskäytävien osoitteistusta. Toinen skenaario on kahden päätelaitteen välinen suojaussidos tunneli- tai kuljetustilassa. Tällöin liikenne on koko matkalta suojattua. Tämä tapa ei kuitenkaan ole kovin suosittua, sillä se ei sovellu IPv4-verkkoihin sen rajoituksien myötä. Viimeinen skenaario on yksittäiseltä päätelaitteelta muodostettu tunneli IPsec-yhdyskäytävälle. Tyypillinen tapaus tälle skenaariolle on etätöitä tekevä työntekijä, jolla tulee olla suojattu yhteys yrityksen omaan verkkoon ja sitä kautta mahdollisesti Internetiin. (Kaufman 2005, 5-6.)



Kuvio 12. IPsec-skenaariot (Kaufman 2005, 5-6.)

Suojaussidosten muodostaminen perustuu aina kahden osallisen välillä käytettäviin pyyntö/vastaus-pareihin, eli kyseessä on aloitteentekijä ja vastaaja. Suojaussidosta muodostettaessa aloitteentekijä lähettää vastaanottajalle IKE-otsikon, suojaussidosehdotuksen, avainten vaihtoon liittyvän Diffie-Hellman-arvon ja oman tunnisteensa. IKE-otsikko sisältää SPI-indeksit, versionumerot sekä liput (flags). Suojaussidosehdotus pitää sisällään aloitteentekijän tukemat salausalgoritmit, jotka tukevat IKE-protokollaa. Vastaanottaja valitsee käytettävän salausalgoritmin aloitteentekijän tarjoamista vaihtoehdoista ja suorittaa Diffie-Hellman avainten muodostamisen. Vastaanottaja lähettää myös oman tunnisteensa. Tässä vaiheessa molemmat osapuolet muodostavat yhteisen jaetun avaimen perustuen lähettämiinsä Diffie-Hellman-arvoihin. Tämän jälkeen osapuolten välinen liikenne on salattua ja eheää. Ensimmäisen vaiheen jälkeen muodostetaan suojaussidokset datalle, joka liikennöi osapuolten kautta. Osapuolet muodostavat yksisuuntaiset tunnelit datalle. Kuviossa 13 on visualisoitu osapuolten välinen neuvottelu. (Kaufman 2005, 7-9.)



Kuvio 13. Suojaussidosten muodostus osapuolten välillä (IKE version 2 protocol, Initial exchanges N.d.)

4 Käytetyt komponentit

4.1 Open Network Operating System

Käytännön toteutuksessa verkon kontrollerina käytetiin Open Network Operating System eli ONOS-kontrolleria. Useimmat SDN-kontrollereista on käyttötarkoitukseltaan suunniteltu datakeskusympäristöihin, mutta ONOSissa on otettu huomioon myös verkko-operaattoreiden tarpeet tarjoamalla skaalautuvuutta, korkeaa saatavuutta ja tärkeimpänä applikaatioita, jotka on suunniteltu operaattoriverkon hallintaan. Kuten muutkin SDN-kontrollerit, myös ONOS on avoimen lähdekoodin ohjelma, joka julkaistiin 5. joulukuuta 2014. ONOS on saanut taakseen suuren määrän tukijoita, joista suurimpia on esimerkiksi AT&T, NTT communications, SK Telecom, Cisco, Fujitsu, Huawei ja Intel. (Open Network Operating System (ONOS) N.d)

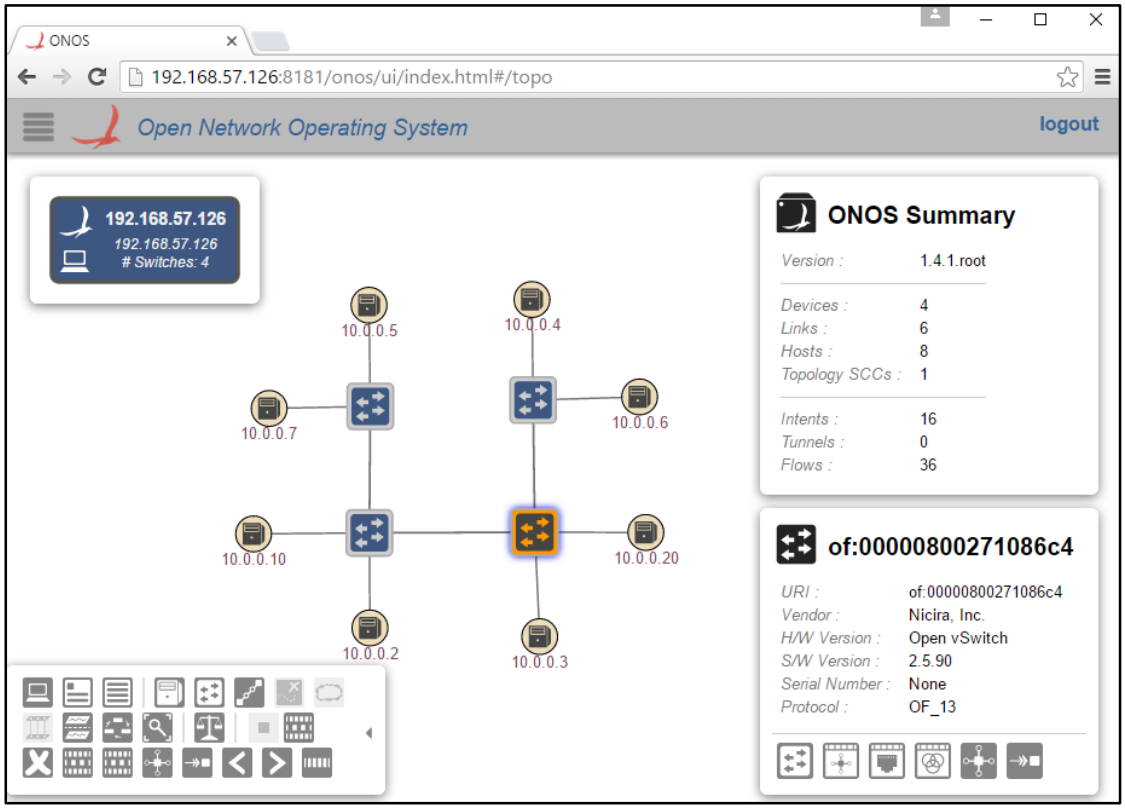
Skaalautuvuus ja korkea saatavuus on tuotu ONOSiin hajauttamalla kontrollitaso useamman ONOS-kontrollerin muodostamaksi klusteriksi. Klusteri-arkkitehtuuri ylläpitää verkon toimintavarmuutta, vaikka verkossa olisi usean laitteen kattava vikatila.

ONOS-klusteri kykenee myös jakamaan kontrollitason kuormaa dynaamisesti kontrollereiden välillä, josta on hyötyä silloin kun verkon kuormittuminen on erityisen radikaalia. (Rao 2015)

ONOSin suunnittelussa on otettu huomioon myös verkonhallinnan helppokäyttöisyys. Tämä tapahtuu intenteillä (eng. intent). Sen sijaan, että verkon ylläpitäjän tarvitsisi ymmärtää koko verkon monimutkainen toimintaperiaate, ylläpitäjälle riittää tieto liikenteen laadusta, päätepisteistä ja halutusta käyttäytymisestä, jotta verkkoon voidaan luoda liikennettä koskeva politiikka eli intentti. Intentille määritellään päätepisteet, jolloin ONOSin intent manager asentaa jokaiselle OpenFlow-kytkimelle päätepisteiden välillä tarvittavat vuot, jotta kommunikaatio päätepisteiden välillä onnistuu. Mikäli verkossa tapahtuu muutoksia päätepisteiden välillä, intentti muodostaa dynaamisesti uuden polun asettamalla uudet liikennettä ohjaavat vuot tarvittaville kytkimille. (Rao 2015)

Verkonhallintaa helpottaa myös ONOSin graafinen käyttöliittymä (kts. kuvio 14), jonka avulla verkon tilaa voidaan monitoroida reaaliajassa. Kuvion 13 vasemmassa alakulmassa olevilla painikkeilla voidaan muokata monitorointinäkymää vastaamaan ylläpitäjän tarpeita. Painikkeilla saadaan näkymään mm.

- ONOS-klusterin instanssit
- Verkon yleisnäkymää kuvastava paneeli
- Verkkoelementin yksityiskohtainen paneeli
- Päätelaitteiden näkyminen topologiassa
- porttinumeroinnin korostus
- Liikennemäärät linkkiväleillä
- Voiden määrä linkkiväleillä
- Intenttien määrä linkkiväleillä



Kuvio 14. ONOSin graafinen käyttöliittymä

4.2 Open vSwitch

Open vSwitch eli OVS on monikerroksinen avoimen lähdekoodin OpenFlow-protokollalla toimiva virtuaalikytkin. Sen tarkoituksena on tarjota kytkinpino virtuaalisiin ympäristöihin kuitenkin tukien myös vanhoja protokollia ja mahdollistaa tehokas verkon automatisointi käyttäen ohjelmoitavia laajennuksia. Linuxin kernel-implementaatio tapahtui maaliskuussa 2012, jolloin viralliset paketit OVS:sta tulivat Debianille, Fedoralle ja Ubuntulle. Myöhemmin tammikuussa 2014 paketit julkaistiin myös FreeBSD:lle ja NETBSD:lle. Suurin osa OVS:sta on kirjoitettu C-ohjelmointikielellä, joka mahdollistaa siirrettävyyden useisiin ympäristöihin. OVS tukee mm. seuraavia ominaisuuksia OpenFlow-protokollan lisäksi:

- NetFlow, sFlow, IPFIX, SPAN, RSPAN, port mirroring
- Link aggregation (LACP)
- 802.1Q VLAN
- Multicast snooping
- STP / RSTP
- QoS, Traffic policing

- IPv6

OVS on suunniteltu tukemaan läpinäkyvää verkon levitystä fyysisten palvelimien välillä niin, että se erottelee alla olevan palvelinarkkitehtuurin verkosta. (Why Open vSwitch? 2014)

OVS-kytkin koostuu kahdesta prosessista; switchd ja ovsdb eli kytkin ja sen tietokanta. Kytkimen asennuksen yhteydessä palvelimelle asentuu myös useita kytkimen hallintaan tarkoitettuja ohjelmia. Useimmin niistä käytetyt ovat ovs-vsctl ja ovs-ofctl. Ovs-vsctl-ohjelmaa käytetään switchd-prosessin konfiguroimiseen syöttämällä konfiguraatioita kytkimen tietokantaan. Tyypillisimpiä ovs-vsctl –ohjelman komentoja ovat esimerkiksi uusien virtuaalikytkinten luominen ja virtuaalisten rajapintojen liittäminen kytkimeen kuten seuraavassa esimerkissä on tehty. (Open vSwitch Manual N.d a.)

```
root@ubuntu-vm:~# ovs-vsctl add-br br-int
root@ubuntu-vm:~# ovs-vsctl add-port br-int veth0
```

Ovs-ofctl-ohjelmaa käytetään kytkinten hallintaan ja monitorointiin. Tyypillisimpiä käyttökohteita ovat tilatietojen kyselyt sekä vuotaulujen luominen ja muokkaaminen. Kuvioissa 15 ja 16 on esitetty OVS-kytkimen porttien tilatietojen ja vuotaulujen kysely. (Open vSwitch Manual N.d b.)

```
root@Docker-OVS4:~# ovs-ofctl show br-int
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000ae9517b0b043
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_src mo
d_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(geneve1): addr:ee:c8:9c:50:8d:89
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(geneve0): addr:5a:ba:a6:4e:0d:43
     config:     0
     state:      STP_FORWARD
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(br-int): addr:ae:95:17:b0:b0:43
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
root@Docker-OVS4:~#
```

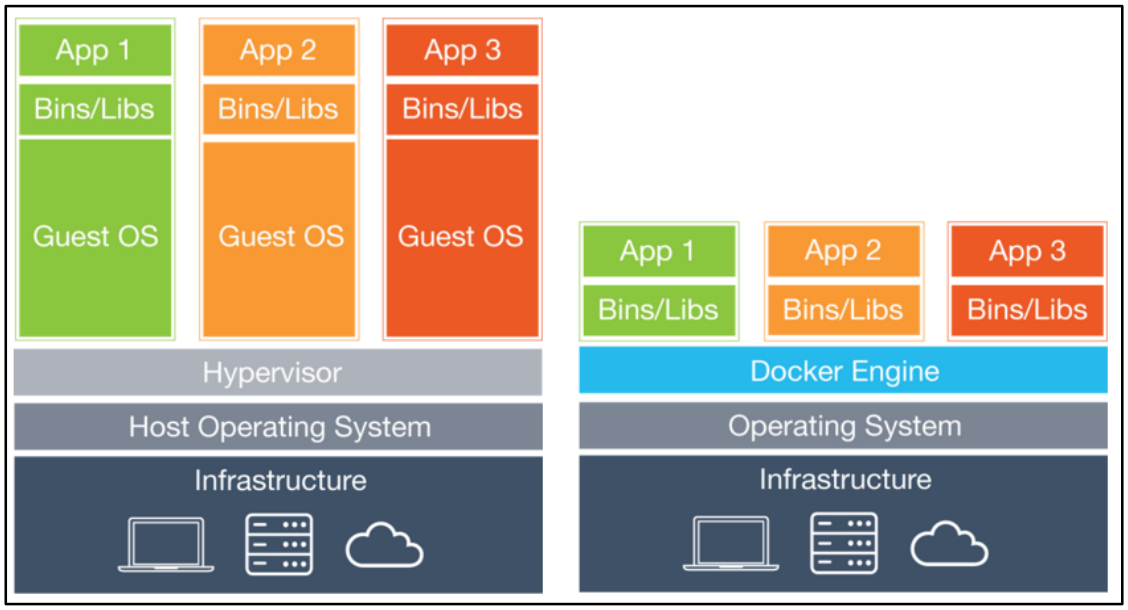
Kuvio 15. OVS-kytkimen porttien tilatietojen hakeminen

```
root@Docker-OVS4:~# ovs-ofctl dump-flows br-int
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=208.870s, table=0, n_packets=477, n_bytes=34290, idle_age=0, actions=NORMAL
root@Docker-OVS4:~#
```

Kuvio 16. OVS-kytkimen vuotaulujen tulostus

4.3 Docker

Docker on avoin konttipohjainen alusta, joka on kehitetty helpottamaan ja nopeuttamaan sovelluskehityksen vaiheita. Dockerin periaatteena on sovelluksen ja kaikkien sen tarvittavien riippuvuuksien pakkaaminen konttiin, jolloin kontti on mahdollista ottaa käyttöön jokaisella Dockeria tukevalla alustalla ilman ristiriitaisuuksia. Kontittamalla sovellukset eristettyihin ympäristöihin, mutta jakamalla yhteinen kernel-taso konteille mahdollistaa kevyemmän ja kustannustehokkaamman ympäristön luomisen verrattuna virtuaalikoneisiin, joita pyöritettäisiin hypervisorin päällä. Klassista virtualisointi-arkkitehtuuria ja docker-arkkitehtuuria on vertailtu kuviossa 17. (Understand the architecture N.d)

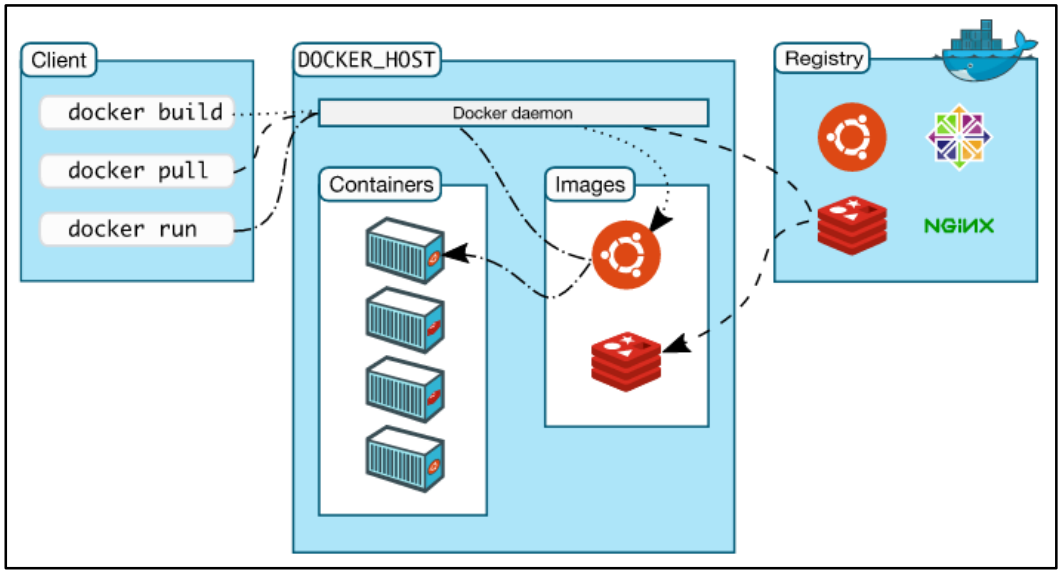


Kuvio 17. Ero virtualisoinnin ja Dockerin välillä (Understand the architecture N.d)

Docker on jaettu kahteen pääkomponenttiin; Docker-alusta, joka on avoimen lähdekoodin kontitusalusta ja DockerHub, joka toimii SaaS-alustana konttien jakamiselle ja hallinnoimiselle. Perustana Docker-alustalle on client-server-arkkitehtuuri, jossa Docker-client käskee Docker-daemonia rakentamaan, suorittamaan tai jakamaan kontteja käyttäen RESTful-ohjelmointirajapintaa. (Understand the architecture N.d)

Dockerin sisältö perustuu docker-levykuviin (image), -rekistereihin ja itse kontteihin. Levykuvat ovat read-only –templaatteja, joita käytetään konttien luomiseen. Tyypillinen levykuva voi sisältää esimerkiksi ubuntu-käyttöjärjestelmän ja siihen asennetun Apache-palvelinohjelman. Docker-rekisterien tarkoituksena on säilyttää templaatteja. Rekisterit voivat olla yksityisiä tai julkisia, joista käyttäjät voivat hakea valmiita docker-levykuvia. DockerHub on suurin julkinen Docker-rekisteri, johon käyttäjät voivat lisätä yleiseen jakoon tarkoitettuja levykuvia. Docker-kontit ovat hieman kuten hakemisto, se sisältää kaiken mitä kontti tarvitsee sovelluksen suorittamiseen ja jokainen kontti luodaan Docker-levykuvasta. (Understand the architecture N.d)

Konttia luodessa, client aloittaa prosessin käskemällä daemonia tarkistamaan onko levykuvaa paikallisesti isäntäkoneen hakemistossa. Mikäli isäntäkoneella ei ole tarvittavaa levykuvaa, daemon hakee sen oletuksena DockerHubista. Sitten daemon käyttää haettua levykuvaa kontin luomiseen, jonka jälkeen kontille allokoidaan hakemisto isäntäkoneen tiedostojärjestelmästä. Kontille allokoidaan myös verkkorajapinta, joka mahdollistaa kommunikaation isäntäkoneen kanssa. Viimeisenä kontti suorittaa sille annetut tehtävät esimerkkinä Apache-prosessin käynnistämisen. Kuviossa 18 on vielä visualisoitu clientin, daemonin ja rekisterin roolit konttia luodessa. (Understand the architecture N.d)



Kuvio 18. Docker-kontin luominen (Understand the architecture N.d)

4.4 strongSwan

strongSwan on avoimen lähdekoodin IPsec-implementaatio Linuxin kerneleille 2.6 ja 3.x, joka perustuu jo lakkautettuun FreeS/Wan-projektiin. strongSwan-projektin painottaa vahvaa autentikointia käyttämällä X.509 julkisten avainten hallintajärjestelmää, konfiguraation yksinkertaisuutta ja vahvoja IPsec-politiikkoja tukien suuria ja monimutkaisia VPN-verkkoja. Kaikki kolme luvussa 3.2.3 olevaa tapausta voidaan toteuttaa käyttämällä strongSwania. Projekti tukee myös molempia IKE-protokollan v1 ja v2-versioita. (strongSwan 2015)

4.5 Ansible

4.5.1 Yleistä

Ansible on automatisointityökalu, jolla voidaan suorittaa mm. pilviympäristön provisiointia, konfiguraatiohallintaa, sovellusten käyttöönottoja, orkestrointia ja monia muita automatisointiin liittyviä tehtäviä. Ansiblen arkkitehtuuri perustuu monitasoiseen käytettävyyteen, joka mahdollistaa usean järjestelmän hallinnan samanaikaisesti. Järjestelmien hallintaan Ansible ei käytä lainkaan agentteja, sillä kaikki tehtävät

tehdään ainoastaan käyttämällä SSH-yhteyksiä. Tehtävien luomiseen käytetään yksinkertaista ja helposti luettavaa YAML-ohjelmointikieltä, joka perustuu Python-ohjelmointikieleen. Ansiblen tehokas arkkitehtuuri perustuu pieniin ohjelmiin eli moduuleihin, jotka työntävät tehtävissä määriteltyjä käskyjä etäjärjestelmiin. Moduulikirjasto voi sijaita millä tahansa koneella, eikä se tarvitse minkäänlaisia taustaprosesseja tai tietokantoja. Ansible ei myöskään tee kohdejärjestelmään muutoksia, jos se havaitsee, että kohde on jo valmiiksi playbookin eli tehtävälistan määrittelemässä tilassa. Ansiblen hosts-tiedostolla voidaan luoda yksittäisistä järjestelmistä ryhmiä, jolloin tehtäviä voidaan kohdistaa useammalle järjestelmälle samanaikaisesti. Seuraavassa esimerkissä on yksinkertainen hosts-tiedosto, jossa on neljä järjestelmää, jotka on jaettu kahteen ryhmään nimeltä ”webservers” ja ”dbservers”. (Overview: How Ansible works N.d)

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```

4.5.2 Playbookit ja roolit

Ansiblen tehtävät suoritetaan playbookkeina, jotka kirjoitetaan YAML-ohjelmointikielellä. Playbookeilla voidaan määritellä yksityiskohtaiset vaiheet, jotka suoritetaan playbookissa määritellyille järjestelmille. Playbookillla voidaan asettaa järjestelmille rooleja, jotka sisältävät tehtäviä, jolloin automatisoinnista voidaan tehdä entistä hienojakoisempaa. Seuraavassa on esimerkki YAML-kielellä kirjoitetusta playbookista, joka asettaa ryhmässä ”webservers” olevat järjestelmät rooliin ”webapp”

```
---
- hosts: webservers


roles:
- webapp
```

Ansible etsii hakemistostaan roles/tasks/main.yml-tiedostoa, johon on määritelty roolin tehtävät esimerkiksi seuraavasti:

```
---
- yum: name=app_server state=installed

- service: name=app_server state=running enabled=yes
```

Tehtävässä kohdejärjestelmille asennetaan app_server-palvelu käyttämällä yum-moduulia, jonka jälkeen palvelu käynnistetään service-moduulia hyödyntäen.

5 Käytännön toteutus

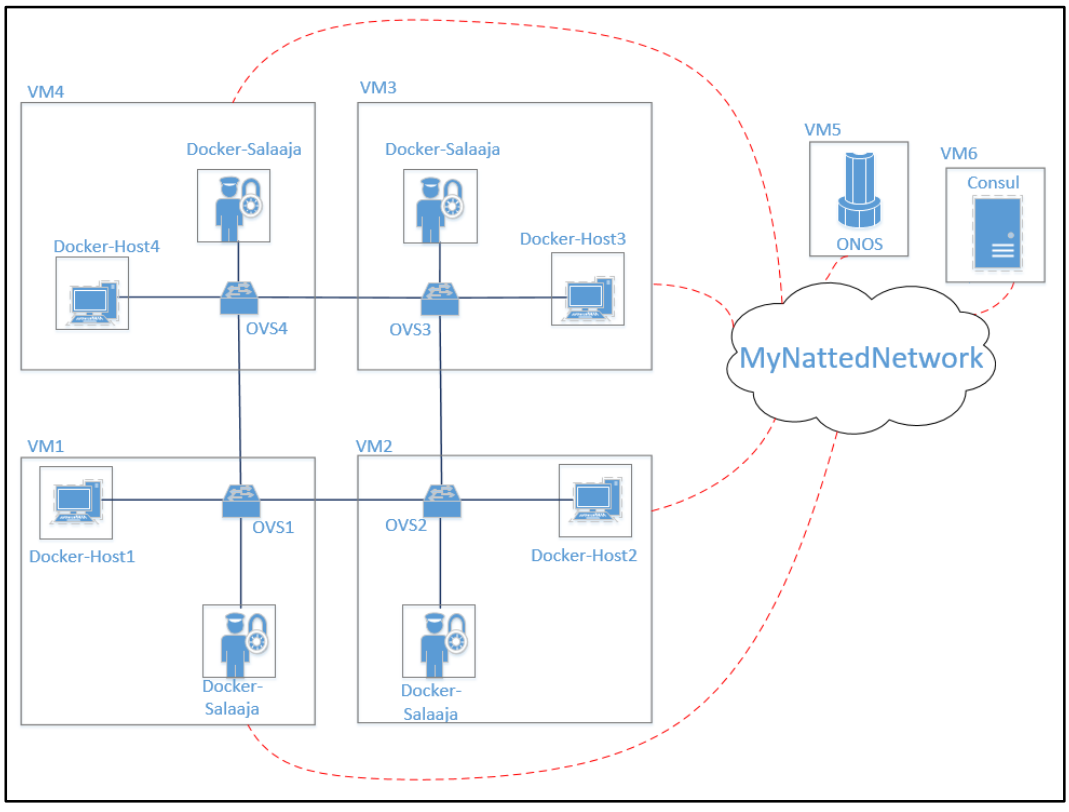
5.1 Ympäristön ja topologia

Ympäristössä käytettiin neljän kytkimen muodostaa verkkotopologiaa, jolla simuloitiin Cyber Trust -projektin testbed-ympäristöä. Työ toteutettiin käyttämällä virtuaalikoneita ja virtualisointialustana toimi Oracle VM Virtualbox. Virtuaalikoneita oli käytössä kuusi kappaletta, jotka on listattu taulukkoon 1. Kaikki ympäristön virtuaalikoneet käyttivät Ubuntu Server 14.04.3 käyttöjärjestelmää ja jokaisella koneella oli yhteys Internettiin Virtualboxin NatNetwork-verkkoadapterin kautta.

Taulukko 1. Ympäristön laitteisto

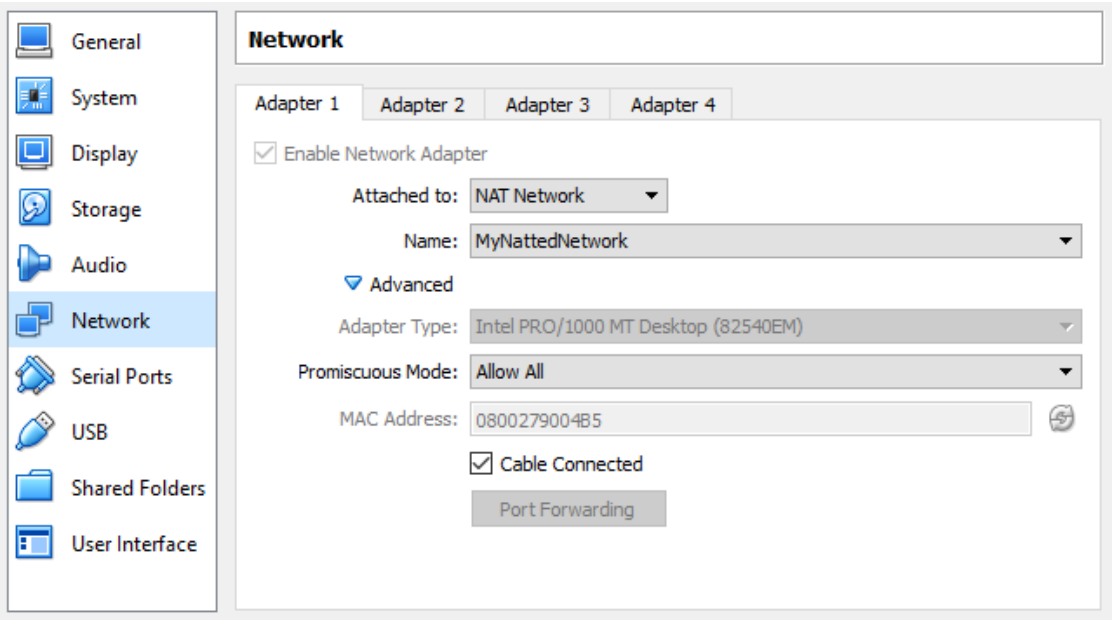
Tunnus	Hostname	Rooli	Rajapinnat	IP-osoite	Verkko
VM1	OVS1	OVS-kytkin / asiakaslaite	eth0	10.0.2.21/24	MyNattedNetwork
			eth1	12.0.0.1/24	VM1-VM2
			eth2	14.0.0.1/24	VM1-VM4
VM2	OVS2	OVS-kytkin / asiakaslaite	eth0	10.0.2.22/24	MyNattedNetwork
			eth1	12.0.0.2/24	VM1-VM2
			eth2	23.0.0.1/24	VM2-VM3
VM3	OVS3	OVS-kytkin / asiakaslaite	eth0	10.0.2.30/24	MyNattedNetwork
			eth1	23.0.0.2/24	VM2-VM3
			eth2	34.0.0.1/24	VM3-VM4
VM4	OVS4	OVS-kytkin / asiakaslaite	eth0	10.0.2.31/24	MyNattedNetwork
			eth1	14.0.0.2/24	VM1-VM4
			eth2	34.0.0.2/24	VM3-VM4
VM5	Consul	Docker-consul / automatisointi	eth0	10.0.2.23/24	MyNattedNetwork
VM6	ONOS	ONOS-kontrolleri	eth0	10.0.2.24/24	MyNattedNetwork

Verkkotopologia on esitetty kuviossa 19, jossa punaisella katkoviivalla on esitetty kytkimien kontrolli- ja laitteiden hallinta- ja Internet-yhteydet käyttämällä verkkokortin NAT-ominaisuutta.

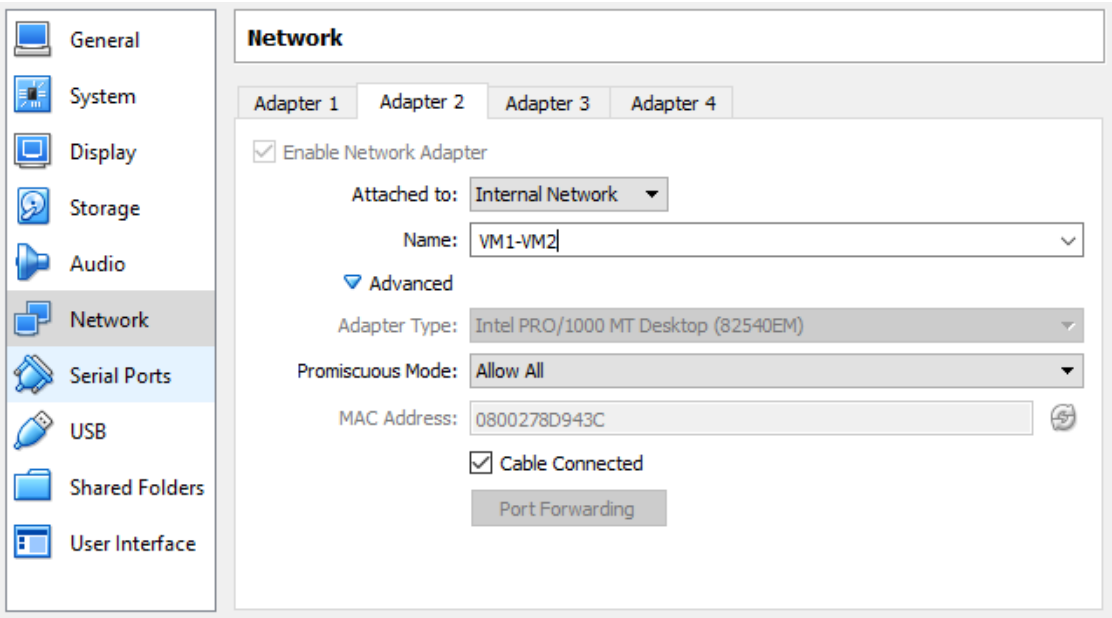


Kuvio 19. Toteutuksen verkkotopologia

Topologian muut linkit on toteutettu käyttämällä Internal Network –asetusta, jolloin virtuaalisia verkkoja voidaan eritellä käyttäen yksillöllistä nimeämiskäytäntöä (kts. Kuviot 20 ja 21). Toteutuksessa verkot nimettiin taulukon 1 ”verkko”-sarakkeen mukaisesti, jossa verkot on nimetty verkon päätepisteiden mukaisesti vastaamaan kuvion 19 virtuaalikoneiden tunnuksia.



Kuvio 20. Virtuaalikoneiden hallintayhteys NAT Network –asetuksen kautta



Kuvio 21. Virtuaalikoneiden verkotus käyttäen Internal Network –asetusta

Toimeksiantajan vaatimukset olivat hyvin avoimet salauksen toteutuksen suhteen, sillä ympäristö toimi täysin suljettuna laboratorioympäristönä. Salauskomponentit päätettiin sijoittaa Docker-kontteihin, jolloin salauksen käyttöönotto olisi vaivatonta millä tahansa Dockeria tukevalla käyttöjärjestelmällä. Myös ympäristössä olevat asiakaslaitteet suoritettiin Docker-konteissa fyysisen isäntäkoneen resurssien säästämisen vuoksi.

5.2 ONOS asennus

Verkkoympäristön kontrolleriksi valittiin ONOS, koska siitä oli jo vahva kokemus Cyber Trust -projektissa. Perusteina ONOSin valintaan oli myös valmiiksi löytyvä org.onosproject.fwd-applikaatio, joka ohjaa liikennettä OSI-mallin L2-tasolla. Toisena tärkeänä perusteena oli toimiva liikenteenohjaus hyödyntämällä intenttejä. ONOS tarvitsi toimiakseen apache karaf- ja apack maven-ohjelmiston sekä javan, jotka otettiin huomioon asennusvaiheessa. ONOSin asennus aloitettiin seuraavilla komennoilla:

```
root@ONOS:~# apt-get update
root@ONOS:~# apt-get install openssh-server git-core wget
root@ONOS:~# mkdir Downloads Applications
```

```
root@ONOS:~# cd Downloads

root@ONOS:~/Downloads# wget
http://archive.apache.org/dist/karaf/3.0.3/apache-karaf-
3.0.3.tar.gz

root@ONOS:~/Downloads# wget
http://archive.apache.org/dist/maven/maven-
3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz

root@ONOS:~/Downloads# tar -zxvf apache-karaf-3.0.3.tar.gz -C
../Applications/

root@ONOS:~/Downloads# tar -zxvf apache-maven-3.3.9-bin.tar.gz
-C ../Applications/

root@ONOS:~# apt-get install software-properties-common -y

root@ONOS:~# add-apt-repository ppa:webupd8team/java -y

root@ONOS:~# apt-get update

root@ONOS:~# apt-get install oracle-java8-installer oracle-
java8-set-default –y

root@ONOS:~# git clone https://gerrit.onosproject.org/onos

root@ONOS:~# cd ~/onos

root@ONOS:~# git checkout onos-1.4
```

Sitten lisättiin seuraavat kolme riviä ~/.bash.rc-tiedoston loppuun:

```
. ~/onos/tools/dev/bash_profile

export ONOS_USER=root

export ONOS_GROUP=root
```

Tämän jälkeen asennusta jatkettiin seuraavasti:

```
root@ONOS:~# . ~/.profile

root@ONOS:~# apt-get install maven

root@ONOS:~# cd ~/onos

root@ONOS:~/onos# mvn clean install
```

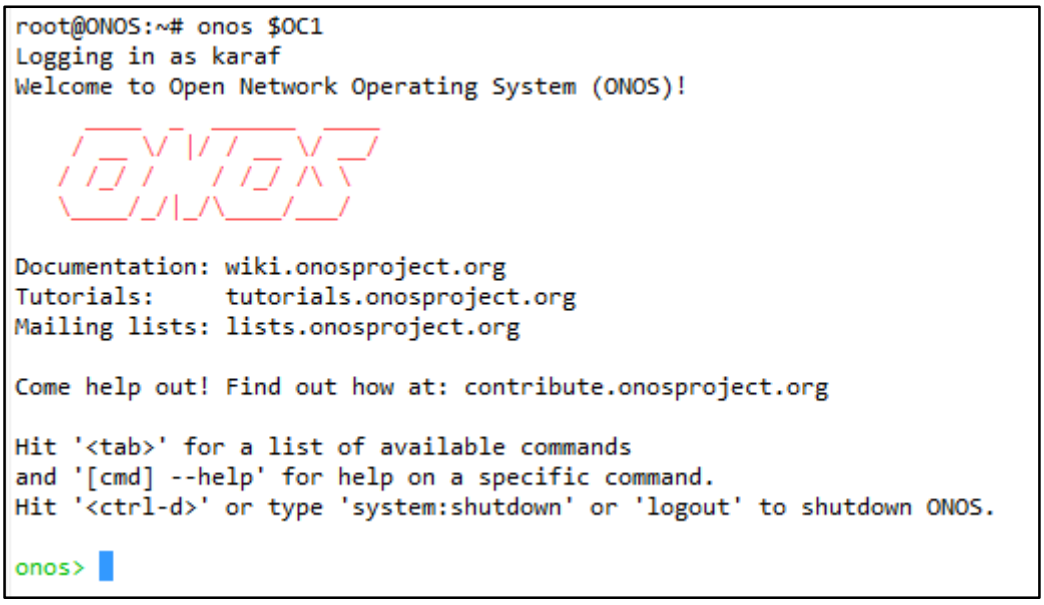
Seuraavaksi lisättiin kaksi riviä ONOSin ~/onos/tools/test/cells/local-tiedostoon:

```
export OC1=10.0.2.24
export ONOS_APPS="drivers,OpenFlow,fwd"
```

ONOSin asennus viimeisteltiin seuraavasti:

```
root@ONOS:~/onos# cell local
root@ONOS:~/onos# onos-push-keys 10.0.2.24
root@ONOS:~/onos# onos-install -f $OC1
```

Tämän jälkeen ONOS on asennettu ja komentolinjalle päästiin komennolla *"onos $OC1"*. ONOSIN käyttöliittymä on esitetty kuviossa 22.



Kuvio 22. ONOSin käyttöliittymä kirjautumisen jälkeen

5.3 OVS asennus

OpenFlow-kytkimiksi toteutukseen valittiin Open vSwitch hyvän dokumentaation ja vahvan aikaisemman kokemuksen vuoksi. Ympäristössä käytettiin neljää kytkintä, jotka olivat OVS:n versiota 2.5.1. Kyseinen versio oli olennainen opinnäytetyön toteutukselle, koska siinä esiteltiin tuki Geneve-tunneloinnille ja Dockerin implementoimiselle. OVS asennettiin suoraan lähdekoodista seuraavasti kaikille neljälle kytkimelle. Esimerkkinä seuraavaksi asennus OVS1-virtuaalikoneelle:

```
root@OVS1:~# apt-get install -y autoconf libtool sparse openssl
pkg-config make gcc libssl-dev git

root@OVS1:~# git clone https://github.com/openvswitch/ovs.git

root@OVS1:~# cd ovs

root@OVS1:~/ovs# git checkout branch-2.5

root@OVS1:~/ovs# ./boot.sh

root@OVS1:~/ovs# ./configure --prefix=/usr --localstatedir=/var
--sysconfdir=/etc --enable-ssl --with-linux=/lib/modules/`uname
-r`/build

root@OVS1:~/ovs# make -j3

root@OVS1:~/ovs# make install

root@OVS1:~/ovs# cp debian/openvswitch-switch.init
/etc/init.d/openvswitch-switch

root@OVS1:~/ovs# apt-get install python-openvswitch

root@OVS1:~/ovs# apt-get install python-setuptools

root@OVS1:~/ovs# easy_install -U pip

root@OVS1:~/ovs# pip install Flask

root@OVS1:~/ovs# /etc/init.d/openvswitch start
```

5.4 Docker asennus

Toteutetussa ympäristössä Docker asennettiin jokaiselle OVS-virtuaalikoneelle sekä MyNattedNetwork-verkossa olevalle Consul-virtuaalikoneelle. Docker asennettiin seuraavasti Consulille:

```
root@consul:~# apt-get update

root@consul:~# apt-get install apt-transport-https ca-
certificates

root@consul:~# apt-key adv --keyserver hkp://p80.pool.sks-
keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
```

Tämän jälkeen lisättiin seuraava rivi eli Dockerin repositorio /etc/apt/sources.list.d/docker.list-tiedostoon.

```
deb https://apt.dockerproject.org/repo ubuntu-trusty main
```

Sitten asennettiin uudet Dockerin tarvitsemat paketit:

```
root@consul:~# apt-get update

root@consul:~# apt-cache policy docker-engine

root@consul:~# apt-get install linux-image-extra-$(uname -r)

root@consul:~# apt-get install apparmor

root@consul:~# apt-get install docker-engine
```

5.5 Kytkinten ja Dockerin liittäminen ympäristöön

Ennen kuin verkon välitystasolle tuotiin uusia laitteita, OVS-kytkimet piti liittää Consuliin hallintaverkon kautta. Consulille oli oma virtuaalikone sijoitettuna taulukon 1 mukaisesti IP-osoitteeseen 10.0.2.23, jossa Consulia suoritettiin Docker-kontissa.

Consulin tehtävänä oli synkronoida OVS1-kytkimellä oleva OVS-tietokanta muille verkon kytkimille. Consul-virtuaalikoneella suoritettiin myös Dockerin privaattia rekisteriä, jonne ladattiin salauskomponentin Docker-levykuva. Dockerin käynnistämiseen käytettiin lisäparametriä, jolla määriteltiin rekisterin sijainti, joten Docker sammutettiin komennolla *"service docker stop"*, jonka jälkeen se käynnistettiin seuraavasti:

```
root@consul:~# docker daemon --insecure-registry=10.0.2.23:5000
&
```

Consul- ja registry-kontit käynnistettiin komennoilla:

```
root@consul:~# docker run -d -p 8500:8500 -h consul –
name=consul progrium/consul -server --bootstrap

root@consul:~# docker run -d -p 5000:5000 --name registry
registry:2
```

Seuraavaksi OVS1-virtuaalikoneella käynnistettiin Docker lisäparametrein määritellen Consulin sijainti ja Consulille mainostettava oma hallintaverkon IP-osoite. Tämän lisäksi avattiin OVS-tietokannan portti, tietokannan northbound-rajapinta, määriteltiin tietokannan sijainti ja käynnistettiin OVS-kytkin Docker-ajurilla komennoilla:

```
root@OVS1:~# docker daemon --cluster-
store=consul://10.0.2.23:8500 --cluster-advertise=10.0.2.21:0 –
insecure-registry=10.0.2.23:5000 &

root@OVS1:~# ovs-appctl -t ovsdb-server ovsdb-server/add-remote
ptcp:6640

root@OVS1:~# /usr/share/openvswitch/scripts/ovn-ctl
start_northd

root@OVS1:~# ovs-vsctl set Open_vSwitch . external_ids:ovn-
remote="tcp: 10.0.2.21:6640" external_ids:ovn-encap-
ip=10.0.2.21 external_ids:ovn-encap-type="geneve"

root@OVS1:~# ovn-docker-overlay-driver –detach
```

Komennolla *”ovs-vsctl show”* tarkistettiin, että kytkin nimeltä br-int on luotu kuvion 23 mukaisesti.

```
root@OVS1:~# ovs-vsctl show
45437884-1a6d-4033-b53b-c0ec910f8113
    Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.5.1"
root@OVS1:~#
```

Kuvio 23. OVS-kytkimen luomisen todennus

Tämän jälkeen ympäristön muut OVS-koneet liitettiin myös OVS1:n tietokantaan ja Consuliin seuraavasti:

```
root@OVS2:~# docker daemon --cluster-store=consul://
10.0.2.23:8500 --cluster-advertise=10.0.2.22:0 –insecure-
registry=10.0.2.23:5000 &

root@OVS2:~# ovs-vsctl set Open_vSwitch . external_ids:ovn-
remote="tcp:10.0.2.21:6640" external_ids:ovn-encap-ip=10.0.2.22
external_ids:ovn-encap-type="geneve"

root@OVS2:~# ovn-docker-overlay-driver –detach
```

Huomioitavaa tässä oli IP-osoitteiden vaihtuminen vastaamaan OVS2-koneen IP-osoitetta kentissä *”--cluster-advertise*=10.0.2.22*:0”* ja *”external_ids:ovn-encap-ip*=10.0.2.22*”*.

Seuraavaksi OVS-koneiden väliset rajapinnat liitettiin kytkimiin käyttäen geneve-tunnelointiprotokollaa. Geneve-tunneloinnin tarkoituksena on erottaa fyysisten laitteiden liikenne alla kulkevasta virtuaalisten laitteiden liikenteestä. Pääasiallisena käyttötarkoituksena geneve-tunneloinnilla on yhdistää konesaleissa eri palvelinkehikoissa olevia virtuaalikytkimiä toisiinsa. (Ganga & Gross 2016, 4-5.)

OVS-virtuaalikoneiden väliset Internal Network –adapterit nimettiin vastaamaan ympäristössä olevien virtuaalikoneiden tunnuksia (kts. taulukko 1), joten samaa käytäntöä jatkettiin geneve-tunneleiden verkkojen aliverkottamisessa. Esimerkkinä OVS1- ja OVS2-koneiden väliseen verkkoon annettiin IP-osoitteet 12.0.0.0/24-osoiteavaruudesta antaen pienemmän tunnuksen omaavalle koneelle ensimmäinen osoiteavaruuden osoite (12.0.0.1) ja suuremmalle tunnukselle seuraavan osoite 12.0.0.2. Osoitteet määriteltiin /etc/network/interfaces-tiedostoon (kts. kuvio 24), jotta osoitteet pysyivät samoina mahdollisen uudelleenkäynnistyksen yhteydessä.

```
root@OVS1:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

#OVS1-OVS2
auto eth1
iface eth1 inet static
address 12.0.0.1
netmask 255.255.255.0

#OVS1-OVS4
auto eth2
iface eth2 inet static
address 14.0.0.1
netmask 255.255.255.0

root@OVS1:~#
```

Kuvio 24. OVS1-virtuaalikoneen rajapintojen osoitteet

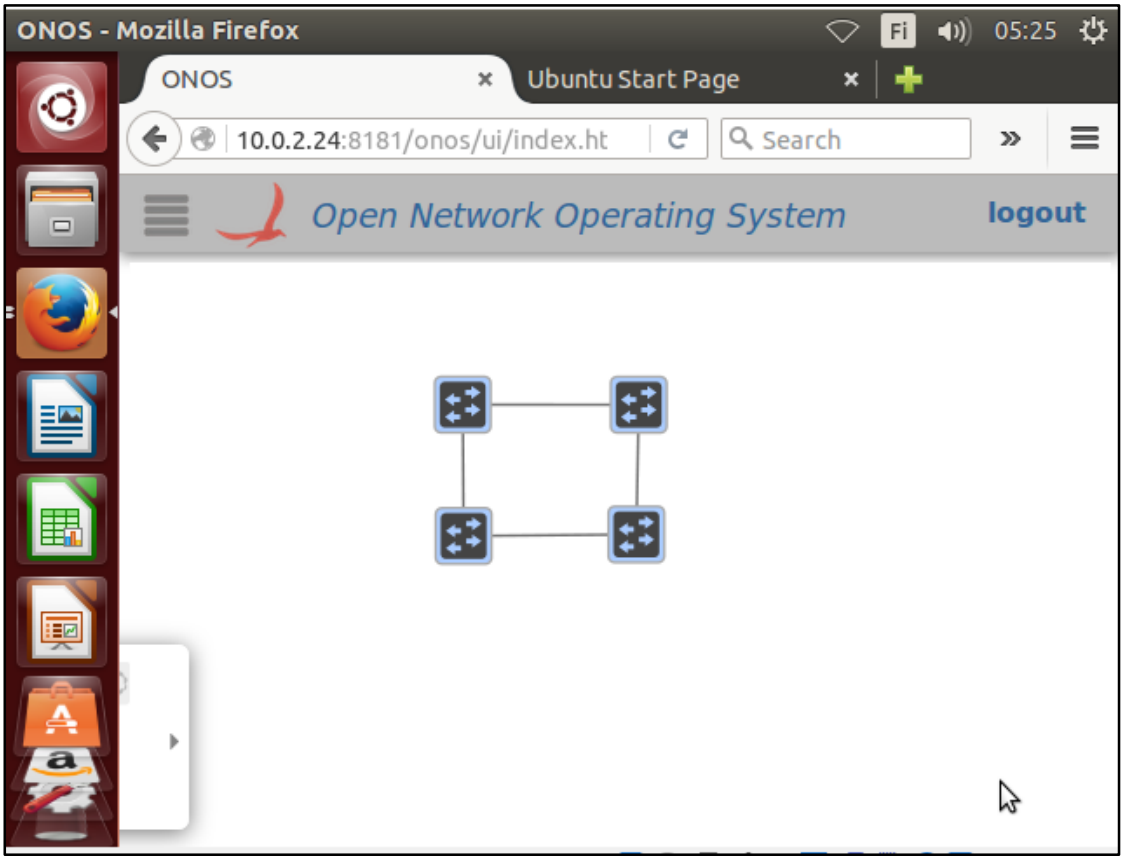
Geneve-tunnelointia tukevat rajapinnat luotiin OVS1-kytkimelle luotiin komennoilla:

```
root@OVS1:~# ovs-vsctl add-port br-int geneve0 -- set interface
geneve0 type=geneve options:remote_ip=12.0.0.2

root@OVS1:~# ovs-vsctl add-port br-int geneve1 -- set interface
geneve1 type=geneve options:remote_ip=14.0.0.2
```

Komennossa annettiin uusille porteille nimet geneve1 ja geneve2 ja määriteltiin tunneleiden vastakkaiset päät vastaamaan OVS2- ja OVS4-koneiden IP-osoitteita. Vastaavilla komennoilla konfiguroitiin kaikki kytkinten väliset linkit. Viimeisenä kytkimet liitettiin verkon kontrolleriin komennolla *”ovs-vsctl set-controller br-int tcp:10.0.2.24:6633”.* Kuviossa 25 on suoritettu komento *”ovs-vsctl show”*, jossa on esitetty kontrolleriyhteyden muodostuminen ja kytkimeen liitetyt geneve-rajapinnat. Kuviossa 26 on esitetty ONOSin graafinen webbikäyttöliittymä, jossa näkyi kytkinten välille geneve-rajapintojen muodostuneet linkkivälit.

```
root@OVS1:~# ovs-vsctl show
45437884-1a6d-4033-b53b-c0ec910f8113
    Bridge br-int
        Controller "tcp:10.0.2.24:6633"
            is_connected: true
        fail_mode: secure
        Port "geneve1"
            Interface "geneve1"
                type: geneve
                options: {remote_ip="14.0.0.2"}
        Port br-int
            Interface br-int
                type: internal
        Port "geneve0"
            Interface "geneve0"
                type: geneve
                options: {remote_ip="12.0.0.2"}
    ovs_version: "2.5.1"
root@OVS1:~#
```

Kuvio 25. Kytkimen kontrolleriyhteys ja geneve-rajapinnat



Kuvio 26 ONOSin webbikäyttöliittymä

6 Salauksen implementointi verkkoon

6.1 Salauskomponentin luonti

Ympäristön pystytyksen jälkeen aloitettiin salauksen implementointi verkkoon. Salauskomponentti toteutettiin Docker-kontissa käyttäen mukautettua konttia, joka perustui GitHubista valmiiksi löytyneeseen philplckthun/strongswan-konttiin. Kontin Dockerfile-tiedostoon tehtiin kuitenkin pieniä muutoksia, jotta se soveltuisi opinnäytetyön toteutukseen.

Jokainen Dockeria tarvitsevan koneen Docker-prosessi käynnistettiin ”--insecure-registry=10.0.2.23:5000” –parametrillä, jolla prosessille kerrottiin privaatin rekisterin sijainti. Rekisteri sijaitsi Consul-virtuaalikoneella, joten mukautettu kontti tehtiin myös kyseisellä koneella. Mukautetun salauskontin luominen aloitettiin lataamalla GitHubista philplckthun/strongswan-repositorio seuraavasti:

```
root@consul:~# git clone https://github.com/philpl/docker-
strongswan.git

root@consul:~# cd docker-strongswan/
```

Repositoriossa olevaan Dockerfile-tiedostoon tehtiin muutoksia, jotka loivat kontin sisälle /etc/ipsec/confugurations-hakemiston, asensi strongSwanin kyseiseen hakemistoon ja lisäsi repositoriossa olevat ipsec.conf- ja strongswan.conf-tiedoston kontin sisälle luotuun uuteen hakemistoon. Kyseiset muutokset ovat kuvioissa 27, 28 ja 29. Koko Dockerfile-tiedosto on esitetty liitteessä 1.

```
root@consul:~/containers/docker-strongswan# cat Dockerfile
FROM buildpack-deps:jessie

RUN mkdir -p /conf
RUN mkdir -p /etc/ipsec/configurations
```

Kuvio 27. Uuden kansion hakemiston Dockerfile-tiedostossa

```
RUN mkdir -p /usr/src/strongswan \
        && curl -SL "https://download.strongswan.org/strongswan-$STRONGSWAN_VERSION.tar.gz" \
        | tar -zxC /usr/src/strongswan --strip-components 1 \
        && cd /usr/src/strongswan \
        && ./configure --prefix=/usr --sysconfdir=/etc/ipsec/configurations \
```

Kuvio 28. Asennuskansion määritteleminen

```
# Strongswan Configuration
ADD ipsec.conf /etc/ipsec/configurations/ipsec.conf
ADD strongswan.conf /etc/ipsec/configurations/strongswan.conf
```

Kuvio 29. Tiedostojen lisäys uuteen hakemistoon

Seuraavaksi rakennettiin muokatusta dockerfile-tiedostosta levykuva komennolla:

```
root@consul:~/containers/docker-strongswan/# docker build -t
ipsec/container:latest .
```

Parametrilla "*-t*" annettiin levykuvalle nimeksi ipsec/container:latest ja pisteellä määriteltiin Dockerfilen olevan kyseisessä kansiossa. Tämän jälkeen luotu levykuva laitettiin vielä rekisteriin seuraavasti:

```
docker tag ipsec/container 10.0.2.23:5000/ipsec/container

docker push 10.0.2.23:5000/ipsec/container
```

"*docker tag*" -komennolla äsken luodun kontin nimeksi vaihdettiin 10.0.2.23:5000/ipsec/container, jossa on viittaus privaattiin rekisteriin ja "*docker push*" -komennolla uudesti nimetty levykuva laitettiin rekisterikontin sisälle. Kuviossa 30 on esitetty Consul-virtuaalikoneen levykuvat, jossa näkyy DockerHubista ladatut registry- ja consul-levykuva sekä juuri luodut mukautetut levykuvat ipsec/container ja 10.0.2.23:5000/ipsec/container.

```
root@consul:~/containers/docker-strongswan# docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
10.0.2.23:5000/ipsec/container      latest              c82aa1cd6279        7 hours ago         672.6 MB
ipsec/container                     latest              c82aa1cd6279        7 hours ago         672.6 MB
buildpack-deps                      jessie              0aa8b81b52ed        7 days ago          606.6 MB
registry                            2                   83139345d017        4 weeks ago         165.8 MB
progrium/consul                     latest              09ea64205e55        9 months ago        69.43 MB
root@consul:~/containers/docker-strongswan#
```

Kuvio 30. Consul-koneen levykuvat

6.2 Salauskontin liittäminen kytkimiin

Ensimmäiseksi SDN-verkon välitystasolle luotiin Docker-verkko, jossa toteutus suoritettaisiin, se tapahtui komennolla:

```
root@OVS1:~#docker network create -d openvswitch --
subnet=10.0.0.0/24 sdn_network
```

Komento luo Docker-verkon käyttäen openvswitch-ajuria konttien liittämiseen OVS-kytkimiin. Aliverkoksi määriteltiin 10.0.0.0/24 ja verkko annettiin nimi ”sdn_network”.

Seuraavaksi salauskontti nostettiin OVS1-virtuaalikoneelle komennolla:

```
root@OVS1:~# docker run -t -i -v /root/cipher-
OVS1/:/etc/ipsec/configurations --privileged --name=cipher-OVS1
--mac-address=00:00:00:00:01:02 --ip=10.0.0.102 --
net=sdn_network 10.0.2.23:5000/ipsec/container /bin/bash
```

Komento käynnisti kontin kiinnittämällä isäntäkoneen /root/cipher-OVS1/-hakemiston kontin sisällä olevaan /etc/ipsec/configurations/-hakemistoon, jolloin kontin sisällä tarvittavia konfiguraatiotiedostoja voitiin hallita kontin ulkopuolelta. Kontti nimettiin cipher-OVS1 –nimiseksi ja sille annettiin staattinen MAC- ja IP-osoite. Se liitettiin myös aikaisemmin tehtyyn ”sdn_network”-verkkoon ja levykuvana käytettiin privaatissa rekisterissä olevaa 10.0.2.23:5000/ipsec/container-levykuvaa. Parametrit ”-t” ja ”-i” sekä ”bin/bash” mahdollistivat kontin sisälle menemisen ja sieltä komentojen suorittamisen. Samalle virtuaalikoneelle luotiin myös asiakaslaitetta simuloiva Docker-kontti komennolla:

```
root@OVS1:~# docker run -t -i --name=host1 --ip=10.0.0.2 --
net=sdn_network ubuntu:latest /bin/bash
```

Komento loi host1 nimisen kontin samaan “sdn_network”-verkkoon. Tällä kertaa levykuvana käytettiin DockerHubissa olevaa ubuntu-käyttöjärjestelmää. Muille kytkin-virtuaalikoneille luotiin myös salaus- ja asiakaslaitetta simuloiva kontti vastaten taulukon 2 osoitteistusta. host1-4 –konttien MAC-osoitteilla ei ollut toteutuksen kannalta merkitystä, joten kiinteille osoitteille ei ollut tarvetta. Sen sijaan salauskonteille annettiin kiinteät MAC-osoitteet manuaalisen liikenteenohjauksen toteuttamisen helpottamiseksi.

Taulukko 2. Konttien MAC- ja IP-osoitteet

Nimi	Sijainti	MAC	IP
Cipher-OVS1	OVS1	00:00:00:00:01:02	10.0.0.102
Cipher-OVS2	OVS2	00:00:00:00:01:03	10.0.0.103
Cipher-OVS3	OVS3	00:00:00:00:01:04	10.0.0.104
Cipher-OVS4	OVS4	00:00:00:00:01:05	10.0.0.106
host1	OVS1	-	10.0.0.2
host2	OVS2	-	10.0.0.3
host3	OVS3	-	10.0.0.4
host4	OVS4	-	10.0.0.5

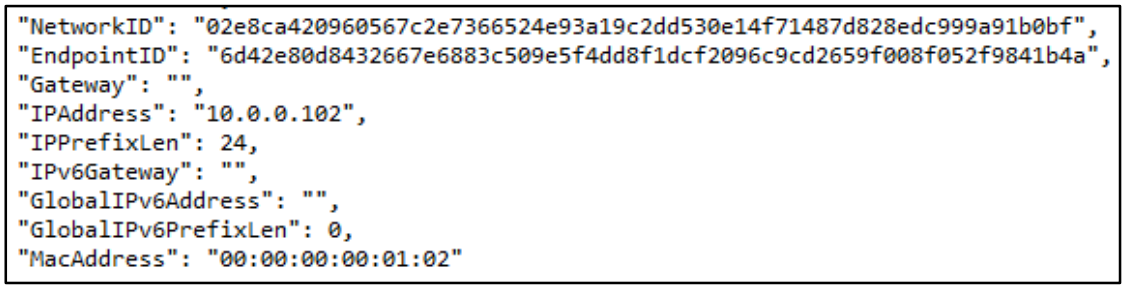
OVS:n docker-ajuri loi automaattisesti kontille rajapinnan ja liitti sen OVS-kytkimeen. Kuviossa 31 suoritettiin *”ovs-vsctl show”*-komento OVS1-virtuaalikoneella, jolla todennettiin konttien liittyminen kytkinverkkoon.

```
root@OVS1:~/cipher-OVS1# ovs-vsctl show
45437884-1a6d-4033-b53b-c0ec910f8113
    Bridge br-int
        Controller "tcp:10.0.2.24:6633"
            is_connected: true
        fail_mode: secure
        Port "geneve1"
            Interface "geneve1"
                type: geneve
                options: {remote_ip="14.0.0.2"}
        Port "eth3"
            Interface "eth3"
        Port br-int
            Interface br-int
                type: internal
        Port "6d42e80d8432667"
            Interface "6d42e80d8432667"
        Port "94014c82802bb31"
            Interface "94014c82802bb31"
        Port "geneve0"
            Interface "geneve0"
                type: geneve
                options: {remote_ip="12.0.0.2"}
    ovs_version: "2.5.1"
root@OVS1:~/cipher-OVS1#
```

Kuvio 31. Docker-konttien rajapinnat OVS1-kytkimellä

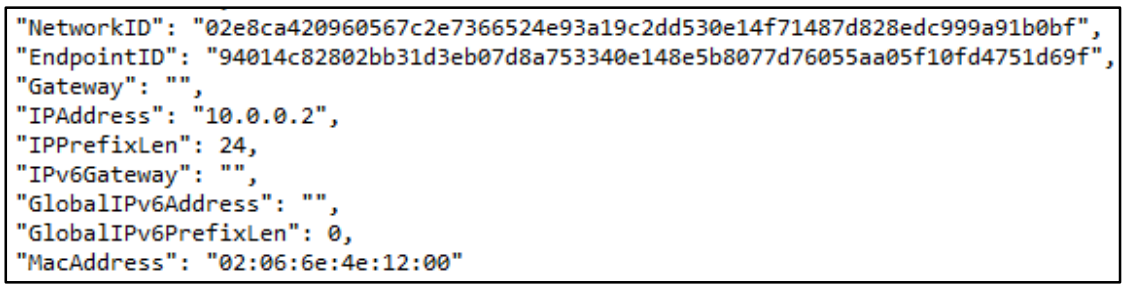
Komennolla *”docker inspect *kontin nimi*”* –komennolla haettiin Cipher-OVS1 ja host1 konteista hyödyllistä informaatiota. Kuvioissa 32 ja 33 on osa tulosteesta, josta

nähtiin ”EndpointID”-kentän 15 ensimmäisen merkin vastaavan OVS-kytkimessä olevan rajapinnan tunnusta.



```
"NetworkID": "02e8ca420960567c2e7366524e93a19c2dd530e14f71487d828edc999a91b0bf",
"EndpointID": "6d42e80d8432667e6883c509e5f4dd8f1dcf2096c9cd2659f008f052f9841b4a",
"Gateway": "",
"IPAddress": "10.0.0.102",
"IPPrefixLen": 24,
"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"MacAddress": "00:00:00:00:01:02"
```

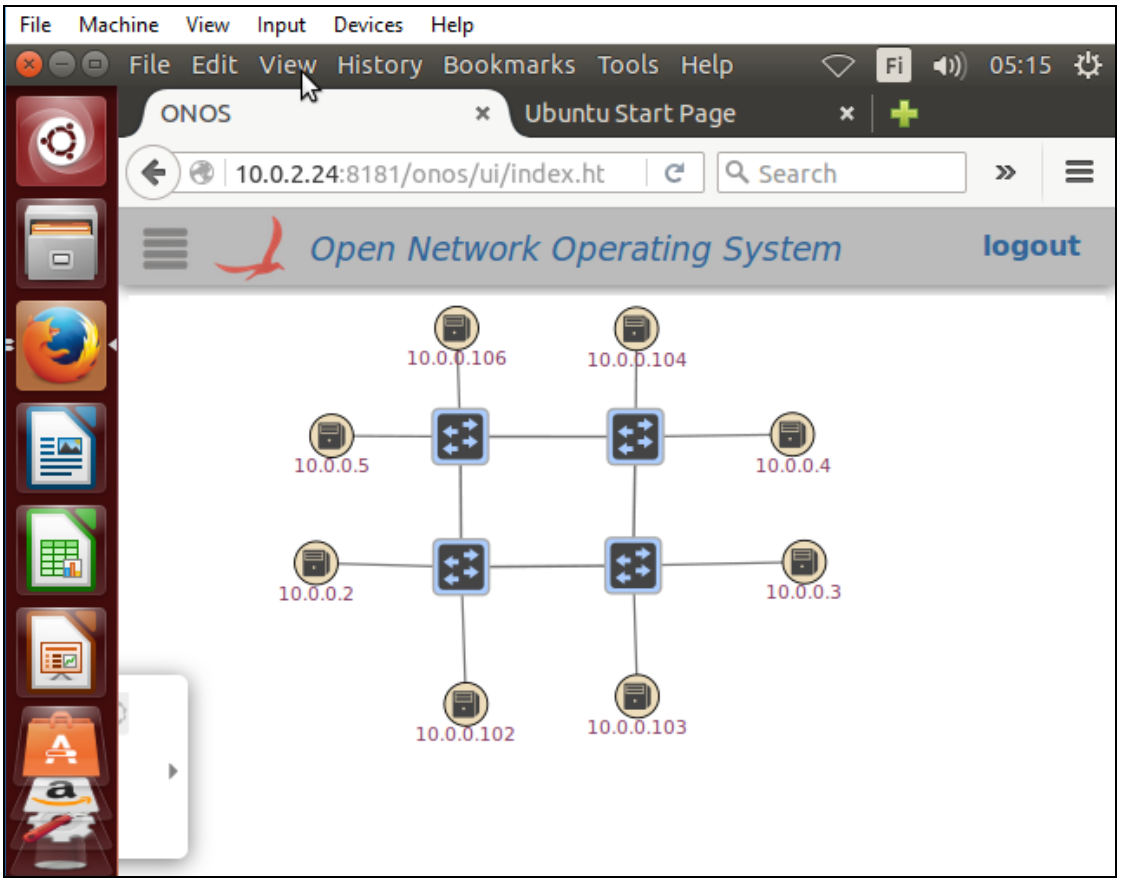

Kuvio 32. Cipher-OVS1-kontin verkkoasetukset



```
"NetworkID": "02e8ca420960567c2e7366524e93a19c2dd530e14f71487d828edc999a91b0bf",
"EndpointID": "94014c82802bb31d3eb07d8a753340e148e5b8077d76055aa05f10fd4751d69f",
"Gateway": "",
"IPAddress": "10.0.0.2",
"IPPrefixLen": 24,
"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"MacAddress": "02:06:6e:4e:12:00"
```


Kuvio 33. Host1-kontin verkkoasetukset

Kun kaikki kontin oltiin luotu jokaiselle OVS-virtuaalikoneelle, ONOSin käyttöliittymästä nähtiin konttien liittyminen verkkoon.



6.3 Salauksen konfigurointi asiakkaiden välille

Salauskontin sisällä oleva strongSwan tarvitsi kolme konfiguraatiotiedostoa toimiakseen. Konfiguraatiotiedostot luotiin kontin isäntäkoneelle kiinnitettyyn kansioon, jolloin tiedosto synkronoituivat myös kontin sisälle. Ipsec.conf-tiedostoon määritettiin oletusparametrit, joita käytettiin jokaisen yhteyden luomiseen, sekä yhteyksille yksilölliset parametrit osoitteiden osalta. Ipsec.secrets-tiedostossa määriteltiin esijaetut avaimet, joita salauskontti käytti suojaussidosten muodostamiseen. Strongswan.conf-tiedostossa määriteltiin strongSwan-salausohjelmiston käyttämät mahdolliset salaus- ja autentikointiprotokollat. Kuviossa 34 on näytetty kiinnitetyn kansion sisältö ja kuviossa 35 on esitetty strongswan.conf sisältö, joka oli sama identtinen kaikille salauskonteille. Esimerkeistä nähdään tarvittavat konfiguraatiot host1 ja host2 välisen tunnelin muodostamiselle.

```
root@OVS1:~/cipher-OVS1# ll
total 20
drwxr-xr-x 2 root root 4096 huhti 11 14:02 ./
drwx------ 6 root root 4096 huhti 14 14:48 ../
-rw-r--r-- 1 root root  312 huhti 14 10:50 ipsec.conf
-rw-r--r-- 1 root root   33 huhti 14 10:33 ipsec.secrets
-rw-r--r-- 1 root root  195 huhti 11 13:12 strongswan.conf
root@OVS1:~/cipher-OVS1#
```

Kuvio 34. Salauskonttiin kiinnitetyn hakemiston sisältö

```
root@OVS1:~/cipher-OVS1# cat strongswan.conf
# /etc/strongswan.conf - strongSwan configuration file

charon {
  load = aes des sha1 sha2 md5 gmp random nonce hmac stroke kernel-netlink socket-default updown
  multiple_authentication = no
}
root@OVS1:~/cipher-OVS1#
```

Kuvio 35. strongSwanin konfiguraatiotiedosto

Ipsec.secrets-tiedostoissa säilöttiin konteille esijaetut avaimet, joita käytettiin suojaussidosten muodostamisessa. Tiedostossa määriteltiin vastapuolen IP-osoitteelle esijaettu avain, jonka tuli olla sama konttien eli tunnelin yhdyskäytävien molemmille päille.

```
root@OVS1:~/cipher-OVS1# cat ipsec.secrets
10.0.0.103 : PSK   "secret_psk"

root@OVS1:~/cipher-OVS1#
```

Kuvio 36. Cipher-OVS1-kontin esijaetut avaimet

```
root@OVS2:~/cipher-OVS2# cat ipsec.secrets

10.0.0.102 : PSK   "secret_psk"
root@OVS2:~/cipher-OVS2#
```

Kuvio 37. Cipher-OVS2 -kontin esijaetut avaimet

Ipsec.conf-tiedoston oletusasetuksissa määriteltiin IKE-avaimen eliniäksi 60 minuuttia ja sisarsuojaussidokselle päätelaitteiden välille avaimen eliniäksi 20 minuuttia. Avaintenvaihdon maksimikestoksi 3 minuuttia ja enimmillään kolme yritystä. Autentikoinniksi määriteltiin esijaettu avain ja avaintenvaihtoprotokollaksi IKEv2. MOBIKE-

protokollan käyttö estettiin, koska kaikki verkossa olevat IP-osoitteet olivat staattisia eivätkä ne olleet NAT:n takana. Oletusasetusten lisäksi luotiin host1- ja host2-kontin välinen salattu yhteys. Kentät ”left” ja ”right” määrittelivät IPsec-yhdyskäytävien IP-osoitteet sekä ”leftsubnet” ja ”rightsubnet” määrittelivät päätelaitteiden IP-osoitteet. ”Left-” ja ”leftsubnet”-kenttä viittaavat aina paikalliseen IP-osoitteeseen, kun taas ”right” ja ”rightsubnet” viittaavat etänä olevaan kohteeseen. Vastapuolelle peilattuna IP-osoitteet ainoastaan vaihtoivat paikkaa kuten kuvioissa 38 ja 39 on tehty.

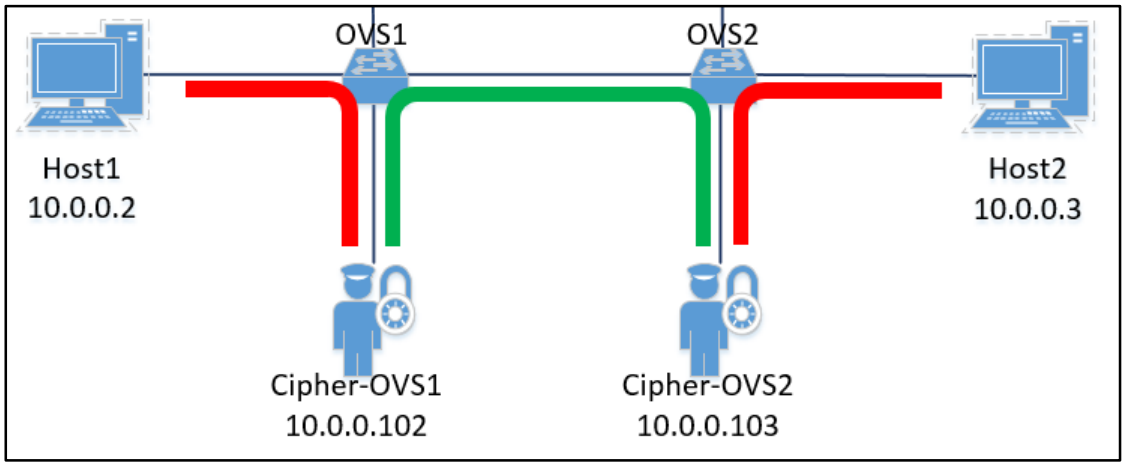
```
root@OVS1:~/cipher-OVS1# cat ipsec.conf
config setup

conn %default
        ikelifetime=60m
        keylife=20m
        rekeymargin=3m
        keyingtries=3
        authby=psk
        keyexchange=ikev2
        mobike=no

conn host1-host2
        left=10.0.0.102
        leftsubnet=10.0.0.2
        right=10.0.0.103
        rightsubnet=10.0.0.3
        auto=start

root@OVS1:~/cipher-OVS1#
```

Kuvio 38. Cipher-OVS1 -kontin IPsec-konfiguraatiot

```
root@OVS2:~/cipher-OVS2# cat ipsec.conf
config setup

conn %default
        ikelifetime=60m
        keylife=20m
        rekeymargin=3m
        keyingtries=3
        authby=psk
        keyexchange=ikev2
        mobike=no

conn host2-host1
        left=10.0.0.103
        leftsubnet=10.0.0.3
        right=10.0.0.102
        rightsubnet=10.0.0.2
        auto=start

root@OVS2:~/cipher-OVS2#
```

Kuvio 39. Cipher-OVS2 -kontin IPsec-konfiguraatiot

Kuviossa 40 on graafisesti esitetty salauskonttien välille konfiguroitu IPsec-tunneli. Punaisella värillä kuvastetaan selkokielistä liikennettä ja kytkinten välissä olevalla vihreällä viivalla kuvastetaan salattua IPsec-tunnelia.

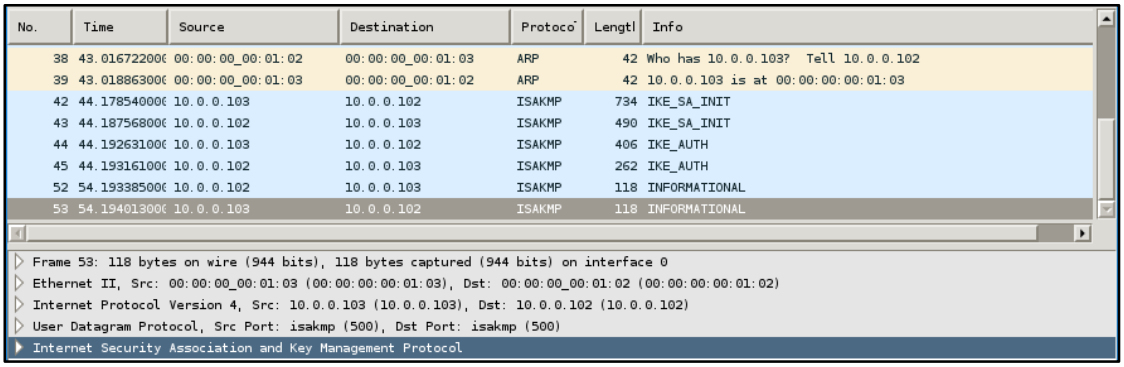


Kuvio 40. Graafinen esitys konfiguroidusta IPsec-tunnelista

strongSwan-ohjelmisto käynnistettiin kontin sisällä suorittamalla komento:

```
root@OVS1:~# docker exec cipher-OVS1 ipsec start
```

Vastaavasti cipher-OVS2 –kontti käynnistettiin OVS2-virtuaalikoneella, jonka jälkeen kontit neuvottelivat suojaussidoksen välilleen. Kuviossa 41 on esitetty wireshark-ohjelmistolla kaapattua liikennettä OVS1-kytkimen rajapinnasta, johon Cipher-OVS1 –kontti aikaisemmin liitettiin.



Kuvio 41. Suojaussidosten neuvottelu konttien välillä

Konttien välistä suojaussidosta voidaan tarkastella suorittamalla komento:

```
root@OVS1:~# docker exec cipher-OVS1 ipsec statusall
```

Kuviosta 42 nähdään IPsec-tunnelin suojaussidos yhdyskäytävien IP-osoitteille 10.0.0.102 ja 10.0.0.103 sekä sisarsuojaussidos määritellyille päätelaitteille 10.0.0.2 ja 10.0.0.3. Tunnelin salaukseen käytettiin AES_CBC_128-algoritmiä ja datan autentikointiin käytettiin HMAC_SHA1-96-tiivistefunktiota.

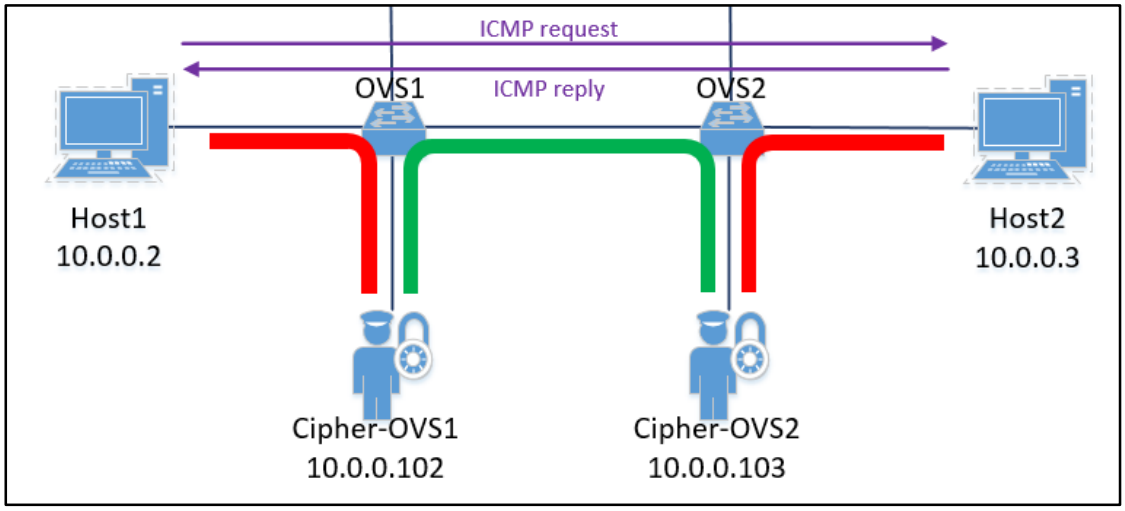
```
root@OVS1:~/cipher-OVS1# docker exec cipher-OVS1 ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.5, Linux 3.19.0-25-generic, x86_64):
  uptime: 19 seconds, since Apr 15 12:12:14 2016
  malloc: sbrk 1351680, mmap 0, used 178832, free 1172848
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 6
  loaded plugins: charon aes des sha1 sha2 md5 gmp random nonce hmac stroke kernel-netlink socket-default updown
Listening IP addresses:
  10.0.0.102
  172.18.0.3
Connections:
 host1-host2:  10.0.0.102...10.0.0.103  IKEv2
 host1-host2:   local:  [10.0.0.102] uses pre-shared key authentication
 host1-host2:   remote: [10.0.0.103] uses pre-shared key authentication
 host1-host2:   child:  10.0.0.2/32 === 10.0.0.3/32 TUNNEL
Security Associations (1 up, 0 connecting):
 host1-host2[2]: ESTABLISHED 17 seconds ago, 10.0.0.102[10.0.0.102]...10.0.0.103[10.0.0.103]
 host1-host2[2]: IKEv2 SPIs: 2f21e7699c5c0d0e_i 347395225a947969_r*, pre-shared key reauthentication in 56 minutes
 host1-host2[2]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
 host1-host2{2}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c3b6980f_i c3142bc5_o
 host1-host2{2}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 15 minutes
 host1-host2{2}:   10.0.0.2/32 === 10.0.0.3/32
root@OVS1:~/cipher-OVS1#
```

Kuvio 42. Muodostunut suojaussidos

Tässä vaiheessa host1 ja host2 välinen liikenne ei kuitenkaan mennyt vielä IPsec-tunnelin kautta, sillä ONOSin L2-tasolla liikennettä ohjaava org.onosproject.fwd-applikaatio käyttää lähde- ja kohde-MAC-osoitteita liikenteen ohjaamiseen. Kuviosta 43 nähdään kuinka host1 ja host2 välisen ping-viestin TTL-arvo on 64, jolloin niiden välillä ei ole yhtäkään hyppyä. Kuviossa 44 on havainnollistettu kuinka liikenne käyttäytyy verkossa.

```
root@OVS1:~/cipher-OVS1# docker attach host1

root@3bb686332e04:/# ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=4.88 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.773 ms
^C
--- 10.0.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.773/2.827/4.882/2.055 ms
root@3bb686332e04:/#
```

Kuvio 43. ICMP-ping host1 ja host2 välillä



Kuvio 44. Havainnollistettu esitys liikenteen kulusta

Jotta päätelaitteiden välinen liikenne saatiin IPsec-tunneliin, tuli ONOSilla ohjata liikenne salauskomponenteille. ONOSilla luotiin intentit, jotka ohjasivat sisääntuloportin ja kohde-IP-osoitteen perusteella liikenteen salaajalle. Intenttien luonnissa tarvitaan kytkimen dpid-arvoa ja porttien ID:tä vastaavaa numerointia. Porttinumeroinnin ja dpid voidaan tarkistaa komennolla *”ovs-ofctl show br-int”* (kts. kuvio 45).

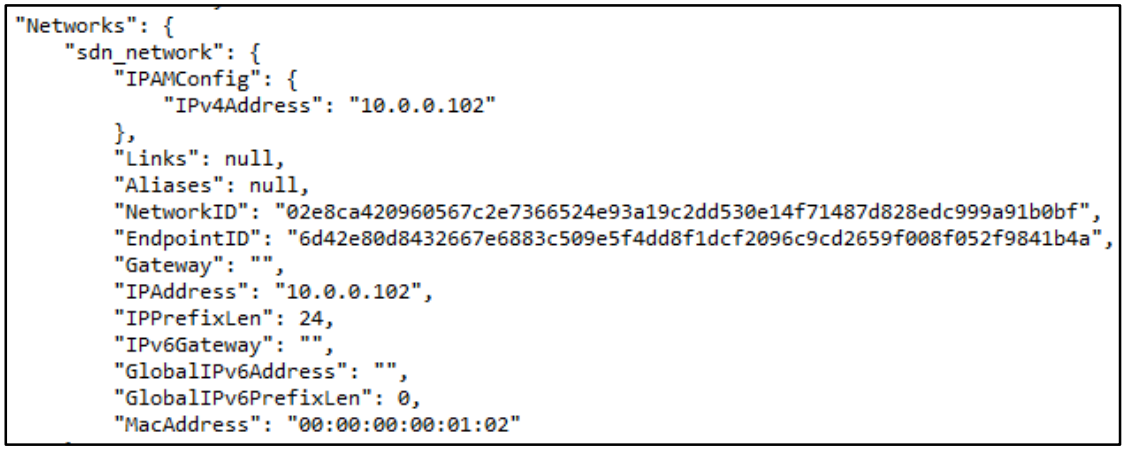
```
root@OVS1:~/cipher-OVS1# ovs-ofctl show br-int
OFPT_FEATURES_REPLY (xid=0x2): dpid:000008002792670f
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst
 1(geneve1): addr:6a:ae:cf:80:8b:49
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(eth3): addr:08:00:27:92:67:0f
     config:     0
     state:      0
     current:    1GB-FD COPPER AUTO_NEG
     advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
     supported:  10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
     speed: 1000 Mbps now, 1000 Mbps max
 3(94014c82802bb31): addr:ee:4d:15:f0:91:90
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 4(6d42e80d8432667): addr:c2:ad:f5:ae:84:4c
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 8(geneve0): addr:8e:bb:67:04:84:62
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(br-int): addr:08:00:27:92:67:0f
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
root@OVS1:~/cipher-OVS1#
```

Kuvio 45. OVS1-kytkimen porttinumerointi ja dpid-arvo

Kuvioissa 46 ja 47 suoritettiin *”docker inspect”* –komennot host1- ja Cipher-OVS1-konteille joista vertaamalla ”EndpointID”-arvoja kuvion 45 tulosteeseen saatiin selville, että host1 oli kiinni portissa 3 ja cipher-OVS1 portissa 4.


```
"Networks": {
    "sdn_network": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "NetworkID": "02e8ca420960567c2e7366524e93a19c2dd530e14f71487d828edc999a91b0bf",
        "EndpointID": "94014c82802bb31d3eb07d8a753340e148e5b8077d76055aa05f10fd4751d69f",
        "Gateway": "",
        "IPAddress": "10.0.0.2",
        "IPPrefixLen": 24,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:06:6e:4e:12:00"
```


Kuvio 46. Docker inspect host1



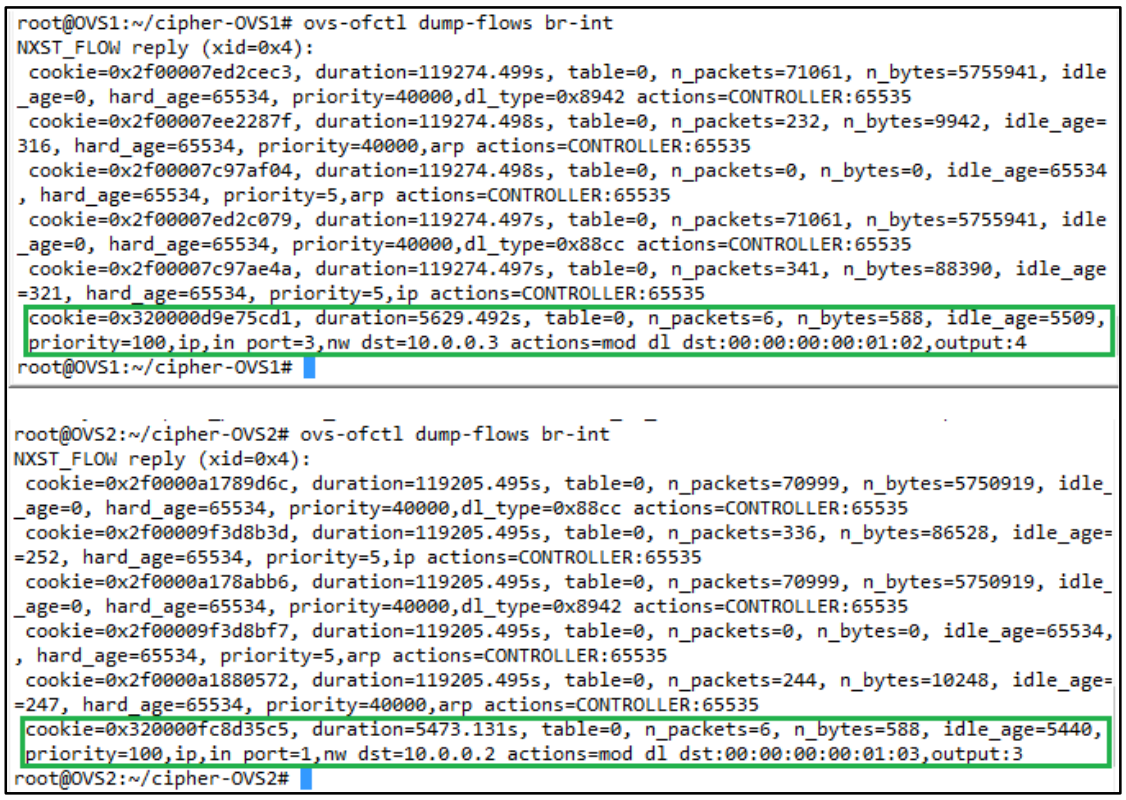
```
"Networks": {
    "sdn_network": {
        "IPAMConfig": {
            "IPv4Address": "10.0.0.102"
        },
        "Links": null,
        "Aliases": null,
        "NetworkID": "02e8ca420960567c2e7366524e93a19c2dd530e14f71487d828edc999a91b0bf",
        "EndpointID": "6d42e80d8432667e6883c509e5f4dd8f1dcf2096c9cd2659f008f052f9841b4a",
        "Gateway": "",
        "IPAddress": "10.0.0.102",
        "IPPrefixLen": 24,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "00:00:00:00:01:02"
```


Kuvio 47. Docker inspect cipher-OVS1

ONOSin komentolinjalta luotiin intentit seuraavasti OVS1- ja OVS2-kytkimille seuraavasti:

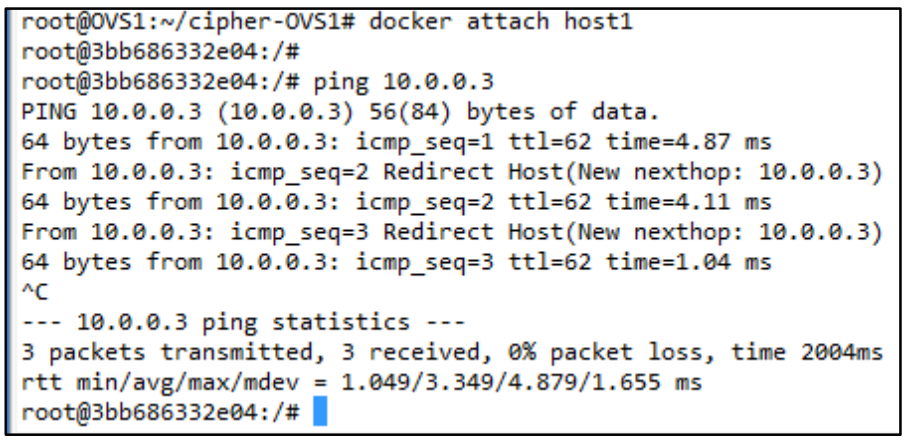
```
onos> add-point-intent -t IPV4 --ipDst 10.0.0.3/32 --setEthDst
00:00:00:00:01:02 of:000008002792670f/3 of:000008002792670f/4

onos> add-point-intent -t IPV4 --ipDst 10.0.0.2/32 --setEthDst
00:00:00:00:01:03 of:0000080027901425/1 of:0000080027901425/3
```

Luotuja intenttejä voidaan tarkastella kytkimiltä komennolla *”ovs-ofctl dump-flows br-int”*, josta nähdään, että kytkimille on ilmestynyt intenttejä vastaavat vuot vuotauluihin (kts. kuvio 48).



Kuvio 48. Intenttien luomat vuot

Kuviossa 49 host1 pingaa host2 ja ICMP-reply-viesti tulee takaisin host1:lle TTL-arvolla 62, eli ICMP-paketti kulkee kahden IPsec-yhdyskäytävän läpi. Tämä voidaan todeta myös kuviosta 50, josta havaitaan, että sisarsuojaussidokseen osuu paketteja.



Kuvio 49. IPsec-tunneliin menevä ping

```
root@OVS1:~/cipher-OVS1# docker exec cipher-OVS1 ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.5, Linux 3.19.0-25-generic, x86_64):
  uptime: 3 hours, since Apr 15 12:12:13 2016
  malloc: sbrk 1347584, mmap 0, used 177984, free 1169600
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 2
  loaded plugins: charon aes des sha1 sha2 md5 gmp random nonce hmac stroke kernel-netlink sock
et-default updown
Listening IP addresses:
  10.0.0.102
  172.18.0.3
Connections:
 host1-host2:  10.0.0.102...10.0.0.103  IKEv2
 host1-host2:   local:  [10.0.0.102] uses pre-shared key authentication
 host1-host2:   remote: [10.0.0.103] uses pre-shared key authentication
 host1-host2:   child:  10.0.0.2/32 === 10.0.0.3/32 TUNNEL
Security Associations (1 up, 0 connecting):
 host1-host2[6]: ESTABLISHED 13 minutes ago, 10.0.0.102[10.0.0.102]...10.0.0.103[10.0.0.103]
 host1-host2[6]: IKEv2 SPIs: 234eda91bee6cc1f_i f18c4a34dc8f238a_r*, pre-shared key reauthentic
ation in 42 minutes
 host1-host2[6]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
 host1-host2{18}:  INSTALLED, TUNNEL, reqid 5, ESP SPIs: c8aea8ec_i cf917820_o
 host1-host2{18}:  AES_CBC_128/HMAC_SHA1_96, 252 bytes_i (3 pkts, 41s ago), 252 bytes_o (3 pkts
, 41s ago), rekeying in 14 seconds
 host1-host2{18}:   10.0.0.2/32 === 10.0.0.3/32
root@OVS1:~/cipher-OVS1#
```

Kuvio 50. Muutokset suojaussidoksen tulosteessa

Kuviossa 51 on wireshark-kaappaus Cipher-OVS1-kontin rajapinnasta, josta voidaan vielä varmentaa, että suojaussidosten neuvottelun jälkeen ICMP-request kapseloidaan ESP-otsikolla, jolloin ICMP-request-viestin alkuperäinen MAC- ja IP-osoite salataan. ESP-otsikon MAC- ja IP-osoite vastaavat salauskonteille annettuja osoitteita.

No.	Time	Source	Destination	Protocol	Length	Info
8337	12690.293039	10.0.0.103	10.0.0.102	ISAKMP	734	IKE_SA_INIT
8338	12690.302533	10.0.0.102	10.0.0.103	ISAKMP	490	IKE_SA_INIT
8339	12690.307847	10.0.0.103	10.0.0.102	ISAKMP	406	IKE_AUTH
8340	12690.308403	10.0.0.102	10.0.0.103	ISAKMP	262	IKE_AUTH
8345	12695.288622	00:00:00_00:01:02	00:00:00_00:01:03	ARP	42	Who has 10.0.0.103? Tell 10.0.0.102
8346	12695.289555	00:00:00_00:01:03	00:00:00_00:01:02	ARP	42	10.0.0.103 is at 00:00:00:00:01:03
8857	13485.041374	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x001b, seq=1/256, ttl=64 (reply in 8860)
8858	13485.041421	10.0.0.102	10.0.0.103	ESP	166	ESP (SPI=0xcf917820)
8859	13485.045216	10.0.0.103	10.0.0.102	ESP	166	ESP (SPI=0xc8aea8ec)
8860	13485.045255	10.0.0.3	10.0.0.2	ICMP	98	Echo (ping) reply id=0x001b, seq=1/256, ttl=62 (request in 885
8861	13486.044146	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x001b, seq=2/512, ttl=64 (reply in 8865)
8862	13486.044196	10.0.0.102	10.0.0.2	ICMP	126	Redirect (Redirect for host)
8863	13486.044288	10.0.0.102	10.0.0.103	ESP	166	ESP (SPI=0xcf917820)
8864	13486.048093	10.0.0.103	10.0.0.102	ESP	166	ESP (SPI=0xc8aea8ec)
8865	13486.048193	10.0.0.3	10.0.0.2	ICMP	98	Echo (ping) reply id=0x001b, seq=2/512, ttl=62 (request in 886
8866	13487.045283	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x001b, seq=3/768, ttl=64 (reply in 8870)
8867	13487.045323	10.0.0.102	10.0.0.2	ICMP	126	Redirect (Redirect for host)
8868	13487.045417	10.0.0.102	10.0.0.103	ESP	166	ESP (SPI=0xcf917820)
8869	13487.046153	10.0.0.103	10.0.0.102	ESP	166	ESP (SPI=0xc8aea8ec)
8870	13487.046271	10.0.0.3	10.0.0.2	ICMP	98	Echo (ping) reply id=0x001b, seq=3/768, ttl=62 (request in 886

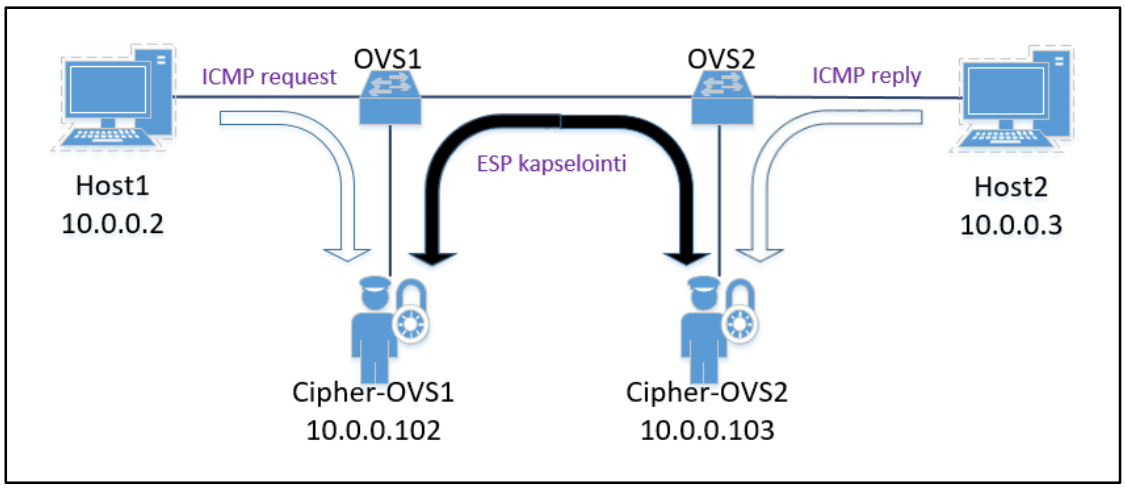
```
▶ Frame 8858: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits) on interface 0
▷ Ethernet II, Src: 00:00:00_00:01:02 (00:00:00:00:01:02), Dst: 00:00:00_00:01:03 (00:00:00:00:01:03)
▷ Internet Protocol Version 4, Src: 10.0.0.102 (10.0.0.102), Dst: 10.0.0.103 (10.0.0.103)
▷ Encapsulating Security Payload
```

Kuvio 51. Salatun liikenteen kaappaus

Liikenteen kulku tapahtuu vaihe kerrallaan seuraavasti:

1. Host1 lähettää ICMP-request-viestin Host2:lle
2. ICMP-paketti osuu OVS1:llä olevaan vuohon, jolloin kohde-MAC-osoite kirjoitetaan uudelleen vastaamaan cipher-OVS1:n MAC-osoitetta ja paketti laitetaan cipher-OVS1:n portista ulos
3. org.onosproject.fwd-applikaatio ohjaa ESP-kapseloidun paketin cipher-OVS1:ltä cipher-OVS2:lle
4. cipher-OVS2 purkaa kapseloinnin ja laittaa alkuperäisen ICMP-request-viestin host2:lle
5. Host2 vastaanottaa ICMP-request-viestin lähettää ICMP-reply-viestin host1:lle
6. ICMP-reply osuu OVS2:lla olevaan vuohon, jolloin kohde-MAC-osoite kirjoitetaan uudelleen vastaamaan cipher-OVS2:n MAC-osoitetta ja paketti laitetaan cipher-OVS2:n portista ulos
7. org.onosproject.fwd-applikaatio ohjaa ESP-kapseloidun paketin cipher-OVS2:ltä cipher-OVS1:lle
8. cipher-OVS1 purkaa kapseloinnin ja laittaa alkuperäisen ICMP-reply-viestin host1:lle

Kuviossa 52 on vielä havainnollistettu liikenteen kulun vaiheet graafisesti.



Kuvio 52. Graafinen esitys salatusta liikenteestä

7 Verkon automatisointi

7.1 Konfiguraatiohallinta

Useiden kytkinten muodostamassa verkkoympäristössä IPsec-tunneleiden konfiguraatiohallinta on erittäin työlästä ja tekstipohjaisten konfiguraatioiden muutoksissa tulee helposti virheitä. Automatisoitu konfiguraatiohallintaprosessi toteutettiin Ansiblella. Toteutuksessa Ansiblea käytettiin Consul-virtuaalikoneelta, mutta Ansiblea

voitaisiin käyttää millä tahansa verkon järjestelmällä. Ansible tarvitsee toimiakseen salasanattoman SSH-yhteyden etäjärjestelmiin, joten Consul-virtuaalikoneelle generoitiin julkinen avain:

```
root@consul:~# ssh-keygen –t rsa
```

Tämän jälkeen Consul-virtuaalikoneen julkinen avain kopioitiin etäjärjestelmien autorisoitujen avainten joukkoon:

```
root@consul:~# ssh-copy-id 10.0.2.21
```

Avaimen kopiointi edellyttää etäjärjestelmän käyttäjätunnuksen ja salasanan tietämistä, jonka jälkeen kirjautuminen SSH-yhteydellä ei vaadi salasanaa. Consulin julkinen avain kopioitiin kaikille OVS-virtuaalikoneille taulukon 1 ”MyNattedNetwork”-osoitteiden mukaisesti. Ansible asennettiin seuraavilla komennoilla:

```
root@consul:~# apt-get install software-properties-common
root@consul:~# apt-add-repository ppa:ansible/ansible
root@consul:~# apt-get update
root@consul:~# apt-get install ansible
```

Ansiblen konfigurointi tapahtuu /etc/ansible/-hakemistossa, jossa olevaan hosts-tiedostoon lisättiin OVS-virtuaalikoneet. Kuviossa 53 on esitetty hosts-tiedostoon tehty ”switches”-ryhmä, jonka alle määriteltiin etäjärjestelmien nimet ja IP-osoitteet. Tiedostoon määriteltiin myös paikallinen järjestelmä ”localhost”, sillä salauskonttien konfiguraatiot generoitiin ensin paikalliselle järjestelmälle, josta ne kopioitiin etäjärjestelmiin.

```
root@consul:/etc/ansible# cat hosts
# This is the default ansible 'hosts' file.

localhost ansible_connection=local

[switches]
OVS1 ansible_host=10.0.2.21
OVS2 ansible_host=10.0.2.22
OVS3 ansible_host=10.0.2.30
OVS4 ansible_host=10.0.2.31
```

Kuvio 53. Ansiblen hosts-tiedosto

Kuviossa 54 on esitetty Ansiblelle luotu hakemistorakenne. Toteutuksessa konfiguraatioita hallittiin ipsec_tunnel.yml-playbookilla, jossa paikalliselle järjestelmälle asetettiin rooli ”ipsec” kuvion 55 mukaisesti. Playbookin loppuosa kopioi konfiguraatiotiedostot etäjärjestelmiin.

```
root@consul:/etc/ansible# tree
.
├── ansible.cfg
├── group_vars
│   └── new_machines
├── hosts
├── ipsec_tunnel.yml
├── new_switch.yml
└── roles
    ├── docker
    │   └── tasks
    │       └── main.yml
    ├── ipsec
    │   ├── tasks
    │   │   └── main.yml
    │   ├── templates
    │   │   ├── salaaja.j2
    │   │   ├── secrets.j2
    │   │   └── strongswan.j2
    │   └── vars
    │       └── main.yml
    └── ovs
        └── tasks
            └── main.yml

10 directories, 12 files
```

Kuvio 54. Ansiblen hakemistorakenne

```
root@consul:/etc/ansible# cat ipsec_tunnel.yml
---
  - name: create configurations to local machine
    hosts: localhost

    roles:
      - ipsec

  - name: copy configurations from local machine to remote locations
    hosts: switches

    tasks:
      - copy: src=/root/configurations/cipher-{{ ansible_nodename }}/ipsec.conf dest=/ro
ot/cipher-{{ ansible_nodename }}/ipsec.conf
        register: result

      - copy: src=/root/configurations/cipher-{{ ansible_nodename }}/ipsec.secrets dest=
/root/cipher-{{ ansible_nodename }}/ipsec.secrets

      - copy: src=/root/configurations/cipher-{{ ansible_nodename }}/strongswan.conf des
t=/root/cipher-{{ ansible_nodename }}/strongswan.conf

      - command: docker exec cipher-{{ ansible_nodename }} ipsec restart
        when: result|changed
```

Kuvio 55. Konfiguraatiohallinnan playbook

/etc/ansible/roles/-hakemistoon luotiin ipsec/hakemisto, jonka sisälle luotiin vielä tasks/-, templates/-, ja vars/-hakemistot. Hakemistoon tasks/ luotiin main.yml-tiedosto, jossa varsinainen konfiguraatioiden generointi tapahtui. Kuviossa on esitetty tasks/-hakemistossa oleva main.yml-tiedosto. Tiedostoon määriteltiin kolme tehtävää, joista kukin käytti Ansiblen template-moduulia. Moduuli käyttää roolin sisällä olevia määriteltyjä jinja2-ohjelmointikielellä tehtyjä templaattiformaatteja ja YAML-kielellä tehtyjä muuttujatiedostoja konfiguraatioiden generoimiseen. Jokaisen strongSwanin käyttämän tiedoston luomiseen käytettiin omaa templaattia, jotka sijoitettiin templates/-hakemistoon. Kuviossa 56 on esitetty main.yml, jonka suorittaminen kopioi konfiguraatiot paikallisen järjestelmän /root/configurations/cipher-OVS1-4/-hakemistoihin (kts. kuvio 57).

```
root@consul:/etc/ansible# cat roles/ipsec/tasks/main.yml
---
- name: use template to create configuration file
  template: src=salaaja.j2 dest=/root/configurations/{{ item.switch_name }}/ipsec.conf
  with_items:
   '{{ switches }}'

- name: create psk file
  template: src=secrets.j2 dest=/root/configurations/{{ item.switch_name }}/ipsec.secrets
  with_items:
   '{{ switches }}'

- name: use template to create parameters for strongswan
  template: src=strongswan.j2 dest=/root/configurations/{{ item.switch_name }}/strongswan.conf
  with_items:
   '{{ switches }}'


root@consul:/etc/ansible#
```

Kuvio 56. roles/ipsec/tasks/main.yml-tiedosto

```
root@consul:~/configurations# tree
.
├── cipher-OVS1
│   ├── ipsec.conf
│   ├── ipsec.secrets
│   └── strongswan.conf
├── cipher-OVS2
│   ├── ipsec.conf
│   ├── ipsec.secrets
│   └── strongswan.conf
├── cipher-OVS3
│   ├── ipsec.conf
│   ├── ipsec.secrets
│   └── strongswan.conf
└── cipher-OVS4
    ├── ipsec.conf
    ├── ipsec.secrets
    └── strongswan.conf

4 directories, 12 files
root@consul:~/configurations#
```

Kuvio 57. Consulilla olevat konfiguraatiot

Kuviossa 58 on salaaja.j2-templaattitiedosto,jota käytettiin ipsec.conf-tiedoston luomiseen. Ulkoasultaan templaatti perustui kuvioiden 38 ja 39 muotoiluun, mutta yhteyksien muodostamiseen käytettiin kolmea FOR-silmukkaa ja vars/main.yml-muuttujatiedostoa.

```
root@consul:/etc/ansible# cat roles/ipsec/templates/salaaja.j2
#Config for {{ item.switch_name }}

config setup

conn %default
        ikelifetime=60m
        keylife=20m
        rekeymargin=3m
        keyingtries=3
        authby=psk
        keyexchange=ikev2
        mobike=no


{% for local in item.hosts %}
        {% for remote in local.connections %}

conn {{ local.name }}-{{ remote.remote_host }}
        left={{ local.local_container }}
        leftsubnet={{ local.local_ip }}
        right={{ remote.remote_container }}
        rightsubnet={{ remote.remote_ip }}
        auto=start

        {% endfor %}
{% endfor %}

root@consul:/etc/ansible#
```

Kuvio 58. roles/ipsc/templates/salaaja.j2-tiedosto

Kuviossa 59 on osa vars/main.yml-muuttujatiedostoa, jota vasten templaattia ajettiin. Templaatin ensimmäinen FOR-silmukka käy läpi tiedostossa olevat kytkimet ”switch_name”-kentän perusteella muodostaen jokaiselle salaajalle oman ipsec.conf-tiedoston. Toinen FOR-silmukka käy läpi kyseisessä kytkimessä olevat paikalliset asiakaslaitteet ja salauskontit käyttäen ”local_ip”- ja ”local_container”-arvoja muodostaakseen ”left”- ja ”leftsubnet”-kentät konfiguraatiotiedostoon, esimerkkinä OVS1:ssä kiinni oleva host1 ja cipher-OVS1. Viimeinen FOR-silmukka muodostaa IPsec-tunneleiden etänä olevat kohteet ”right” ja ”rightsubnet” käyttäen ”remote_ip”- ja ”remote_container”-arvoja. Yhteyden nimeämiseen käytetiin keskimmäisessä silmukassa olevaa ”name”-arvoa ja kolmannessa silmukassa olevaa ”remote_host”-arvoa. Toteutuksessa jokaiselle päätelaitteelle konfiguroitiin IPsec-tunneli kaikkiin muihin päätelaitteisiin eli full-mesh-topologian mukaisesti, kuten kuviossa 59 host1:lle on määritelty. Kokonainen vars/main.yml-tiedosto on liitteessä 2.

```
root@consul:/etc/ansible# cat roles/ipsec/vars/main.yml
---
switches:
    - switch_name: cipher-OVS1
      hosts:
        - name: host1
          local_ip: 10.0.0.2
          local_container: 10.0.0.102
          connections:
           - remote_host: host2
             remote_ip: 10.0.0.3
             remote_container: 10.0.0.103
           - remote_host: host3
             remote_ip: 10.0.0.4
             remote_container: 10.0.0.104
           - remote_host: host4
             remote_ip: 10.0.0.5
             remote_container: 10.0.0.106
```

Kuvio 59. osa roles/ipsec/vars/main.yml-tiedostoa

Esijaetut avaimet generoitiin käyttämällä secrets.2-templaattia (kts. kuvio 60) ja samaa vars/main.yml-tiedostoa. Jokaisella kontilla tuli olla esijaettu avain etänä oleviin kontteihin. Secrets.j2-templaatti kävi läpi vars-tiedostossa olevat ”remote_container”-arvot muodostaen kontille ipsec.secrets-tiedoston.

```
root@consul:/etc/ansible# cat roles/ipsec/templates/secrets.j2
#Pre shared keys for {{ item.switch_name }}
{% for local in item.hosts  %}
        {% for remote in local.connections %}

{{ remote.remote_container }} : PSK "secret_psk"
        {% endfor %}
{% endfor %}
```

Kuvio 60. roles/ipsec/templates/secrets.j2-tiedosto

strongswan.conf-tiedoston muodostamiseen käytetiin myös templaattia, mutta sen luomiseen ei tarvittu muuttujia, koska tiedosto on identtinen kaikille konteille. Kuvissa 61 on esitetty strongswan.j2-templaatti, josta generoitiin strongswan.conf-tiedosto.

```
root@consul:/etc/ansible# cat roles/ipsec/templates/strongswan.j2
# /etc/strongswan.conf - strongSwan configuration file

charon {
  load = aes des sha1 sha2 md5 gmp random nonce hmac stroke kernel-netlink socket-default updown
  multiple_authentication = no
}

root@consul:/etc/ansible#
```

Kuvio 61. roles/ipsec/templates/strongswan.j2-tiedosto

Ansiblella luotu ipsec-rooli (kts. kuvio 55) kopioi templaattien tekemät konfiguraatiotiedostot Consul-virtuaalikoneen paikalliseen hakemistoon, josta /etc/ansible/ipsec_tunnel.yml-playbook kopioi tiedostot etäjärjestelmien kontteihin kiinnitettyihin hakemistoihin. Playbookissa käytettiin ”ansible_nodename”-muuttujaa, jolla etäjärjestelmät eriteltiin toisistaan. Playbookin viimeisessä kentässä käytettiin ”when”-ehtoa, jolla salauskontti käynnistettiin uudelleen ainoastaan silloin, kun ipsec.conf-tiedostoon tehtiin muutoksia. Playbook suoritettiin komennolla:

```
root@consul:/etc/ansible# ansible-playbook ipsec_tunnel.yml
```

jolloin tiedostot generoitiin ja kopioitiin etäjärjestelmiin. Viimeisenä playbookin tehtävänä konttien sisällä olevat salausohjelmistot käynnistettiin uudelleen ja suojaussidokset muodostuivat konttien välille kuvion 62 mukaisesti.

```
root@OVS1:~/cipher-OVS1# docker exec cipher-OVS1 ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.5, Linux 3.19.0-25-generic, x86_64):
  uptime: 2 minutes, since Apr 16 14:08:04 2016
  malloc: sbrk 1351680, mmap 0, used 229776, free 1121904
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 13
  loaded plugins: charon aes des sha1 sha2 md5 gmp random nonce hmac stroke kernel-netlink socket-default updown
Listening IP addresses:
  10.0.0.102
  172.18.0.3
Connections:
 host1-host2:  10.0.0.102...10.0.0.103  IKEv2
 host1-host2:   local:  [10.0.0.102] uses pre-shared key authentication
 host1-host2:   remote: [10.0.0.103] uses pre-shared key authentication
 host1-host2:   child:  10.0.0.2/32 === 10.0.0.3/32 TUNNEL
 host1-host3:  10.0.0.102...10.0.0.104  IKEv2
 host1-host3:   local:  [10.0.0.102] uses pre-shared key authentication
 host1-host3:   remote: [10.0.0.104] uses pre-shared key authentication
 host1-host3:   child:  10.0.0.2/32 === 10.0.0.4/32 TUNNEL
 host1-host4:  10.0.0.102...10.0.0.106  IKEv2
 host1-host4:   local:  [10.0.0.102] uses pre-shared key authentication
 host1-host4:   remote: [10.0.0.106] uses pre-shared key authentication
 host1-host4:   child:  10.0.0.2/32 === 10.0.0.5/32 TUNNEL
Security Associations (3 up, 0 connecting):
 host1-host4[6]: ESTABLISHED 2 minutes ago, 10.0.0.102[10.0.0.102]...10.0.0.106[10.0.0.106]
 host1-host4[6]: IKEv2 SPIs: aa637ed824f7055f_i 992f90a98dd8409f_r*, pre-shared key reauthentication in 53 minutes
 host1-host4[6]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
 host1-host4{5}:  INSTALLED, TUNNEL, reqid 3, ESP SPIs: cff68750_i cdae1d7b_o
 host1-host4{5}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 12 minutes
 host1-host4{5}:   10.0.0.2/32 === 10.0.0.5/32
 host1-host3[5]: ESTABLISHED 2 minutes ago, 10.0.0.102[10.0.0.102]...10.0.0.104[10.0.0.104]
 host1-host3[5]: IKEv2 SPIs: 0b06b5122d710992_i 915bab6ece375b40_r*, pre-shared key reauthentication in 53 minutes
 host1-host3[5]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
 host1-host3{4}:  INSTALLED, TUNNEL, reqid 2, ESP SPIs: c299201e_i c071a27e_o
 host1-host3{4}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 11 minutes
 host1-host3{4}:   10.0.0.2/32 === 10.0.0.4/32
 host1-host2[2]: ESTABLISHED 2 minutes ago, 10.0.0.102[10.0.0.102]...10.0.0.103[10.0.0.103]
 host1-host2[2]: IKEv2 SPIs: 6e609f6d603ebf1f_i* d4c2e8c2657b8e0b_r, pre-shared key reauthentication in 48 minutes
 host1-host2[2]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_2048
 host1-host2{1}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c8221d22_i cd15e60c_o
 host1-host2{1}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 12 minutes
 host1-host2{1}:   10.0.0.2/32 === 10.0.0.3/32
root@OVS1:~/cipher-OVS1#
```

Kuvio 62. Automatisoinnilla luodut suojaussidokset

Liitteessä 3 on playbookin kulku vaiheittain esitettynä. Liitteessä 4 ja 5 on cipher-OVS1-kontin automatisoinnin luomat konfiguraatiotiedostot.

7.2 Uuden kytkimen luonti

Verkon dynaamisuutta haluttiin lisätä myös automatisoimalla uusien kytkinten implementointi verkkoon. Asennusprosessi tapahtui kääntämällä luvuissa 5.3, 5.4 ja 5.5 käytetyt asennusohjeet ja komennot YAML-ohjelmointikielelle. Ansiblen hosts-tiedostoon luotiin uusi ryhmä ”new_machines”, jonka sisälle lisättiin uusi etäjärjestelmä (kts. kuvio 63). Asennusta varten ryhmälle tehtiin ryhmän muuttujatiedosto /etc/ansible/group_vars/new_machines-tiedosto, johon määriteltiin OVS-tietokannan, rekisterin ja Consulin IP-osoitteet (kts. kuvio 64).

```
root@consul:/etc/ansible# cat hosts
# This is the default ansible 'hosts' file.

localhost ansible_connection=local

[switches]
OVS1 ansible_host=10.0.2.21
OVS2 ansible_host=10.0.2.22
OVS3 ansible_host=10.0.2.30
OVS4 ansible_host=10.0.2.31

[new_machines]

test_machine ansible_host=10.0.2.32
```

Kuvio 63. Uusi hosts-tiedosto

```
root@consul:/etc/ansible# cat group_vars/new_machines
consul_ip: 10.0.2.23
registry_ip: 10.0.2.23
ovn_remote_ip: 10.0.2.21

root@consul:/etc/ansible#
```

Kuvio 64. new_machines-ryhmän muuttujat

Playbookissa new_machines.yml määriteltiin ryhmään kuuluvat jäsenet ovs- ja docker-rooleihin. Ovs-rooli suoritti OVS:n asennuksen ja käynnistyksen ja docker-rooli asensi dockerin ja teki tarvittavat liitokset rekisteriin, OVS-tietokantaan ja Consuliin.

```
root@consul:/etc/ansible# cat new_switch.yml
---

 - name: create new openvswitch
   hosts: new_machines

   roles:
     - ovs
     - docker

root@consul:/etc/ansible#
```

Kuvio 65. Playbook uuden kytkimen luomiseen

Ovs-roolin roles/ovs/tasks/main.yml-tiedostoon tehtiin varsinainen komentojen käännöstyö. Ensimmäisenä apt-moduulilla päivitettiin ja ladattiin tarvittavat paketit, jonka jälkeen git-moduulilla ladattiin OVS:n git-repositoriosta OVS:n versio 2.5. Seuraavaksi ansible suoritti ”ps aux”-komennon, jonka tuloste otetaan muistiin register-moduulilla. Tämän jälkeen käytettiin Ansiblen shell-moduulia, jolla suoritettiin asennuksen vaatimat komennot etäjärjestelmässä. Shell-moduulin kanssa käytettiin ”item”-muuttujaa, jolloin shellin suorittamat komennot suoritettiin vaaditussa järjestyksessä. ”args”-parametrilla määritettiin kansio, jossa komennot suoritettiin. ”when”-ehdolla estettiin shell-moduulin käyttö hyödyntäen register-moduulia, mikäli etäjärjestelmässä oli jo OVS-prosessi käynnissä. Roolin viimeisessä vaiheessa OVS-palvelu käynnistettiin. Liitteessä 6 on esitetty ovs-roolin tehtävät.

Dockerin asennus aloitettiin myös pakettien päivittämisellä hyödyntäen apt-moduulia. Apt-moduulilla lisättiin myös Dockerin repositorion vaatima GPG-avain. Tämän lisäksi Dockerin repositorio lisättiin etäjärjestelmän sources-listaan. Pakettien asennuksen jälkeen playbook sammutti Docker-prosessin, sillä Docker tuli käynnistää lisävaadittavilla lisäparametreilla. Consulin, privaatin rekisterin ja OVS-tietokannan yhteyksiin käytettiin samoja komentoja kuin luvussa 5.5, mutta ne suoritettiin ovs-roolin tavoin käyttäen shell-moduulia. Yhteyksien luomiseen hyödynnettiin ryhmämuuttujia, jolloin ansible haki new_switches-ryhmän group_vars-tiedostosta ”consul_ip”-, ”ovn_remote_ip”- ja ja ”registry_ip”-kenttiä vastaavat arvot komentoa suorittaessa. Etäjärjestelmän oman IP-osoitteen mainostamiseen käytettiin Ansiblen keräämiä faktoja, jossa ”ansible_default_ipv4.address”-kentällä tarkoitetaan etäjärjestelmän oletusrajapintaa (kts. kuvio 66). Liitteessä 8 on esitetty koko roles/docker/tasks/main.yml-tiedosto. Ansible ei tee suorita komentoja, jos muutoksia ei ole havaittavissa. Shell-moduulilla on kuitenkin ominaisuus, joka suorittaa komennot aina, sillä Ansible ei kykene havaitsemaan aikaisemmin etäjärjestelmässä suoritettuja komentoja. Tämän vuoksi register-moduulilla ja ”when”-ehdolla totutettiin ehto, joka esti shell-moduulin tekemät käskyt, jos Docker oli jo käynnissä.

```
- name: start docker daemon with consul connection
  shell: "{{ item }}"
  with_items:
    - docker daemon --cluster-store=consul://{{ consul_ip }}:8500 --cluster-advertise={{ ansible_default_ipv4.add
ress }}:0 --insecure-registry={{ registry_ip }}:5000 &
    - ovs-vsctl set Open_vSwitch . external_ids:ovn-remote="tcp:{{ ovn_remote_ip }}:6640" external_ids:ovn-encap-
ip={{ ansible_default_ipv4.address }} external_ids:ovn-encap-type="geneve"
    - ovn-docker-overlay-driver --detach
  when: "{{ 'docker daemon --cluster-store=consul' not in consul.stdout }}"
```

Kuvio 66. Docker-roolin shell-moduuli

Toteutuksessa uusi virtuaalikone lisättiin new_machines-ryhmään ja sille suoritettiin new_switch-playbook. Ansible asensi OVS-kytkimen ja Dockerin virtuaalikoneeseen ja loi sille tarvittavat yhteydet. Kuviossa 67 on todennettu onnistunut asennus, sillä *”docker network ls”*-komento tulostaa Consulin synkronoiman ”sdn_network”-verkon uudelle virtuaalikytkimelle ja *”ovs-vsctl show”*-komento näyttää uuden br-int-kytkimen. New_switch-playbookin suorittaminen on esitetty liitteessä 8.

```
root@test-vm:~# docker network ls
NETWORK ID          NAME                DRIVER
a698f77cbf82        bridge              bridge
c69b02d952c7        host                host
5a926ea3cc11        none                null
02e8ca420960        sdn_network         openvswitch
root@test-vm:~# ovs-vsctl show
d055e55a-dfb1-4a9c-bf81-a333a045df76
    Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.5.1"
root@test-vm:~#
```

Kuvio 67. Dockerin ja OVS:n asennus

8 Johtopäätökset

8.1 Työn lopputulos

Toimeksiannon oltua melko avoin ja vapaasti toteutettavissa, tavoitteena oli luoda lähes mielivaltainen salauspalvelu SDN-pohjaiseen kytkinverkkoon. Toteutuksessa käytettyjen komponenttien valinnat onnistuivat erittäin hyvin, sillä suuria ongelmia

ohjelmistojen yhteensopivuudessa ei tullut juuri missään toteutuksen vaiheessa. Vaikka aiheesta ei juurikaan ollut saatavilla käytännön läheisiä julkaisuja, runko salauspalvelulle sai muotonsa jo melko aikaisessa vaiheessa opinnäytetyöprosessia. Salauksen suorittaminen Docker-kontin sisällä helpotti ja yksinkertaisti ympäristön hallintaa merkittävästi, jolla kyettiin vähentämään toteutuksessa tarvittavien virtuaalikoneiden määrää. Ansiblen tuominen ympäristöön mahdollisti IPsec-tunneleiden keskitetyn konfiguroinnin merkittävästi pienemmällä työmäärällä verrattuna manuaalisesti ja paikallisesti konfiguroitaviin tunneleihin. Automatisoinnilla on suuri merkitys suuremmissa verkkokokonaisuuksissa, jos kytkimiä ja palvelun asiakkaita on huomattavasti enemmän. Salauspalvelu ei kuitenkaan tullut täysin valmiiksi, sillä verkon kontrollerilla tuli silti manuaalisesti luoda kytkinkohtaisia voita, jotta liikenne saatiin ohjattua salauskomponenteille. Palvelun täydellinen automatisointi olisi vaatinut verkon kontrollerin tietokannasta haettuja kytkinten tilatietoja, jotta voita oltaisiin pystytty käsittelemään dynaamisesti.

Verkon laajentamiseen luotiin myös Ansiblella oma playbook, jotta tulevaisuudessa verkon laajentaminen olisi nopeampaa. Playbookin tarkoituksena oli luoda tyhjästä ubuntu-palvelintemplaatista täysin käyttövalmis kytkin osaksi jo olemassa olevaa verkkoa. Prosessi automatisoitiin niin pitkälle kuin mahdollista. Playbookin suorittamisen jälkeen ainoa manuaalisesti tehtävä konfgurointi on haluttujen porttien lisääminen kytkimeen. Toteutuksessa playbookilla ei ollut kovinkaan suurta arvoa, sillä tarkoituksena ei ollut laajentaa ympäristöä, mutta jos hypoteettisessa skenaariossa tuotantoverkkoa laajennettaisiin, automatisointi varmasti nopeuttaisi virtuaalikytkinten käyttöönottoprosessia.

8.2 Jatkokehitys

Työnlopputulosta tarkastellessa muutamia jatkokehitysideoita heräsi aiheesta. Selvittämättömästä syystä Docker-salaajakontit toimivat ainoastaan suoritettaessa toteutuksessa käytettävin parametrein. Kyseiset parametrit estävät konttien käynnistämisen taustaprosessina, joka johtaa siihen, että kontteja ei voi käynnistää Ansiblella, vaikka siihen on valmiina toimiva docker-moduuli. Kontit kuitenkin nousevat taustaprosessina käyntiin ja suojaussidokset muodostuvat konttien välille, mutta paketit eivät mene tunneleihin. Tähän yritettiinkin etsiä ratkaisua mutta, siinä ei kuitenkaan

onnistuttu. Seuraava jatkokehitysmahdollisuus olisi rakentaa PKI-sertifikaatti-infrastruktuuri konteille, jolloin tarve esijaetuille avaimille poistuisi. strongSwanilla on jo valmiiksi tuki sertifikaateille, joilla varmennus toteutettaisiin. Myös automatisoitu kytkinten vuotaulujen päivittäminen olisi täysin kehitettävissä, sillä ONOSin tietokannasta on mahdollista hakea tarvittavat tilatiedot ja tätä ominaisuutta onkin jo työstetty muissa Cyber Trust –projektin hankkeissa.

8.3 Pohdinta

Oppimisprosessina opinnäytetyö toimi erinomaisesti, sillä tutkinto-ohjelman aikaisemmissa vaiheissa ei ole kertaakaan tarvinnut syventyä opinnäytetyön vaatimella teoreettiselle tai käytännön tasolle ja koenkin oppineeni verkoista, tietoturvasta ja SDN:stä, mutta ennenkaikkea itse prosessin eteenpäin viemisestä ja omistautumisesta parhaan mahdollisen lopputuloksen saavuttamiseksi. Opinnäytetyön tekemisessä aikaisemmin suoritetusta työharjoittelusta samaiseen projektiin oli valtavasti hyötyä ja toteutuksesta ei olisi millään tullut näin laaja ilman kuukausien mittaista aikaisempaa perehtymistä SDN-verkkotekniikkaan. Työn edetessä myös tilaajan ja opinnäytetyön ohjaajalla oli suuri merkitys toteutuksen onnistumiseen, sillä palavereita järjestettiin tiuhaan ja ongelmatilanteissa apua oli helppo kysyä.

Opinnäytetyöstä on merkittävästi hyötyä itselleni syventämällä virtualisoinnin merkityksen tärkeyttä tietoverkoissa, mutta myös toimeksiantajalle, sillä salauksen jatkokehityksen kaikki toiminnot ovat täysin valmiita opiskelijoiden työharjoittelun ja opinnäytöiden aiheita. Dokumentointi toteutettiinkin kattaen jokainen mahdollinen työvaihe, jotta työtä voidaan käyttää jatkossa projektin kehitykseen. Opinnäytetyö toimii myös valmiina julkaisuna Cyber Trust-tutkimusohjelmalle, jota voidaan esitellä osana ammattikorkeakoulun projektin aikaansaannoksia tulevissa kokoontumistapahtumissa muille tutkimusohjelman organisaatioille.

Lähteet

An Introduction to IPsec (INTERNET PROTOCOL SECURITY). 2001. Perehdytys IPsec-protokollapinoon. Viitattu 15.2.2016. http://csrc.nist.gov/publications/nistbul/03-01.pdf

Anderson, T., Balakrishnan, H., McKeown, N., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. & Turner, J. 2008. OpenFlow: Enabling Innovation in Campus Networks. OpenFlow-kotisivut. Viitattu 3.2.2016. http://archive.OpenFlow.org/documents/OpenFlow-wp-latest.pdf

Appenzeller G., Balland P., Barker C., Beckmann C., Casado M., Cohn D., Crabbe E., Curtis S., Das S., dHeureuse N., Ding W., Dunbar L., Erickson D., Gandham S., Gibb G., Heller B., Kis Z., Kobayashi M., Lantz B., Madabushi R., Malek D., McDysan D., McKeown N., Mizrahi T., Moses Y., Nygren A., Orr M., Pettit J., Pfaff B., Poutievski L., Ramanathan R., Price R., Sherwood R., Schneider F., Takahashi M., Talayco D., Tonsing J., Tourrilhes J., Vicisano L., Ward D., Yabe T., Yadav N., Yap K. & Yiakoumis Y. 2014. Open Networking Foundation. OpenFlow Switch Specification version 1.5.0. Viitattu 5.2.2016. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/OpenFlow/OpenFlow-switch-v1.5.0.noipr.pdf

Benitez J., Bugenhagen M., Chiosi M., Clarke D., Cui C., Damker H., Delisle D., Demaria E., Deng H., Fargano M., Feger J., Fukui M., Guardini I., Khan W., Kolias C., Lopez D., Loudier Q., Manzalini A., Matsuzaki T., Michel U., Minerva R., Ogaki K., Prodip S., Reid A., Ruhl F., Salguero F., Shimano K. & Willis P. 2012. Network Functions Virtualisation. Viitattu 14.2.2016. https://portal.etsi.org/NFV/NFV_White_Paper.pdf

Chouhan P. K., Hinnegan J., Fraser B., Lake D., Miller M., Rao N., Scott-Hayward S., Sezer S. & Viljoen N. 2013. Are We Ready for SDN? Implementation Challenges for Software-Defined-networks. Viitattu 4.2.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6553676

IKE version 2 protocol, Initial exchanges. N.d. IKE version 2 Avaintenvaihtoprosessin esitys IBM:n tietopankissa. Viitattu 17.2.2016. https://www-01.ibm.com/support/knowledgecenter/SSLTBW_1.13.0/com.ibm.zos.r13.hald001/initex.htm%23initex

Ganga, I. & Gross, J. 2016. Geneve: Generic Network Virtualization Encapsulation draft-ietf-nvo3-geneve-01. Viitattu 13.4.2016. https://tools.ietf.org/html/draft-ietf-nvo3-geneve-01

Kaufman, C. 2005. RFC 4306 Internet Key Exchange (IKEv2) Protocol. Viitattu 17.2.2016. https://tools.ietf.org/html/rfc4306

Kent, S. 2005a. RFC 4302 IP Authentication Header. Viitattu 16.2.2106. https://www.ietf.org/rfc/rfc4302.txt

Kent, S. 2005b. RFC 4303 Encapsulating Security Payload. Viitattu 16.2.2016. https://www.ietf.org/rfc/rfc4303.txt

Open Network Operating System (ONOS). N.d. Artikkeli ONOS:sta. Viitattu 25.2.2016 https://www.sdxcentral.com/projects/on-lab-open-network-operating-system-onos/

Open vSwitch Manual. N.d a. ovs-vsctl –ohjelman manuaali. Viitattu 8.3.2016. http://openvswitch.org/support/dist-docs/ovs-vsctl.8.html

Open vSwitch Manual. N.d b. ovs-ofctl –ohjelman manuaali. Viitattu 8.3.2016. http://openvswitch.org/support/dist-docs/ovs-ofctl.8.html

Overview: How Ansible works. N.d. Yleiskuva Ansiblen toiminnasta. Viitattu 31.3.2016. https://www.ansible.com/how-ansible-works

Rao S. 2015. SDN Series Part Seven: ONOS. Viitattu 8.3.2016. http://thenewstack.io/open-source-sdn-controllers-part-vii-onos/

Savola, R. 2014. DIGILE CyberTrust and Related Security Projects. Viitattu 25.1.2016. http://www.vtt.fi/files/sites/eemeli18/12_Reijo_Savola_security_calls.pdf

Software-Defined Networking: The New Norm for Networks. 2012. Open networking foundation julkaisu uudesta verkkotekniikasta. Viitattu 3.2.2016 https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

strongSwan – About. 2015. Info strongSwanista. Viitattu 10.3.2016. https://www.strongswan.org/about.html

Understand the architecture. N.d. Docker-alustan verkkosivut. Viitattu 29.2.2016. https://docs.docker.com/engine/understanding-docker/

What are SDN Controllers (or SDN Controllers Platforms)?. N.d. Artikkeli SDN:stä. Viitattu 25.2.2016. https://www.sdxcentral.com/resources/sdn/sdn-controllers/

Why Open vSwitch?. 2014. Tiivistelmä OVS:n ominaisuuksista. Viitattu 25.2.2016 https://github.com/openvswitch/ovs/blob/master/WHY-OVS.md

Liitteet

Liite 1. Dockerfile salaajakonttia varten

```
FROM buildpack-deps:jessie

RUN mkdir -p /conf
RUN mkdir -p /etc/ipsec/configurations

RUN apt-get update && apt-get install -y \
  libgmp-dev \
  iptables \
  xl2tpd \
  module-init-tools

ENV STRONGSWAN_VERSION 5.3.5

RUN mkdir -p /usr/src/strongswan \
        && curl -SL
"https://download.strongswan.org/strongswan-
$STRONGSWAN_VERSION.tar.gz" \
        | tar -zxC /usr/src/strongswan --strip-components 1 \
        && cd /usr/src/strongswan \
        && ./configure --prefix=/usr --
sysconfdir=/etc/ipsec/configurations \
                --enable-eap-radius \
                --enable-eap-mschapv2 \
                --enable-eap-identity \
                --enable-eap-md5 \
                --enable-eap-mschapv2 \
                --enable-eap-tls \
                --enable-eap-ttls \
                --enable-eap-peap \
                --enable-eap-tnc \
                --enable-eap-dynamic \
                --enable-xauth-eap \
                --enable-openssl \
        && make -j \
        && make install \
        && rm -rf /usr/src/strongswan

# Strongswan Configuration
ADD ipsec.conf /etc/ipsec/configurations/ipsec.conf
ADD strongswan.conf /etc/ipsec/configurations/strongswan.conf

# XL2TPD Configuration
ADD xl2tpd.conf /etc/xl2tpd/xl2tpd.conf
ADD options.xl2tpd /etc/ppp/options.xl2tpd

ADD run.sh /run.sh
ADD vpn_adduser /usr/local/bin/vpn_adduser
ADD vpn_deluser /usr/local/bin/vpn_deluser
ADD vpn_setpsk /usr/local/bin/vpn_setpsk
ADD vpn_unsetpsk /usr/local/bin/vpn_unsetpsk
ADD vpn_apply /usr/local/bin/vpn_apply

# The password is later on replaced with a random string
ENV VPN_USER user
ENV VPN_PASSWORD password
ENV VPN_PSK password
```

```
VOLUME ["/etc/ipsec.d"]

EXPOSE 4500/udp 500/udp 1701/udp

CMD ["/run.sh"]
```

Liite 2. roles/ipsec/vars/main.yml-tiedosto

```
---
switches:
    - switch_name: cipher-OVS1
      hosts:
        - name: host1
          local_ip: 10.0.0.2
          local_container: 10.0.0.102
          connections:
           - remote_host: host2
             remote_ip: 10.0.0.3
             remote_container: 10.0.0.103
           - remote_host: host3
             remote_ip: 10.0.0.4
             remote_container: 10.0.0.104
           - remote_host: host4
             remote_ip: 10.0.0.5
             remote_container: 10.0.0.106

    - switch_name: cipher-OVS2
      hosts:
        - name: host2
          local_ip: 10.0.0.3
          local_container: 10.0.0.103
          connections:
           - remote_host: host1
             remote_ip: 10.0.0.2
             remote_container: 10.0.0.102
           - remote_host: host3
             remote_ip: 10.0.0.4
             remote_container: 10.0.0.104
           - remote_host: host4
             remote_ip: 10.0.0.5
             remote_container: 10.0.0.106


    - switch_name: cipher-OVS3
      hosts:
        - name: host3
          local_ip: 10.0.0.4
          local_container: 10.0.0.104
          connections:
           - remote_host: host1
             remote_ip: 10.0.0.2
             remote_container: 10.0.0.102
           - remote_host: host2
             remote_ip: 10.0.0.3
             remote_container: 10.0.0.103
           - remote_host: host4
             remote_ip: 10.0.0.5
             remote_container: 10.0.0.106

    - switch_name: cipher-OVS4
```

```
hosts:
  - name: host4
    local_ip: 10.0.0.5
    local_container: 10.0.0.106
    connections:
     - remote_host: host1
       remote_ip: 10.0.0.2
       remote_container: 10.0.0.102
     - remote_host: host2
       remote_ip: 10.0.0.3
       remote_container: 10.0.0.103
     - remote_host: host3
       remote_ip: 10.0.0.4
       remote_container: 10.0.0.104
```

Liite 3. Konfiguraatiohallinnan suorittaminen

```
root@consul:/etc/ansible# ansible-playbook ipsec_tunnel.yml

PLAY [create configurations to local machine] *********************************

TASK [setup] *******************************************************************
ok: [localhost]

TASK [ipsec : use template to create configuration file] ***********************
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.103',
u'remote_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host1', u'local_container': u'10.0.0.102',
u'local_ip': u'10.0.0.2'}], u'switch_name': u'cipher-OVS1'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host2', u'local_container': u'10.0.0.103',
u'local_ip': u'10.0.0.3'}], u'switch_name': u'cipher-OVS2'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host3', u'local_container': u'10.0.0.104',
u'local_ip': u'10.0.0.4'}], u'switch_name': u'cipher-OVS3'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remote_hos
t': u'host3', u'remote_ip': u'10.0.0.4'}], u'name': u'host4', u'local_container': u'10.0.0.106',
u'local_ip': u'10.0.0.5'}], u'switch_name': u'cipher-OVS4'})

TASK [ipsec : create psk file] *************************************************
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.103',
u'remote_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host1', u'local_container': u'10.0.0.102',
u'local_ip': u'10.0.0.2'}], u'switch_name': u'cipher-OVS1'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host2', u'local_container': u'10.0.0.103',
u'local_ip': u'10.0.0.3'}], u'switch_name': u'cipher-OVS2'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host3', u'local_container': u'10.0.0.104',
u'local_ip': u'10.0.0.4'}], u'switch_name': u'cipher-OVS3'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remote_hos
t': u'host3', u'remote_ip': u'10.0.0.4'}], u'name': u'host4', u'local_container': u'10.0.0.106',
u'local_ip': u'10.0.0.5'}], u'switch_name': u'cipher-OVS4'})
```

```
TASK [ipsec : use template to create parameters for strongswan] ***************
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.103',
u'remote_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host1', u'local_container': u'10.0.0.102',
u'local_ip': u'10.0.0.2'}], u'switch_name': u'cipher-OVS1'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.104', u'remo
te_host': u'host3', u'remote_ip': u'10.0.0.4'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host2', u'local_container': u'10.0.0.103',
u'local_ip': u'10.0.0.3'}], u'switch_name': u'cipher-OVS2'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.106', u'remote_hos
t': u'host4', u'remote_ip': u'10.0.0.5'}], u'name': u'host3', u'local_container': u'10.0.0.104',
u'local_ip': u'10.0.0.4'}], u'switch_name': u'cipher-OVS3'})
changed: [localhost] => (item={u'hosts': [{u'connections': [{u'remote_container': u'10.0.0.102',
u'remote_host': u'host1', u'remote_ip': u'10.0.0.2'}, {u'remote_container': u'10.0.0.103', u'remo
te_host': u'host2', u'remote_ip': u'10.0.0.3'}, {u'remote_container': u'10.0.0.104', u'remote_hos
t': u'host3', u'remote_ip': u'10.0.0.4'}], u'name': u'host4', u'local_container': u'10.0.0.106',
u'local_ip': u'10.0.0.5'}], u'switch_name': u'cipher-OVS4'})

PLAY [copy configurations from local machine to remote locations] **************

TASK [setup] *******************************************************************
ok: [OVS2]
ok: [OVS3]
ok: [OVS4]
ok: [OVS1]

TASK [copy] ********************************************************************
changed: [OVS3]
changed: [OVS1]
changed: [OVS2]
changed: [OVS4]

TASK [copy] ********************************************************************
changed: [OVS2]
changed: [OVS3]
changed: [OVS1]
changed: [OVS4]

TASK [copy] ********************************************************************
changed: [OVS2]
changed: [OVS3]
changed: [OVS1]
changed: [OVS4]

TASK [command] *****************************************************************
changed: [OVS3]
changed: [OVS4]
changed: [OVS2]
changed: [OVS1]

PLAY RECAP *********************************************************************
OVS1                       : ok=5    changed=4    unreachable=0    failed=0
OVS2                       : ok=5    changed=4    unreachable=0    failed=0
OVS3                       : ok=5    changed=4    unreachable=0    failed=0
OVS4                       : ok=5    changed=4    unreachable=0    failed=0
localhost                  : ok=4    changed=3    unreachable=0    failed=0

root@consul:/etc/ansible#
```

Liite 4. Cipher-OVS1-kontin ipsec.conf-tiedosto

```
#Config for cipher-OVS1

config setup

conn %default
        ikelifetime=60m
        keylife=20m
```

```
        rekeymargin=3m
        keyingtries=3
        authby=psk
        keyexchange=ikev2
        mobike=no


conn host1-host2
        left=10.0.0.102
        leftsubnet=10.0.0.2
        right=10.0.0.103
        rightsubnet=10.0.0.3
        auto=start


conn host1-host3
        left=10.0.0.102
        leftsubnet=10.0.0.2
        right=10.0.0.104
        rightsubnet=10.0.0.4
        auto=start


conn host1-host4
        left=10.0.0.102
        leftsubnet=10.0.0.2
        right=10.0.0.106
        rightsubnet=10.0.0.5
        auto=start
```

Liite 5. Cipher-OVS1-kontin ipsec.secrets-tiedosto

```
#Pre shared keys for cipher-OVS1

10.0.0.103 : PSK "secret_psk"

10.0.0.104 : PSK "secret_psk"

10.0.0.106 : PSK "secret_psk"
```

Liite 6. /etc/ansible/roles/ovs/tasks/main.yml-tiedosto

```
---

 - name: update apt
   apt: update_cache=yes

 - name: install required packages
   apt: name={{ item }} state=latest
   with_items:
     - autoconf
     - libtool
     - sparse
     - openssl
```

```
     - pkg-config
     - make
     - gcc
     - libssl-dev
     - git

 - name: download ovs from github
   git: repo=https://github.com/openvswitch/ovs.git
dest=/root/ovs version=branch-2.5

 - name: check for installed OVS
   command: ps aux
   register: ovs_running

 - name: install ovs
   shell: "{{ item }}"
   args:
     chdir: /root/ovs/
   with_items:
     - ./boot.sh
     - ./configure --prefix=/usr --localstatedir=/var  --
sysconfdir=/etc --enable-ssl --with-linux=/lib/modules/`uname -
r`/build
     - make -j3
     - make install
     - cp debian/openvswitch-switch.init
/etc/init.d/openvswitch-switch
   when: "{{ 'ovs-vswitchd: monitoring pid' not in
ovs_running.stdout }}"

 - name: install python-openvswitch
   apt: name={{ item }} state=latest
   with_items:
     - python-openvswitch


 - name: start OVS
   command: /etc/init.d/openvswitch-switch start
```

Liite 7. /etc/ansible/roles/docker/tasks/main.yml

```
---

- name: Update apt cache
  apt: update_cache=yes

- name: Add the new GPG key
  apt_key: keyserver="hkp://p80.pool.sks-keyservers.net:80"
id="58118E89F3A912897C070ADBF76221572C52609D"

- name: Add docker repository to sources list
  apt_repository: repo="deb https://apt.dockerproject.org/repo
ubuntu-trusty main"

- name: Update apt cache
  apt: update_cache=yes

- name: Install docker
  apt: name={{ item }} state=present
```

```
    with_items:
      - apt-transport-https
      - ca-certificates
      - apparmor
      - docker-engine
      - python-setuptools

- name: install pip
  easy_install: name=pip state=latest

- name: install flask
  pip: name="{{ item }}" state=latest
  with_items:
    - flask

- name: Stop docker service
  service: name=docker state=stopped

- name: check for consul connection
  command: ps aux
  register: consul

- name: start docker daemon with consul connection
  shell: "{{ item }}"
  with_items:
    - docker daemon --cluster-store=consul://{{ consul_ip
}}:8500 --cluster-advertise={{ ansible_default_ipv4.address
}}:0 --insecure-registry={{ registry_ip }}:5000 &
    - ovs-vsctl set Open_vSwitch . external_ids:ovn-
remote="tcp:{{ ovn_remote_ip }}:6640" external_ids:ovn-encap-
ip={{ ansible_default_ipv4.address }} external_ids:ovn-encap-
type="geneve"
    - ovn-docker-overlay-driver --detach
  when: "{{ 'docker daemon --cluster-store=consul' not in
consul.stdout }}"
```

Liite 8. Uuden kytkimen asennus Ansiblella

```
root@consul:/etc/ansible# ansible-playbook new_switch.yml

PLAY [create new openvswitch] **************************************************

TASK [setup] *******************************************************************
ok: [test-vm]

TASK [ovs : update apt] ********************************************************
ok: [test-vm]

TASK [ovs : install required packages] *****************************************
changed: [test-vm] => (item=[u'autoconf', u'libtool', u'sparse', u'openssl', u'pkg-config', u'make', u'gcc', u'li
bssl-dev', u'git'])

TASK [ovs : download ovs from github] ******************************************
changed: [test-vm]

TASK [ovs : check for installed OVS] *******************************************
changed: [test-vm]

TASK [ovs : install ovs] *******************************************************
changed: [test-vm] => (item=./boot.sh)
changed: [test-vm] => (item=./configure --prefix=/usr --localstatedir=/var  --sysconfdir=/etc --enable-ssl --with
-linux=/lib/modules/`uname -r`/build)
changed: [test-vm] => (item=make -j3)
changed: [test-vm] => (item=make install)
changed: [test-vm] => (item=cp debian/openvswitch-switch.init /etc/init.d/openvswitch-switch)

TASK [ovs : install python-openvswitch] ****************************************
changed: [test-vm] => (item=[u'python-openvswitch'])

TASK [ovs : start OVS] *********************************************************
changed: [test-vm]
```

```
TASK [docker : Update apt cache] ***********************************************
ok: [test-vm]

TASK [docker : Add the new GPG key] ********************************************
changed: [test-vm]

TASK [docker : Add docker repository to sources list] **************************
changed: [test-vm]

TASK [docker : Update apt cache] ***********************************************
ok: [test-vm]

TASK [docker : Install docker] *************************************************
changed: [test-vm] => (item=[u'apt-transport-https', u'ca-certificates', u'apparmor', u'docker-engine', u'python-
setuptools'])

TASK [docker : install pip] ****************************************************
changed: [test-vm]

TASK [docker : install flask] **************************************************
changed: [test-vm] => (item=flask)

TASK [docker : Stop docker service] ********************************************
changed: [test-vm]

TASK [docker : check for consul connection] ************************************
changed: [test-vm]

TASK [docker : start docker daemon with consul connection] *********************
changed: [test-vm] => (item=docker daemon --cluster-store=consul://10.0.2.23:8500 --cluster-advertise=10.0.2.32:0
 --insecure-registry=10.0.2.23:5000 &)
changed: [test-vm] => (item=ovs-vsctl set Open_vSwitch . external_ids:ovn-remote="tcp:10.0.2.21:6640" external_id
s:ovn-encap-ip=10.0.2.32 external_ids:ovn-encap-type="geneve")
changed: [test-vm] => (item=ovn-docker-overlay-driver --detach)

PLAY RECAP *********************************************************************
test-vm                    : ok=18   changed=14   unreachable=0    failed=0

root@consul:/etc/ansible#
```