

Henri Silfver

# Tunkeilijan havaitsemisjärjestelmän testaus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

3.5.2016

Tekijä Otsikko	Henri Silfver Tunkeilijan havaitsemisjärjestelmän testaus
Sivumäärä Aika	36 sivua + 1 liite 3.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Yliopettaja Janne Salonen
<p>Yritysten tietoverkot ja niihin kohdistuvat hyökkäykset ovat kehittyneet entistä monimutkaisemmiksi. Ratkaisuna on toteuttaa järjestelmä, jonka avulla voidaan havaita tunkeutuminen tietoverkossa ja auttaa tietomurron tutkintaa.</p> <p>Insinööriyön aihe on tunkeilijan havaitsemisjärjestelmän testaus laboratorioympäristössä. Tunkeilijan havaitsemisjärjestelmää testataan todennäköisillä tunkeutumistilanteilla tietoverkkoa vastaan.</p> <p>Työn tavoite oli tunkeilijan havaitsemisjärjestelmän lisääminen laboratorioympäristöön. Laboratorioympäristössä testataan järjestelmän toimivuutta ja pyritään tarkemmin analysoida tunkeutumisen laajuutta. Tarkoituksena on helpottaa haitallisen verkkoliikenteen tutkimista lähiverkossa. Työssä näytetään tunkeutumisen havaitsemisjärjestelmän toiminta käytännössä ja pohditaan mahdollisia parannusehdotuksia järjestelmään.</p> <p>Insinööriyössä tutustuttiin erilaisiin tietoliikennepakettien kaappaustapoihin. Tietoliikennepaketteja kaapataan riippuen tarkoituksesta ja toimintaympäristöstä. Tunkeilijan havaitsemisjärjestelmien toteutustavoissa on eroavaisuuksia toimintaympäristöstä riippuen. Näitä eroavaisuuksia pyritään näyttämään selvittämällä toteutustapojen etuja ja heikkouksia.</p> <p>Tunkeilijan havaitsemisjärjestelmän asennus ja testaaminen suoritettiin laboratorioympäristössä. Työn lopputuloksena on järjestelmä ja toimintamalli, jonka avulla voidaan havaita tunkeutuminen lähiverkkoon ja tarkemmin analysoida mahdollisen tietomurron vakavuutta. Insinööriyön pohjalta on mahdollista toteuttaa tietoturvaratkaisu jo olemassa olevaan tietoverkkoon lisäämään järjestelmän turvallisuutta. Tämän lisäksi annetaan merkittäviä parannusehdotuksia järjestelmän ylläpitoon ja tehokkaampaan käyttöön.</p>	
Avainsanat	IDS, Tietoturva, Pakettikaappaus, Snort

Author Title	Henri Silfver Intrusion Detection System testing
Number of Pages Date	36 pages + 1 appendix 3 May 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Computer Networks
Instructor	Janne Salonen, Principal Lecturer
<p>Corporate networks and threat models against these networks have become even more complicated than ever. A solution to this ever evolving security landscape is to implement a system which detects threats and helps incident response.</p> <p>The topic of this thesis is intrusion detection system testing on a lab environment. Intrusion detection system will be tested against threats which can be considered as possible attack scenarios.</p> <p>The aim of this study is to implement intrusion detection system to an existing lab network. Functionality of the intrusion detection system will be tested and a scope of the incident will be further identified and analyzed. The purpose of this study is to reduce the chance of missing security threats which could compromise critical assets. Performance of intrusion detection system is tested in practice and considerable improvements to system are discussed.</p> <p>Common packet capture methods are explained in local area networks. The purpose of collecting network traffic is demonstrated depending on network environment and goals of the packet capture. Scale of implementing an intrusion detection system varies across different networks. These differences as well as pros and cons of different implementations will be covered in this study.</p> <p>Installation and testing of intrusion detection system was performed in a lab environment. The end-result of this study is a system and an operating model which helps addressing and managing the aftermath of a security breach or attack. Based on this study it is possible to implement a functional security solution which significantly increases security of an existing network. Suggestions and recommendations are given in the end to enhance maintenance and efficiency of implemented intrusion detection system.</p>	
Keywords	IDS, Information Security, Packet Capturing, Snort

# Sisällys

## Lyhenteet

1	Johdanto	1
2	Verkkoliikenteen kaappaus	2
2.1	Käyttöjärjestelmäkohtaiset toimintatavat	2
2.1.1	Unixin kaltaisissa ympäristöissä	3
2.1.2	Windows-ympäristössä	3
2.2	Pcap-tiedostomuoto	4
2.2.1	Yleinen otsake	4
2.2.2	Paketin otsake	5
2.2.3	Paketin data	6
2.3	Pakettikaappausohjelmat	7
2.3.1	Wireshark	8
2.3.2	Tcpdump	8
3	Tunkeilijan havaitsemisjärjestelmät	9
3.1	Arkkitehtuuri ja toteutustavat	10
3.1.1	Yksitasoinen	10
3.1.2	Monitasoinen	11
3.2	Snort	14
3.2.1	Tarkastelutila	14
3.2.2	Pakettilokitustila	15
3.2.3	Tunkeutumisen havaitsemistila	15
3.2.4	Säännöt	15
3.3	Kaupalliset seuraavan sukupolven IPDS-ratkaisut	17
3.3.1	Verkkoympäristötietoisuus	17
3.3.2	Sovellus ja käyttäjätietoisuus	17
3.3.3	Laitekohtainen käyttäytyminen	18
3.3.4	Automatisointi	18
4	Snort-asennus ja konfigurointi	18
4.1	Esivaatimusten asennus	19
4.2	Snortin asentaminen	20
4.3	Konfigurointi	20
4.3.1	Muutokset NIDS-tilaa varten	21

4.3.2	Sääntöjen hakeminen	21
5	Tunkeilijan havaitsemisjärjestelmän testaus	23
5.1	Testiympäristö	23
5.2	Hyökkäyspinta-ala	25
5.3	Hälytyksien testaus ja pakettianalysointi	26
5.3.1	Hyökkäys Web-palvelinta vastaan	26
5.3.2	Käyttäjöpäänteen saastuminen	29
5.4	Järjestelmän kehitysehdotuksia	32
6	Yhteenveto	33
	Lähteet	35
	Liitteet	
	Liite 1. Snort asennus Ubuntu 14.04.01 versioon komentoriviltä	

## Lyhenteet

API	<i>Application Programming Interface</i> . Ohjelmistorajapinta, jonka avulla ohjelmistot pystyvät keskustelemaan keskenään.
BSD	Avoimen lähdekoodin lisenssi, jonka avulla ohjelmistoja voidaan jakaa vapaasti tietyin ehdoin.
DNS	<i>Domain Name System</i> . Nimipalvelujärjestelmä, joka muuntaa helposti muistettavat verkkotunnukset IP-osoitteiksi.
DOS	<i>Denial of Service</i> . Palvelunestohyökkäys, jossa estetään tai häiritään väliaikaisesti verkkopalvelun toimintaa.
GUI	<i>Graphical User Interface</i> . Graafinen käyttöliittymä, joka mahdollistaa käyttäjäystävällisen työskentelemisen ohjelmiston tai laitteiston kanssa.
HIDS	<i>Host-based Intrusion Detection System</i> . Konekohtainen tunkeutumisen havaitsemisjärjestelmä, joka pyrkii havaitsemaan, analysoimaan ja ilmoittamaan käyttäjää koneen sisällä olevista haitallisista ohjelmista.
HTTP	<i>Hypertext Transfer Protocol</i> . Tiedonsiirtoprotokolla, jota muun muassa selaimet käyttävät web-sivujen tiedonsiirtoon.
HTTPS	<i>Hypertext Transfer Protocol Secure</i> . Käytetään turvalliseen tiedonsiirtoon verkkosovelluksien välillä.
IDS	<i>Intrusion Detection System</i> . Tunkeutumisen havaitsemisjärjestelmä, joka pyrkii havaitsemaan ja hälyttämään järjestelmän ylläpitäjää verkossa tapahtuvasta haitallisesta liikenteestä.
IP	<i>Internet Protocol</i> . Huolehtii pakettikytkentäisessä tietoverkossa IP-pakettien toimituksesta IP-osoitteiden perusteella.
IPS	<i>Intrusion Prevention System</i> . Tunkeilijan estojärjestelmä toteutetaan yleensä IDS-järjestelmän kanssa. Pyrkii estämään tunkeutumisen heti sen havaitsemisen jälkeen.

MAC	<i>Media Access Control</i> . Verkko liittymän yksilöivä tunnistenumero.
MySQL	Avoimeen lähdekoodiin perustuva tietokantojen hallintajärjestelmä.
NIDS	<i>Network-based Intrusion detection system</i> . Verkkopohjainen tunkeutumisen havaitsemisjärjestelmä pyrkii havaitsemaan verkossa liikkuvaa verkkoliikennettä ja ilmoittamaan haitallisista tapahtumista.
SMS	<i>Short Message Service</i> . Matkapuhelinverkossa lähetettävät tekstiviestit.
SOHO	<i>Small Office/Home Office</i> . Kotiverkko tai pienyritys verkkoratkaisuihin viittaava käsite.
SQL	<i>Structured Query Language</i> . Tietokantojen käsittelyyn käytetty ohjelmointikieli.
TCP	<i>Transmission Control Protocol</i> . Tietoliikenneprotokolla, jossa luodaan istunto kahden koneen välille. Koneiden välillä TCP-protokolla pitää huolen kuljetuskerroksen tiedonsiirrosta.
TCP/IP	<i>Transmission Control Protocol / Internet Protocol</i> . Viitekehys, jonka avulla tietokoneet keskustelevalt tietoverkon välityksellä.
Win32	Microsoft Windows -käyttöjärjestelmissä käytettävä rajapinta sovelluksille ja ohjelmistoille.

## 1 Johdanto

Monet pienet ja keskisuuret yritykset olettavat, etteivät kyberhyökkäykset saavuttaisi tai vaikuttaisi heihin, sillä hyökkääjä ei näkisi mitään arvoa hyökätä yritykseen. Täytyy kuitenkin muistaa, että hyökkäys voidaan suorittaa suorasti tai epäsuorasti. Näistä kummankin hyökkäystavan tuloksena voi olla pääsy uhrin tärkeisiin tietoihin ja varastamaan arkaluontoista materiaalia. Joissain tapauksissa hyökkäyksen kohteena olevalle koneelle voidaan suorittaa vain häiriötä tai palvelunestohyökkäys. Väliaikainen häiriö tai palvelunestohyökkäys yrityksen toiminnan kannalta tärkeisiin laitteisiin tai ohjelmistoihin voi aiheuttaa yritykselle suuret kustannukset ja estää mahdollisten uusien kauppajen syntymisen.

Opinnäytetyön tarkoitus on helpottaa haitallisen verkkoliikenteen tutkimista lähiverkossa. Tavoite on mahdollistaa järjestelmän valvojalle rajapinta, jonka kautta tämä pystyy vaivattomasti tarkastelemaan tunkeutumisen havaitsemisjärjestelmän aiheuttamia hälytyksiä. Tämän lisäksi hän pystyy vetämään nopean johtopäätöksen hyökkäystavasta ja mahdollisen uhan vaikutuksesta tietojärjestelmään.

Aluksi tutustutaan verkkoliikenteen kaappaukseen. Verkkoliikenteen kaappausosiossa otetaan katsaus menetelmiin ja tapoihin, joita käytetään tietoverkkoliikenteessä pakettien kaappaukseen. Tässä huomataan, että järjestelmillä on erilaiset tavat käsitellä verkossa kulkevaa liikennettä. Voimme huomata standardeja, jotka ovat vakiinnuttaneet paikkansa verkkoliikenteen kaappauksessa.

Luvussa tunkeilijan havaitsemisjärjestelmät perehdytään erilaisiin järjestelmiin, joilla voidaan tunnistaa verkkoon suuntautuvat hyökkäykset. Pohditaan järjestelmälle keskeistä sijoituspaikkaa verkossa ja sen tavoitetta. Käydään läpi laitteita, jotka ovat tehty tunkeilijan havaitsemista varten sekä laitteita, joihin on mahdollista asentaa tällainen ominaisuus.

Varsinainen laitteiston asennus ja konfigurointi suoritetaan työn lopuksi. Tässä vaiheessa käydään läpi asennettavan järjestelmän asennusvaiheet. Pyritään selittämään yksityiskohtaisesti, mihin konfigurointi vaikuttaa sekä tämän tarkoitus. Itse testausluvussa simuloidaan oikeaa tilannetta, jossa näytetään, kuinka tunkeilija hyökkää ulko-verkosta sisäverkon palvelimeen. Tässä pystymme tarkastelemaan hälytyksiä, joita



hyökkäys aiheuttaa IDS-järjestelmän kautta. Hälytyksien lisäksi huomaamme yksityiskohtaisemman pakettitarkastelun merkityksen selvittäessä tietomurron laajuutta.

Työn pohjalta voidaan näyttää, että asianmukainen järjestelmä voidaan asentaa jo olemassa olevaan tietoverkkoon. Tällaisen järjestelmän hyödyt ovat suuret ja helpottavat merkittävästi tietomurron selvitystyötä ja tutkimista.

## 2 Verkkoliikenteen kaappaus

Verkkoliikennettä monitoroidaan monesta eri syystä. Verkossa voi liikkua monenlaista liikennettä. Tietoverkkoinfosiööriä ja järjestelmän ylläpitäjää kiinnostavat verkon suorituskyky, korkea käytettävyys sekä verkossa tapahtuvien ongelmien ratkaisu. Näitä pyritään valvomaan näihin piirteisiin soveltuvilla työkaluilla. Vaihtoehtoisesti seurataan loki-tiedostoja suoraan verkon aktiivilaitteiden ja palvelimien muistista, jolloin hukkuminen monen megatavun tai gigatavujen kokoisiin tapahtuma lokeihin on mahdollinen. Henkilön, joka ylläpitää tai valvoo verkkoa, on tunnettava molempien toimintatapojen hyödyt ja sovellettava näitä toimintaperiaatteita maksimoidakseen tyotehokkuuden.

### 2.1 Käyttöjärjestelmäkohtaiset toimintatavat

Verkkoliikenteen kaappaukseen käytetään käyttöjärjestelmäkohtaisia työkaluja. Nämä työkalut mahdollistavat pakettikaappauksen alhaisella verkkotasolla. Seuraavaksi käydään läpi, kuinka näiden työkalujen toiminta eroaa Windowsin ja Unixin kaltaisissa ympäristöissä.

Tietokoneella toimivat verkkosovellukset keräävät verkkoliikennettä yleensä suoraan verkkokannasta (engl. network socket). Tässä käyttöjärjestelmä toimii välikätenä ja käsittelee esimerkiksi protokollakohtaiset toiminnot sekä pakettien uudelleenkokonaisuuden. Tällöin verkko-ohjelman kehittäjän ei tarvitse miettiä, mitä tapahtuu alhaisella tasolla liikenteen käsittelyssä. Ohjelman kehittäjän tarvitsee vain käsitellä verkkokantaa, kuten normaalia tietovirtaa tiedostosta. Tämä menettely ei aina vastaa erikoisimpien sovellusten tarpeita. Tällöin on sovelluksen päästävä kiinni suoraan raakoihin paketteihin ilman käyttöjärjestelmän väliintuloa.

### 2.1.1 Unixin kaltaisissa ympäristöissä

Libpcap on Unix-ympäristöissä ratkaisu riippumaton rajapinta käyttäjätason pakettikaappaukseen. Tämä mahdollistaa viitekehyksen verkkoliikenteen monitorointiin alhaisella tasolla. Libpcap-kirjastoa käytetään verkkoliikenteen nappaamiseen verkkoliitynnästä, joka on käyttöjärjestelmälle annettu. Tämä mahdollistaa verkkopakettien tallentamisen tarkastelua varten käyttäjätason prosessissa. Useimmissa Unixin kaltaisissa käyttöjärjestelmissä tämä kirjasto yleisesti on käytettävissä. Tätä kirjastoa hyödyntää esimerkiksi tilastollinen verkkoliikenne keräys-, tietoturva-, monitorointi- ja virheenjäljennösohjelmistot. [1.]

Kirjaston avulla voidaan määritellä paketit, jotka suodatetaan koko verkkoliikenteestä pakettikaappaukseen. Kirjaston avulla ei pystytä yhdistämään, mitä verkkopaketteja yksittäinen prosessi lähettää ja vastaanottaa. [2.]

Pcap tulee sanoista Packet Capture. Pcap-kirjasto on korkeatason rajapinta pakettikaappaus järjestelmille. Tämä mahdollistaa eri ohjelmointikielillä verkkoliikenteen tarkastelun. Näin voit itse ohjelmoida vaikka Python-ohjelmointikielillä pakettikaappaussovelluksen. Pcap toimii suoraan C- ja C++-kielillä, mutta muilla ohjelmointikielillä pääsy pcap-rajapintaan tarvitaan kiertofunktiota (engl. wrapper function).

### 2.1.2 Windows-ympäristössä

Microsoft Windows -ympäristössä käytetään WinPcap-työkalua verkkoliikenteen kaappaamiseen. WinPcap on Windows-versio avoimen lähdekoodin libpcap-kirjastosta, joka sieppaa ja analysoi verkkopaketteja Win32-alustalla.

Tunnetuimmat sovellukset Windows-ympäristössä, jotka käyttävät WinPcap-työkalua ovat Wireshark, Zenmap sekä WinDump. Wireshark on hyvin laajasti tunnettu vapaa ja avoimen lähdekoodin pakettikaappaus sekä analysointiohjelma. WinDump on kehitetty Windows-ympäristöön Linux-ohjelman Tcpdumpin pohjalta. Zenmap on GUI kuuluksaan Nmap-ohjelmaan, jota käytetään verkkolaitteiden löytämiseen sekä tietoturvakartoitukseen.

## 2.2 Pcap-tiedostomuoto

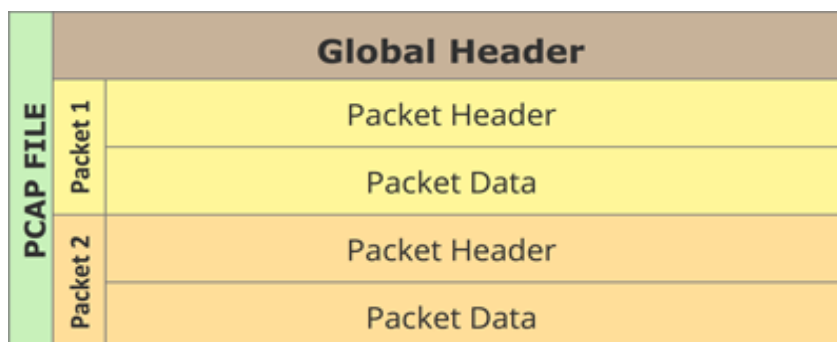
Pcap-tiedostomuoto on yleinen käytössä oleva verkkopakettien/datan tallennustiedostomuoto. Tcpdump, Snort ja monta muuta verkkotyökalua käyttävät tätä tiedostomuotoa verkkoliikennekaappauksessa oletuksena. Tästä formaatista on liikkeellä muokattuja versioita, mutta seuraavaksi tutustutaan yleisesti käytettyyn 2.4-versioon pcap-tiedostomuodosta. [3.]

### 2.2.1 Yleinen otsake

Ensiksi pcap-tiedostoon lisätään yleinen otsakeosio. Tämän otsakkeen kooksi on määriteltä 24 tavua. Tämä osio kirjoitetaan pcap-tiedostoon vain kerran. Ensimmäiset neljä tavua määrittelevät tiedoston tyypin. Kuvasta 1 voidaan nähdä pcap-tiedoston neljä ensimmäistä tavua (0xd4c3b2a1). Tällä nelitavuisella heksadesimaaliarvolla voidaan tunnistaa pcap-tiedosto. Tätä arvoa kutsutaan myös englanninkielisellä nimellä magic number. Näitä arvoja on varattu neljä kappaletta erilaisille pcap-tiedostomuodoille.

```
d4 c3 b2 a1 02 00 04 00 00 00 00 00 00 00 00 00
ff ff 00 00 01 00 00 00 47 02 1a 57 55 b1 00 00
```

Kuva 1. Pcap-tiedoston ensimmäiset 24 tavua on korostettu valkoisella pohjalla.



Kuva 2. Pcap-tiedoston koostumus.

Seuraavat kahdeksan tavua määrittelevät pcap-tiedostomuodon version. Näistä neljä ensimmäistä tavua määrittävät tiedoston pääversion ja neljä seuraavaa alaversion. Tämän jälkeen otsakkeesta löytyy aikavyöhykkeen oikaisu sekunteina GMT ja paikallisen aikavyöhykkeen väliltä ja aikaleimojen tarkkuus. Käytännössä aikaleimat on sijoit-

tettu GMT-aikajärjestelmään, joten tämän kentän arvo on yleensä 0 ja aikaleimojen tarkkuudeksi ilmoitetaan käytännössä 0.

Kaapattujen pakettien enimmäiskoko ilmoitetaan nelitavuisena heksadesimaaliluvulla. Pakettikaappausohjelmat Wireshark sekä Tcpdump asettavat oletuksena enimmäiskooksi 65535. Viimeiset yleisen otsakkeen neljä tavua koostuvat linkkitason otsakkeen tyypistä. Tämä määrittää linkkitason protokollan, joka on kuvassa 1 Ethernet. Linkkitason protokolla voisi olla myös esimerkiksi Frame Relay, ATM, 802.11, PPP tai Token Ring.

Yleisen otsakkeen jälkeen pcap-tiedostosta löytyy tietty määrä paketin otsake ja data-pareja. Näitä pareja tarkastellaan seuraavissa luvuissa.[3.]

## 2.2.2 Paketin otsake

Jokaista verkkoliikenteestä kaapattua pakettia kohden on pakettiotsake tallennettu pcap-tiedostoon. Paketin otsakkeen tarkoitus on näyttää paketin kaappaus hetki pakettikaappauksen aloittamisesta lähtien sekä varsinaisen paketin koko. Paketin otsakkeen neljä ensimmäistä tavua on aika sekunteina, joka on kulunut vuoden 1970 alusta lähtien paketin kaappaukseen. Kuvan 3 tapauksessa heksadesimaaliluku 0x5721fae6 kääntyy desimaaliluvuksi 1461844710. Kun desimaaliluku syötetään aika muuntimeen, näyttää pakettikaappaus kuvan 3 mukaisen ajan. Seuraavat neljä tavua otsakkeesta tarkentavat tämän ajan millisekunteihin.

```

ff ff 00 00 71 00 00 00 e6 fa 21 57 da bd 01 00
58 01 00 00 58 01 00 00 00 01 00 01 00 06 08 00
27 33 da 8d 00 00 08 00 45 10 01 48 00 00 00 00

```

1461844710    Timestamp to Human date    reset

GMT: Thu, 28 Apr 2016 11:58:30 GMT  
Your time zone: 4/28/2016, 2:58:30 PM GMT+3:00 DST

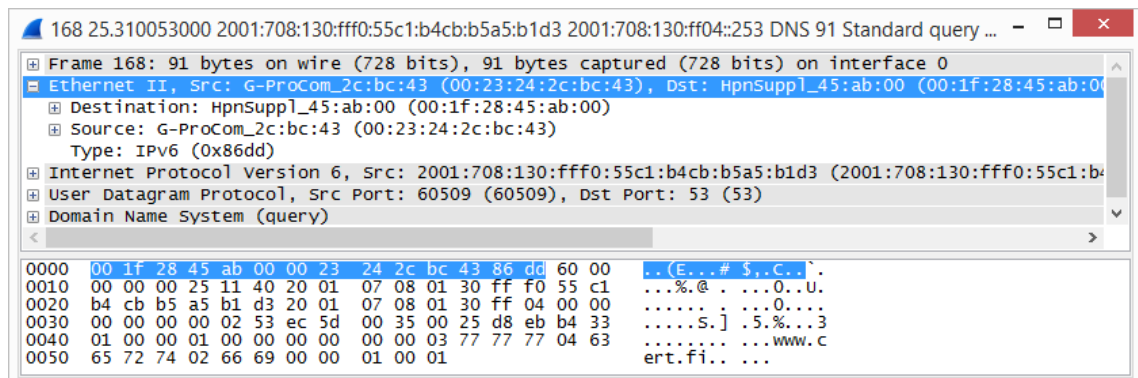
Kuva 3. Paketin otsake 16 tavua korostettu valkoisella pohjalla.

Tiedostoon kaapatun paketin koko tavuissa on ilmoitettu nelitavuisena. Tämä luku ei pitäisi olla koskaan suurempi kuin oikea verkosta napattu paketin koko tai yleisotsak-

keen kaapattujen pakettien enimmäiskoko. Neljäs kenttä paketin otsakkeessa on neljä tavua pitkä ja sisältää paketin koon, joka kaapattiin suoraan verkosta. Nämä kaksi viimeistä nelitavuista lukua ovat samat kuvassa 3, sillä verkosta kaapattu paketti on pienempi kuin yleisotsakkeessa määritelty paketin enimmäiskoko. Yleisotsakkeen paketin enimmäiskoon ylittyessä tiedostoon kaapattupakettikoko olisi maksimipaketin pituus ja paketin oikea koko olisi yhtä suuri kuin verkosta napattu paketin koko. [3.]

### 2.2.3 Paketin data

Itse paketti otsakkeen jälkeen tulee varsinainen data, mikä kulkee verkkoliikenteessä. Alkaen verkkoliikenteen alhaiselta tasolta huomaamme, että ensimmäiset tavut muodostavat Ethernet-kehiksen. Tähän kehykseen kuuluu kohde ja lähde MAC-osoitteet, joista huomataan paketin suunta linkkitasolla. Kuvasta 4 voimme nähdä, että paketin data alkaa kohde-MAC-osoitteella, jota seuraa lähde MAC-osoite. Tämän lisäksi viimeisenä Ethernet-kehyksessä on EtherType-kenttä. EtherType on 4-tavuinen luku, joka osoittaa Ethernet-kehikseen tiivistetyn protokollan. Tässä tapauksessa se osoitetaan heksadesimaaliluvulla 0x86dd, joka viittaa IPv6-protokollaan.



Kuva 4. Wireshark-pakettikaappaus DNS-kyselystä.

Kuvassa 4 on Wireshark-pakettikaappaus ohjelmasta kuva, jossa on yksi kaapattu paketti verkkoliikenteestä. Pakettikaappauksesta voimme päätellä, että korkealla tasolla kyseessä on DNS-kysely. Wiresharkin hexdumpista, eli binäärinäkymästä, voimme päätellä, mihin toimialue nimeen tarvitsemme DNS-vastauksen. Tullessamme alaspäin protokollapinoa kuljetuskerroksella voimme havaita portin, johon kyseinen DNS-kysely on kohdistettu. Tässä tapauksessa kysely kohdistuu kyselyn vastaanottavaan porttiin

53, joka on standardiportti nimipalvelu kyselyjen vastaanottamiseen. Verkkokerroksella huomaamme kohde IPv6-osoitteen, johon nimipalvelukysely on osoitettu.

### 2.3 Pakettikaappausohjelmat

Pakettikaappausohjelmat voidaan luokitella moneen eri ryhmään. Luokittelu tapahtuu ensimmäiseksi alustan mukaan, jossa pakettikaappaus tapahtuu. Jaottelu tehdään pääosin Windows- ja Unix-kaltaisten järjestelmien välillä. Jo aikaisemmin on käsitelty toimintatapoja, jotka erottavat nämä alustat toisistaan.

Näihin alustoihin on tehty erinomaisia ohjelmia, jotka mahdollistavat pakettien tehokkaan analysoinnin ja seulomisen erilaisten suodattimien avulla. Käyttöjärjestelmän valitsemisen jälkeen on tehtävä päätös graafisen ja konsoli-pohjaisen pakettikaappausohjelman väliltä. Molemmissa ohjelmissa on hyvät ja huonot puolensa. Graafinen käyttöliittymä näyttää yleensä reaaliajassa pakettivirran, josta voidaan nopeasti tehdä johtopäätöksiä tai päätellä liikenteen laatua. Tämän lisäksi graafisen käyttöliittymä helpottaa huomattavasti verkkoliikenteen analysointia, kun kyseessä on paljon erilaista liikennettä. Pakettikaappauksessa graafisen käyttöliittymän huonona puolena voidaan pitää resurssien tuhlausta.

Varsinaiseen pakettien kaappaukseen ei tarvitse ohjelmaa, joka näyttää päätteen näytölle reaaliaikaisesti verkkoliikennettä. Näin ollen paketti kaappauksessa käytetään yleensä komentoriviltä ajettavaa ohjelmaa. Komentoriviltä käynnistetään pakettikaappausohjelma halutuilla säätöarvoilla. Yleisimmät säätöarvot ovat verkkoliitännän ja verkkoliikenteen suodattimien valitseminen. Tällöin pakettikaappaus tapahtuu mahdollisimman alhaisella kustannuksella ja resursseja jää muuhun työskentelyyn.

Pakettikaappaus lopetetaan käyttäjän toimesta, ennalta määritetyn ajan kuluttua tai kaappaavan laitteen muistin loputtua. Tulokset tallennetaan pakettikaappaustiedostoon analysointia varten. Pakettien analysointiin graafinen käyttöliittymä on huomattavasti tehokkaampi kuin komentoriviltä analysointiin tarkoitetut sovellukset.

### 2.3.1 Wireshark

Wireshark-pakettikaappausohjelma sai alkunsa Ethernet-projektista. Vuonna 1997 Gerald Combs tarvitsi työkalun verkkoliikenne ongelmien jäljittämiseen ja halusi oppia lisää verkkoliikenteestä. Hän aloitti Ethernet-projektin, joka ensimmäisen kerran julkaistiin versiona 0.2.0 heinäkuussa 1998. Tästä eteenpäin projektiin on ollut osallisena huomattava määrä henkilöitä. Nämä henkilöt ovat tukeneet projektia tuomalla uusia verkkoprotokolla tukia ja tarjoamalla omaa koodia projektiin. Vuonna 2006 Combs joutui luopumaan Ethernet-tuotemerkistä, ja projekti jatkui nimellä Wireshark. Viimeisin vakaa versio Wiresharkista on 2.0.1, joka on julkaistu 29. joulukuuta 2015.[4; 5.]

Wireshark on ilmainen ja vapaan lähdekoodin paketti analysoija. Sitä käytetään tietoverkko ongelmien vian selvitykseen, analysointiin sekä opetustarkoitukseen. Avoimen lähdekoodin lisenssi sallii verkkoasiantuntijoille lisätä parannuksia ohjelmaan omien tarkoitusten mukaan. Wireshark-paketti analysoija on niin monipuolinen, että sitä käyttävät lähes poikkeuksetta kaikki verkkoliikenteen kanssa työskentelevät asiantuntijat. Järjestelmäasiantuntijat käyttävät sitä verkko-ongelmien selvittämiseen. Tietoturva-alan tekijät analysoivat verkkoliikennettä haavoittuvuuksien näkökulmasta. Ohjelmistokehittäjät käyttävät sitä ohjelmistojen testaamiseen. Erityisesti opetus käyttöön Wireshark on poikkeuksellinen. Sen helppokäyttöisyys ja visuaaliset apuvälineet tarjoavat ideaalisen ympäristön oppimiseen.[6.]

### 2.3.2 Tcpcap

Tcpcap on Unix-kaltaisissa järjestelmissä toimiva pakettikaappaus työkalu. Tämä on tarkoitettu toimimaan komentoriviltä suoraan, joka mahdollistaa sen nopean ja tehokkaan käyttöönoton. Tcpcap on vapaan lähdekoodin ohjelmisto, jota pystyy jakamaan BSD-lisenssin alla. [7.]

Tcpcap käyttää libpcap-kirjastoa pakettikaappaukseen. Yleisimmät Linux-pohjaiset jakelupaketit asentavat käyttöjärjestelmän yhteydessä Tcpcap-ohjelmiston. Tämä myös riippuu käyttöjärjestelmän asennuksen yhteydessä valittavista asennusvaihtoehdoista. [8.]

Pakettikaappauksen pystyy kirjoittamaan suoraan vakiotulosteeseen tai tiedostoon Tcpcapin avulla. Vakiotulosteeseen kannattaa ainoastaan kirjoittaa, kun on valinnut

riittävät suodattimet pakettikaappauksen suorittamiseen. Ilman suodattimia tuloste täytyy nopeasti epäoleellisesta liikenteestä tuotantoympäristössä. [8.]

### **3 Tunkeilijan havaitsemisjärjestelmät**

Tunkeilijan havaitsemisjärjestelmillä on tärkeä merkitys toimialueen puolustuksessa verkkouhilta. Tämä antaa mahdollisuuden analysoida verkossa tapahtuvaa haitallista liikennettä sekä tarkkailla verkolle epätyypillisiä piirteitä. IDS luotiin alun perin havaitsemaan haavoittuvuuksien hyväksikäyttämistä kohdesovelluksia ja tietokoneita vastaan. [9.]

Tunkeilijan havaitsemisjärjestelmät voidaan jakaa kone kohtaisiin ja verkko kohtaisiin järjestelmiin. Konekohtainen järjestelmä (HIDS) suojaa tiettyä laitetta havaitsemalla laitteen sisällä olevia muutoksia. Tietokoneen sisällä aktiivisesti toimivaa tietoturvaohjelmistoa voidaan pitää konekohtaisena tunkeilijan havaitsemisjärjestelmänä. [9.]

Verkkokohtainen järjestelmä (NIDS) on suunniteltu havaitsemaan uhkia verkkotasolla tiettyä kohdejärjestelmää tai kone ryhmää vastaan. NIDS tarvitsee vain havaita mahdollisia uhkia, joten tämä sijoitetaan verkossa sivuun hyötyliikenteen tieltä. Tämä tarkoittaa sitä, että NIDS ei ole sijoitettu suoraan tiedon lähettäjän ja vastaanottajan väliin. NIDS-ratkaisut hyödyntävät yleensä portti peilaustekniikoita verkkoliikenteen vastaanottamiseen. [9.]

Tunkeilijan havaitsemisjärjestelmät jaotellaan havaitsemistekniikan mukaan kahteen eri ryhmään. Tilastollista havaitsemistekniikkaa käyttäen IDS vertaa verkkoliikennettä ennalta määritellyyn sallittuun liikenteeseen. Sallittu verkkoliikenne on siis otos verkolle tyypillisestä toiminnasta. Toinen menetelmä on tunnistepohjainen havaitseminen. Tässä verkkoliikennettä verrataan tietokantaan, jossa on tunnisteita ja piirteitä jo ennalta tunnetuista haitallisista uhkista. Tämä toteutustapa on samanlainen nykyaikaisiin tietoturvaohjelmiin, jotka havaitsevat haittaohjelmia. [10; 11.]



### 3.1 Arkkitehtuuri ja toteutustavat

IDS-arkkitehtuuri on tärkein osa toteutettaessa tunkeilijan havaitsemisjärjestelmää. Järjestelmän on oltava tehokas ja koordinoitu. Tästä johtuen jokaisella IDS-komponentilla, kuten koneella, laitteella tai prosessilla pitää olla ennalta määritelty tehtävä. Tehtävä pitää olla selvä, jotta tietoa voidaan tehokkaasti prosessoida ja analysoida. Arkkitehtuurin rooli korostuu etenkin, kun toteutetaan liiketoiminnalle tai yritykselle sopivaa toteutusta. Toteutuksen pitää tällöin sopia juuri tämän organisaation tarpeisiin ja toimintaan. Heikosti toteutettu ja suunniteltu arkkitehtuuri organisaatiolle voi luoda häiriöitä verkkoon, jotka näkyvät linkkien hidastumisena, reagoitakyvyn heikentymisenä tunkeutumisiin sekä mahdollisesti vaikuttavat negatiivisesti tiedon saatavuuteen.

Tunkeutumisen havaitsemisarkkitehtuurit voidaan jaotella eri kerroksiin niiden toteutustavan ja ympäristön tarpeiden mukaan. Tässä kerrosmallissa näytetään toteutustavat pienemmille ja suuremmille IDS-ratkaisuille.

#### 3.1.1 Yksitasoinen

Yksitasoinen rakenne on yksinkertaisin tapa toteuttaa IDS. Tämä rakenne sopii erityisesti SOHO-ratkaisuihin. Tässä mallissa on vain yksi komponentti, joka kerää ja prosessoi sille syötetyn datan. Yksinkertaisimmillaan yksitasoisen mallin esimerkki on konekohtainen tunkeutumisen havaitsemistyökalu, joka vastaanottaa järjestelmästä tulevat lokitiedostot ja vertaa niitä tunnettuihin tunkeutumismalleihin. [12.]

Yksitasoisen IDS-mallin rakentamisen hyötyjä ovat sen suhteellisen helppo asennus ja alhaiset kustannukset. IDS-ratkaisun ollessa yksitasoinen sen ylläpitäminen on huomattavasti helpompaa kuin monesta eri IDS-komponentista huolehtiminen samanaikaisesti. Tämän lisäksi on saatavilla paljon avoimeen lähdekoodin perustuvia IDS-työkaluja. Näiden ansiosta yksitasoisen IDS on käytännöllinen ratkaisu pieniin ympäristöihin.[12.]

Suuremmissa ympäristöissä yksitasoisten IDS-ratkaisujen haitta saattaa olla hyötyjä suurempi. Analysoitavan tiedon määrän ollessa suuri yksitasoinen IDS ylikuormittuu, ja tunkeutumisen havaitseminen vaikeutuu. Tunkeutumisen havaitseminen vaikeutuu, koska järjestelmään tulee enemmän vääriä hälytyksiä ja haitallinen liikenne hukkuu taustakohinaan. Yksitasoisen IDS-ratkaisun ylikuormittumista voidaan estää pienentä-

mällä analysoitavan datan määrä. Tämä käytännössä tarkoittaa sitä, että kavennetaan tarkasteltavaa pinta-alaa järjestelmässä. NIDS-ratkaisussa tämä voi tarkoittaa sitä, että koko liikenteen analysoinnin sijasta kohdennetaan tarkastelu vain tietyn tyyppiseen liikenteeseen. [12.]

### 3.1.2 Monitasoinen

Monitasoinen IDS-ratkaisu koostuu nimensä mukaan monesta eri tasosta ja komponentista. Komponentit pystyvät kommunikoimaan keskenään sekä rinnakkain analysoimaan dataa tehokkaammin. IDS-järjestelmä, joka käyttää tätä mallia, koostuu yleensä kolmesta pääkomponentista. [12.]

NIDS-sensorille syötetään raakoja paketteja suoraan verkkoliikenteestä. Tunkeilijan havaitsemisjärjestelmissä on aina sensori, jonka vastuulla on kerätä tietoa verkkoliitännöistä, järjestelmälöki tiedostoista ja palomuurista. Sensorin vastuulla on kerätä tästä tiedosta oleellinen data ja lähettää se eteenpäin analysoitavaksi agentille. Sensorit voivat olla verkkopohjaisia tai konekohtaisia.[11.]

Verkkopohjaiset sensorit ovat ohjelmistoja tai rautatason ratkaisuja, jotka keräävät tietoa verkkoliikennepaketeista. Paketteja kaapattaessa verkkoliikenteestä tietoa tulee sensorille huomattava määrä. Paketteja kerätään sensorin sijoituksesta ja suodattimista riippuen. Verkkoliikenteen ruuhkautuessa sensori voi ylikuormittua ja olla tällöin olla huomaamatta mahdollisia haitallisia uhkia. Laajasti käytetyt työkalut sensoreissa verkkoliikenteen kaappaamiseen ovat tcpdump ja libpcap. Tunnetuin avoimen lähdekoodin tunkeutumisen havaitsemisjärjestelmä Snort käyttää libpcap-kirjastoa. [12; 13.]

Konekohtaiset sensorit pyrkivät ottamaan paketteja verkkoliitynnälle tulevasta liikenteestä. Eroavaisuus verkkopohjaiseen sensoriin on sen toimintatilassa. Konekohtainen sensori kaappaa vain ja ainoastaan sille koneelle tarkoitettun verkkoliikenteen. Sensorin ollessa konekohtaisessa tilassa verkkoliityntä tarkistaa MAC-osoitteen perusteella sille tarkoitettut kehykset. Näin ollen verkkoliityntä pudottaa kaikki sille koneelle kuulumattomat kehykset. [12.]

Sensorien, jotka on asetettu kaappaamaan kaiken verkkoliikenteen, sijoitus verkossa voidaan toteuttaa kolmella eri tavalla. Ensimmäinen tapa on sijoittaa sensori verkon reunalla olevan palomuurin ulkopuolelle. Tällöin voidaan tallentaa tietoa hyökkäyksistä,

jotka ovat alkuperäisin Internetistä. Tässä toteutustavassa on huomioitava, että sensori on täysin avoin Internetistä tuleville hyökkäyksille. Toinen tapa on sijoittaa sensori verkon sisäpuolelle. Verkon sisäpuolella sensori on ulkoisen palomuurin ja sisäverkon välissä, jolloin sensori on suojaisemmassa paikassa kuin verkon ulkopuolella. Verkon sisällä oleva sensori tallentaa tietoa hyökkäyksistä, jotka ovat alkuperäisin sisäverkosta. Tällä sensorin sijoituksella tarkastellaan myös palomuurin läpäisseitä hyökkäyksiä Internetistä. Kolmas sensorien toteutustapa verkkoon on yhdistää äsken mainitut toteutukset. Tässä toteutus tavassa sijoitetaan siis sensori ulkoisen palomuurin molemmille puolille. Tämä mahdollistaa täydellisen tarkastelun verkkoon, kuten myös sisäverkosta, kohdistuvista hyökkäyksistä. Korkean tietoturvatason omaavissa tietoverkoissa pyritään toteuttamaan ratkaisu, jossa sensorit on sijoitettu palomuurin molemmille puolille. Sensorien sijoitukseen ja toteutukseen liittyy myös haasteita, jotka on otettava huomioon. [12.]

Pakettikaappaussovellukset tarvitsevat jonkun asteiset ylläpitäjätason käyttöoikeudet. On otettava huomioon mahdollisuus, jossa sensoriin voidaan murtautua. Onnistunut hyökkäys sensoria vastaan voi tällöin pahimmillaan antaa tunkeutujalle suoran pääsyn käyttämään järjestelmää ylläpitäjän oikeuksilla. Murtautumisen vaikutuksia on pienennettävä niin pieniksi kuin mahdollista. Tämä vaatii käyttäjä tason oikeuksien säätelyä sensorissa. Sensoriin luodun käyttäjän, joka hoitaa pakettikaappausmekanismien, käyttöoikeuksia itse järjestelmään ja tiedostoihin on rajoitettava. Hyökkääjän on tästä huolimatta mahdollista käyttää käyttöoikeuksien ohittamistekniikoita koko järjestelmän halluunottamiseen. [12.]

Koko tietoliikenteen kaappaaminen tarkastelua varten kuluttaa huomattavan osan levy-pinnasta. Tästä syystä muistikapasiteetin hallinta on erittäin tärkeää IDS-ratkaisuissa. Levytilaa on tarkkailtava sen kapasiteetin ja luku- sekä kirjoitusnopeuden kannalta. Kovalevyllä olevaa kaapattua verkkoliikennettä on varastoitava ja puhdistettava levytilan täyttymisen mukaan. Nämä piirteet ovat pakollisia tunkeutumisen havaitsemisjärjestelmissä. [12.]

Sensorille tuleva liikenne voi jossain tapauksissa ylittää sensorin maksimiprosessointi-kapasiteetin. Tämä johtaa pakettien tiputtamiseen verkkoliitynnässä. Pahimmillaan sensorin ruuhkautuminen voi johtaa sen kaatumiseen ja tästä seuraa sensorin uudelleen käynnistäminen takaisin operointitilaan. [12.]

Salattu verkkoliikenne on otettava huomioon sensorin sijoittamisen kannalta. Tietoverkkojen välillä voi olla staattinen salattu tunneli, joka mahdollistaa tiedon turvallisen lähettämisen Internetin ylitse. Käytännöllinen ratkaisu olisi sijoittaa sensori paikkaan, jossa tieto on selkokielisenä. Hyvien käytäntöjen mukaista on salata sensorin ja muiden tunkeutumisen havaitsemisjärjestelmien välinen tietoliikenne. Tunkeutujan kaapattessa IDS-komponenttien välinen liikenne on havaitsemisjärjestelmä käytännössä hyödytön. Helpoin menettely on asettaa IDS-komponenttien välille autentikointi sekä liikenteen salaus. [12.]

Agenttia kutsutaan myös verkkoliikenteen tutkijaksi. Päätehtävä on tutkia sensorilta lähetettyä tietoa. Agentille on määritetty havaitsemispolitiikka liikenteelle, jota tämän tehtävä on tutkia. Tunkeutumisen havaitsemisjärjestelmässä agentit ovat yksittäisiä toisista agentti prosesseista riippumattomia. Tyypillisesti agentti on määritelty tutkimaan tiettyä verkkoliikennettä. Tarkastelu voi olla jaoteltu siten, että tietyt prosessit tarkastelevat TCP ja toiset muuta verkkoliikennettä TCP/IP-verkkokerrokselta. Nämä tehtävät voivat olla vielä syvemmin jaoteltu TCP-agentti prosessien välille. Yhdelle agentille voi olla määritelty HTTP-liikenteen tarkastelu ja toiselle muu TCP-liikenne. Agentti ollessa muista agentti prosesseista riippumaton, yhden prosessin kaatuminen ei vaikuta muiden agenttien toimintaan. Tästä syystä agenttiprosesseja on helppo lisätä tai poistaa tunkeutumisen havaitsemisjärjestelmän mukaan. [12.]

Päätehtävä isäntäpalvelimella on hallita koko tunkeilijan havaitsemisjärjestelmää. Agentit havaitsevat tunkeutumisen ja lähettävät tästä tiedon eteenpäin isäntäpalvelimelle. Tämä palvelin toimii keskitettynä hallinta järjestelmänä, johon kaikki agentit ottavat yhteyttä tunkeutumisen havaittuaan. Isäntäpalvelimella on tunkeutumisen havaitsemisjärjestelmään jonkin tason käyttöliittymä. Tämä käyttöliittymä voi olla graafinen tai pelkkä komentorivipohjainen. Käyttöliittymän avulla järjestelmänvalvoja tai käyttäjä voi tarkastella ja muuttaa järjestelmän toimintaa. [12.]

Aikaisemmin määriteltyjen sääntöjen mukaan isäntäpalvelin tekee hälytyksen tunkeutumisesta. Tunkeutumisen havaitsemisjärjestelmässä agentit pystyvät tekemään hälytyksiä tunkeutumisesta, mutta on käytännöllisempää, jos ilmoitukset mahdollisesta tunkeutumisesta tulisivat keskitetystä paikasta. Ennalta määritelty ilmoitus tunkeutumisesta voi laukaista kirjoituksen tapahtumasta lokitiedostoon tai ajaa sovelluksen, taikka skripti-tiedoston. Näiden lisäksi havaitsemisjärjestelmän on mahdollista lähettää SMS- tai s-postiviesti ennalta määritetylle kohdejoukolle. [12.]

Isäntäpalvelin tehtävä on koota tunkeutumisista tapahtumaraportti käyttöliittymään, joka on isäntäpalvelimelle määritelty. Näin järjestelmänvalvoja voi tarkastella järjestelmän tuottamia raportteja hälytyksistä. Isäntäpalvelimen kerätessä tietoa eri agenteilta on tämän selvitettävä tapahtumien välisiä riippuvuussuhteita. Tämä tarkoittaa tapahtumien yhdistelemistä toisiinsa. Yhdisteleminen tapahtuu esimerkiksi vertaamalla tapahtumien yhteistä lähdettä tai kohdetta, ja tällöin isäntäpalvelin päättää, kuuluvatko hälytykset samaan tietoturvatapahtumaan. [12.]

## 3.2 Snort

Snort on verkkotasolla toimiva tunkeilijan esto- ja havaitsemisjärjestelmä. Se on kevyt avoimeen lähdekoodiin perustuva ohjelma. Snortin voi asentaa melkein mihin tahansa nykyaikaiseen tietokone arkkitehtuuriin ja käyttöjärjestelmään. Yleisimmin Snorttia ajetaan Linux/Unix alustalla, mutta käytettävissä on myös Windows-alustalle versio Snortista. [13.]

Tämä verkkotasolla toimiva järjestelmä on luokiteltu parhaimpien IDS/IPS-ratkaisujen joukkoon. Snort on laajasti käytössä erilaisissa yrityksissä ja korkean tietoturvatason omaavissa järjestelmissä. Suurena etuna pidetään sen avoimen lähdekoodin ohjelmistoa. [14.]

Käytännössä Snort toimii etsimällä paketteja verkkoliikenteestä ja pyrkii vertaamaan näitä ennalta määritettyihin sääntöihin. Snort voidaan asettaa toimimaan kolmeen eri toiminta tilaan. Nämä toimintatilat ovat tarkastelu-, pakettilokitus- ja tunkeutumisen havaitsemistila. [13.]

### 3.2.1 Tarkastelutila

Tässä tilassa Snort yksinkertaisesti katsoo verkkoliikennettä ja ohjaa tämän näytölle. Komentoriviltä pakettien tarkastelu voi olla haastavaa, mutta Snortilla pakettien lukeminen on tehty yllättävän käyttäjä ystävälliseksi. Tarkastelutilan lopuksi Snort antaa verkkoliikenteestä yhteenvedon. [13.]

Snortilla pakettien tarkastelutila voidaan aloittaa Ubuntu-käyttöjärjestelmässä syöttämällä komentoriville seuraava komento:

- `sudo snort -vde -i eth0`

Komentorivi parametrilla `-v` Snort tietää aloittaa tarkastelutilan. Parametreilla `d` ja `e` tuostetaan verkkoliikenteessä olevan paketin hyötydata sekä siirtoyhteyserroksen Ethernet-otsakkeen.

### 3.2.2 Pakettilokitustila

Käsiteltäessä pieniä määriä verkkoliikennettä lyhyessä ajassa on tarkastelutila hyvä vaihtoehto. Verkkoliikenteen kasvaessa haluamme tallentaa verkkoliikenteen levyille, josta voidaan tarkastella sitä lähemmin. Pakettilokitustilassa verkkopaketit tallennetaan pysyvään muistiin koneella. [13.]

### 3.2.3 Tunkeutumisen havaitsemistila

Koneelle, jossa Snort IDS toimii, ohjataan tarkasteltavaa verkkoliikennettä. Tässä tilassa verkkoliikennettä verrataan sääntöihin, jotka on asetettu Snort konfiguraatiodostoon. Verkkoliikennettä voidaan myös kirjoittaa levyille riippuen järjestelmälle asetetuista tavoitteista. Tunkeutumisen havaitsemistilaa käsitellään tarkemmin seuraavassa pääluvussa, jossa asennetaan ja konfiguroidaan tunkeilijan havaitsemisjärjestelmä. [13.]

### 3.2.4 Säännöt

Säännöt määritellään konfiguraatiodostoon, jossa yksittäinen sääntö määritellään yleensä yhdelle riville. Yksittäinen sääntö koostuu kahdesta loogisesta alueesta. Loogiset alueet ovat säännön otsake- ja tarkennuskentät. Säännön otsake-kentässä määritellään säännön toimenpide, protokolla, lähde ja kohde-IP-osoitteet sekä aliverkonpeite ja porttitiedot. Säännön tarkennuskenttä määrittelee hälytystoimenpiteet, tarkastettavat kentät paketista sekä määriteltävä, pitääkö sääntötoimenpiteet toteuttaa vai ei.

**alert tcp \$EXTERNAL\_NET any -> 192.168.0.0/24 80 (msg: "Sample alert");**

Kuva 5. Yksinkertainen Snort-sääntö.

Kuvassa on yksinkertainen Snort-sääntö. Säännön tavoitteena on aiheuttaa hälytys, kun ulkoverkosta otetaan yhteyttä sisäverkon osoite avaruuteen. Tässä tapauksessa on tarkennettu, että verkkoliikenteen pitää käyttää TCP-protokollaa sekä kohdistettu sisäverkon osoitteen porttiin 80. Säännön tarkennuskenttä on suluissa. Siinä on määriteltä viesti, joka lähetetään hälytyksen yhteydessä.

```
alert tcp $EXTERNAL_NET any -> 192.168.0.0/24 80 (msg: "Sample alert"; \
content: "http|3a|//www.sampledomain.com/admin.php"; nocase; \
offset:12; classtype: web-application-activity; \
reference:url,http://www.sampledomain.com/threats/2016.html; \
sid:1000001, rev:1;)
```

Kuva 6. Snort-sääntö aiheuttaa hälytyksen huomattaessaan tietyn tyyppistä HTTP-liikennettä.

Kuvassa 6 on lisätty edelliseen Snort-sääntöön tarkennuksia. Jatkettaessa Snort-sääntöä uudelle riville on lisättävä takakeno. Säännön tarkennuskenttään on lisätty "content:" määrittely, jossa on tarkennettu paketin sisällöstä löydyttävä merkkijono. Merkkijonossa määritellään erikseen kahden pystyviivan väliin ei-tulostettavat merkit sekä merkit, joita ei saa käyttää sisällön tarkistuksessa. Pystyviivojen välissä oleva luku on binääriluku, joka ilmoitetaan heksadesimaalina. Tässä tapauksessa heksadesimaaliluku "3a" tarkoittaa ascii-merkinä kaksoispistettä. Tarkennuskentässä "nocase" tarkoittaa, että paketin sisällöstä löydyttävä merkkijono ei ole merkkikokoriippuvainen. Seuraavaksi säännössä määritellään poikkeama (engl. offset), josta Snort alkaa etsimään merkkijonoa paketin aloituspisteeseen riippuen. Tällä tavoin paketista hypätään poikkeaman verran eteenpäin ja vähennetään prosessointiaikaa pakettia kohden. Kentässä luokkatyyppi (engl. classtype) luokitellaan hälytyksen vakavuusaste. Oletuksena Snortissa on neljä vakavuusluokkaa, joista "web-application-activity" luokitellaan toiseksi korkeimmaksi vakavuusasteella. Viittauskentässä (engl. reference) osoitetaan, mihin kyseinen hälytys viittaa. Viittauskentässä voisi olla viittaus esimerkiksi CVE-numeroon tai tietoturvaanlogiin, jossa on lisää tietoa kyseisestä haavoittuvuudesta tai tietoturvaongelmasta. Seuraava kenttä määrittelee säännölle yksilöllisen numeron. Snort dokumentoinnissa määritellään, että paikallisille säännöille on varattu SID-numerot miljoonasta eteenpäin. [14.]

### 3.3 Kaupalliset seuraavan sukupolven IPDS-ratkaisut

Nykyajan kaupalliset tunkeilijan havaitsemisjärjestelmät ovat yhdessä tunkeilijan estojärjestelmän kanssa. Tämä johtuu siitä, että IDS ja IPS tekevät jatkuvasti yhteistyötä havaitessa ja torjuessa viimeisimpiä uhkia. Kaupalliset ratkaisut tarjoavat yleensä tunkeilijan havaitsemis- ja estojärjestelmän täyden ylläpidon ja tunkeutumisiin reagoinnin. Tämä tarkoittaa IDS-järjestelmän ulkoistamista ulkoiselle yritykselle, jolla on resurssit hallita tietoturvajärjestelmiä. Käytössä on yleensä tietoturva-asiantuntijoista koostunut ryhmä, jonka tehtävä on pitää huolta järjestelmän ylläpidosta sekä olla valmiina vastaamaan viimeisimpiin tietoturvauhkiin. [15.]

#### 3.3.1 Verkkoympäristötietoisuus

Seuraavan sukupolven ratkaisut pyrkivät parantamaan tietoisuutta ympäristöstä, jossa toimivat, keräämällä tietoa verkossa olevien laitteiden tyypeistä. Keräämällä tietoa laitteista, jotka toimivat verkossa IDS-järjestelmän kanssa, IDS pyrkii määrittelemään verkolle tyypillistä liikennettä. IDS-järjestelmän tiedostaessa verkossa toimivat laitteet, tämä pystyy havaitsemaan verkossa liikkuvaa toimintaa, joka on alkuperäisin verkoon kuulumattomista laitteista. [15.]

#### 3.3.2 Sovellus ja käyttäjätietoisuus

Tavoitteena on pystyä määrittelemään verkossa käytettävät sovellukset ja ohjelmat. Näin voidaan luokitella verkossa toimivat ohjelmat sallittuihin ja estettyihin. Tällä menettelyllä pyritään älykkäämpään verkkoliikenteen tarkkailuun, jossa voidaan tehokkaammin tulkita verkkoliikennettä normaaliksi tai epänormaaliksi. [15.]

Tunkeutumisen havaitsemisjärjestelmä pyrkii tunnistamaan verkossa toimivia käyttäjiä. Tällä pyritään yhdistämään käyttäjä tiettyyn palveluun, sovellukseen tai IP-osoitteeseen. Näin on mahdollista luokitella ja säädellä yksittäisten käyttäjien ja käyttäjäryhmien oikeuksia käyttää eri sovelluksia ja palveluja. IDS-järjestelmä voi hakea verkon käyttäjistä tietoa keskitetyltä käyttäjähallinta palvelimelta, kuten Microsoft aktiivihaikemistosta tai LDAP-palvelimelta. [15.]



### 3.3.3 Laitekohtainen käyttäytyminen

Seuraavan sukupolven IPDS-ratkaisut seuraavat verkkolaitteiden toimintaa ja etsivät poikkeuksia normaaliverkkoliikenteeseen verrattuna. Normaalista verkkoliikenteestä poikkeavaa tietoa rinnastetaan verkon jatkuviin tai kaavamaisiin tapahtumiin. Verkkolaitteelle epätyypillisiä piirteitä voivat olla esimerkiksi hetkellinen korkea kaistan käyttö tai suorituskyvyn heikkeneminen. IPDS-järjestelmä kerää epätyypillisiä piirteitä ja ilmoittaa tai tekee järjestelmään muutoksia ennalta määriteltyjen sääntöjen perusteella. [15.]

### 3.3.4 Automatisointi

Tärkeä ominaisuus seuraavan sukupolven IPDS-järjestelmille on omien tunnisteiden säätämisen automatisointi. Järjestelmän on mahdollista säätää itse hälytystasoja sekä sallittua ja estettyä liikennettä pohjautuen verkosta kerättyyn tietoon ja ennalta määritettyihin sääntöihin. IPDS-järjestelmä havaitsee luotetun ohjelmiston käyttöönoton käyttäjäkoneella, jolloin järjestelmä osaa automaattisesti ottaa käyttöön tai poistaa käytöstä tunnisteita liittyen käyttöönotettavaan ohjelmistoon. [15.]

## 4 Snort-asennus ja konfigurointi

Asennettavaksi tunkeilijan havaitsemisjärjestelmäksi valittiin Snort. Snortin laaja dokumentointi, saatavuus ja käyttöönoton yksinkertaisuus tekevät siitä oivallisen valinnan IDS-järjestelmän testaukseen. Ohjelmisto on ladattu tähän päivään mennessä jo yli neljä miljoonaa kertaa ja yli puoli miljoonaa käyttäjää tekevät siitä yhden suosituimmista IDS-järjestelmistä. Tämän johdosta on joksenkin helppoa löytää ratkaisuja mahdollisiin ongelmatilanteisiin ja tietoa on helposti saatavilla. [13.]

Tunkeilijan havaitsemisjärjestelmä tullaan asentamaan Unix-pohjaisen käyttöjärjestelmän päälle. Unix-pohjaisen käyttöjärjestelmän etuna pidetään sen helposti säädettävää ympäristöä. Tällöin voidaan myös säästää resursseja ja parantamalla suorituskykyä minimoimalla koneessa pyörivien tarpeettomien palveluiden määrää. Snort päätettiin asentaa Ubuntu-käyttöjärjestelmään sen helppokäyttöisyyden sekä kattavan dokumentoinnin perusteella.

IDS-järjestelmän asennuksen jälkeen konfiguroidaan Snort-järjestelmä testausta varten. Konfigurointivaiheessa asetetaan järjestelmään viimeisimmät säännöt, joilla pyritään havaitsemaan mahdollisia uhkia verkossa. Konfiguroinnilla pyritään siihen, että järjestelmä on valmis toimimaan NIDS-järjestelmänä ja helpottamaan verkkotasolla lähiverkon turvallisuutta.

#### 4.1 Esivaatimusten asennus

Snortin esivaatimukset tullaan asentamaan suoraan pakettihallintakirjaston avulla ja osa käännetään lähdetiedostoista. Debian-käyttöjärjestelmissä on käytössä APT-työkalu, engl. Advanced Package Tool, jota käytetään pääasiassa pakettihallinnan helpottamiseen. Asennus tapahtuu suoraan komentoriviltä. Unix-käyttöjärjestelmissä komentorivi on tehokas työkalu, joka mahdollistaa esimerkiksi monen eri paketin asentamisen yhden komennon avulla. Snort on myös mahdollista asentaa kokonaan yhden skripti-tiedoston avulla, mutta tällöin on vaikeampi ratkaista mahdollisia vikatilanteita asennuksen yhteydessä. Esivaatimusten asennus suoritetaan virallisen Snort-dokumentoinnin mukaan. Asennukseen käytetyt komennot löytyvät liitteestä 1.

Ensiksi asennetaan työkalut, joilla voidaan koota lähdetiedostoista toimivia ohjelmia. Ohjelmien kokoamiseen ja rakentamiseen tarvittavat työkalut löytyvät Debian-käyttöjärjestelmiin build-essential-paketista. Ohjelmien kokoamiseen tarvittavan paketin jälkeen asennetaan riippuvuudet, joita Snort käyttää toimiakseen. Toimiakseen Snort tarvitsee libpcap-kirjaston, jonka avulla paketteja kaapataan suoraan verkkoliitännästä. PCRE-paketeista löytyy Perl-yhteensopivien säännöllisten lausekkeiden kirjastot. Tätä kirjastoa käytetään merkkijonojen vertailemiseen Perl-yhteensopivien säännöllisten lausekkeiden avulla.

Snort käyttää libdumbnet-pakettia rajapintana alhaisen tason verkkotoimintojen ja rutiinien käyttämiseen. Tämän rajapinnan avulla voidaan muuttaa koneen IP-osoitetta, ARP- ja reititystaulukkoa. Näiden lisäksi on mahdollista käsitellä palomuurin asetuksia sekä verkkoliitännällä suoraan raakojen IP-pakettien ja kehyksien siirtämistä. [16.]

DAQ-moduulia käytetään IP-pakettien syöttämiseen ja tulostamiseen Snort IDS -järjestelmään. Tiedonhankinta kirjasto, engl. Data Acquisition Library, korvaa suorat kutsut libpcap-funktioihin ja vähentää Snorttiin tehtävien muutosten määrää verkkota-

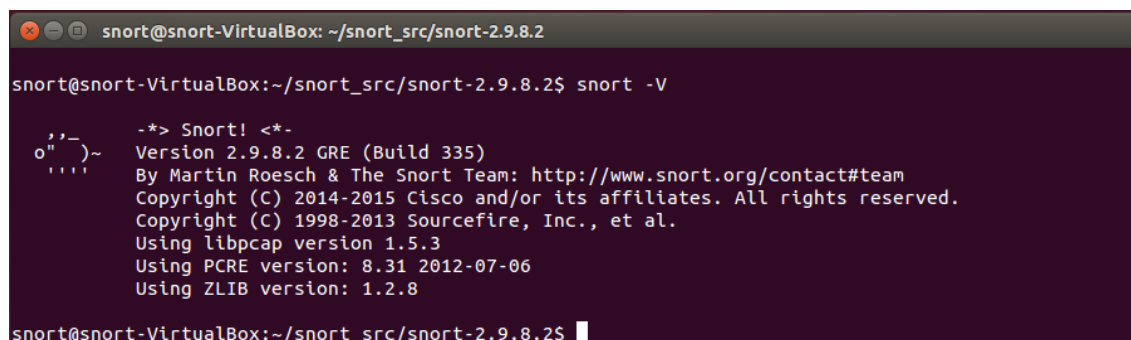
solla. DAQ asennetaan suoraan lähdetiedostosta, jolloin se pitää ensiksi kääntää järjestelmään sopivaksi ja tämän jälkeen annetaan asennuskomento. [17.]

Viimeiseksi asennetaan zlib1g-dev-kirjasto, joka mahdollistaa tiedon pakkauksen ja purkamisen. Tämän ohella asennetaan ei-pakolliset paketit libzma-dev, openssl sekä libssl-dev. Nämä mahdollistavat adobe flash -tiedostojen (.swf tiedostot) purkamisen sekä SHA- ja MD5-sormenjälkien ottamisen tiedostoista.

## 4.2 Snortin asentaminen

Snort tullaan asentamaan lähdetiedoista, joten on ensiksi ladattava Snort lähdetiedostot. Lähdetiedostot on pakattu yhdeksi arkistoksi. Tämä puretaan kansioon ja käännetään järjestelmään sopivaksi suoritettavaksi tiedostoksi. Kääntämisen jälkeen annetaan asennuskomento. Asennuskomennon jälkeen päivitetään järjestelmän jaetut kirjastot ja luodaan symbolinen linkki Snort suoritettavaan tiedostoon.

Suoritetaan Snort ajamalla kuvassa 7 oleva komento. Tämä komento testaa Snort binääritiedoston toimivuuden ja konfiguraatitiedostot, jotka on sille annettu. Kuvasta 7 voidaan havaita Snort sekä sen riippuvuuksien versionumerot.



```
snort@snort-VirtualBox: ~/snort_src/snort-2.9.8.2
snort@snort-VirtualBox:~/snort_src/snort-2.9.8.2$ snort -V
-*> Snort! <*-
o" )~
' ''
Version 2.9.8.2 GRE (Build 335)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.5.3
Using PCRE version: 8.31 2012-07-06
Using ZLIB version: 1.2.8
snort@snort-VirtualBox:~/snort_src/snort-2.9.8.2$
```

Kuva 7. Snort näyttää komentorivillä riippuvuuksien versionumerot.

## 4.3 Konfigurointi

Seuraavaksi konfiguroidaan Snort ajettavaksi NIDS-tilassa. Tämä vaatii käyttäjän luomisen lisäksi Snort-hakemistojen lisäystä, sääntö tiedostojen luomista sekä käyttäjäoikeuksien muokkausta. Kaikki konfiguroinnissa käytettävät komennot löytyvät jälleen

liitteestä 1. Konfiguraatio komentojen lisäksi löytyy liitteestä 1 muokkaukset tarvittaviin Snort-tiedostoihin.

#### 4.3.1 Muutokset NIDS-tilaa varten

Tällä hetkellä Snort toimii pääkäyttäjän oikeuksilla. Tämä ei kuitenkaan ole tarpeellista, sillä haluamme rajoittaa mahdollisen tunkeutumisen vaikutusta ajamalla Snorttia pienemmillä käyttäjätason oikeuksilla. Luodaan uusi käyttäjä nimeltään "snort\_ids" ja käyttäjäryhmä "snort\_ids".

Jotta Snorttia voidaan ajaa NIDS-tilassa, tarvitsee luoda sääntöjä sekä lokitusta varten hakemistoja. Hakemistojen luomisen jälkeen muokataan tiedosto-oikeudet. Tässä vaiheessa Snort pää- ja lokitushakemiston oikeudet muokataan siten, että snort\_ids käyttäjällä sekä snort\_ids-ryhmään kuuluvilla on täydet tiedosto-oikeudet Snort-hakemistoissa.

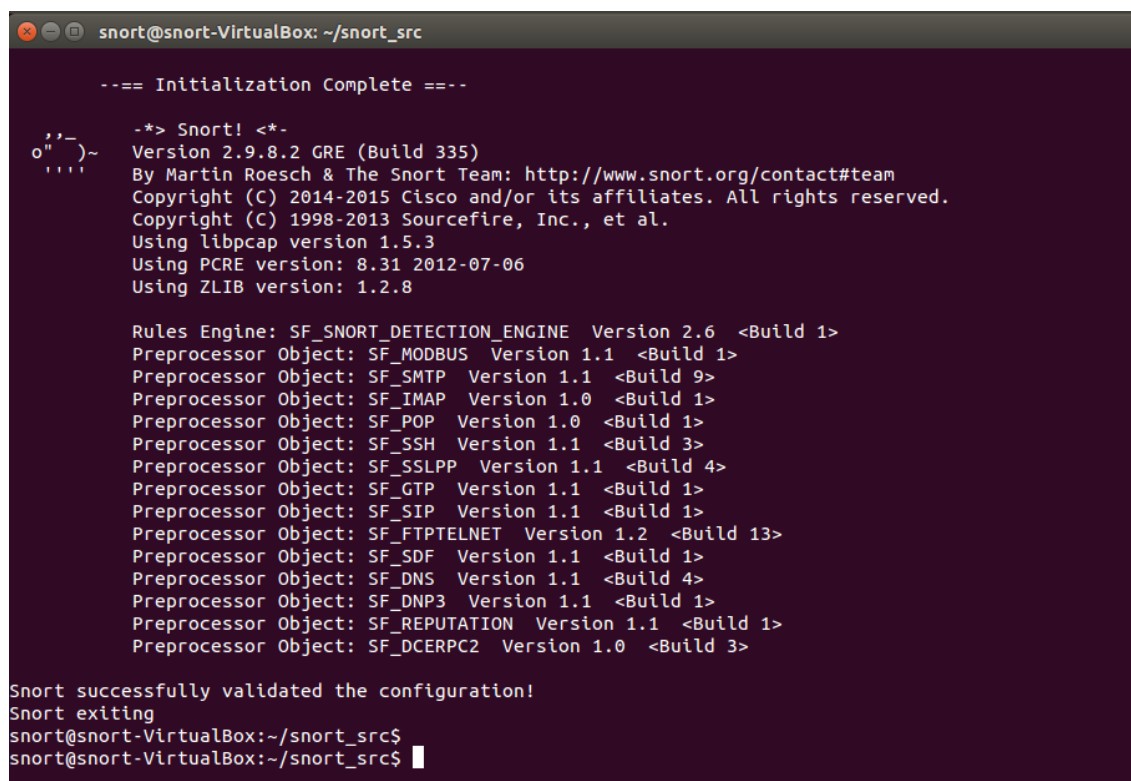
Snort tarvitsee käyttöönsä konfiguraatitiedostoja Snort-lähdekansioista. Nämä tiedostot kopioidaan Snort-hakemistoon. Näiden tiedostojen joukossa on Snort-pääkonfiguraatitiedosto nimeltään "snort.conf". Käytettäessä tätä tiedostoa argumentina Snort tietää, että käyttäjä haluaa ajaa ohjelmaa NIDS-tilassa. Pääkonfiguraatitiedostossa on määritelty sääntötiedostot, joita käytetään NIDS-tilaa käytettäessä. Nämä tiedostot otetaan kuitenkin pois käytöstä, sillä säännöt tullaan hakemaan yhteen tiedostoon, joka tullaan ottamaan käyttöön myöhemmin. Seuraavaksi muokataan pääkonfiguraatitiedostosta muutama muuttuja testi-ympäristöä vastaavaksi. Näihin kuuluu esimerkiksi sisäverkon osoitteen muuttaminen ja tiedostopolkujen muokkaus.

#### 4.3.2 Sääntöjen hakeminen

Sääntöjä on saatavilla monesta eri lähteestä. Säännöt tullaan kuitenkin hakemaan keskitetyllä työkalulla, joka kokoaa Snort-säännöt yhdeksi sääntötiedostoksi. Säännöt haetaan työkalulla nimeltään PulledPork. Tämä on perl-skripti, joka lataa, yhdistää ja päivittää säännöt Snort NIDS -tilaa varten. Käytettävät komennot PulledPorkin asentamiseen ja sääntöjen hakemiseen löytyvät täydellisinä jälleen liitteestä 1.

PulledPork ladataan avoimen lähdekoodin projektista ja annetaan perl-skriptin suorittamiselle tarvittavat oikeudet. Ennen PulledPork-skriptin ajamista luodaan käyttäjä tili snort.org-sivustolle. Tilin avulla otamme käyttöön sääntöjen haku-koodin, jolla pystymme hakemaan tavalliset säännöt Snort-sivulta. Tämän jälkeen sijoitamme hakukoodin PulledPork-konfiguraatitiedostoon liitteen 1 mukaisesti. Sääntöjen hakukoodin lisäksi muokkaamme muuttujien arvoa ympäristöämme vastaavaksi.

Seuraavaksi testataan PulledPork-skriptin toimivuutta. PulledPork-konfiguraatitiedosto on muokattu ympäristöämme vastaavaksi ja sääntöjen hakeminen pitäisi toimia hakukoodilla Snort-web-rajapintaa hyväksi käyttäen. PulledPork-skriptin ajettua otetaan säännöt käyttöön muokkaamalla Snort-pääkonfiguraatitiedostoa. Pääkonfiguraatitiedoston muokkaamisen jälkeen testaamme, hyväksyykö Snort tekemämme muutokset. Kuvasta 8 voimme nähdä, että Snort on onnistuneesti hyväksynyt konfiguroinnit.



```

snort@snort-VirtualBox: ~/snort_src

--== Initialization Complete ==--

-*> Snort! <*-
o" )~
' ' '
Version 2.9.8.2 GRE (Build 335)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.5.3
Using PCRE version: 8.31 2012-07-06
Using ZLIB version: 1.2.8

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.6 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>

Snort successfully validated the configuration!
Snort exiting
snort@snort-VirtualBox:~/snort_src$
snort@snort-VirtualBox:~/snort_src$

```

Kuva 8. Snort on onnistuneesti hyväksynyt konfiguraatitiedoston.

## 5 Tunkeilijan havaitsemisjärjestelmän testaus

Tunkeilijan havaitsemisjärjestelmää tullaan testaamaan laboratorioympäristössä, jossa on mahdollista simuloida oikeita hyökkäys- ja tunkeutumismetodeja. Tavoitteena NIDS-järjestelmän testauksessa on selvittää ja huomata järjestelmän vahvuudet ja heikkoudet tunkeutumisia vastaan. Näiden pohjalta voidaan tehdä järjestelmään muutoksia verkon tietoturvallisuuden parantamiseksi. Tästä voidaan myös tehdä johtopäätöksiä, jonka perusteella kartoitetaan NIDS-järjestelmän tehokkuutta tietoverkon puolustuksessa. Testauksessa saattaa myös tulla ilmi, mitä tunkeutumistapoja NIDS-järjestelmän on vaikea havaita.

Tutustumme ensiksi käytettävään testausympäristöön, jossa NIDS-järjestelmä toimii osana valmiiksi toimivaa lähiverkkoa. Tässä osiossa tutustutaan myös käytettäviin laitteisiin ja verkossa toimiviin palveluihin. Tämän jälkeen tehdään varsinainen järjestelmän testaus. Testauksessa simuloidaan tunkeutumista ja seurataan, mitä havaintoja NIDS-järjestelmä tekee hyökkäyksen aikana. Lopuksi pohditaan parannus- ja kehitystapoja NIDS-järjestelmää koskien.

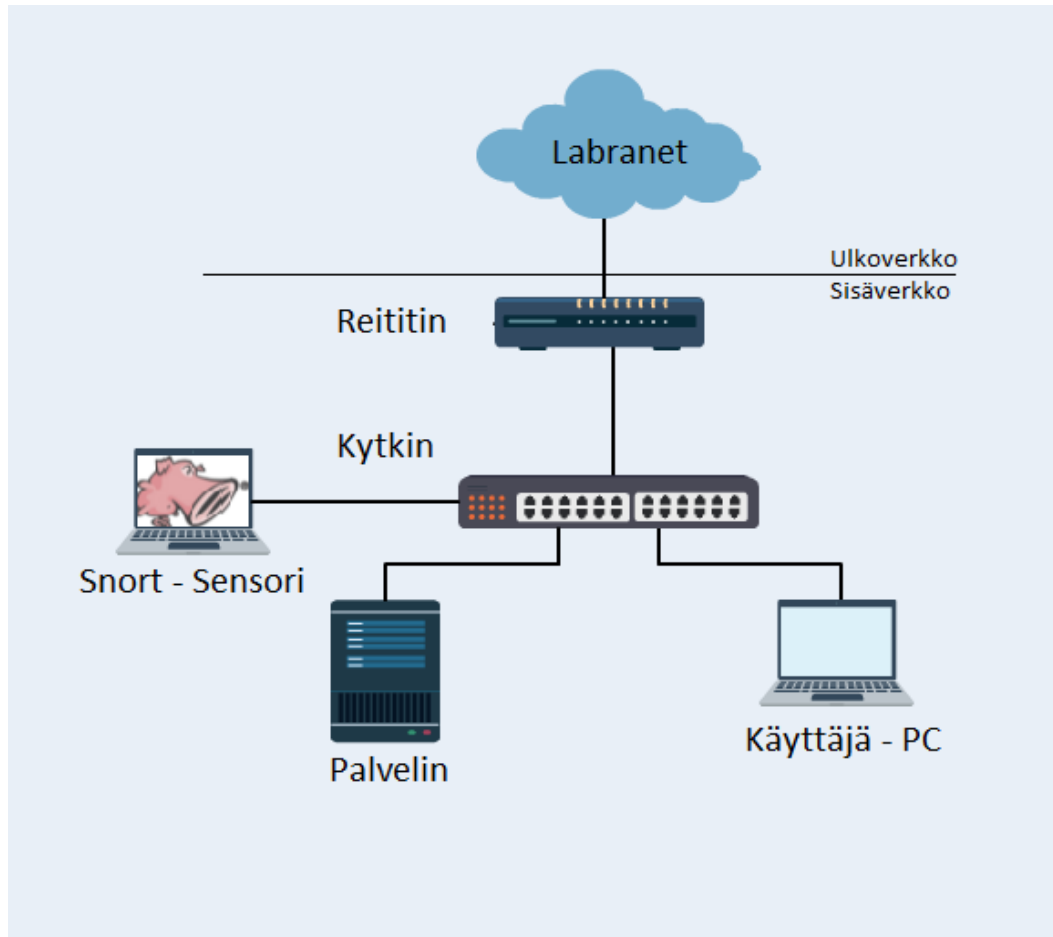
### 5.1 Testiympäristö

Testaus suoritetaan yksinkertaisessa verkkoympäristössä, joka koostuu lähiverkolle tutuista laitteista. Ympäristössä on lähiverkon runko eli reititin ja kytkin. Nämä laitteet tarjoavat verkossa toimiville päätelaitteille yhteyden muihin verkossa oleviin laitteisiin sekä Labranettiin. Käyttäjäpääte on tavallinen Windows-käyttöjärjestelmällä toimiva kone, jonka tarkoituksena on olla käyttäjälle toimistotyökaluja ja selainta pyörittävä laite.

Testissä pyritään simuloimaan pienyrityksen verkkoa, jossa palvelin on sijoitettu ehkä hieman kyseen alaiseen paikkaan lähiverkossa. Palvelimella toimii web-sovellus ja tiedostonjakopalvelin. Web-sovellus on käytettävissä myös ulkoverkosta eli tässä tapauksessa Labranetistä, jota simuloidaan erillisellä reitittimellä. Palvelin toimii Unix-pohjaisella käyttöjärjestelmällä.

Lähiverkossa toimiva palvelin sekä tavallinen käyttäjäpääte on liitetty Snort NIDS -järjestelmän kanssa kytkimeen. Kytkimeltä on konfiguroitu niin, että kytkimen ja reititti-

men välinen liikenne peilataan Snort-sensorille. Tämä tarkoittaa sitä, että kytkimeen liitettyjen laitteiden keskustellessa ulkoverkon laitteen kanssa kaikki verkkoliikenne kopioidaan Snort-sensorille. Tämä mahdollistaa verkkoliikenteen seurannan sisäverkon ja ulkoverkon laitteiden välillä.



Kuva 9. Laboratorioympäristötopologia.

Kuvasta 9 voidaan huomata laboratorioympäristössä käytettävä topologia ja verkkokuva. Sisäverkolle on varattu taulukon 1 mukaisesti IP-osoiteavaruus 192.168.10.0/24. Taulukosta 1 voidaan nähdä myös jokaisen sisäverkon laitteen yksittäinen IP-osoite.

Taulukko 1. Laboratorioympäristön laitteiden IP-osoitteet.

Laboratorioympäristö	IP	Lisätietoja
Sisäverkko	192.168.10.0 /24	
Reititin - Oletusyhdykäytävä	192.168.10.1	Portti Sisäverkkoon
Palvelin	192.168.10.10	Web-sovellus palvelin
Käyttäjä – PC	192.168.10.104	eth0
Snort – Sensori	192.168.10.6	NIDS

## 5.2 Hyökkäyspinta-ala

Tunkeilijan havaitsemisjärjestelmää testataan simuloimalla tunkeutumista ulkoverkosta sisäverkkoon. Ensiksi kuitenkin tehdään nopea katsaus mahdolliseen hyökkäyspinta-alaan. Hyökkäyspinta-ala käsittää joukon laitteita ja sovelluksia, joita hyväksikäyttämällä tunkeutuja voi pyrkiä sisäverkkoon. Tämän jälkeen käydään läpi simuloituja tunkeutumistapauksia ja miltä ne näyttävät NIDS-järjestelmän näkökulmasta.

Tunkeutumisen ennaltaehkäisemiseksi on ajateltava mahdollisia hyökkäystapoja, joita vihamielinen hyökkääjä voi käyttää tietoverkon hyväksikäyttämiseksi. Ensimmäiseksi uhan alla ovat laitteet, jotka ovat julkisesti näkyvillä Internetiin. Internetiin näkyvillä olevat laitteet ovat yleensä palvelimia tai muita sulautettuja järjestelmiä. Palvelimilla toimii yrityksen kannalta kriittisiä palveluita, kuten web-sovellukset, VPN-palvelut tai nimipalvelimet. Nämä tarjoavat hyökkääjälle oivan mahdollisuuden kerätä tietoa kohdeyrityksestä ja sen työntekijöistä. Yrityksen web-sivuilla voidaan julkaista yksityiskohtaisempaa tietoa yrityksen työntekijöistä ja tavoista ottaa heihin yhteyttä. Yksityiskohtaisemat tiedot voivat sisältää henkilön sähköposti-osoitteen, puhelinnumeron tai jopa työtehtävän yrityksessä.

Tunkeilijan on mahdollista suorittaa hyökkäys suoraan palvelinta vastaan käyttämällä hyväksi palvelimella toimivaa web-sovellus alustaa tai itse web-sovellusta. Web-sovelluksen hyväksikäytön mahdollistaa useimmin riittämätön käyttäjän syöttämä teksti tai syötteen tarkistus. Riittämätön syötteen tarkistus voi tarkoittaa erikoismerkkien syöttämistä järjestelmään, jotka mahdollistavat esimerkiksi alkuperäisen tietokantakyselyn muokkaamisen tai mielivaltaisen suoritettavan koodin syöttämisen järjestelmään. [18.]



Sulautettujen järjestelmien turvallisuus on huomioitava kartoittaessa lähiverkon turvallisuutta. Sulautetuiksi järjestelmiksi voidaan luokitella verkkolaitteita, kuten reititin, IP-kamera tai muu tietoverkkoon liitetty pienelektroniikka-laite. Tunkeutujan on mahdollista hyväksikäyttää näitä laitteita hankkiakseen pääsyn sisäverkkoon. Näissä laitteissa on yleensä hallintakonsoli, jonka avulla järjestelmän ylläpitäjän on mahdollista hallita laitetta. Puutteellisesti konfiguroitu tai heikot tietoturvaominaisuudet omaava laite voi vaarantaa koko tietoverkon turvallisuuden.

Tietoverkossa olevien laitteiden lisäksi tunkeutujan on mahdollista käyttää ihmisten toimintatapoja ja sosiaalista manipulointia, engl. Social engineering, hyväkseen tunkeutuessa kohdejärjestelmään. Tyypillinen sosiaalista manipulointia hyväksikäyttävä hyökkäys on s-postin kautta toimitettu haitallinen liite tai hyperlinkki. Haitallinen liite sähköpostin mukana voi näyttää ihan tavalliselta MS Office -pakettiin kuuluvalla tiedostolta tai PDF-asiakirjalta. [19.]

### 5.3 Hälytyksien testaus ja pakettianalysointi

Snort NIDS -järjestelmän tuottamia hälytyksiä tarkastellaan simuloimalla kahta hyökkäystapaa testiympäristöä vastaan. Simuloinnin tarkoituksena on testata NIDS-järjestelmän toimivuutta pienessä lähiverkossa. Testauksen aikana voimme huomata Snort-hälytyksistä hyökkäystoimenpiteiden ajankohdan sekä tunkeutumisen jälkeisen verkkoliikenteen.

#### 5.3.1 Hyökkäys Web-palvelinta vastaan

Web-palvelimella on tietoturva-aukko, joka mahdollistaa SQL-injektiohyökkäyksen suorittamisen palvelimen tietokantaa vastaan. Palvelimen tietokanta voi mahdollisesti sisältää käyttäjätilitietoja yrityksen työntekijöistä sekä yritykselle tärkeitä asiakastietoja.

Snort kirjoittaa tiedon mahdollisesti haitallisesta verkkoliikenteestä lokitiedostoon, joka sijaitsee hakemistopolussa `/var/log/snort/`. SQL-injektiohyökkäys suoritetaan IP-osoitteesta 202.254.186.190. Hyökkäyksen seuraamuksena voidaan tarkastella NIDS-järjestelmän tuottamia hälytyksiä juuri mainitun hakemistopolun tiedostosta `alert`. Lokitiedostosta on kuvakaappaus kuvassa 10.

```

[**] [1:2017808:1] ET WEB_SERVER Possible MySQL SQLi Attempt Information Schema Access [**]
[Classification: Web Application Attack] [Priority: 1]
04/26-12:08:53.303669 202.254.186.190:47484 -> 192.168.10.10:80
TCP TTL:64 TOS:0x0 ID:28123 IpLen:20 DgmLen:671 DF
***A**** Seq: 0xEA57257F Ack: 0xE39B8A4 Win: 0x1B80 TcpLen: 32
[Xref => http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet]

[**] [1:2006446:11] ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT [**]
[Classification: Web Application Attack] [Priority: 1]
04/26-12:08:53.303669 202.254.186.190:47484 -> 192.168.10.10:80
TCP TTL:64 TOS:0x0 ID:28123 IpLen:20 DgmLen:671 DF
***A**** Seq: 0xEA57257F Ack: 0xE39B8A4 Win: 0x1B80 TcpLen: 32
[Xref => http://doc.emergingthreats.net/2006446][Xref => http://en.wikipedia.org/wiki/SQL_injection]

[**] [1:2006445:12] ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM [**]
[Classification: Web Application Attack] [Priority: 1]
04/26-12:08:53.303669 202.254.186.190:47484 -> 192.168.10.10:80
TCP TTL:64 TOS:0x0 ID:28123 IpLen:20 DgmLen:671 DF
***A**** Seq: 0xEA57257F Ack: 0xE39B8A4 Win: 0x1B80 TcpLen: 32
[Xref => http://doc.emergingthreats.net/2006445][Xref => http://en.wikipedia.org/wiki/SQL_injection]

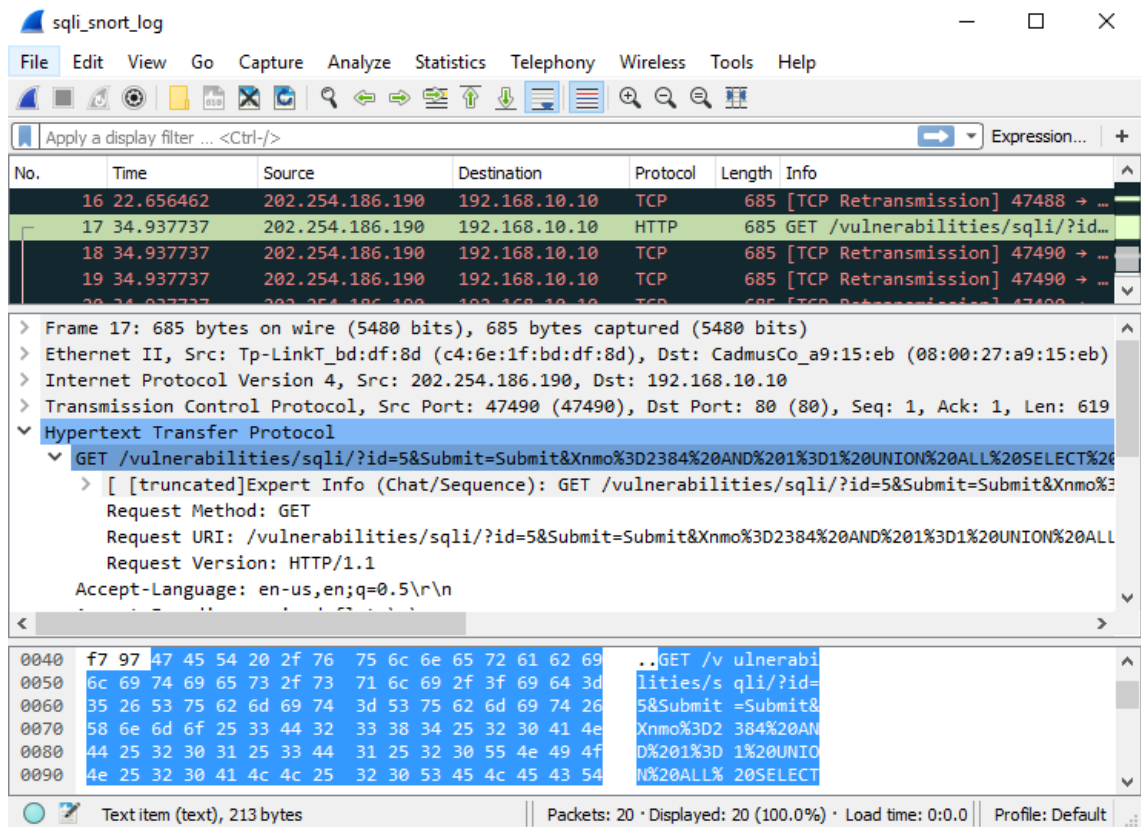
[**] [1:19439:8] SQL 1 = 1 - possible sql injection attempt [**]
[Classification: Web Application Attack] [Priority: 1]
04/26-12:08:53.303669 202.254.186.190:47484 -> 192.168.10.10:80
TCP TTL:64 TOS:0x0 ID:28123 IpLen:20 DgmLen:671 DF
***A**** Seq: 0xEA57257F Ack: 0xE39B8A4 Win: 0x1B80 TcpLen: 32
[Xref => http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/]
:

```

Kuva 10. Snort NIDS -järjestelmä hälyttää tunkeutumisesta.

Kuvasta voimme huomata, että Snort on havainnut hyökkäyksen web-palvelinta vastaan. Snort on huomannut verkkoliikenteessä yhtäläisyyksiä määriteltyihin Snort-sääntöihin ja tuottanut hälytyksen. Hälytyksen ensimmäisellä rivillä on säännön tunnusnumero sekä kuvaus hyökkäystavasta. Hyökkäystavaksi voidaan todeta SQL-injektio. Kuvasta voimme huomata hyökkäyksen ajankohdan sekä tunkeutujan että uhrin IP-osoitteet. Tämän lisäksi on tietoa verkko- ja kuljetuskerroksen arvoista. Näiden jälkeen on viittaus lisätietoihin hyökkäystapaan liittyen.

Näiden tietojen perusteella voidaan tehdä suuntaa antavia johtopäätöksiä tunkeutumis- tavasta ja ajankohdasta. Näillä tiedoilla voidaan aloittaa yksityiskohtaisempi tarkastelu hyökkäyksen laajuudesta. Snort lokittaa paketit pcap-tiedostoon snort.log.XXXXXXXXXX , joka sijaitsee hakemistopolussa /var/log/snort/. Pcap-tiedoston nimessä X-kirjaimien sijalla on aikaleima. Tiedosto on siis pakettikaappaus Snortin havaitsemasta haitallisesta liikenteestä. Pakettikaappausta on käytännöllistä tarkastella Wireshark-pakettianalysointi-ohjelmalla.



Kuva 11. Pakettikaappaus SQL-injektioista.

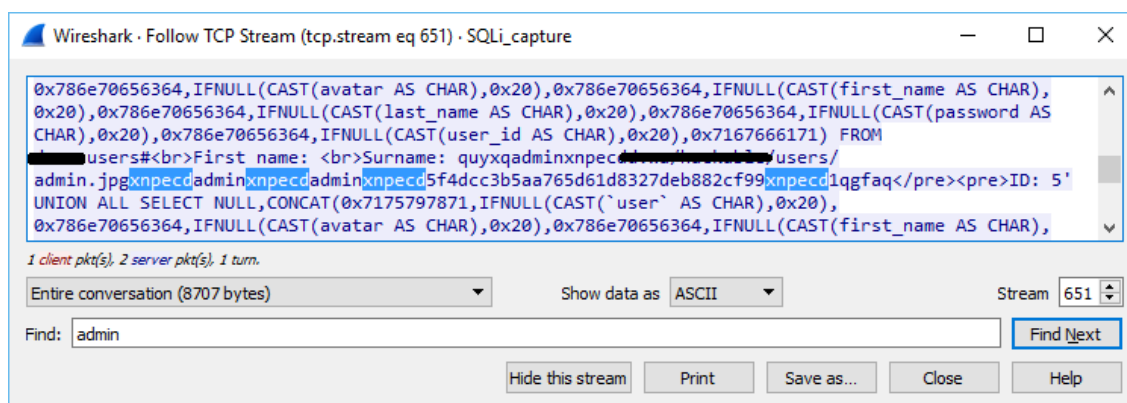
Pakettikaappauksesta voimme päätellä, että kyseessä on HTTP GET-pyyntö, jossa on mukana SQL-injektio. SQL-injektio on URL-koodattu, ja se voidaan avata helpommin tarkasteltavaksi syöttämällä se URL-dekooderiin. Wiresharkin hexdump-työkalun tulosteesta voimme päätellä, että SQL-injektio käyttää AND boolean totuusarvoon perustuvaa menetelmää, jolla voidaan tulostaa tietokannasta taulukoiden metadataa ja arvoja.

Seuraavaksi katsotaan web-palvelimen tietokannan taulukoiden arvoja ja vertaillaan niitä verkkoliikenteeseen. Tällä tavoin voimme helpommin todeta hyökkäyksen laajuuden ja kartoittaa tietokannasta urkittujen tietojen määrää.

user_id	user	avatar	password
1	admin	/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99
2	gordonb	/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03
3	1337	/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b
4	pablo	/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7
5	smithy	/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99

Kuva 12. Käyttäjä-taulukon arvot MySQL-tietokannassa.

Web-palvelimen tietokannassa on kuvan 12 mukaiset arvot käyttäjät-taulukossa. Taulukko sisältää käyttäjätunnuksia, nimiä sekä salasanoja. Salasanat on salattu kryptografisen tiivistefunktion ansiosta, mutta nämä on mahdollista murtaa vertailemalla arvoja esilaskettuihin tiivistefunktion arvoihin. Vertailemalla Wiresharkin avulla web-palvelimelta ulos menevää liikennettä ja tietokannan taulukon arvoja voimme havaita hyökkäyksen laajuuden.



Kuva 13. Web-palvelimelta lähtevä verkkoliikenne ASCII-muodossa.

Kuvassa 13 on etsitty palvelimelta ulos menevästä verkkoliikenteestä admin-merkkijonoa. Voimme havaita, että neljännellä rivillä on admin-merkkijono ja tätä seuraa hyvin tutun näköisiä merkkijonoja. Viidennellä rivillä verkkoliikenteessä taulukon arvot on eroteltu merkkijonolla "xnpecd". Vertailemalla merkkijonoja tietokannan ja ulos menevän verkkoliikenteen väliltä havaitsemme admin-käyttäjän salasanan vuodon. Tästä voimme päätellä, että hyökkääjä on onnistuneesti hyväksikäyttänyt palvelimen haavoittuvuutta ja hakenut tietokannasta arkaluontoista tietoa.

### 5.3.2 Käyttäjätunnuksen saastuminen

Tavallisen käyttäjätunnuksen saastumisella voi olla suuri merkitys yrityksen tietoturvan kannalta. Käyttäjän tunnuksen saastuessa, tunkeutuja voi käyttää saastunutta konetta hyväkseen ohittamalla käyttäjä oikeuksia paikallisesti sekä tiedustelemalla sisäverkon palveluita ja verkkoliikennettä.

Kohdistetut hyökkäykset käyttävät usein sähköpostia saastuttaakseen sisäverkon käyttäjätunnuksia. Tämänkaltaiset hyökkäykset käyttävät sosiaalista manipulointia huijatakseen käyttäjän avaamaan haitallinen liite tai linkki sähköpostin kautta. Sähköpostili-

kenne on salattu käyttäjöpäätteelle asti, joten NIDS-järjestelmän on mahdoton havaita haitallisen sähköpostin saapumista sisäverkkoon.

```

[**] [119:31:1] (http_inspect) UNKNOWN METHOD [**]
[Classification: Unknown Traffic] [Priority: 3]
04/26-12:35:20.597273 192.168.10.104:1063 -> 222.173.190.239:80
TCP TTL:62 TOS:0x0 ID:54052 IpLen:20 DgmLen:166 DF
***A**** Seq: 0x617F5573 Ack: 0x44D385D9 Win: 0x7210 TcpLen: 20

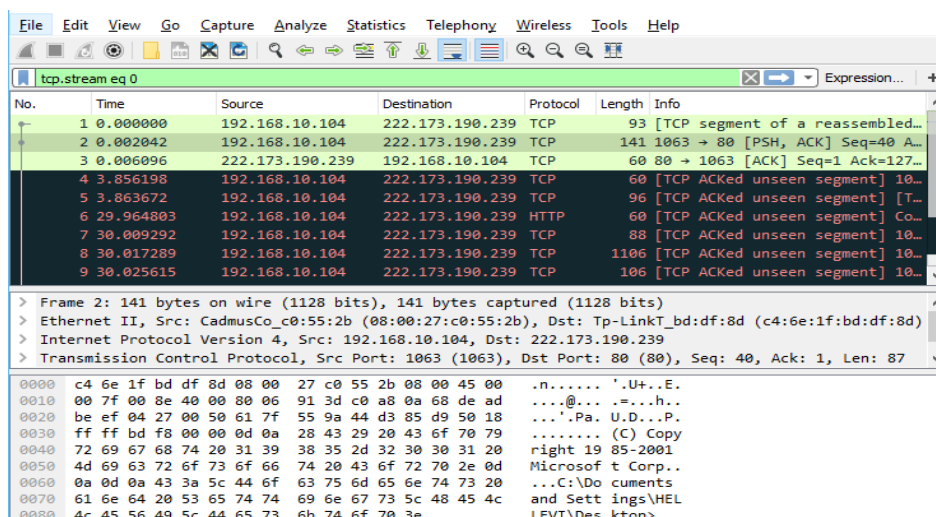
[**] [119:32:1] (http_inspect) SIMPLE REQUEST [**]
[Classification: Unknown Traffic] [Priority: 3]
04/26-12:35:20.603369 222.173.190.239:80 -> 192.168.10.104:1063
TCP TTL:62 TOS:0x0 ID:54052 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x44D385D9 Ack: 0x617F55F1 Win: 0x7210 TcpLen: 20

[**] [119:33:1] (http_inspect) UNESCAPED SPACE IN HTTP URI [**]
[Classification: Unknown Traffic] [Priority: 3]
04/26-12:35:24.453471 192.168.10.104:1063 -> 222.173.190.239:80
TCP TTL:62 TOS:0x0 ID:58184 IpLen:20 DgmLen:2113 DF
***A**** Seq: 0x617F55F1 Ack: 0x44D385F6 Win: 0x8380 TcpLen: 20

```

Kuva 14. Snort on havainnut epäilyttävää verkkoliikennettä.

Käyttäjöpäätteen saastuessa Snort NIDS -järjestelmä huomaa epäilyttävää verkkoliikennettä. Snort luokittelee verkkoliikenteen tuntemattomaksi. Snort-hälytyslokiteidostosta on vaikea päätellä verkkoliikenteen haitallisuutta. Snort huomaa verkkoliikenteen olevan epätyypillistä HTTP-liikenteelle ja tästä syystä se aiheuttaa hälytyksen. On tarkasteltava Snort NIDS -järjestelmän pakettikaappauksia, jotta voimme tarkalleen todeta verkkoliikenteen haitallisuuden.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.104	222.173.190.239	TCP	93	[TCP segment of a reassembled...
2	0.002042	192.168.10.104	222.173.190.239	TCP	141	1063 → 80 [PSH, ACK] Seq=40 A...
3	0.006096	222.173.190.239	192.168.10.104	TCP	60	80 → 1063 [ACK] Seq=1 Ack=127...
4	3.856198	192.168.10.104	222.173.190.239	TCP	60	[TCP ACKed unseen segment] 10...
5	3.863672	192.168.10.104	222.173.190.239	TCP	96	[TCP ACKed unseen segment] [T...
6	29.964803	192.168.10.104	222.173.190.239	HTTP	60	[TCP ACKed unseen segment] Co...
7	30.009292	192.168.10.104	222.173.190.239	TCP	88	[TCP ACKed unseen segment] 10...
8	30.017289	192.168.10.104	222.173.190.239	TCP	1106	[TCP ACKed unseen segment] 10...
9	30.025615	192.168.10.104	222.173.190.239	TCP	106	[TCP ACKed unseen segment] 10...

```

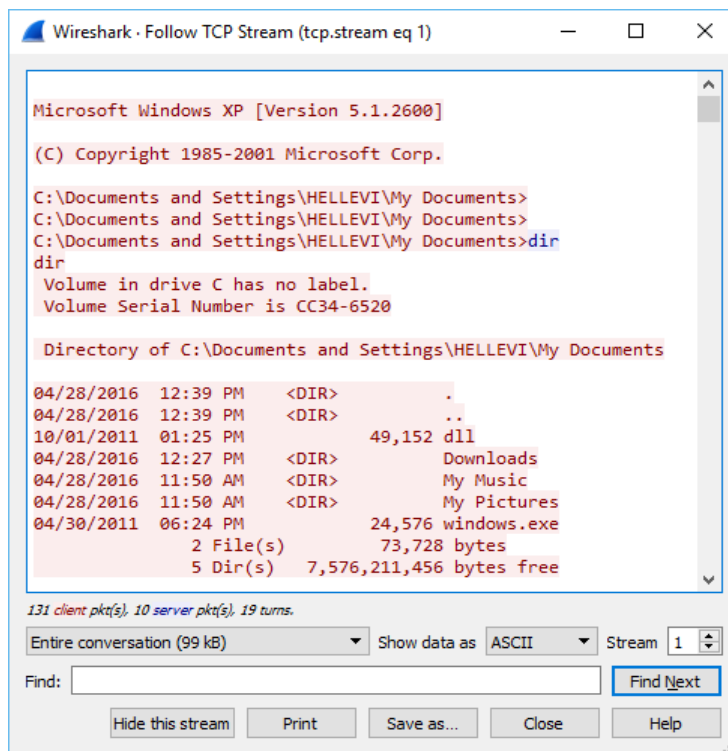
> Frame 2: 141 bytes on wire (1128 bits), 141 bytes captured (1128 bits)
> Ethernet II, Src: CadmusCo_c0:55:2b (08:00:27:c0:55:2b), Dst: Tp-LinkT_bd:df:8d (c4:6e:1f:bd:df:8d)
> Internet Protocol Version 4, Src: 192.168.10.104, Dst: 222.173.190.239
> Transmission Control Protocol, Src Port: 1063 (1063), Dst Port: 80 (80), Seq: 40, Ack: 1, Len: 87
0000 c4 6e 1f bd df 8d 08 00 27 c0 55 2b 08 00 45 00  .n.....'.U+..E.
0010 00 7f 00 8e 40 00 80 06 91 3d c0 a8 0a 68 de ad  ....@... ..h.
0020 be ef 04 27 00 50 61 7f 55 9a 44 d3 85 d9 50 18  ....Pa. U.D...P.
0030 ff ff bd f8 00 00 0a 28 43 29 20 43 6f 70 79  .... (C) Copy
0040 72 69 67 68 74 20 31 39 38 35 2d 32 30 30 31 20  right 19 85-2001
0050 4d 69 63 72 6f 73 6f 66 74 20 43 6f 72 70 2e 0d  Microsof t Corp..
0060 0a 0d 0a 43 3a 5c 44 6f 63 75 6d 65 6e 74 73 20  ...C:\Do cuments
0070 61 6e 64 20 53 65 74 74 69 6e 67 73 5c 48 45 4c  and Sett ings\HEL
0080 4c 45 56 49 5c 44 65 73 6b 74 6f 70 3e          LEVI\Des ktop>

```

Kuva 15. Pakettikaappaus epäilyttävästä verkkoliikenteestä.

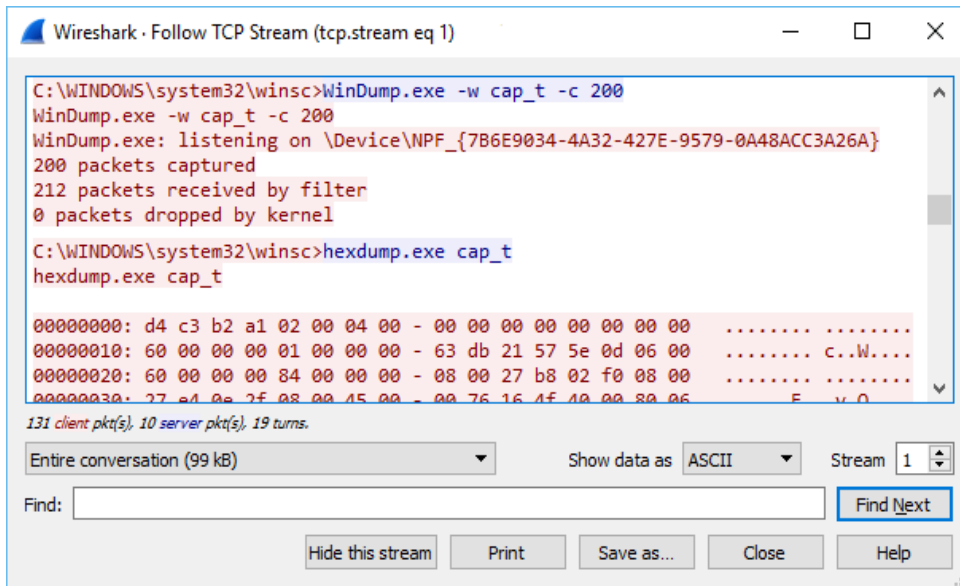
Kuvassa 15 on pakettikaappaus verkkoliikenteestä. Pakettikaappauksesta voimme hyvinkin päätellä verkkoliikenteen olevan hyvin epätyypillistä HTTP-liikennettä. Valit-

semme Wiresharkista "Follow → TCP Stream", jolloin voimme tarkastella istunnon hyötykuormaa.



Kuva 16. Verkkoliikenteen hyötykuorma TCP-istunnosta.

Kuvassa 16 on saastuneelta koneelta lähtenyt ja tullutta verkkoliikennettä. Pääteeltä lähtenyt liikenne on punaisella ja sinisellä saapuva verkkoliikenne ASCII-muodossa. Verkkoliikenteessä on Windows-komentorivin tekstiä ja tiedostolistauskomento. Kuvasta voimme epäillä, että kyseessä on takaovi, engl. backdoor, jota tunkeutuja käyttää hyväkseen. Haittaohjelma on todennäköisesti luonut käänteisen komentotulkki yhteyden, engl. reverse shell, takaisin komentokoneelle, josta tämä käynnistää komentoja sisäverkon käyttäjäpäätteelle tietoverkon ylitse.



```

Wireshark · Follow TCP Stream (tcp.stream eq 1)
C:\WINDOWS\system32\winsc>WinDump.exe -w cap_t -c 200
WinDump.exe -w cap_t -c 200
WinDump.exe: listening on \Device\NPF_{7B6E9034-4A32-427E-9579-0A48ACC3A26A}
200 packets captured
212 packets received by filter
0 packets dropped by kernel

C:\WINDOWS\system32\winsc>hexdump.exe cap_t
hexdump.exe cap_t

00000000: d4 c3 b2 a1 02 00 04 00 - 00 00 00 00 00 00 00 00 .....
00000010: 60 00 00 00 01 00 00 00 - 63 db 21 57 5e 0d 06 00 ..... C..W....
00000020: 60 00 00 00 84 00 00 00 - 08 00 27 b8 02 f0 08 00 .....
00000030: 27 e4 0e 2f 08 00 45 00 - 00 76 16 4f 40 00 80 06 ..... F v O

131 client pkt(s), 10 server pkt(s), 19 turns.
Entire conversation (99 kB) Show data as ASCII Stream 1
Find: Find Next
Hide this stream Print Save as... Close Help

```

Kuva 17. Tunkeutuja suorittaa pakettikaappauksen kohdekoneella.

Kuvassa 17 on samaisesta tietoliikenneistunnosta hyötykuormakaappaus. Haittaohjelma on asentanut Windump- ja hexdump-ohjelmat käyttäjätasolle. Windump mahdollistaa tietoliikennepakettien kirjaamisen tiedostoon ja hexdump-työkalulla voidaan tulostaa tiedostojen binääridataa. Tunkeutuja on käyttänyt Windump-ohjelmaa kirjoittaakseen cap\_t-tiedostoon käyttäjätasolta lähtevää ja tulevaa verkkoliikennettä. Tiedostoon tietoliikennepakettien kirjaamisen jälkeen tunkeutuja on käyttänyt Hexdump-työkalua tulostaakseen pakettikaappauksen tulokset vakiotulosteeseen. Tunkeutujan on mahdollista koota tiedostot uudestaan omalla päätteellään jäsentämällä verkkoliikennettä.

Snort NIDS -järjestelmän tuottaman hälytyksen kautta voitiin käynnistää tarkempi verkkoliikenteen tarkastelu. Tarkastelun pohjalta havaittuun tunkeutumiseen voidaan nyt puuttua tilanteeseen sopivilla toimenpiteillä. Tilanteeseen sopiva toimenpide voisi olla käyttäjätasotteen eristäminen ja Snort-sääntöjen muokkaaminen.

#### 5.4 Järjestelmän kehitysehdotuksia

Snort NIDS -järjestelmä voi ruuhkautua liiallisen verkkoliikenteen seurauksena. Tästä syystä Snort voidaan konfiguroida kirjoittamaan tapahtumat binäärimuodossa tiedostoon ja Snortille jää enemmän aikaa prosessoida verkkoliikennepaketteja. Kun Snort on kirjoittanut tapahtumat binäärimuotoon, työkalu nimeltään Barnyard2 hakee binäärida-

tan ja sijoittaa sen MySQL-tietokantaan. Tietokannassa tapahtumien käsittely ja lukeminen on käytännöllisempää kuin suoraan Snort-lokeista lukeminen.

Snort-tapahtumien lukemista helpottaisi Snorby-graafinen web-liittymä. Snorby hakee tapahtumat tietokannasta ja näyttää ne järjestelmän ylläpitäjälle selkokielenä. Käyttöliittymän avulla olisi käytännöllistä analysoida tunkeutumiseen liittyvää tietoliikennettä suoraan web-liittymästä. Tällöin ei tarvitsisi erikseen hakea käsin lokeista pakettikaappaus tiedostoa ja sijoittaa sitä sitten pakettianalysointiohjelmaan.

Snortin olisi ideaalista käynnistyä automaattisesti järjestelmän käynnistyessä. Tällöin NIDS-järjestelmällä olisi vähemmän aikaa olla alhaalla järjestelmän käynnistyessä tai huoltokatkon aikana. Viimeisimmät Snort-säännöt olisi hyvä ladata päivittäin. Ladattaessa uudet säännöt NIDS-järjestelmä vaatisi aina uudelleenkäynnistymisen.

Näillä toimenpiteillä NIDS-järjestelmästä saataisiin kokonaisuudessaan hyvin toimiva kokonaisuus, joka ei liiallisesti kuormittaisi järjestelmän ylläpitäjää. Näiden muutosten pohjalta järjestelmästä tulisi hyvin automaattinen, ja hälytyksen tuottamat tapahtumat voitaisiin tutkia nopeammin ja tehokkaammin.

## 6 Yhteenveto

Insinööriyön tarkoituksena oli testata tunkeilijan havaitsemisjärjestelmän toimintaa. Testaamisen lisäksi tarkoituksena oli havaita, miten tarkasti NIDS-järjestelmällä on mahdollista selvittää tunkeutumisen laajuutta ja vakavuutta.

Tavoitteena oli asentaa NIDS-järjestelmä ja testata sen toimivuutta aitoja tietoverkkouhkia vastaan. Tunkeutumisen tarkempi tarkastelu tietoliikennepaketti tasolla antoi paljon hyödyllistä tietoa tietoverkkouhan selvittämisessä. Pääasiassa tavoitteeseen päästiin, sillä rakennetulla ympäristöllä voidaan havaita tietoverkko uhkia ja tarkastella niitä lähemmin pakettianalysointiohjelmalla. Järjestelmää olisi voinut vielä kehittää lisäämällä web-rajapinta, josta tapahtumia olisi voinut tarkastella. Työssä halusin kuitenkin enemmän keskittyä järjestelmän testaamiseen ja pakettien analysointiin. Pidän työtä onnistuneena sillä tavoitteisiin päästiin ja työn tuloksena on toimiva NIDS-järjestelmä.



IDS-järjestelmät ovat kokonaisuudessaan tärkeä osa tietoverkon puolustusta. Suuremmissa tietoverkoissa uhkien havaitseminen voi olla haastavaa, sillä tapahtumien määrän kasvaessa haitallista ja hyödyllistä liikennettä on vaikeampi erottaa toisistaan. Tästä johtuen on tärkeä mukauttaa Snort-sääntöjen herkkyyttä ympäristön mukaan. NIDS-järjestelmien haasteena on salattu verkkoliikenne. Käyttäjätasolle asti salattua liikennettä ei pystytä purkamaan, ellei NIDS-järjestelmässä ole asianmukaista sertifikaattia.

Työstä on tarkoituksella jätetty pois testauksen esivalmisteluvaihe, jossa suunniteltiin tunkeutumiset tietoverkkoa vastaan. Tunkeutumisten suunnittelussa aikaa meni eniten aikaa haittaohjelman muuttamiseen testauksen tarpeisiin. Tämän lisäksi aikaa kului SQL-injektion ja virtuaalikoneiden konfiguroinnin kanssa. Nämä vaiheet jätettiin pois, sillä halusin keskittyä olennaiseen ja työhön olisi tullut mielestäni liikaa ylimääräistä IDS-järjestelmiin kuulumatonta tietoa.

Insinööriyön lopputuloksena syntyi toimiva NIDS-järjestelmä. Järjestelmän avulla voidaan havaita uhkia tietoverkkoa vastaan ja tarkastella niitä lähemmin tietoliikennepaketitason tasolla. Työn testaus kappaleessa käytiin läpi eri vaiheet tunkeutumisen havaitsemisesta tarkempaan analysointiin. Insinööriyön pohjalta on mahdollista rakentaa NIDS-järjestelmä jo olemassa olevaan lähiverkkoon tietoturvan lisäämiseksi.

Verkkotasolla IDS-järjestelmän toteuttaminen on hyvin tärkeää, mutta siitä ei silti ole yksinään turvaamaan koko tietoverkkoa. IDS-järjestelmät toteutetaan yleensä suuremmissa yrityksissä tunkeutumisen estojärjestelmän kanssa. Täydellinen tietoturva ratkaisu tarvitsee monitasoista lähestymistapaa tietoverkon suojaamiseksi. Monitasoisella ratkaisulla en tarkoita pelkästään tietoturvaohjelmistojen ja laitteiden lisäämistä, vaan yhtä tärkeää on jatkuva henkilöstön kouluttaminen tietoturvalliseen työskentelyyn.

## Lähteet

- 1 <https://github.com/the-tcpdump-group/libpcap>. Luettu 21.1.2015.
- 2 Using libPcap for monitoring distributed applications. Sivut 92-97. Julkaisija IEEE. Julkaisu päivämäärä 28.6.–2.7.2010. Paikka kansainvälinen konferenssi, High Performance Computing and Simulation (HPCS). Luettu 21.1.2015.
- 3 [wiki.wireshark.org/Development/LibpcapFileFormat](http://wiki.wireshark.org/Development/LibpcapFileFormat). Luettu 10.2.2015.
- 4 [https://www.wireshark.org/docs/wsug\\_html\\_chunked/ChIntroHistory.html](https://www.wireshark.org/docs/wsug_html_chunked/ChIntroHistory.html). Luettu 18.2.2016.
- 5 [www.wireshark.org](http://www.wireshark.org). Luettu 18.2.2016.
- 6 Analysis and Application of Wireshark in TCP/IP Protocol Teaching. Sivut 269-272. Julkaisija IEEE. Julkaisu paikka: E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on 17-18 April 2010. Luettu 16.2.2016.
- 7 <https://github.com/mcr/tcpdump/blob/18a28b6e5f4b7f31375779232f0af38b24d031c8/LICENSE>. Luettu 12.3.2016.
- 8 <http://www.tcpdump.org/manpages/tcpdump.1.html>. Luettu 16.3.2016.
- 9 <https://www.paloaltonetworks.com/documentation/glossary/what-is-an-intrusion-detection-system-ids>. Luettu 16.3.2016.
- 10 <http://handle.dtic.mil/100.2/ADA516706>. Luettu 13.3.2016.
- 11 <http://ieeexplore.ieee.org.ezproxy.metropolia.fi/stamp/stamp.jsp?tp=&arnumber=6014986> A Parallel Technique for Improving the Performance of Signature-Based Network Intrusion Detection System IEEE. Luettu 16.3.2016.
- 12 Intrusion Alert : An Ethical Hacking Guide to Intrusion Detection. E-kirja. Luettu 20.3.2016.
- 13 Snort dokumentointi. <https://snort.org/documents/snort-2-9-8-x-on-ubuntu-12-lts-and-14-lts-and-15>. Luettu 25.3.2016.
- 14 <http://resources.infosecinstitute.com/snort-rule-writing-for-the-it-professional/>. Luettu 26.3.2016.

- 15 <http://www.tomsitpro.com/articles/intrusion-detection-intrusion-prevention-systems-ids-ips,2-959.html>. Luettu 3.4.2016.
- 16 <https://community.linuxmint.com/software/view/libdumbnet-dev>. Luettu 5.4.2016.
- 17 <https://www.sans.org/reading-room/whitepapers/intrusion/analysis-snort-data-acquisition-modules-34027>. Luettu 10.4.2016.
- 18 <https://www.sans.org/reading-room/whitepapers/securecode/web-application-injection-vulnerabilities-web-app-039-s-security-nemesis-34247>. Luettu 12.4.2016.
- 19 <http://www2.fireeye.com/rs/fireeye/images/fireeye-how-stop-spearphishing.pdf>. Luettu 14.4.2016.

## Snort asennus Ubuntu 14.04.01-versioon komentoriviltä

### Esivaatimukset

```
sudo apt-get install -y build-essential
sudo apt-get install -y libpcap-dev libpcrc3-dev libdumbnet-dev
mkdir ~/snort_src cd ~/snort_src
sudo apt-get install -y bison flex
cd ~/snort_src
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
tar -xvzf daq-2.0.6.tar.gz cd daq-2.0.6
./configure
make
sudo make install
sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev
```

### Snort asennus

```
cd ~/snort_src
wget https://snort.org/downloads/snort/snort-2.9.8.0.tar.gz
tar -xvzf snort-2.9.8.0.tar.gz
cd snort-2.9.8.0
./configure --enable-sourcefire
make
sudo make install
sudo ldconfig
sudo ln -s /usr/local/bin/snort /usr/sbin/snort
```

### Konfigurointi

```
# Luo snort_ids käyttäjä ja ryhmä
sudo groupadd snort_ids
sudo useradd snort_ids -r -s /sbin/nologin -c SNORT_IDS -g snort_ids
```

```
# Luo snort hakemistot
sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /etc/snort/so_rules
```

```
# Luo tiedostot, joihin lisätään snort sääntöjä
sudo touch /etc/snort/rules/iplists/black_list.rules
sudo touch /etc/snort/rules/iplists/white_list.rules
sudo touch /etc/snort/rules/local.rules
sudo touch /etc/snort/sid-msg.map
```

```
# Luo Snortille lokitus hakemistot
sudo mkdir /var/log/snort
sudo mkdir /var/log/snort/archived_logs
```

```
# Muokkaa tiedosto-oikeuksia
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /var/log/snort/archived_logs
sudo chmod -R 5775 /etc/snort/so_rules
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

# Tee snort_ids käyttäjistä seuraavien hakemistojen omistaja
sudo chown -R snort_ids:snort_ids /etc/snort
sudo chown -R snort_ids:snort_ids /var/log/snort
sudo chown -R snort_ids:snort_ids /usr/local/lib/snort_dynamicrules

# Kopioi Snort lähde kansioista tiedostoja
cd ~/snort_src/snort-2.9.8.0/etc/
sudo cp *.conf* /etc/snort
sudo cp *.map /etc/snort
sudo cp *.dtd /etc/snort
cd ~/snort_src/snort-2.9.8.0/src/dynamic-
preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
sudo cp * /usr/local/lib/snort_dynamicpreprocessor/

# Ottaa pois käytöstä sääntötiedostot
sudo sed -i "s/include \$RULE_PATH/#include \$RULE_PATH/" /etc/snort/snort.conf

# muokkaa /etc/snort/snort.conf tiedostosta seuraavat rivit:
ipvar HOME_NET 192.168.10.0/24
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
var WHITE_LIST_PATH /etc/snort/rules/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists

Pulledpork asennus

# Pulledpork tarvitsee seuraavia ohjelmisto paketteja toimiakseen
sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl

# Lataa Pulledpork 0.7.2-196 skripti ja konfiguraatitiedostot
cd ~/snort_src
wget
https://github.com/finchy/pulledpork/archive/8b9441aeeb7e1477e5be415f27dbc4eb25d
d9d59.tar.gz \ -O pulledpork-0.7.2-196.tar.gz

tar xvfz pulledpork-0.7.2-196.tar.gz
mv pulledpork-8b9441aeeb7e1477e5be415f27dbc4eb25dd9d59 pulledpork-0.7.2-196
cd pulledpork-0.7.2-196/
sudo cp pulledpork.pl /usr/local/bin
sudo chmod +x /usr/local/bin/pulledpork.pl
sudo cp etc/*.conf /etc/snort
```

## Pulledporkilla sääntöjen haku

```
# Tee käyttäjä tili www.snort.org sivustolle
# Hae tiliisi sidottu sääntöjen haku koodi "oinkcode".
# Muokkaa pulledpork.conf tiedoston muuttujia:
sudo vim /etc/snort/pulledpork.conf
Muuta riveille 19 & 26 <oinkcode> merkkijonon tilalle sääntöjen haku-koodi.
Muuta rivillä 74 oleva muuttuja seuraavaksi: rule_path=/etc/snort/rules/snort.rules
Muuta rivillä 89 oleva muuttuja seuraavaksi: local_rules=/etc/snort/rules/local.rules
Muuta rivillä 92 oleva muuttuja seuraavaksi: sid_msg=/etc/snort/sid-msg.map
Muuta rivillä 119 oleva muuttuja seuraavaksi: config_path=/etc/snort/snort.conf
Muuta rivillä 133 oleva muuttuja seuraavaksi: distro=Ubuntu-14-04
Muuta rivillä 141 oleva muuttuja seuraavaksi:
black_list=/etc/snort/rules/iplists/black_list.rules
Muuta rivillä 150 oleva muuttuja seuraavaksi: IPRVersion=/etc/snort/rules/iplists

# Hae säännöt manuaalisesti seuraavalla skriptillä
sudo /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l

# Sääntöjen käyttöönotto

Lisätään rivi /etc/snort/snort.conf tiedostoon, jolla snort ids ottaa käyttöön ladatut säännöt.
sudo vim /etc/snort/snort.conf
Riville 547: include $RULE_PATH/snort.rules

# Käyttäjäoikeuksien päivitys snort hakemistossa sääntöjen haun jälkeen.
sudo chmod -R 5775 /etc/snort
sudo chown -R snort_ids:snort_ids /etc/snort

# Testataan Snort konfiguraatitiedosto
sudo snort -T -c /etc/snort/snort.conf -i eth0
```