Teemu Toivanen

Tunkeutumisen havaitsemisjärjestelmä Snort

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

10.05.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Teemu Toivanen Tunkeutumisen havaitsemisjärjestelmä Snort 42 sivua + 2 liitettä 10.05.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaajat	Yliopettaja Janne Salonen Operatiivinen johtaja Juha-Pekka Järvenpää

Insinöörityön tavoitteena oli selvittää tapoja, joilla Sigmatic Oy:n asiakkaiden sisällönhallintajärjestelmillä toteutettujen sivujen murtoihin voitaisiin reagoida nopeammin. Tämä insinöörityö tehtiin Sigmatic Oy:lle Helsingissä vuoden 2016 keväällä.

Sigmatic Oy on tietoverkkopalveluita tarjoava yritys. Sen tuotteisiin kuuluu muun muassa webhotellit, joihin asiakkaat voivat asentaa omia kotisivujaan.

Kotisivujen ylläpitämiseen ja toteuttamiseen käytetään yleensä erilaisia sisällönhallintajärjestelmiä (esimerkiksi Wordpress). Suuren suosion ja heikon tietoturvan takia nämä sisällönhallintajärjestelmät ovat usein rikollisten tähtäimessä. Rikolliset voivat käyttää murrettuja palvelunestohyökkäyksissä ja roskapostituksessa.

Tutustuin työssäni avoimeen lähdekoodiin perustuvaan tunkeutumisen havaitsemisjärjestelmään Snorttiin.

Työssä käyn lävitse tunkeutumisen havaitsemis- ja tunkeutumisen estojärjestelmien erot. Selitän mihin toimintamalleihin kyseiset järjestelmät jaetaan ja miten järjestelmät luokittelevat hälytykset. Lisäksi kerron mitä eroa on verkkopohjaisella- ja isäntäpohjaisella järjestelmällä.

Järjestelmiin tutustumisen jälkeen kerron tarkemmin Snortista ja sen toiminnasta. Kerron yksityiskohtaisesti, miten Snort kaappaa verkkopaketin ja tunnistaa siinä olevan haitallisen toiminnan. Tämän jälkeen kerron miten Snortille luodaan sääntöjä, ja mistä nämä säännöt koostuvat. Lopuksi vielä käyn lävitse Snortin ja siihen asennettavien hyödyllisten lisäosien asennuksen itse pystyttämääni testiympäristöön.

Työn tuloksena saatiin testiympäristö, jossa voidaan kokeilla uusimpia hyökkäyksiä valvotusti. Analysoin ja tutkin hyökkäyksiä testiympäristössä. Testiympäristössä tehtyjen havaintojen perusteella voidaan luoda sääntöjä, joiden avulla voidaan havaita haitallinen toiminta tuotannossa.

Avainsanat	IDS, NIDS, IPS, CMS, Snort, Wordpress

Author Title Number of Pages Date	Teemu Toivanen Intrusion Detection System Snort 42 pages + 2 appendices 10 May 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data Networks
Instructors	Janne Salonen, Principal Lecturer Juha-Pekka Järvenpää, Chief Operating Officer

The goal of this thesis was to figure out ways how Sigmatic Oy can react quicker when its customers Wordpress sites is being hacked.

Sigmatic Oy is a tech company offering web hosting, virtual servers and dedicated servers. The main focus in this thesis is web hosting customers who have created Wordpress sites using Wordpress.

Many people create their websites with content management systems because they are quite easy to use and do not require programming skills. Because of high popularity and weak security these systems are in big favor by hackers. Hackers can use hacked sites to use Distributed Denial of Service attacks or generate a lot of email spam.

In this thesis the open source software Snort was used as Intrusion Detection System (IDS).

The study explains the main differences between an intrusion detection system and intrusion prevention system. It also explains the differences between a fingerprint system and anomaly-based system and how these systems classify generated alerts.

The study also introduces Snort and how it works detailing and how Snort handles captured packets and generates alerts of harmful packets. Furthermore, the study explains how one can generate rules for Snort and what the rules contain as well as how Snort and add-ons are installed into a test environment.

The result of the project was a test environment where rules for Snort can be tested and rules for attack generated.

Keywords	IDS, NIDS, IPS, CMS, Snort, Wordpress

Sisällys

Lyhenteet

IaaS	Infrastructure as a Service. Palvelimien ulkoistaminen.
IDS	Intrusion Detection System. Tunkeutumisen havaitsemisjärjestelmä.
IPS	Intrusion Prevention System. Tunkeutumisen estojärjestelmä.
P2P	Peer to Peer. Vertaisverkko.
HIDS	Host Intrusion Detection System. Isäntäpohjainen tunkeutumisen havaitsemisjärjestelmä.
NIDS	Network Intrusion Detection System. Verkkopohjainen tunkeutumisen havaitsemisjärjestelmä.
DDoS	Distributed Denial of Service. Hajautettu palvelunestohyökkäys.
SQL	Structured Query Language. Tietokantojen kyselykieli.
RPC	Remote Procedure Call. Etäproseduurikutsu.
XSS	Cross site scripting. Tietoturva-aukko, joka esiintyy web-palveluissa.
CMS	Content management system. Sisällönhallintajärjestelmä (Wordpress).

1 Johdanto

Insinöörityön tavoitteena on selvittää tapoja, joilla yleisimpiin sisällönhallintajärjestelmiin kohdistuvista tunkeutumisista, tai niiden yrityksistä ilmoitettaisiin automaattisesti käyttäen tunkeutumisen havaitsemisjärjestelmää, Intrusion Detection System (IDS).

Sigmaticin asiakaspalvelu vastaanottaa jatkuvasti ilmoituksia, että asiakkaiden ylläpitämille sisällönhallintajärjestelmille on murtauduttu ja sivuille on lisätty haittakoodia. Valitettavasti nämä murtautumiset johtuvat julkaisualustojen suuresta suosiosta ja niihin ladattavien lisäosien huonosta tietoturvasta. Rikolliset etsivät jatkuvasti päivittämättä jääneitä sisällönhallintajärjestelmiä, joihin voisivat murtautua. Näitä murrettuja sivuja rikolliset voivat käyttää useisiin eri tarkoituksiin, koska sisällönhallintajärjestelmällä on laajat pääsyoikeudet palvelimella. Insinöörityössä on tavoitteena selvittää mahdollisia tapoja huomata automaattisesti murrettuja sivuja jo ennen kuin niillä aletaan toteuttamaan haitallista toimintaa.

Työssä perehdytään tunkeutumisen havaitsemis- ja tunkeutumisen estojärjestelmiin. Havaitsemisjärjestelmänä työssä käytetään avoimeen lähdekoodiin perustuvaa Snortia, joka kykenee monitoroimaan verkkoliikennettä ja vertaamaan kaapattuja paketteja ennalta määritettyihin sääntöihin. Snort voi toimia myös tunkeutumisen estojärjestelmänä, Intrusion Prevention System (IPS), jolloin verkkoliikenne kulkisi Snortin lävitse, jolloin Snort pystyisi automaattisesti estämään haitalliset paketit.

Insinöörityössä käsittelen ensin teoriaa havaitsemisjärjestelmistä. Teorian jälkeen kerron Snortista ja muista insinöörityössä käytetyistä sovelluksista. Tämän lisäksi käyn vielä läpi testiympäristön pystyttämisen. Testiympäristössä Snortin toimintaa voidaan testata, jotta tuotannossa Snortin hälytysten luontia ei tarvitse erikseen tehdä.

2 Sigmatic Oy

Sigmatic on vuonna 2001 perustettu suomalainen, pääkaupunkiseudulla sijaitseva tietoverkkopalveluita tarjoava yritys. Sigmaticin tarjoamiin tuotteisiin kuuluvat webhotellit, virtuaalipalvelimet ja fyysiset palvelimet. Palvelintilan lisäksi Sigmatic tarjoaa asiantuntijatyötä palvelinylläpidossa ja ohjelmistokehityksessä. Sigmatic tarjoaa myös verkkotunnuspalvelua verkkotunnukset.fi-nimisellä sivulla. [1.]

Sigmatic tuottaa palveluitaan kahdessa täysin omassa hallinnassaan olevassa laitesalissa, joista toinen sijaitsee Otaniemessä ja toinen Lauttasaaressa. Yrityksen liikevaihto oli vuonna 2015 noin kolme miljoonaa euroa ja asiakkaita oli noin 15 tuhatta. Sigmatic työllisti samana vuonna kolmetoista IT-alan ammattilaista. [1.]

Vuonna 2012 Sigmaticistä irtaantui yritys nimeltä UpCloud, joka toimii Sigmaticin kanssa samoissa tiloissa. UpCloud tuottaa Infrastructure as a Service (IaaS) palveluita. Talouskasvu kyseisellä yrityksellä on ollut nopeaa. UpCloud hakee talouskasvua Suomen lisäksi myös ulkomailta. Vuonna 2015 UpCloudillla oli palvelinsaleja Helsingissä, Lontoossa, Frankfurtissa ja Chicagossa. [26.]

3 Tunkeutumisen havaitsemisjärjestelmä

Tunkeutumisen havaitsemisjärjestelmän (IDS) tavoitteena on havaita mahdolliset tunkeutujat, ja luoda hälytys havaitsemastaan haitallisesta toiminnasta. Vastaavaa tehtävää arkielämässä toteuttavat muun muassa auton varashälyttimet, liiketunnistimet ja valvontakamerat. Näiden jokaisen järjestelmän tavoitteena on huomata ja ilmoittaa mahdollisesta tunkeutujasta. [2, s. 216.]

Usein ihmiset sekoittavat tunkeutumisen havaitsemisjärjestelmän tunkeutumisen estojärjestelmään, Intrusion Prevention System (IPS). IDS vain monitoroi sille määritettyä palvelua ja ilmoittaa haitallisesta liikenteestä eteenpäin. IDS:n luoman hälytyksen jälkeen erikseen määritelty henkilö voi estää mahdollisesti haitallisen toiminnan. Sen sijaan IPS pystyy monitoroinnin lisäksi automaattisesti estämään mahdollisen haitallisen toiminnan ilman ulkopuolista tahoa. [2, s. 216.]

Esimerkiksi IDS voidaan asentaa valvomaan yrityksen sisäverkkoa ja luomaan hälytys, jos se havaitsee verkossa vertaisverkkoliikennettä, peer to peer (P2P). Erotuksena IDS:ään IPS voitaisiin määritellä estämään automaattisesti P2P-paketit, jotka se havaitsee. [50.]

3.1 Miksi IDS on hyödyllinen?

IDS on tärkeä lisäkerros yrityksen tietoturvaan, koska yhdenkään yrityksen verkko ei ole niin pieni tai tarpeeksi turvallinen, etteikö sitä pitäisi valvoa mahdollisilta hyökkäyksiltä ja väärinkäytöksiltä. Jos palveluun on pieninkään mahdollisuus tunkeutua, joku tai jokin onnistuu kuitenkin tunkeutumisessa ennemmin tai myöhemmin.

Rikolliset haluavat saada hallintaansa mahdollisimman monta tietokonetta ja verkkolaitetta, joilla he voisivat esimerkiksi generoida suuria palvelunestohyökkäyksiä, Distributed Denial of Service, (DDoS), tai lähettää valtavia määriä roskapostia. Jos verkkoa ei valvota lainkaan, niin rikollinen toiminta saatetaan huomata vasta pitkän ajan kuluttua, tai pahimmassa tapauksessa se saattaa jäädä kokonaan huomaamatta. [3, s. 25.] Myös verkonvalvonnalla voidaan huomata, jos yrityksen sisäisiä tietoturvamääräyksiä rikotaan. Esimerkiksi, jos yrityksen sisäverkosta yritetään siirtää tiettyjä tiedostoja ulkoverkkoon tai yrityksen työntekijät vierailevat kielletyillä sivuilla.

Tietoverkkotunkeutumiset ovat nykyään lähes jokapäiväinen ongelma. Vaikka yrityksen verkko olisi suojattu kuinka hyvin tahansa, suojaus ei välttämättä pysty estämään kaikkia mahdollisia tunkeutumisia yrityksen verkkopalveluille. Ennemmin tai myöhemmin tunkeutujat voivat löytää reitin yrityksen verkkoon, ja toteuttaa jotain ei-toivottua. Luottamalla pelkästään verkon suojaukseen on mahdollista, että tunkeutumiset huomataan paljon myöhemmin. Silloin tunkeutujat ehtivät aiheuttamaan suurempaa vahinkoa ja heidän poistaminen verkosta voi olla työläämpää.

Valvomalla verkkoa saadaan tietoa myös tunkeutumisyrityksistä, jolloin mahdolliseen suurempaan hyökkäykseen voidaan varautua paremmin. Esimerkiksi yrityksen porttien skannaaminen ja useiden pienempien palvelunestohyökkäysten toteuttaminen voi tarkoittaa sitä, että joku yrittää etsiä reittiä palveluihin tai testata yrityksen reagointikykyä.

Vaikka käytössä olisi tehokkaat suojaukset ja verkkoa valvottaisiin, on mahdollista, että rikolliset pääsevät käsiksi johonkin osaan verkkoa. Murtautumisen selvittämisessä tulisi etsiä vastaus seuraaviin viiteen kysymykseen:

- Mitä tarkalleen on tapahtunut?
- Milloin mahdollinen tunkeutuminen on tapahtunut?
- Missä järjestelmissä tunkeutuja(t) on käynyt?
- Kuka on tunkeutunut?
- Miksi on tunkeuduttu? [2, s. 421.]

Vaikka IDS ei estä hyökkäyksiä, se voi tarjota vastauksen heti useampaan edellä lueteltuihin kysymyksiin, ja täten nopeuttaa tunkeutujan poistamista verkosta. Mitä aikaisemmin ylläpitäjä saa tietää murtautumisesta, sitä enemmän hänelle jää aikaa minimoida hyökkääjän aiheuttamat vahingot ja estää hyökkääjän mahdollinen eteneminen ja verkossa.

Markkinoilla on lukuisia tunkeutumisen havaitsemisjärjestelmiä, mutta käytetyin on avoimeen lähdekoodiin perustuva Snort. [2, s. 216.] Snortista ja sen toiminnasta kerron luvussa 4.

3.2 Järjestelmän toimintamallit

Tunkeutumisen havaitsemisjärjestelmän voi jakaa kahteen eri toimintamalliin. Ensimmäinen malleista on sormenjälkiin perustuva järjestelmä SB (Signature-Based) ja toinen on anomaliapohjainen järjestelmä AB (Anomaly-Based), eli poikkeuksiin perustuva järjestelmä.

3.2.1 Sormenjälkiin perustuva järjestelmä (SB)

Sormenjälkiin perustuva järjestelmä luottaa tunkeutujien aiheuttamien hyökkäysten jättämiin sormenjälkiin. Järjestelmään tulee olla tallennettu jokaisen tunnetun hyökkäyksen jättämä sormenjälki. Järjestelmä vertaa tallennettuja sormenjälkiä kaappaamiensa pakettien sormenjälkiin ja etsii mahdollisia yhtenäisyyksiä. Kun järjestelmä huomaa yhteneväisyyden kaapatun paketin ja tallennetun sormenjäljen välillä, järjestelmä voi luoda

automaattisesti hälytyksen. Hälytyksiä hallinnoidaan säännöillä, (Rules) jotka hyödyntävät näitä sormenjälkiä. Esimerkkinä seuraava sääntö, joka generoi hälytyksen, kun ulkoverkosta yritetään kirjautua valvottavaan verkkoon root-käyttäjällä SSH-yhteyttä käyttämällä. [5, s. 132.]

```
alert tcp any any -> $HOME_NET 22 (msg:“SSH-login attempt”; flow:to_server,
established;content:”root”;)
```

Sormenjälkiin perustuvan järjestelmän suurimpana heikkoutena voidaan pitää järjestelmän toistuvaa päivittämistä ja nollapäivähaavoittuvaisuuksia. Nollapäivähaavoittuvaisuudella tarkoitetaan tietoturva-aukkoa, jolle ei ole vielä olemassa korjausta. Järjestelmän sääntöjä täytyy päivittää jatkuvasti, jotta myös uusimmat hyökkäykset pystyttäisiin huomaamaan.

Myös hyökkäysten sormenjälkiä on helppo muokata. Pelkästään tietokantakyselyiden, eli SQL-kyselyiden suuri monimuotoisuus, tekee lähes mahdottomaksi huomata jokaista muokattua kyselyä. Jotta kaikki SQL-hyökkäykset huomattaisiin, jokaisesta haitallisesta tietokantakyselystä täytyy luoda erillinen sääntö. Järjestelmän etuna on, että se antaa vähemmän virheellisiä hälytyksiä, koska hälytykset perustuvat aina tallennettuihin tunkeutumismalleihin. [4.; 5, s. 133-134.; 6.]

Hyvänä esimerkkinä sormenjälkien heikkoudesta voidaan pitää vuonna 2001 paljastuneita matoja nimiltään ”The Code Red v1 ja v2”. Kumpikin madoista hyödynsi puskurin ylivuotoa Microsoftin IIS-järjestelmässä. Kyseisten matojen tavoitteena oli hyödyntää IIS-järjestelmää DDoS-hyökkäyksessä Valkoista taloa vastaan. IDS:n kannalta kyseiset madot ovat mielenkiintoisia, koska alkuperäisen ”The Code Red v1” HTTP-pyyntö poikkesi v2-versioista vain hieman. Ohessa v1-version HTTP-pyyntö:

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%
u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b
%u53ff%u0078%u0000%u00=a  HTTP/1.0
```

Kun oheinen hyökkäys saatiin tallennettua IDS:n sääntöihin, madon käyttö pystyttiin huomaamaan ja estämään. Meni vain kaksi viikkoa, kun julkaistiin uusi mato (The Code Red v2), jonka HTTP-pyyntö oli:

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u90
90%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
```

Koska uuden hyökkäyksen jättämä sormenjälki oli eri kuin alkuperäisellä, uusi versio jäi järjestelmältä huomaamatta. Tämä on aina ongelmana sormenjälkiin perustuvassa IDS-järjestelmässä. [4.; 5, s. 133-134.; 6.]

3.2.2 Poikkeamiin perustuva järjestelmä (AB)

Poikkeamiin perustuva järjestelmä luottaa pakettien vertailuun ja verkkoliikenteen normaalitilan muutoksien havaitsemiseen. AB-järjestelmä luo parhaan mahdollisen määrityksen normaalille verkkoliikenteelle. Tämän jälkeen monitoroitavaa verkkoliikennettä verrataan määritettyyn normaaliin liikenteeseen.

Esimerkiksi, jos normaalitilanteessa maanantaiaamuna yhteyksiä palvelimille muodostetaan 10 kappaletta, mutta jokin maanantaiaamu yhteyksiä tuleekin 1000. Tämän poikkeaman IDS huomaa ja luo siitä hälytyksen. Hälytyksen perusteella ei osata vielä sanoa, onko verkkoon tunkeuduttu, joten asia pitää erikseen tarkistaa. [27, s. 334.]

Poikkeuksien vertailun takia AB-järjestelmä toimii paremmin mahdollisia nollapäivähyökkäyksiä vastaan. Suurimpana heikkoutena voidaan kuitenkin pitää sitä, että järjestelmä antaa paljon enemmän virheellisiä hälytyksiä, jotka kuormittavat järjestelmänvalvojaa turhaan. Ovela hyökkääjä voi myös toteuttaa hyökkäyksen hitaasti, jolloin hyökkäystä saatetaan luulla verkon normaaliksi liikenteeksi.

3.3 Hälytykset

Tunkeutumisen havaitsemisjärjestelmät koostuvat sensoreista, joita asennetaan valvottavaan verkkoon. Asentamisen jälkeen nämä sensorit monitoroivat määritettyä verkkoa ja tallentavat havaitsemansa tunkeutumisyritykset.

Tunkeutumisen havaitsemisjärjestelmän luomat hälytykset voidaan jakaa kolmeen eri tyyppiin, jotka ovat False Negative, False Positive ja True positive.

3.3.1 False Negative (FN)

False negative eli *”väärät negatiiviset”* ovat tunkeutumisia, jotka jäivät syystä tai toisesta havaitsemisjärjestelmältä huomaamatta. False negativet ovat vakava ongelma, koska silloin järjestelmä ei toimi vaaditulla tavalla, mikä tekee järjestelmästä melkein hyödyttömän. False negative -hälytysten lukumäärää on vaikea määrittää, koska tunkeutuminen täytyy huomata jollakin toisella tavalla. Tämän jälkeen voidaan todeta, että järjestelmään oli murtauduttu. Esimerkiksi IDS valvoo asunnon lukittua ulko-ovea, mutta tunkeutuja murtautuukin asuntoon rikotun ikkunan kautta, minkä takia IDS ei huomaa murtautumista. Murtautuminen huomataan vasta jälkikäteen murron aiheuttamien lasinsirujen takia. [7, s. 149-173.; 8, s. 49.; 9.]

3.3.2 False Positive (FP)

False positive eli ”väärät positiiviset” ovat hälytyksiä, jotka aiheutuvat normaalista verkkoliikenteestä, mutta havaitsemisjärjestelmä luokittelee verkkoliikenteen (mahdolliseksi) tunkeutumisyritykseksi. Järjestelmän ylläpitäjän kannalta on parempi, että tulee vääriä hälytyksiä kuin, ettei hälytyksiä tulisi ollenkaan. False positivet aiheuttavat kuitenkin ylimääräistä työtä ylläpitäjälle. Jos turhia hälytyksiä tulee paljon, ylläpitäjän tarkkuus saattaa laskea. Tällöin haitallinen toiminta voidaan vahingossa pitää false positivena. False positiven havainnollistaa seuraava esimerkki. Esimerkissä IDS valvoo taas kotiovea. Tällä kertaa ei olla kotona, kun naapuri soittaa ovikelloa pyytääkseen apua kotiverkon pystyttämisessä. Kun ovikello soi, IDS luo hälytyksen mahdollisesta tunkeutumisyrityksestä. [7, s. 149-173.; 8, s. 49.; 9.]

3.3.3 True Positive (TP)

True positive eli ”tosi positiivinen” on hälytys, jonka IDS on generoinut oikein. IDS on onnistuneesti huomannut tunkeutumisen, tai sen yrityksen, ja ilmoittanut siitä järjestelmän ylläpitäjälle. Hälytys voidaan generoida joko sormenjälkiin perustuvalla tavalla tai havaitun poikkeaman perusteella. [7, s. 149-173.]

3.4 Verkkopohjainen tunkeutumisen havaitsemisjärjestelmä

NIDS-järjestelmiä (Network Intrusion Detection System) käytetään nykyään lähes kaikissa suuremmissa verkoissa, koska hyökkäysten määrä ja samankaltaisuus ovat kasvaneet huomattavasti. [6, s. 1.] NIDS yrittää paikantaa hyökkäysyrityksen verkkoliikenteen perusteella.

NIDS koostuu sensoreista, jotka monitoroivat verkkoliikennettä, josta ne yrittävät paikantaa mahdollisia väärinkäytöksiä. Nämä sensorit tallentavat huomaamansa väärinkäytökset tietokantaan, josta ylläpitäjät voivat seurata hälytyksiä, minkä jälkeen niihin voidaan reagoida halutulla tavalla. NIDS voi käyttää tunkeutujien havaitsemiseen sormenjälkiin ja poikkeamiin perustuvaa järjestelmää. [7, s. 149-173.]

NIDS suositellaan asennettavaksi palomuurin sisäpuolelle, jotta palomuuri suodattaa suurimman osan haitallisesta liikenteestä. Liikenne voidaan tuoda NIDS-sensorille esimerkiksi peilaamalla palomuurilta sisäverkkoon tulevan portin liikenne. Tällä tavalla NIDS on lähes huomaamaton verkossa, joten hyökkääjät eivät tiedä niin helposti, että heitä voidaan seurata.

Paras käytäntö olisi asentaa NIDS molemmille puolille verkkoa (eli sisäverkkoon ja ulkoverkkoon), jotta pystyttäisiin valvomaan niin sisäisiä uhkia kuin ulkoverkosta tulevia uhkia. Suurin osa vakavimmista hyökkäyksistä toteutetaan kuitenkin sisältäpäin, joten sisäverkkoa ei suositella jättää kokonaan valvomatta. Sisäverkosta hyökkäyksiä voivat tehdä esimerkiksi turhautuneet työntekijät tai yritysvakoilijat, joille on sallittu pääsy verkkoon. Myös, jos työntekijän VPN-yhteys (Virtual Private Network) kaapataan, hyökkääjät pystyvät täten kiertämään palomuurin ja aloittamaan hyökkäyksen sisältäpäin. [27, s. 330.]

3.5 Isäntäpohjainen tunkeutumisen havaitsemisjärjestelmä

Isäntäpohjainen tunkeutumisen havaitsemisjärjestelmä HIDS (Host Intrusion Detection System) huomaa tunkeutumiset ja niiden yritykset suoraan laitetasolla. HIDS:iin voidaan laskea kuuluvan käyttöjärjestelmiin asennettavat ohjelmat, jotka varoittavat tunkeutumisesta. Esimerkkinä HIDS-sovelluksesta on avoimen lähdekoodin OSSEC. OSSEC on

suosittu sovellus noin viidellä tuhannella latauksella joka kuukausi. Sovellusta käyttävät niin yliopistot, valtiot kuin suuret yritykset. [6, s. 1.; 51.; 52.]

HIDS ei tarkkaile verkkopaketteja, vaan seuraa kyseisen isäntäkoneen tapahtumia ja pyrkii niiden perusteella huomaamaan tunkeutujan. HIDS voidaan mieltää virustorjunta-ohjelmistona, vaikka HIDS ei torju haitallista toimintaa. [27, s. 331-332.]

Vaikka HIDS ei valvo verkkoa, niin sillä pystytään tehokkaasti valvomaan valittua palvelinta ja havaitsemaan kyseisen palvelimen väärinkäytökset. Esimerkiksi **N**IDS ei pysty huomaamaan, jos yrityksen tiedostoja ladataan USB-tikulle palvelimelta, koska tiedostot eivät kulje verkon yli. Jos palvelimelle on asennettua **H**IDS, lataus voidaan huomata ja siihen voidaan reagoida. Yhdessä **N**IDS ja **H**IDS luovat tehokkaan valvontajärjestelmän.

3.6 IDS:n tehokkuus

IDS:n tehokkuus tai hyödyllisyys voidaan määrittää sen mukaan, kuinka tehokkaasti se tunnistaa tunkeutumiset, ja minkä verran se tuottaa vääriä hälytyksiä. Tehokkuuden ja tarkkuuden määrittämiseen voidaan käyttää kaavioita:

$$tehokkuus = \frac{TP}{(TP + FN)}$$

$$tarkkuus = \frac{TP}{(TP + FP)}$$

Oheisissa kaavioissa TP on True positiven määrä ja FN on False negativen määrä. FP on False positive-hälytysten lukumäärä valvotulla aikajaksolla.

Esimerkiksi kuukauden aikana tuli hälytyksiä 2901 kappaletta, joista oli 1400 TP, yksi FN ja 1500 FP. IDS:N tehokkuutena voidaan pitää 99,9 %, eli IDS huomaa 99,9 % tapahtuneista hyökkäyksistä.

$$tehokkuus = \frac{1400}{(1400 + 1)} = 0{,}9992$$

IDS:n tarkkuus on silloin 48,28 %. Tämä tarkoittaa sitä, että 51,72 % hälytyksistä on turhia. [6, s. 7.]

$$tarkkuus = \frac{1400}{(1400 + 1500)} = 0{,}4827$$

Esimerkissä käytetyt luvut ovat täysin keksittyjä, jotta laskutustoimitukset voidaan havainnollistaa.

4 Snort

Snort on avoimeen lähdekoodiin perustuva NIDS ja IPS. Havaitsemisen lisäksi Snortia voidaan käyttää perinteisenä pakettikaapparina, kuten TcpDump.

Snort on sormenjälkiin perustuva järjestelmä, joten se tarvitsee sääntöjä toimiakseen tehokkaasti. Snortia on myös mahdollista käyttää anomaliapohjaisena. Snortille on saatavissa lukuisia ilmaisia sääntöjä, mutta näiden sääntöjen lataamista varten täytyy rekisteröityä käyttäjäksi Snortin kotisivuilla. Rekisteröityneet käyttäjät saavat uudet säännöt noin kaksi viikkoa sen jälkeen, kun ne on julkaistu. Snortista maksavat tilaajat saavat uusimmat säännöt heti, kun ne julkaistaan.

Snortin sääntöjä voidaan hallinnoida automaattisesti erillisellä sovelluksella nimeltään PulledPork.

Snort tallentaa hälytyksen aiheuttaneet paketit tietokantaan, josta ylläpitäjä näkee hälytyksen aiheuttaneet paketit. Snort voidaan myös määrittää tallentamaan paketit erilliseen binääritiedostoon, jotta Snortille jäisi enemmän aikaa pakettien käsittelyyn. Tätä binääritiedostoa voidaan käsitellä erillisellä sovelluksella nimeltä Barnyard2. Barnyard2 tallentaa binäärimuotoisen tiedon helpommin luettavaan muotoon esimerkiksi MySQL-tietokantaan.

Snorttia hallinnoidaan komentoriviltä, mutta tallennettujen hälytysten monitorointi on huomattavasti helpompaa graafisesti. Tätä varten Snortille on kehitetty sovelluksia, jotka näyttävät Barnyardin luoman tietokannan sisällön graafisesti. Tällä hetkellä suosituimmat näistä sovelluksista ovat Snorby ja Base. [25, s. 28.; 10.; 11.]

4.1 Snortin historia

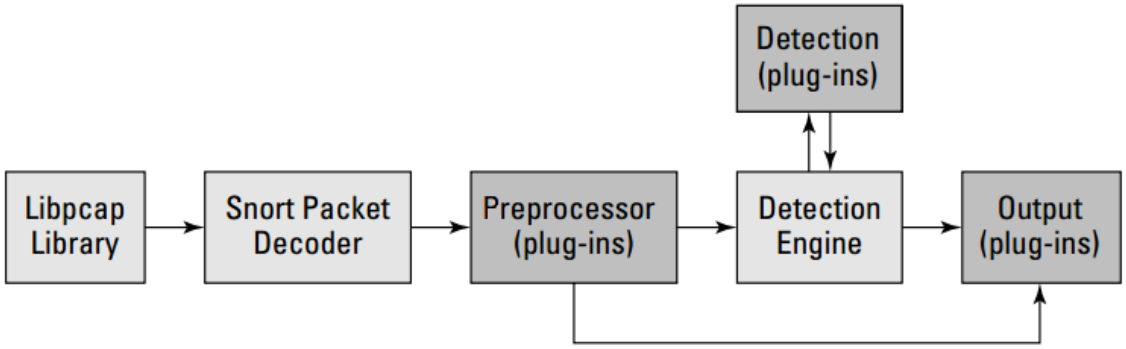
Vuonna 1998 Marty Roesch aloitti Snortin kehittämisen. Roesch ohjelmoi Linux-pohjaisen pakettien tiedusteluun perustuvan sovelluksen nimeltään APE. Myöhemmin siitä tuli Snort. Roesch halusi, että APE toimisi useilla eri käyttöjärjestelmillä, tallentaisi kaappaamansa paketit binäärimuodossa järjestelmän levylle sekä näyttäisi useat eri verkkopaketit samalla tavalla. Roeschin tavoite oli luoda tehokkaampi pakettien tiedusteluun perustuva sovellus omaan käyttöönsä. Roesch halusi, että Snort hyödyntää myös libpcap-kirjastoa, joka antaisi Snortille joustavuutta verkkoliikenteen suodattamisen ja pakettien tiedustelun välille. [25, s. 33-34.]

Snort tarjosi järjestelmän ylläpitäjille uuden vaihtoehdon pakettien tiedusteluun, koska Snortin julkaisemisen aikaan vain TcpDump hyödynsi libpcap-kirjastoa. Snort julkaistiin virallisesti Packet Stormin kotisivuilla 22.12.1998. Ensimmäinen julkaistu versio sisälsi noin 1600 riviä koodia ja vain kaksi tiedostoa. [25, s. 33-34.]

Vuonna 2001 Roesch perusti yrityksen nimeltä Sourcefire. Sourcefire jatkoi Snortin kehittämistä kaupallisesti vuoteen 2013 asti, kunnes Cisco Systems osti Sourcefiren 2,7 miljardilla eurolla heinäkuussa 2013. Ennen myyntiä Sourcefire työllisti noin 560 työntekijää. [13.; 11.; 27.] Vuoden 2016 tammikuuhun mennessä Snort on ladattu yli neljä miljoonaa kertaa ja sillä on viisi sataa tuhatta rekisteröitynyttä käyttäjää. [13.; 11.; 27.]

4.2 Snortin pakettien käsittely

Pakettien käsittely on koko Snortin toiminnan mielenkiintoisin osa-alue. Snort suorittaa kaapatulle paketille viisi eri vaihdetta, joista ensimmäinen on paketin kaappaaminen. Paketin kaappauksen jälkeen Snort siirtää paketin dekooderille (vaihe kaksi), jonka jälkeen paketille tehdään esiprosesseja (vaihde kolme). Jos esiprosessit eivät luo paketista hälytystä, Snort siirtää paketin tunnistusmoottorille (vaihde neljä), joka vertaa pakettia tallennettuihin sääntöihin. Jos kaapattu paketti vastaa tallennettua sääntöä, paketti siirretään outputille (vaihe viisi), joka tallentaa paketin ja luo siitä hälytyksen. Kuvassa 1 havainnollistetaan kaapatun paketin matka Snortin lävitse.



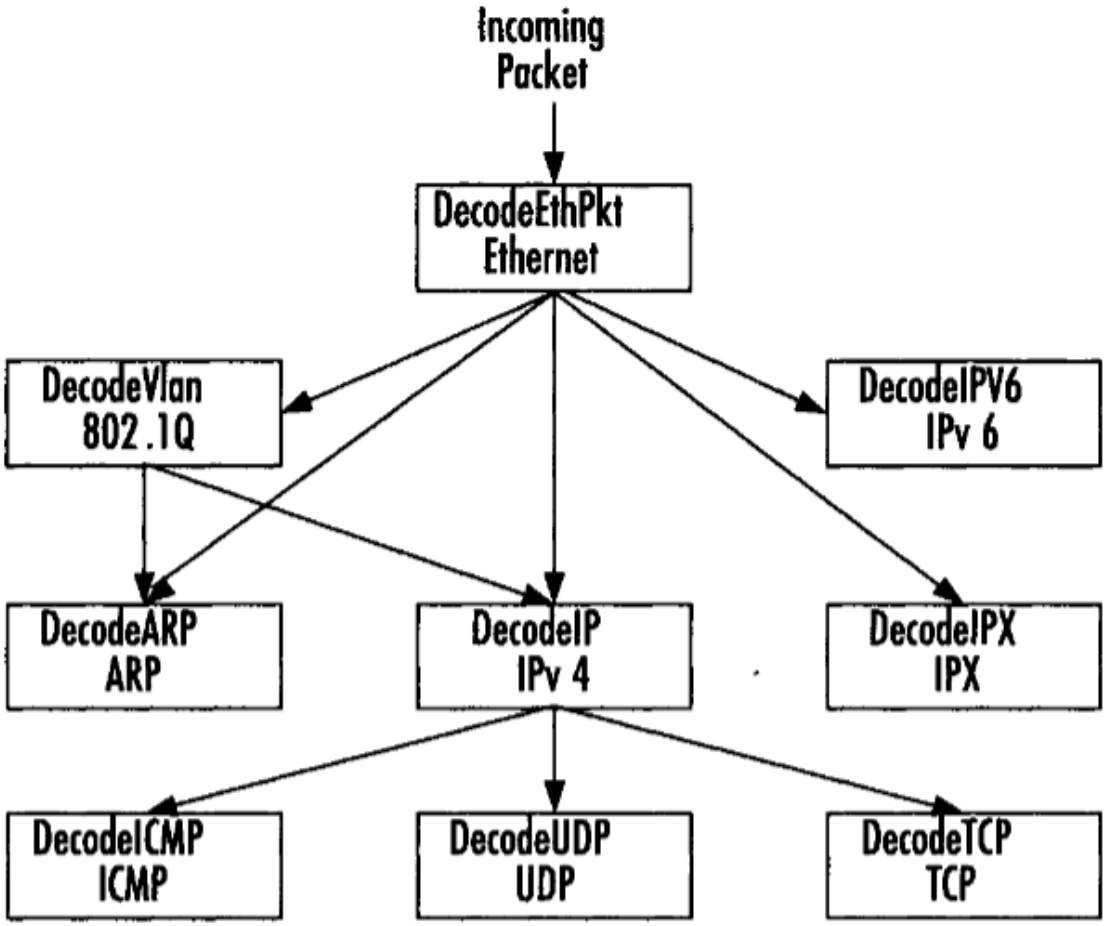
Kuva 1. Snortin pakettien käsittely. [58, s. 13.]

4.2.1 Paketin kaappaaminen (Libpcap)

Snort käyttää paketin kaappaamiseen pcap-kirjastoa, eli Linuxissa libpcap ja Window:ssa WinPcap. Libpcap-kirjastoa hyödyntää myös muun muassa TcpDump. Libpcapillä kaapattu paketti on täysin käsittelemätön siirtokerroksen paketti esimerkiksi Ethernet-kehys. Tämä paketti tulee purkaa, ennen kuin Snort pystyy tunnistamaan haitallisen toiminnan. [58, s. 14.]

4.2.2 Paketin tulkinta eli purkaminen (Decoding)

Paketin kaappaamisen jälkeen se siirretään Snortin dekoodereille. Koska Snort tukee useiden eri pakettien purkamista, paketin siirtokerroksessa käytetty protokolla (esim Ethernet, 802.11) määrittelee mitä dekooderia Snort käyttää. Tämän jälkeen, paketti siirretään esimerkiksi IP-paketin dekooderille, josta se siirtyy TCP, ICMP tai UDP dekoodereille. Lopulta kun paketti on purettu kokonaan, se voidaan siirtää esiprosesseille. Kuvassa kaksi nähdään ethernet-paketin dekooderit. [8, s. 183.; 58, s. 14.]



Kuva 2. Kaapatun ethernet-paketin matka dekooderien lävitse. [8, s. 184.]

Suurin osa internetissä kulkevista paketeista on ethernet-pohjaisia, joten käyn tässä UDP-paketin (ethernet-kehyksessä) matkan dekooderien lävitse.

Koska paketti oli ethernet-kehyksessä, ensimmäisenä Snort kutsuu funktiota *"DecodeEthPkt"*, (esim 802.11 paketeille käytetään *"DecodeIEEE80211Pkt"* -funktiota). Tältä funktiolta saadaan tieto paketin *"EtherType"* arvo. Ethernet-kehyksen ylätunnisteen *"EtherType*" kentän arvon (esimerkiksi, IPv4) mukaan paketti ohjataan seuraavalle dekooderille. Koska kaapattu paketti oli UDP, kentän arvo vastaa IP-protokollaa. Tämän jälkeen funktio *"DecodeEthPkt"* kutsuu *"DecodeIP"* –funktiota. Koska paketti oli UDP niin *"DecodeIP"* kutsuu lopuksi *"DecodeUDP"* –funktiota. [8, s. 184.]

4.2.3 Esiprosessit (Preprosessors)

Jos Snort on konfiguroitu toimimaan IDS:nä, purkamisen jälkeen kaapatulle paketille toteutetaan esiprosesseja, koska muuten Snort vain tallentaa kaikki paketit purkamisen jälkeen.

Esiprosessit pystyvät luokittelemaan ja tunnistamaan haitallisia paketteja ja näin siirtää paketti suoraan Snortin outputille. Nämä esiprosessit ajetaan paketille, ennen kuin paketti siirretään tunnistusmoottorille. [56.; 58, s. 14.]

Esiprosessien tarkoituksena on vähentää Snortin tuottamaa suoritin-kuormaa. Suodattamalla paketteja ennen kuin ne siirretään tunnistusmoottorin analysoitavaksi. [58. s, 200.] Näitä esiprosesseja hallinnoidaan Snortin konfiguraatiotiedostossa, *”snort.conf”*. Esimerkiksi Snortista löytyy esiprosessi *”portscan”,* jonka avulla voidaan huomata Nmapilla tehtyjä porttiskannauksia, asettamalla kyseinen esiprosessi hälyttämään, jos samasta IP-osoitteesta käydään lävitse useita eri portteja tietyssä ajassa. [58, s. 210.]

Vaikka libpcap tarjoaa mahdollisuuden useamman paketin käsittelyyn samanaikaisesti, Snort kykenee käsittelemään vain yhden paketin kerrallaan. Jos Snort käyttää liikaa aikaa yhden paketin analysointiin, alkaa se tiputtamaan paketteja ja mahdolliset haitalliset paketit voivat päästä Snortin ohitse. [8, s. 183.]

4.2.4 Tunnistusmoottori (Detection engine)

Kun kaapattu paketti on purettu ja se on käynyt esiprosessien lävitse, se siirretään tunnistusmoottorille. Tässä vaiheessa Snort vertaa kaapatun paketin dataa sääntöihin ja etsii niistä yhteneväisyyksiä. [58.]

Snort vertaa ensimmäisenä kaapatun paketin protokollaa sääntöihin. Jos paketti on tullut TCP-protokollana, Snort ei huomioi lainkaan sääntöjä, jotka koskevat UDP:ta. Tämän jälkeen Snort vertaa lähde- ja kohdeosoitteita. Jos nämä kaikki vastaavat kaapattua pakettia, Snort vertaa vielä sääntöön liitettyjä sormenjälkiä pakettiin. [31.]

4.2.5 Hälyttäminen (Output)

Jos esiprosessit tai tunnistusmoottori huomaa haitallisen paketin, pakettista luodaan hälytys. Kaapattu paketti, joka aiheutti hälytyksen, voidaan tallentaa joko ASCII-muodossa, binäärimuodossa tai paketti voidaan tallentaa suoraan tietokantaan. Hälytysten tallentamiseen suositellaan käyttämään binääritiedostoa, koska tallentaminen tietokantaan kuluttaa huomattavasti enemmän suoritin-aikaa ja täten pienentää hukattujen pakettien määrää.

4.3 Snortin lisäosat

Vaikka Snort toimii yksinään, on siihen hyvä asentaa lisäosia, joilla pystytään tehostamaan Snortin toimintaa. Alla käyn lävitse muutaman hyödyllisen lisäosan, jotka tehostavat Snortin toimintaa.

4.3.1 Barnyard

Koska Snort voidaan asentaa tallentamaan hälytyksiä binäärimuodossa, tarvitaan sovellus, joka muuntaa hälytyksen ihmiselle helpommin luettavaan muotoon esimerkiksi tietokantaan tai erilliselle syslog-palvelimelle. Tähän tarkoitukseen kehitettiin Barnyard.

Vaikka Snort pystyy itsekin tallentaman paketit tietokantaan, Barnyard voi hoitaa asian Snortin puolesta. Tämä tarkoittaa, että tallentaminen binäärimuotoon on nopeampaa, ja Snortille jää enemmän aikaa verkkoliikenteen pakettien analysointiin. Käytännössä Barnyard odottaa, että Snort luo hälytyksen paketista ja tallentaa sen tiedostoon. Tämän jälkeen Barnyard tallentaa kaapatun paketin esimerkiksi MySQL-tietokantaan. Toisin kuin Snort, joka vaatii jopa root-tason oikeudet toimiakseen, Barnyard ei tarvitse suurempia oikeuksia järjestelmään. Barnyardin tehtävä on vain käsitellä Snortin kaappaamia paketteja. Barnyard uusin versio on julkaistu nimellä Barnyard2. [8, s. 647.; 28.; 16.; 17.]

4.3.2 PulledPork

PulledPork on Snortille kehitetty sovellus, jonka avulla voidaan pitää Snortin sääntöjä ajan tasalla. PulledPork lataa automaattisesti Snortille julkaistut uusimmat säännöt. Pul-

ledPork vaatii toimiakseen niin sanotun OinkCoden, jonka jokainen Snortin sivuilla rekisteröitynyt käyttäjä saa. OinkCoden avulla pystytään lataamaan uusimmat säännöt Snortille. PulledPork on Perl-kielellä ohjelmoitu sovellus, joten palvelimen on tuettava kyseistä kieltä. [10.; 14.]

4.3.3 Snorby

Snortin luomien hälytysten seuraaminen on graafisena huomattavasti helpompaa, joten Snortille on kehitetty lukuisia ”frontend”-sovelluksia. Näistä suosituimmat ovat Snorby ja Base. Snorby on ohjelmoitu Ruby on Rails:iä käyttäen, joten palvelimelta tulee löytyä tuki Rubylle. Snorby on myös täysin avointa lähdekoodia, joten sovelluksen voi ladata suoraan Githubista. Snorby tulee käyttää samaa tietokantaa, johon Barnyard on tallentanut hälytykset, joten hälytykset saadaan näkymään selaimella. [29.]

4.3.4 Daemonlogger

Vaikka Daemonlogger ei ole Snortin lisäosa, se on silti mainitsemisen arvoinen sovellus. Daemonlogger on myös Martyn Roeschin kirjoittama, Libpcap-kirjastoa käyttävä pakettien kaappaaja ja toistaja.

Daemonlogger voi toimia kahdella eri tavalla, pakettien kaappaajana ja pakettien uudelleenlähettäjänä (toistajana). Kaappaajana Daemonlogger tallentaa paketit suoraan kiintolevylle. Toinen hyödyllinen toiminta Daemonloggerissa on se, että sillä pystyy lähettämään kopion jokaisesta kaapatusta paketista toiseen verkkokorttiin (interface). Esimerkiksi kaikki paketit jotka kulkevat rajapinnasta eth0, voidaan uudelleen lähettää rajapintaan eth1. Daemonlogger on Sourcefiren (Ciscon) omistama tavaramerkki, mutta se on avoimen GPLv2-lisenssin alaisuudessa. [59.]

5 Snortin säännöt

Snortin toiminnan kannalta tärkein osa ovat säännöt. Ilman kunnollisia sääntöjä Snort ei pysty huomaamaan hyökkäysyrityksiä.

Snortille voidaan tehdä yksinkertaisia sääntöjä esimerkiksi säännön joka hälyttää tulevasta ICMP-paketista, tai todella monimutkaisia ja tietenkin kaikkea siltä väliltä. Snortille

on olemassa todella paljon valmiita sääntöjä, jotka voidaan ladata suoraan Snortin palvelimilta. Sääntöjen muokkaaminen on kuitenkin todella tärkeää, koska säännöillä voidaan vaikuttaa suoraan Snortin tehokkuuteen väärien hälytysten ja aiheellisten hälytysten luomisessa. [8, s. 299.; 30.]

Snortin säännöt tallennetaan *"snort.rules"*-tiedostoon. Tiedostoa voidaan ylläpitää PulledPorkilla tai manuaalisesti. Snortille on myös hyvä luoda yksilöityjä sääntöjä tiedostoon *"local.rules"*. Yksilöidyissä säännöissä voidaan vahtia tiettyjen avainsanojen esiintymistä paketeissa. Esimerkiksi, yrityksellä on *"salaisuudet.txt"*-nimisiä tiedostoja, joita ei saa vuotaa ulkoverkkoon. Jos yrityksen palvelimilta yritetään ladata FTP:llä näitä tiedostoja, yksilöidyllä säännöllä se huomataan, ja Snort tekee hälytyksen.

Snort sisältää myös tiedoston, jossa on mustalla listalla IP-osoitteita. Mustaa listaa ylläpitää TALOS (entiseltä nimeltään VRT), joka on ryhmä johtavia tietoturva-asiantuntijoita. TALOS on Ciscon omistama tavaramerkki. Listaan voi myös lisätä omia mustalistattuja IP-osoitteita. [49.]

5.1 Sääntöjen koostumus

Snortin säännöt koostuvat kahdesta osasta: *"tunnisteesta" (Header)* ja *"vaihtoehdoista" (Options)*. Tunniste sisältää sen, mitä protokollaa sääntö koskee (esimerkiksi TCP tai UDP), sekä paketin lähde- ja kohdeosoitteen ja porttinumeron. Tämän jälkeen vaihtoehdoissa määritetään tietueet, joita Snort vertaa kaapattuihin paketteihin. Taulukossa näkyy Snortin sääntöjen osat. [58, s. 180.]

Tunniste (Header)					Vaihtoehdot (Options)
Toiminta	Protokolla	Lähdeosoite ja portti	Suunta	Kohdeosoite ja portti	
alert	udp	$EXTERNAL_NET any	->	$HOME_NET any	(msg:"MALWARE-CNC TRUFFLEHUNTER SFVRT-1013 attack attempt"; sid:29308; gid:3; rev:2; classtype:trojan-activity; metadata: engine shared, soid 3|29308;)

5.2 Tunniste

Tunnisteen ensimmäiseen kohtaan toiminta (action) voidaan määritellä:

- ”alert”, kun halutaan, että Snort luo hälytyksen, jos sääntö vastaa kaapattua pakettia.
- ”log”, jos sääntöä vastaava paketti halutaan vain tallentaa.
- ”pass”, jos tietystä säännöstä tulee paljon vääriä positiivisia, kyseinen komento jättää säännön huomioimatta.
- ”activate” tai ”dynamic” jotka toimivat yhdessä. ”Dynamic”-sääntö voidaan mieltää vain sääntönä, jonka tarkoituksena on esimerkiksi tallentaa kaapatut paketit. ”Active”-säännöllä määritellään milloin ”dynamic”-sääntö tulee voimaan.

Esimerkki *”activate”* ja *”dynamic”* säännön toiminnasta:

```
activate tcp any any -> $HOME_NET 22 (content:"/bin/sh"; activates:1; msg:"Possible SSH buffer overflow"; )

dynamic tcp any any -> $HOME_NET 22 (activated_by:1; count:100;)
```

Esimerkissä Snortille on määritetty sääntö, jossa seurataan komennon *"/bin/sh”* ajamista portissa 22 (määritellyssä kotiverkossa). Jos *”activate”* sääntö muodostaa hälytyksen, siirtyy Snort automaattisesti *”dynamic”*-sääntöön. Tässä säännössä Snort tallentaa seuraavat 100 pakettia *(count: 100)*, jotka tulevat portista 22 valvottavaan verkkoon. [30.; 48, s. 240.]

Säännön tunnisteen protokolla-kohdassa määritellään, mitä protokollaa sääntö käyttää. Tällä hetkellä Snort tukee neljää eri protokollaa, jotka ovat TCP, UDP, ICMP ja IP. Säännön suunta-kohdassa valitaan, kumpi puolista on lähdeosoite ja kumpi kohdeosoite. Nuolella -> tarkoitetaan, että vasemmalla puolella on lähdeosoite ja oikealla kohdeosoite. Snort tukee myös kaksisuuntaisia sääntöjä, eli molemmat osoitteista voivat olla kohde- tai lähdeosoitteita. [30.]

5.3 Vaihtoehdot (options)

Sääntöjen vaihtoehdoilla päätetään, mihin tietoihin Snort vertaa kaapattuja paketteja. Tässä osassa sääntöä määritetään kaikki haitallisten pakettien tunnistamiseen vaadittavat tiedot. Vaihtoehdot jaetaan neljään pääkategoriaan. Ensimmäinen kategorioista on ”general”, toinen ”payload”, kolmas ”non-payload” ja neljäs ”post-detection”.

5.3.1 General

General valinnat vaikuttavat informaatioon, jota Snortin säännöt tarjoavat, eli näillä määritelmillä ei ole suoraa vaikutusta säännön aktivoitumiseen. General-valinnoissa määritetään muun muassa ilmoitus <msg>-säännölle, kuten *”msg:"PING INC";”*. Jos sääntö huomataan, Snort tallentaa hälytykseen määritellyn ilmoituksen. Näin hälytyksiä on helpompi seurata ja listata.

Ilmoitusten lisäksi general-valinnoissa määritellään jokaiselle säännölle oma tunniste ID (sid). Ilman ID:tä Snort ei suostu käyttämään sääntöjä.

> ERROR: /etc/snort/rules/local.rules(4) Each rule must contain a rule sid.

Säännön ID määritellään <sid>-valinnalla. Omissa säännöissä tulisi aina käyttää ID-arvoja, jotka alkavat miljoonasta. Alle miljoonan ID:t ovat varattu Snortin omille säännöille. Esimerkiksi oman säännön ID voi olla *”sid:1000009;”*.

ID:n lisäksi säännöille tulisi aina määrittää tarkistusnumero <rev>, jotta Snort tietää, onko sääntö ajan tasalla, esimerkiksi *”rev:1;”*. Tarkistusnumerolla pystytään seuraamaan, että sääntö on ajan tasalla. Jos säännöllä ei ole tarkistusnumeroa, sääntö ei tallennu oikeassa muodossa PulledPorkin muodostamaan *”sig-msg.map”* tiedostoon. Barnyard2 ei käynnisty, jos kyseisessä tiedostossa on syntaksivirheitä. Ohessa on virhe palvelimen syslogista.

> snort barnyard2[12376]: FATAL ERROR: [ParseSidMapLine()], File [/etc/snort/**sid-msg.map**],#012Error in map definition [1 || 10000042 || || suspicious-login || 0 || FTP user ftpuser] for value []

Säännön classtypellä voidaan määritellä sääntö tiettyyn kategoriaan. Näillä kategorioilla sääntö voidaan määrittää tiettyyn prioriteettiin. Snortin säännöille on olemassa neljä eri prioriteettia, jotka ovat ”high”, ”medium”, ”low”, ja ”very low”. Esimerkiksi *”classtype:icmp-*

event;", on ”low” tason hälytys, kun taas *”classtype:policy-violation”* on ”high” tason hälytys. Tarkempi lista classtype-valinnoista löytyy liitteestä yksi. Jos säännölle ei valita classtypeä, on sääntö automaattisesti NOCLASS. [32.]

5.3.2 Payload

Payload eli hyötykuorma-valinnoissa vaikutetaan paljon sääntöjen tarkkuuteen. Hyötykuormavalintoja on noin 50 erilaista, joista esittelen tärkeimmän.

Content on yksi sääntöjen tärkeimpiä ominaisuuksia. Contentin avulla paketin hyötykuormasta voidaan etsiä tiettyä sisältöä vastaavaa tietoa, jonka jälkeen Snort pystyy luomaan hälytyksen. Jos contentissa määritetty tieto löytyy mistä tahansa osasta pakettia, Snort voi luoda hälytyksen, esimerkiksi säännöllä:

```
alert tcp any any -> $HOME_NET 21 (msg:"FTP user ftpuser"; content:"user
ftpuser"; nocase; sid:10000042; rev:001; classtype:suspicious-login;)
```

Edellä mainitussa säännössä seurataan FTP-liikennettä (portti 21), ja luodaan hälytys, kun käyttäjä ”ftpuser” yrittää tai kirjautuu palvelimelle. Säännössä oleva *”nocase;”* tarkoittaa, että contentissa ei huomioida eroja isojen tai pienien kirjaimien välillä, eli myös käyttäjä ”FtPuSer” huomataan. Ohessa on Snortin luoma hälytys FTP-käyttäjän sisäänkirjautumisesta käyttämällä sääntöä 10000042. [33.]

```
04/09-16:19:50.272750  [**] [1:10000042:1] FTP user ftpuser [**] [Classification:
An attempted login using a suspicious username was detected] [Priority: 2] {TCP}
84.249.XXX.XXX:1113 -> 83.136.XXX.XXX:21
```

5.3.3 Non-payload

Non-payload valinnoilla voidaan tarkistaa suoraan paketin tietoja. Kyseisillä valinnoilla voidaan seurata suoraan paketin lippuja (flags, kuten SYN, ACK, FIN). Lippujen lisäksi voidaan seurata paketin elinaikaa (TTL). Esimerkkinä on sääntö, jonka avulla Snort luo hälytyksen havaittuaan icmp-paketin. [34.]

```
alert icmp any any -> $HOME_NET any (msg:"PING INC"; icode:0; itype:8;
sid:1000001; rev:001; classtype:icmp-event;)
```

5.3.4 Post-detection

Post-detection:lla tarkoitetaan valintoja, jotka tehdään, kun Snort on huomannut tiettyä sääntöä vastaavan paketin.

Post-detection valinnoilla voidaan esimerkiksi tallentaa 11 seuraavaa pakettia, jotka tulevat tietyn säännön aktivoitumisen jälkeen. Post-detection valinnoissa on vaihtoehto *”detection_filter”*, jonka avulla sääntöjä voidaan muokata hälyttämään vasta, kun sääntöä on rikottu X kertaa. Seuraavassa esimerkissä on sääntö, joka hälyttää, kun SSH:lla on yritetty kirjautua 30 kertaa 60 sekunnin sisään. [35.]

```
alert tcp any any > $HOME_NET 22 (msg:"SSH Brute-force Attack"; flow:established,to_server; content:"SSH"; nocase; offset:0; depth:4; detection_filter:track by_src, count 30, seconds 60; sid:1000001; rev:1;)
```

6 Sisällönhallintajärjestelmät (CMS)

Sisällönhallintajärjestelmät tarjoavat helpon työkalun kotisivujen luomiseen ja päivittämiseen. Sisällönhallintajärjestelmillä voidaan luoda nopeasti hyvin näyttäviä sivuja ilman ohjelmointitaitoa. Suuren suosion takia rikolliset etsivät jatkuvasti haavoittuvaisuuksia, joita he voivat hyödyntää. Rikolliset voivat esimerkiksi lisätä murretuille sivuille haitallisia uudelleenohjauksia, joilla ohjataan vierailevat ihmiset rikollisten sivuille, josta vierailijat saadaan lataamaan automaattisesti haitallisia tiedostoja.

Viestintävirasto havaitsee Suomessa kymmeniä murrettuja sivuja viikoittain. Viestintävirasto on yhteydessä murretuista sivuista sivujen palveluntarjoajaan. Tämän jälkeen palveluntarjoaja estää sivuston käytön, yleensä estämällä suoritusoikeudet saastuneilta tiedostoilta. Eston jälkeen palveluntarjoaja ilmoittaa sivun ylläpitäjälle murrosta, ja häntä opastetaan sivujen siivoamisessa. [36.]

Sisällönhallintajärjestelmistä Suomessa helmikuussa 2016 suosituimmat järjestelmät olivat Wordpress ja Joomla! yhteensä yli 50 % osuudella kaikista hallintajärjestelmillä toteutetuista sivuista. [37.]

Suosituin sisällönhallintajärjestelmä on Wordpress, joka on avoimeen lähdekoodiin perustuva sovellus. Wordpress on ohjelmoitu PHP:llä. Tietojen tallentamiseen Wordpress käyttää MySQL-tietokantaa. Wordpress on kaikkein suosituin sisällönhallintajärjestelmä

noin 50 % prosentin osuudella kaikista maailman verkkosivuista. Tämän takia suurin osa murretuista sivuista on juuri Wordpress-pohjaisia. [23.]

7 Web-sovelluksien yleisimmät uhat

Web-sovelluksiin on olemassa lukuisia eri hyökkäyksiä, joten tässä kappaleessa käyn lävitse yleisimmät uhat joita web-palvelimille toteutetaan.

7.1 Cross-site scripting (XSS)

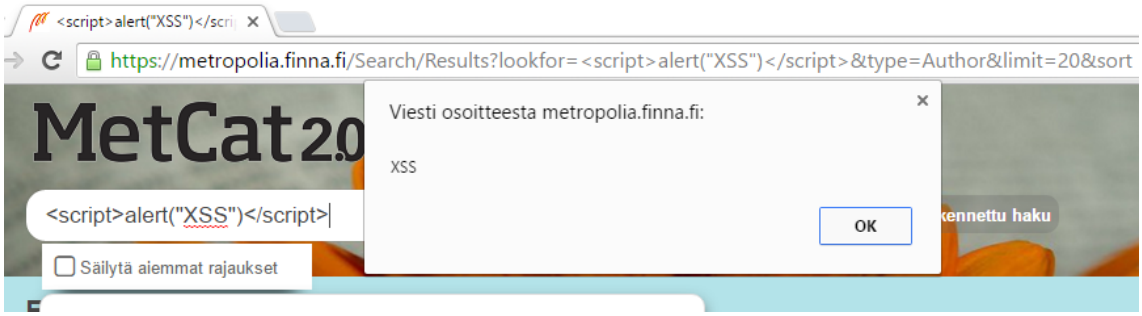
XSS on hyvin vanha haavoittuvuus, ensimmäiset tunnetut hyökkäykset ovat jo vuodelta 1996, kun JavaScript alkoi yleistymään sen aikaisissa web-sivustoissa. JavaScriptin avulla pystyttiin luomaan esimerkiksi pop-up ikkunoita ja laajenevia valikkoja. Tämä mahdollisti myös hyökkääjille uusia tapoja suorittaa vahingollista toimintaa. Hyökkääjät esimerkiksi pystyisivät ohjaamaan käyttäjän web-palvelimelta toiselle, varastamaan käyttäjien salasanoja (jotka kirjotiettiin hyökkääjien luomiin HTML-kenttiin) ja ottamaan käyttöönsä uhrin evästeet (cookies) jotka asiallinen sivu oli luonut. XSS-hyökkäys tarvitsee toimiakseen haavoittuvaisen web-palvelimen, uhrin jolta yritetään varastaa tietoja ja hyökkääjien oman web-palvelimen, johon uhri ohjataan. [60, s. 2.]

XSS-hyökkäykset voidaan jakaa heijastettuihin (ei-pysyvät) ja pysyviin hyökkäyksiin. [61, s. 433.]

7.1.1 Heijastetut hyökkäykset

Näissä hyökkäyksissä haitallista on se, että itse scriptiä ei tallenneta web-palvelimelle ollenkaan, vaan hyökkäyksessä hyödynnetään osoitekenttää (URL). Tämän hyökkäyksen avulla hyökkääjät voivat ohjata uhrin haluamalleen sivulle ilman, että uhri edes huomaa uudelleenohjausta, koska osoiterivi alkaa luotettavan sivun osoitteella. Hyökkäys onnistuu, jos uhri avaa haitallisen linkin, esimerkiksi:

https://metropolia.finna.fi/Search/Results?look-for=<script>alert("XSS")</script>&type=Author&limit=20&sort



Kuva 3. Pop-up ikkuna avautuu, kun uhri avaa linkin.

Kun haitallinen linkki on luotu, se voidaan välittää uhrille, esimerkiksi sähköpostin kautta. Uskoisin, että jos uhri huomaa linkin alkavan http://turvallinensivu.fi/.../.../.../ hän todennäköisemmin avaa sen, vaikka linkki sisältää haitallisen skriptin. Hyökkääjät voivat kopioida turvallisen sivun ulkoasun omalle palvelimelle, joten kohde ei välttämättä huomaa olevansa väärällä sivulla, vaikka hänet olisi ohjattu haitallisen linkin avulla sinne. [61, s. 434-436.]

7.1.2 Pysyvät hyökkäykset

Pysyvissä hyökkäyksissä hyökkääjä onnistuu syöttämään vahingollista koodia suoraan web-palvelimelle. Esimerkiksi hyökkääjä pystyy kommenttikentän kautta lisäämään sivuille haitallisia kenttiä tai estämään muita ihmisiä kommentoimasta.

Pysyviä XSS-hyökkäyksiä pidetään suurempana uhkana (kuin ei-pysyviä) turvallisuudelle, koska niillä pystytään vaikuttamaan useampaan henkilöön kerralla. [61, s. 438.]

7.2 SQL-injektio

SQL-injektiolla tarkoitetaan hyökkäystä, joka on kohdistettu sovelluksiin, jotka keskustelevat tietokannan kanssa. Esimerkiksi, jos sovellus ei puhdista mahdollisia haitallisia tai virheellisiä merkkejä tietokantakyselyistä, hyökkääjä voi ujuttaa haitallisen SQL-kyselyn

oikean kyselyn sekaan. Tällä haitallisella kyselyllä tunkeutuja voi pakottaa tietokannan paljastamaan arkaluontoista tietoa, esimerkiksi salasanoja tai tallennettuja yhteystietoja tai jopa henkilötunnuksia.

Esimerkiksi oheisella HTTP-kyselyllä voidaan saada PostNuke-sisällönhallintajärjestelmää kertomaan käyttäjien salasanoja. Tämä hyökkäys hyödyntää juuri web-sovelluksen haavoittuvuutta. [6, s. 5-6.; 24.]

```
http://[target]/[postnuke_dir]/modules.php?op=modload&name=Messa-
ges&file=readpmsg&start=0%20UNION%20SELECT%20
pn_uname,null,pn_uname,pn_pass,pn_p
```

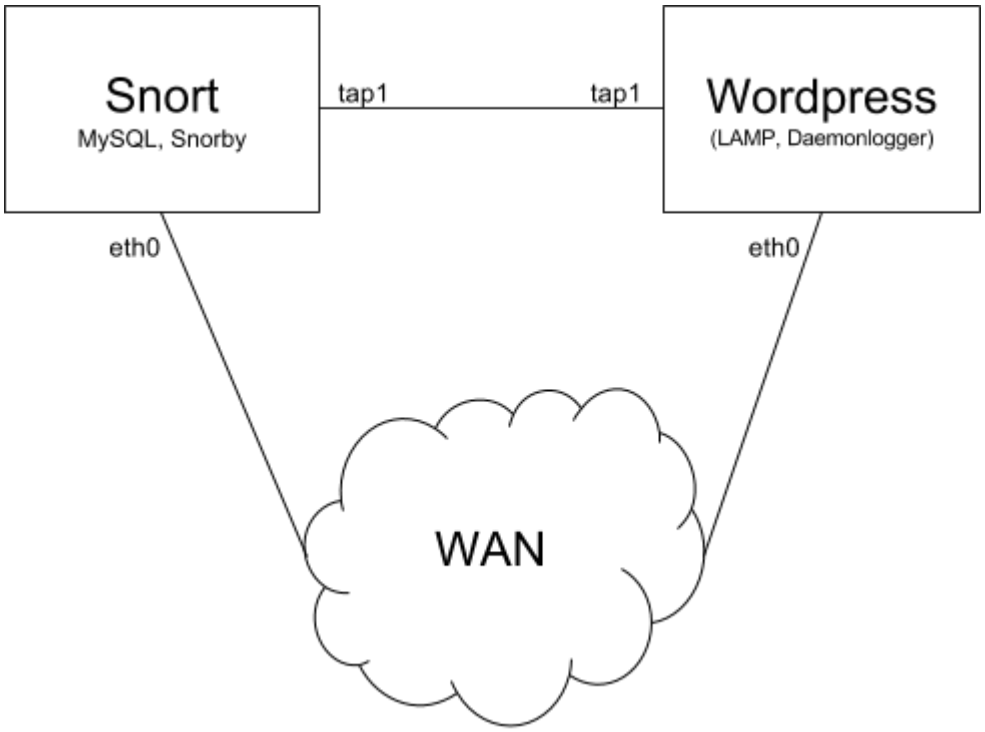
8 Asentaminen

8.1 Testiympäristö

Pystytin testiympäristön UpCloudin virtualisointialustalle. UpCloudiin luotuja virtuaalipalvelimia voi hallinnoida UpCloudin tarjoaman web-käyttöliittymän kautta. UpCloudin kautta virtuaalipalvelin pystytetään noin 45 sekunnin kuluessa palvelimen luonnin jälkeen. [53.]

UpCloud veloittaa asiakkaitaan tuntipohjaisella laskutuksella, käytettyjen resurssien pohjalta. Kun palvelin on sammutettuna, kustannuksia tulee vain levykuvien säilyttämisestä, ja mahdollisista varatuista IPV4-osoitteista, IPV6-osoitteet ovat ilmaisia. Omalla käyttäjätunnuksella on sisäinen tili, josta maksetaan palvelimien kustannukset. Tämän ominaisuuden avulla palvelimen resursseja voidaan kasvattaa, ja laskea omien tarpeiden mukaisesti. [54.]

Molemmat ympäristössä käytettävistä palvelimista toimivat UpCloudin Lontoon palvelinsalissa. Kaikki UpCloudin saman käyttäjätunnuksen alla olevat palvelimet ovat samassa sisäverkossa, vaikka palvelimet sijaitsisivat eri konesaleissa.



Kuva 4. Testiympäristö.

Kuva 4 havainnollistaa pystytettyä testiympäristöä, jossa on kaksi palvelinta, Snort ja Wordpress. Molemmat ovat kytkettyinä eth0-porteista ulkoverkkoon. Wordpressille on asennettuna Daemonlogger, joka kuuntelee palvelimen ulkoverkonporttia (eth0) ja kopioi kaikki lähtevät ja saapuvat paketit tunnelia (tap1) pitkin Snortille.

Palvelinten välinen sisäverkko on tunneloitu L2-tason tunnelilla, koska Wordpress-palvelimessa asennettu Daemonlogger ei tue pakettien kopiointia L3-tunnelin yli. Tunnelia käytetään myös siksi, koska UpCloudin palomuuri ei salli pakettien kopiointia kahden palvelimen välillä. Sama paketti, joka tulee Wordpress-palvelimelle eth0-portista, ei voida välittää suoraan palvelimen eth1-portista Snort-palvelimen eth1-porttiin.

8.2 Palvelin: Snort

Asensin Snortin Debian 7.8-käyttöjärjestelmälle, josta löytyi valmis levykuva UpCloudin levykuvista. Snortin palvelin sisältää neljä ydintä, 4096 Mt muistia ja 100 GB tallennustilaa.

8.2.1 Snort

Palvelimen paketinhallinnan päivittämisen jälkeen latasin Snortin tiedostot Snortin palvelimilta. Asennuksen aikana uusin versio oli 2.9.8.0. Pelkkä Snortin asentaminen onnistuu muutamalla komennolla, mutta Snortin saamiseksi toimimaan tehokkaasti NIDS:inä vaatii se enemmän asentamista ja konfigurointia. Asennuksen jälkeen siirsin Snortin tiedostot *”/etc/snort*” hakemistoon, jonne loin vielä erilliset hakemistot sääntöjä varten.

Snortin toiminnan kannalta tärkein tiedosto on *”snort.conf”*, joka sisältää kaikki Snortin konfiguraatiot. Kyseiseen tiedostoon määritetään esimerkiksi mistä hakemistosta Snortin säännöt löytyvät, mikä on valvottavan verkon IP-osoite ja mihin Snort tallentaa hälytyksen luomat paketit. Kyseisessä tiedostossa ei määritetä valvottavaa verkkoliitäntää (interfacea) vaan se määritetään erikseen vivulla –i, kun komento ajetaan. Snort tallentaa hälytyksen aiheuttaneet paketit *”snort.u2”*-tiedostoon, joka sijaitsee hakemistossa *”/var/log/snort”*.

Asennuksen lopuksi asetin vielä Snortin toimimaan daemonina, jotta sovellusta olisi helpompi hallita. Lisäämällä *”snort.service”* tiedoston *”/lib/systemd/system”* hakemistoon saadaan Snort käynnistymään automaattisesti daemonina, kun palvelin käynnistyy.

8.2.2 Barnyard

Snortin asentamisen jälkeen asensin palvelimelle Barnyard2-sovelluksen, jolla voidaan tallentaa Snortin tallentamat hälytykset (snort.u2-tiedostosta) MySQL-tietokantaan. Barnyardin saa ladattua suoraan Githubista.

Lataamisen jälkeen MySQL:lle tulee luoda uusi käyttäjä ja tietokanta, jota Barnyard käyttää hälytysten tallentamiseen. Loin aluksi tietokantaan kannan ”Snort” ja käyttäjän ”Snort” (myöhemmin muutin tietokanta toimimaan myös Snorbyn kanssa), jolla on täydet oikeudet kyseiseen tietokantaan. Barnyardin latauksen yhteydessä saadaan tiedosto *”create_mysql”* (schemas, kansion alla), jolla voidaan generoida Barnyardin vaatimat taulut tietokantaan.

Tietokannan jälkeen Barnyardin konfiguraatiotiedostoon *”barnyard2.conf”* tulee lisätä tieto tietokannasta ja käyttäjästä, jotta Barnyard tietää, mitä tietokantaa käytetään. Kyseiseen konfiguraatiotiedostoon tulisi olla vain luku- ja muokkausoikeudet *”root”*-käyttäjällä, koska tiedosto sisältää salasanan tietokantaan selkokielisenä.

Asentamisen jälkeen konfiguroin Barnyardin käynnistymään automaattisesti, kun palvelin käynnistetään. Kansioon *”/lib/systemd/system/”* loin tiedoston *”barnyard2.service”*, johon määritin Barnyardin käynnistyskomennon (vivulla –D Barnyard saadaan pyörimään daemonina). Käynnistyskomennon yhteydessä tulee määrittää Snortin luoman binääritiedoston polku, jotta hälytykset saadaan tallennettua tietokantaan (vivulla –f).

8.2.3 PulledPork

Snortin ja Barnyardin asentamisen jälkeen asensin palvelimelle PulledPorkin, jolla on mahdollista hallinnoida Snortin sääntöjä. Koska PulledPork on Perlillä ohjelmoitu sovellus, tulee palvelimella olla tuki kyseiselle kielelle. PulledPorkia ei myöskään löydy Debianin repoista, joten asentaminen tapahtuu suoraan Githubista. Insinöörityön aikana PulledPorkista oli julkaistu versio 0.7.2.

PulledPorkin käyttämä OinkCode (joka toimii API-avaimena sääntöjen lataamisen Snortin palvelimilta), tulee syöttää PulledPorkin omaan konfiguraatiotiedostoon *”pulledpork.conf”*. Konfiguraatiotiedostoon määritellään myös hakemisto, johon PulledPork tallentaa lataamansa Snortin säännöt. Eli säännöt ladataan automaattisesti *”/etc/snort/rules”*-hakemistoon. PulledPorkin lataamat säännöt menevät tiedostoon *”snort.rules”*.

Aina kun PulledPork lataa säännöt Snortin palvelimilta, sovellus päivittää tiedoston *”sid-msg.map”* jolle myös määritetään tallennuspolku konfiguraatiotiedostossa. Tähän tiedostoon PulledPork tallentaa säännöt Barnyardille helpommin luettavaan muotoon. Eli säännöt tallennetaan niiden *”msg:”* arvon mukaan. Ilman tätä tiedostoa Barnyard tallentaa Snortin hälytyksen tietokantaan sääntöjen *”gid:sid:rev”*:n mukaan, esimerkiksi (1:1000012:1).

PulledPork tallentaa lataamansa tar-paketin (joka sisältää säännöt) palvelimen *”/tmp/”* hakemistoon. Lataamisen jälkeen PulledPork automaattisesti purkaa paketin ja kopioi uusimmat säännöt konfiguraatiotiedostossa määritettyyn polkuun. PulledPork on suositeltava määrittää lataamaan säännöt automaattisesti, joten ajastin komennon (joka lataa

säännöt) tapahtumaan joka yö kello 04.00. Koska PulledPork ajetaan root-käyttäjällä, ajastettu komento tulee syöttää root-käyttäjän crontabiin.

8.2.4 Snorby

Viimeisenä asensin Snort-palvelimelle Snorbyn, jotta voisin saada mukavan frontend näkymän Snortin generoimille hälytyksille. Latasin myös Snorbyn Githubista, minkä jälkeen siirsin puretut tiedostot palvelimen *”/var/www/html/snorby”* kansioon.

Esivaatimusten asentamisen jälkeen Snorbylle tulee määrittää tietokanta ja tietokannan käyttäjä, josta Snorby hakee Barnyardin tallentamat hälytykset. Tämä tiedosto löytyy nimellä *”database.yml”*. Ensimmäisen kerran, kun Snorby ajetaan, *”database.yml”* tiedostoon tulee määrittää tietokannan root-käyttäjä, jotta Snorby pystyisi generoimaan vaaditut taulut tietokantaan.

Asennuksen aikana Snorby generoi automaattisesti tietokannan ”Snorby”, johon tarvittavat taulut määritellään. Asentamisen jälkeen loin erikseen tietokantaan käyttäjän ”Snorby”, jolle asetin täydet oikeudet juuri luotuun tietokantaan. Tämä uusi tietokantakäyttäjä ja tietokanta tallennettiin Snorbyn ja Barnyardin konfiguraatiotiedostoihin, jotta Snorby osaisi hakea oikeasta tietokannasta Barnyardin tallentamat hälytykset.

Olin aikaisemmin asentanut palvelimelle Apachen, joten seuraavaksi loin Apacheen uuden virtuaalihostin (vhost) *”snorby”*, joka kuuntelee porttia 80. Jotta vhostin saa käyttöön, sivu pitää vielä erikseen enabloida *”a2ensite”*-komennolla. Apachen uudelleenkäynnistämisen jälkeen Snorbyn hallintapaneeliin saa selaimella HTTP-yhteyden Snort-palvelimen IP-osoitteella.

Koska hallintapaneeli vaatii erillisen sisäänkirjautumisen (jotta hälytyksiä voi selata vain valtuutetut tahot), kirjautumissivu tulisi asentaa tukemaan vain HTTPS-yhteyksiä. Insinöörityön aikana Let’s Encrypt oli jo ”Public Beta” vaiheessa, joten päätin hyödyntää sitä työssä. Let’s Encryptin avulla saa sivuilleen ladattua virallisen sertifikaatin, jonka selaimet tunnistavat. Let’s Encrypt tarjoaa ilmaisia sertifikaatteja, jotta internet liikennettä saataisiin salattua enemmän. [55.]

Asensin Let’s Encryptin Githubin kautta komennolla

```
sudo git clone https://github.com/letsencrypt/letsencrypt
```

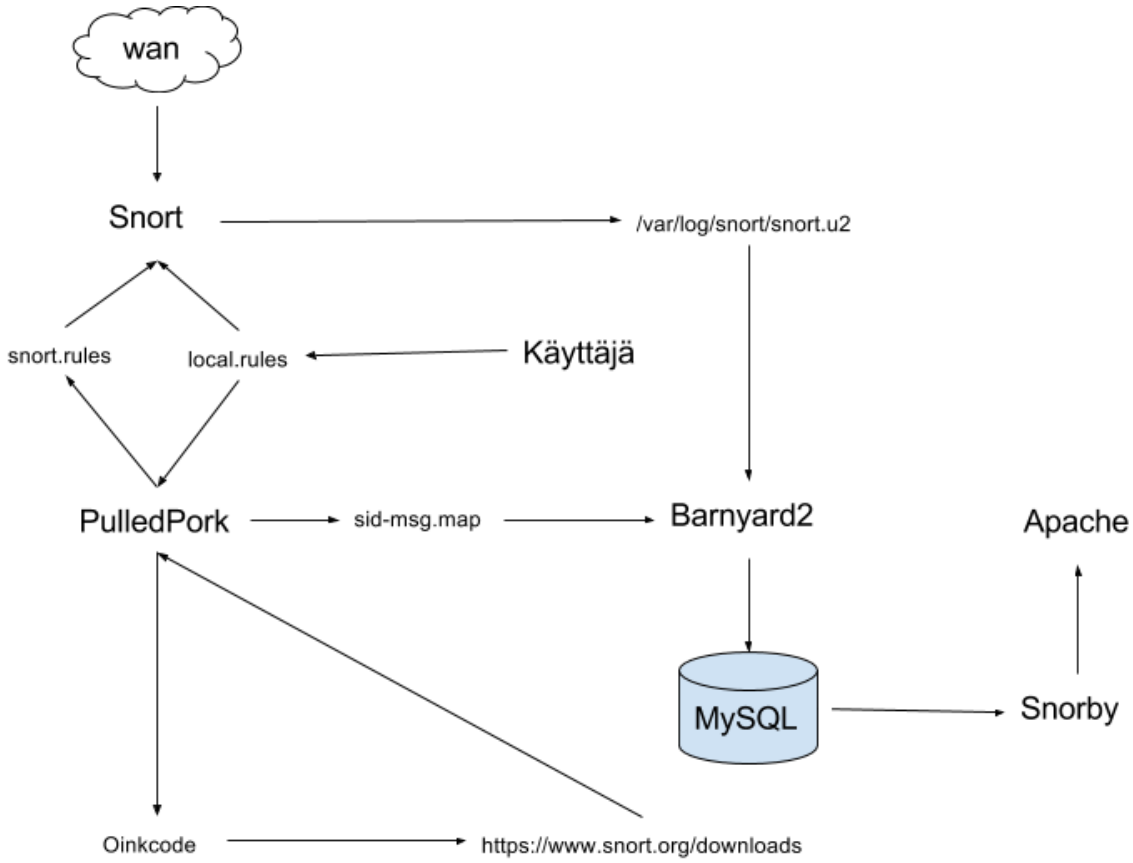
Lataamisen jälkeen loin sertifikaatin sivuille komennolla. [38.]

```
sudo ./letsencrypt-auto --apache -d snort.palvelin.biz
```

Asennuksen aikana HTTP-liikenteen voi pakottaa käyttämään salattua liikennettä. Let's Encryptin sertifikaatit ovat voimassa 90 päivää ja sertifikaatin voi uusia, jos sertifikaatti on menossa vanhaksi 30 päivän kuluessa. Voimassa olemassa olevan sertifikaatin voi uusia suoraan komennolla:

```
sudo ./letsencrypt-auto renew
```

Ajastin kyseisen komennon (crontab) tapahtumaan joka sunnuntai, jotta sertifikaatti ei pääse vanhentumaan. Lopuksi vielä määritin Snorbyn pyörimään daemonina, kuten Snortin ja Barnyardin. Kuvassa 5 havainnollistetaan Snortin toiminta lisäosien kanssa.



Kuva 5. Snort-palvelin.

8.3 Palvelin: Wordpress

Wordpress vaatii toimiakseen LAMP-stakin (Linux Apache MySQL PHP). Linuxina Wordpress palvelimessa toimii myös Debian 7.8 kuten Snortissa. LAMP-stackin lisäksi palvelimelle asennettiin Daemonlogger, jotta kaikki paketit, jotka tulevat Wordpress-palvelimella voitaisiin ohjata suoraan Snortille.

Palvelimena oli Upcloudin 20 dollaria kuukaudessa maksama paketti, joka sisälsi kaksi ydintä, 2048 Mt muistia ja 50 GB levytilaa.

8.3.1 Wordpress

LAMP-stackin asennuksen jälkeen latasin Wordpressin tiedostot Wordpress.org:n kotisivuilta. Lataamisen jälkeen siirsin tiedostot palvelimen *”/var/www/html/wordpress”* hakemistoon. Siirron jälkeen määritin Apachelle vielä vhostin, joka käyttää kyseistä tiedostohakemistoa.

Koska Wordpress vaatii toimiakseen tietokannan, loin MySQL-tietokantaan käyttäjän *”wp_user”*, jolla on täydet oikeudet tietokantaan *”wordpress”*. Luotu tunnus ja tietokanta tulee syöttää suoraan Wordpressin konfiguraatiotiedostoon *”wp-config.php”*. Tietokannan luomisen jälkeen Wordpressin asennus sujuu selaimella kyseisen palvelimen IP-osoitteen kautta.

8.3.2 Daemonlogger

Wordpress on täysin oma palvelin, johon en ollut asentanut Snorttia. Tämän takia Wordpress-palvelimen paketit tulisi erikseen lähettää Snortille analysoitavaksi. Tähän tehtävään sopii sovellus Daemonlogger, joka kopioi kaikki paketit halutusta verkkoportista ja lähettää paketit kopiona toiseen porttiin.

Koska Daemonlogger löytyy Debianin paketeista, niin sain asennettua sovelluksen suoraan ”aptitude”-komennolla. Daemonlogger tekee vaaditun tehtävän yhdellä komennolla, joka on:

```
sudo daemonlogger –i eth0 –o tap1 -d
```

Tässä –i-vivulla määritetään valvottava verkkoportti ja –o on ulosmenevä verkkoportti.

Koska halusin, että Daemonlogger käynnistyy aina automaattisesti palvelimen käynnistyksen yhteydessä, niin lisäsin kyseisen komennon *”rc.local”*-tiedostoon. Komentoon lisäsin vielä –d vivun, joka kertoo Daemonloggerille, että sovellus ajetaan daemonina.

8.4 OpenVPN

Koska Upcloudin palomuuri ei tue samojen pakettien lähettämistä suoraan sisäverkon koneiden välillä (eli pakettia ei voi reitittää Wordpress palvelimen eth1 portista Snort palvelimen eth1 porttiin), niin asensin palvelinten välille OpenVPN-tunnelin. Tunnelin avulla sain kytkettyä molempien palvelimien portit yhteen, jotta paketit kulkisivat suoraan palvelimelta toiselle. Valitsin OpenVPN, koska Snort ei tue PPTP-pakettien dekoodausta enää. Snort ei pysty uusimmissa versioissa käsittelemään PPTP-paketteja.

Kuten Daemonloggerin komennosta huomaa tunnelina oli *”tap1”* eikä *”tun1”*. Päädyin tähän sen takia, koska Daemonlogger ei toiminut L3-tunnelin ylitse. Eli kopioidut paketit eivät kulkeneet L3-tunnelin ylitse Snortille asti. *”Tap”*-portilla kahden palvelimen välinen yhteys sillataan verkon yli.

Määritin OpenVPN:n käynnistymään palvelimien *”rc.local”* tiedostossa, jotta tunneli avataan automaattisesti palvelimen käynnistyksen yhteydessä. OpenVPN pyörii myös Daemonina, jotta palvelin kykenee käynnistymään normaalisti. Esimerkissä oleva IP-osoite on toisen palvelimen eth1-portin osoite (sisäverkko).

```
openvpn --remote 10.2.2.245 --dev tap0 –daemon
```

8.5 Säännöt

Työn aikana tein omia sääntöjä, jotta voisin huomata tunnista toiminnan tehokkaammin. Ohessa esittelen muutaman luomani säännön ja kerron, miten ne toimivat.

8.5.1 XML-RPC

Hyökkääjät voivat käyttää XML-RPC:tä DDoS-hyökkäysten toteuttamiseen ja raakaa voimaa (Brute-force) Wordpress käyttäjien salasanojen arvaamiseksi XML-RPC:n avulla. [39.]

XML-RPC on API-avain, jonka avulla sovelluskehittäjät voivat tehdä erilaisia sovelluksia, jotka keskustelevat suoraan Wordpressin kanssa. Eli, jos käyttäjä haluaa moderoida Wordpressin kommentteja tai luoda uusia julkaisuja esimerkiksi iPhonelle kehitetyllä sovelluksella, niin puhelimen sovellus tarvitsee XML-RPC:n toimiakseen. Joten, jos kyseinen palvelu (XML-RCP) poistetaan käytöstä, kyseiset sovellukset lakkaavat toimimasta. [39.]

XML-RCP:llä on mahdollista toteuttaa useampi metodi yhdessä HTTP-pyynnössä. Tämä mahdollistaa sen, että hyökkääjät voivat kokeilla jopa satoja salasanoja yhdellä HTTP-pyynnöllä. Joten muutamalla pyynnöllä hyökkääjät voivat kokeilla jopa tuhansia eri salasanoja, mutta lokitiedostoihin jää vain muutama HTTP-pyyntö. Hyökkääjät voivat esimerkiksi hyödyntää *”wp.getCategories”* metodia, joka käyttää Wordpressin käyttäjätunnusta ja salasanaa. [40.; 41.]

Wordpress sisältää niin sanotun ”pingback”-ominaisuuden, jota XML-RPC hyödyntää. Tämä ominaisuus mahdollistaa hyökkääjille suojaamattomien Wordpressien hyödyntämisen omissa DDoS-hyökkäyksissä. Esimerkiksi Wordpressin pingback POST-pyynnön *”xmlrpc.php”* pyyntö näyttää tältä:

```
<?xml version 1.0?>

<methodCall><methodName>pingback.ping</methodName>

  <params>

    <param><value><string>http://kohde.fi</string></value></param>

    <param><value><string>http://anywordpresssite.com
</string></value></param>

</params></methodCall>
```

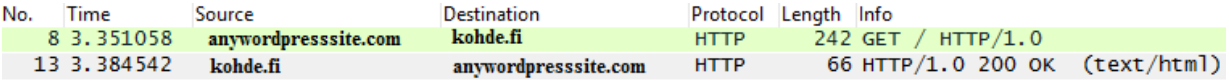
Kyselyn tuloksena on, että käyttäjän oma Wordpress-sivusto lähettää PING-vastauksen kohteen osoitteeseen. Esimerkiksi ”pingback”-ominaisuutta voidaan hyödyntää yksinkertaisella Linux-komennolla:

```
curl -D - "www.anywordpresssite.com/xmlrpc.php" -d '<?xml version=1.0?><methodCall><methodName>pingback.ping</methodName><params><param><value><string>http://kohde.fi</string></value></param><param><value><string>http://anywordpresssite.com/postnro</string></value></param></params></methodCall>'
```

Komennon avulla voidaan hyödyntää suojaamattomia Wordpress-sivustoja häiritsemään kohteeksi valittua sivustoa. Kuten kuvista 6 ja 7 huomataan, hyökkääjä saa haluamansa Wordpress-sivuston lähettämään pyynnön kohdeosoitteeseen. Paketit menevät järjestyksessä 6. -> 21. -> 8. -> 13. -> 27. -> 30. [42.; 43.]

No.	Time	Source	Destination	Protocol	Length	Info
6	2.210301	Hyökkääjä	anywordpresssite.com	HTTP	568	POST /xmlrpc.php HTTP/1.1
21	3.314012	anywordpresssite.com	kohde.fi	HTTP	242	GET / HTTP/1.0
27	3.348610	kohde.fi	anywordpresssite.com	HTTP	66	HTTP/1.0 200 OK (text/html)
30	3.350975	anywordpresssite.com	Hyökkääjä	HTTP/X	626	HTTP/1.1 200 OK

Kuva 6. Suojaamaton Wordpress (anywordpressite.com).



No.	Time	Source	Destination	Protocol	Length	Info
8	3.351058	anywordpresssite.com	kohde.fi	HTTP	242	GET / HTTP/1.0
13	3.384542	kohde.fi	anywordpresssite.com	HTTP	66	HTTP/1.0 200 OK (text/html)

Kuva 7. Kohdesivuston vastaanottamat paketit (kohde.fi).

Liitteessä 2 ovat kuvat kaikista paketeista, joista näkee, mitä tietoa kuvassa olevat paketit sisältävät.

XML-RPC on Wordpressissä automaattisesti päällä, joten XML-RPC:n häirintä tulisi estää asentamisen jälkeen. Häirinnän estämiseen on olemassa useita lisäosia, joita Wordpressiin voi ladata. Suosituin lisäosa tällä hetkellä on ”JetPack”. [41.; 44.]

XML-RPC-pyyntöjä varten loin säännön:

```
alert tcp any any -> $HOME_NET any (msg:"XML-RPC ddos"; flow:to_server,established; content:"xmlrpc.php"; http_uri; nocase; detection_filter:track by_src, count 100, seconds 120; sid:1000014; rev:001; classtype:attempted-dos;)
```

Oheisella säännöllä seurataan paketin http_url-kenttää. Jos paketista löytyy *xmlrpc.php* Snort huomaa, että kyseistä sivua kutsutaan. Koska kyseistä sivua avaavat myös asialliset sovellukset, niin rajoitin hälytyksen tapahtumaan, mikäli sama IP-osoite avaa *”xmlrpc.php*”-osoitteen 100 kertaa kahden minuutin sisään *”detection_filter:track by_src, count 100, seconds 120”*.

8.5.2 Säännölliset lausekkeet

Wordpress-versioista <= 4.3 löytyy haavoittuvuus, jonka avulla hyökkääjät voivat ajaa SQL-injektioita ja suorittaa XSS-hyökkäyksiä. Haavoittuvuus on korjattu jo 4.3.1 versiossa. Haavoittuvuuden voi todeta oheisella komennolla. Komento tulee syöttää uuteen postaukseen tai lisätä sivuille, koska kommenttikentässä komento ei toimi. [45.; 46.]

```
TEST!!![caption width="1" caption='<a href="' ">]</a><a href="http://onMouseOver='alert(1)'">Click me</a>
```

Komennossa olevalla lyhytkoodilla *“caption”* käyttäjä voi lisätä liitetylle kuvalle kuvatekstiä. Oheisessa komennossa kutsutaan caption kaksi kertaa, joten Snortille voi luoda säännön, joka etsii pakettia, jossa kyseiset sanat löytyvät. [45.; 46.]

Koska caption komentojen väliin voidaan syöttää eri merkkejä, olisi mahdotonta tallentaa jokaista eri variaatiota Snortin sääntöihin. Tätä varten säännöissä voidaan hyödyntää ”säännöllisiä lausekkeita”, Regular-Expressions (regexp). Näiden avulla voidaan esimerkiksi valita kaikki txt-tiedostot välittämättä minkä nimisiä tiedostot ovat.

```
.*\.txt$.
```

Snort tukee säännöllisiä lauseita joten, Säännön ”payload”–valinnoista löytyy valinta *“pcre”*, jonka avulla säännöllisiä lauseita voidaan käyttää. Pcre:tä käyttämällä loin säännön, jolla pystytään huomaamaan, mikäli kyseinen komento lisätään Wordpress-sivuille. [47.; 33.]

```
alert tcp any any -> $HOME_NET 80 (msg:" Regular-Expressions"; flow:to_server,established; pcre:"/%5b.*caption.*caption.*%5d/i"; sid:10000020; rev:001; classtype:policy-violation;)
```

Vaikka haavoittuvuus koskee vain vanhempia Wordpress-versioita, säännön ideana on hyödyntää Snortin säännöllisiä lauseita. Säännöllisten lauseiden avulla Snortille voidaan luoda todella tehokkaita sääntöjä, mutta nämä säännöt vievät paljon prosessoriaikaa.

8.5.3 WP-admin-kirjautuminen (Brute-force)

Oheisella säännöllä seurataan, jos palvelimen *”wp-login.php”* polku avataan ja palvelimelle lähetetään POST-komento.

```
alert tcp any any -> $HOME_NET 80 (msg:"Potential wp-admin Brute-force"; flow:to_server,established; content:"POST"; nocase; http_method; content:"wp-login.php"; http_uri; nocase; detection_filter:track by_src, count 30, seconds 60; sid:1000030; rev:001; classtype:attempted-admin;)
```

Sääntö toimii samalla tavalla kuin XML-RPC, mutta tässä säännössä hyödynnetään *“http_method-”* ja *“http_uri”*-valintoja. Tällä tavalla säännöistä voidaan tehdä yksilöllisempiä ja tarkempia. Eli molempien vaihtoehtojen tulee löytyä paketista, jotta sääntö aktivoituu.

9 Yhteenveto

Insinöörityön tavoitteena oli hyödyntää Snortia havaitsemaan tunkeutumisyrityksiä Sigmaticin asiakkaiden ylläpitämille Wordpress-sivuille.

Tunkeutumisen havaitsemisjärjestelmän tavoitteena on havaita haitallinen toiminta ja ilmoittaa siitä eteenpäin. Tunkeutumisen estojärjestelmä estää havaitsemansa haitallisen toiminnan automaattisesti. Nämä järjestelmät voivat toimia joko sormenjälkiin tai poikkeamiin perustuen.

Sormenjälkiin perustuvassa järjestelmässä verrataan haitallisen toiminnan jättämiä jälkiä ennalta määritettyihin jälkiin. Sormenjälkiin perustuvat järjestelmät luovat vähemmän vääriä hälytyksiä, koska hälytys generoidaan aina ennalta määritetyn säännön perusteella.

Poikkeamiin perustuvassa järjestelmässä määritellään verkolle tai palvelimelle tietty normaalitila, johon tulevaisuudessa toimintaa voidaan verrata. Esimerkiksi, jos normaalisti SSH-yhteyksiä palvelimelle muodostetaan kymmenen kappaletta päivässä, mutta jokin päivä yhteyksiä muodostetaankin tuhat kappaletta, niin tämä poikkeama huomataan ja siitä luodaan hälytys. Suurimpana järjestelmän heikkoutena pidetään kuitenkin sen luomia useita virheellisiä hälytyksiä.

Havaitsemisjärjestelmä voi olla joko verkkopohjainen tai isäntäpohjainen. Verkkopohjainen järjestelmä monitoroi verkkoliikennettä ja pyrkii sitä kautta havaitsemaan haitallisen liikenteen. Isäntäpohjainen järjestelmä valvoo isäntäkoneen tapahtumia ja pyrkii tapahtumien avulla huomaamaan haitallisen toiminnan isäntäkoneessa. Yhdessä nämä järjestelmät luovat tehokkaan havaitsemisjärjestelmän.

Insinöörityössä Snort toimii verkkopohjaisena tunkeutumisen havaitsemisjärjestelmänä, joka vertaa kaapattuja verkkopaketteja sääntöihin. Kaapatulle verkkopaketille suoritetaan Snortin toimesta viisi eri vaihetta, jotka ovat paketin hankinta (Libpcap), paketin dekoodaus, paketin esiprosessointi, paketin vertaaminen sääntöihin ja lopulta mahdollinen hälyttäminen.

Vaikka Snort toimii myös tehokkaasti yksinään, on Snortille kehitetty hyödyllisiä lisäosia. Näistä lisäosista hyödyllisimmät ovat hälytysten tallentamisen tietokantaan hoitava Barnyard, sääntöjen automaattisen päivittämisen hoitava PulledPork ja tallennettujen sääntöjen visuaalisen näyttämisen hoitava Snorby.

Snortin säännöt koostuvat kahdesta osasta: tunnisteesta ja vaihtoehdoista. Tunnisteella vaikutetaan säännön toimintaan ja vaihtoehdoissa määritetään haitallisen toiminnan jättämä sormenjälki.

Testiympäristön pystyttämisen aikana tutustuin pilvipalveluihin ja huomasin niiden tehokkuuden. Pilviympäristössä uuden virtuaalipalvelimen pystyy luomaan noin minuutissa, joten uusia palvelimia pystyy luomaan todella paljon lyhyessä ajassa. Myös virtuaalipalvelimen tehoja pystyy kasvattamaan tai laskemaan hetkellisesti, hyvinkin nopeasti.

Insinöörityön lopputuloksena oli, että Snort asennettiin ensin testiympäristöön, jossa pystytään luomaan tehokkaita hälytyksiä ja tutustumaan yleisimpiin hyökkäyksiin. Testiympäristössä kehitettyjä sääntöjä voidaan hyödyntää tuotantoympäristössä.

Vaikka insinöörityö päättyy, niin Snortin ylläpitäminen jatkuu. Tunkeutumisen havaitsemisjärjestelmät vaativat jatkuvaa ylläpitämistä, jotta niillä pystyttäisiin huomaamaan myös uusimmat hyökkäykset, jotka eroavat jo tunnetuista.

Lähteet

1 Toimintamme lyhyesti. Sigmatic Oy. Verkkodokumentti. <https://www.sigmatic.fi/yritys/>. Luettu 15.2.2016.

2 Eric Seagren, Weasley J. Noonan. 2007. Secure Your Network for Free : Using Nmap, Wireshark, Snort, Nessus, and MRTG. Canada: Syngress.

3 Jay Beale, James C. Foster, Jeffrey Posluns and Brian Caswell. 2003. Snort Intrusion Detection 2.0.

4 Defending against Code Red II using Symantec NetProwler and Intruder Alert. 2001. SANS Institute. Verkkodokumentti. <https://www.sans.org/reading-room/whitepapers/threats/defending-code-red-ii-symantec-netprowler-intruder-alert-ddos-461>. Luettu 10.2.2016.

5 Michael S Collins. 2014. Network Security Through Data Analysis: Building Situational Awareness. O'Reilly Media.

6 Roberto Di Pietro, Luigi V. Mancini. 2008. Intrusion Detection Systems. Springer US.

7 Chris Sanders, Jason Smith. 2013. Applied Network Security Monitoring. Elsevier / Syngress.

8 Brian Caswell, Jay Beale, Toby Kohlenberg, Andrew R. Baker. 2007. Snort IDS and IPS Toolkit. Syngress.

9 Strategies to Reduce False Positives and False Negatives in NIDS. 2001. Symantec. Verkkodokumentti. <http://www.symantec.com/connect/articles/strategies-reduce-false-positives-and-false-negatives-nids>. Luettu 15.2.2016.

10 Oinkcodes. 2016. Snort. Verkkodokumentti. <https://www.snort.org/oinkcodes>. Luettu 15.2.2016.

11 Snort FAQ/Wiki 2016. Snort. Verkkodokumentti. <https://www.snort.org/faq>. Luettu 15.2.2016.

12 Cisco to Buy Sourcefire. 2013. NYtimes. Verkkodokumentti. <http://dealbook.nytimes.com/2013/07/23/cisco-to-buy-sourcefire-a-cybersecurity-company-for-2-7-billion/?_r=1>. Luettu 15.2.2016.

13 Sourcefire. 2015. Wikipedia. Verkkodokumentti. <https://en.wikipedia.org/wiki/Sourcefire>. Luettu 15.2.2016.

14 finchy/pulledpork. 2015. GitHub. Verkkodokumentti. <https://github.com/finchy/pulledpork>. Luettu 15.2.2016.

15 Complete snort installation. 2014. Thomas Elsen Security Blog. Verkkodokumentti. <https://www.rivy.org/technologies/snort/>. Luettu 15.2.2016.

16 Installing Snort, Barnyard, Pulledpork and Snorby on CentOS6. 2015. Panagioto. Verkkodokumentti. <https://panagioto.com/installing-snort-barnyard-pulledpork-and-snorby-on-centos-6-6/>. Luettu 15.2.2016.

17 firnsy/barnyard2. 2015. GitHub. Verkkodokumentti. <https://github.com/firnsy/barnyard2>. Luettu 15.2.2016.

18 Snorby/snorby. 2016. GitHub. Verkkodokumentti. <https://github.com/Snorby/snorby>. Luettu 15.2.2016.

19 What is BASE. 2013. Base. Verkkodokumentti.<http://base.professionallyevil.com/>. Luettu 15.2.2016.

20 The Official Blog of the World Leading Open-Source IDS/IPS Snort. 2011. Snort. Verkkodokumentti. <http://blog.snort.org/2011/01/guis-for-snort.html>. Luettu 15.2.2016.

21 Analyzing Snort Data With the Basic Analysis and Security Engine (BASE). 2005. Oracle. Verkkodokumentti. <http://www.oracle.com/technetwork/systems/articles/snort-base-jsp-138895.html>. Luettu 15.2.2016.

22 Rules Headers. Snort. Verkkodokumentti. <http://manual.snort.org/node29.html>. Luettu 15.2.2016.

23 CMS Usage Statistics. 2016. Built With. Verkkodokumentti. <http://trends.builtwith.com/cms>. Luettu 15.2.2016.

24 PostNuke Input Validation Error in 'readpmsg.php' Permits SQL Injection and Cross-Site Scripting Attacks. 2005. Security tracker. Verkkodokumentti. <http://securitytracker.com/id/1014066>. Luettu 15.2.2016.

25 Cisco to Buy Sourcefire, a Cybersecurity Company, for $2.7 Billion. 23.7.2013. New York Times. Verkkodokumentti. <http://dealbook.nytimes.com/2013/07/23/cisco-to-buy-sourcefire-a-cybersecurity-company-for-2-7-billion/?_r=0>. Luettu 21.2.2016.

26 About us. UpCloud. Verkkodokumentti. <https://www.upcloud.com/company/>. Luettu 11.4.2016.

27 Thomas Tom. 2005. Verkkojen tietoturva: perusteet. IT Press.

28 Thomas Elsen Security. 13.4.2014. Blog. Rivy.org. Verkkodokumentti <https://www.rivy.org/technologies/snort/>. Luettu 11.4.2016.

29 What and Why of Snorby. 11.2.2015. GitHub. Verkkodokumentti. <https://github.com/Snorby/snorby/wiki/What-and-Why-of-Snorby>. Luettu 11.4.2016.

30 Rules Headers. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node29.html>. Luettu 11.4.2016.

31 The Basics. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node28.html>. Luettu 11.4.2016.

32 General Rule Options. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node31.html>. Luettu 11.4.2016.

33 Payload Detection Rule Options. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node32.html>. Luettu 11.4.2016.

34 Non-Payload Detection Rule Options. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node33.html>. Luettu 11.4.2016.

35 Post-Detection Rule Options. 2008. Snort. Verkkodokumentti. <http://manual.snort.org/node34.html>. Luettu 11.4.2016.

36 Murretut Wordpress ja Joomla -sivustot ohjaavat haittaohjelmia jakaville sivustoille. 16.12.2015. Viestintävirasto. Verkkodokumentti. <https://www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2015/12/ttn201512161547.html>. Luettu 11.4.2016.

37 CMS usage in Finland. 11.4.2016. BuiltWith. Verkkodokumentti. <http://trends.builtwith.com/cms/country/Finland>. Luettu 11.4.2016.

38 About the Let's Encrypt. 2016. GitHub. Verkkodokumentti. <https://github.com/letsencrypt/letsencrypt>. Luettu 11.4.2016.

39 Should You Disable XML-RPC on WordPress? 12.10.2015. Wordfence. Verkkodokumentti. <https://www.wordfence.com/blog/2015/10/should-you-disable-xml-rpc-on-wordpress/>. Luettu 11.4.2016.

40 Code Reference. WordPress. Verkkodokumentti. <https://developer.wordpress.org/reference/classes/wp_xmlrpc_server/mw_getcategories/>. Luettu 11.4.2016.

41 Brute Force Amplification Attacks Against WordPress XMLRPC. 8.10.2015. Sucuri. Verkkodokumentti. <https://blog.sucuri.net/2015/10/brute-force-amplification-attacks-against-wordpress-xmlrpc.html>. Luettu 11.4.2016.

42 More Than 162,000 WordPress Sites Used for Distributed Denial of Service Attack. 10.3.2014. Sucuri. Verkkodokumentti. <https://blog.sucuri.net/2014/03/more-than-162000-wordpress-sites-used-for-distributed-denial-of-service-attack.html>. Luettu 11.4.2016.

43 Wordpress "Pingback" DDoS Attacks. 2014. Sans ISC. Verkkodokumentti. <https://isc.sans.edu/forums/diary/Wordpress+Pingback+DDoS+Attacks/17801>. Luettu 11.4.2016.

44 Warning: XMLRPC WordPress Exploit DDOS. 2015. WordPress. Verkkodokumentti. <https://wordpress.org/support/topic/warning-xmlrpc-wordpress-exploit-ddos>. Luettu 11.4.2016.

45 WordPress <= 4.3 - Authenticated Shortcode Tags Cross-Site Scripting (XSS). 15.9.2015. WPvulndb. Verkkodokumentti. <https://wpvulndb.com/vulnerabilities/8186>. Luettu 11.4.2016.

46 Finding Vulnerabilities in Core WordPress: A Bug Hunter's Trilogy, Part III – Ultimatum. 15.9.2015. Check Point. Verkkodokumentti. <http://blog.checkpoint.com/2015/09/15/finding-vulnerabilities-in-core-wordpress-a-bug-hunters-trilogy-part-iii-ultimatum/>. Luettu 11.4.2016.

47 The Premier website about Regular Expressions. 29.3.2016. Regular Expressions. Verkkodokumentti. <http://www.regular-expressions.info/>. Luettu 11.4.2016.

48 Lockhart Andrew. 2006. Network Security Hacks, 2nd Edition Tips & Tools for Protecting Your Privacy. O'Reilly Media.

49 We Keep Your Network Safe. 2016. Talos. Verkkodokumentti. <http://www.talosintel.com/>. Luettu 11.4.2016.

50 Frequently Asked Questions About Intrusion Prevention Systems. Corero. Verkkodokumentti. <https://www.corero.com/products/faq-about-ips.html>. Luettu 14.4.2016.

51 Tripwire Open Source vs. OSSEC : Which Is Right For You? UpGuard. Verkkodokumentti. <https://www.upguard.com/articles/tripwire-open-source-vs.-ossec-which-is-right-for-you>. Luettu 14.4.2016.

52 About. 2015. OSSEC. Verkkodokumentti. <http://ossec.github.io/about.html>. Luettu 14.4.2016.

53 Cloud servers with MaxIOPS. 2015. UpCloud. Verkkodokumentti. <https://www.upcloud.com/features/>. Luettu 14.4.2016.

54 High performance cloud servers, from $10/mo. 2015. UpCloud. Verkkodokumentti. <https://www.upcloud.com/pricing/>. Luettu 14.4.2016.

55 About. 2015. Let's Encrypt. Verkkodokumentti. <https://letsencrypt.org/about/>. Luettu 14.4.2016.

56 Preprocessors. Snort. Verkkodokumentti. <http://manual.snort.org/node17.html>. Luettu 17.4.2016.

57 Preprocessors. 12.9.2003. InformIT. Verkkodokumentti. <http://www.informit.com/articles/article.aspx?p=101148&seqNum=2>. Luettu 17.4.2016.

58 Charlie Scott, Paul Wolfe, Bert Hayes. 2004. Snort For Dummies. Wiley Publishing, Inc.

59 Daemonlogger. 03.04.2013. Sourceforge. Verkkodokumentti. <https://sourceforge.net/projects/daemonlogger/>. Luettu 17.4.2016.

60 Seth Fogie (Author), Jeremiah Grossman (Author), Robert Hansen (Author), Anton Rager (Author), Petko D. Petkov. 2008. XSS Attacks: Cross-site Scripting Exploits and Defense. Syngress.

61 Dafydd Stuttard (Author), Marcus Pinto (Author). 2011. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws 2nd Edition. Wiley Publishing, Inc.

Snortin säännöt Classtype

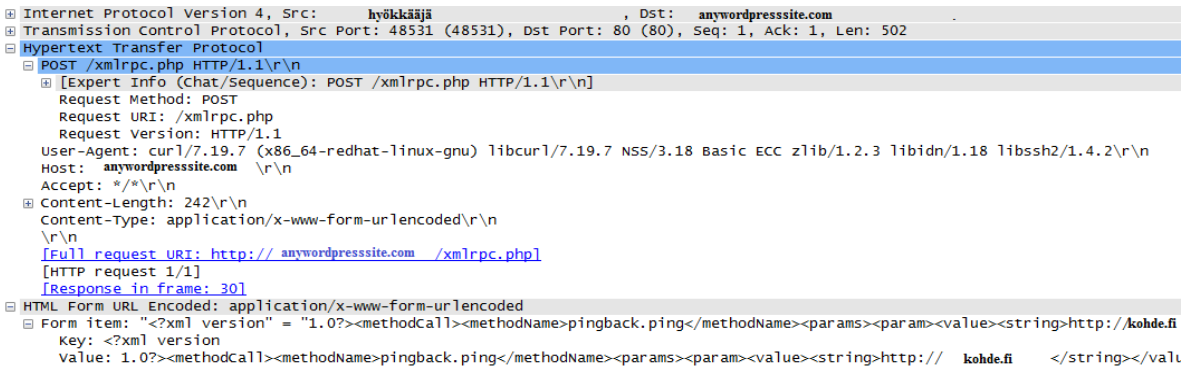
Table: Snort Default Classifications

Classtype	Description	Priority
attempted-admin	Attempted Administrator Privilege Gain	high
attempted-user	Attempted User Privilege Gain	high
inappropriate-content	Inappropriate Content was Detected	high
policy-violation	Potential Corporate Privacy Violation	high
shellcode-detect	Executable code was detected	high
successful-admin	Successful Administrator Privilege Gain	high
successful-user	Successful User Privilege Gain	high
trojan-activity	A Network Trojan was detected	high
unsuccessful-user	Unsuccessful User Privilege Gain	high
web-application-attack	Web Application Attack	high
attempted-dos	Attempted Denial of Service	medium
attempted-recon	Attempted Information Leak	medium
bad-unknown	Potentially Bad Traffic	medium
default-login-attempt	Attempt to login by a default username and password	medium
denial-of-service	Detection of a Denial of Service Attack	medium
misc-attack	Misc Attack	medium

non-standard-protocol	Detection of a non-standard protocol or event	medium
rpc-portmap-decode	Decode of an RPC Query	medium
successful-dos	Denial of Service	medium
successful-recon-largescale	Large Scale Information Leak	medium
successful-recon-limited	Information Leak	medium
suspicious-filename-detect	A suspicious filename was detected	medium
suspicious-login	An attempted login using a suspicious username was detected	medium
system-call-detect	A system call was detected	medium
unusual-client-port-connection	A client was using an unusual port	medium
web-application-activity	Access to a potentially vulnerable web application	medium
icmp-event	Generic ICMP event	low
misc-activity	Misc activity	low
network-scan	Detection of a Network Scan	low
not-suspicious	Not Suspicious Traffic	low
protocol-command-decode	Generic Protocol Command Decode	low
string-detect	A suspicious string was detected	low
unknown	Unknown Traffic	low
tcp-connection	A TCP connection was detected	very low

XML-RPC PingBack paketit

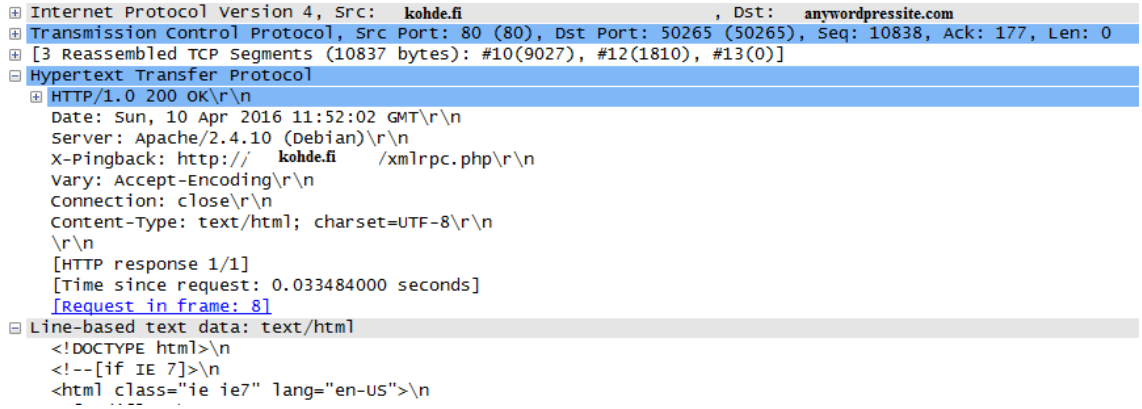
6. Paketti:



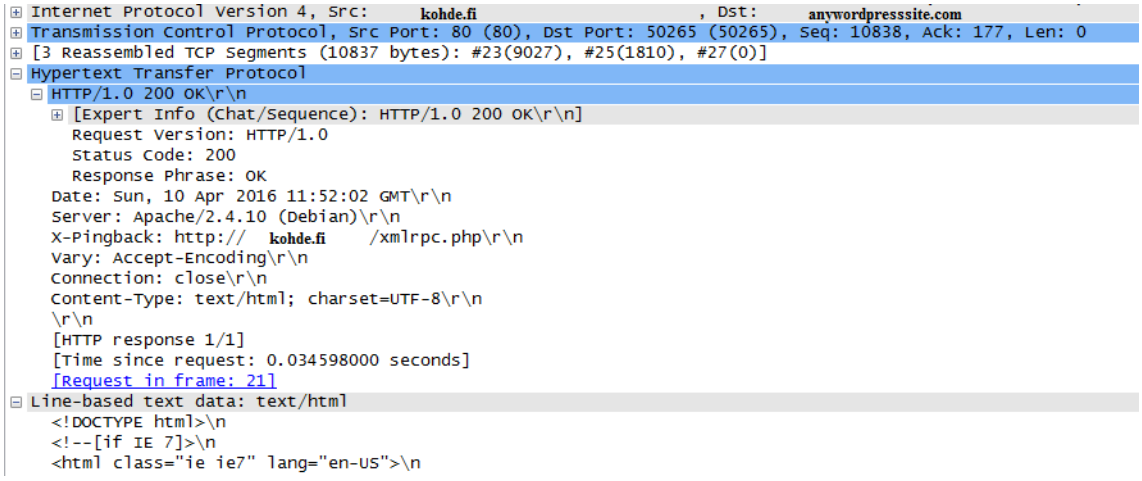
21. Paketti:

8. Paketti:

13. Paketti:



27. Paketti:



30. Paketti: