jamk.fi

Keskitetty haittaohjelmien tarkastusjärjestelmä

Nikke Kettunen

Opinnäytetyö
Toukokuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kettunen, Nikke	Opinnäytetyö, AMK	Toukokuu 2016
	Sivumäärä 85 + 8	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: X

Työn nimi
Keskitetty haittaohjelmien tarkastusjärjestelmä

Tutkinto-ohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Antti Häkkinen, Mika Rantonen

Toimeksiantaja(t)
JYVSECTEC
Marko Vatanen

Tiivistelmä

Opinnäytetyö toteutettiin JYVSECTEC-hankkeelle, joka on Jyväskylän Ammattikorkeakoulun tiloissa operoiva kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus. Hankkeen tavoitteena on luoda yksi Suomen johtavista kyberturvallisuuskeskuksista, sekä kehittää turvallisuusalan yhteistyöverkostoa jopa kansainvälisesti.

Työn tavoitteena oli toteuttaa avoimen lähdekoodin järjestelmä suljettuun ympäristöön, joka vastaisi toiminnaltaan Internetissä tarjottavaa VirusTotal-palvelua. VirusTotal on Googlen omistama palvelu, joka tarjoaa käyttäjille mahdollisuuden tarkastaa epäilyttäviä tiedostoja monella eri antivirusskannerilla yhtäaikaisesti. Web-sivun kautta käyttäjä lähettää palveluun tiedostonsa analysoitavaksi, jonka jälkeen järjestelmä tarkastaa tiedoston yli viidelläkymmenellä eri antivirusskannerilla ja palauttaa tulokset kootusti takaisin web-sivulle käyttäjän nähtäväksi.

Opinnäytetyössä tehtiin alkuun pikainen kartoitus ja vertailu avoimen lähdekoodin järjestelmistä, joilla oli mahdollista toteuttaa VirusTotalin kaltainen keskitetty haittaohjelmien tarkastusjärjestelmä. Vertailun jälkeen valittiin alusta nimeltään IRMA (Incident Response & Malware Analysis), jonka ominaisuudet eivät rajoittuneet pelkästään monen antivirusskannerin implementointiin. Työssä käytiin läpi IRMAn toteuttaminen kokonaisuudessaan, kuten myös skannereiden asentaminen järjestelmään. Järjestelmään tehtiin myös eri toiminnallisuuksia, sekä lopuksi koko järjestelmä testattiin tuloksineen.

Opinnäytetyön lopputuloksena saatiin toteutettua avoimen lähdekoodin keskitetty haittaohjelmien tarkastusjärjestelmä. Järjestelmään saatiin implementoitua 13 eri antiviruskanneria, sekä neljä muuta haittaohjelman analysointimenetelmää, kuten YARA ja StaticAnalyzer. Järjestelmää testattiin erilaisilla tiedostoilla, jotka sisältivät haitallista koodia. Toteutettua järjestelmää tullaan käyttämään JYVSECTECin kyberturvallisuustoiminnassa.

Avainsanat (asiasanat)

Antivirus, Analysointi, Haittaohjelma, Haittakoodi, Kyberturvallisuus, VirusTotal

jamk.fi

Description

Author(s)	Type of publication	Date
Kettunen, Nikke	Bachelor's thesis	April 2016
		Language of publication: Finnish
	Number of pages 85 + 8	Permission for web publication: X

Title of publication

Centralized malware analysis platform

Degree programme

Information Technology

Supervisor(s)

Häkkinen Antti, Rantonen Mika

Assigned by

JYVSECTEC

Marko Vatanen

Abstract

The bachelor's thesis was assigned by JYVSECTEC which is a cyber security project that operates in fields such as cyber research, development and training. JYVSECTEC project aims to create one of the leading cyber security centers in Finland and also to improve security co-operation network internationally.

The aim of the thesis was to implement an Open Source system which would correspond to the functionality of online service VirusTotal. VirusTotal is an online service owned by Google that allows users to check their files for malicious code. The file is scanned with over 50 different antivirus scanners, and the results of the analysis are returned back to the user.

At the beginning fast mapping and comparison were conducted between the possible Open Source systems that multiple antivirus scanning could be implemented with. After the comparison, a platform called IRMA (Incident Response & Malware Analysis) was selected to be implemented. IRMA is not only limited to wrapping antivirus scanners together but wrapping almost any kind of analyzers together, even one's own. The thesis covers the documentation of IRMA implementation followed by the scanner implementation. Different functionalities were also created for the platform and finally the whole platform was tested.

As a result of the thesis, a fully functional Open Source malware analysis platform was built. 13 antivirus scanners were implemented in the system with four other analysis tools such as YARA and StaticAnalyzer. The functionality of the platform was proven by testing it with a set of files that contained malicious code. The platform will be used in the JYCSECTEC cyber testing environment.

Keywords/tags (subjects)

Antivirus, Analysis, Malware, Malicious code, Cyber security, VirusTotal

Sisältö

Kuviot

Lyhenteet

AMQP	Advanced Message Queuing Protocol
API	Application Program Interface
AV	Antivirus
CA	Certificate Authority
DB	Database
DBIR	Data Breach Investigations Report
EICAR	European Institute for Computer Antivirus Research
EXE	Executable file
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
MD	Message-Digest
ORDBMS	Object-Relational Database Management System
ORM	Object-Relational Mapping
PC	Personal Computer
PDF	Portable Document Format
PE	Portable Executable
REST	Representational State Transfer
RGCE	Real Global Cyber Environment
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator

1 Johdanto

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii JYVSECTEC, joka on Jyväskylän ammattikorkeakoulun tiloissa operoiva kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus. Sen päämääränä on luoda Suomen yksi johtavimmista kyberturvallisuuskeskuksista ja kehittää turvallisuusalan yhteistyöverkostoa, jopa kansainvälisellä tasolla. JYVSECTEC sisältää harjoitustoimintaa, konsultointia, testausta, tutkimusta ja koulutusta. (JYVSECTEC – Tietoa meistä 2016.)

JYVSECTEC-hanke sai alkunsa syyskuussa 2011, kun sitä alettiin kehittämään ja rakentamaan projektimuotoisena eteenpäin. Se tarjoaa ja tuottaa kyberturvallisuusharjoituksia, kuten myös konsultointi-, tutkimus-, testaus- ja koulutuspalveluja. JYVSECTECin RGCE-ympäristö (Real Global Cyber Environment) tarjoaa nykyaikaisen järjestelmän, jonka avulla saadaan kehitettyä kyberturvallisuustietämystä, sekä harjoiteltua kyberuhkia vastaan. RGCE on realistinen, eristetty ja kontrolloitu ympäristö, jossa voidaan simuloida todellista verkkoliikennettä, kuten myös testata tietoturva uhkia, haavoittuvuuksia ja hyökkäysvektoreita. (JYVSECTEC – Tietoa meistä 2016.)

JYVSECTEC-hankeessa Keski-Suomen Liitto ja Euroopan aluekehitysrahasto ovat toimineet osarahoittajina, sekä yhteistyökumppaneina toimii usea yritys. Vuonna 2015 JYVSECTECille myönnettiin kahden miljoonan osarahoitus toiminnan laajentamiseen, kuten myös tilannekuvan tutkimiseen ja kehittämiseen kyberturvallisuuden parissa. (JYVSECTEC – Tietoa meistä 2016.)

1.2 Tavoitteet

Opinnäytetyön tavoitteena on toteuttaa avoimen lähdekoodin järjestelmä, jossa voidaan keskitetysti tarkastaa tiedostoja mahdollisimman monella eri antivirusskannerilla yhtäaikaisesti. Tavoite on toteuttaa toiminnaltaan samankaltainen järjestelmä, kuin Internetissä tarjottava palvelu VirusTotal. VirusTotal on nykyään Googlen omistama online-palvelu, jonka avulla käyttäjät voivat tarkistaa epäilyttäviä tiedostoja haittakoodin varalta. VirusTotalin idea on, että se implementoi useita eri antivirusskannereita, jotka keskitetysti tarkastavat käyttäjän lähettämän tiedoston järjestelmään.

Tuotoksessa käyttäjän pitää pystyä lähettämään tiedostoja web-sivulle, joita haluaa analysoida haitallisen koodin varalta. Järjestelmän tulee analysoida tiedostot käyttäen useaa eri antivirusskanneria, jonka jälkeen järjestelmä palauttaa skannerien tulokset kootusti takaisin web-sivulle käyttäjän nähtäväksi. Tarkoituksena on myös aluksi nopeasti vertailla mahdollisia eri avoimen lähdekoodin vaihtoehtoja, joilla kyseinen järjestelmä olisi mahdollista toteuttaa. Pikaisen vertailun kautta tarkoituksena on valita laadukkain ja monipuolisin alusta, jonka jälkeen lähteä toteuttamaan valittua alustaa.

1.3 Ajankohtaisuus

Kyberturvallisuus on kasvanut nyky-yhteiskunnan yhdeksi suurimmista huolista, eikä syyttä. IBM:n tekemien tutkimuksien mukaan noin 90 prosenttia olemassa olevasta datasta on kertynyt viimeisen kahden vuoden aikana. Datan suuren kasvun myötä myös kyberhyökkäykset ovat samalla lisääntyneet huomattavasti. Suurten hyökkäysten kohteeksi ovat joutuneet suuret yritykset kuten Sony, Adobe, Ashley Madison ja eBay.

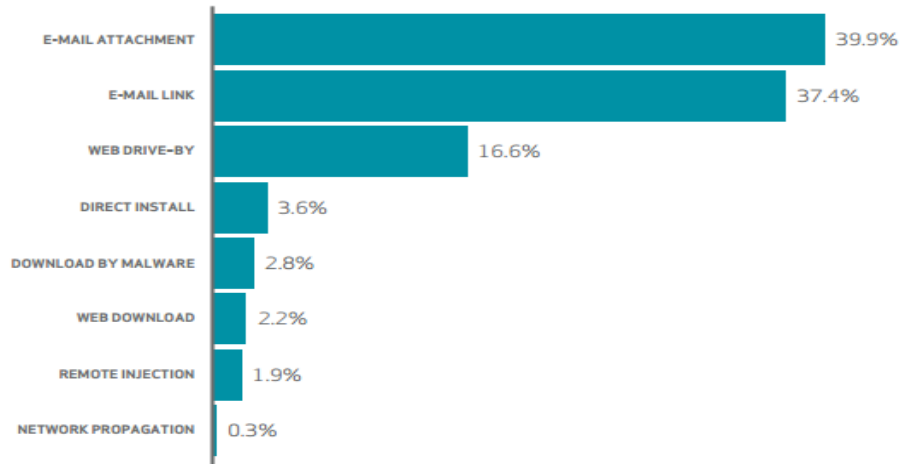
Kyberhyökkäysten ja haittaohjelmien nopean kehityksen takana on yleensä yksinkertainen motivaatio: raha. Nykyään haittaohjelmien kehitys ja niiden levitys ovat todella tuottavaa toimintaa rahallisesti. Haittaohjelmilla voidaan varastaa ihmisten eri verkkopalveluiden tunnuksia ja luottokorttitietoja, sekä päästä tilisiirtojen väliin. Valtiot ja yritykset käyttävät myös haittaohjelmia suorittaakseen vakoilua toisiin yrityksiin, valtioihin tai jopa yksittäisiin ihmisiin. (Koret & Bachaalany 2015, 5.)

Viime vuosien aikana haittaohjelmien torjunta on noussut suureksi haasteeksi kehittyneiden hyökkäysten myötä. Verkkorikollisuus on muuttumassa koko ajan ammattimaisemmaksi, ja näin ollen haittaohjelmia kehitetään samalla tavalla, kuin yritykset kehittävät tuotteitaan. Rahaa laitetaan valtavia summia haittaohjelmien kehitykseen, mikä ajaa antivirusyhtiöitä intensiivisempään vastatoimintaan. Nykyään matkapuhelimilla surffataan entistä enemmän verkossa, joten ilmiö on leviämässä myös mobiililaitteisiin. (Vakaasti kohti kyberturvallisuuden kärkimaata 2015.)

Ovelimmat kyberhyökkäykset ja tietomurrot havaitaan vasta paljon myöhemmin itse hyökkäyksestä, tai joskus niistä ei saada ikinä tietää. Sairasvakuutusyhtiö Premera Blue Cross joutui kyberhyökkäyksen kohteeksi 2014 vuoden toukokuussa, mutta hyökkäys tuli ilmi vasta vuonna tammikuussa 2015. Hyökkäys oli vaikuttanut arviolta 11 miljoonaan asiakkaaseen. Hyökkääjillä on ollut pääsy potilaiden henkilötunnuksiin, nimiin, osoitetietoihin, taloudellisiin tietoihin, pankkitilinumeroihin, sekä vakuutustietoihin. (This big U.S. health insurer just got hacked 2015.)

Kiristämistä (eng. blackmailing) tapahtuu nykypäivänä myös kybermaailmassa paljon. Kiristyksen kohteena ovat joko yhtiöt tai yksityiset henkilöt. Haittaohjelman pääsy yrityksen verkkoon ei siis vaaranna välttämättä vain yrityksen tietoja, vaan myös asiakkaiden. Ashley Madison –deittisivuston hakkerointi vuonna 2015 on yksi sen vuoden eniten kohua herättäneistä hyökkäyksistä. Hyökkäys vaikutti noin 37 miljoonaan ihmiseen ja se on yksi ensimmäisistä esimerkeistä, kun kyberrikolliset kiristävät yhtiötä ja sen asiakkaita samaan aikaan. Tämä hyökkäys on hyvä esimerkki sanonnalle ”Data mikä menee verkkoon, pysyy verkossa.” (A Year in Review: Cybersecurity Highlights of 2015)

Haittaohjelmista 70-90 % on tehty uniikisti juuri tietylle organisaatiolle. Talouden sektorille haittaohjelmatapauksia on keskimäärin 350 viikossa, kun taas koulutusala kohtasi keskimäärin 2332 tapausta viikossa. Verizonin tutkimuksen mukaan 67 prosenttia kyberhyökkäyksistä kohdistuu tietokantoihin. (Data Breach Investigation Report 2015, 22.) Kuten Verizonin vuoden 2015 raportista otetusta graafista (Kuvio 1) nähdään, lähes 80 prosenttia haittaohjelmista tarttuu sähköpostin liitetiedoston tai sähköpostissa olevan haitallisen linkin kautta. Yli 16 prosenttia haittaohjelmista tarttuu niin sanotun ”drive-by download” -hyökkäyksen kautta. Drive-by download -hyökkäys tarkoittaa, että web-sivusto asentaa haitallisen ohjelman tai takaoven sivuston selaajan koneelle ilman, että selaaja tekee mitään toimia sivulla. Pelkästään sivustolla oleminen riittää haitalliselle koodille tehdäkseen hyökkäyksen. Hyökkäys kuitenkin vaatii haavoittuvuuden laajennuksessa esimerkiksi Javassa tai Flashissa. (Newsletter Geekbuddy 2015.)



Kuvio 1. Haittaohjelmien tarttumislähteet (Data Breach Investigation Report 2015)

2 Haittaohjelmat

Vuonna 1971 ilmestyi ensimmäinen virus ”Creeper”, joka saastutti DEC PDP-10 –tietokoneita. Kuitenkin ensimmäinen PC-virus ”Brain” löydettiin vuonna 1986. Myöhemmin alkoi myös esiintyä verkon ylitse leviäviä matoja, kuten ”Love Letter” ja Nimda. 80-luvun virusten kohteina olivat aluksi Macintosh-koneet, minkä jälkeen kohteiksi joutuivat nopeasti yleistyvät PC-koneet ja näiden dos-käyttöjärjestelmät. (Järvinen 2012, 178.)

Nykypäivän virukset ja haittaohjelmat eroavat vanhoista huomattavasti. Ennen viruksia kirjoiteltiin lähinnä harrastukseksi teinipoikien toimesta. Käyttäjä myös tiesi ja näki, kun koneeseen oli päässyt virus. Virus teki yleensä jotain visuaalista käyttäjän koneella, kuten esimerkiksi ”Walker”-viruksessa mies käveli ruudun poikki tai Joshi-viruksessa piti toivottaa hyvää syntymäpäivää Joshille tiettynä vuoden päivämääränä, jotta pystyi jatkamaan käyttöjärjestelmän käyttämistä. Pahempiakin viruksia oli, kuten Disk Killer, joka pyyhki dataa kiintolevyiltä. (Järvinen 2012, 178.)

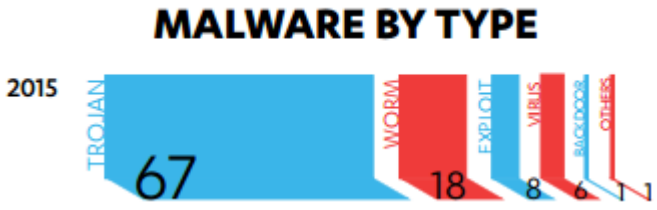
Nykyään virukset ja haittaohjelmat ovat kuitenkin paljon hienostuneempia ja motivaationa toimii pääasiassa raha. Haittaohjelmat kehittyvät nykyään nopeammin ja muuttuvat jatkuvasti. Virusten ja haittaohjelmien tekeminen on nykyään rahoitettu organisaatioiden ja jopa hallitusten toimesta. Tätä ajaa suuri kannustin varastaa käyttäjän finanssitietoja tai yritysten kauppasalaisuuksia. Lisäksi haittaohjelmat mahdollistavat hallitusten suorittaa vakoilua tai potentiaalista kybersotaa kohteisiinsa. (Limnéll, Majewski & Salminen 2014, 111-124.) Joidenkin ohjelmien tarkoituksena on siis sabotoida valtion infrastruktuuria, kuten esimerkiksi Stuxnetin. Stuxnet-mato onnistui sabotoimaan Iranin atomivoimalaa käyttäen viittä eri nollapäivä-haavoittuvuutta. (Koret & Bachaalany 2015, 28-29.) Kuviossa 2 on avattu kyberuhkan merkitystä kuvion avulla ja siinä on esitetty kybermaailman toimijoita motivaatioineen ja kohteineen.

	Motivaatio	Toimijat	Kohde
Kybersota	Poliittinen/ sotilaallinen hallinta	Valtiot	Kriittinen infra ja muut strategiset kohteet
Kyberterrorismi	Poliittinen muutos, pelko	Terroristit	Infra, voimavarat ja julkiset kohteet
Kybervakoilu	Tiedon varastaminen	Valtiot ja yritykset	Hallitukset, yritykset, yksilöt
Kyberrikollisuus	Taloudellinen hyötyminen	Rikolliset	Yritykset ja yksilöt
Haktivismi	Polittinen muutos, egoismi	Aktivistit, haktivistit ja yksilöt	Hallitukset, yritykset, yksilöt

Kuvio 2. Kyberuhkien komponentit (Limnéll, Majewski & Salminen 2014, 113.)

2.1 Malware

Malware tulee sanoista malicious software. Suomeksi se tarkoittaa haittaohjelmaa, joka on yleisnimitys tiedostoille, jotka sisältävät haitallista koodia. Haittaohjelmat on suunniteltu tietokonejärjestelmiin tai muihin laitteisiin, ja ne tekevät haitallista toimintaa käyttäjäänsä vastaan. Malware kuvaa yleisesti kaikkia viruksia, matoja, spywareja ja periaatteessa kaikkea, mikä on suunniteltu tekemään harmia käyttäjälle. Kuviossa 3 nähdään F-Securelle raportoitujen haittaohjelmien tyypit prosenttilukuina vuonna 2015. Troijalaiset ovat kärjessä suurella 67 prosentin osuudellaan, jonka jälkeen tulevat madot, exploitit, virukset, takaovet ja muut haittaohjelmat. (What Is the Difference: Viruses, Worms, Trojans, and Bots? n.d.)



Kuvio 3. Vuoden 2015 haittaohjelmatyypit (F-Secure Threat report 2015.)

2.2 Troijalainen

Nimi ”Troijalainen” tulee Kreikan legendan Troijan suuresta puisesta hevosesta, jota kreikkalaiset tarjosivat troijalaisille rauhan merkkinä. Kreikkalaiset sotilaat piiloutuivat hevosen sisään ja sen avulla pääsivät vihollisen muurien taakse. Yön tullen sotilaat avasivat vihollisen portit omille joukoilleen ja sen avulla saivat vallattua kaupungin. Troijalaisessa haittaohjelmassa on sama idea. Se esittää olevansa jokin hyödyllinen ohjelma, mutta oikeasti suorittaa toista toimintoa taustalla ilman käyttäjän tietoa asiasta. Troijalainen näyttää tarjoavan käytännöllistä palvelua, kuten näytönsäästäjää, apuohjelmaa tai ohjelmapäivitystä, jotta käyttäjä asentaisi troijalaisen. Kun troijalainen on asennettu, se tekee luvatonta toimintaa vitsimäisestä työpöydän kuvakkeiden siirtelemisestä aina vakavampiin käyttäjää estäviin toimintoihin. Troijalainen voi myös ladata lisää haittaohjelmia koneelle, tuhota uhrin tiedostot, tehdä koneesta osan bottiverkkoa tai lukita käyttäjän ulos käyttöjärjestelmästä. Troijalaiset voidaan jakaa alaluokkiin nimillä Downloader, Dropper, Proxy, PWS ja Spy. (What Is the Difference: Viruses, Worms, Trojans, and Bots? n.d.)

2.3 Mato

Madot ovat ohjelmia, jotka replikoivat itseään järjestelmästä järjestelmään verkon ylitse ilman tarvetta isäntätiedostolle. Tämä on toisin kuin viruksissa, jotka tarvitsevat isäntätiedoston levitäkseen. Madot yleensä sijaitsevat muiden tiedostojen sisällä, kuten Word- ja Excel–dokumenteissa, mutta madolla ja viruksella on ero, miten ne käyttävät isäntätiedostoa. Yleensä mato julkaisee dokumentin, jossa on jo valmiiksi madon makro sisällä. Koko dokumentti kulkee koneelta koneelle, joten koko dokumenttia voidaan pitää matona. (Threat description 2015.)

Teoreettisesti matoja voidaan kuljettaa minkä tahansa verkon ylitse. Internet, sähköpostijärjestelmät ja pikaviestintä ovat yleisimpiä kanavia matojen levittämiseen. Mato voi myös levitä irrotettavan median kautta, joten laitteet, joissa ei ole verkkoa ovat myös tartutettavissa. Uhreja uskotellaan muun muassa videoilla ja kuvilla, joista uhrit painavat ja painalluksen jälkeen mato asentuu laitteelle huomaamatta. Kun mato on asentunut, se yleensä yrittää levitä verkon ylitse. Mato tyypillisesti varaa suuren kaistan tiedonsiirrossa, kun se leviää. Äärimmäisillään verkon käyttö on mahdotonta, kunnes mato on suorittanut leviämisrutiininsa. (Threat description 2015.)

2.4 Virus

Virus on haittaohjelma, joka on tarkoitettu vahingoittamaan tietokonejärjestelmää. Virus on itsestään replikoiva ja itsestään asentuva, joka voi levitä tiedostosta toiseen. Virus tarvitsee aina isäntätiedoston. Useimmat virukset liittävät itsensä suoritettaviin tiedostoihin, mutta jotkut voivat kohdistua pääkäynnistystietueelle, skriptien automaattiseen ajamiseen, makroihin tai joissain tapauksissa sattumanvaraisiin tiedostoihin. Virus voi olla polymorfinen eli se replikoi itseään, mutta replikaatio ei ole sama kuin alkuperäinen koodi. Polymorfisella viruksella on sama hyötykuorma, mutta tiedostokoko ja muoto muuttuvat. Tätä tapaa käyttävä virus tekee sen antivirus ohjelmalle vaikeamman havaita. (Arntz 2015.)

2.5 Takaovi

Takaovi eli backboor on etähallinnan apukeino, jonka avulla hyökkääjä voi tehdä uhrin koneella haitallista toimintaa. Takaoven avulla on usein mahdollisuus saada kontrolli laitteesta, koska se käyttää hyväkseen dokumentoimattomia prosesseja laitteen koodissa. Tyypillinen takaovi koostuu clientistä ja palvelimesta. Hyökkääjä käyttää clienttiä kommunikoidakseen uhrin laitteelle asennettujen komponenttien avulla. Nämä komponentit voivat tulla uhrin laitteelle muun muassa madon osana, Troijalaisen hyötykuormana tai sähköpostiliitteenä. Takaovea käytetään yleensä, kun halutaan varastaa käyttäjän tietoja tai levittääkseen haittaohjelmaa eteenpäin. (Threat description 2015.)

2.6 Ramsomware

Ransomware on haittaohjelma, joka kryptaa dataa uhrin koneelta ja vaatii uhrin maksamaan lunnaita, jotta tiedostot saataisiin palautettua eli periaatteessa ostetaan salainen dekryptausavain. Ransomwareen liittyy yleensä aika, jonka umpeuduttua salattu data tuhotaan, minkä jälkeen dataa on enää mahdoton saada takaisin, koska datan päälle kirjoitetaan. Kun ransomware on päässyt koneelle, ohjelma aktivoidaan ja se lukitsee käyttöjärjestelmän ja laukaisee lunnasvaatimuksen. Tämän hetken tunnetuimpia ransomwareja ovat Cryptolocker, Cryptowall ja kaikista viimeisin TeslaCrypt. (Gheorghe 2015)

2.7 Keylogger

Keylogger seuraa ja pitää lokia kaikesta, mitä uhri näppäilee näppäimistöllään. Tämä seuranta tapahtuu yleensä uhrilta näkymättömissä, ja sen avulla uhrilta saadaan tilitietoja, käyttäjänimiä ja salasanoja, sekä muuta salaista dataa. Keyloggerit voivat tulla ohjelmamuodossa, kuten esittämällä näppäimistön ajuria, käyttäen APIa, nauhoittamalla selainpohjaisten lomakkeiden täyttöä tai kaappaamalla HTTP POST-tapahtumia. Keyloggereita voi myös olla firmware-pohjaisia, sekä hardware-pohjaisia. (Security news 2013.)

2.8 Rootkit

Rootkitit ovat ohjelmia, jotka on suunniteltu piilottamaan prosesseja, tiedostoja tai rekisterimerkintöjä. Tämän tyyppisiä ohjelmia hakkerit käyttävät piilottaakseen jälkensä tai laittaakseen uhkia salaa kaapatuille tietokoneille. Rootkitit naamioivat virukset ja muut haittaohjelmat tarpeellisiksi tiedostoiksi, ja tämän takia antivirusohjelma ei välttämättä huomaa haittaohjelmia. Rootkitit itsessään eivät ole haitallisia, vaan yksinkertaisesti niitä käytetään piilottamaan haittaohjelmia. (Security news 2013)

Termi ”rootkit” on Unix -termi pääkäyttäjän admin-tunnukseen ”root”, ja ”kit” viittaa ohjelmaan, joka implementoi työkalun. Asentaakseen rootkitin, hyökkääjän täytyy ensiksi saada root-oikeudet, joko käyttämällä haavoittuvuutta, kräkkäämällä salasanan tai käyttämällä sosiaalista manipulointia (social engineering). Koska rootkit aktivoidaan jo ennen käyttöjärjestelmän käynnistämistä, niitä on todella vaikea havaita. Sen takia se tarjoaa tehokkaan tavan hyökkääjälle päästä uhrin koneelle tämän tietämättä. (Security news 2013)

2.9 Adware, Spyware ja browser hijacker

Adware-ohjelmat ovat yleensä mainostajien tukemia ohjelmia, jotka suorittavat, näyttävät tai lataavat mainossisältöä koneelle ilman käyttäjän lupaa. Esimerkiksi pop-up ikkunat ovat esimerkki adwaresta. Adware on yleensä sisällytetty hyväksynnällä käyttäjän lataamaan ohjelmaan. Jotkut yhtiöt jopa laittavat adwareja heidän tuotteisiinsa alentamalla kustannuksia tai tehdäkseen heidän tuotteestaan ilmaisen. Adware voi kerätä tietoja käyttäjän selainkäyttäytymisestä, minkä jälkeen se lähettää tiedot yhtiölle, joka taas lähettää suunnattuja mainoksia käyttäjälle. (Security news 2013.)

Adware voi myös sisältää tai se voidaan luokitella Spywareksi. Spyware on haittaohjelma, joka on yksityisyyteen tunkeutuva ja nimensä mukaisesti ”vakoilee” käyttäjää eli voi varastaa käyttäjän tietoja tai korruptoida tiedostoja. Spywareen voi sisältyä esimerkiksi keylogger tai se voi hidastaa tietokoneen toimintaa asentamalla lisäohjelmia, uudelleenohjata selainhakuja tai muuttaa tietokoneen asetuksia. Tämä kaikki tapahtuu yleensä ilman käyttäjän tietämystä asiasta. (Security news 2013.)

Browser Hijacker eli selaimen kaappaus tarkoittaa, että selaimen asetuksia muutetaan ilman käyttäjän tietoa. Kotisivu saattaa muuttua ja paljon pop-up ikkunoita alkaa esiintyä. Tämä tapahtuu hijackwaren avulla, joka on usein asennettu ilmaisohjelman kautta, mutta se voi myös tulla muutakin kautta, kuten sähköpostiliitteenä. Selainkaappauksen avulla rikollinen voi tehdä paljon vahinkoa. Ohjelma voi vaihtaa selaimen kotisivuksi haitallisen sivun, kaataa tietokoneen tai asentaa vakoiluohjelmia. Esimerkiksi tunnettu hijacker CoolWebSearch asettaa selaimen kotisivuksi heidän hakusivunsa ja sen kaikki hakutulokset johtavat sivuille, joista klikkaaminen tuo rahaa yritykselle. Rikolliset voivat myös myydä tietoja uhrin selailukäyttäytymisestä kolmansille osapuolille. (Security news 2013.)

3 Haittaohjelmien torjunta

3.1 Antivirusohjelmat

Antivirusohjelma on tietoturvaohjelma, jonka tarkoituksena on antaa parempi suoja, kuin mitä alla oleva käyttöjärjestelmä tarjoaa (kuten Windows tai Mac OS X). Antivirusohjelma on suunniteltu estääkseen tietokoneen saastumista havaitsemalla haitallisia ohjelmia. Useimmissa tapauksissa sitä käytetään ehkäisevänä ratkaisuna. Kun ehkäisevä tapa pettää, antivirusohjelmaa käytetään puhdistaakseen saastuneet tiedostot tai poistamaan koko saastunut tiedosto käyttöjärjestelmästä. (Koret & Bachaalany 2015, 3-5.)

Ensimmäisiä antivirus tuotteita kutsuttiin yksinkertaisesti skannereiksi, koska ne olivat komentoriviskannereita, jotka yrittivät tunnistaa haitallisia malleja suoritettavista tiedostoista. Nykyään antivirus tuotteet sisältävät graafisen käyttöliittymän ja voivat myös sisältää eri lisäosia suojellakseen tietokonetta, kuten palomuurin tai selaimen lisäosia. Ennen antivirus yhtiöt saivat vain kourallisen haittaohjelmia tietoonsa viikossa, mutta nykyään puhutaan tuhansista päivittäin. Tämä haittaohjelmien kasvu on ajanut yhtiöt keskittymään enemmän automatisoituun tunnistamiseen ja kehittämään heuristista tunnistusmenetelmää ei vielä tiedossa olevien haittaohjelmien varalle. (Koret & Bachaalany 2015, 3-5.)

90-luvulla virukset olivat todella yleisiä, joten tästä syystä torjuntaohjelmia on markkinoitu ”Antivirus” termeillä. Kuitenkin nykyään virukset ovat vähemmistöä verrattuna muihin haittaohjelmiin. Vaikka virukset ovat jääneet vähemmistöön, silti tällä hetkellä kehittyy ja leviää viruksia kuten ”Sality”, ”Virut” ja ”Conficker”. Vaikka yhtiöt markkinoivat ohjelmiaan antivirusohjelmina, ei se tarkoita, etteivätkö ne pystyisi löytämään muitakin haittaohjelmia kuin viruksia. (Machine Learning for Anti-virus Software 2013.)

Alun perin antivirusohjelmat rakennettiin manuaalisesti. Asiantuntija tarkisti jokaisen haitallisen ohjelman ja tiedoston sekä loi niiden allekirjoituksen. Sitä allekirjoitusta vasten tiedostoja tai sen osia verrattiin. Haittaohjelmien kasvun myötä asiantuntijat eivät riitä generoimaan jokaiselle haittaohjelmalle omaa allekirjoitusta. Tämä antaa aiheen heuristisille tai geneerisille allekirjoituksille, jotka voivat käsitellä monta eri variaatiota samasta tiedostosta. Kuitenkin uudentyyppisiä haittaohjelmia luodaan lisää joka päivä. (Machine Learning for Anti-virus Software 2013.)

3.2 Antivirusohjelmien toiminta

Antivirusohjelma käyttää tyypillisesti montaa eri menetelmää havaitakseen ja poistaakseen viruksia, matoja ja muita haittaohjelmia. Eri antivirusohjelmilla on eri menetelmiä ja algoritmeja, joilla ne tunnistavat eri haittaohjelmia. Kuitenkin kaksi eniten käytettyä tunnistusmenetelmää ovat allekirjoitukseen ja heuristiikkaan perustuvat menetelmät. (Anand 2012.)

Haittaohjelmille nimiä antaessa, ei ole mitään standardia jota antivirusyhtiöiden pitäisi noudattaa. Sama haittaohjelma voi olla nimettynä toisessa yhtiössä aivan eri nimellä, kuin toisessa yhtiössä. Tämä voi luoda sekaannusta, kun aletaan vertailla eri antivirusskannereiden tuloksia. (Landesman n.d.)

3.2.1 Haittaohjelmien nimien ymmärtäminen

Antivirusyhtiöt antavat yleisesti haittaohjelmille nimiä, jotka koostuvat etuliitteestä (prefix), nimestä ja jälkiliitteestä (suffix). Kaikki yhtiöt eivät käytä kyseistä mallia ja ne jotka käyttävät, voivat myös joskus käyttää muita malleja. Yhtiöt voivat siis käyttää toinen toistaan erikoisempia haittaohjelmien kutsumanimiä, mutta on hyvä ymmärtää perusasiat, kuinka perus malli rakentuu. (Landesman n.d.)

Etuliitteestä tunnistaa haittaohjelman tyypin. W32 tai Win32 tarkoittaa, että se on 32-bittisen Windows-käyttöjärjestelmän saastuttaja, joka vaikuttaa myös Windows 95, 98,2000, 2003, XP, Me ja NT 4.0 käyttöjärjestelmiin. Windows 95/98 käyttöjärjestelmille oleva saastuttaja kulkee etuliitteellä W95. Jotkut yhtiöt käyttävät etuliitteitä jotka kuvaavat itse uhan tyyppiä, eikä niinkään käyttöjärjestelmää, jonka se saastuttaa. Esimerkiksi TROJ etuliite kuvastaa Troijalaista, I-Worm verkko/sähköpostimatoa ja OM viestii Microsoft Officen makroviruksesta. (Landesman n.d.)

Etuliitteen jälkeen tulee itse haittaohjelman nimi. Esimerkiksi W32/Bagle haittaohjelmalla on etuliite W32, jota seuraa nimi Bagle. Bagle taas on sähköpostimato, tarkemmin sanottuna massapostitusmato. (Landesman n.d.)

Moni haittaohjelma kuuluu samaan perheeseen, mutta eri versiot ovat hieman erilaisia. Erottaakseen samaan perheeseen kuuluvien haittaohjelmien eri variantit, yhtiöt lisäävät yleensä aakkosellisen jälkiliitteen nimen perään. Esimerkiksi W32/Bagle nimestä tuli W32/Bagle.A, kun toinen variantti haittaohjelmasta löydettiin. (Landesman n.d.)

3.2.2 Allekirjoitukseen perustuva tunnistus

Allekirjoitukseen perustuva on kaikista yleisin käytetty metodi, johon liittyy tunnettujen rakenteiden etsiminen tarkastettavasta tiedostosta. Jokaisella antivirusohjelmalla on ”sanakirja” haitallisten koodien näytteistä tietokannassa, joita kutsutaan tunnisteiksi. Aina kun tiedosto on tarkastettu, antivirusohjelma vertaa sanakirjasta tunnisteita tarkastettavaan tiedostoon. Jos tiedoston osa koodia on sama kuin tunnistetietokannassa, tiedosto merkitään ja toimitaan tarpeen mukaisesti. Antivirusohjelma voi joko korjata tiedoston, laittaa karanteeniin tai poistaa sen perustuen haitallisen koodin riskiin. Koska uusia haittaohjelmia tehdään lisää joka päivä, tämä havaitsemismetodi ei voi puolustaa uusia haittaohjelmia vastaan ennen kuin niiden tunniste on kerätty ja lisätty tietokantaan antivirusyhtiön toimesta. Monet yhtiöt kannustavat lähettämään heille uusia haittakoodia sisältäviä ohjelmia, jotta niiden tunnisteet voidaan lisätä heidän tietokantaansa. (Anand 2012.)

Allekirjoitukset ovat tyypillisesti tiivisteitä (hash) tai bittijonoja (bit-stream), joita käytetään haittakoodin etsimisessä tiedostoista tai puskureista. Allekirjoitus sisältää yleensä tarpeeksi tietoa tunnistaakseen tiedoston haitalliseksi, kun sitä verrataan tunnettuun malliin (pattern). Kun tiivisteitä käytetään allekirjoituksissa, ne yleensä generoidaan algoritmeilla kuten CRC tai MD5. Nämä algoritmit ovat tyypillisesti nopeita ja niitä voidaan laskea monta kertaa sekunnissa ilman huomattavaa suorituskyvyn heikkenemistä. Tiivisteet ovat tyypillisimpiä ja suosituimpia tapoja haitallisten ohjelmien osien tunnistamiseen, koska ne ovat nopeita ja helppo käyttöönottaa. (Koret & Bachaalany 2015, 77-78.)

Yleensä algoritmeissa nopeus ja virheherkkyys kulkevat käsikädessä. Nopeasti tunnistavilla algoritmeilla on korkeampi väärien tunnistuksien (false-positive) suhde, kun taas enemmän aikaa vievät algoritmit tarjoavat laadukkaamman tuloksen. (Koret & Bachaalany 2015, 77-78.) Allekirjoitukseen perustuva havaitsemistapa on tehokas tunnettujen haittaohjelmien havaitsemiseen, mutta vaatii tiheän allekirjoitusten päivittämisen tietokantaan. Näin ollen käyttäjien pitää päivittää heidän antivirusohjelmiaan säännöllisesti puolustaakseen uusia, joka päivä syntyviä haittaohjelmia vastaan. (Anand 2012.)

3.2.3 Heuristiikkaan perustuva tunnistus

Virustunniste on yksinkertaisesti binäärimalli, jonka avulla voidaan tunnistaa tiedettyjä uhkia. Käyttäen virustunnistetietokantaansa antivirusskannerit tarkastavat tiedoston vertaamalla tunnisteita tarkastettavaan tiedostoon. Tämä lähestymistapa ei pysty tunnistamaan uusia haittaohjelmia, koska niiden tunnisteita ei vielä ole tietokannassa. Tätä varten on kehitetty heuristinen analysointi. Esimerkiksi jos ohjelma yrittää kirjoittaa dataa suoritettavaan tiedostoon, se merkitään ja käyttäjää varoitetaan tästä. Tämä toimintatapa tarjoaa uuden turvallisuustason tunnistamattomia uhkia vastaan. (Anand 2012.)

Heuristinen tunnistustapa perustuu tiettyyn tietämykseen (heuristiikka) tietyistä toiminnoista (piirre), jotka voivat olla tyypillisiä haittaohjelmalle itselleen tai päinvastoin – jotka ovat todella harvinaisia haittaohjelmissa. Kullakin piirteellä on painokerroin, joka määrää sen vakavuuden ja luotettavuuden. Painokerroin voi olla positiivinen, jos piirre ilmaisee haitallisesta koodista - tai se voi olla negatiivinen, jos piirre ei ole luonteenomaista uhalle. Heuristinen moottori laskee yhteisen painoarvon tarkastettavalle tiedostolle ja sen perusteella ilmoittaa onko kyseessä saastunut vai puhdas tiedosto. (Fighting computer threats 2015.)

Heuristisia moottoreita on kahden tyyppisiä: staattisia ja dynaamisia. Staattiseen dataan perustuvan moottorin ei tarvitse suorittaa tai emuloida itse näytettä tutkiakseen onko ohjelma haitallinen. Dynaamiset moottorit taas monitoroivat ohjelman ajamista käyttöjärjestelmässä, kuten antivirusohjelman hiekkalaatikossa (sandbox). Heuristiset moottorit ovat yleensä alttiita väärille tunnistuksille, koska ne ovat todiste-pohjaisia eli etsivät eri toiminnollisuuksia tiedostosta hyvinkin laajalti.
(Koret & Bachaalany 2015, 65-66.)

4 Järjestelmät

4.1 Avoin lähdekoodi

Useimmat ohjelmat, joita ostetaan tai ladataan, tulevat valmiiksi kootussa ready-to-run versiossa. On todella vaikeaa muokata koottua versiota ja miltei mahdotonta nähdä tarkalleen, miten kehittäjä on luonut eri ohjelman osat. Useimmissa kaupallisissa ohjelmissa valmistajat näkevät tämän etuna, jotta muut yhtiöt eivät kopioisi heidän lähdekoodiaan ja täten säilyttävät kilpailukykynsä. Tämä myös tuo kontrollin laadun- ja toimintojenhallintaan. Koottua lähdekoodia kutsutaan yleensä suljetuksi lähdekoodiksi (Closed Source). Closed Sourcen esimerkkejä ovat Microsoft Word ja Adobe Photoshop. Ennen näiden ohjelmien käyttöä pitää hyväksyä ehdot, että käyttäjä ei tee mitään ohjelmalle, jota ohjelman auktoriteetit eivät ole hyväksyneet. (Opensource 2016.)

Avoimen lähdekoodi (Open Source) on suljetun Closed Sourcen vastakohta. Lähdekoodi on kootun version mukana, ja koodin muokkaaminen on jopa rohkaistua. Ohjelmien kehittäjät, jotka tukevat Open Source ajatusta uskovat, että kun lähdekoodin muokkaaminen on sallittua, ohjelmasta tulee paljon hyödyllisempi ja virheettömämpi pitkällä aikavälillä. (Opensource 2016.)

Jotta ohjelmaa voidaan pitää Open Sourcena, seuraavat kriteerit pitää täyttyä:

- Ohjelman pitää olla vapaasti jaeltavissa
- Lähdekoodin pitää olla ohjelman mukana
- Kuka tahansa voi muokata lähdekoodia
- Muokattuja versioita voi uudelleen jakaa, ja lisenssi ei saa vaatia muiden ohjelmien poisjättämistä tai se ei saa haitata muiden ohjelmien toimintoja (Opensource 2016.)

Open Sourcesta eivät hyödy vain itse ohjelmoijat, vaan myös kaikki muut. Internet on rakennettu myös monen Open Source teknologian päälle, kuten Linux-käyttöjärjestelmät ja Apache-palvelimet. Monet ihmiset pitävät Open Sourcea parempana, koska heillä on kontrolli käyttämistään ohjelmista. He voivat tutkia koodia, varmistaa ettei se tee mitään, mitä he eivät halua, ja voivat muuttaa sitä haluamakseen. Jotkut pitävät Open Sourcea sen takia parempana, että se auttaa heitä kehittymään ohjelmoijina, koska Open Source on julkisesti saatavissa. Oppilaat voivat opetella tekemään parempia ohjelmia, koska ovat opiskelleet, mitä muut ovat koodanneet Open Source-ohjelmiin. He voivat myös jakaa työnsä muiden kanssa ja pyytää kommenttia sekä kritiikkiä. Eli Open Sourcen hyötyjä ovat vapaa muokkaus ja jakelu, vapaa pääsy lähteeseen, korkeatasoinen yhteistyö sekä se, että se estää yhden toimijan lähdekoodin ”lukitsemisen”. (What open source 2016.)

4.2 Järjestelmän hyödyt

Tarkastusjärjestelmän toteuttaminen suljettuun ympäristöön tuo Internetissä tarjottaviin verrattuna tärkeitä etuja. Verkossa muutamat sivut tarjoavat käyttäjille mahdollisuuden tarkistaa tiedostoja monella eri antivirusmoottorilla. Internetissä tarjottavia tarkastuspalveluita ovat muun muassa VirusTotal, Metascan Online ja JOTTI. Käyttäjien ei tarvitse asentaa antivirusohjelmia yhdelle tietokoneelle tai tarkastaa tiedostoja eri ympäristöissä, mikä on kätevää, kun tiedostoja halutaan tarkastaa monella eri antivirusohjelmalla. Ongelmaksi koituu, kun tarkastettavat tiedostot ovat salaisia tai muuten sensitiivisiä. Esimerkiksi suurin verkossa toimiva monen antivirusohjelman skanneri, Googlen omistama VirusTotal lähettää kaikki tarkastettavat tiedostot eteenpäin antivirusohjelmien tekijöille. Ennen VirusTotalissa oli optio, että tarkastettavia tiedostoja ei lähetetä eteenpäin yhtiöille, mutta se otettiin pois rikollisten mahdollisen väärinkäytön takia. Koska online-palveluissa tarkastettavat tiedostot lähetetään eteenpäin, ei salaisia ja herkkiä tiedostoja ole järkevää tarkistaa verkossa olevilla skannereilla.

Yrityksen ei kannata käyttää online-skannereita myöskään sen takia, että se voi paljastaa rikolliselle hänen haittaohjelman kiinnijäämisen. Kun online-palvelussa osa skannereista tunnistaa haittakoodin, haittakoodin näyte lähetetään myös muille antivirusyrityksille, jotta he voivat lisätä näytteen heidän omaan näytetietokantaansa. Tämä tulee ongelmaksi hienostuneissa hyökkäyksissä, koska niissä hyökkääjä on vakoillut yrityksen käyttämän virustorjuntaohjelmiston ja muokannut haittakoodia siten, että se läpäisee yrityksen käyttämän torjuntaohjelmiston. Jos haittaohjelma lähetetään online-skannereiden tarkastettavaksi ja osa skannereista tunnistaa ohjelman haitalliseksi, tiedon saa myös melko varmasti kohdeyrityksen käyttämä antivirusyhtiö. Kun rikollinen ajaa haittaohjelmaansa uudelleen yrityksen käyttämän skannerin läpi, se jääkin kiinni, mikä johtaa haittaohjelman muokkaamiseen. Jos rikollinen ei ikinä saa tietää jääneensä kiinni, hän jatkaa saman haittaohjelman jakamista, joka on positiivinen asia kohdeyrityksen kannalta.

VirusTotal ja sen kaltaiset online-skanneripalvelut verkossa eivät myöskään ole Open Sourcea. Ei voida tietää, miten antivirusskannerit ovat konfiguroitu, kuten esimerkiksi mitä optioita skannauksen aikana viedään. Heuristiikka ja arkistojen purkamisoptiot voivat olla otettu pois käytöstä, jotta skannaus suoritettaisiin mahdollisimman nopeasti. Tämän takia joitain epäilyttäviä tiedostoja voi jäädä huomaamatta tarkastuksessa. Omassa järjestelmässä voidaan konfiguroida skannereiden heuristiikka tasoja, sekä monia muita ohjelmakohtaisia asetuksia.

Verkossa oleviin palveluihin käyttäjä voi lähettää vain yhden tiedoston kerrallaan tarkastettavaksi, sekä tiedostolle on asetettu maksimikoko, esimerkiksi VirusTotalissa tiedoston maksimikoko saa olla 128MB. Toteutetussa järjestelmässä on tarkoitus, että käyttäjä voi lähettää monta tiedostoa kerralla tarkastettavaksi, sekä tiedostojen maksimikoot ovat muokattavissa skannerikohtaisesti.

4.3 Järjestelmien vertailu

Internetistä löytyy muutamia avoimen lähdekoodin alustoja ja toteutuksia koskien keskitettyä tiedostojen tarkastamista eri antivirusskannereilla. Yleisesti tiedostojen staattiseen analysointiin avoimen lähdekoodin alustoja löytyy useampia, mutta harvassa ovat alustat liittyen antivirusohjelmien keskittämiseen. Alustojen tarjonta ei siis ole huima ja monipuolisuudessa onkin isoja eroja. Vertailussa on hyvä ottaa huomioon, että nykypäivän haittaohjelmien torjunta ei ole pelkästään haittaohjelmasta oppimista, vaan myös saada hyvä yleiskuva tapahtumasta, eli missä ja milloin tiedosto on nähty ja mikä antivirusohjelma on havainnut sen.

Kaikki vertailtavat avoimen lähdekoodin julkaisut löytyvät githubista. Github on verkkopohjainen git repositorioiden hostauspalvelu, johon voi jakaa omia Open Source projektejaan tai kehittää muiden projekteja eteenpäin. Vertailuun on otettu neljä alustaa: MultiAV, Malice, PlagueScanner ja IRMA. IRMAn ollessa selvästi laajin kokonaisuus tässä tapauksessa, muita alustoja vertaillaan IRMAan.

4.3.1 MultiAV

Joxean Koretin tekemän avoimen lähdekoodin MultiAV wrapperin idea on toimia rajapintana, joka liimaa useita antivirusskannereita yhteen. IRMAan verrattuna MultiAV on vain yksinkertainen wrapperi Linux pohjaisille antivirusmoottoreille. MultiAV kuitenkin tukee kirjoitushetkellä 18 antivirusskanneria, joka on saman verran, kuin mitä IRMA tukee. MultiAV tarjoaa myös yksinkertaisen web-käyttöliittymän käytettäväksi, jonka takia MultiAV onkin myös varteenotettava vaihtoehto toteutukseen. Kuitenkin verrattuna IRMAn laajuuteen ja monipuolisuuteen MultiAV jää väistämättä toiselle sijalle.

4.3.2 Malice

Malicen ideana on luoda VirusTotalin kaltainen järjestelmä ja projektin slogan kuuluukin ”VirusTotal Wanna Be”. Kirjoitushetkellä Malice on kuitenkin vielä alkuvaiheessa, eikä alusta tue kuin seuraavia antivirusskannereita: Avg, Avast, Comodo ja F-Prot. Projektista ei myöskään ole olemassa kunnollista dokumentaatiota. Malicea ei GitHubin mukaan kehitetä kovin aktiivisesti varsinkin, jos verrataan IRMAan, joka on miltein päivittäisessä aktiivisessa kehityksessä.

4.3.3 PlagueScanner

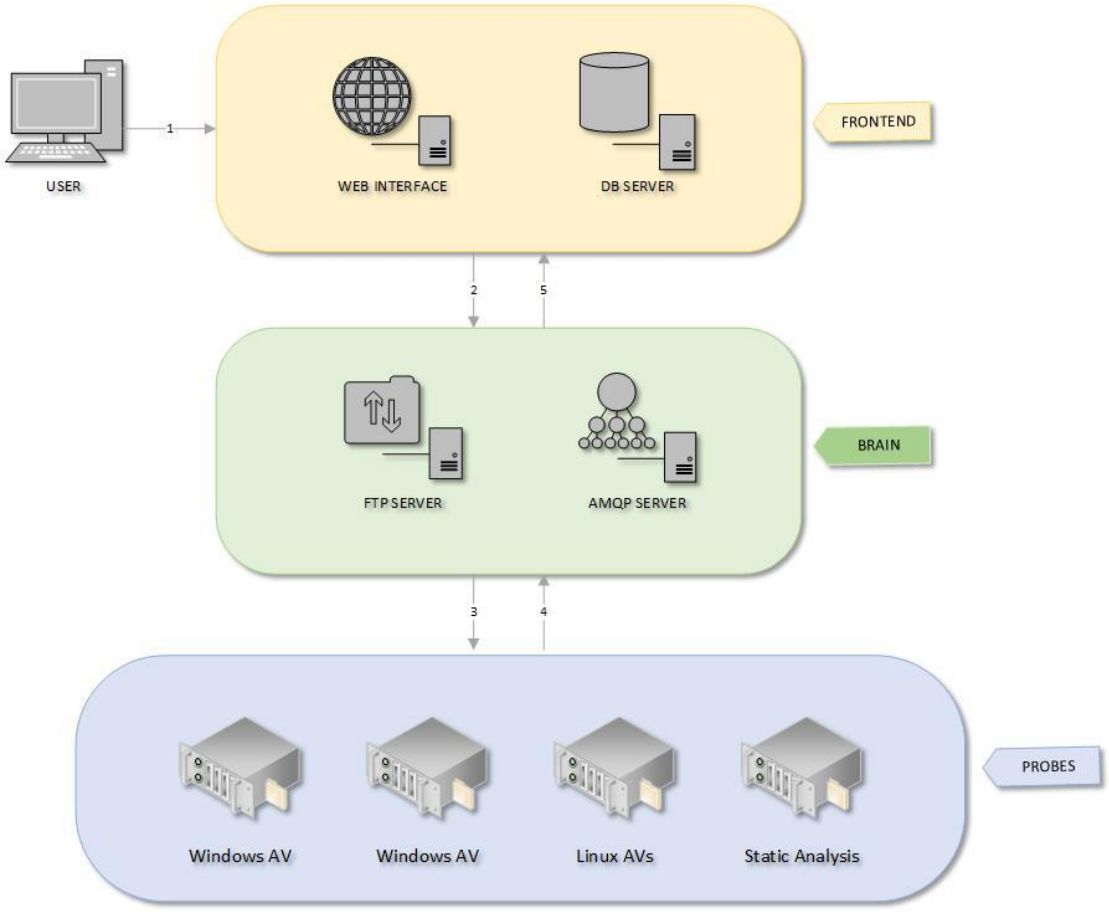
PlagueScannerissa on samanlainen ajatus, kuin aikaisemmin mainitulla MultiAVlla. Se niputtaa yhteen monta eri antivirusmoottoria ja toimii VirusTotalin kaltaisesti. PlagueScannerissa ei ole graafista käyttöliittymää, vaan se ajetaan komentorivin kautta. PlagueScanner tukee kuitenkin vain seuraavia kolmea antivirusohjelmaa: ClamAV, Eset ja BitDefender. PlagueScannerin kehitys on kirjoitus hetkellä ollut jäissä jo melkein vuoden, joten näyttäisi siltä, että PlagueScanner on haudattu projekti.

4.3.4 IRMA

IRMA on kolme osainen järjestelmä, joka koostuu Frontendistä, Brainista ja Probeista. IRMA tulee sanoista Incident Response and Malware Analysis eli suomennettuna tapahtumien hallinta ja haittaohjelmien analysointi. IRMA on Quarkslabin projekti ja on jatkuvan kehityksen alla. IRMAn tarkoituksena on olla Open Source-alusta, joka auttaa tunnistamaan ja analysoimaan haitallisia tiedostoja. Kun IRMA asennetaan omaan verkkoon, verkossa liikkuva data myös pysyy myös siellä. Tällä hetkellä IRMAn kehittäjät keskittyvät usealla antivirusohjelmalla skannaukseen yhtäaikaisesti, mutta IRMA ei rajoitu vain antivirusskannereihin, vaan myös muita tiedoston tarkastusmenetelmiä on mahdollista implementoida. (Readthedocs – IRMA 2016.) IRMAan on myös mahdollista implementoida myös omia analysointityökaluja. IRMAn suuri etu on myös se, että käyttäjä voi lähettää useamman tiedoston kerralla tarkastettavaksi, eikä vain yhtä kerrallaan. Liitteessä 1 on esitetty IRMAn tukemat ja siinä testatut antivirusskannerit. Kuten Liitteestä 1 nähdään, tuettuja skannereita on sekä Windows, että Linux käyttöjärjestelmille.

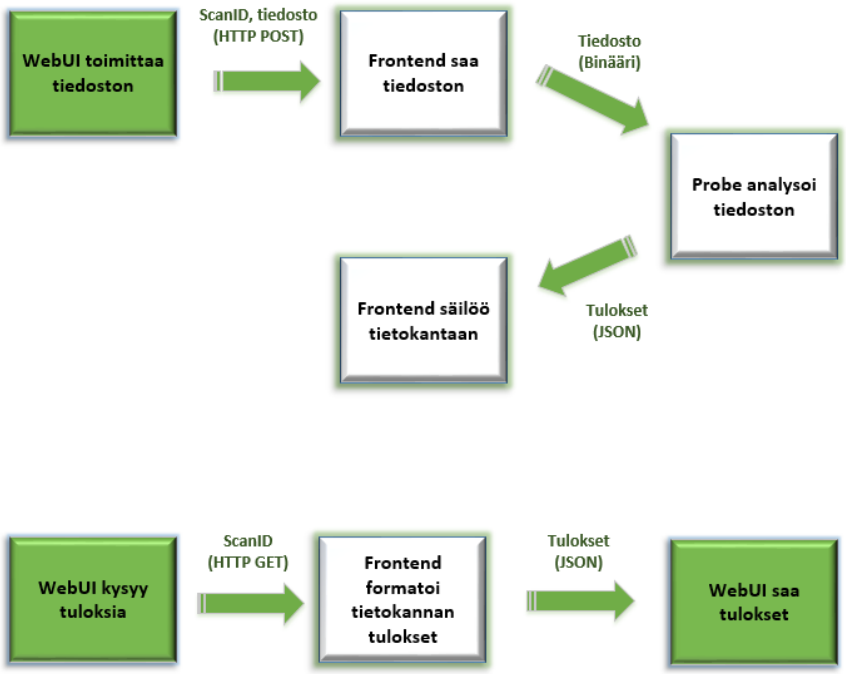
IRMAssa tiedostojen analysoiminen tapahtuu Kuvion 4 mukaisesti:

1. Analyysi alkaa, kun käyttäjä lataa tiedostot Frontendiin (web-sivulle).
2. Frontend tarkistaa onko tiedostoja olemassa jo tietokannassa ja antaa tarvittaessa skannaustehtävän Brainille.
3. Brain muokkaa ja lähettää skannaustehtäviä Probeille
4. Probet skannaavat tiedostot ja lähettävät tulokset takaisin Brainille
5. Brain lähettää tulokset Frontendille (näytetään web-sivulla.)



Kuvio 4. Tiedostojen analysointiprosessi IRMAssa. (IRMA 2016.)

Kuviossa 5 on esitetty, miten IRMAssa käsitellään tuloksia ja miten niiden työnkulku järjestelmässä menee. Frontend saa skannattavan tiedoston ja skannaus IDn web-sivulta HTTP POSTin kautta. Frontend taas lähettää tiedoston FTP-palvelimelle ja skannaus IDn Brainille, josta Probe saa ne tarkastettavaksi binääri muodossa. Tarkastuksen jälkeen tulokset palautetaan Brainin kautta Frontendille JSON-formatoidussa muodossa ja ne raakatulokset tallennetaan Frontendin MongoDB tietokantaan. Web-sivu saa kootusti tulokset näkyviin lähettämällä HTTP GETin Frontendille, joka taas hakee tulokset tietokannasta. Kun taas käyttäjä etsii web-sivulla (Search-sivu) jo skannattujen tiedostojen tuloksia, ne saadaan sivulle esiin samalla HTTP GET-metodilla.



Kuvio 5. Analysaattoreiden tuloksien työnkulku

5 Järjestelmän toteutus

Toteutettavaksi alustaksi valittiin IRMA sen monipuolisuuden takia. IRMAn kaikki komponentit ja skannerit asennettiin yhdelle virtuaalikoneelle. Käyttöjärjestelmänä toimi Debian Jessie 8.4 eli kaikki toteutetut antivirusskannerit olivat Linux-järjestelmälle. Windows käyttöjärjestelmälle oli mahdollista toteuttaa myös muutama antivirusskanneri, mutta tässä työssä keskityttiin vain Linuxille tarkoitettuihin antivirusskannereihin, koska niitä oli tuettu toistakymmentä ja alusta asennettiin ”all-in-one” tyyppiseksi. Alustaan saatiin toimintaan kaikki 13 tuettua antivirusskanneria ja lisäksi asennettiin muutama muu tiedoston analysointimenetelmä.

Virtuaaliympäristönä toimi VirtualBox, joka ilmainen ja yksi suosituimmista virtualisointiin tarkoitetusta ympäristöstä. VirtualBox on Orcalen kehittämä ja on myös Open Sourcea (dual-licence). Virtuaaliympäristössä on mahdollista ajaa montaa eri käyttöjärjestelmää yhtäaikaisesti yhden isäntäjärjestelmän alaisuudessa. Virtuaalikoneita voidaan hallita monipuolisesti ja komponentteja, kuten verkkokortteja voidaan lisätä ja poistaa.

5.1 Käytettävät tekniikat ja komponentit

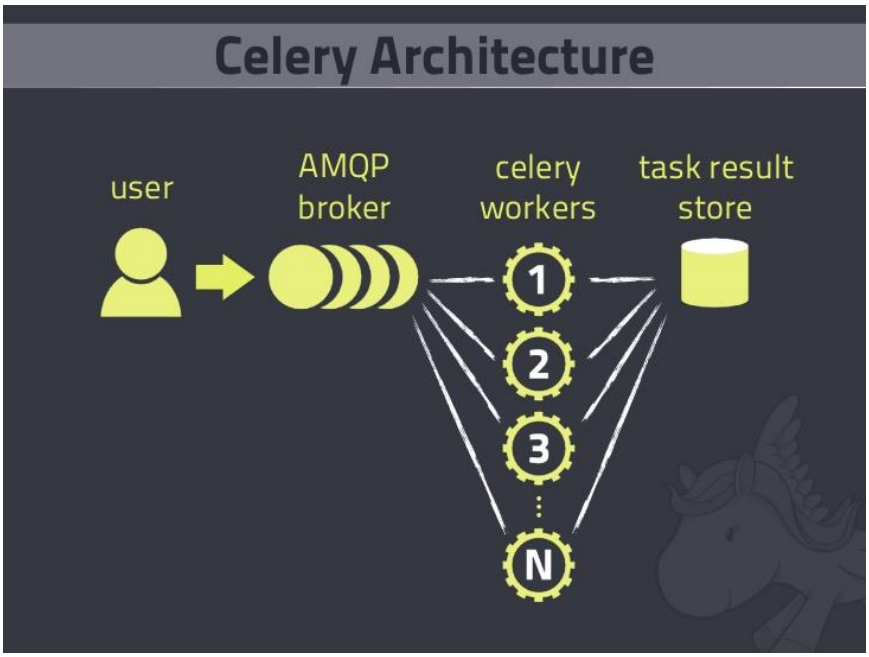
5.1.1 Python

Python on tulkattava ja oliosuuntaunut korkean tason ohjelmointikieli. Sitä voidaan käyttää muun muassa skriptaamiseen, sekä sen avulla voidaan yhdistää jo olemassa olevia komponentteja yhteen. Python on yksinkertainen ja helppo kieli kirjoittaa, sekä ymmärtää. Python tukee moduuleja ja paketteja, joka kannustaa modulaarisuuteen ja koodin uudelleenkäyttöön. (What is Python? Executive Summary 2016) IRMAn ohjelmoinnissa on käytetty Pythonia. Myös suurin osa IRMAssa käytettävistä komponenteista on Python-pohjaisia.

5.1.2 Celery

Celery on Python-pohjainen Open Source sovelluskehys (framework) tehtävien reaaliaikaiseen operointiin, sekä tarvittaessa tehtävien ajastamiseen. Sitä on helppo käyttää ja ylläpitää, eikä se tarvitse konfiguraatiotiedostoja. Celeryllä voidaan suorittaa tehtäviä asynkronisesti tai synkronisesti. Asynkroninen suorittaa tehtäviä taustalla, kun taas synkronisessa suorittamisessa pitää odottaa, kunnes tehtävä on valmis. (An Introduction to Celery 2009.)

Celeryn arkkitehtuuri koostuu viestien välittäjästä (broker), työskentelijöistä (workers) ja tietokannasta, johon tehtävien tulokset tallennetaan. Kun tehtävä toimitetaan välittäjälle, se työntää alitehtäviä workereille. Workerit suorittavat halutun tehtävän ja tulokset tallennetaan tietokantaan. Kuviosta 6 nähdään myös graafisesti Celeryn toiminta yksinkertaisimmillaan. (An Introduction to Celery 2009.)

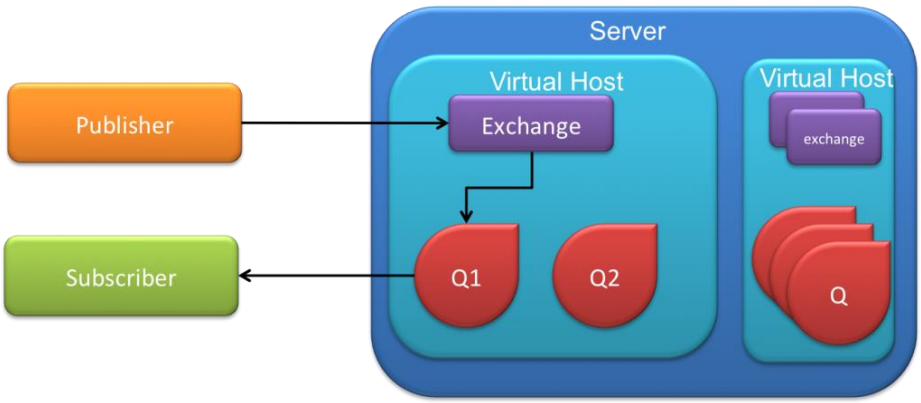


Kuvio 6. Celeryn arkkitehtuuri (An Introduction to Celery 2009.)

5.1.3 RabbitMQ

RabbitMQ on kevyt ja skaalautuva Erlang kielellä ohjelmoitu Open Source ohjelmisto, joka implementoi AMQP viestin välitysprotokollan. Yksinkertaisesti RabbitMQ on ohjelmisto, johon voidaan määrittää jonoja ja sovellukset voivat siirtää jonojen kautta erilaisia viestejä. Viesti voi sisältää esimerkiksi tehtävän, joka on tarkoitus suorittaa toisessa sovelluksessa. (Johansson 2015.)

RabbitMQ koostuu erilaisista komponenteista. Virtuaalihosti on kontaineri, joka tarjoaa pääsynvalvontaa eli virtuaalihostille luodaan käyttäjät ja salasanat, joilla kontarineria voi käyttää. Jokaiseen kontaineriin sisältyy oma joukko reititysagentteja (exchange). Viestit tuodaan reititysagentille ja se reitittää viestit jonoihin, jotka ovat määritelty tälle agentille. Jos agentille ei ole määritelty jonoja, viesti katoaa. Jonojen määrittely agenteille tapahtuu bindaamalla. Jonotuksessa käytetään tekniikkana FIFO-menetelmää eli first in first out. Kun viesti on jonossa niin viestin kuluttajat saavat sen sieltä ja voivat suorittaa tehtävänsä. (RabbitMQ and a Short Intro to AMQP 2012.) Kuviosta 7 nähdään graafisesti RabbitMQn toiminta virtuaalihostien kanssa.



Kuvio 7. RabbitMQn toiminta (Rotem-gal-oz 2012.)

5.1.4 Pure-FTPd

PureFtpd on ilmainen ja turvallinen FTP-palvelin, joka keskittyy tehokkuuteen ja helppokäyttöisyyteen. PureFtpdtä kehitetään aktiivisesti pitämällä turvallisuutta ensisijaisesti silmällä. PureFtpd tukee SSL/TLS salausta, sekä palvelinta voidaan ajaa tarvittaessa täysin ilman root oikeuksia. Palvelinta voidaan ajaa erilaisissa Unix-pohjaisissa käyttöjärjestelmissä, kuten Linux ja Android. (Pure-FTPd 2015.) IRMAssa Pure-Ftpdtä käytetään tarkastettavien tiedostojen väliaikaisena säilytyspaikkana. Frontend lähettää itse tiedoston FTP-palvelimelle, josta Probe noutaa sen tarkastettavaksi. Tiedosto poistetaan FTP-palvelimelta tarkastamisen jälkeen.

5.1.5 SQLite

SQLite on sulautettu relaatiotietokanta, joka implementoi palvelittoman SQL tietokantamoottorin ilman ylimääräisä konfiguraatioita. SQLite on ilmainen, C-kielellä ohjelmoitu, nopea, luotettava ja yksinkertainen käyttää. SQLiteä kehittää kansainvälinen tiimi, joka työskentelee SQLiten kanssa kokoaikaisesti. (About SQLite 2016.) IRMAssa SQLite tietokantaa käytetään Brainilla ja sen avulla seurataan skannaustehtävien tilaa (status). Tämä auttaa siinä tilanteessa, kun web-rajapinnan käyttäjä haluaa peruuttaa keskeneräisen skannauksen. Tällöin SQLite kertoo Brainille mitkä aliskannaustehtävät eivät ole suorittaneet loppuun ja tappaa nämä kesken jääneet tehtävät.

5.1.6 MongoDB

MongoDB on Open Source NoSQL tietokanta, joka käyttää dokumentti-orientoitunutta tietomallia. MongoDB ei siis ole relaatiotietokanta, vaan puolestaan käyttää dynaamisia skeemoja. MongoDB on korkean suorituskyvyn omaava, skaalautuva tietokanta ja nimi Mongo tuleekin sanasta ”humongous”. (What is MongoDB 2016.) MongoDB tietokantaa käytetään IRMAssa analysaattorien raakatuloksien säilömiseen. Raakatulokset tallennetaan tietokantaan JSON-formatoituna.

5.1.7 PostgreSQL

PostgreSQL on tehokas objektirelaatiotietokannan hallintajärjestelmä. PostgreSQL on ilmainen ja Open Sourcea, sekä se on yksi kehittyneimmistä tietokannoista. PostgreSQLn yksi tavoitteista on säilöä dataa turvallisesti. PostgreSQL on kirjoitettu C-kielellä ja sen kilpailijoita ovat muun muassa Oracle DB, SQL Server ja MySQL. PostgreSQL on suunniteltu Unix-pohjaisille käyttöjärjestelmille, mutta sitä voidaan käyttää nykyään myös Mac, Solaris ja Windows alustoilla. (PostgreSQL – About 2016.)

IRMAssa Mongoon tallennetaan analysaattoreiden raakatulokset, mutta PostgreSQL tietokantaan tallennetaan kaikki muu data skannauksesta. Nämä muut tiedot sisältävät tiedoston nimen, koon, MIME-tyypin, tarkastussummat (MD5, SHA-1, SHA-256) ja ensimmäisen, sekä edellisen skannauksen päivämäärän. PostqreSQL tietokantaan tallennetaan myös Mongo ID, jonka avulla voidaan MongoDB tietokannasta hakea analysaattoreiden JSON-formatoidut tulokset.

5.1.8 Bottle

Bottle on minimaalinen Python-pohjainen web sovelluskehys. Se on yhden tiedoston moduuli, sekä sillä ei ole muita riippuvuuksia, kuin Pythonin standardi kirjasto. Bottle on WSGI-yhteensopiva, kevyt, nopea ja helppokäyttöinen, sekä sopii hyvin RESTful palveluiden rakentamiseen. (The uWSGI Project 2014.) Bottle sovelluskehyksellä on IRMAssa toteutettu Frontendin RESTful ohjelmointirajapinta. Se mahdollistaa SQL ja NoSQL kyselyt tietokantoihin, sekä asynkronisten tehtävien aikataulutuksen Brainille.

5.1.9 Nginx

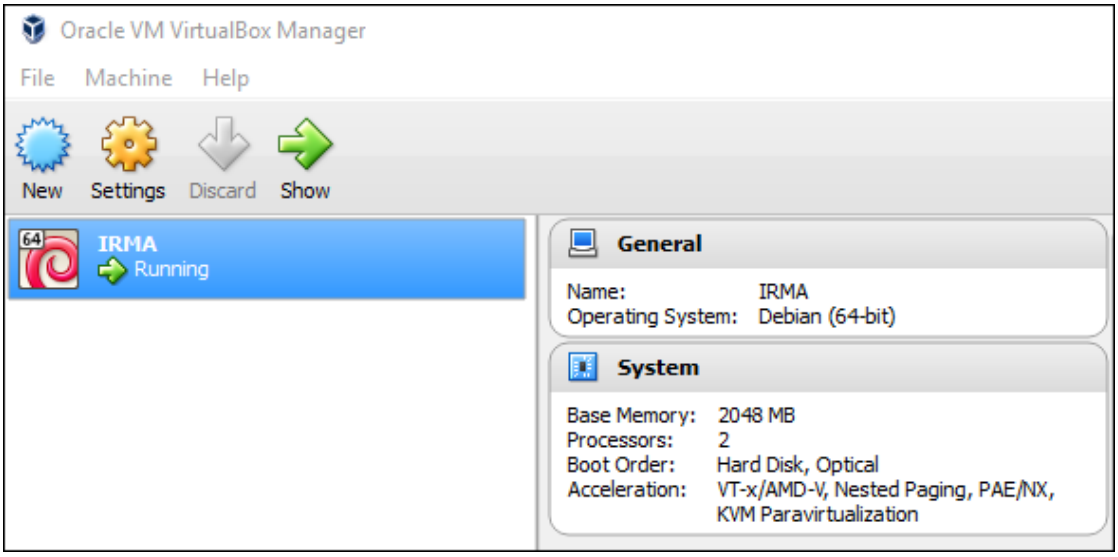
Nginx (lausutaan engine-X) on ilmainen ja Open Source web-palvelin. Nginx on tunnettu korkeasta suorituskyvystään ja vakaudestaan, jonka takia se onkin yksi suosituimmista web-palvelimista markkinoilla. Nginx on tapahtumapohjainen web-palvelin, sekä se on nopea ja helppo konfiguroida käyttöön. (Nginx Wiki 2016.) Nginx web-palvelinta käytetään Frontendillä ja sen staattiset web-sivut näkyvät käyttäjälle. Nginx kyselee ohjelmointirajapinnalta (Bottle API) tiedoston skannaustuloksia ja esittää tulokset käyttäjälle web-sivulla. Nginx palvelee uWSGI sovellusta.

5.1.10 uWSGI

uWSGI palvelin on nopea, itsestään korjaantuva ja kehittäjäystävällinen sovelluspalvelin, joka on koodattu C-kielellä. uWSGI projekti tähtää hostaus palveluiden full-stackin kehittämiseen ja on uWSGI yhteensopiva Bottle, Flask, sekä Django sovelluskehyksien kanssa. WSGI ja uwsgi ovat protokollia, joita ei pidä taas sekoittaa uWSGI palvelimen kanssa. uWSGI on taas palvelin, joka implementoi esimerkiksi WSGI-protokollan. (The uWSGI Project 2014.) uWSGI palvelee RESTful ohjelmointirajapintaa Frontendilla. uWSGI palvelin tarjoaa siis käyttöympäristön web-palvelimen ja WSGI-yhteensopivan sovelluksen välille.

6 Järjestelmän asennus

Järjestelmän asennus aloitettiin lataamalla Debianin kotisivuilta Debian Jessien (8.4) ISO-levykuva ja asennettiin se VirtualBoxiin aluksi perusasennuksella, ilman graafista käyttöliittymää, mutta asennuksen jälkeen asennettiin gnome-käyttöliittymä. Asennusta ei käydä vaihe vaiheelta läpi, mutta Debian käyttöjärjestelmälle annettiin 2048 MB keskusmuistia ja prosessorille kaksi ydintä. Kuviossa 8 nähdään Debian asennettuna VirtualBoxiin.



Kuvio 8. Debian asennettuna VirtualBoxiin

6.1 Brain

Brainilla toimivat scan_app ja results_app celery workerit. Scan_app käsittelee skannauspyynnöt Frontendilta ja results_app käsittelee skannaustulokset Probelta. Taustalla Celery käyttää RabbitMQta tehtävien välittämiseen ja niiden jonottamiseen. Brainilla toimii myös FTP-palvelin, jonka kautta itse tarkastettavat tiedostot kulkevat.

RabbitMQ

Lisätään RabbitMQn repositorio (deb http://www.rabbitmq.com/debian/ testing main) sources.list-tiedostoon:

sudo nano /etc/apt/sources.list

Ladataan RabbitMQn julkinen avain ja lisätään se luotetuksi:

sudo wget https://www.rabbitmq.com/rabbitmq-signing-key-public.asc

sudo apt-key add rabbitmq-signing-key-public.asc

Päivitetään repositoriot ja asennetaan RabbitMQ palvelin:

sudo apt-get update

sudo apt-get install rabbitmq-server

Luodaan RabbitMQlle käyttäjät ja salasanat, virtuaaliset hostit ja asetetaan käyttäjille pääsyoikeudet vastaaville virtuaalihosteille. Luodut käyttäjät ja virtuaaliset hostit voidaan tarkistaa Kuvioiden 9 ja 10 mukaisesti.

sudo rabbitmqctl add_user brain brain

sudo rabbitmqctl add_vhost mqbrain

sudo rabbitmqctl set_permissions -p mqbrain brain "." ".*" ".*"*

sudo rabbitmqctl add_user frontend frontend

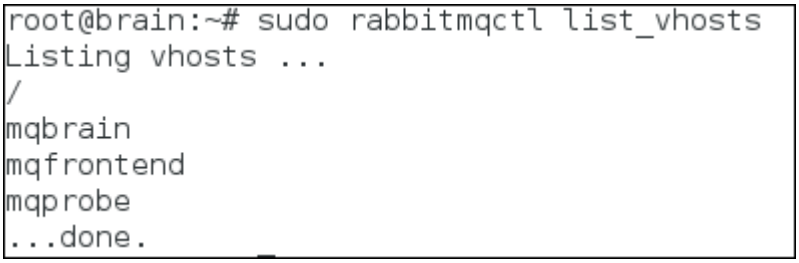
sudo rabbitmqctl add_vhost mqfrontend

sudo rabbitmqctl set_permissions -p mqfrontend frontend "." ".*" ".*"*

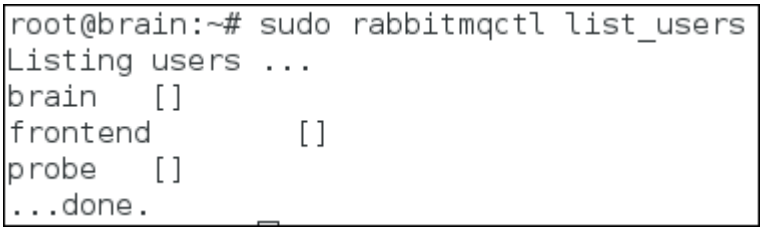
sudo rabbitmqctl add_user probe probe

sudo rabbitmqctl add_vhost mqprobe

sudo rabbitmqctl set_permissions -p mqprobe probe "." ".*" ".*"*



Kuvio 9. RabbitMQ virtuaalihostit



Kuvio 10. RabbitMQ käyttäjät

Tarvittaessa käyttäjän salasana voidaan uusia komennolla:

sudo rabbitmqctl change_password <käyttäjä> <uusi salasana>

Lopuksi käynnistetään RabbitMQ palvelin uudestaan:

sudo service rabbitmq-server restart

Pure-FTPd

Asennetaan Pure-FTPd:

sudo apt-get install pure-ftpd

Luodaan käyttöjärjestelmään käyttäjä ja ryhmä FTPtä varten. Liitetään myös käyttäjä luotuun ryhmään:

sudo groupadd ftpgroup
sudo useradd -g ftpgroup -d /dev/null -s /etc ftpuser

Luodaan itseallekirjoitettu varmenne FTP-palvelimen TLS-salausta varten:

sudo mkdir -p /etc/ssl/private/

cd /etc/ssl/private/

sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 -subj "/CN=$(hostname --fqdn)/" -keyout pure-ftpd.pem -out pure-ftpd.pem

sudo chmod 600 pure-ftpd.pem

IRMA tarjoaa valmiin mallin FTP-palvelimen konfiguroimiseen, kopioidaan mallit niille tarkoitettuihin polkuihin:

sudo cp /opt/Irma/Irma-brain/extras/pure-ftpd/conf/ /etc/pure-ftpd/conf/*

sudo ln -s /etc/pure-ftpd/conf/PureDB /etc/pure-ftpd/auth/50puredb

Luodaan vielä käyttäjät, jotka tulevat käyttämään FTP-palvelinta. Luodaan virtuaalikäyttäjät IRMAn tarjoamalla skriptillä *ftpd-adduser.sh,* jota käytetään seuraavasti:

sudo ftpd-adduser.sh <käyttäjä> <virtuaalikäyttäjä> <kotikansio>

Ensin luodaan kuitenkin kansiot, joita virtuaalikäyttäjät käyttävät, sekä asetetaan oikeudet kansioille. Kun skripti ajetaan, pitää sen jälkeen syöttää salasana virtuaalikäyttäjälle kahdesti.

sudo mkdir -pv /home/ftpuser/frontend
sudo chown -R ftpuser:ftpgroup /home/ftpuser

sudo extras/scripts/pure-ftpd/ftpd-adduser.sh frontend ftpuser /home/ftpuser/frontend

sudo extras/scripts/pure-ftpd/ftpd-adduser.sh probe ftpuser /home/ftpuser/

Lopuksi käynnistetään FTP-palvelin uudestaan:

sudo service pure-ftpd restart

BRAIN

Brainin lähdekoodi Quarkslabin git-repositoriosta, mutta ennen sitä palvelimelle pitää olla asennettuna Python 2.7.x ja pip.

Ensin asennetaan vaaditut paketit:

sudo apt-get install build-essential checkinstall

sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev

Ladataan Python 2.7.9 ja asennetaan se:

cd /usr/src

wget https://www.python.org/ftp/python/2.7.9/Python-2.7.9.tgz

tar xzf Python-2.7.9.tgz

cd Python-2.7.9

sudo ./configure

sudo make altinstall

Asennetaan vielä pip:

wget https://bootstrap.pypa.io/get-pip.py

python get-pip.py

Kloonataan Brainin lähdekoodi rekursiivisesti git-repositoriosta polkuun /opt/irma/irma-brain:

git clone --recursive https://github.com/quarkslab/irma-brain /opt/irma/irma-brain

Kloonauksen jälkeen asennetaan Python-riippuvaisuudet manuaalisesti pipin avulla. Riippuvaisuuksiin sisältyvät Celery, Redis, Sqlalchemy, Nose, Coverage, Pylint ja Mock.

pip install -r /opt/irma/irma-brain/requirements.txt

Riippuvaisuuksien asentamisen jälkeen konfiguroidaan Brain sen konfiguraatiotiedostosta /opt/irma/irma-brain/config/brain.ini. Tiedostoon määritellään jokaisen välittäjän (brain, probe ja frontend), FTP-palvelimen ja tietokannan asetukset. RabbitMQ välittäjille ja FTP-palvelimen osoitteeksi asetetaan localhost eli 127.0.0.1, sekä Jokaiselle välittäjälle konfiguroidaan omat virtuaalihostit, tunnukset ja jonot. Brainin SQLitelle ei luonnollisesti tarvitse dialektia, käyttäjätunnusta, salasanaa tai hostia, koska tietokanta on käytännössä vain tiedosto. Brainin konfiguraatiot ja selitykset ovat nähtävissä Liitteessä 2.

SQLite käyttäjätietokannan luomiseen on olemassa python-skripti ”scripts”-kansiossa, jota käytetään tyyliin:

python -m scripts.create_user <käyttäjänimi> <rmqvhosti> <ftp käyttäjä> [quota]

Ennen skriptin ajamista, pitää konfiguraatiotiedostoon määritetyn tietokantapolun olla luotuna. Luodaan kansiot komennolla:

mkdir -p /var/irma/db

Ajetaan skripti irma-brain kansiossa halutuilla parametreilla. Quota tarkoittaa tiedostojen skannauskiintiötä, joten asetetaan sen arvoksi 0, eli pois käytöstä.

python -m scripts.create_user frontend mqfrontend frontend 0

Käyttäjällä, joka ajaa celery workeria pitää olla oikeudet kirjoittaa SQLite-tiedostoon. Vaihdetaan tietokannan omistajaa, sekä asetetaan kaikille kirjoitusoikeus:

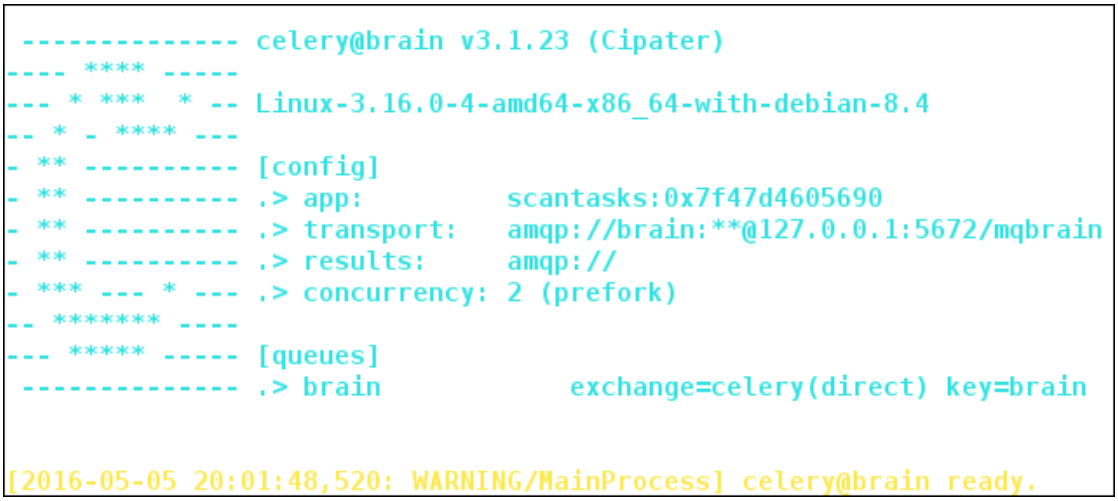
sudo chown irma:irma /var/irma/db/brain.db

sudo chmod a+w /opt/irma/irma-brain

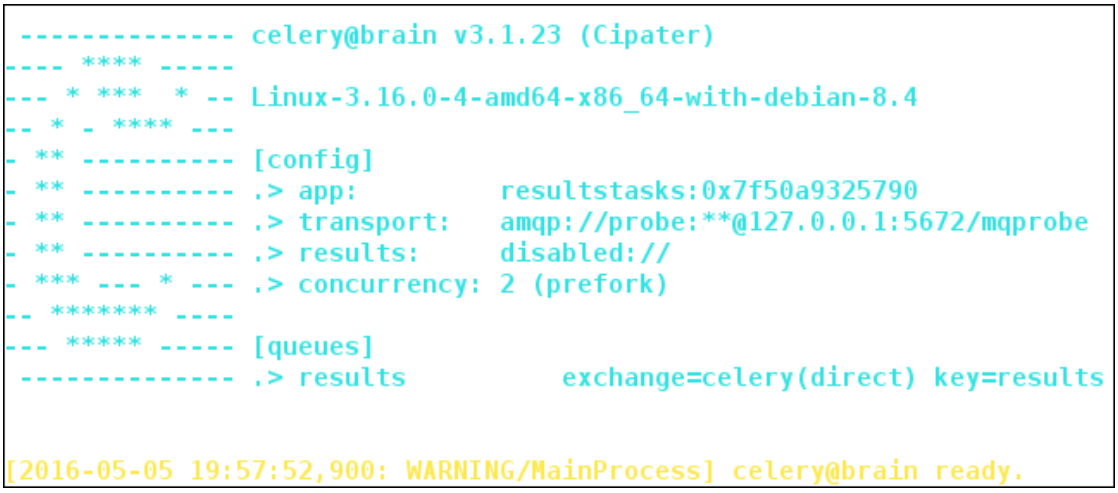
Testataan python sovellusten toimivuus celeryn kautta RabbitMQlle ajamalla molemmat celery workerit scan_app (Kuvio 11) ja results_app (Kuvio 12). Scan_app vastaa kokonaisen skannaustehtävän jakamisesta useampiin alitehtäviin probelle. Results_app vastaa taas tulosten keräämisestä ja seurannasta. Celery workerit saadaan ajettua manuaalisesti komennoilla:

celery worker --app=brain.tasks:scan_app --workdir=/opt/irma/irma-brain

celery worker --app=brain.tasks:results_app --workdir=/opt/irma/irma-brain



Kuvio 11. Celery worker scan_app



Kuvio 12. Celery worker results_app

Testataan myös FTP-palvelimen toimivuus SSL-salauksella lataamalla ftp-ssl ja otetaan yhteys FTP-palvelimeen. Liitteestä 3 nähdään FTP-palvelimen toimivuuden testaus.

sudo apt-get ftp-ssl

ftp-ssl localhost

6.2 Frontend

Frontend välittää tiedostojen skannauspyynnöt Brainille, sekä tallentaa tietokantaansa skannauksien tulokset. Skannauksien tulokset näytetään käyttäjälle graafisesti web-sivulla Frontendin toimesta. Frontendilla toimii python-pohjainen restful API, jota palvelee nginx web-palvelin ja uWSGI sovelluspalvelin. Frontend käyttää myös MongoDB ja Postgresql tietokantoja tulosten tallentamiseen. Postgresql tietokantaan tallennetaan kaikki metadata tiedostosta, kuten tiedoston nimi, päivämäärä, koko ja Mongo ID. Mongo IDn avulla haetaan MongoDBn tietokannasta itse skannauksien raakatulokset, jotka ovat formatoitu JSON-muotoon Proben toimesta.

Frontend vaatii myös python 2.7.x ja pipin asennuksen, mutta ne asennettiin jo aikaisemmassa vaiheessa, joten voidaan siirtyä suoraan lähdekoodin lataamiseen. Kloonataan Frontendin lähdekoodi rekursiivisesti git-repositoriosta polkuun /opt/irma/irma-frontend:

git clone --recursive https://github.com/quarkslab/irma-frontend /opt/irma/irma-frontend

Kloonaamisen jälkeen asennetaan riippuvaisuudet suoraan requirement.txt -tiedoston avulla. Riippuvaisuuksiin sisältyy PyYAML, Celery, Pymongo, Bottle, SQLAlchemy, Marshmallow, Bottle-sqlalchemy, Mock, Coverage, Requests ja Python-magic.

pip install -r /opt/irma/irma-frontend/requirements.txt

Web client muodostuu staattisista HTML-tiedostoista yhdessä JavaScriptin kanssa. Aloitetaan web clientin rakentaminen asentamalla aluksi Node.js ja npm:

curl -sL https://deb.nodesource.com/setup | sudo bash -

sudo apt-get install -y nodejs

curl -sL https://www.npmjs.org/install.sh | sudo bash -

Node.js ja npm asentamisen jälkeen rakennetaan sivut, jonka jälkeen web-kansion alle ilmestyy dist-kansio, jossa sijaitsee itse HTML- ja JavaScript tiedostot.

cd /opt/irma/irma-frontend/web

sudo npm install

sudo node_modules/.bin/bower install --allow-root

node_modules/.bin/gulp dist

Konfiguroidaan Frontend sen konfiguraatiotiedostosta config/frontend.ini. Tiedostoon konfiguroidaan MongoDB, Postgresql, tiedostojen näytteiden polku, RabbitMQ välittäjät, FTP-palvelin ja cron työ. Liitteestä 4 nähdään konfiguroidut arvot, sekä selitykset kentille. Konfiguraatioiden jälkeen luodaan kansiopolku näytteitä varten, joka on määritelty Frontendin konfiguraatiotiedostossa. Konfiguroimisen jälkeen siirrytään asentamaan itse MongoDB ja Postgresql tietokantapalvelimet, uWSGI sovelluspalvelin, sekä nginx web-palvelin.

mkdir -p /var/irma/samples

MongoDB

Lisätään MongoDBn julkinen GPG avain järjestelmän avainrenkaaseen:

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10

Lisätään MongoDBn repositorio ja päivitetään repositoriot:

echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list

sudo apt-get update

Asennetaan MongoDB palvelin:

sudo apt-get install -y mongodb-org-server

Asetetaan MongoDB kuuntelemaan vain loopback rajapintaa muokkaamalla /etc/mongodb.conf tiedostoa. Asetetaan loopbackin IP ja oletus MongoDBn portti

bind_ip = 127.0.0.1
port = 27017

PostgreSQL

Lisätään Postgren julkinen avain järjestelmän avainrenkaaseen:

*wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | *

sudo apt-key add -

Lisätään Postgren repositorio ja päivitetään repositoriot:

echo ' deb http://apt.postgresql.org/pub/repos/apt/ jessie-pgdg main' | sudo tee /etc/apt/sources.list.d/pgdg.list

sudo apt-get update

Asennetaan PostgreSQL:

apt-get install postgresql-9.4

Asennuksen jälkeen vaihdetaan postgres-käyttäjään, luodaan käyttäjä tietokannalle, sekä itse tietokanta. Käyttäjänimi, salasana ja tietokannan nimi asetetaan samaksi, kuin mitä Frontendin konfiguraatiotiedostossa on määritelty, jotta yhteys tietokantaan voidaan muodostaa.

su – postgres

createuser -P irma
Enter password for new role: irma
Enter it again: irma

createdb irma-frontend

Annetaan käyttäjälle irma kaikki oikeudet tietokantaan irma-frontend:

psql irma-frontend
GRANT ALL PRIVILEGES ON DATABASE "irma-frontend" to irma;

uWSGI sovelluspalvelin luo tietokantaan tarvittavat taulukot automaattisesti käynnistyessään, joten niitä ei tarvitse manuaalisesti lisätä. Kuviosta 13 nähdään luodut taulukot.

```
irma-frontend=# \dt
                List of relations
 Schema |           Name           | Type  | Owner
--------+--------------------------+-------+-------
 public | irma_file                | table | irma
 public | irma_fileAgent           | table | irma
 public | irma_fileWeb             | table | irma
 public | irma_probeResult         | table | irma
 public | irma_probeResult_fileWeb | table | irma
 public | irma_scan                | table | irma
 public | irma_scanEvents          | table | irma
 public | irma_submission          | table | irma
 public | irma_tag                 | table | irma
 public | irma_tag_file            | table | irma
(10 rows)
```

Kuvio 13. Automaattisesti luodut taulukot

uWSGI

Asennetaan viimeisin stable versio uWSGI palvelimesta, sekä python liitännäinen:

pip install uwsgi uwsgi-plugin-python

Asennuskansion alta löytyy extras-kansio, joka sisältää valmiita skriptejä. Kopioidaan uWSGI API-skripti uWSGIn apps-available kansioon ja luodaan siitä symbolinen linkki apps-enabled kansioon.

sudo cp extras/uwsgi/frontend-api.xml /etc/uwsgi/apps-available/

sudo ln -s /etc/uwsgi/apps-available/frontend-api.xml /etc/uwsgi/apps-enabled/frontend-api.xml

Lopuksi käynnistetään uWSGI palvelin uudelleen

sudo service uwsgi restart

Nginx

Asennetaan Nginx web-palvelin:

sudo apt-get install nginx

Kopioidaan nginx-palvelimen skripti ”frontend” tutusta extras-kansiosta nginxin sites-available kansioon ja luodaan siitä symbolinen linkki sites-enabled kansioon.

sudo cp extras/nginx/frontend /etc/nginx/sites-available/

sudo ln -s /etc/nginx/sites-available/frontend /etc/nginx/sites-enabled/frontend

Käynnistetään lopuksi nginx web-palvelin uudelleen:

sudo service nginx restart

Testataan, että python sovellus onnistuu kommunikoimaan RabbitMQn ja Rediksen kanssa celeryn kautta ajamalla manuaalisesti frontend_app celery worker (Kuvio 14). Tämän jälkeen todetaan vielä restful ohjelmointirajapinnan toimivuus (Kuvio 15).

celery worker --app=frontend.tasks:frontend_app --workdir=/opt/irma/irma-frontend

```
 -------------- celery@brain v3.1.23 (Cipater)
---- **** -----
--- * ***  * -- Linux-3.16.0-4-amd64-x86_64-with-debian-8.4
-- * - **** ---
- ** ---------- [config]
- ** ---------- .> app:         frontendtasks:0x7f0d6d995fd0
- ** ---------- .> transport:   amqp://frontend:**@127.0.0.1:5672/mqfrontend
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 2 (prefork)
-- ******* ----
--- ***** ----- [queues]
 -------------- .> frontend         exchange=celery(direct) key=frontend


[2016-05-08 17:11:34,146: WARNING/MainProcess] celery@brain ready.
```

Kuvio 14. Celery worker frontend_app

```
root@brain:~# curl http://localhost/api/v1/probes
{"total": 17, "data": ["StaticAnalyzer", "ClamAV", "Unarchive", "EsetNod32", "FProt
", "Sophos", "AVGAntiVirusFree", "VirusTotal", "BitdefenderForUnices", "EScan", "Zo
ner", "FSecure", "McAfeeVSCL", "ComodoCAVL", "AvastCoreSecurity", "DrWeb", "Yara"]}
```

Kuvio 15. Ohjelmointirajapinnan toimivuus

6.3 Probe

Probe on Python-pohjainen applikaatio, jonka takana itse skannerit sijaitsevat. Probe käsittelee skannauspyynnöt, jotka tulevat Brainilta. Jokaisen skannerin voi asentaa omalle Probelle tai kaikki skannerit voidaan asentaa samalle Probelle, kunhan ne eivät interferoi keskenään. Probe rekisteröi jokaisen skannerin Brainille, jonka jälkeen Brainille luodaan oma jono per skanneri. Jokainen skanneri kuuntelee omaa jonoansa ja kun jonoon tulee tehtävä, skanneri skannaa tiedoston ja palauttaa tulokset.

Analysaattorit rekisteröityvät omatoimisesti. Ensin liitännäisriippuvuudet tarkistetaan ja haetaan määritetystä polusta analysaattori. Kun rekisteröinti on onnistunut, analysaattori aktivoidaan. Aktivoinnin jälkeen Celery-jono luodaan jokaiselle rekisteröityneelle analysaattorille.

Yleensä moduulissa on vähintään seuraavat tiedostot:

init.py – Pakollinen python moduuli
pe.py – Erillinen analysaattori moduuli
plugin.py – Yhdistää analysaattorin celery workerin kanssa
requirements.txt – Python riippuvuudet analysaattorille

On olemassa antivirusskannereita, jotka toimivat pelkästään Windows käyttöjärjestelmällä. Probe applikaatio voidaan siis asentaa Linux- tai Windows käyttöjärjestelmälle, koska IRMA tukee sekä Linux, että Windows-skannereita. Tässä työssä kuitenkin keskityttiin all-in-one tyyppiseen toteutukseen eli Windows skannerit jätettiin pois toteutuksesta, koska käyttöjärjestelmänä toimi Linux.

Probe vaatii myös Python 2.7.x ja pipin asennuksen, mutta ne ovat jo asennettu käyttöjärjestelmään aikaisemmassa vaiheessa, joten voidaan siirtyä suoraan lähdekoodin kloonaamiseen git-repositoriosta.

git clone --recursive https://github.com/quarkslab/irma-probe

Kloonaamiseen jälkeen asennetaan Proben python riippuvaisuudet. Riippuvaisuuksiin sisältyy Celery, Redis ja python-magic.

pip install -r /opt/irma/irma-frontend/requirements.txt

Kun python applikaatio on asennettu, pitää se vielä konfiguroida koko järjestelmään konfiguroimalla config/probe.ini tiedosto Liitteen 5 mukaisesti. Skannereiden asentamisesta ja konfiguroinnista Probelle on kerrottu lisää kappaleessa ”Antivirusskannerit” ja ”Muut analysointimenetelmät”.

Ajetaan manuaalisesti celery worker probe_app, josta voidaan todeta, että jokaiselle tiedostoanalysaattorille on luotu oma jononsa (Kuvio 16).

celery worker --app=probe.tasks:probe_app --workdir=/opt/irma/irma-probe

```
 -------------- celery@brain v3.1.23 (Cipater)
---- **** -----
--- * ***  * -- Linux-3.16.0-4-amd64-x86_64-with-debian-8.4
-- * - **** ---
- ** ---------- [config]
- ** ---------- .> app:         probe.tasks:0x7fbe1a207050
- ** ---------- .> transport:   amqp://probe:**@127.0.0.1:5672/mqprobe
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 2 (prefork)
-- ******* ----
--- ***** ----- [queues]
 -------------- .> AVGAntiVirusFree exchange=celery(direct) key=AVGAntiVirusFree
                .> AvastCoreSecurity exchange=celery(direct) key=AvastCoreSecurity
                .> BitdefenderForUnices exchange=celery(direct) key=BitdefenderForUnices
                .> ClamAV           exchange=celery(direct) key=ClamAV
                .> ComodoCAVL       exchange=celery(direct) key=ComodoCAVL
                .> DrWeb            exchange=celery(direct) key=DrWeb
                .> EScan            exchange=celery(direct) key=EScan
                .> EsetNod32        exchange=celery(direct) key=EsetNod32
                .> FProt            exchange=celery(direct) key=FProt
                .> FSecure          exchange=celery(direct) key=FSecure
                .> McAfeeVSCL       exchange=celery(direct) key=McAfeeVSCL
                .> Sophos           exchange=celery(direct) key=Sophos
                .> StaticAnalyzer   exchange=celery(direct) key=StaticAnalyzer
                .> Unarchive        exchange=celery(direct) key=Unarchive
                .> VirusTotal       exchange=celery(direct) key=VirusTotal
                .> Yara             exchange=celery(direct) key=Yara
                .> Zoner            exchange=celery(direct) key=Zoner

[2016-05-08 19:00:31,652: WARNING/MainProcess] celery@brain ready.
```

Kuvio 16. Celery worker probe_app

6.4 Supervisor

Celery workerit voidaan Supervisorin avulla asettaa käynnistymään järjestelmän mukana automaattisesti. Supervisor on eräänlainen keskitetty prosessien hallintatyökalu, joka helpottaa monen eri prosessin hallittavuutta. Supervisor käynnistyy järjestelmän käynnistyessä ja se taas käynnistää määrättyjä aliprosesseja. Irmassa Supervisoria käytetään celery workereiden hallintaan. Jokaiselle workerille on oma konfiguraatiotiedostonsa (Kuvio 17), jonka avulla workerit käynnistyvät käyttöjärjestelmän käynnistyksen yhteydessä ja niitä voidaan hallita muun muassa komennoilla start, stop ja restart.

```
root@brain:/etc/supervisor/conf.d# ls
frontend_api  frontend_app  probe_app  result_app  scan_app
```

Kuvio 17. Supervisor konfiguraatiotiedostot

7 Analysaattoreiden implementointi

7.1 Antivirusskannerit

Useimmat antivirusskannerit ovat suljettua lähdekoodia (Closed Source), joten skannereiden lisensseistä on huolehdittava. Työssä käytetään Closed Source -ohjelmien osalta kokeiluversioita tai ilmaislisenssejä kotikäyttöön. Työssä ei kuitenkaan näytetä lisenssiavaimia, joten jos järjestelmä otetaan käyttöön muualla, niin luonnollisesti ohjelmien lisensointi jää järjestelmän ylläpitäjän vastuulle.

IRMAan antivirusskannerit asennetaan niiden oletuspolkuihin. Modules-kansio skannataan probe_appin käynnistyksen yhteydessä. Modules-kansiossa sijaitsevat liitännäiset skannereille. Liitännäisissä on annettu polku skannerille, joka on skannerin oletus asennuspolku. Skannerit voidaan myös asentaa muualle, kuin oletus asennuskansioon, mutta tällöin pitää muokata kyseisen skannerin liitännäisiä.

Kun antivirusskannereita asennetaan ja konfiguroidaan käyttöjärjestelmään, on tärkeää olla asentamatta tai jälkeenpäin poistaa käytöstä on-access skannaus eli taustalla toimiva reaaliaikaskannaus. Useamman kuin yhden reaaliaikaskannauksen päällä pitäminen samassa käyttöjärjestelmässä yhtäaikaisesti johtaa muun muassa muistin loppumiseen ja skannereiden kaatumiseen. Eri skannereiden reaaliaikaskannaukset interferoivat keskenään esimerkiksi siten, että skanneri A kopioi tarkastettavan tiedoston väliaikaiskansioonsa ja skanneri B huomaa tämän, jonka jälkeen skanneri B kopioi tiedoston skanneri A:n kansiosta omaan väliaikaiskansioonsa tarkastusta varten. Tämä tiedostontarkastus ja kopiointi toistuvat loputtomiin eli jää ”looppaamaan”. Tämä looppaus aiheuttaa edellä mainittua muistin loppumista, sekä skannereiden kaatumista.

Toteutetussa järjestelmässä antivirusskannereita on toistakymmentä, joten on-access skannauksen sammuttaminen on todella tärkeää. Antivirusskannerit asennetaan on-demand skannereiksi eli tarvittaessa kutsuttaviksi komentoriviskannereiksi. Tämän avulla vältytään siltä, että antivirusohjelmat eivät interferoi keskenään ja skanneri tarkistaa halutun tiedoston vain silloin, kun sitä kutsutaan.

Koska asentamamme käyttöjärjestelmä on 64-bittinen ja muutama antivirusskanneri tarjoaa vain 32-bittisen version Linuxille, pitää käyttöjärjestelmään lisätä i386-arkkitehtuuri. Tämä onnistuu dpkg:n avulla. Lisätään 32-bittinen arkkitehtuuri ja päivitetään repositoriot.

dpkg --add-architecture i386

apt-get update

Kuvion 18 mukaisilla komennoilla voidaan tarkistaa järjestelmän oma arkkitehtuuri, sekä myös vieraat arkkitehtuurit.

```
root@brain:/etc/avast# dpkg --print-architecture
amd64
root@brain:/etc/avast# dpkg --print-foreign-architectures
i386
```

Kuvio 18. Järjestelmän arkkitehtuurit

Antivirusskannereiden testaamisen yhteydessä tuli myös huomattua, että eScan, F-prot ja Avast eivät toimi supervisorin kanssa, joten yksi mahdollisuus oli pysäyttää probe_app supervisorin kautta ja manuaalisesti ajaa celery worker probe_app. Tätä manuaalista ajotapaa käytettiin, jotta saatiin kaikki antivirusskannerit toimimaan järjestelmässä.

7.1.1 Avast

Otetaan IRMAan käyttöön Avastin ”Avast Core Security”, joka on Linux-alustalle tehty antivirusskanneri. Avast Core Security on maksullinen, mutta tiedot kokeiluversion lataamiseen löytyvät Avastin kotisivuilta. Kokeiluversio kestää 30 päivää ja vaatii rekisteröitymisen. Rekisteröitymisen jälkeen saadaan sähköpostiin ohjeet ohjelman lataamiseen, sekä liitteenä lisenssitiedosto ja Avastin julkinen GPG-avain. Avastin asentaminen tapahtuu seuraavasti:

Lisätään Avastin repositorio

echo "deb http://deb.avast.com/lin/repo debian release" \ >> /etc/apt/sources.list

Ladataan sähköpostilla saatu julkinen GPG-avain ja lisätään se avainketjuun, jonka jälkeen päivitetään repositoriot ja asennetaan Avastin antivirusskanneri:

apt key add avast.gpg

apt-get update

apt-get install avast

Skannerin asentamisen jälkeen siirretään lisenssitiedosto polkuun */etc/avast/*:

mv license.avastlic /etc/avast/

Avast-käyttäjä pitää lisätä saman käyttäjän ryhmään, kuin kuka ajaa celery workeria, jotta Avastilla on oikeus hakea skannaustulos väliaikaiskansiosta. Esimerkiksi jos root-käyttäjä ajaa celery workeria, niin lisätään avast-käyttäjä ryhmään root. Muistetaan vielä, että ryhmämuutokset tulevat vasta voimaan, kun käyttäjä kirjautuu uudelleen.

sudo usermod -G root -a avast

Avastin näytetietokannat löytyvät polusta */var/lib/avast/defs* ja virustunnisteet voidaan päivittää manuaalisesti ajamalla avast.vpsupdate kansiossa */var/lib/avast/Setup.*

Avast voidaan päivittää myös ilman Internet-yhteyttä. Helpoin tapa on asentaa Avast Core Security toiselle koneelle ja päivittää se siellä avast.vpsupdaten avulla. Tämän jälkeen kopioidaan kansio */var/lib/avast/defs* ja siirretään IRMA palvelimelle. Kun virustunnisteita siirretään, pitää Avastin olla pois päältä.

Avastin Linux skannerissa ei ole reaaliaikaskannausta, joten sen disabloimisesta ei tarvitse huolehtia. Avast ajetaan järjestelmässä parametreilla *-b -f -u.*

7.1.2 AVG

AVGn 2013 antivirusskanneri Linuxille on ilmainen ja se voidaan ladata AVGn verkkosivuilta. Ladataan ja asennetaan AVGn antivirusskanneri:

apt-get install gdebi

wget http://download.avgfree.com/filedir/inst/avg2013flx-r3118-a6926.i386.deb

gdebi avg2013flx-r3118-a6926.i386.deb

Ohjelma asentuu kansioon /opt/avg/av ja ohjelman päivittäminen tapahtuu yksinkertaisesti komennolla:

sudo avgupdate

Ilman Internet-yhteyttä on mahdollista myös päivittää virustunnisteet. Tunnisteet ovat saatavissa AVGn verkkosivuilta osoitteesta http://www.avg.com/us-en/download-update-2013. Ladataan sivuilta .bin-tiedosto kuvauksella ”Virus definitions”. Ladatut tunnisteet voidaan asentaa ajamalla AVGn päivittäjä laittamalla source parametriksi folder ja antamalla polku, jossa tunnistetiedosto sijaitsee.

avgupdate --source=folder --path="<POLKU TIEDOSTOON>”

AVG command line update
Copyright (c) 2013 AVG Technologies CZ
Running update.
Initializing...
Analyzing...
100% [==========>]

You are currently up-to-date
Update was successfully completed.

AVGn skannerissa ei ole oletuksena päällä reaaliaikaskannausta, joten siitä ei tarvitse huolehtia tässä tapauksessa. AVG ajetaan järjestelmässä parametreilla: *--heur --paranoid --arc --macrow --pwdw –pup.*

7.1.3 Bitdefender

BitDefender for Unices on antivirusskanneri Linux ja FreeBSD järjestelmille. Skannerista voi ladata kokeiluversion tai voi pyytää ilmaista lisenssiä. Ilmainen lisenssi on tarkoitettu vain henkilökohtaiseen käyttöön. Rekisteröidytään sivulle, jotta saadaan ilmainen lisenssi. Kun rekisteröinti on tehty, saadaan sähköpostiin linkki ohjelman lataamiseksi, sekä lisenssiavain. Ladataan BitDefenderin asennustiedosto ja asennetaan se.

sudo bash BitDefender-Antivirus-Scanner-7.7-1-linux-amd64.deb.run

Skanneri asentuu kansioon */opt/BitDefender-scanner/,* jonka jälkeen voidaan laittaa sähköpostilla saatu lisenssiavain tiedostoon /opt/BitDefender-scanner/etc/bdscan.conf.

BitDefender voidaan päivittää ajamalla päivittäjä tai lataamalla virustunnisteet manuaalisesti verkkosivuilta. Päivitetään Internet-yhteyden avulla komennolla:

bdscan --update

Pakattu virustunnistetiedosto voidaan ladata osoitteesta http://download.bitdefender.com/updates/update_av64bit/. Pakattu tiedosto on nimeltään cumulative.zip, joten puretaan se suoraan kansioon, jossa sijaitsee BitDefenderin virustunnisteet eli */opt/BitDefender-scanner/var/lib/scan* ja korvataan vanhat tunnisteet.

unzip cumulative.zip -d /opt/BitDefender-scanner/var/lib/scan

BitDefenderin Linux skannerissa ei ole reaaliaikaskannausta, joten sen disabloimisesta ei tarvitse huolehtia. BitDefender ajetaan järjestelmässä parametreilla *--action=ignore --no-list.*

7.1.4 ClamAV

ClamAV on Open Source antivirusmoottori, jota voidaan käyttää sähköpostiskannaukseen, verkkoskannaukseen ja päätelaitteen turvaamiseksi. Se tarjoaa useita apuohjelmia, kuten joustavan ja skaalautuvan daemonin, komentoriviskannerin ja työkalun automaattiseen tietokannan päivittämiseen. ClamAVssa on sisäänrakennettu tuki muun muassa standardeille sähköposti formaateille ja monelle arkistointi formaateille, kuten Zip, RAR, TAR, Dmg, Gzip ja monta muuta. (About ClamAV 2016.)

Asennetaan ClamAV daemon komennolla:

sudo apt-get install clamav-daemon

ClamAVssa voidaan päivittää virustunnisteet manuaalisesti käyttämällä freshclamia:

sudo freshclam

Näytetietokannan päivittämisen jälkeen käynnistetään daemon uudelleen:

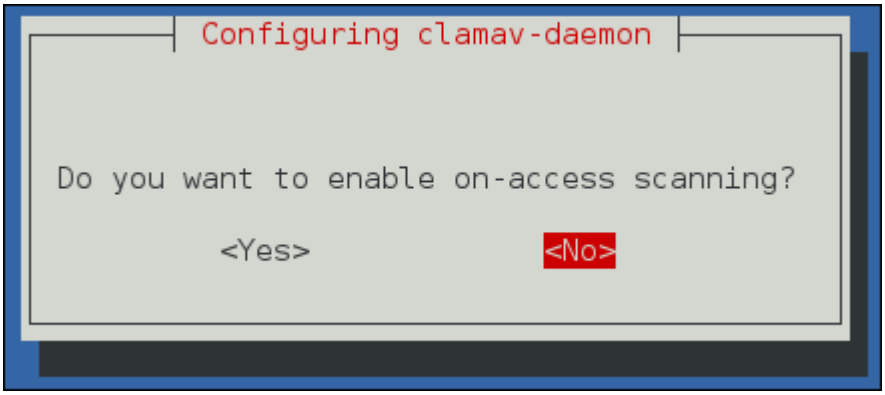
sudo service clamav-daemon restart

ClamAVn virustunnisteet sijaitsevat kansiossa /var/lib/clamav ja uudet tunnisteet voidaan ladata verkkosivuilta osoitteesta https://www.clamav.net/downloads. Sivuilta ”Virus Database” -valikon alta ladataan main ja daily tunnistetiedostot. Tämän jälkeen korvataan vanhat tiedostot uusilla polussa /var/lib/clamav.

ClamAVn skannaus konfiguraatioita voidaan muokata tiedostossa /etc/clamav/clamd.conf tai ohjeistettu konfiguroiminen voidaan ajaa komennolla:

sudo dpkg-reconfigure clamav-daemon

On-access skannaus voidaan ottaa pois päältä muun muassa ohjeistetun konfiguroimisen kautta yksinkertaisesti vastaamalla kysymykseen “No” (Kuvio 19). ClamAV ajetaan järjestelmässä parametreilla *--infected --fdpass --no-summary –stdout*.



Kuvio 19. Reaaliaikaskannauksen disabloiminen ClamAVssa

7.1.5 Comodo

Comodon antivirusskanneri Linuxille (Comodo Antivirus for Linux) on Ilmainen ja se on ladattavissa Comodon verkkosivuilta. Verkkosivuilta ladatessa voi kätevästi valita käytössä olevan Unix-pohjaisen käyttöjärjestelmän, sekä minkä bittisen skannerin haluaa ladata. Comodon antivirusskanneri tukee kuitenkin vain Debian 6.0 versiota, joten skannerin asentamiseksi muutama toimenpide tarvitaan.

Comodon antivirusskanneri ei ole pakattu Debianin Stable julkaisulle, joten se pitää asentaa manuaalisesti. 64-bittistä ei saatu toimimaan 8.4 Jessie distribuutiossa, joten asennettiin i386-arkkitehtuurin versio skannerista. Ladataan skanneri ja asennetaan se manuaalisesti juureen. Manuaalisen asennuksen jälkeen ajetaan lopuksi jälkiasennus.

sudo apt-get install binutils

ar x cav-linux_1.1.268025-1_i386.deb

sudo tar xvf ~/data.tar.gz -C /

/opt/COMODO/post_install.sh

Asennetaan riippuvuudet, jotta voidaan päivittää Comodon tunnistetietokanta:

sudo apt-get install libqt4-sql libqt4-network libqtgui4

Comodon skannerin tunnistetietokanta saadaan päivitettyä ajamalla manuaalisesti:

/opt/COMODO/cavupdater

Tunnistetietokanta voidaan myös päivittää lataamalla koko tietokanta Comodon verkkosivuilta, jota päivitetään päivittäin. Ladataan tietokanta osoitteesta https://www.comodo.com/home/internet-security/updates/vdp/database.php ja korvataan vanha tietokanta polussa */opt/COMODO/scanners/bases.cav* ladatulla tietokannalla (bases.cav)

cp bases.cav /opt/COMODO/scanners/bases.cav

Reaaliskannauksen saa Comodossa otettua pois päältä graafisen käyttöliittymän kautta asettamalla ”Real-Time Scanning” disabled-tilaan Kuvion 20 mukaisesti. Comodo ajetaan järjestelmässä parametreilla *-v -s.*

Kuvio 20. Reaaliaikaskannauksen disabloiminen Comodossa

7.1.6 Dr.Web

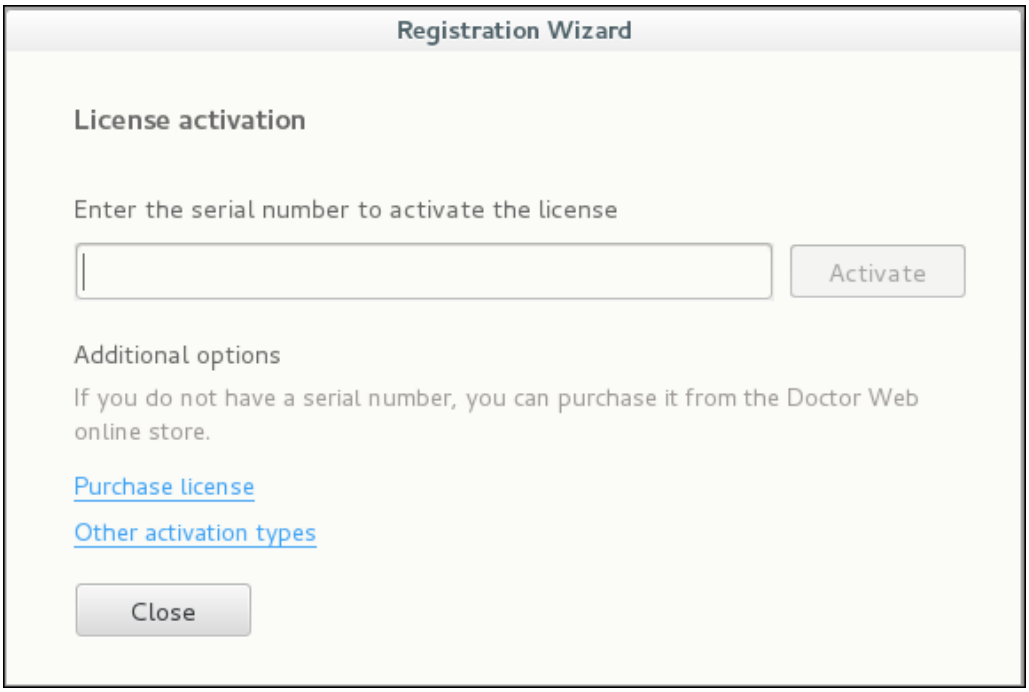
Dr.Web antivirusskanneri Linuxille on ladattavissa yhtiön verkkosivuilta. Ohjelma on maksullinen, mutta rekisteröitymällä saa käyttöönsä kolme kuukautta kestävän kokeiluversion, joka on selvästi muiden skannereiden kokeiluaikaa pidempi. Rekisteröitymisen jälkeen saadaan sähköpostiin lisenssiavain.

Ladataan Dr.Web anti-virus for Linux ja asennetaan ladattu asennustiedosto. Ohjelma asentuu kansioon */opt/drweb.com/.*

chmod +x drweb-11.0.0-av-linux-amd64.run

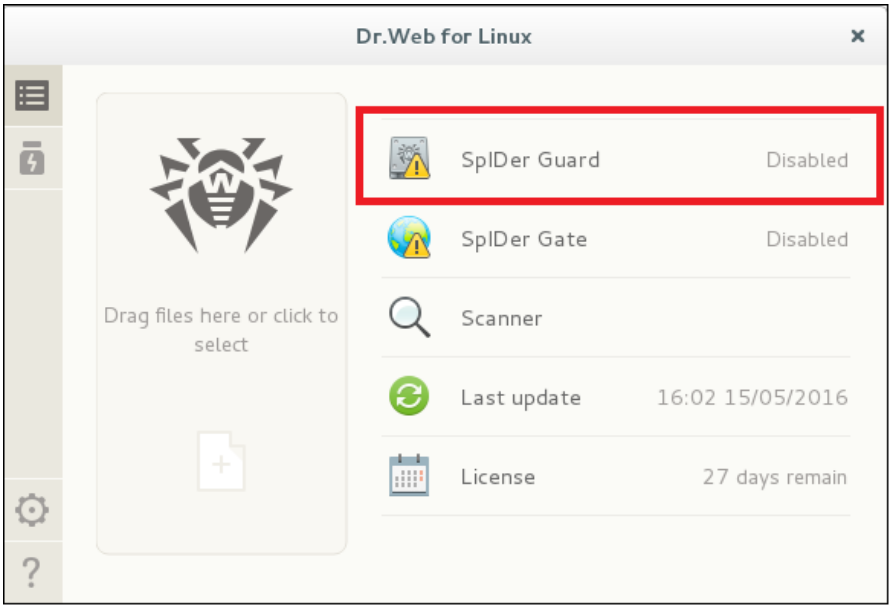
./ drweb-11.0.0-av-linux-amd64.run

Kun Dr.Web on asennettu, voidaan graafisen käyttöliittymän kautta käydä asettamassa sähköpostilla saatu lisenssi avain. Käynnistetään graafinen käyttöliittymä komennolla */opt/drweb.com/bin/drweb-gui.* Asetetaan lisenssi avain painamalla ”Licence information”, jonka jälkeen ”Get new license” ja asetetaan avain (Kuvio 21).



Kuvio 21. Dr.Web aktivointi

Spider Guard on Dr.Webin reaaliaikaskannaus, joten otetaan se pois käytöstä esimerkiksi ohjelman graafisen käyttöliittymän kautta. (Kuvio 22.)



Kuvio 22. Reaaliaikaskannauksen disablointi Dr.Webissä

Dr.Webin virustunnistetietokanta voidaan päivittää Internetistä komennolla:

drweb-ctl update

Muutama vuosi sitten Dr.Web tarjosi vielä virustunnistetietokantansa ladattavaksi kotisivuillaan, mutta nykyään näyttäisi siltä, että niitä ei ole enää saatavilla ainakaan julkisesti. Joten jos IRMAa ei haluta käyttää Internetissä, niin yksi vaihtoehto olisi asentaa Dr.Web toiselle Linux-koneelle ja päivittää se siellä. Tämän jälkeen kopioitaisiin päivitetyt virustunnisteet ja siirrettäisiin IRMA palvelimelle. Virustunnisteet sijaitsevat kansiossa */var/opt/drweb.com/bases/.* Dr.Web ajetaan järjestelmässä parametrilla *scan.*

7.1.7 EsetNod32

EsetNod32 Antivirus 4 Linuxille on mahdollista ladata Esetin verkkosivuilta. Valitaan käyttöjärjestelmä ja kieli, jonka jälkeen ladataan asennustiedosto. Asennetaan ladattu tiedosto. Asennuksen yhteydessä valitaan ”Activate Trial License”. Kokeiluversio kestää 30 päivää.

sudo chmod u+x eset_nod32av_64bit_en.linux

sudo apt-get install libgtk2.0-0 libc6-i386

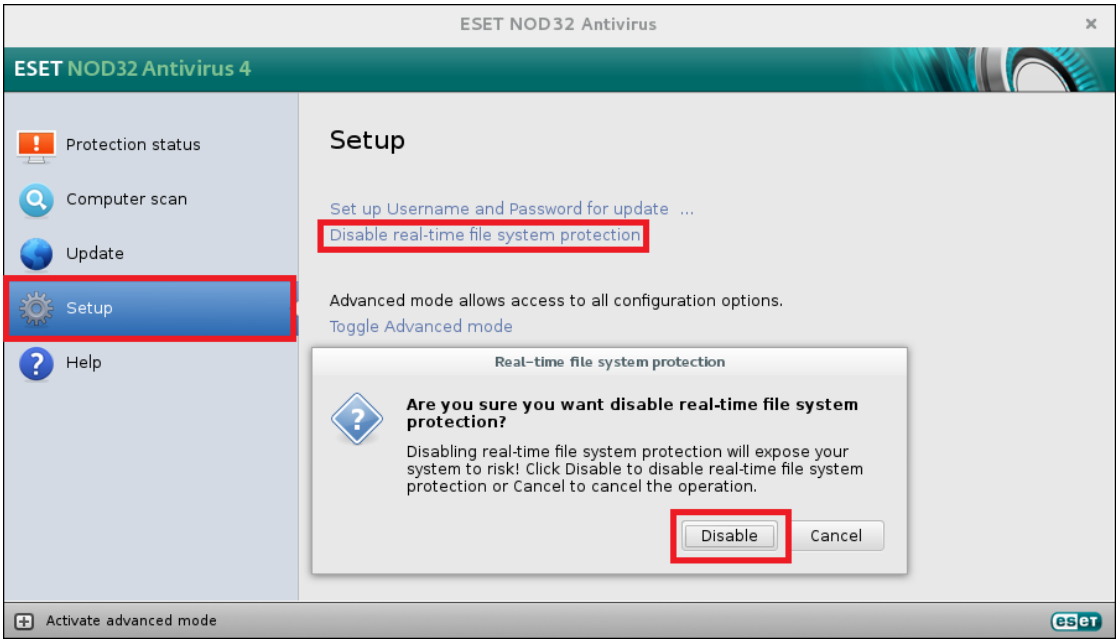
sudo ./eset_nod32av_64bit_en.linux

Disabloidaan Esetin daemoni, jotta se ei suojele käyttöjärjestelmää käynnistyksen yhteydessä:

sudo service esets stop

sudo mv /etc/init.d/esets /etc/init.d/esets.disable

EsetNod32 reaaliskannaus voidaan disabloida graafisen käyttöliittymän kautta Kuvion 23 mukaisesti.



Kuvio 23. Reaaliaikaskannauksen disablointi EsetNod32

Antivirusskannerin virustunnisteiden päivitys tapahtuu myös graafisen käyttöliittymän kautta kohdasta “Update”. EsetNod32 ei tarjoa mahdollisuutta ladata virustunnistepäivityksiä manuaalisesti verkosta. Jos halutaan päivittää ohjelma ilman yhteyttä Internetiin, asennetaan EsetNod32 toiselle Linux-koneelle, päivitetään se ja siirretään virustunnisteet IRMA palvelimelle. Virustunnisteet sijaitsevat kansiossa */var/opt/eset/esets/lib/.* EsetNod32 ajetaan järjestelmässä parametreilla *--clean-mode=NONE --no-log-all.*

7.1.8 eScan

EScanista on saatavilla Linux versio eScanin verkkosivuilta Debian ja Redhat pohjaisille distribuutioille. Ladataan sivuilta 64-bittinen asennuspaketti Debianille. Asennetaan ladattu paketti. eScan asentuu kansioon */var/MicroWorld/.*

gdebi escan-antivirus-wks-7.0-20.amd64.deb

Jos järjestelmällä on yhteys Internetiin, voidaan eScan päivittää ajamalla komento:

/opt/MicroWorld/usr/bin/updatenow

Jos yhteyttä Internetiin ei ole, niin helpoimmalla pääsee, kun asentaa eScanin toiselle Linux-koneelle. Tämän jälkeen kopioidaan eScanin virustunnisteet järjestelmästä toiseen. Virustunnisteest sijaitsevat kansiossa */opt/MicroWorld/var/avsupd/.* EScanin antivirusskannerissa ei ole reaaliaikaskannaus mahdollisuutta, joten sen disabloimisesta ei tarvitse huolehtia. EScan ajetaan järjestelmässä parametreilla *--log-only --recursion --pack --archives --heuristic --scan-ext --display-none --display-infected.*

7.1.9 F-Prot

F-protin antivirusskanneri Linuxille on ilmainen kotikäyttöön henkilökohtaisilla työasemilla käytettäessä. Ohjelmasta on tarjolla 32-bittinen versio F-protin verkkosivuilla. Ladataan TAR- pakattu ohjelma ja puretaan se polkuun /usr/local/. Kopioidaan myös konfiguraatiotiedosto uuteen polkuun päivittämisen onnistumisen takia.

sudo tar xvf fp-Linux.x86.32-ws.tar.gz -C /usr/local/

sudo cp /usr/local/f-prot/f-prot.conf.default /etc/f-prot.conf

Virustunnisteet voidaan päivittää kansiossa /usr/local/f-prot ajamalla komento:

sudo ./fpupdate

F-prot ei myöskään tarjoa sivuillaan virustunnisteiden manuaalista lataamista. Ilman Internetiä voidaan F-prot asentaa toiselle Linux-koneelle, päivittää virustunnisteet, jonka jälkeen kopioida uudet tunnisteet IRMA palvelimelle. Virustunnisteet sijaitsevat tiedostossa */usr/local/f-prot/antivir.def.* F-prot ajetaan järjestelmässä parametreilla *--report --verbose=0*

7.1.10 F-Secure

F-secure tarjoaa 30 päivän kokeiluversion heidän Linux antivirusskannerillensa. Skanneri voidaan ladata F-securen verkkosivuilta. Ladataan ”F-Secure Linux Security 11.00”, joka tulee pakatussa muodossa. Latauksen jälkeen puretaan pakattu tiedosto ja asennetaan Linux skannerista pelkkä komentoriviskanneri asettamalla parametri –command-line-only. Asennuksen yhteydessä kysytään myös lisenssiavainta, mutta painamalla Enter kyseisessä kohdassa, aktivoidaan 30 päivän kokeiluversio.

tar xvf fsls-11.00.79-rtm.tar.gz
cd fsls-11.00.79-rtm
./fsls-11.00.79-rtm --command-line-only

F-securen komentoriviskanneri ei tue reaaliaikaskannausta, joten sen disabloimisesta ei tarvitse välittää. F-securen Linux antivirusskanneri voidaan päivittää suoraan Internetistä ajamalla komento:

/opt/f-secure/fsav/bin/dbupdate

Antivirusskanneri voidaan päivittää myös manuaalisesti ilman Internet-yhteyttä lataamalla päivitetty virustunnistetiedosto osoitteesta http://download.f-secure.com/latest/fsdbupdate9.run.

Lataamisen jälkeen ajetaan *dpupdate* kansiossa */opt/f-secure/fsav/bin:*

/opt/f-secure/fsav/bin# ./dbupdate <POLKU TIEDOSTOON fsdbupdate9.run>

F-securen sivuilta voidaan myös seurata virustietokannan päivittymistä osoitteessa https://www.f-secure.com/dbtracker/Aquarius/index.html. F-secure ajetaan järjestelmässä parametreilla *--allfiles=yes --scanexecutables=yes --archive=yes --mime=yes --virus-action1=report --suspected-action1=report --virus-action2=none.*

7.1.11 McAfee

McAfeen ilmainen kokeiluversio 64-bittisestä Linux command-line skannerista on ladattavissa McAfeen verkkosivuilta. Ennen latausta pitää täyttää lomake ja sen täyttämisen jälkeen voidaan pakattu antivirusskanneri ladata. Puretaan ladattu TAR-pakkaus kansioon /usr/local/uvscan, mutta ennen sitä luodaan kyseinen kansio.

sudo mkdir /usr/local/uvscan

sudo tar xvf vscl-l64-606-e.tar.gz -C /usr/local/uvscan

sudo chmod +x /usr/local/uvscan/uvscan

Virustunnisteet sijaitsevat kansiossa /usr/local/uvscan ja ne voidaan päivittää vain manuaalisesti lataamalla uudet tunnisteet verkkosivuilta. Tunnisteet voidaan ladata pakatussa muodossa osoitteesta http://www.mcafee.com/apps/downloads/security-updates/security-updates.aspx. Lataamisen jälkeen puretaan yksinkertaisesti uudet tunnisteet kansioon /usr/local/uvscan/.

McAfeen Linux command-line skannerissa ei ole reaaliaikaskannauksen mahdollisuutta, joten sen disabloinnista ei tarvitse huolehtia. McAfee ajetaan järjestelmässä parametreilla */ANALYZE /MANALYZE /RECURSIVE /UNZIP /NOMEM.*

7.1.12 Sophos

Sophoksen Endpoint Protection Linuxille voidaan ladata Sophoksen verkkosivuilta. Ohjelma vaatii rekisteröitymisen ja kokeiluversio on voimassa 30 päivää. Sähköpostiin saadaan käyttäjänimi ja salasana. Ohjelman lataamisen jälkeen puretaan pakattu tiedosto ja suoritetaan asennus. Asennuksen yhteydessä määritetään parametreja ja tunnukset.

tar zxf sav-linux-9-i386.tgz

./sophos-av/install.sh /opt/sophos-av --acceptlicence --autostart=False --enableOnBoot=False --automatic --ignore-existing-installation

/opt/sophos-av/bin/savconfig set EnableOnStart false

/opt/sophos-av/bin/savconfig set PrimaryUpdateSourcePath "sophos:"

/opt/sophos-av/bin/savconfig set PrimaryUpdateUsername "< SÄHKÖPOSTILLA SAATU KÄYTTÄJÄNIMI>"

/opt/sophos-av/bin/savconfig set PrimaryUpdatePassword "<SÄHKÖPOSTILLA SAATU SALASANA>"

Sophoksen virustunnisteet voidaan päivittää ajamalla komento:

/opt/sophos-av/bin/savupdate

Manuaalista virustunnisteiden päivittämistä verkkosivuilta lataamalla ei enää tueta. Jos skannerin virustunnisteet halutaan päivittää ilman Internet-yhteyttä, pitää Sophos asentaa toiselle Linux-koneelle, josta päivitetyt virustunnisteet saadaan kopioitua IRMA palvelimelle. Sophoksen virustunnistetietokannat sijaitsevat kansiossa */opt/sophos-av/lib/sav.* Sophos ajetaan järjestelmässä parametreilla *-archive -cab -loopback -tnef -mime -oe -pua -ss -nc -nb.*

Sophoksen reaaliaikaskannaus saadaan disabloitua ajamalla:

/opt/sophos-av/bin/savdctl disable

7.1.13 Zoner

Zonerin ilmainen antivirusskanneri Linuxille on ladattavissa Zonerin verkkosivuilta. Verkkosivuilla rekisteröidytään myös samalla, jonka jälkeen saa Zonerin aktivointikoodin annettuun sähköpostiin. Rekisteröityminen ei ole vaatimus skannerin ilmaiselle käytölle, mutta sen avulla saadaan ladattua moottori- ja virustunnistepäivitykset. Ladataan 64-bittinen skanneri ja asennetaan se oletuspolkuun.

gdebi zav-1.3.0-debian-amd64.deb

Aktivoidaan Zoner asettamalla sähköpostilla saatu lisenssiavain konfiguraatiotiedostoon, joka löytyy polusta /etc/zav/zavd.conf. Haetaan pitkästä konfiguraatiotiedosta kohta ”UPDATE_KEY” ja asetetaan sen arvoksi saatu lisenssiavain.

Ennen Zonerin käyttöä daemoni pitää käynnistää manuaalisesti ajamalla komento:

/etc/init.d/zavd start

Zonerin virustunnisteet taas voidaan päivittää Internetistä ajamalla komento:

/etc/init.d/zavd update

Jos Zoneria ei haluta päivittää suoraan Internetin kautta, voidaan päivitetyt virustunnisteet kopioida toisesta Zonerin Linux-asennuksesta. Tämä taas vaatii Zonerin asentamisen toiselle Linux-koneelle, johon tunnisteet päivitetään Internetin välityksellä. Zonerin virustunnistetietokannat sijaitsevat kansiossa */opt/zav/lib*. Zonerin Linux skannerissa ei ole reaaliaikaskannausta, joten sen disabloimisesta ei tarvitse huolehtia. Zoner ajetaan järjestelmässä parametreilla *--scan-full --scan-heuristics --scan-emulation --scan-archives --scan-packers --scan-gdl --scan-deep --show=infected –quiet*

7.2 Muut analysointimenetelmät

7.2.1 Static Analyzer

Static Analyzer on PE-tiedostojen (Portable Executable) analysointiin staattinen analysaattori. Static Analyzer on adoptoitu Cuckoo Sandbox -järjestelmästä, joka on pääasiassa dynaaminen haittaohjelmien analysointijärjestelmä, jossa haittaohjelmat suoritetaan eristetyssä käyttöjärjestelmässä. Static Analyzer on helppo asentaa ja ei vaadi erillistä konfiguroimista. Static Analyzer voidaan asentaa komennolla:

pip install pefile

7.2.2 VirusTotal API

VirusTotalin ohjelmointirajapintaa käytetään vain tarkastussumman vertailuun eli mitään tiedostoja ei siirretä ulos verkosta. Vaikka tehtävänanto oli järjestelmän toteuttaminen suljettuun ympäristöön, niin otettiin VirusTotalin API silti pikaiseen tarkasteluun. Rajapinnan kautta siis vain verrataan tiettyä SHA-256 tarkastussummaa VirusTotalin tietokantaan. Jos tarkastussumma löytyy tietokannasta, niin nähdään VirusTotalin tulokset kyseiselle tiedostolle. Ohjelmointi rajapinnan käyttöön ottaminen on helppoa. Asennetaan ensin APIn Python-riippuvuudet komennolla, jonka jälkeen konfiguroidaan moduuli.

pip install virustotal-api

VirusTotalin verkkosivuille pitää rekisteröityä, jotta saadaan henkilökohtainen API-avain, jonka avulla päästään hakemaan VirusTotalin tietokannasta tuloksia. Muokataan tiedostoa *config.ini* polussa */opt/irma/irma-probe/modules/external/virustotal/.* Tiedostossa on kaksi kohtaa: Private ja Apikey. Asetetaan Private-kentän arvoksi False ja Apikey-kenttään saamamme API-avain. Private API-avain on maksullinen ja luonnollisesti sen avulla pyyntöjä VirusTotalin tietokantaan saa tehdä enemmän, kuin julkisella API-avaimella. Julkisen API-avaimen rajoitukset tietokannasta hakuun ovat 4 pyyntöä/minuutti, 5760 pyyntöä/päivä ja 178560 pyyntöä/kuukausi.

7.2.3 YARA

YARA on monipuolinen työkalu, jonka avulla voidaan tunnistaa ja luokitella haittaohjelmanäytteitä. YARA mahdollistaa esimerkiksi haittaohjelmaperheiden tunnistamista perustuen binäärikuvioihin tai tiettyyn tekstiin. YARAN tunnistuksen vaihtoehdot perustuvat Boolean algebrahan, joka sisältää muun muassa ehdot JA, TAI, ja EI. (YARA 2016)

Kuviossa 24 on esitetty esimerkki YARA-säännöstä. Säännölle annetaan kuvaileva nimi, jonka jälkeen voidaan antaa tälle säännölle metadataa, kuten haittaohjelman kuvaus ja uhkataso, sekä onko haittaohjelma aktiivisessa levityksessä (in the wild). Sääntöön luonnollisesti asetetaan myös tunnistussäännöt, joita tarkastettavasta tiedostosta etsitään. YARAssa on laaja skaala, mitä tunnistussääntöjä voidaan asettaa, mutta yleisimmät ovat tekstipohjaiset merkkijonot, tai merkkijonot heksadesimaali muodossa. Lopuksi voidaan asettaa ehdot, jolloin haittaohjelma havaitaan sääntöjen perusteella. Esimerkissä on käytetty TAI-ehtoja (OR) eli kunhan joku kolmesta tunnistussäännöstä löytyy, haittaohjelma tunnistetaan. Jos TAI-ehdot korvattaisiin esimerkissä JA-ehdoilla (AND), se tarkoittaisi, että kaikkien tunnistussääntöjen pitää löytyä tiedostosta, jotta haittaohjelma tunnistetaan.

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

Kuvio 24. Esimerkki YARA-säännöstä

Asennetaan aluksi YARAn riippuvuudet, jonka jälkeen asennetaan itse YARAn Python-moduulit.

```
sudo apt-get install libtool automake bison

git clone https://github.com/plusvic/yara.git
autoreconf -i –force
./configure
make
sudo make install
python setup.py build
sudo python setup.py install
sudo ldconfig
```

Kun YARA on asennettu, voidaan alkaa luoda YARA-sääntöjä. Järjestelmä hakee säännöt tiedostosta */opt/irma/yara_rules/ yara_rules.yar*, joten tähän tiedostoon luodaan halutut säännöt. Polkua voidaan halutessa vaihtaa muuttamalla polku tiedostoon modules/metadata/yara/config.ini. Oletuksena sektio *rule_path* on arvolla */opt/irma/yara_rules/yara_rules.yar.* Luodaan seuraavaksi todella yksinkertainen YARA-sääntö luomalla ja muokkaamalla tiedostoa *yara_rules.yar*. Kirjoitetaan sääntö EICAR-testijonolle *yara_rules* -tiedostoon.

```
nano /opt/irma/yara_rules/yara_rules.yar

rule eicar_testaus
{
   meta:
      description = "Testisaanto Eicarille"
      threat_level = 0
   strings:
      $E = " X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-
TEST-FILE!$H+H* "
   condition:
      $E
}
```

7.2.4 Unarchiver

Antivirusskannerit osaavat itse purkaa yleisimmät pakatut tiedostot, mutta esimerkiksi Static Analyzeria ja YARAa varten pitää pakattu tiedosto purkaa ennen analysointia. Tähän käyttöön soveltuu Pyunpack, joka osaa purkaa suuren määrän eri pakkausmuotoja. Pyunpack on komentorivin kautta kutsuttava ja sillä voidaan purkaa vain salasana suojaamattomia pakkauksia. Jotta Pyunpack saadaan toimimaan järjestelmässä, ei konfigurointia vaadita, vaan asennetaan se yksinkertaisesti komennolla

pip install pyunpack

8 Toimintoja

8.1 Antivirusskannereiden päivittäminen

Antivirusskannereiden näytetietokantojen eli tunnisteiden päivittäminen on tärkeää, kun halutaan pitää tietoturvataso korkealla. Järjestelmän kohdeympäristö on suljettu ympäristö ja koska suljetussa ympäristössä eli ole jatkuvaa yhteyttä – tai ei välttämättä edes hetkellistä yhteyttä Internetiin, pitää miettiä vaihtoehtoja virustunnisteiden päivittämiseen. Yksi vaihtoehto on käyttää järjestelmä hetkellisesti Internetissä ja ladata jokaisen antivirusohjelman virustunnisteet ohjelman omalla päivittäjällä. Tätä voidaan kutsua manuaaliseksi päivitykseksi, koska yleensä antivirusohjelmat päivittävät tunnisteensa noin muutaman tunnin välein automaattisesti.

Toinen vaihtoehto on ladata antivirusskannereiden valmistajien sivuilta päivitetyt virustunnisteet. Kaikki valmistajat eivät kuitenkaan tarjoa virustunnisteidensa lataamista erikseen, vaan ne on päivitettävä ohjelman omalla päivittäjällä. Useimmat yhtiöt kuitenkin tarjoavat verkkosivuillansa uusimman virustunnistekantansa. Kun verkkosivulta ladataan tunnistekanta, se on lähes aina koko tunnistekanta eli erikseen ei voi ladata päivityksiä. Kun koko tunnistekanta ladataan kerralla, se on tiedostokooltaan suuri tiedosto, jonka lataaminen voi kestää.

8.2 Analysaattorin disablointi

IRMAssa ei ainakaan vielä ole suoraa toiminnallisuutta, millä voisi disabloida halutun skannerin. Disabloinnilla tarkoitetaan tässä tapauksessa sitä, että käyttäjä ei näkisi web-sivulla pois käytöstä otettua skanneria, eikä näin ollen pystyisi skannaamaan kyseisellä skannerilla tiedostoja. Skanneri voidaan ottaa pois käytöstä esimerkiksi siirtämällä sen liitännäiset sisältävän liitännäis-kansio pois antivirus-kansiosta. Jos skanneri halutaan tuoda takaisin toimintaan, voidaan se toteuttaa yksinkertaisesti siirtämällä skannerin liitännäis-kansio takaisin antivirus-kansioon. Tehdään ensin kansio DISABLED modules-kansion alle, johon voidaan siirtää skannerien liitännäis-kansiot.

mkdir DISABLED

Otetaan pois käytöstä esimerkiksi F-secure, joten siirretään sen liitännäiskansio DISABLED kansioon. (Kuvio 25)

mv antivirus/fsecure DISABLED/

```
root@brain:/opt/irma/irma-probe/current/modules# ls
antivirus  custom  database  DISABLED  external  __init__.py  __init__.pyc  metadata  tools
root@brain:/opt/irma/irma-probe/current/modules# ls antivirus/
avast  base.py      clamav  emsisoft  fprot     __init__.py   interface.pyc  sophos        zoner
avg    base.pyc     comodo  escan     fsecure   __init__.pyc  kaspersky      symantec
avira  bitdefender  drweb   eset      gdata     interface.py  mcafee         virusblokada
root@brain:/opt/irma/irma-probe/current/modules# mv antivirus/fsecure/ DISABLED/
root@brain:/opt/irma/irma-probe/current/modules# ls DISABLED/
fsecure
```

Kuvio 25. Skannerin disablointi

Liitännäis-kansion siirtämisen jälkeen käynnistetään celery workerit uudelleen, pysäytetään probe_app ja lopuksi ajetaan probe_app celery worker uudelleen:

supervisorctl restart all

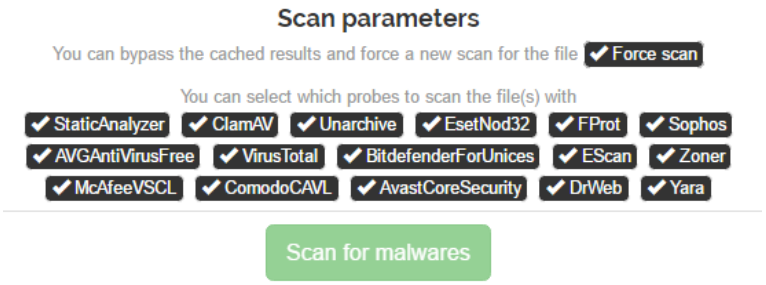
supervisorctl stop probe_app

celery worker --app=probe.tasks:probe_app --workdir=/opt/irma/irma-probe

Kuviosta 26 nähdään, että F-Securen jono on hävinnyt ja Kuviosta 27 voidaan myös todeta, että web-sivulta käyttäjä ei voi enää valita F-securen skanneria. Kun skanneri halutaan ottaa taas käyttöön, siirretään liitännäinen DISABLED kansiosta moduulit-kansioon ja käynnistetään result_app ja scan_app uudelleen, sekä ajetaan probe_app manuaalisesti.

```
 -------------- celery@probe_app.brain v3.1.23 (Cipater)
---- **** -----
--- * ***  * -- Linux-3.16.0-4-amd64-x86_64-with-debian-8.4
-- * - **** ---
- ** ---------- [config]
- ** ---------- .> app:         probe.tasks:0x7fc766c766d0
- ** ---------- .> transport:   amqp://probe:**@brain.irma:5672/mqprobe
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 2 (prefork)
-- ******* ----
--- ***** ----- [queues]
 -------------- .> AVGAntiVirusFree exchange=celery(direct) key=AVGAntiVirusFree
                .> AvastCoreSecurity exchange=celery(direct) key=AvastCoreSecurity
                .> BitdefenderForUnices exchange=celery(direct) key=BitdefenderForUnices
                .> ClamAV           exchange=celery(direct) key=ClamAV
                .> ComodoCAVL       exchange=celery(direct) key=ComodoCAVL
                .> DrWeb            exchange=celery(direct) key=DrWeb
                .> EScan            exchange=celery(direct) key=EScan
                .> EsetNod32        exchange=celery(direct) key=EsetNod32
                .> FProt            exchange=celery(direct) key=FProt
                .> McAfeeVSCL       exchange=celery(direct) key=McAfeeVSCL
                .> Sophos           exchange=celery(direct) key=Sophos
                .> StaticAnalyzer   exchange=celery(direct) key=StaticAnalyzer
                .> Unarchive        exchange=celery(direct) key=Unarchive
                .> VirusTotal       exchange=celery(direct) key=VirusTotal
                .> Yara             exchange=celery(direct) key=Yara
                .> Zoner            exchange=celery(direct) key=Zoner
```

Kuvio 26. Jonot F-securen disabloinnin jälkeen



Kuvio 27. Valittavat skannerit F-securen disabloinnin jälkeen

8.3 Tulosten poisto tietokannasta

Tulokset ovat tallennettu Frontendin PostgreSQL-tietokantaan SHA-256 tarkastussummalla. IRMA ei tarjoa suoraa työkalua tuloksien poistoon tietokannasta. Tuloksia voidaan kuitenkin poistaa yksitellen skriptin avulla tai kaikki tulokset voidaan poistaa kerralla pudottamalla koko tietokanta ja luomalla se uudestaan. Luodaan tiedosto *delete.py* polkuun */opt/irma/irma-frontend/* ja tehdään sinne skripti, joka poistaa tuloksen tietokannan taulusta SHA-256 tarkastussumman perusteella.

```
touch /opt/irma/irma-frontend/delete.py
nano /opt/irma/irma-frontend/delete.py

import sys
from frontend.helpers.sessions import *
from frontend.models.sqlobjects import *

if len(sys.argv) != 2 :
  print "Usage: %s <file SHA-256 to remove>"%(sys.argv[0])
  sys.exit(0)

sha256_value = sys.argv[1]
with session_transaction() as db:
    file = db.query(File).filter_by(sha256=sha256_value).one()
    for fw in file.files_web:
        db.delete(fw)
    db.delete(file)
```

Skriptiä käytetään komennolla:

```
python delete.py <Poistettavan tiedoston SHA-256 tarkastussumma>
```

Kaikkien tuloksien poisto tietokannasta onnistuu poistamalla tietokanta ja luomalla se uudestaan. Ennen poistamista pitää kuitenkin celery workerit pysäyttää, koska ne käyttävät tietokantaa. Laitetaan celery workerit tietokannan uudelleenluonnin jälkeen takaisin päälle.

supervisotctl stop all

frontend_api: stopped
frontend_app: stopped
result_app: stopped
scan_app: stopped

su – postgres
psql
DROP DATABASE "irma-frontend";
CREATE DATABASE "irma-frontend";

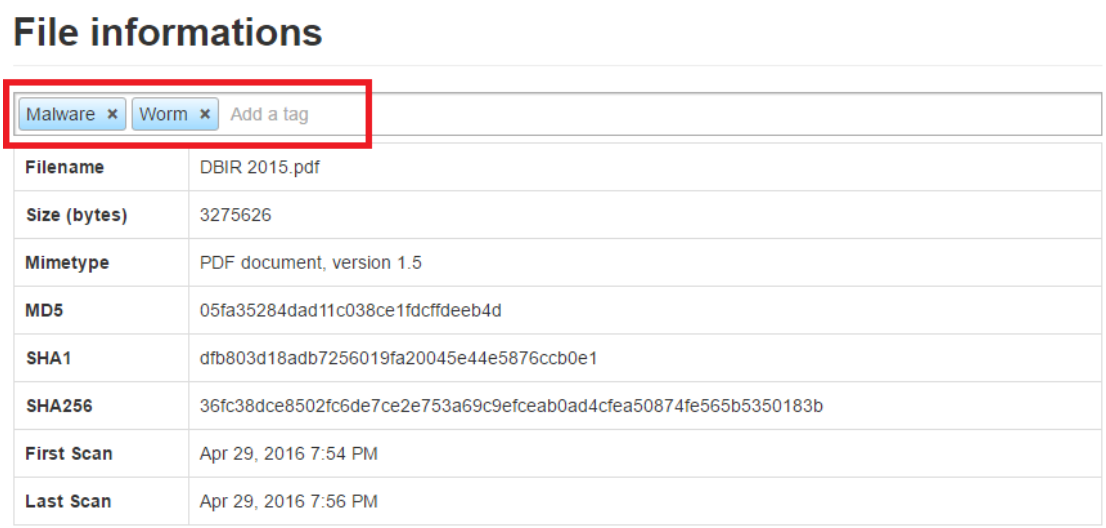
supervisotctl start all

result_app: started
scan_app: started
frontend_app: started
frontend_api: started
probe_app: started

8.4 Tagin lisääminen tiedostoon

Tarkastettuun tiedostoon on mahdollista liittää tunniste (tag), jonka avulla tiedostoja voidaan luokitella ja myöhemmin hakea tietokannasta tiedostojen tuloksia tietyllä tunnisteella. Tagi pitää ensin lisätä järjestelmään, jotta sillä voidaan merkata tiedostoja. Tageja voidaan lisätä esimerkiksi curlin avulla postaamalla tekstiavain. Kuviossa 28 nähdään, kuinka tarkastettuun tiedostoon voidaan lisätä tagi web-käyttöliittymässä.

curl -H "Content-Type: application/json; charset=UTF-8" -X POST -d '{"text":"<TAGIN NIMI>"}' http://172.16.1.30/api/v1.1/tags

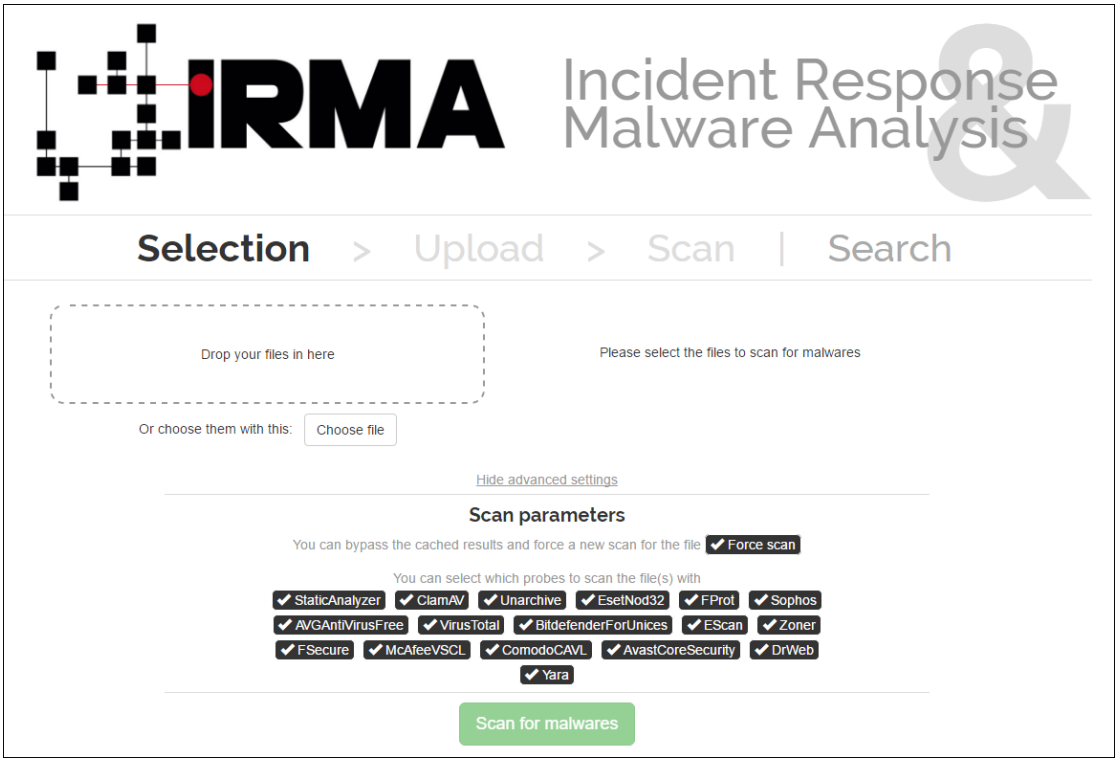


Kuvio 28. Tagin lisääminen tarkastettuun tiedostoon

9 Testaus ja tulosten tarkastelu

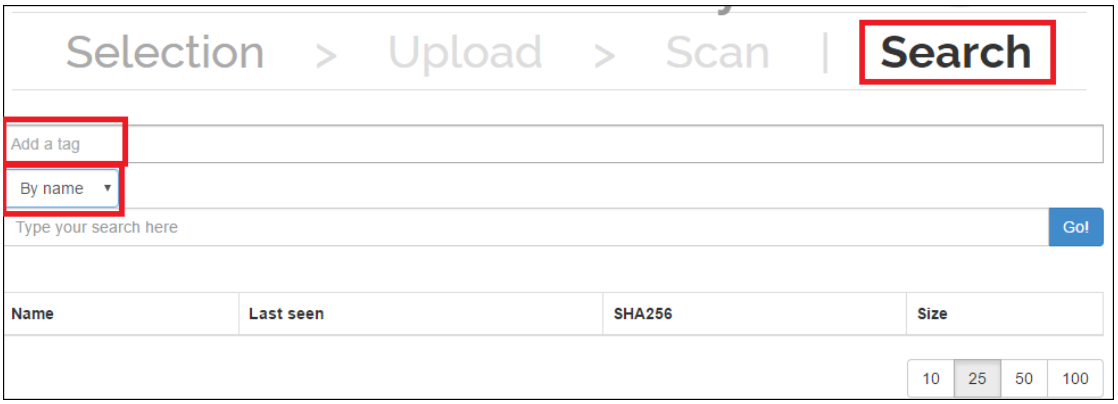
9.1 Web-käyttöliittymä

Käyttäjä pystyy web-käyttöliittymän avulla lähettämään useita tiedostoja kerralla järjestelmän tarkastettavaksi, joko vetämällä tiedostot käyttöliittymään tai valitsemalla ne ”Choose File” -painikkeen kautta omalta koneeltasi. Web-käyttöliittymässä käyttäjä näkee kaikki saatavilla olevat analysaattorit ja voi valita niistä käytettävät analysaattorit tiedostojen tarkastuksessa. Käyttöliittymässä on myös optio pakotettu tarkastaminen (Force scan). Jos pakotettu tarkastaminen on käytössä, niin järjestelmä tarkastaa tiedostot riippumatta siitä, onko niitä tarkastettu aikaisemmin järjestelmässä. Jos taas pakotettu tarkastaminen on otettu pois käytöstä ja tarkastettavien tiedostojen aikaisemmat tulokset löytyvät jo tietokannasta, niin tällöin aikaisempien analysointien tulokset palautetaan suoraan tietokannasta käyttäjälle. Kuviossa 29 on esitetty web-käyttöliittymän etusivu.



Kuvio 29. Web-käyttöliittymän etusivu

Web-käyttöliittymässä on mahdollista myös hakea tuloksia tietokannasta ”Search”-sivulla. Tarkastetun tiedoston tuloksia voidaan hakea nimen, tagin tai tarkastussumman perusteella (MD5, SHA1, SHA-256). Jos tulokset löydetään tietokannasta, ne tuodaan käyttäjän nähtäville ja niistä painamalla päästään itse tuloksien tarkastelemiseen. Kuviossa 30 nähdään web-käyttöliittymän ”Search”-sivu.



Kuvio 30. Web-käyttöliittymän tuloksien hakusivu

9.2 Testaus

Toteutetun järjestelmän toimivuuden testaamiseen käytettiin neljää eri tiedostoa. Käytetyt tiedostot koostuivat EICAR testitiedostosta, sekä kolmesta haittakoodia sisältävästä tiedostosta Windows-käyttöjärjestelmälle. Haittatiedostot Windows-käyttöjärjestelmälle sisälsivät kaksi haitallista EXE-tiedostoa, sekä haitallisen PDF-tiedoston. EXE ja PDF-tiedostot olivat pakattu ZIP-formaattiin.

Aluksi testattiin nopeasti järjestelmän ja antivirusskannereiden toiminta ”EICAR”-kappaleessa. Toiminnan varmistamisen jälkeen kappaleessa ”Haitalliset tiedostot” käydään web-käyttöliittymän sisältö tarkemmin läpi, sekä tarkastetaan kolme haittakoodia sisältävää tiedostoa.

9.2.1 EICAR

Antivirusskannereiden toimivuus järjestelmässä testattiin EICAR testitiedostolla. EICAR tiedosto ei siis ole haitallinen vaan tarkoitettu antivirusskannereiden toimivuuden tarkastamiseen. Kaikkien antivirusskannereiden tunnistetietokannasta löytyy näyte EICARia varten. EICAR on vain 68 merkkiä pitkä ja kooltaan tasan 68 tavua. Se koostuu merkkijonosta:

*X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H**

Web-käyttöliittymän sisältö on tarkemmin esitelty seuraavassa kappaleessa. Tässä kappaleessa todetaan vain järjestelmän toimivuus pikaisesti. Web-käyttöliittymään siis lähetettiin EICAR testitiedosto, jonka järjestelmä tarkasti. Kaikki antivirusskannerit tunnistivat tiedoston, joten koko järjestelmän, sekä antivirusskannereiden toimivuus saatiin todettua (Kuvio 31).

Antivirus

Name	Result	Version	Duration (in secs)
Clam AntiVirus Scanner	Eicar-Test-Signature	0.99	0.04
ESET NOD32 Antivirus Business Edition for Linux Desktop	Eicar test file	4.0.81	1.94
F-PROT Antivirus	EICAR_Test_File (exact)	6.7.10.6267	0.69
Sophos Anti-Virus	EICAR-AV-Test	5.21.0	6.73
AVG AntiVirus Free	EICAR_Test	13.0.3114	0.13
Bitdefender Antivirus Scanner for Unices	EICAR-Test-File (not a virus)	7.141118	4.33
eScan Antivirus for Linux Desktop	EICAR-Test-File (not a virus) (DB)	7.0-20	4.47
Zoner Antivirus for Linux Desktop	EICAR.Test.File-NoVirus	1.3.0	0.03
FSecure Antivirus for Linux Desktop	EICAR_Test_File [FSE]	11.00	4.55
McAfee VirusScan Command Line scanner	EICAR test file	6.0.6.653	2.09
Comodo Antivirus for Linux	Malware	1.1.268025.1	1.6
Avast Core Security	EICAR Test-NOT virus!!!	2.1.1	0.17
DrWeb Antivirus for Linux Desktop	EICAR Test File (NOT a Virus!)	10.1.0.0.1503311845	12.02

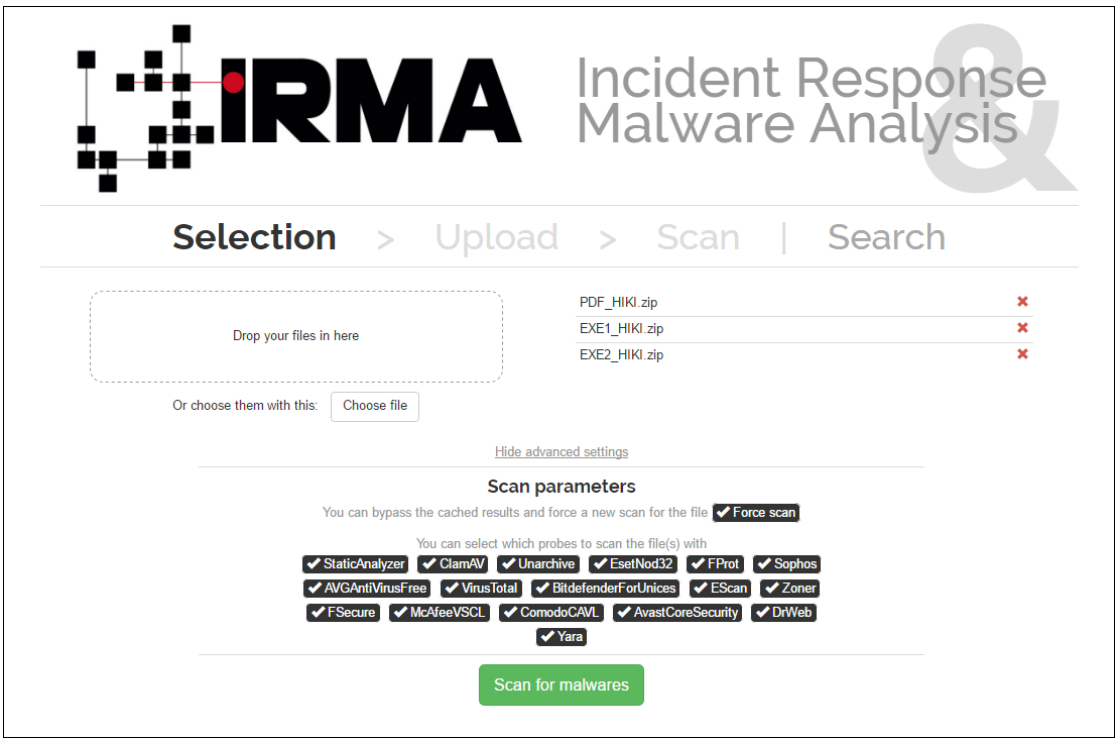
Kuvio 31. Tulokset EICAR

EICAR-tiedostolla saatiin todennettua myös YARAn toimivuus. YARAan aikaisemmin luotu sääntö EICAR-tiedostoa varten toimii, kuten Kuviosta 32 voidaan todeta.

Kuvio 32. YARAn toimivuuden todennus

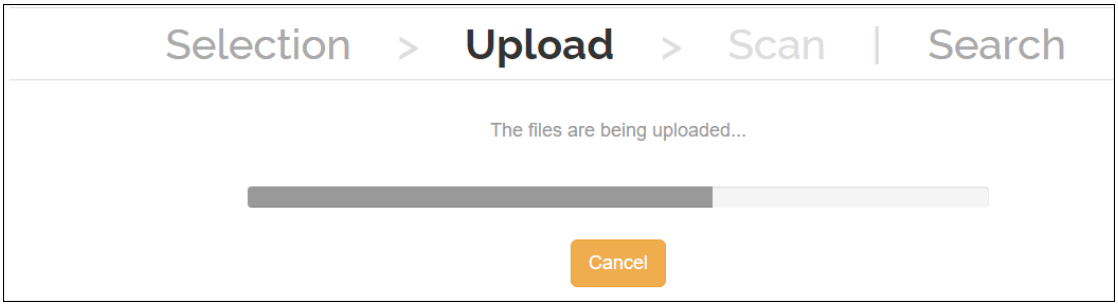
9.2.2 Haitalliset tiedostot

Käyttöliittymään lähetetään tarkastettavaksi ZIP-formaattiin pakatut haittakoodia sisältävät tiedostot. Halutaan tarkastaa tiedostot kaikilla analysaattoreilla, joten valitaan kaikki saatavilla olevat analysaattorit web-käyttöliittymästä, joilla tiedosto tarkastetaan. Kuviossa 33 käyttöliittymään on viety tarkastettavat tiedostot ja valittu analysaattorit. Lopuksi aloitetaan tarkastaminen painamalla painiketta ”Scan for malwares”.



Kuvio 33. Tarkastettavat tiedostot web-käyttöliittymässä

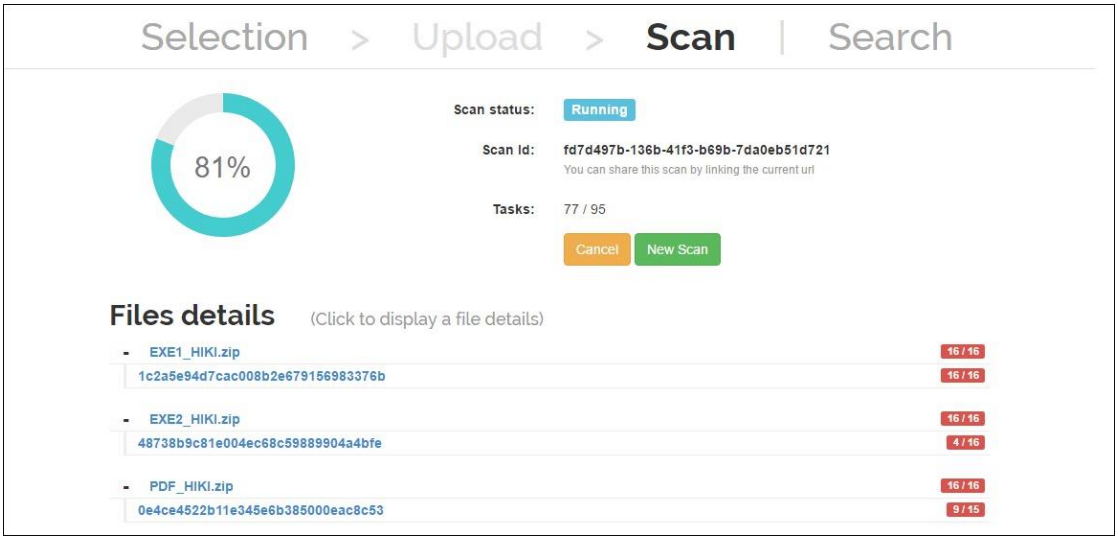
Tiedostojen lähettämiseen kuluu hetki aikaa riippuen tiedostojen koosta. Tämän aikana käyttäjälle näytetään mittaria, josta voidaan seurata tiedostojen lähetyksen etenemistä (Kuvio 34).



Kuvio 34. Mittari tiedoston lähetyksen valmistumisesta

Kun lähetys on valmis, käyttäjälle tuodaan esiin ”Scan”-sivu, josta tiedostojen tarkastamista voidaan seurata reaaliaikaisesti. Kuviosta 35 on nähtävissä käyttäjälle näytettävä tarkastuksen tila, skannaus ID, skannaustehtävien yhteismäärä, sekä tarkastuksen eteneminen prosenteissa. Tarkastaminen voidaan keskeyttää (Cancel-painike) tai voidaan siirtyä suorittamaan uutta tarkastusta.

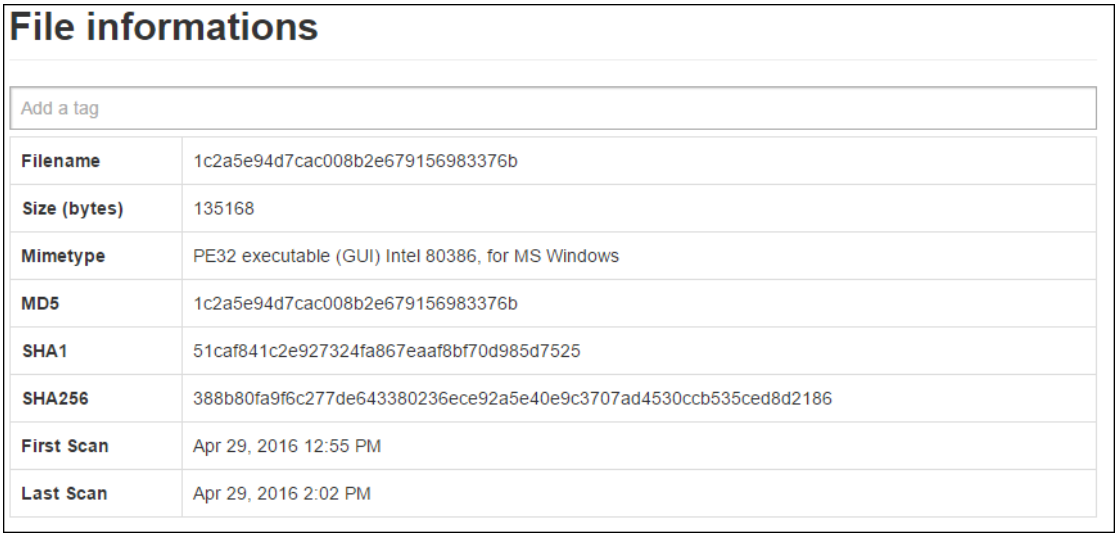
Tarkastettavien tiedostojen tiedoissa (file details) nähdään, että Pyunpack on avannut pakatun ZIP-tiedoston ja sen alla näkyy sen sisältämä tiedosto. Antivirusskannerit tarkastavat molemmat pakatun tiedoston, että sen sisällön, koska Pyunpack on purkanut pakatun tiedoston. Tiedoston kohdalla oikealla nähdään myös tiedostokohtaiset tehtävät eli kuinka monta analysaattoria on jo suorittanut tarkastamisen kyseiselle tiedostolle. Jos yksikin antivirusskanneri tunnistaa tiedoston haitalliseksi, sektio muuttuu vihreästä punaiseksi.



Kuvio 35. Käynnissä oleva tiedoston tarkastaminen

Kun tiedostot ovat tarkastettu, voidaan tiedoston nimestä painaa, jonka jälkeen avautuu itse tulokset. Liitteessä 6 on esitetty yleiskuva koko käyttäjälle aukeavasta sivusta, jossa tulokset näkyvät. Tulossivu koostuu skannatun tiedoston perustiedoista, antivirusskannereiden tuloksista, Metadata-sektiosta, sekä External-sektiosta.

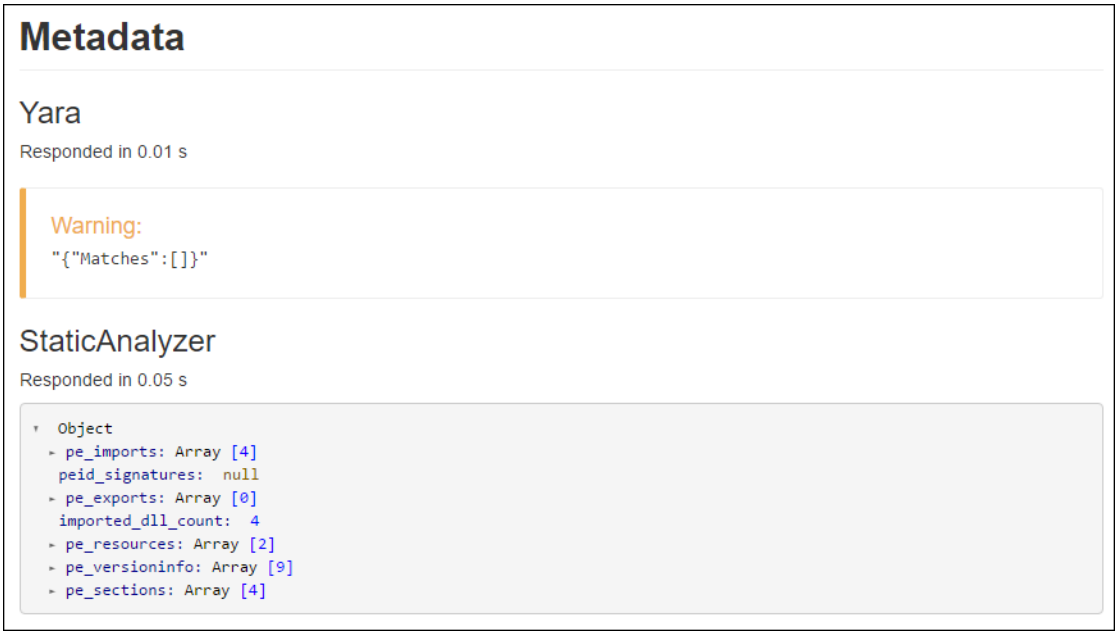
Perustiedot sisältävät tiedostonimen, tiedoston koon tavuina, tiedostosta lasketut tarkastussummat (MD5, SHA1 ja SHA-256), sekä milloin tiedosto on ensimmäisen kerran ja edellisen kerran tarkastettu järjestelmässä. Kuviosta 36 nähdään yhden EXE-tiedoston perustiedot tulossivulla.



Kuvio 36. EXE-tiedoston perustiedot

Toisena sektiona tulossivulla on antivirusskannereiden tulokset. Jos skanneri on tunnistanut tiedoston haitalliseksi, rivi on punainen. Jos taas skanneri pitää tiedostoa puhtaana, rivi on vihreä. Rivi voi myös olla keltainen, jos antivirusskanneri ei toimi oikein. Antivirusskannereiden tulokset sisältävät sarakkeet Name, Result, Version ja Duration. Nimi on luonnollisesti antivirusskannerin nimi, Result kertoo haittaohjelman nimen, versio tunnistetietokannan version ja Duration kertoo sekunneissa, kuinka kauan skannerilla meni aikaa tiedoston tarkastamiseen. Antivirusskannerien tulokset kolmelle tarkastetulle tiedostolle löytyvät Liitteistä 7, 8 ja 9. Kuten tuloksista näkyy, kaikki antivirusskannerit eivät välttämättä tunnistaneet haittaohjelmaa. Tarkastusaika kuitenkin on mukavaa katsottavaa, kun se elää 1-10 sekunnin välissä.

Kolmantena osiona tuloksissa on Metadata sektio. Tähän sisältyy YARA ja Static Analyzer. Kuviossa 37 on nähtävissä Metadata sektio, kun tarkastuskohteena on EXE-tiedosto. YARAlle ei ole luotu tässä työssä muita sääntöjä, kuin EICAR testitiedostolle, joten se palauttaa EXE-tiedostoa tarkastaessa varoituksen. Static Analyzer palauttaa tulokset vain, jos kyseessä on PE-tiedosto (Portable Executable).



Kuvio 37. Metadata sektio tulossivulla

Tulossivun alimpana osiona on External sektio. Tähän kuuluu tässä työssä vain Virus-Total APIn tuottamat tulokset. EXE-tiedoston SHA-256 tarkastussummaa verrataan VirusTotalin tietokannasta löytyvään ja linkin kautta voidaan mennä tarkastelemaan itse skannereiden tuloksia VirusTotalin sivuilta. Moduuli kertoo vain sen, että kuinka moni antivirusskanneri VirusTotalissa tiedoston on havainnut eli tässä tapauksessa 20/43. Kuviossa 38 nähdään VirusTotal APIn tuottama tulos.

Kuvio 38. External sektio tulossivulla

10 Yhteenveto

10.1 Pohdinta

Opinnäytetyön tavoitteena oli luoda VirusTotalin kaltainen avoimen lähdekoodin tarkastusjärjestelmä, jossa käyttäjän lähettämä tiedosto tarkastetaan keskitetysti monella eri valmistajan antivirusskannerilla. Tavoite saatiin toteutettua ja järjestelmään saatiin implementoitua 13 eri valmistajan antivirusskanneria. Lisäksi toteutettuun järjestelmään implementoitiin neljä muuta tiedoston analysointimenetelmää. Opinnäytetyön lopputuloksena tehty haittaohjelmien skannausjärjestelmä tulee varmasti tukemaan JYVSECTECin toimintaa sen kyberharjoituksissa ainakin haittaohjelmien osalta.

Opinnäytetyön aihe oli kokonaisuudessaan itselle sopivan haastava. Ongelmia tuli runsaasti vastaan, mutta niistä tuli selvittyä opiskelemalla käytettävien komponenttien dokumentaatiota, sekä selailemalla eri keskustelupalstoja Internetissä. Aikaisempaa kokemusta aiheesta ei ollut lukuun ottamatta muutamia koulussa opiskeltuja asioita liittyen tietoturvaan. Kuitenkin itse Linux käyttöjärjestelmästä kokemusta oli ennestään jonkin verran. Opinnäytetyön aikana tuli opittua todella paljon järjestelmässä käytetyistä komponenteista ja niiden toiminnasta. Myös antivirusskannereiden toiminta ja konfigurointi tuli opinnäytetyön aikana tutuksi. Yleisesti opinnäytetyön tekeminen opetti paljon asioita haitallisten tiedostojen analysoinnista ja käsittelystä, sekä yleinen kyberturvallisuuden näkemys globaalisti kehittyi.

Toteutetusta järjestelmästä, IRMAsta, oli saatavissa melko hyvä dokumentaatio, mutta puutteita dokumentaatiossa oli runsaasti, sekä osa tiedoista oli jo vanhentunut. Aikaa meni paljon tutustuessa IRMAssa käytettävien komponenttien dokumentaatioihin ja niiden konfigurointiin. Aikaa kului koko järjestelmän toimintakuntoon laittamisessa, mutta myös jokaisen skannerin kanssa oli oma työnsä ja ongelmansa, jotta ne sai toimimaan järjestelmässä.

Yllättävää oli se, kuinka nopeasti toteutettu järjestelmä toimi haittaohjelmien tarkastamisessa eli kuinka nopeasti saatiin tulokset takaisin käyttäjälle tarkastamisen aloittamisen jälkeen. Käyttöjärjestelmän uudelleenkäynnistyksen jälkeen joillain skannereilla saattoi mennä 20-30 sekuntia yksinkertaisen tiedoston tunnistamiseen, joten skannerit näköjään tarvitsivat yhden ”lämmittely kierroksen”. Muuten skannerit palauttivat tulokset noin 1-10 sekunnin aikana.

Opinnäytetyön aikana varsinkin tietoperustan hankkimisen aikana tuli luettua paljon artikkeleita ja uutisia tietoturvallisuudesta. Yleisen kuvan hankkiminen nykypäivän tietoturvan merkittävyydestä toi paljon motivaatiota itse opinnäytetyön toteuttamiseen. Opinnäytetyön aikana myös mielenkiinto tietoturvaan ja kyberturvallisuuteen kasvoi merkittävästi entisestään.

10.2 Jatkokehitysehdotukset

Toteutetun skannausjärjestelmän toimintaa olisi mielenkiintoista nähdä käytettävän esimerkiksi USB-tikkujen automaattiseen tarkastamiseen. Alustana järjestelmälle voisi toimia Rasperry Pi pientietokone, johon käyttäjä asettaisi USB-tikkunsa tarkastettavaksi. Tietokoneeseen tulisi olla myös liitettynä erillinen näyttö, jossa voisi seurata tikulla sijaitsevien tiedostojen analysointia. Lopuksi näytöllä esitettäisiin tiedostoanalysaattoreiden tulokset kootusti. USB-tikun irrottamisen jälkeen järjestelmä palautuisi alkutilaan ja olisi valmiina tarkastamaan seuraavan USB-tikun sisällön.

Kyseinen USB-tikkujen skannausjärjestelmä voisi tuoda yrityksille aivan uuden tason turvallisuutta. Pihoilta löydetyt USB-tikut työnnetään valitettavan usein yrityksen verkossa olevaan tietokoneeseen katsoakseen tikun sisältö, jolloin mahdollinen haittaohjelma on jo päässyt yrityksen verkkoon. Kuka tietää, olisiko Stuxnet-madon leviäminen Iranin tehtaan järjestelmiin saatu estettyä käyttämällä USB-tikku ensin tarkastettavana erillisessä skannausjärjestelmässä?

Opinnäytetyössä tehtyyn järjestelmään olisi mahdollista vielä implementoida muutama Windows-pohjainen antivirusskanneri. Tämä vaatisi Windows-käyttöjärjestelmän liittämistä itse pääjärjestelmään, mikä on mahdollista toteuttaa. Windows-pohjaisten skannereiden tuominen järjestelmään toisi tällä hetkellä viisi antivirusskanneria lisää. Toteutettuun järjestelmään on mahdollista liittää myös omia tiedostoanalysaattoreita, joilla tiedostoja tarkastetaan. Toisaalta vain mielikuvitus on rajana järjestelmän kehittämisessä analysaattoreiden osalta, koska järjestelmään voidaan ottaa käyttöön omia analysointimenetelmiä.

Toteutetun järjestelmän Nginx-palvelimen yhteys olisi hyvä muuttaa SSL/TLS-salatuksi. Helpointa olisi käyttää mahdollisesti JYVSECTECin omaa PKI-infrastruktuuria, jossa infran varmentaja (Certification Authority) allekirjoittaisi Nginx-palvelimen varmenteen. Yksi mahdollisuus myös olisi luoda itse allekirjoitettu varmenne, mutta tällöin kaikille web-palvelimeen yhdistäville käyttäjille pitäisi asentaa palvelimen itse allekirjoitettu varmenne, jotta luottamussuhde syntyisi. Siinä mielessä oman PKI-infran käyttäminen olisi järkevää, koska oletettavasti oman infran laitteet luottavat omaan varmentajaansa.

Lähteet

A Year in Review: Cybersecurity Highlights of 2015. 2015. Mcafeen verkkosivut. Viitattu 29.1.2016. https://blogs.mcafee.com/consumer/2015-security-in-review/

About ClamAV. 2016. ClamAVn verkkosivut. Viitattu 10.4.2016. https://www.clamav.net/about

About SQLite. 2016. SQLiten kotisivut. Viitattu 15.5.2016. https://www.sqlite.org/about.html

An Introducytion to Celery. 2009. PowerPoint-esitys. Viitattu 14.5.2016. http://www.slideshare.net/idangazit/an-introduction-to-celery/

Anand, A. 2012. How Antivirus Software Works. Viitattu 13.2.2016. http://bypassthesecurity.blogspot.in/2012/11/how-antivirus-software-works.html

Arntz, P. 2015. Virus or Malware. Viitattu 14.2.2016. https://blog.malwarebytes.org/online-security/2015/05/virus-or-malware/

Data Breach Investigation Report. 2015. Verizonen verkkosivut. Viitattu 27.1.2016. http://www.verizonenterprise.com/DBIR/2015/

Fighting Computer Threats. 2015. Dr.Webin online manuaali. Viitattu 22.5.2016. http://download.geo.drweb.com/pub/drweb/unix/doc/HTML/ControlCenter/en/

F-Secure Threat Report 2015. 2015. F-Securen uhkaraportti vuodelta 2015. Viitattu 13.2.2016. https://www.f-secure.com/en/web/labs_global/whitepapers

Gheorghe, A. 2015. How Does Ransomware Work? Viitattu 14.2.2016. http://www.hotforsecurity.com/blog/

IRMA. 2016. IRMAn dokumentaatio. Viitattu 31.1.2016. https://irma.readthedocs.org/

IRMA. 2016. Quarkslabin verkkosivut. Viitattu 31.1.2016. http://irma.quarkslab.com/

Johansson, L. 2015. Artikkeli RabbitMQsta. Viitattu 14.5.2016. https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

JYVSECTEC – Tietoa meistä. 2016. JYVSECTECin verkkosivut. Viitattu 13.2.2016. http://jyvsectec.fi/fi/tietoa-meista/

Järvinen, P. 2012. Arjen Tietoturva – Vinkit & Ratkaisut. Docendo: Jyväskylä.

Koret, J. & Bachaalany, E. 2015. The Antivirus Hacker's Handbook. Wiley: Indianapolis.

Landesman, M. N.d. Understanding virus names. Viitattu 22.5.2016. http://antivirus.about.com/od/whatisavirus/a/virusnames.htm

Limnéll, J., Majewski, K. & Salminen, M. 2014. Kyberturvallisuus. Docendo: Jyväskylä.

Machine Learning for Anti-virus Software. 2013. Aboutdm verkkosivut. Viitattu 13.2.2016. http://www.aboutdm.com/2013/04/machine-learning-for-anti-virus-software.html

Newsletter. 2015. Comodon verkkosivut. Viitattu 29.1.2016. https://www.comodo.com/resources/home/newsletters/nov-10/ask-geekbuddy.php

Nginx wiki. 2016. Nginxn kotisivut. Viitattu 15.5.2016. https://www.nginx.com/resources/wiki/

PostgreSQL – About. 2016. PostgreSQLn kotisivut. Viitattu 15.5.2016. http://www.postgresql.org/about/

Pure-FTPd. 2016. Pure-FTPdn kotisivut. Viitattu 15.5.2016. https://www.pureftpd.org/project/pure-ftpd

Rotem-gal-oz, A. 2012. Artikkeli RabbitMQsta. Viitattu 14.5.2016. https://dzone.com/articles/rabbitmq-and-short-intro-amqp

Security news. 2013. PCtools verkkosivut. Viitattu 10.2.2016. http://www.pctools.com/security-news/

Sililiano, R. 2014. What is Browser Hijacking. Viitattu 14.2.2016. https://blogs.mcafee.com/consumer/browser-hijacking/

The uWSGI Project. 2014. Dokumentaatio uWSGIstä. Viitattu 15.5.2016. https://uwsgi-docs.readthedocs.io/en/latest/

This big U.S. health insurer just got hacked. 2015. Fortunen verkkosivut. Viitattu 29.1.2016. http://fortune.com/2015/09/10/hack-health-insurer-bluecross/

Threat description. 2015. F-Securen verkkosivut. Viitattu 14.2.2016. https://www.f-secure.com/v-descs/

Vakaasti kohti kyberturvallisuuden kärkimaata. 2015. Viestintäviraston verkkosivut. Viitattu 27.1.2016. https://www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2015/12/ttn2015123 01329.html

What is MongoDB. 2016. MongoDBn kotisivut. Viitattu 15.5.2016. https://www.mongodb.com/what-is-mongodb

What is Python? Executive Summary. 2016. Pythonin kotisivut. Viitattu 14.5.2016. https://www.python.org/doc/essays/blurb/

What Is the Difference: Viruses, Worms, Trojans, and Bots? N.d. Ciscon verkkosivut. Viitattu 1.4.2016. http://www.cisco.com/c/en/us/about/security-center/virus-differences.html

What open source. 2016. Open Sourcen verkkosivut. Viitattu 10.2.2016. https://opensource.com/resources/what-open-source

YARA. 2016. YARAn dokumentaatio. Viitattu 20.5.2016. https://yara.readthedocs.io/en/v3.4.0/

Liitteet

Liite 1. IRMAn tukemat antivirusskannerit

Antivirusskanneri	Käyttöjärjestelmä
Emsisoft	Windows
Avira	Windows
Avast	GNU/Linux
AVG	GNU/Linux
BitDefender	GNU/Linux
ClamAV	GNU/Linux
Comodo	GNU/Linux
Dr.Web	GNU/Linux
EsetNod32	GNU/Linux
eScan	GNU/Linux
F-prot	GNU/Linux
F-secure	GNU/Linux
G Data	Windows
Kapersky	Windows
McAfee	GNU/Linux / Windows
Sophos	GNU/Linux /Windows
Symantec	Windows
VirusBlokAda	GNU/Linux
Zoner	GNU/Linux

Liite 2. Brainin konfiguraatiotiedoston sisältö ja selitteet

Sektio	Kenttä	Arvo	Selite
broker_brain	host	127.0.0.1	Osoite, jossa RabbitMQ sijaitsee
	vhost	mqbrain	Brainille konfiguroitu virtuaalihosti
	username	brain	Brainin käyttäjänimi RabbitMQlle
	password	brain	Brainin salasana RabbitMQlle
	queue	brain	Jono, joka kyselee uusia tehtäviä
broker_probe	host	127.0.0.1	Osoite, jossa RabbitMQ sijaitsee
	vhost	mqprobe	Probelle konfiguroitu virtuaalihosti
	username	probe	Proben käyttäjänimi RabbitMQlle
	password	probe	Proben salasana RabbitMQlle
	queue	probe	Jono, joka kyselee uusia tehtäviä
broker_frontend	host	127.0.0.1	Osoite, jossa RabbitMQ sijaitsee
	vhost	mqfrontend	Frontendille konfiguroitu virtuaalihosti
	username	frontend	Frontendin käyttäjänimi RabbitMQlle
	password	frontend	Frontendin salasana RabbitMQlle
	queue	frontend	Jono, joka kyselee uusia tehtäviä
ftp_brain	host	127.0.0.1	Osoite, jossa FTP-palvelin sijaitsee
	username	probe	Käyttäjänimi, jota Probe käyttää FTP-palvelimella
	password	probe	Salasana, jota Probe käyttää FTP-palvelimella
sqldb	dbms	sqlite	Tietokantamoottori
	dialect		SQLAlchemy dialekti
	username		Tietokannan käyttäjänimi
	password		Tietokannan salasana
	host		Tietokannan hosti
	dbname	/var/irma/db/brain.db	Tietokannan nimi
	tables_prefix	irma	Tietokannan taulukoiden prefix

Liite 3. Yhteys FTP-palvelimeen FTP-SSLn avulla

```
root@brain:~# ftp-ssl localhost
Connected to localhost.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 17:47. Server port: 21.
220-This is a private system - No anonymous login
220 You will be disconnected after 15 minutes of inactivity.
Name (localhost:irmeli): probe
500 This security scheme is not implemented
234 AUTH TLS OK.
[SSL Cipher DHE-RSA-AES256-GCM-SHA384]
200 PBSZ=0
200 Data protection level set to "private"
331 User probe OK. Password required
Password:
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> exit
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
221 Logout.
root@brain:~# ftp-ssl localhost
Connected to localhost.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 17:47. Server port: 21.
220-This is a private system - No anonymous login
220 You will be disconnected after 15 minutes of inactivity.
Name (localhost:irmeli): frontend
500 This security scheme is not implemented
234 AUTH TLS OK.
[SSL Cipher DHE-RSA-AES256-GCM-SHA384]
200 PBSZ=0
200 Data protection level set to "private"
331 User frontend OK. Password required
Password:
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

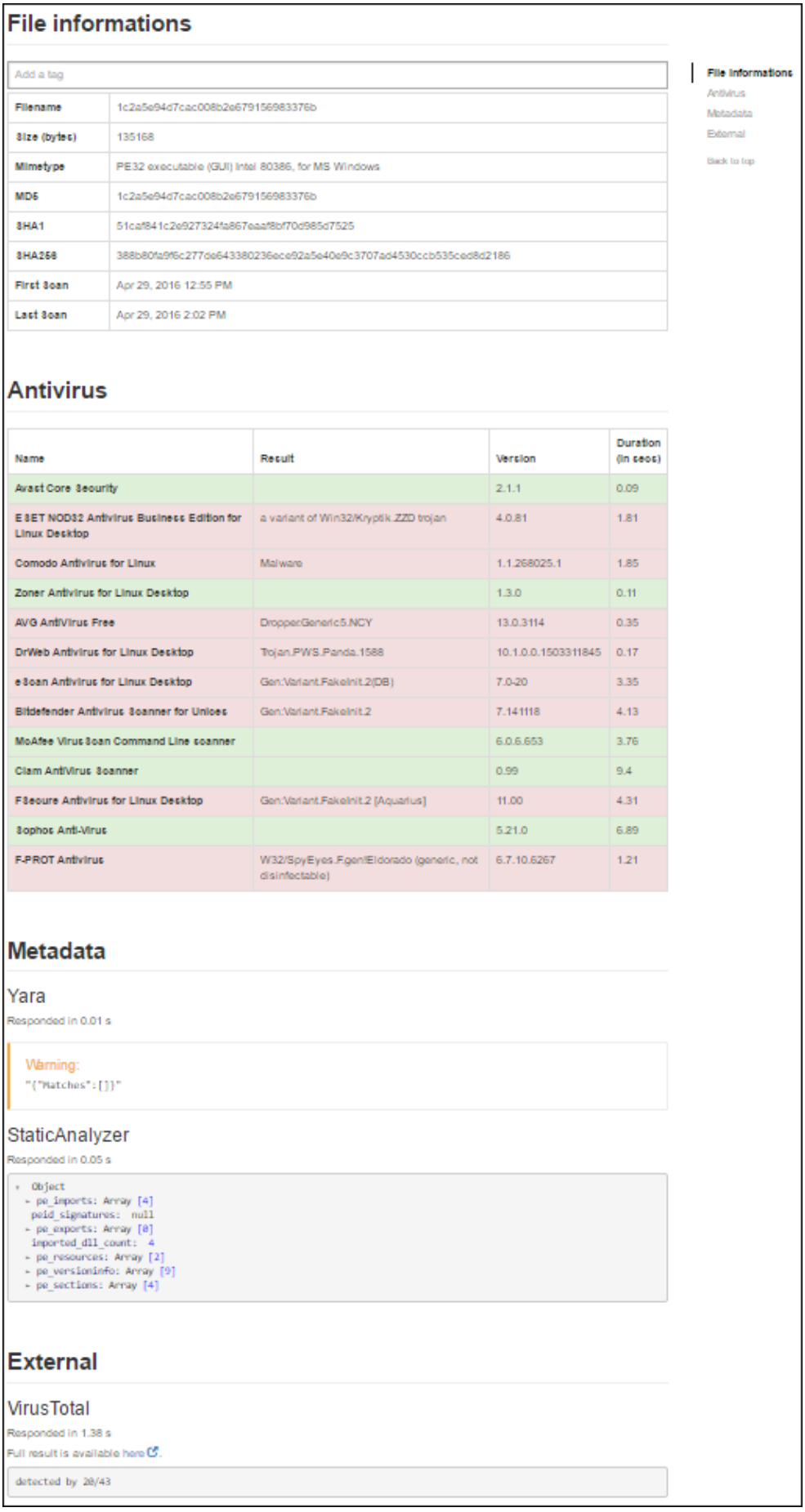
Liite 4. Frontendin konfiguraatiotiedoston sisältö ja selitteet

Sektio	Kenttä	Arvo	Selite
mongodb	host	127.0.0.1	Osoite, jossa MongoDB sijaitsee
	port	27017	Portti, jota MongoDB kuuntelee
	dbname	irma	Tietokannan nimi
	collections_prefix	irma	Prefix MongoDB:n keräyksille
sqldb	dbms	postgresql	Tietokantamoottori
	dialect	psycopg2	SQLAlchemy dialekti
	username	irma	Käyttäjänimi tietokantaan
	password	irma	Salasana tietokantaan
	host	127.0.0.1	Tietokannan osoite
	dbname	irma-frontend	Tietokannan nimi
	tables_prefix	irma	Tietokannan taulukoiden prefix
samples_storage	path	/var/irma/samples/	Polku tiedostojen näytteisiin
broker_brain	host	127.0.0.1	Osoite, jossa RabbitMQ sijaitsee
	vhost	mqbrain	Brainille konfiguroitu virtuaalihosti
	username	brain	Brainin käyttäjänimi RabbitMQlle
	password	brain	Brainin salasana RabbitMQlle
	queue	brain	Jono, joka kyselee uusia tehtäviä
broker_frontend	host	127.0.0.1	Osoite, jossa RabbitMQ sijaitsee
	vhost	mqfrontend	Frontendille konfiguroitu virtuaalihosti
	username	frontend	Frontendin käyttäjänimi RabbitMQlle
	password	frontend	Frontendin salasana RabbitMQlle
	queue	frontend	Jono, joka kyselee uusia tehtäviä
ftp_brain	host	127.0.01	Osoite, jossa FTP-palvelin sijaitsee
	username	frontend	Käyttäjänimi, jota Frontend käyttää FTP-palvelimella
	password	frontend	Salasana, jota Frontend käyttää FTP-palvelimella
cron_frontend	clean_db_file_max_age	0	Poistaa tiedoston X päivän jälkeen (0=ei poisteta)
	clean_db_cron_hour	0	Cron työ tunnin tarkkuudella
	clean_db_cron_minute	0	Cron työ minuutin tarkkuudella
	clean_db_cron_day_of_week	*	Cron työ tiettynä päivänä viikosta
backend_brain	host	127.0.0.1	Parametri RabbitMQlle

Liite 5. Proben konfiguraatiotiedoston sisältö ja selitteet

Sektio	Kenttä	Arvo	Selite
probe	name	hostname	Nimi Probelle
broker_probe	host	127.0.0.1	RabbitMQn osoite
	vhost	mqprobe	Proben virtuaalihosti
	username	probe	Proben käyttäjänimi RabbitMQlla
	password	probe	Proben salasana RabbitMQlla
broker_brain	host	127.0.0.1	RabbitMQn osoite
	vhost	mqbrain	Brainin virtuaalihosti
	username	brain	Brainin käyttäjänimi RabbitMQlla
	password	brain	Brainin salasana RabbitMQlla
	queue	brain	Jono josta uusia tehtäviä kysellään
ftp_brain	host	127.0.0.1	Osoite, jota FTP-palvelin käyttää
	username	probe	Käyttäjänimi, jota Probe käyttää FTP-palvelimella
	password	probe	Salasana, jota Probe käyttää FTP-palvelimella

Liite 6. Tulossivun yleiskatsaus



File informations

Add a tag

Filename	1c2a5e94d7cac008b2e679156983376b
Size (bytes)	135168
Mimetype	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	1c2a5e94d7cac008b2e679156983376b
SHA1	51caf841c2e927324fa867eaaf8bf70d985d7525
SHA256	388b80fa9f6c277de643380236ece92a5e40e9c3707ad4530ccb535ced8d2186
First Seen	Apr 29, 2016 12:55 PM
Last Seen	Apr 29, 2016 2:02 PM

File informations
Antivirus
Metadata
External
Back to top

Antivirus

Name	Result	Version	Duration (in secs)
Avast Core Security		2.1.1	0.09
ESET NOD32 Antivirus Business Edition for Linux Desktop	a variant of Win32/Kryptik.ZZD trojan	4.0.81	1.81
Comodo Antivirus for Linux	Malware	1.1.268025.1	1.85
Zoner Antivirus for Linux Desktop		1.3.0	0.11
AVG AntiVirus Free	DropperGeneric5.NCY	13.0.3114	0.35
DrWeb Antivirus for Linux Desktop	Trojan.PWS.Panda.1588	10.1.0.0.1503311845	0.17
eScan Antivirus for Linux Desktop	Gen:Variant.FakeInit.2(DB)	7.0-20	3.35
Bitdefender Antivirus Scanner for Unices	Gen:Variant.FakeInit.2	7.141118	4.13
McAfee VirusScan Command Line scanner		6.0.6.653	3.76
Clam AntiVirus Scanner		0.99	9.4
FSecure Antivirus for Linux Desktop	Gen:Variant.FakeInit.2 [Aquarius]	11.00	4.31
Sophos Anti-Virus		5.21.0	6.89
F-PROT Antivirus	W32/SpyEyes.FgenlEldorado (generic, not disinfectable)	6.7.10.6267	1.21

Metadata

Yara

Responded in 0.01 s

Warning:
"{"Matches":[]}"

StaticAnalyzer

Responded in 0.05 s

```
▾ Object
  ▸ pe_imports: Array [4]
    peid_signatures: null
  ▸ pe_exports: Array [0]
    imported_dll_count: 4
  ▸ pe_resources: Array [2]
  ▸ pe_versioninfo: Array [9]
  ▸ pe_sections: Array [4]
```

External

VirusTotal

Responded in 1.38 s

Full result is available here.

detected by 20/43

Liite 7. Antivirusskannereiden tulokset EXE1-tiedostolle

Antivirus

Name	Result	Version	Duration (in secs)
Clam AntiVirus Scanner		0.99	30.57
ESET NOD32 Antivirus Business Edition for Linux Desktop	a variant of Win32/Kryptik.ZZD trojan	4.0.81	2.02
F-PROT Antivirus	W32/SpyEyes.F.gen!Eldorado (generic, not disinfectable)	6.7.10.6267	1.54
Sophos Anti-Virus		5.21.0	7.3
AVG AntiVirus Free	Dropper.Generic5.NCY	13.0.3114	0.13
Bitdefender Antivirus Scanner for Unices	Gen:Variant.Fakelnit.2	7.141118	3
eScan Antivirus for Linux Desktop	Gen:Variant.Fakelnit.2(DB)	7.0-20	2.91
Zoner Antivirus for Linux Desktop		1.3.0	0.02
FSecure Antivirus for Linux Desktop	Gen:Variant.Fakelnit.2 [Aquarius]	11.00	0.63
McAfee VirusScan Command Line scanner		6.0.6.653	1.53
Comodo Antivirus for Linux	Malware	1.1.268025.1	2.13
Avast Core Security		2.1.1	0.04
DrWeb Antivirus for Linux Desktop	Trojan.PWS.Panda.1588	10.1.0.0.1503311845	0.89

Liite 8. Antivirusskannereiden tulokset EXE2-tiedostolle

Antivirus

Name	Result	Version	Duration (in secs)
Clam AntiVirus Scanner	Win.Trojan.Agent-36200	0.99	8.11
ESET NOD32 Antivirus Business Edition for Linux Desktop	Win32/Spatet.I trojan	4.0.81	1.68
F-PROT Antivirus	W32/Rebhip.A.gen!Eldorado (generic, not disinfectable)	6.7.10.6267	1.5
Sophos Anti-Virus	Mal/Behav-328	5.21.0	7.38
AVG AntiVirus Free	PSW.Delf.3.E	13.0.3114	0.24
Bitdefender Antivirus Scanner for Unices	Trojan.Generic.6296148	7.141118	3.09
eScan Antivirus for Linux Desktop	Trojan.Generic.6296148(DB)	7.0-20	2.74
Zoner Antivirus for Linux Desktop	Trojan.Spatet.I	1.3.0	0.01
FSecure Antivirus for Linux Desktop	Trojan.Generic.6296148 [Aquarius]	11.00	3.83
McAfee VirusScan Command Line scanner		6.0.6.653	1.44
Comodo Antivirus for Linux	Backdoor.Win32.Delf.~DF	1.1.268025.1	2.1
Avast Core Security	Win32:BackDoor-ACX [Trj]	2.1.1	0.09
DrWeb Antivirus for Linux Desktop	Trojan.PWS.Stealer.15081	10.1.0.0.1503311845	0.71

Liite 9. Antivirusskannereiden tulokset PDF-tiedostolle

Antivirus

Name	Result	Version	Duration (in secs)
Clam AntiVirus Scanner	Win.Trojan.MSShellcode-7	0.99	0.07
ESET NOD32 Antivirus Business Edition for Linux Desktop	PDF/Exploit.Pidief.PFW trojan	4.0.81	3.13
F-PROT Antivirus	W32/Swrort.A.gen!Eldorado (generic, not disinfectable)	6.7.10.6267	1.16
Sophos Anti-Virus	Troj/PDFJs-AIA	5.21.0	7.46
AVG AntiVirus Free	Exploit.PDF	13.0.3114	0.37
Bitdefender Antivirus Scanner for Unices	Exploit.PDF-Dropper.Gen	7.141118	2.68
eScan Antivirus for Linux Desktop	Exploit.PDF-Dropper.Gen(DB)	7.0-20	3.06
Zoner Antivirus for Linux Desktop		1.3.0	0.11
F-Secure Antivirus for Linux Desktop	Exploit.PDF-Dropper.Gen [Aquarius]	11.00	0.6
McAfee VirusScan Command Line scanner	Swrort.i trojan	6.0.6.653	1.43
Comodo Antivirus for Linux		1.1.268025.1	1.32
Avast Core Security	Win32:SwPatch [Wrm]	2.1.1	0.09
DrWeb Antivirus for Linux Desktop	Trojan.Swrort.1	10.1.0.0.1503311845	0.72