jamk.fi

Haittaohjelmien analysointijärjestelmä

Cuckoo Sandbox

Mika Kultanen

Opinnäytetyö
Toukokuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietotekniikan tutkinto-ohjelma
Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kultanen,Mika	Opinnäytetyö, AMK	20.5.2016
	Sivumäärä 61	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Haittaohjelmien analysointijärjestelmä
Cuckoo Sandbox

Tutkinto-ohjelma
Tietotekniikan (Tietoverkkotekniikan) koulutusohjelma

Työn ohjaaja(t)
Antti Häkkinen, Sampo Kotikoski

Toimeksiantaja(t)
Marko Vatanen

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun JYVSECTEC-hanke. JYVSECTEC on kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus.

Opinnäytetyön tavoitteena oli toteuttaa, testata ja dokumentoida avoimen lähdekoodin haittaohjelmien analysointijärjestelmä Cuckoo Sandbox. Lisäksi toteutuksessa otettiin käyttöön Cuckoo Sandboxin laajennus Android-käyttöjärjestelmille nimeltä CuckooDroid.

Työn teoriaosuudessa keskityttiin haittaohjelmien analyysin perusteoriaan. Sen lisäksi käytiin läpi työn toteutuksessa vaaditut tiedot virtualisoinnista ja sandboxauksesta.

Järjestelmä toteutettiin kokonaisuudessaan JYVSECTECin vCloud virtuaaliympäristöön. Palvelimena järjestelmälle käytettiin Linux Debian-palvelinta ilman graafista käyttöliittymää. Kaikki toteutuksessa käytetyt työkalut ovat kuluttajille ilmaiseksi saatavilla. Järjestelmässä haittaohjelmia analysoitiin eristetyssä sandbox-ympäristössä. Toteutuksessa analyysiä suorittavat tietokoneet toteutettiin virtualisoinnin avulla VirtualBox-työkalulla.

Tuloksena saatiin toimivan Cuckoo Sandbox analysointijärjestelmän Windows- ja Android-käyttöjärjestelmien haittaohjelmille. Analyysin suorittaminen työkalulla onnistuu helposti web-käyttöliittymän avulla. Oli huomattavaa, että näinkin kattavan järjestelmän voitiin Open Source –tuotteiden avulla toteuttaa vähillä resursseilla ja ilman kuluja.

Avainsanat (asiasanat)
Haittaohjelma-analyysi, JYVSECTEC, Cuckoo Sandbox, Cuckoo Droid

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Kultanen, Mika	Bachelor's thesis	20.5.2016
		Language of publication: Finnish
	Number of pages 61	Permission for web publication: x

Title of publication
Malware Analysis System
Cuckoo Sandbox

Degree programme

Supervisor(s)
Antti Häkkinen, Sampo Kotikoski

Assigned by
Marko Vatanen

Abstract

This bachelor's thesis was assigned by JYVSECTEC. JYVSECTEC is a cybersecurity investigation, development and training center at JAMK University of Applied Sciences.

The goal of the thesis was to implement, test and document an Open Source malware analysis system called Cuckoo Sandbox. Along with Cuckoo Sandbox an extension for Android operating systems called CuckooDroid was implemented.

The theoretical part of the thesis focuses on the basic theory of malware analysis. The information needed about virtualization and sandboxing for implementation was also discussed.

The system was implemented to the vCloud virtual environment of JYVSECTEC. The system was built on Linux Debian server without a graphical user interface. All tools used in the implementation are free for personal use. The malware was analyzed in an isolated sandbox environment. The guest machines were created using virtualization with VirtualBox tool.

The thesis resulted in a fully functioning Cuckoo Sandbox malware analyzing system for Windows and Android malware. With the system it easy to analyze malware using a web interface. It was notable that such an extensive system could be implemented with limited resources and without costs using Open Source products.

Keywords/tags (subjects)
Malware analysis, JYVSECTEC, Cuckoo Sandbox, CuckooDroid

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

APK	Android application package
API	Application programming interface
DNS	Domain Name system
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JYVSECTEC	Jyväskylä Security Technology
PE	Portable Executable
RGCE	Realistic Global Cyber Enviroment
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VDI	Virtual desktop infrastructure
VMDK	Virtual Machine Disk
VRDP	VirtualBox Remote Display Protocol

1 Lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi JYVSECTEC. Se on Jyväskylän ammattikorkeakoulun IT-instituutissa toimiva kyberturvallisuuden tutkimus-, kehitys- ja koulutuskeskus. JYVSECTEC tarjoaa asiakkailleen kyberturvallisuusharjoituksia, konsultointi-, tutkimus-, testaus- ja koulutuspalveluja. JYVSECTEC-projekti käynnistyi vuonna 2011 vastaamaan silloin jo ajankohtaiseen haasteeseen kyberturvallisuus. Sen tavoitteena oli luoda Keski-Suomeen yksi Suomen johtavimmista kyberturvallisuuden keskittymistä sekä kehittää turvallisuusalan toimijoiden yhteistyöverkostoa. Vuonna 2015 saadun jatkorahoituksen ansiosta jatkuu JYVSECTECin työ kyberturvallisuuden parissa ainakin vuoden 2017 loppuun asti. (JYVSECTEC – Tieto meistä 2016)

JYVSECTECin kehittämä ja ylläpitämä RGCE-kybertoimintaympäristö pyrkii simuloimaan oikeaa Internetiä. Ympäristö vastaa rakenteeltaan ja toiminnallisuuksiltaan todellista julkista Internetiä. RGCE-ympäristö on kuitenkin eristetty julkisesta verkosta, jolloin siellä voidaan turvallisesti ja kontrolloidusti testata mahdollisia tietoturvauhkia. Ympäristöön voidaan simuloida todellista verkkoliikennettä niin normaalia käyttäjäliikennetteä kuin hyökkäysliikennetäkin. (RGCE - kybertoimintaympäristö 2016)

1.2 Opinnäytetyön aihe ja tavoitteet

Opinnäytetyön aihe on automatisoitu haittaohjelmien analysointijärjestelmä. Työssä järjestelmänä käytettiin avoimen lähdekoodin tuotetta Cuckoo Sandbox. Työn teoriaosuudessa tavoitteena oli kiteyttää haittaojelmien analysointi ja siihen käytettävät tekniikat. Järjestelmäntoteuteutuksessa tavoitteena oli analysoida haittaohjelmia Windows 7 (32- ja 64 bittiset) ja Android-käyttöjärjestelmissä. Android käyttöjärjestelmää varten järjestelmään integroitiin CuckooDroid -laajennus. Järjestelmän toteutus käytiin läpi dokumentissa lisäosineen kohta kohdalta. Lisäksi järjestelmän käyttö ja toiminnallisuus dokumentoitiin työhön niin, että analyysit ovat

myöhemmin toistettavissa dokumentin avulla. Lisäksi tavoitteena oli avata käyttäjälle analyysiraportin tuloksia.

2 Haittaohjelmien analysointi

2.1 Haittaohjelmat

Haittaohjelma (engl. Malicious Software) eli malware on sovellus, joka aiheuttaa jollain tapaa haittaa tietokoneelle, käyttäjälle tai tietoverkolle. Tyypillisiä haittaohjelmatyyppejä ovat muun muassa virukset, troijalaiset, madot, vakoiluohjelmat, takaovet ja rootkitit. (Sikorski & Honig 2012.)

On tärkeää tietää minkä tyyppisiä haittaohjelmia on olemassa, jotta voidaan paremmin arvioida mitä vastassa oleva haittaohjelma tyypillisesti tekee. Yleensä haittohjelma kuuluu useaan kategoriaan. Ohessa selitetty yleisimpiä haittaohjelmatyyppejä (Sikorski & Honig 2012):

Takaovi - Takaovi nimensä mukaisesti avaa hyökkääjälle pääsyn paikalliseen järjestelmään ohi suojausten ”takaoven kautta” suorittamaan haluamiaan toimenpiteitä.

Bottiverkko (engl. Botnet) – Bottiverkossa kaikkia tarttuneita tietokoneita eli botteja hallitaan yhteisesti hallintapalvelimelta.

Downloader – Haittaohjelma, jonka ainoa tehtävä on ladata ja asentaa järjestelmään toinen haittaohjelma.

Tietoja varastavat haittaohjelmat – Haittaohjelma kerää tietoja tarttuneelta tietokoneelta ja lähettää ne hyökkääjälle. Yleensä tavoitteena on saada käsiin uhrin pankkitietoja ja muita kirjautumistunnuksia. Tyypillisesti tälläiset haittaohjelmat ovat keyloggereita tai sniffereitä.

Launcher – Haittaohjelman tehtävä on käynnistää toinen haittaohjelma. Tavoitteena on haittaohjelman parempi salassa pysyminen ja parempi pääsy järjestelmiin.

Rootkit – Rootkitit tulevat yleensä muiden haittaohjelmien mukana, ja niiden tehtävä on vaikeuttaa varsinaisen haittaohjelman havaitsemista kätkemällä se.

Peloitteluohjelma (engl. Scareware) – Ohjelmien tavoitteena on saada tarttunut käyttäjä kuluttamaan rahaa pelottelemalla tätä esimerkiksi virusvaroituksilla.

Roskapostiohjelma – Haittaohjelma käyttää tarttunutta tietokonetta lähettämään roskapostia verkkoon. Hyökkääjä hyötyy tästä myymällä roskapostituspalvelua.

Madot ja virukset – Haittaohjelmaia, jotka voivat monistaa itsensä ja tarttua automaattisesti muihin koneisiin esimerkiksi lähiverkossa.

Haittaohjelmat voidaan myös luokitella siten, onko hyökkääjän kohteena suuri massa uhreja vai onko hyökkäys kohdennettu tiettyyn järjestelmään. Massahyökkäyksissä tavoitteena on tartuttaa mahdollisimman monta konetta. Koska kohteita on paljon, ovat haittaohjelmat usein tunnettuja ja ne jäävät kiinni ajantasaisille virustorjuntaohjelmistoille. Myöskään koodit eivät ole niin hienostuneita ja ne on helpompi havaita. (Sikorski & Honig 2012.)

Kohdennetuissa hyökkäyksissä haittaohjelma suunnitellaan varta vasten kohteena olevaan järjestelmään, jolloin se ei ole ennalta tunnettu. Tällaisilta haittaohjelmilta on erittäin vaikea, ellei mahdoton, suojautua ilman tarkkaa analyysiä. Näissä tapauksissa myös ohjelman koodi on kehittynyttä. (Sikorski & Honig 2012.)

2.2 Miksi?

Haittaohjelmien kehittyessä jatkuvasti on tärkeää pystyä suojautumaan niiltä entistä tehokkaammin. Jotta haittaohjelmilta voidaan suojautua, täytyy ne osata tunnistaa ja ymmärtää, kuinka ne toimivat. Tähän tarvitaan haittaohjelmien analysointia. Analysointi on jatkuvaa kamppailua koko ajan uudistuvia haittaohjelmia vastaan, joiden mukana on pysyttävä. Analysointiin on monia erilaisia työkaluja ja tekniikoita, mutta kaikkien tavoitteena on saada selville haittaohjelman heikkoudet, osat ja käyttäytyminen. Tekniikat voidaan karkeasti jakaa kahteen ryhmään: staattinen ja dynaaminen analyysi. Tässä työssä keskitytään automatisoituun analyysiin Cuckoo Sandbox –työkalulla, jossa on yhdistetty molemmat tekniikat.

2.3 Staattinen analyysi

Staattista analysointia tehtäessä ei haittaohjelmaa tarvitse suorittaa, vaan analysoidaan suoraan ohjelman lähdekoodia. Lähdekoodinäytteitä ei ole juuri saatavilla, joten tätä varten täytyy ohjelman suoritustiedosto kääntää takaisin lähdekoodiksi. Tämä tekniikka on turvallisempaa, mutta se vaatii paljon ohjelmointiosaamista. Tekniikkaa käytetään usein analysointiprosessin alkuvaiheessa sen turvallisuuden vuoksi. Haasteena on kuitenkin nykyaikaisen koodin monimutkaisuus. Haittaohjelmasta voidaan kuitenkin saada paljon hyödyllistä tietoa hyvinkin yksinkertaisilla tekniikoilla (Oktavianto & Muhardianto 2013.)

2.3.1 Antivirus

Hyvä käytäntö analyysin alussa skannata haittaohjelma olemassa olevilla antivirus-ohjelmistoilla. Haittaohjelma voi olla jo aikaisemmin tunnistettu jonkun antiviruksen toimesta. Antivirukset luottavat pääasiassa tietokantoihin, jotka sisältävät tunnistettavia osia tiedostoista, joihin näytettä verrataan. Jotta haittaohjelmat löytyvät tietokannasta, on niiden oltava aikaisemmin tunnettuja. Haittaohjelmien kehittäjät myös aktiivisestiä muokkaavat ohjelmaansa antivirus-ohjelmien varalle. Kohdennetummat haittaohjelmat täten epätodennäköisemmin löytyvät tietokannoista. Eri antivirukset käyttävät hiukan erilaisia tapoja haittaohjelmien tunnistamiseen, jonka vuoksi on suotavaa ajaa haittaohjelma useamman antivirus-skannin läpi. (Sikorski & Honig 2012.)

2.3.2 Fuzzy Hashing

Mahdollisesti haitallisia ohjelmia voidaan tunnistaa vertailemalla tiedostojen kryptografisten tiivistealgoritmien kuten MD5, SHA-1 tai SHA-256 tuloksia toisiinsa. Näin voidaan vertailla haittaohjelmia esimerkiksi haittaohjelmatietokannoista. Tiedoston ”hash” pysyy samana, vaikka sen nimeä muuttaa. Tavallisten tiivistefunktioiden heikkous tässä tarkoituksessa on kuitenkin se, että kun tiedoston sisältö muuttuu vähääkään, muuttuu koko tulos tunnistamattomaksi. Kyseiseen ongelmaan ratkaisu on Fuzzy Hashing. Fuzzy Hashingin avulla löydetään vertailtavista tiedostoista samankaltaisuuksia helposti, sillä algoritmi ei muuta koko arvoa vaan siitä on tunnistettavissa yhtenäisyyksiä. Tämän avulla voidaan siis vertailla

haittaohjelmanäytettä olemassa olevaan tietokantaan tai tutkia kahta eri näytettä samankaltaisuuksien varalta. (Dunham 2013.)

2.3.3 Merkkijonojen analysointi

Epäilyttävästä suoritustiedostosta voidaan hakea hyödyllisiä merkkijonoja. Kun merkkijonot käännetään tavuista ASCII-muotoon, on iso osa tuloksista lukukelvotonta, mutta ohjelman kannalta merkittävää dataa. Jonoista etsitään kuitenkin luettavaa tietoa, kuten tiedostonimiä, IP-osoitteita tai muuta haittaohjelmaan toimintaan viittaavaa tietoa. (Sikorski & Honig 2012.)

2.3.4 Tiedoston pakkaaminen ja obfuskointi

Luotettavat ohjelmat sisältävät yleisesti suuren määrän luettavia ASCII-jonoja. Haittaohjelmat tyypillisesti pyrkivät estämään tai vaikeuttamaan staattista analysointia, jolloin ohjelma pakataan tai obfuskoidaan eli tahallisesti sekoitetaan. Pakattu tai obfuskoitu haittaohjelma sisältää erittäin vähän jonoja, jonka pitäisi herättää epäilys. Pakkauksessa varsinainen ohjelma piilotetaan pienemmän wrapper-ohjelman taakse, josta ei saada haittaohjelmaa vaarantavaa tieto irti. (Sikorski & Honig 2012.)

2.3.5 PE-tiedostot

Tiedoston muodosta voidaan saada paljon tietoa ohjelman toiminnallisuudesta. PE-tiedostomuotoa käytetään Windowsin suoritustiedostoissa, objektitiedostoissa ja DLL-tiedostoissa. PE-tiedostomuoto tietorakenne, joka sisältää Windows-käyttöjärjestelmälle oleelliset tiedot ohjelmasta. Lähes kaikki Windowsissa ajettavat ajotiedostot ovat PE-muodossa. PE-tiedoston otsikkokentästä saadaan analyysin kannalta hyödyllistä tietoa, kuten sovelluksen tyyppi, tarvittavat funktiokirjastot ja tilavaatimukset. (Sikorski & Honig 2012.)

2.4 Dynaaminen analyysi

Dynaamisessa analyysissä suoritetaan haittaohjelma ja valvotaan sen toimintaa. Erityisesti valvotaan muutoksia järjestelmässä haittaohjelman suorittamisen jälkeen. Haittaohjelmien suorittamisella on aina riskinsä, ja sen vuoksi dynaamista

analysointia tehtäessä täytyy ympäristö olla hyvin eristetty. Järjestelmään tapahtuu muutoksia ja se voi tuhoutua, joten sen täytyy olla nopeasti ja helposti palautettavissa lähtötilaan. Ympäristön pitää olla eristettynä tuotantoverkosta analyysiä tehtäessä. Dynaamista analyysiä tehtäessä seurataan muutoksia tiedostojärjestelmässä, rekisterissä, prosesseissa ja verkkoliikenteessä. Dynaamisen analyysin hyöty on haittaohjelman toiminnan täydellinen ymmärtäminen. (Oktavianto & Muhardianto 2013.)

2.4.1 Käyttöjärjestelmän monitorointi

Kun haittaohjelma suoritetaan analyysiympäristössä, monitoroidaan käyttöjärjestelmässä tapahtuvia muutoksia rekisteriin ja tiedostojärjestelmään. Lisäksi seurataan käyttöjärjestelmän prosesseja. Muutoksien seuraaminen tapahtuu erillisellä työkalulla. Tapahtumissa rekisterissä voidaan päätellä, kuinka haittaohjelma asentaa itsensä rekisteriin. Tiedostojärjestelmän muutoksista puolestaan voidaan saada selville haittaohjelman luomat tiedostot ja mahdolliset sen käyttämät konfiguraatiotiedostot. Prosesseja seuraamalla nähdään haittaohjelman luomat uudet prosessit käyttöjärjestelmään. Prosesseista koostetaan prosessipuu, joka havainnollistaa haittaohjelman käyttäytymistä järjestelmässä. (Sikorski & Honig 2012.)

2.4.2 Verkon monitorointi

Verkkoa voidaan monitoroida analyysin aikana sovelluksella, joka kaappaa analyysikoneen verkkorajapinnan läpi kulkevan verkkoliikenteen. Verkkoliikenteestä voidaan saadaan selville mihin haittaohjelma on yhteydessä eli mahdollisen isäntäpalvelin osoitteet. Lisäksi voidaan seurata haittaohjelman tekemiä DNS-kyselyitä. Mahdolliset haitalliset verkko-osoitteet voidaan esimerkiksi estää palomuurilla. Tämän lisäksi tiedetään haittaohjelman käyttämät verkkoprotokollat. (Sikorski & Honig 2012.)

2.4.3 Muistianalyysi

Tietokoneen keskusmuistista eli RAMista voidaan saada hyödyllistä tietoa järjestelmän tilasta sen ollessa aktiivinen. Muistianalyysissä täytyy tallentaa kopio

muistista eli muistivedos (engl. memory dump) ja analysoida se valitulla työkalulla erillisellä tietokoneella. Täten saadaan selville analyysin aikainen muistin tila järjestelmässä. Nähdään myöskin mitkä sovellukset oli käynnissä, sovellusten käyttämät tiedostot, avoimet verkkoyhteydet ja paljon muuta tietoa haittaohjelman käyttäytymisestä. (Ligh, Adair & Hartstein 2010.)

3 Sandboxing

3.1 Yleistä

Sandboxing tarkoittaa turvallista eristettyä ympäristöä, joka simuloi todellista tuotantoympäristöä. Ympäristössä voidaan turvallisesti suorittaa esimerkiksi uusia epäluotettavia ohjelmia tai haittaohjelmia. Tapahtuneiden muutosten jälkeen on järjestelmä pystyttävä palauttamaan nopeasti lähtötilaansa. Sandbox-ympäristö luodaan tarkoitusta varten sopivaksi käyttöjärjestelmän, sovellusten, yhteyksien ja muun osalta. Yleensä se toteutetaan virtualisoinnin avulla, mutta se voidaan toteuttaa myös fyysisillä laitteilla. Oleellista sandbox-ympäristössä on se ettei ympäristöstä ole pääsyä host-koneelle tai muuhun tuotantoympäristöön.

3.2 Haittaohjelmat sandbox-ympäristössä

Koska sandboxing on tehokas ja paljon käytetty tekniikka haittohjelmien päihittämiseksi, on myös haittaohjelmien kehittäjien täytynyt reagoida tähän. Monet haittaohjelmat pyrkivät havaitsemaan sandboxin tai virtuaalikoneen ennen haitallisen koodin suorittamista. Näin voidaan saada virheellinen tieto tiedoston puhtaudesta ja haittaohjelma pääsee kiinni oikeaan ympäristöön. Haittaohjelmaa on tällöin myös vaikeampi tutkia. Yleisimmät tekniikat voidaan jakaa karkeasti kolmeen luokkaan: Inhimillisen toiminnan tunnistaminen, konfiguraation hyödyntäminen ja virtuaaliympäristön tunnistaminen. (Vashisht & Singh 2014)

Haittohjelmat voivat vaatia jotakin toimia käyttäjältä ennen haittaohjelman suorittamista, jolloin varmistuttaisiin aidosta ympäristöstä. Sandbox kehittäjät kuitenkin pyrkivät simuloimaan sandboxissa käyttäjän hiirenliikkeitä ja muuta

toimintaa, jotta se vaikuttaisi realistiselta. Haittaohjelmia on kuitenkin täten vaikea huijata aina jatkuvassa kehityksessä. (Vashisht & Singh 2014.)

Toinen tekniikka on sandbox-ympäristölle ominaisten asetusten hyödyntäminen. Yleisin tällainen tekniikka on viivästyttää haittaohjelman oikeaa suorittamista tietyllä ajanjaksolla, jolloin mahdollinen sandbox-analyysi aikakatkaistaan ja todetaan tiedosto puhtaaksi. (Vashisht & Singh 2014.)

Kolmantena tekniikkana haittaohjelma pyrkii tunnistamaan sen, että sitä ajetaan virtualisoidussa ympäristössä. Sandboxit toteutetaan yleensä virtualisoidusti ja siksi haittaohjelmat pyrkivät tunnistamaan virtuaalikoneet. Mitä tunnetumpia virtualisointituotteita käytetään sitä enemmin haittaohjelmat pyrkivät heikkouksia löytämään. Haittaohjelmat voivat tunnistaa esimerkiksi tunnettuja virtualisointituotteiden prosesseja käynnissä kuten vboxtray.exe, vmtoolsd.exe ja vboxservice.exe, joten ainakin VM ajureiden asennusta tulisi välttää (Guest Additions). Lisäksi virtuaalikone voidaan tunnistaa esimerkiksi rekisteritiedoista, dll-moduuleista, tiedostojärjestelmän virtualisoinnille tyypillisistä tiedostoista ja esimerkiksi verkkorajapinnan MAC-osoitteesta. Tekniikoita tunnistamiseen on paljon virtualisointituotteesta riippuen. Haittaohjelman tunnistaessa sitä ajettavan virtuaalisessa ympäristössä keskeytetään suoritus. (Singh n.d.)

4 Virtualisointi

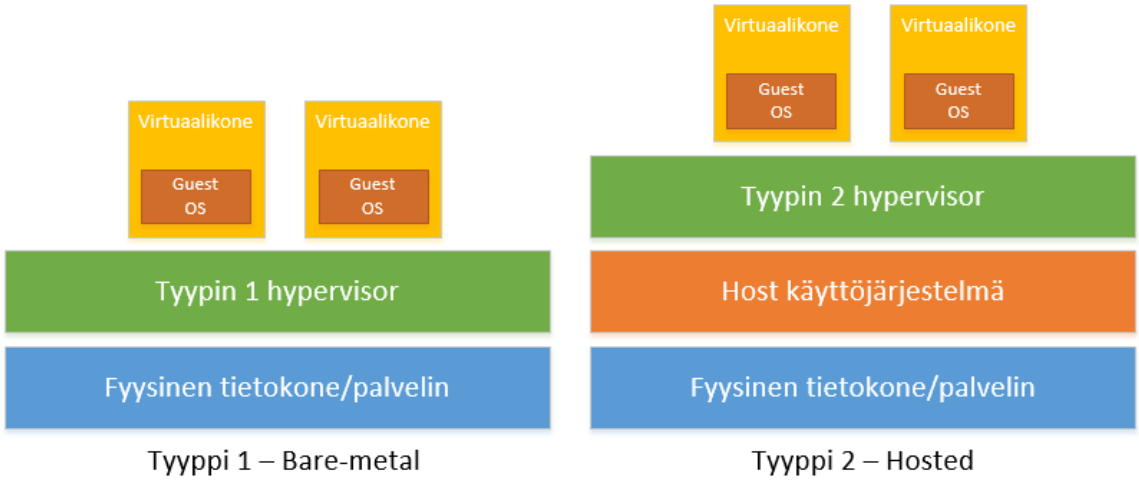
4.1 Hypervisor

Virtualisoinnilla tarkoitetaan tietokoneen fyysisien resurssien jakamista loogisiin osiin, tavoitteenaan käyttää tehokkaammin hyödyksi käytettävissä olevat resurssit. Työasemavirtualisoinnissa näitä resursseja on esimerkiksi prosessori, keskusmuisti, levytila ja verkkokortti. Fyysisen tietokoneen ja virtuaalikoneen välillä on sovelluskerros eli hypervisor virtualisoinnin ohjaamiseksi. Hypervisor voidaan toteuttaa kahdella tapaa: (Portnoy 2012)

Tyypin 1 hypervisor toteutetaan suoraan fyysisen tietokoneen päälle ilman erillistä käyttöjärjestelmää sen taustalla. Tätä tekniikkaa käytetään palvelinvirtualisoinnissa sen tehokkuuden ja paremman luotettavuuden vuoksi. Tunnetuimpia tyypin 1

hypervisoreita ovat Microsoft Hyper-V, Vmware ESX/ESXi ja Citrix XenServer. Tekniikkaa kutsutaan myös nimellä ”bare-metal hypervisor”. (Portnoy 2012)

Tyypin 2 hypervisor toteutetaan siten, että tietokoneelle asennetaan normaalisti esimerkiksi Linux- tai Windows-käyttöjärjestelmä, jonka päällä hypervisor pyörii. Tekniikka ei ole niin tehokas tai luotettava kuin tyyppi 1, mutta se on helppo asentaa ja käyttää. Tämä tekniikka on monesti käytössä testauksessa tai muissa tilanteissa, jossa käytetään useita käyttöjärjestelmiä työasemalla. Tunnetuimpia tyypin 1 hypervisoreita ovat Oracle Virtual Box ja VMware Workstation. Tämä tunnetaan myös nimellä ”hosted hypervisor”. Kuviossa 1 on havainnollistettu eri tyyppien eroavasuus ja rakenne. (Portnoy 2012)



Kuvio 1. Hypervisor-tyypit

4.2 Virtuaalikoneet

Virtuaalikone pyörii hypervisorin päällä ja siihen asennetaan oma käyttöjärjestelmä tavallisen tietokoneen tapaan. Virtuaalikoneelle allokoidut resurssit näkyvät ja käyttäytyvät koneessa kuten fyysisissäkin tietokoneissa. Virtuaalikone koostuu konfiguraatiotiedostosta, jossa määritetään sen ominaisuudet, ja lisäksi virtuaalinen levytiedosto esimerkiksi .VDI tai .VMDK. Virtalisointisovellusta ajavaa palvelinta kutsutaan isännäksi eli hostiksi ja sen ajamia virtuaalikoneita kutsutaan vieraiksi eli guesteiksi. Virtuaalikoneen voi yksinkertaistetusti jakaa kerroksiin virtual hardware-, käyttöjärjestelmä- ja sovelluskerros. (Portnoy 2012)

4.3 Ominaisuuksia

Etuna fyysiseen toteutukseen virtualisoinnilla on, paitsi tehokkaampi resurssien hyödyntäminen, myös reduntanttisuus ja skaalautuvuus. Virtuaalikoneita on helppo kopioida eli kloonata ja järjestelmiin voidaan myös helposti tuoda jo valmiita työasemapohjia eli templateja. Näiden lisäksi virtuaalikoneen sen hetkinen tila voidaan helposti varmuuskopioida snapshot-ominaisuudella, josta palautuminen tapahtuu hetkessä. Snapshotin avulla saadaan virtuaalikone myös käynnistämään aina tietyssä tilassa. Jos halutaan 64-bittinen guest-käyttöjärjestelmä, on isäntäkoneelta löydyttävä hardware virtualisoinnin tuki VT-x tai AMD-V.

4.4 VirtualBox

VirtualBox on Oraclen alustariippumaton virtualisointisovellus. Sovellus on ilmaiseksi ladattavissa yksityiseen ja koulutuskäyttöön. Sovellus toimii Windows, Linux, Macintosh, ja Solaris host-käyttöjärjestelmissä. Guest-käyttöjärjestelmien tuki kattaa käytännössä kaikki tunnetut käyttöjärjestelmät. VirtualBoxia voidaan käyttää graaffisessa käyttöliittymässä tai terminaalissa VboxManagen avulla. Sovellus tukee myös headless-asennusta ilman graafista käyttöliittymää, mutta päästäkseen virtuaalikoneisiin käsiksi etäyhteydellä vaaditaan Extension Pack –laajennus.

5 Cuckoo Sandbox

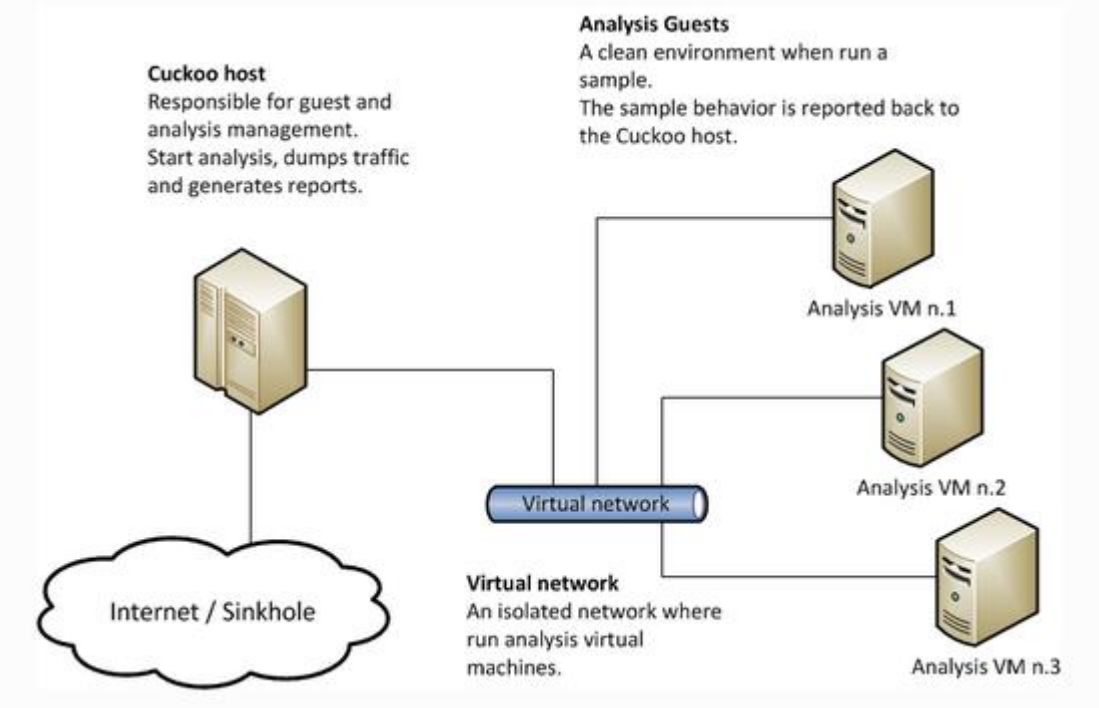
5.1 Yleistä

Cuckoo Sandbox on avoimen lahdekoodin automatisoitu haittaohjelmien analysointisovellus. Sen tehtävänä on analysoida epäilyttävät tiedostot eristetyssä sandbox-ympäristössä. Cuckoo sai alkunsa vuonna 2010 Google Summer Of Code-projektina, jonka jälkeen tuotetta on kehitetty tiiviisti aina nykyhetkeen saakka. Cuckoon avulla saadaan selville muun muassa haittaohjelman käynnistämät prosessit, haittaohjelman luodut, poistetut tai ladatut tiedostot, muistivedokset, verkkoliikenteen kaappaus ja kuvankaappaukset työasemalta. Cuckoo on erittäin modulaarinen ja muokattavissa. Sitä voidaan käyttää niin itsenäisesti kuin suuremmissakin ympäristöissä. Cuckoolla voidaan analysoida melkein mitä vain

tiedostotyyppejä suoritustiedostoista pdf-dokumentteihin. Cuckoolla on mahdollista analysoida haittaohjelmia Windows, Linux, OS X ja Android käyttöjärjestelmissä. (Cuckoo - What is Cuckoo 2016)

5.2 Arkkitehtuuri

Cuckoo Sandbox koostuu isäntäkoneesta ”Cuckoo Host” ja analyysikoneista ”Analysis Guest”. Host-palvelimelle asennetaan Cuckoon vaatimat komponentit ja se hallinnoi analyysiprosesseja. Palvelimelta on myös yhteys Internettiin. Isäntäkoneeseen asennetaan myös vapaavalintainen virtualisointisovellus, johon luodaan halutut guest-koneet. Guest-koneet ovat yhteydessä isäntäpalvelimeen virtuaalisen verkon ylitse. Jokainen guest on oma eristetty ympäristö, jossa voidaan suorittaa ja analysoida haittaohjelmia truvallisesti. Analyysit kuitenkin suoritetaan ja tulokset saadaan host-palvelimelta. Kuviossa 2 on havainnollistettu arkkitehtuurin rakennetta. (Cuckoo - Architechture 2016)



Kuvio 2. Cuckoo-arkkitehtuuri (Cuckoo - Architechture 2016)

5.3 CuckooDroid

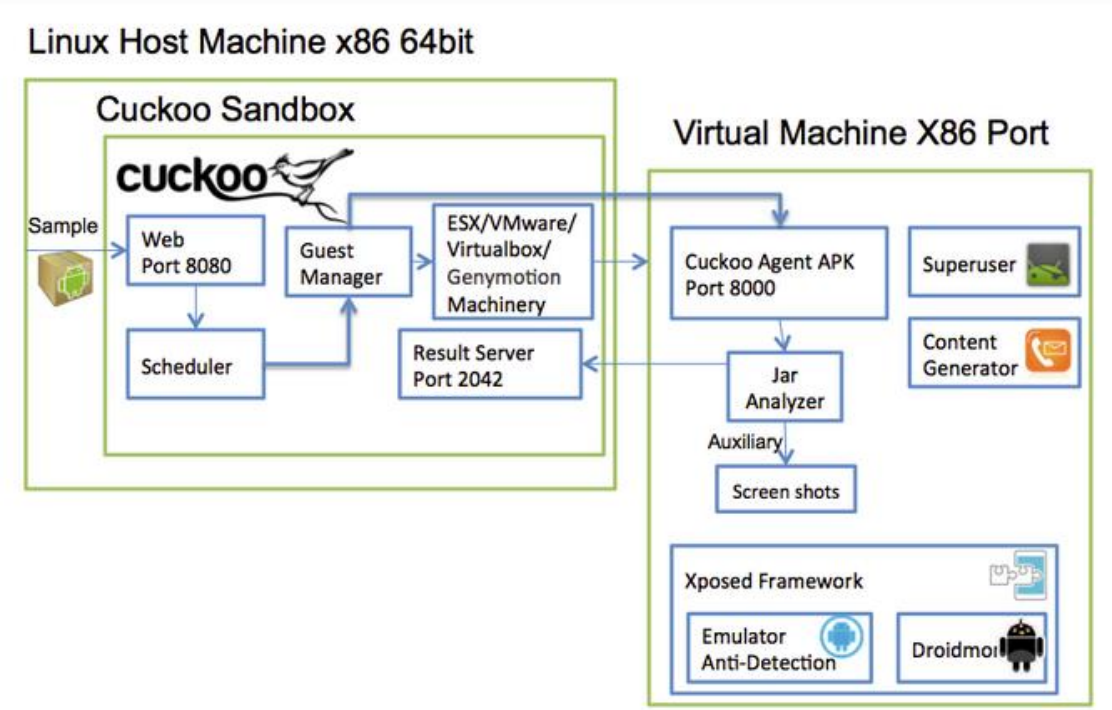
CuckooDroid on Cuckoo Sandboxin ympärille rakennettu laajennus. Sen avulla voidaan analysoida näytteitä myös Android-käyttöjärjestelmissä. CuckooDroid on kehitetty Cuckoo Sandbox version 1.2 laajennukseksi, joten täysin toiminnallisen CuckooDroidin pohjalla täytyy käyttää versiota 1.2 uuden 2.0 version sijasta. Cuckoon versioon 2.0 on kuitenkin sisällytetty Android analysointi Android Virtual Device eli AVD-tekniikalla CuckooDroidin pohjalta. Tässä toteutuksessa kuitenkin halutaan samaan järjestelmään Windows- ja Android-analyysikoneet, jolloin on käytettävä CuckooDroidia.

CuckooDroidissa on kolme tapaa toteuttaa Android-guest: Android on Linux Machine, Android Emulator tai Android Device Cross-platform.

Android on Linux Machine –ratkaisussa asennetaan Cuckoo-hostille virtuaalikone esimerkiksi Ubuntu, johon asennetaan Android emulaattori eli AVD. Cuckoossa guestiä ohjataan machinery-moduulilla ja host on siihen yhteydessä python agentilla Ubuntu virtuaalikoneen kautta. Tämän ratkaisun heikkous on sen raskaus virtualisoidussa ympäristössä. Vaatii toimiakseen OpenGL grafiikkakortin.

Android Emulator-ratkaisussa asennetaan AVD-emulaattori suoraan Cuckoo Host-palvelimelle. Eroavaisuutena toisiin toteutuksiin ohjataan tätä AVD-moduulilla, jonka vuoksi ei ole käytännöllistä pitää samassa järjestelmässä Androidin lisäksi muita analyysikoneita. AVD:hen ollaan yhteydessä myöskin python agentilla.

Viimeisessä ratkaisussa, Android Device Cross-platform, asennetaan hostille suoraan Android-virtuaalikone. Virtuaalikoneena toimii avoimenlähdekoodin Android käännös x86-alustoille Android-x86. Cuckoo ohjaa konetta machinery-moduulilla ja siihen ollaan yhteydessä APK agentin avulla. Tämä ratkaisu sopii parhaiten työn toteutukseen, joten se otetaan käyttöön. Kuviossa 3 on Android Device Cross-platform arkkitehtuuri avattuna. (CuckooDroid - Installation 2016)



Kuvio 3. Guest arkkitehtuuri (CuckooDroid - Guest Machine Architecture 2016)

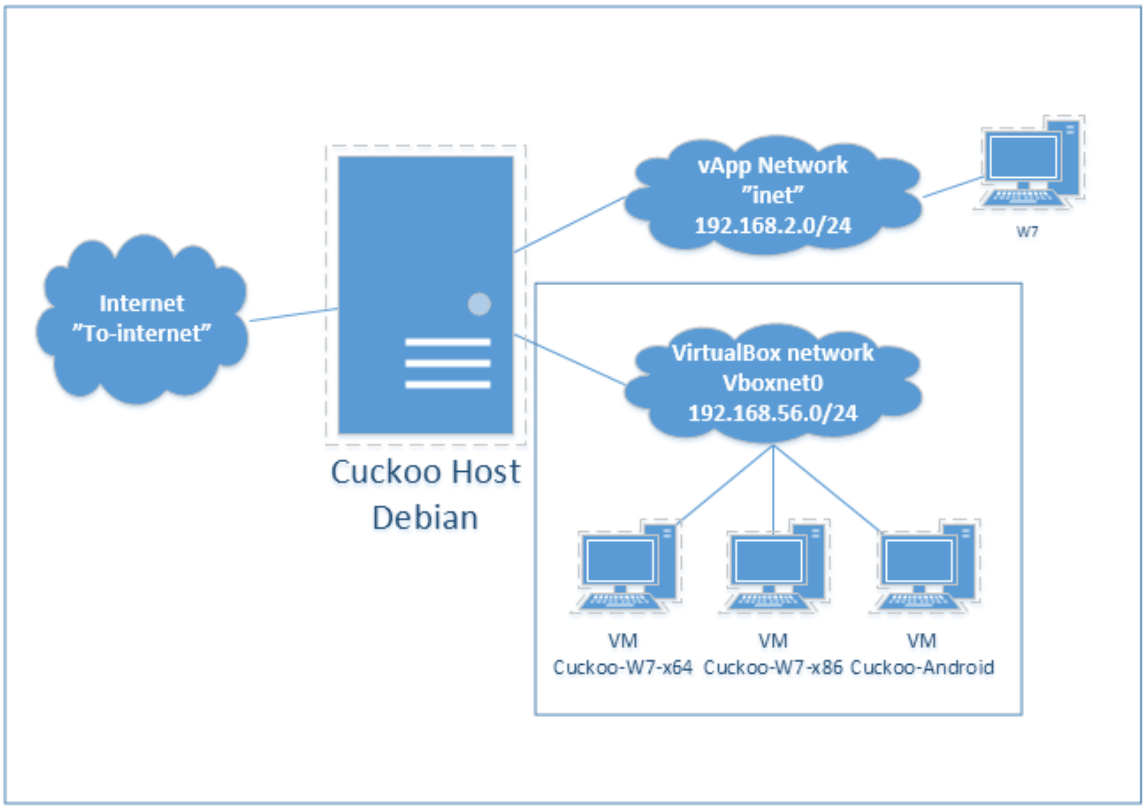
Superuser-sovelluksella myönnetään laitteeseen Superuser-oikeudet. Content Generator puolestaan generoi laitteeseen yhteystietoja luodakseen realistisemman vaikutelman.

Xposed Framework sovelluksen sisälle asennettavat moduulit voivat muuttaa järjestelmän ja sovellusten käyttäytymistä ilman vaikutusta APK-paketteihin. Xposedin sisäisistä moduuleista Droidmon tuottaa Cuckoo Droidin dynaamisen analyysin valvomalla sovellusten käyttäytymistä käyttöjärjestelmässä. Toinen Xposed moduuleista on Android Blue Pill, joka pyrkii ehkäisemään Android-emulaattorin paljastumista tunnetulla menetelmillä. Jar Analyzer komponentti tuodaan Androidille analyysin alkaessa. (CuckooDroid - Guest Machine Architecture 2016.)

6 Toteutus

6.1 Toteutusympäristö

Cuckoo Sandbox-järjestelmä toteutettiin JYVSECTECin Vmware vCloud-virtuaaliympäristöön. Ympäristöön tehtiin opinnäytetyötä varten oma vApp, jonka sisälle varsinaiset virtuaalikoneet asennettiin. Cuckoota varten luotiin 64-bittistä Linux Debian 8 (Jessie)-käyttöjärjestelmää pyörittävä palvelin ilman graafista käyttöliittymää. Etäkäyttöä varten luotiin Windows 7-työasema. vCloudista Internetiin pääsee käyttämällä ”To-Internet”-verkkoa rajapinnoissa. Lisäksi voidaan tehdä vAppin sisäinen aliverkko ”Inet”. Toteutuksessa käytetty ympäristö on kuvattu kuviossa 4.



Kuvio 4. Toteutusympäristö

Cuckoossa käytettävät virtuaalokoneet on listattu taulukossa 1.

Taulukko 1. Virtuaalikoneet

Nimi	Alusta	IP-osoite
Cuckoo-W7-x64	Windows 7 64-bit	192.168.56.10
Cuckoo-W7-x86	Windows 7 32-bit	192.168.56.11
Cuckoo-Android	Android-x86 4.4	192.168.56.12

6.2 Cuckoo-host

Alkuvalmisteluna palvelimelle lisättiin kaksi verkkokorttia, joista toinen oli kiinni Internetissä ja toinen vApp sisäverkossa. Täten on helppo asennusvaiheessa ladata tarvittavat tiedostot Internetistä ja testata toimintoja sisäverkosta. Rajapinta-asetukset Debianista saatiin asetettua tiedostossa */etc/network/interfaces* kuvion 5 mukaisesti.

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 192.168.2.100
        netmask 255.255.255.0
        gateway 192.168.2.1

auto eth1
allow-hotplug eth1
iface eth1 inet dhcp
```

Kuvio 5. Network interfaces

Tämän lisäksi lisättiin toinen kovalevy guest-virtuaalikoneita varten. Alustettiin levy fdisk-työkalulla ja mountattiin haluttuun paikkaan. Tässä tapauksessa */VM*.

Uusin tuotantoon soveltuva versio Cuckoosta on 2.0. Tässä toteutuksessa kuitenkin jouduttiin käyttämään versiota 1.2 CuckooDroid integraation vuoksi. Cuckoo Sandbox on saatavilla GitHubissa, joten sen lataamiseksi oli asennettava git-työkalu komennolla:

$ apt-get install git-core

Nyt voitiin Cuckoon tiedostot kloonata palvelimelle komennolla:

$ git clone git://github.com/cuckoosandbox/cuckoo.git cuckoo –b 1.2

Ennen kuin Cuckoo voitiin suorittaa, täytyi palvelimelle asentaa tarvittavat riippuvuudet.

6.2.1 Python kirjastot

Cuckoo on toteutettu python-ohjelmointikielellä, joten oli palvelimelle asennettava tarittavat python ja sen komponentit. Cuckoon Sandbox suosittelee käytettävän Python julkaisua 2.7. Python ja sen tarvittavat komponentit asennettiin komennolla:

$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev

Lisäksi Cuckoo-paketin mukana tuli kuvion 6. mukainen tiedosto requirements.txt, jonka avulla saatiin asennettua loput vaatimukset komennolla:

$ sudo pip install -r requirements.txt

```
  GNU nano 2.2.6

sqlalchemy
bson
jinja2
pymongo
bottle
pefile
django
chardet
nose
```

Kuvio 6. Requirements.txt

6.2.2 VirtualBox

Virtualisointityökalu on Cuckoo-järjestelmässä vapaavalintainen, mutta VirtualBox on ollut tuettuna pisimpään ja oletusvaihtoehto, minkä vuoksi sitä päädyttiin käyttämään myös tässä toteutuksessa.

Jotta VirtualBoxin lataaminen onnistui, täytyi tiedostoon */etc/apt/sources.list* lisätä seuraava rivi:

deb http://download.virtualbox.org/virtualbox/debian jessie contrib

Lisäksi pitää ladata ja rekisteröidä Oraclen julkinen avain seuraavalla komennolla:

$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add –

Itse ohjelmisto saadaan nyt asennettua seuraavilla komennoilla:

$ sudo apt-get update

$ sudo apt-get install virtualbox-5.0

Koska järjestelmä toteutetaan ilman graafista käyttöliittymää, tarvitaan VirtualBoxin laajennuspaketti RDP:tä eli etätyöpöytäkäyttöä varten. Laajennuksen lataaminen ja asentaminen onnistuu seuraavilla komennoilla:

$ wget
http://download.virtualbox.org/virtualbox/5.0.16/Oracle_VM_VirtualBox_Extension_Pack-5.0.16-105871.vbox-extpack
$ VBoxManage extpack install Oracle_VM_VirtualBox_Extension_Pack-5.0.16-105871.vbox-extpack

Lisäksi Cuckoo vaatii virtuaalisen rajapinnan vboxnet0 host- ja guest-koneiden välille. Oletuksena IP-osoite on 192.168.56.1/24. Rajapinta saadaan luotua VBoxManagen avulla seuraavasti:

$ VBoxManage hostonlyif create

$ ip link set vboxnet0 up

$ ip addr add 192.168.56.1/24 dev vboxnet0

Nyt rajapinta näkyy Debianissa kuvion 7 mukaisesti.

```
vboxnet0  Link encap:Ethernet  HWaddr 0a:00:27:00:00:00
          inet addr:192.168.56.1  Bcast:0.0.0.0  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Kuvio 7. ifconfig vboxnet0

Jotta rajapinta nousee pystyyn järjestelmän käynnistymisen yhteydessä, lisätään tiedostoon */etc/rc.local* seuraava rivi ennen riviä exit 0:

$ VboxManage hostonlyif ipconfig vboxnet0 –ip 192.168.56.1

6.2.3 TCPDUMP

Verkkoliikennettä analysoimaan Cuckoo vaatii verkkoliikennettä kaappaavan snifferin. Oletuksena Cuckoo käyttää Tcpdump-työkalua, kuten käytettiin myös tässä järjestelmässä. Tcpdump asennetaan yksinkertaisuudessaan komennolla

$ Sudo apt-get install tcpdump

Oletuksena tcpdump vaatii root-oikeudet sen suorittamiseen. Koska Cuckoota ei suositella ajettavan roottina, täytyy binääriin tehdä suraava muutos:

$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump

6.2.4 Volatility

Cuckoossa Windowsin muistivedosten prosessoimiseksi asennetaan Volatility Framework. Volatility asennetaan seuraavalla tavalla:

$ wget downloads.volatilityfoundation.org/releases/2.4/volatility-2.4.tar.gz

$ tar –zfx volatility-2.4.tar.gz

$ cd volatility-2.4

$ python setup.py build

$ python setup.py install

Jos Volatility halutaan käyttöön, täytyy Cuckoon konfiguraatiotiedostosta *conf/cuckoo.conf* ottaa käyttöön *memory dump.* Jokaisesta analyysistä taltiodaan täten memory dump-tiedosto raporttia varten. Valinta voidaan myös tehdä analyysiä suorittaessa. Tiedostosta *conf/processing.conf* on otettava käyttöön moduuli *memory*.

Kun käytössä on VirtualBoxin versio 5.0 ja Cuckoo Sandboxista 1.2, on virtualbox.py-moduulia muokattava kuvion 8 mukaisesti. Syynä muokkaukseen on VirtualBoxin muuttunut komento memory dumpin taltioimiseksi. Tiedosto virtualbox.py löytyy hakemistosta *cuckoo/modules/machinery/virtualbox.py.* Rivillä 206 komento

”dumpguestcore” korvataan komennolla ”dumpvmcore”. Tämä on päivitetty Cuckoon versiossa 2.0.

```
199
200     def dump_memory(self, label, path):
201         """Takes a memory dump.
202         @param path: path to where to store the memory dump.
203         """
204         try:
205             subprocess.call([self.options.virtualbox.path, "debugvm",
206                              label, "dumpvmcore", "--filename", path],
207                             stdout=subprocess.PIPE,
208                             stderr=subprocess.PIPE)
209             log.info("Successfully generated memory dump for virtual machine "
210                      "with label %s to path %s", label, path)
211         except OSError as e:
212             raise CuckooMachineError("VBoxManage failed to take a memory "
213                                      "dump of the machine with label %s: %s" %
214                                      (label, e))
```

Kuvio 8. virtualbox.py

Volatilityn asetukset löytyvät tiedostosta *cuckoo/conf/memory.conf,* johon asetetaan oikea käyttöjärjestelmäprofiili. Tässä tapauksessa profiili asetettiin seuraavasti:

guest_profile = Win7SP1x86

Volatility tarvitsee toimiakseen toimiakseen Distorm3 dissambler-kirjaston, joka asennettiin seuraavasti:

$ wget https://distorm.googlecode.com/files/distorm3.zip

$ unzip distorm3.zip

$ cd distorm3/

$ python setup.py build

$ python setup.py install

6.2.5 Mongodb

Graafista selainpohjaista käyttöliittymää varten täytyy asentaa MongoDB seuraavasti:

$ sudo apt-get install mongodb

Käyttöliittymä on Django-pohjainen, ja se hakee tarvittavat tiedot tietokannasta, joka on määritetty tiedostoon *conf/reporting.conf.* Kun tietokanta asennetaan

paikallisesti, ei tiedostoon vaadita muuta kuin ottaa MongoDB käyttöön kuvion 9. mukaisesti.

```
[mongodb]
enabled = yes
host = 127.0.0.1
port = 27017
db = cuckoo
```

Kuvio 9. reporting.conf

6.2.6 Yara

Yara on valinnainen lisäosa Cuckoo-järjestelmään. Yara-työkalun tehtävänä on auttaa järjestelmää tunnistamaan ja luokittelemaan haittaohjelmanäytteitä. Yara asennetaan seuraavasti:

$ wget https://github.com/plusvic/yara/archive/v3.4.0.tar.gz

$ tar -zxf yara-3.1.0.tar.gz

$ cd yara-3.1.0

$ ./bootstrap.sh

$ ./configure

$ make

$ sudo make install

$ ./configure --enable-cuckoo

Lisäksi asennetaan yara-python laajennus seuraavasti:

$ cd yara-python

$ python setup.py build

$ sudo python setup.py install

Yaraan voidaan luoda sääntöjä, joiden perusteella yara pyrkii luokittelemaan haittaohjelmat. Säännöt löytyvät hakemistosta */cuckoo/data/yara/binaries.* Hakemisto sisältää .yar päätteisiä tiedostoja, jotka sisältävät liitteen 1 mukaisen

koodin. Cuckoossa on oletuksena muutama yara-sääntö, mutta osaamisen riittäessä niitä voi luoda itse tai kehittää olemassa olevia. Liitteen 1 esimerkissä yara pyrkii löytämään analysoitavasta haittaohjelma viitteitä virtuaalokoneen havaitsemiseen. Yara-sääntöön on listattu tunnettuja virtualisoinni havaitsemiseen käytettyjä merkkijonoja. Jos yksikin merkkijono osuu haittaohjelman sisältöön, laukaisee yara ilmoituksen vmdetect.

6.2.7 Ssdeep

Fuzzy Hashing-sovellus Ssdeep ladataan ja asennetaan seuraavilla komennoilla:

$ wget http://sourceforge.net/projects/ssdeep/files/ssdeep-2.12/ssdeep-2.12.tar.gz

$ tar -xvzf ssdeep-2.12.tar.gz

$ cd ssdeep-2.12/

$ ./configure && make && make install

Tämän lisäksi tarvitaan Python-laajennus Pydeep. Tämän asennus onnistuu seuraavasti:

$ git clone https://github.com/kbandla/pydeep

$ cd pydeep

$ python setup.py build

$ python setup.py install

6.3 Cuckoo-guest

6.3.1 Virtuaalikoneen luominen

Työasemat, joilla haittaohjelma-analyysit ajetaan, virtualisoitiin Cuckoo-palvelimelle VirtualBoxin avulla. Virtuaalikone voidaan luoda alusta alkaen itse tai voidaan järjestelmään tuoda (import) valmiita virtuaalikoneita. Tässä tapauksessa käytettiin JYVSECTECiltä saatuja Windows 64- ja 32-bittisiä templateja. VBoxManagen avulla virtuaalikoneen tuominen tapahtuu helposti. Ensin ajetaan komento:

$ VBoxManage import -n template.ovf

Varmistetaan myös, että tulosteen ”target path” on hakemisto, johon virtuaalikone halutaan tuoda. Kyseisessä hakemistossa tulee olla tarpeeksi tilaa varattuna ja oikeudet Cuckoo-käyttäjällä, sekä vboxusers-ryhmällä. Tämä ei vielä luo virtuaalikonetta vaan tulostaa sen tiedot, jotta koneen asetuksia voidaan tarvittaessa muokata jo tässä vaiheessa. Seuraava komento tuo koneen järjestelmään ja nimeää sen Cuckoo-guestiksi:

$ VBoxManage import temlate.ovf --vsys 0 --vmname Cuckoo-guest

Jotta virtuaalikone löytyy VirtualBox:n listoilla, täytyy se rekisteröidä ilmoittamalla .vbox tiedoston polku komennolla:

$ *VBoxManage registervm /VM/cuckoo-w7-x64/ cuckoo-w7-x64.vbox*

Lisäksi otetaan käyttöön verkkokortti aikaisemmin luodulle rajapinnalle vboxnet0 komennolla:

$ VboxManage modifyvm ”Koneen_nimi” –nic1 hostonly –hostonlyadapter1 vboxnet0

Jotta virtuaalikoneeseen pääsee käsiksi, täytyy siihen asettaa vrde-toiminto käyttöön komennolla:

$ VboxManage modifyvm ”Koneen_nimi” –vrde on

Nyt virtuaalikone voidaan käynnistää komennolla:

$ VboxHeadless –startvm ”Koneen_nimi”

Nyt työasemaan pääsee käsiksi saman aliverkon työasemilta. Niihin voidaan ottaa etätyöpöytäyhteys esimerkiksi Remote Desktop-sovelluksella Cuckoo-palvelimen IP-osoitteella. Oletuksena vrde kuuntelee porttia 3389. Komennon perään lisäämällä &-merkin, jää prosessi pyörimään taustalle.

Jotta virtuaalikoneilla on yhteys Internettiin, täytyy Cuckoo-palvelimen välittää IP-paketit rajapintojen välillä. Tämä saadaan käyttöön lisäämällä tiedostoon */etc/sysctl.conf* rivi:

net.ipv4.ip_forward = 1

Lisäksi palvelimen palomuurin on sallittava rajapintojen välinen liikenne. Tarvittavat säännöt iptables:iin lisätään seuraavasti:

iptables -A FORWARD -o eth1 -i vboxnet0 -s 192.168.56.0/24 -m conntrack --ctstate NEW -j ACCEPT

iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -A POSTROUTING -t nat -j MASQUERADE

Tässä tapauksessa eth1 on ulkoverkon rajapinta ja vboxnet0 virtuaalinen rajapinta. Jotta säännöt tulevat käyttöön järjestelmän käynnistyksen yhteydessä, täytyy ne tallentaa komennolla:

$ iptables-save > /etc/iptables.up.rules

Lisäksi luodaan tiedosto */etc/network/if-pre-up.d/iptables* ja lisätään siihen seuraavat rivit:

#!/bin/sh

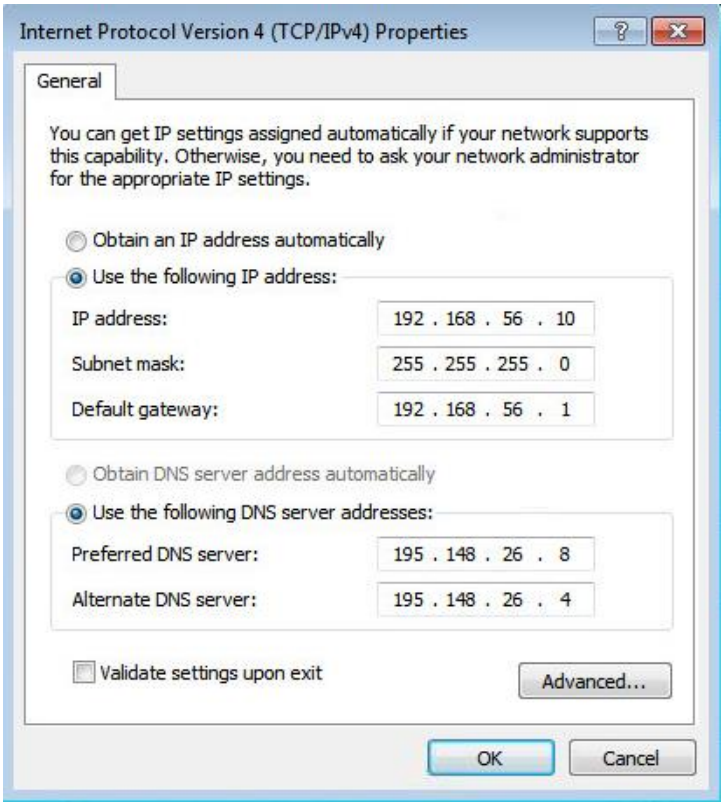
/sbin/iptables-restore < /etc/iptables.up.rules

Muutetaan vielä tiedoston oikeudet seuraavasti:

chmod +x /etc/network/if-pre-up.d/iptables

6.3.2 Windows-guest

Etäkäyttöohjelmiston kautta Windows-virtuaalikoneelle asetetaan ensimmäiseksi verkko-osoitteet manuaalisesti esimerkiksi kuvion 10 mukaisesti. Eli IP-osoiteeksi asetetaan osoite vboxnet0-aliverkosta ja oletusyhdyskäytäväksi Cuckoo-palvelimen IP-osoite. Lisäksi asetetaan toimivat DNS-osoitteet.

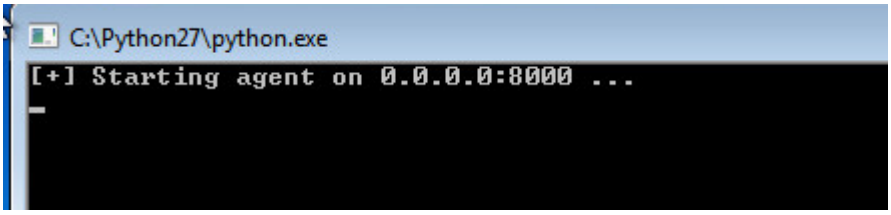


Kuvio 10. Verkko-osoitteet

Cuckoon agent toimii Pythonin avulla, joten ladataan Pythonin viralliselta sivulta Windowsille asennuspaketti ja asennetaan se. Suositeltavaa on käyttää versiota 2.7. Windowsista on myös otettava palomuuri, sekä automaattiset päivitykset kokonaan pois käytöstä.

Jotta työasemalta saadaan analyysistä kuvankaappauksia, asennetaan guestille myös Python Image Library eli PIL osoitteesta: http://www.pythonware.com/products/pil/. Versio valitaan Python version mukaisesti eli tässä tapauksessa 2.7.

Cuckoo-palvelimelta hakemistossa *cuckoo/agent* sijaitsee agent.py-tiedosto. Tämä tiedosto täytyy kopioida virtuaalikoneelle esimerkiksi luomalla väliaikaisen jaetun kansion työasemien välille tai scp:llä. Kun agentti suoritetaan guest-koneella, avautuu kuvion 11 mukainen ikkuna.



Kuvio 11. Cuckoo agent

Agentti kuuntelee porttia 8000 ja sen toiminta voidaan testata ottamalla palvelimelta telnet-yhteyden seuraavasti:

$ telnet 192.168.56.10 8000

Jos yhteyden muodostaminen onnistuu, on agentin asentaminen onnistunut.

6.3.3 Virtuaalikoneen tallentaminen

Kun kaikki seuraavat muutokset guest-koneelle on tehty, tallennetaan koneen tila VirtutalBoxiin snapshotin avulla seuraavasti.

$ VBoxManage snapshot " Koneen_nimi”take "snapshotin_nimi" –pause

$ VBoxManage controlvm " Koneen_nimi >" poweroff

$ VBoxManage snapshot " Koneen_nimi “restorecurrent

Täten virtuaalikone käynnistyy aina samaan tilaan agent käynnissä ja tarvittavat muutokset tehtynä.

6.3.4 Virtualbox.conf

Cuckoon VirtualBox konfiguraatiot sijaitsevat tiedostossa *cuckoo/conf/virtualbox.conf*. Tiedostossa ilmoitetaan, että VirtualBoxia käytetään headless modessa seuraavasti:

mode=headless

Asetetaan myös Cuckoo käyttämään aiemmin luotua virtuaalista verkkorajapintaa vboxnet0:llaa valinnalla:

interface=vboxnet0

Lisäksi tiedostoon listataan kaikki käytössä olevat analyysejä suorittavat virtuaalikoneet seuraavasti:

machines=Koneen_nimi1,Koneen_nimi2

Tämän lisäksi jokaisesta listatusta koneesta ilmoitetaan sen nimi VirtualBoxissa, käyttöjärjestelmä ja IP-osoite kuvion 12 mukaisesti.

```
machines = cuckoo-w7-x64,cuckoo-w7-x86

[cuckoo-w7-x86]
label=cuckoo-w7-x86
platform=windows
ip=192.168.56.11

[cuckoo-w7-x64]
```

Kuvio 12. virtualbox.conf

Tiedostoon on kommentoitu tarkemmat ohjeustukset tietojen täyttämiseen.

6.4 CuckooDroid

Androidin haittaohjelmia analysoidakseen on Cuckoo Sandboxiin asennettava CuckooDroid –laajennus. CuckooDroidin integroiminen järjestelmään on yksinkertaista. Ensimmäisenä haetaan tarvittavat tiedostot git:n avulla seuraavasti:

$ git remote add droid https://github.com/idanr1986/cuckoo-droid

$ git pull --no-edit -s recursive -X theirs droid master

Tämän jälkeen lisätään Cuckoon konfiguraatiotiedostoihin *processing.conf* ja *reporting.conf* kuvioiden 13 ja 14 mukaiset rivit seuraavilla komennoilla:

$ cat conf-extra/processing.conf >> conf/processing.conf

$ cat conf-extra/reporting.conf >> conf/reporting.conf

```
  GNU nano 2.2.6           File: processing.conf


[apkinfo]
enabled = yes
#Decompiling dex with androguard in a heavy operation and for a big dex's
#he can really consume performance from the cuckoo host ,so it's recommended to$
#decompilation_threshold=2000000

[droidmon]
enabled = yes

[googleplay]
enabled = no
android_id =
google_login =
google_password =
```

Kuvio 13. processing.conf

Kuvio 14. reporting.conf

Lisäksi lisätään *requirements.txt* tiedostoon rivi *protobuf,*

$ echo "protobuf" >> requirements.txt

ja ajetaan tiedoston muutokset:

$ sudo pip install -r requirements.txt

6.5 Android Guest

Toteutuksessa käytetään Linuxin päällä pyörivää Android x86 porttausta. Analysointitarkoituksessa käytetään versiota *Android x86 4.4 RC2* Cuckoo yhteensopivuuden vuoksi. Asennusmedia .iso –tiedostolla on kaikkien saatavilla virallisilla Android-x86 sivuilla: *http://www.android-x86.org*.

Cuckoo-palvelimelle luodaan VirtualBoxilla uusi virtuaalikone Liitteen 2. ohjeiden mukaisesti. Käyttöjärjestelmä asennetaan normaalisti kovalevylle asennusohjeita seuraamalla. Ennen kuin työasemaan päästään käsiksi täytyy verkkoasetukset asettaa manuaalisesti taustalla pyörivään Linuxiin. Linux-terminaali saadaan näkyviin näppäinyhdistelmällä: ALT+CTRL+F1. Verkkoasetukset saadaan muutettua muokkaamalla */etc/init.sh* –tiedostoa, johon päästään käsiksi seuraavasti:

$ su

$ vi /etc/init.sh

Tiedostoon lisätään rivit ennen riviä ”return 0” kuvion 15 osoittamalla tavalla.

```
                ;;
esac
ifconfig eth0 192.168.56.13 netmask 255.255.255.0 up
route add default gw 192.168.56.1 dev eth0
ndc resolver setifdns eth0 195.148.26.8 195.148.26.4
ndc resolver setdefaultif eth0


return 0
I /etc/init.sh [Modified] 367/367 100%
```

Kuvio 15. /etc/init.sh

Vi-editorista poistutaan tallentaen muutokset, painamalla ESC ja kirjoittamalla :wq (write and quit). Tämän jälkeen käynnistetään kone uudelleen, ja sen pitäisi muodostaa Internet-yhteys.

Nyt siirrytään taas Cuckoo-palvelimelle, jolle asennetaan sovellus Android Debug Bridge eli adb:

$ sudo apt-get install android-tools-adb

Adb:n avulla saadaan siirrettyä tiedostoja ja sovelluksia palvelimen ja android-laitteen välillä. Yhteys laitteeseen muodostetaan roottina seuraavasti:

$ adb connect 192.168.56.13

$ adb root

Hakemistossa *utils/android_emulator_creator* on skripti *create_guest_device.sh*, joka asentaa android-guestiin tarvittavat sovellukset: Xposed, Droidmon, Anti Emulator Detection, Content Generator ja Cuckoo Agent. Skripti ajetaan seuraavasti:

$ ./utils/android_emulator_creator/create_guest_device.sh

Lisäksi Droidmonia varten luodaan konfiguraatiotiedosto menemällä Cuckoon hakemistoon *cuckoo/utils/hooks_generator.* Hakemistossa suoritetaan seuraava komento:

$ python generate_hook_json.py

Komento tuottaa hakemistoon tiedoston *hooks.json*, joka viedään Android-virtuaalikoneen hakemistoon */data/local/tmp* seuraavalla komennolla:

*$ adb push */cuckoo/utils/hooks_generator/hooks.json /data/local/tmp*

Nyt voidaan taas avata android-virtuaalikone, johon muutetaan seuraavat asetukset:

Näytön lukitus pois käytöstä: *settings->security->screenlock->none*

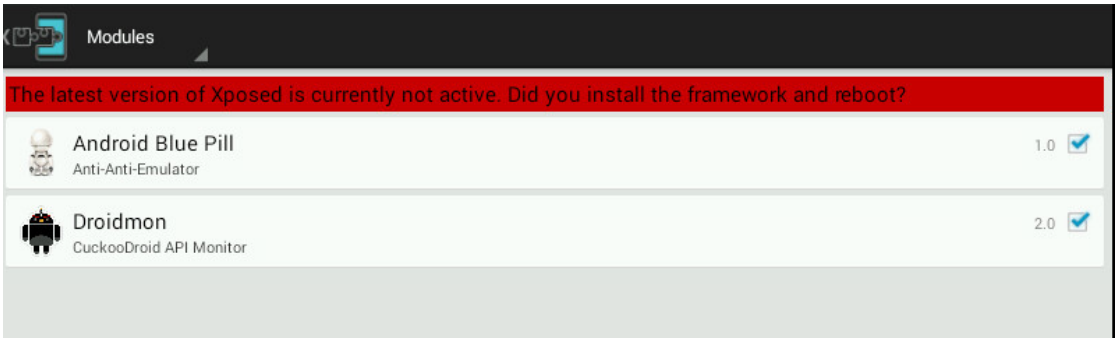
Näytön aikakatkaisu pois käytöstä: *settings->security->screenlock->none*

Sovellusten turvallisuusasetukset: Ruksi pois kohdasta*: settings->security-> verify apps* ja ruksi kohtaan: *settings->security->Unknown sources*

Seuraavaksi generoidaan yhteystietoja laitteeseen, käynnistämällä sovellus **Generate Contacts.**

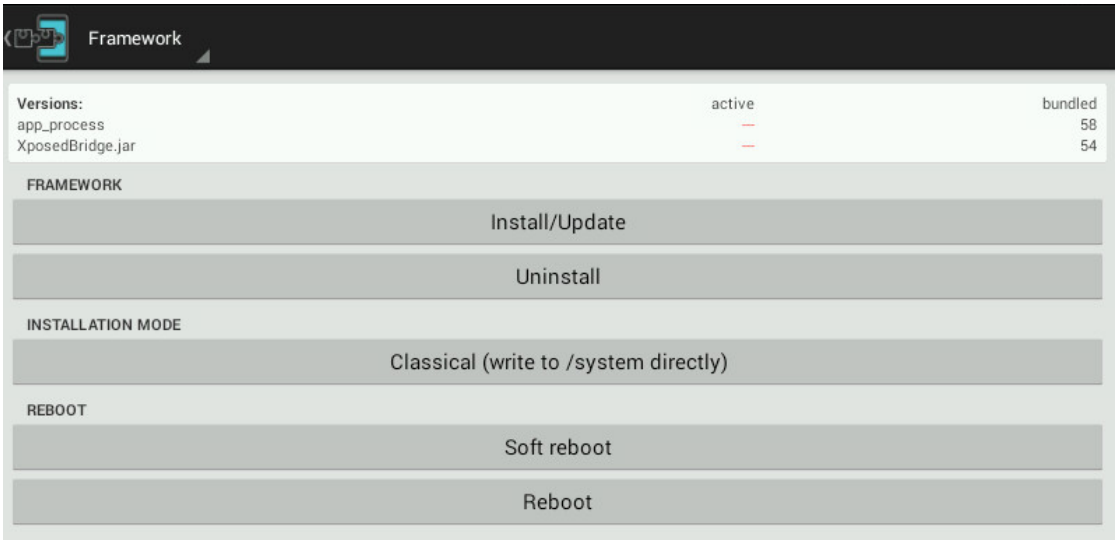
Superuser –sovelluksesta sallitaan automaattiset vastaukset ja otetaan ilmoitukset pois käytöstä. *Settings-> automatic response -> allow* ja *-> notification None*

Seuraavaksi avataan **xposedinstaller**-sovellus, josta kuvion 16 mukaisesti valitaan molemmat moduulit: Droidmon ja Andoid Blue Pill.



Kuvio 16. xposed modules

Tämän jälkeen avataan kuvion 17 mukainen ikkuna Xposed Framework ja asennetaan se valitsemalla *Install/update.* Kun asennus on valmis, ei käynnistetä laitetta uudelleen heti, vaan valitaan *Soft reboot.*



Kuvio 17. Xposed Framework

Laitteen uudelleenkäynnistyttyä avataan sovellus **cuckooAgent** taustalle. Kaiken ollessa kunnossa voidaan virtuaalikoneen tila tallentaa luvun 6.3.3 mukaisesti.

Nyt täytyy virtuaalikone lisätä vielä Cuckoon analyysikoneiden listalle tiedostoon *conf/virtualbox.conf* kuvion 18 osoittamalla tavalla.

```
  GNU nano 2.2.6          File: conf/virtualbox.conf

# Default network interface.
interface = vboxnet0

# Specify a comma-separated list of available machines
# specified ID you have to define a dedicated section c
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuck
machines = cuckoo-w7-x64,cuckoo-w7-x86,android

[android]
label=android
platform=android_device
ip=192.168.56.13
snapshot=snapshot-droid
interface=vboxnet0
resultserver_ip=192.168.56.1
resultserver_port=2042
```

Kuvio 18. virtualbox.conf

6.6 Analyysin suorittaminen

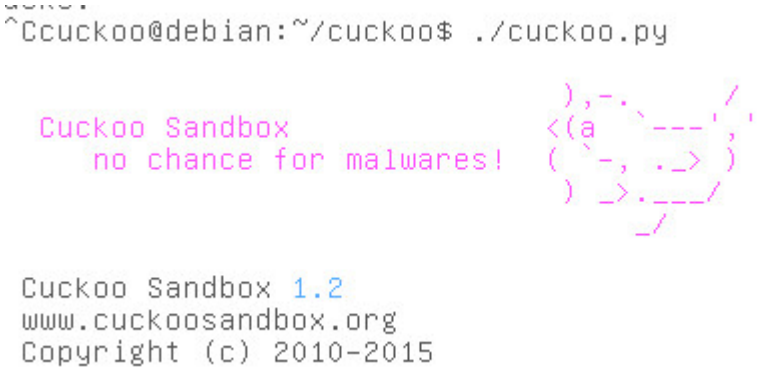
6.6.1 Yleistä

Cuckoolla on mahdollista ajaa analyysi monella tapaa. Yksi vaihtoehto on ajaa analyysi paikallisesti palvelimelta ja lukea HTML-raportti selaimella. Toinen tapa ja

tähän toteutekseen sopivampi on lähiverkosta Web-käyttöliittymän avulla suoritettava analyysi. Ensimmäisenä on joka tapauksessa käynnistettävä Cuckoo Sandbox palvelimelta kuvion 17. mukaisesti komennolla:

$ *./cuckoo.py &*

Cuckoo tarkistaa käynnistettäessä, ovatko järjestelmän päivitykset ajan tasalla. Se lataa myös käytettävissä olevat analyysikoneet ja jää odottamaan tulevaa tehtävää.



Kuvio 19. Cuckoo.py

Hyödyllinen komento on myös Cuckoon käynnistäminen Debug-modessa. Debug-moden avulla nähdään konsolista analyysin eteneminen tarkemmin, ja se on myös suuri apu ongelmien ratkaisussa esimerkiksi virtuaalikoneongelmissa. Käynnistys tapahtuu seuraavalla komennolla:

$ *./cuckoo.py –d*

Toinen hyödyllinen komento on kaikkien jo suoritettujen analyysien poistaminen järjestelmästä. Se tapahtuu seuraavalla komennolla:

$ ./cuckoo.py --clean

Komennolla poistetaan suoritettujen analyysien kaikki tallennettu tieto tiedostoista ja tietokannoista.

6.6.2 Komentokehotteelta

Analyysi voidaan käynnistää myös suoraan palvelimen terminaalista. Hakemistosta *cuckoo/utils* löytyy tiedosto *submit.py.* Tämän avulla analyysi voidaan ajaa seuraavasti:

$ ./submit.py näytteen/polku.exe

Voidaan tehdä myös lisävalintoja analyysin suorittamiseen esimerkiksi:

-- memory – memory dump

--platform windows – analyysi suoritetaan windows-koneella

--machine cuckoo-w7-x86 – analyysi suoritetaan halutulla koneella

--priority 5 – analyysin prioriteetti

--timeout 60 – analyysin aikakatkaisu

Analyysin raportti tallennetaan hakemistoon *cuckoo/storage/analysis/id.*

6.6.3 Web-käyttöliittymässä

Cuckoon django-pohjainen web-käyttöliittymä saadaan käynnistettyä komennolla:

$ python manage.py runserver 0.0.0.0:8080

Tällöin web-palvelin käyttää asetuksia tiedostossa *web.settings* ja kuuntelee porttia 8080 kuvion 20 mukaisesti.

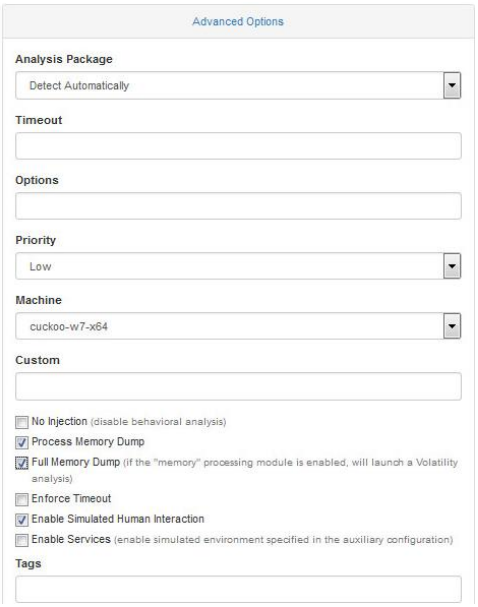
```
cuckoo@debian:~/cuckoo/web$ Performing system checks...

System check identified no issues (0 silenced).
March 29, 2016 - 19:09:05
Django version 1.8.4, using settings 'web.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
```

Kuvio 20. runserver

Analyysin suuorittaminen on käyttöliittymän kautta yksinkertaista. *Submit*-välilehdeltä voidaan valita paikalliselta koneelta tai URL-osoitteesta analysoitavan tiedoston kuvion 21 mukaisesti. Analyysi käynnistyy valitsemlla *Analyze.*

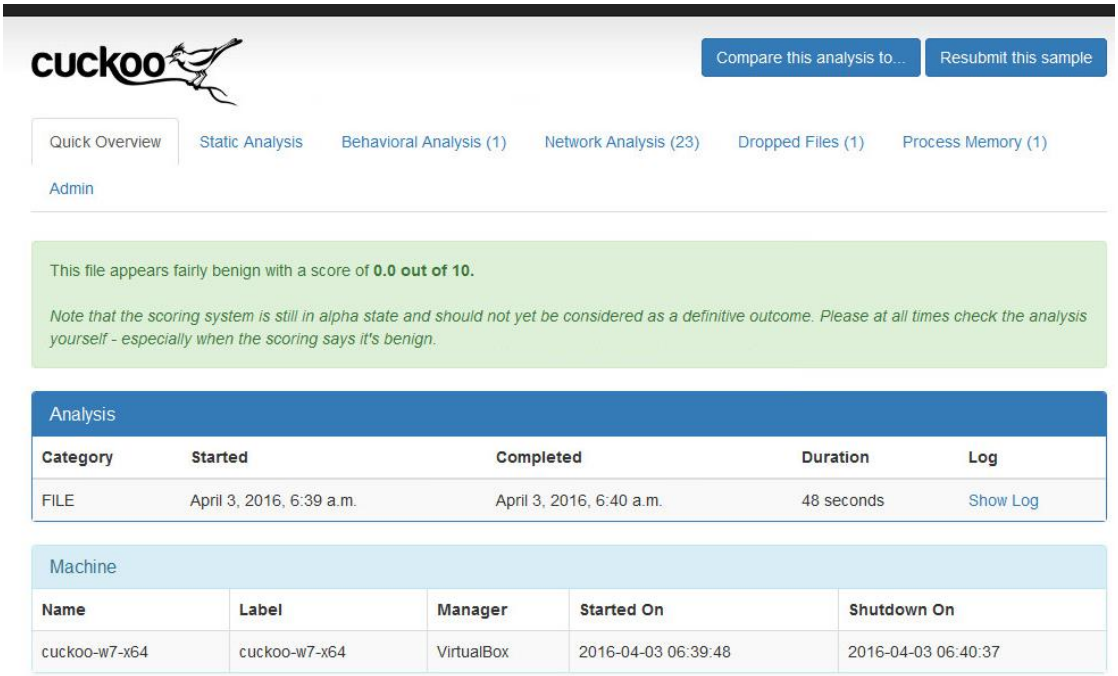
Kuvio 21. Web-Submit

Lisäasetuksia löytyy *Advanced Options* alta. Asetuksissa voidaan muun muassa vaikuttaa analyysin sisältöön, analyysin prioriteettiin, suorittavaan koneeseen ja aikakatkaisun pituuteen. Kuviossa 22 kaikki asetukset.



Kuvio 22. Advanced options

Kun analyysi on valmis, voidaan valmiita raportteja selata *recent* –välilehdeltä. Kaikki analyysit on myös jälkikäteen haettavissa *search-välilehdeltä*. Oletuksena haku tapahtuu tiivistefunktioilla.



Kuvio 23. Raportti

Analyysejä voidaan hakea myös kuvion 24 osoittamilla tunnisteilla asettamalla hakuun oikean prefixin.

Prefix	Description
name:	File name pattern
type:	File type/format
string:	String contained in the binary
ssdeep:	Fuzzy hash
crc32:	CRC32 hash
imphash:	Search for PE Imphash
file:	Open files matching the pattern
key:	Open registry keys matching the pattern
mutex:	Open mutexes matching the pattern
ip:	Contact the specified IP address
domain:	Contact the specified domain
url:	Search for Cuckoo Sandbox URL analysis
signature:	Search for Cuckoo Sandbox signatures

Kuvio 24. Search prefix

6.7 Signatures

Cuckooseen voidaan luoda signatureja eli python koodinpätkiä, jotka laukaisevat eritasoisen hälytyksen raportista saatavien tietojen perusteella. Signatureja tulee oletuksena asennuspaketin mukana ja niitä on saatavilla aktiiviselta yhteisöltä. Niitä voidaan myös koodata itse, jos pythonin perusteet on hallussa. Signaturet ovat Cuckoossa hakemistossa *cuckoo/modules/signatures.* Signaturejen avulla voidaan nopeasti raporttia vilkaisemalla saada selville haittaohjelman käyttäytyminen tai esimerkiksi luokitella näytteitä niiden perusteella. Liitteessä 3 on esimerkki, jossa säännöllä tarkistetaan luoko haittaohjelmanäyte Windows-työasemaan exe-tiedoston. Jos exe-tiedosto havaitaan raportista laukaisee sääntö vakavuudeltaan tason 2 ilmoituksen. Vakavuuden taso näkyy ilmoituksessa sen värinä.

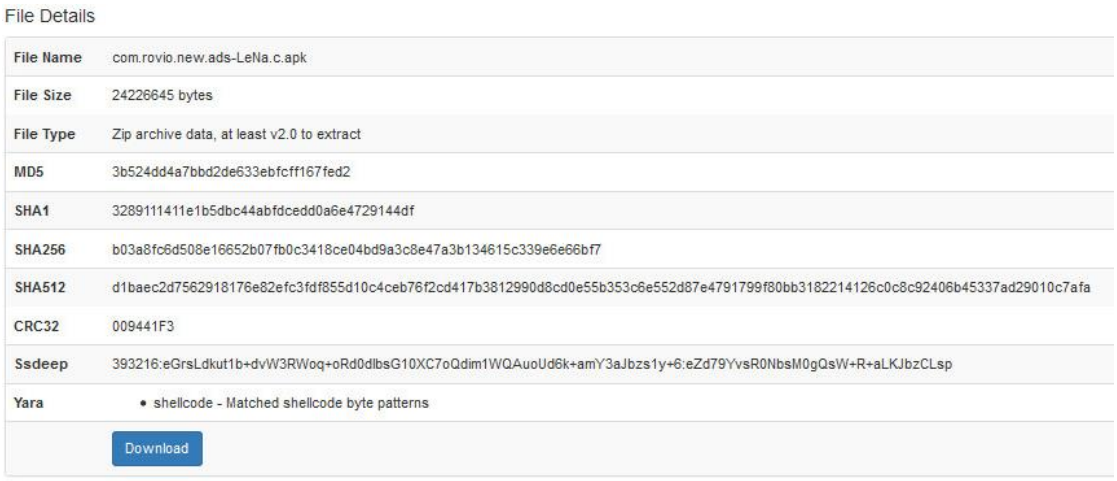
7 Haittaohjelma-analyysi

7.1 Raportti

Suoritetusta analyysistä Cuckoo Sandbox tuottaa mittavan raportin, josta olisi osattava poimia oikea tieto mahdollisesta haittaohjelmasta. CuckooDroidin ja Cuckoo Sandboxin raportit eroavat toisistaan hiukan. Molemmista raporteista kuitenkin löytyvät staattinen, dynaaminen ja verkkoanalyysi

7.2 File Details

Ensimmäisenä raportista löytyy analysoidun tiedoston tiedot. Tiedostosta selviää kuvion 25 mukaiset tiedot. Tiedostosta on muunnettu tiivistefunktiot MD5, SHA1, SHA256, SHA512 ja CRC32 tiivistealgoritmeillä. Lisäksi Ssdeep-sovelluksella muodostettu fuzzy hash. Funktioita käytetään haittaohjelman tunnistamiseen tietokannoista. Yara-ilmoittaa mahdollisista osumista yara-sääntöihin. Analysoitu tiedosto on myös mahdollista ladata jälkikäteen järjestelmästä. Näillä kaikilla tunnisteilla voidaan raportti hakea järjestelmän haku-välilehdeltä myöhemmin.

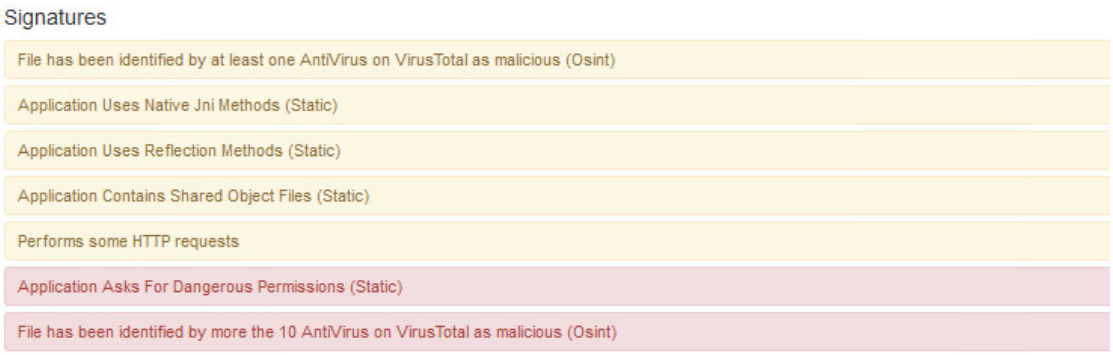

File Details

File Name	com.rovio.new.ads-LeNa.c.apk
File Size	24226645 bytes
File Type	Zip archive data, at least v2.0 to extract
MD5	3b524dd4a7bbd2de633ebfcff167fed2
SHA1	3289111411e1b5dbc44abfdcedd0a6e4729144df
SHA256	b03a8fc6d508e16652b07fb0c3418ce04bd9a3c8e47a3b134615c339e6e66bf7
SHA512	d1baec2d7562918176e82efc3fdf855d10c4ceb76f2cd417b3812990d8cd0e55b353c6e552d87e4791799f80bb3182214126c0c8c92406b45337ad29010c7afa
CRC32	009441F3
Ssdeep	393216:eGrsLdkut1b+dvW3RWoq+oRd0dlbsG10XC7oQdim1WQAuoUd6k+amY3aJbzs1y+6:eZd79YvsR0NbsM0gQsW+R+aLKJbzCLsp
Yara	• shellcode - Matched shellcode byte patterns
	Download

Kuvio 25. File Details

7.3 Signatures

Seuraavana raportin etusivulta löytyy Signatures-ilmoitukset. Näistä voidaan nopeasti tulkita haittaohjelman toimia. Vakavat ilmoitukset ovat väriltään punaisia ja vähemmän vakavat huomiot keltaisia, kuten kuviosta 26 havaitaan. Signaturet voivat kertoa haittaohjelman toiminnasta, kuten HTTP-pyynnöistä tai sen vaatimista oikeuksista. Signaturen voi myös laukaista esimerkiksi VirusTotalin osumien määrä. Kuvion 26 signaturet ovat Troijalaisesta haittaohjelmasta Android-käyttöjärjestelmälle.


Signatures

File has been identified by at least one AntiVirus on VirusTotal as malicious (Osint)

Application Uses Native Jni Methods (Static)

Application Uses Reflection Methods (Static)

Application Contains Shared Object Files (Static)

Performs some HTTP requests

Application Asks For Dangerous Permissions (Static)

File has been identified by more the 10 AntiVirus on VirusTotal as malicious (Osint)

Kuvio 26. Signatures

7.4 Summary

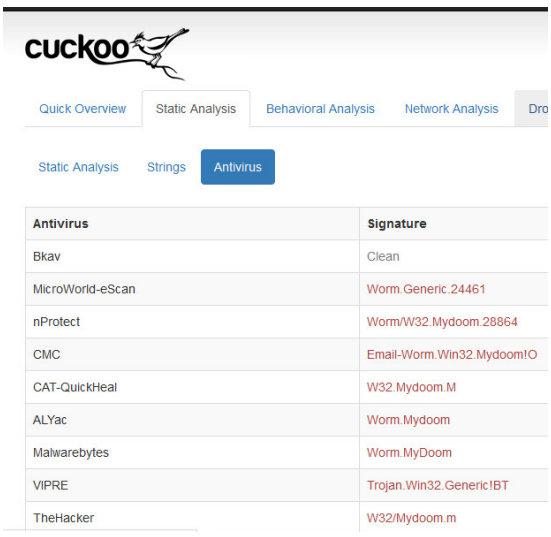
Quick Overview-välilehdeltä Summary-kohdasta nähdään analyysin aikana muokatut tiedostot ja rekisteriavaimet, joista voidaan jäljittää haittaohjelman toimintaa. Lisäksi nähdään luodut mutexit. Haittaohjelmat käyttävät mutexeja varmistaakseen, että

vain yksi haittaohjelmainstanssi on ajossa samaan aikaan. Kuviossa 27 esimerkki summary-kohdan erään haittaohjelman mutex-listasta.

Kuvio 27. Summary

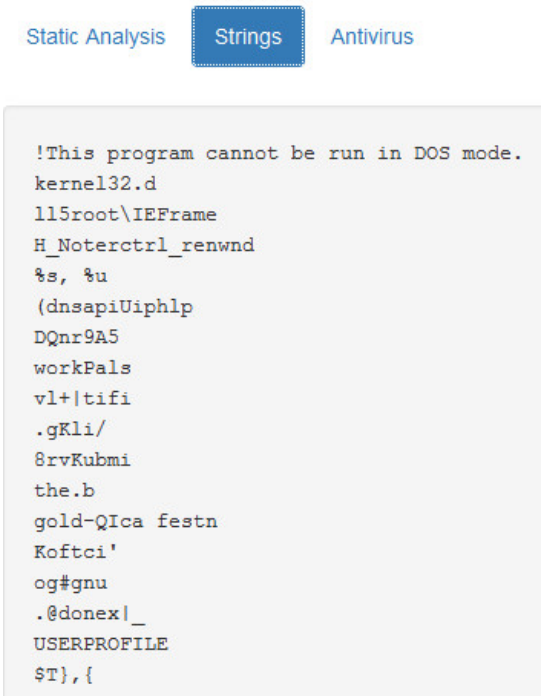
7.5 Static analysis

Static analysis-välilehdeltä löytyy näytteen staattinen analyysi. Cuckoon raporttiin on integroitu VirusTotal.com-palvelun tulokset. Jokainen näyte etsitään palvelun tietokannasta tiivistefunktioiden avulla. Näytettä ei siis uploadata palveluun, joten tuloksia löytyy vain, jos haittaohjelma löytyy jo tietokannasta. Virustotalissa näyte on skannattu 56:n eri tunnetun valmistajan antivirus-ohjelmistossa, joiden tulokset näkyvät raportissa. Kuvion 28 esimerkissä 53/56 Virustotalin skanneista toteaa näytteen haitalliseksi. Haittaohjelma on esimerkissä tyypiltään mato nimeltään ”mydoom”.



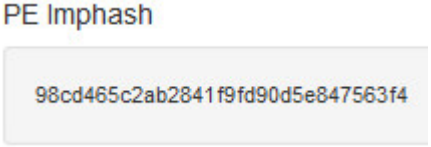
Kuvio 28. Virustotal

Strings-välilehdeltä löytyy kaikki yli 4-merkkiä pitkät ASCII-merkkijonot analysoidusta binääristä. Jonoista voidaan saada vihiä haittaohjelman toiminnasta. Jonoja tulostuu paljon ja suurin osa ei kerro näytteestä mitään. Niistä täytyisi osata haravoida tärkeä tieto. Kuviossa 29 esimerkki haittaohjelman merkkijonoista. Kuvion 29 esimerkistä voidaan päätellä tiedoston olevan pakattu, koska luettavissa olevia jonoja ei juuri löydy.



Kuvio 29. Strings

Static Analysis-välilehdeltä saadaan Windowsin PE-tiedostoista analyysin kannalta hyödyllistä tietoa sen headeristä. Tiedoston imhash eli import hash koostuu tiedoston kirjastojen nimistä ja niiden järjestyksestä. Imphashin avulla voidaan tunnistaa ja luokitella haittaohjelmanäytteitä. Kuviossa 30 esimerkki erään tiedoston Imphashistä.



Kuvio 30. Imphash

Seuraavaksi raportista löytyy PE-tiedoston osiot eli sectionit, joista voidaan saada hyödyllistä tietoa. Esimerkiksi kuvion 31 tiedoston osiosta .rsrc voidaan päätellä sen sisältävän exe-tiedoston vaatimia resursseja, kuten kuvia, ikoneita, valikoita tai

merkkijonoja. UPX-nimet viittaavat pakkaajaan nimeltä UPX Packer. Lisäksi, jos Virtual Size on paljon suurempi kuin Raw Data, vie se enemmän tilaa keskusmuistista kuin levyltä. Lisäksi Resources-taulukko näyttää tiedostoon sisällytetyt resurssit. (Sikorski & Honig 2012.)

Sections

Name	Virtual Address	Virtual Size	Size of Raw Data	Entropy
UPX0	0x00001000	0x00008000	0x00000000	0.0
UPX1	0x00009000	0x00006000	0x00006000	7.85908669132
.rsrc	0x0000f000	0x00001000	0x00000800	2.6542421842

Resources

Name	Offset	Size	Language	Sub-language	File type
RT_ICON	0x0000f3c4	0x00000128	LANG_ENGLISH	SUBLANG_ENGLISH_US	GLS_BINARY_LSB_FIRST
RT_ICON	0x0000f3c4	0x00000128	LANG_ENGLISH	SUBLANG_ENGLISH_US	GLS_BINARY_LSB_FIRST
RT_GROUP_ICON	0x0000f4f0	0x00000022	LANG_ENGLISH	SUBLANG_ENGLISH_US	MS Windows icon resource - 2 icons, 32x32, 16 colors

Kuvio 31. Sections ja Recources

Kuvion 32 mukaisesta Import-luettelosta nähdään haittaohjelman käyttämiä funktioita muista kirjastoista. Funktioista voidaan päätellä haittaohjelman käytöstä. Tässä tapauksessa voidaan todeta haittaohjelman olevan pakattu, koska importteja on huomattavan vähän.

Imports

Library KERNEL32.DLL:
- 0x50f58c LoadLibraryA
- 0x50f590 GetProcAddress
- 0x50f594 ExitProcess

Library ADVAPI32.dll:
- 0x50f59c RegCloseKey

Library MSVCRT.dll:
- 0x50f5a4 memset

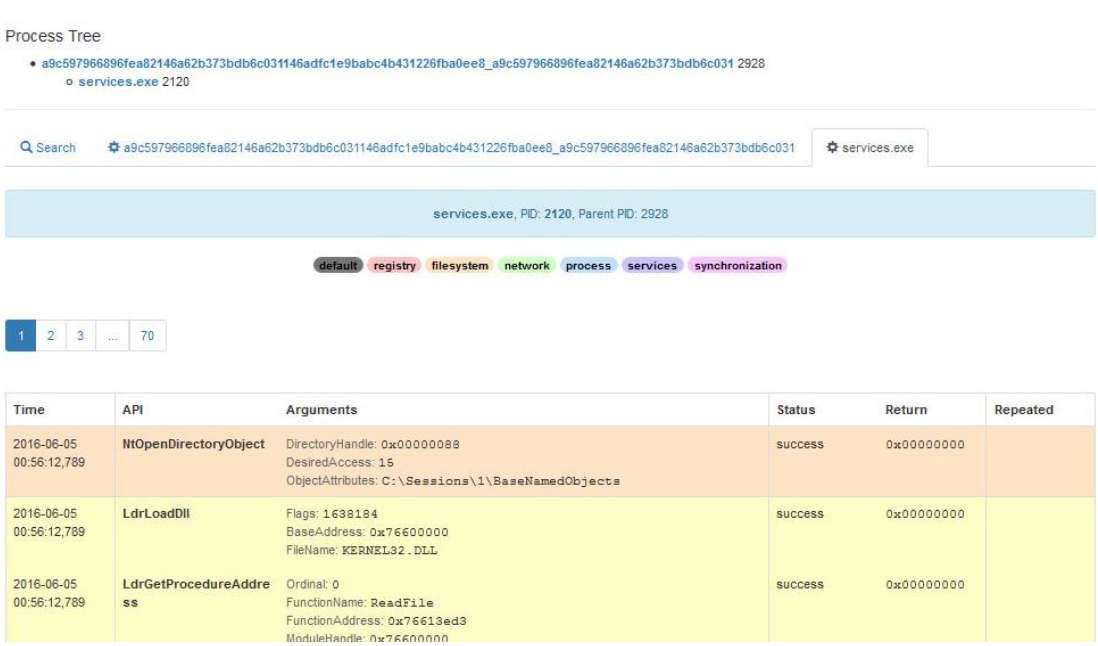
Library USER32.dll:
- 0x50f5ac wsprintfA

Library WS2_32.dll:
- 0x50f5b4 gethostname

Kuvio 32. Imports

7.6 Behavioral Analysis

Behavioral Analysis-välilehdeltä saadaan tietoja haittaohjelmanäytteen suorittamista toimenpitestä analyysikoneessa. Cuckoo monitoroi laitteen muutoksia tiedostojärjestelmässä ja rekistereissä. Myöskin monitoroidaan luotuja prosesseja ja palveluita järjestelmässä. Ensimmäisenä raportissa nähdään prosessipuu, joka havainnollistaa haittohjelman luomat prosessit. Kuvion 33 mukaisesti voidaan halutusta prosessista suodattaa toiminnot hakusanalla tai toiminnon tyypin perusteella.



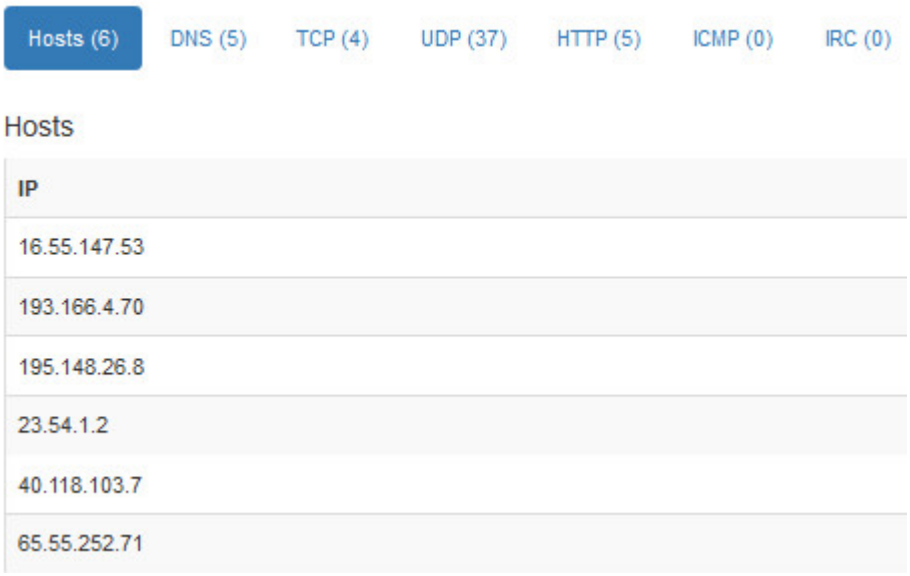
Kuvio 33. Process tree

7.7 Network Analysis

Nework analysis-välilehdeltä voidaan tarkastella analyysin aikana tapahtunutta verkkoliikennettä. Verkkoliikenne kaapataan Tcpdump-sovelluksella vboxnet-rajapinnasta. Raportista voidaan ladata koko pcap-tiedosto ja tarkastella esimerkiksi wireshark-sovelluksessa. Cuckoon raportti kuitenkin suodattaa oleellisen tiedon verkkoliikenteestä.

Liikenteestä saadaan tieto siitä, että mihin osoitteisiin tietokone on ollut yhteydessä analyysin aikana. Raportista nähdään myös DNS-kyselyt, UDP- ja TCP-yhteydet

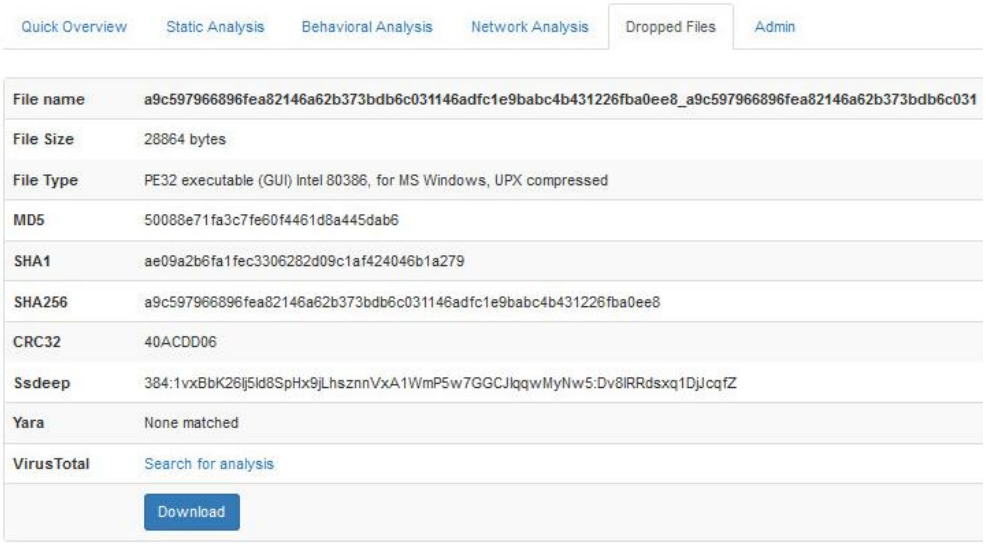
portteineen, HTTP-liikenteen oleellinen sisältö, ICMP- ja IRC-liikenne. Kuviossa 34 on havainnollistettu esimerkkihaittaohjelman verkkoanalyysi.



Kuvio 34. Network analysis

7.8 Dropped Files

Dropped Files-välilehdeltä nähdään kaikki mahdollisesti haittaohjelman luomat tiedostot analyysin aikana. Kaikki tiedostot on listattu raporttiin ja niistä nähdään kuvion 35 mukaiset tiedot jokaisesta. Lisäksi kaikista voidaan helposti tehdä haku VirusTotaliin.

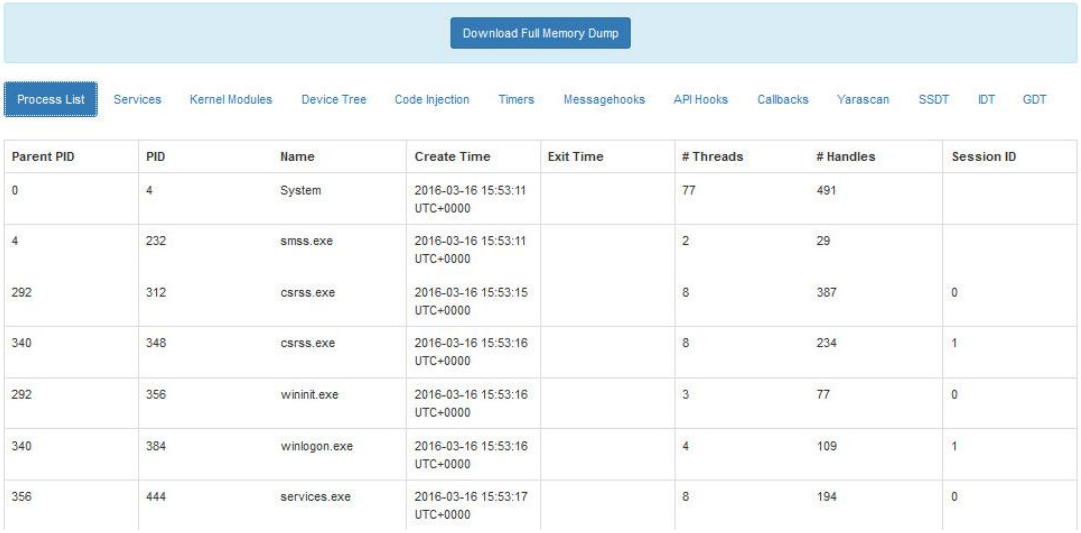


Kuvio 35. Dropped Files

7.9 Memory analysis

Memory analysis-välilehti on raportissa, jos haittaohjelma näytteestä on haluttu muistivedoksen analyysi. Analyysi tehdään Volatility-ohjelmistolla. Tässä toteutuksessa Volatility toimii vain 32-bittisessä Windows 7-käyttöjärjestelmässä. Volatilityn ajettavia moduuleja voi ottaa käyttöön ja pois käytöstä tiedostosta *conf/memory.conf.* Jos kaikki moduulit on käytössä, voi Volatility-raportin muodostamisessa kestää yli 30 minuuttia.

Raportista nähdään analyysin aikaiset prosessit ja palvelut käyttöjärjestelmässä. Raportista nähdään kaikkien Volatility moduulien tulokset kuvion 36 mukaisesti.



Parent PID	PID	Name	Create Time	Exit Time	# Threads	# Handles	Session ID
0	4	System	2016-03-16 15:53:11 UTC+0000		77	491	
4	232	smss.exe	2016-03-16 15:53:11 UTC+0000		2	29	
292	312	csrss.exe	2016-03-16 15:53:15 UTC+0000		8	387	0
340	348	csrss.exe	2016-03-16 15:53:16 UTC+0000		8	234	1
292	356	wininit.exe	2016-03-16 15:53:16 UTC+0000		3	77	0
340	384	winlogon.exe	2016-03-16 15:53:16 UTC+0000		4	109	1
356	444	services.exe	2016-03-16 15:53:17 UTC+0000		8	194	0

Kuvio 36. Memory analysis

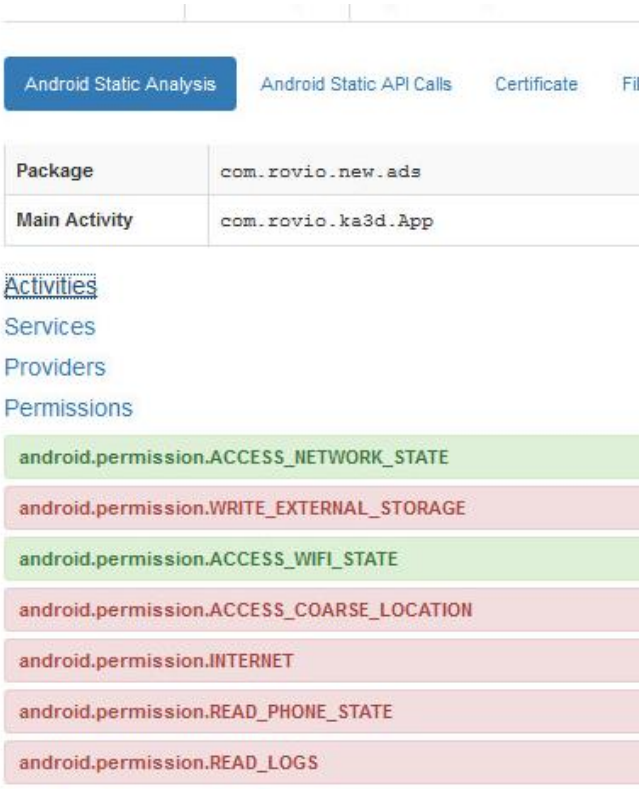
7.10 CuckooDroid-raportti

7.10.1 Static Analysis

CuckooDroid-raportissa Static Analysis-välilehdeltä VirusTotalin tulosten ja merkkijonojen lisäksi nähdään APK-paketin sisältämät tiedostot, sertifikaatti, staattiset API-kutsut ja APK-paketin staattinen analyysi. Kuviossa 37 lista esimerkkitiedostosta kerätyistä API Call-tyypeistä.

Kuvio 37. Android API Calls

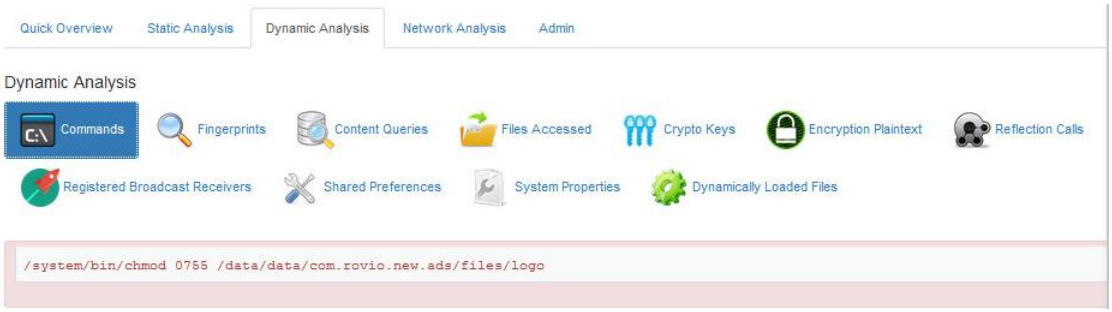
Sovelluksen staattinen analyysi näyttää kuvion 38 mukaisesti tiedot kerätyistä muun muassa sovelluksen toiminnoista, palveluista ja vaatimista oikeuksista.



Kuvio 38. Android Static Analysis

7.10.2 Dynamic Analysis

Droidmon seuraa Android-käyttöjärjestelmän Dalvik API-kutsuja, joista saadaan selville tietoja sovelluksen toiminnasta. Kuvion 39 mukaisesti raportista nähdään esimerkiksi komentorivikomennot, sormenjäljet, salausavaimet, käytetyt tiedostot ja järjestelmän asetukset.



Kuvio 39. Dynamic Analysis

8 Testaus

8.1 Testi 1

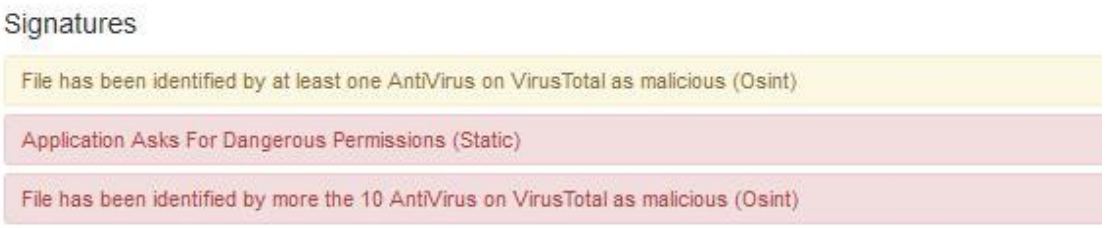
Ensimmäisessä testissä haittaohjelmanäyte oli apk-tiedosto ja tartutettava käyttöjärjestelmä Android 4.4. Android haittaohjelmia analysoitaessa on syytä käyttää CuckooDroid web-käyttöliittymää hakemistossa *cuckoo/web_android.* Kuviosta 40 nähdään raportoidut tiedoston tiedot.

File Details

File Name	fake angry.apk
File Size	74310 bytes
File Type	Java archive data (JAR)
MD5	394dc498f9ee2e61fb1959bebe1da2b4
SHA1	eefb761be6dce5afd4086f0e149166f7578d59b1
SHA256	d63857461a281f32233b548b411227d2318ae85ac3695d600391d0aa0ac63b6d
SHA512	75bbf00774850aa627967ce52ad9079ee9d442e350747443a6131db85949076320b4ca2de5ef637fca109673bf9a01f3e493455eed97447747ead3330a2d3ebf
CRC32	D21CFEA9
Ssdeep	1536:xTX+9Z5SAijeMPKTCZcsbNsgv9biZ+iqfsCSRtFwQYfYEUSPl0FC:xTqEjRCCZclGNiZ+icsCSRnwQ5EUSPlD
Yara	None matched
	Download

Kuvio 40. Testi 1 file details

Signatureista voitiin kuvion 41 mukaisesti heti päätellä, että näyte on tunnistettu haitalliseksi yli kymmenessä skannissa VirusTotalissa. Sovellus myös kysyy epäilyttäviä oikeuksia.



Kuvio 41. Testi 1 signatures

VirusTotalin mukaan näyte on troijalainen haittaohjelma Android-käyttöjärjestelmiin. Se kulkee nimellä FakeAngry. Osa VirusTotalin tuloksista kuviossa 42.

Ad-Aware	Android.Trojan.FakeAngry.B
Sophos	Andr/Fakengry-A
Comodo	UnclassifiedMalware
F-Secure	Riskware:Android/FakeAngry
DrWeb	Android.Anzhu
Zillya	Trojan.Agent..480
TrendMicro	AndroidOS_FAKEAPP.A
McAfee-GW-Edition	Artemis!Trojan
Emsisoft	Android.Trojan.FakeAngry.B (B)
Cyren	AndroidOS/GenBl.394DC498!Olympus
Jiangmin	Backdoor/AndroidOS.cac
Avira	ANDROID/FakeAngry.A.1
Antiy-AVL	Trojan[Backdoor]/AndroidOS.Fakengry.a
Microsoft	Backdoor:AndroidOS/Fakengry.A
Arcabit	Android.Trojan.FakeAngry.B

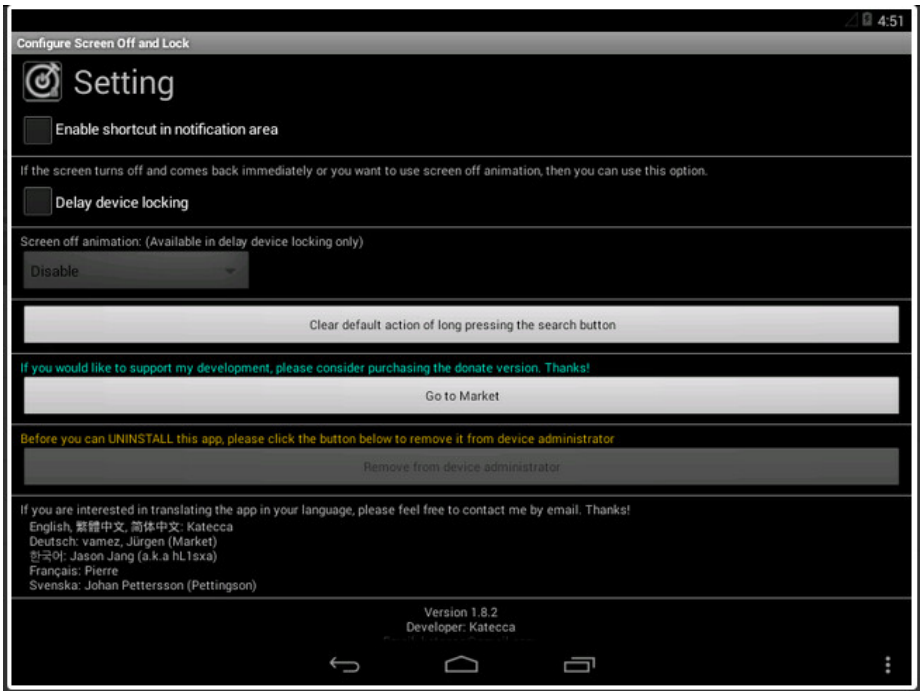
Kuvio 42. FakeAngry

Haittaohjelma pyytää kuvion 43 mukaisesti oikeuksia käyttöjärjestelmään. Käynnössä haittaohjelmalla on täten oikeus saada selville laitteen tiedot ja käyttäjän henkilökohtaisia tietoja. Haittaohjelma voi kirjoittaa SD-kortille, muodostaa verkkoyhteyksiä, lukea ja muokata selaimen historiaa.

Application Asks For Dangerous Permissions (Static)	
android.permission.INTERNET	Allows an application to create network sockets.
android.permission.READ_PHONE_STATE	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.WRITE_EXTERNAL_STORAGE	Allows an application to write to the SD card.
com.android.browser.permission.READ_HISTORY_BOOKMARKS	Allows the application to read all the URLs that the browser has visited and all of the browser's bookmarks.
com.android.browser.permission.WRITE_HISTORY_BOOKMARKS	Allows an application to modify the browser's history or bookmarks stored on your phone. Malicious applications can use this to erase or modify your browser's data.
android.permission.READ_LOGS	Allows an application to read from the system's various log files. This allows it to discover general information about what you are doing with the phone, potentially including personal or private information.

Kuvio 43. Dangerous permissions

Käyttäjän tietoja siis urkitaan ja lähetetään eteenpäin laitteesta. Käyttäjälle sovelluksesta avautuu vain kuvion 44 mukainen Screen Off and Lock-näkymä.



Kuvio 44. Screenofflock

8.2 Testi 2

Tosessa testissä haittaohjelmanäyte on exe-tiedosto ja tartutettavana käyttöjärjestelmänä 32-bittinen Windows 7. Analysoidun tiedoston tiedot ovat nähtävissä kuviossa 45.

File Details

File Name	48738b9c81e004ec68c59889904a4bfe.exe
File Size	262063 bytes
File Type	Zip archive data, at least v2.0 to extract
MD5	b5e444fa55829efb00d26319f9ec987d
SHA1	e1c10e2881b19407e03b7cb1bea2d5cba2c0347f
SHA256	ef604c3e1b855c12c1280f35fa2f57c910f40fa088ea21d3463b3bbd38ad2e3a
SHA512	91d87b81c6f7f5c337662b0a69514ef76cf980241b1964cd1aadc47de9fc11803c2f14267ac46b650e42d129f165a91edafb34845d62cdbc1b2308b7e186018e
CRC32	8CC2F7A1
Ssdeep	6144:mvT5nn/EcF8w+ICBJf1l1wrTxeJfLVy1l6Q8lpD:mbecB+lqfn1UUFLV8z4R
Yara	None matched
	Download

Kuvio 45. Testi 2 file details

Kuvion 46 mukaisesti ohjelma laukaisi vain yhden signaturen.

Signatures

File has been identified by at least one AntiVirus on VirusTotal as malicious (Osint)

Kuvio 46. Testi 2 signatures

Signaturen mukaisesti vain yksi VirusTotalin antivirus tuntee haittaohjelman. Kuvion 47 mukaisesti NANO-Antivirus kertoo tiedoston olevan troijalainen haittaohjelma Windowsille ”Llac.dsnuug”.

NANO-Antivirus	Trojan.Win32.Llac.dsnuug

Kuvio 47. Testi 2 VirusTotal

Haittaohjelma on staattisen analyysiä vastaan hyvin toteutettu, eikä siitä saada irti tietoja.

Kuviossa 48 esitettävän prosessipuun mukaan haittaohjelma luo käyttöjärjestelmään prosessin *remote.exe* haittaohjelman suoritettaessa, mutta juuri muuta hyödyllistä ei dynaamisestakaan analyysistä saada irti.

Kuvio 48. Testi 2 Process Tree

Remote.exe-tiedostoa analysoidessa Cuckoo 2.0 versiolla hälytti yara merkeistä virtuaalikoneen havaitsemisesta. Näissä tapauksissa analyysistä on vaikea saada mitään irti ilman syvällisempää tulkintaa.

9 Pohdinta

Opinnäytetyön tavoitteet oli toteuttaa, dokumentoida ja testata Cuckoo Sandbox haittaohjelmien analysointijärjestelmä. Lisäksi järjestelmään oli tavoitteena integroida CuckooDroid-laajennus Android-käyttöjärjestelmiä varten. Järjestelmään saatiin toteutettua tarvittavat ominaisuudet, joten tavoitteet sen osalta täyttyi. Cuckoo Droid-integraation vuoksi jouduin käyttämään Cuckoo Sandboxista versiota 1.2 uuden 2.0 sijaan, minkä vuoksi kaikkia Cuckoon ominaisuuksia ei saatu käyttöön. Mielestäni se oli kuitenkin paras ratkaisu saada Windows- ja Android-käyttöjärjestelmät samaan analysointijärjestelmään.

Cuckoo Sandbox-järjestelmän pystyttäminen on kohtalaisen mutkatonta, jos on osaamista Linux-palvelimista ja virtualisoinnista. Silti asennuksessa on turhan paljon vaiheita ja se vie aikaa. Python-kielen osaaminen tulee tarpeen järjestelmää mukauttaessa. Koska kyseessä on Open Source-tuote, meni ongelmatilanteissa paljon aikaa ongelman selvittämiseen osin puutteellisen dokumentaation vuoksi. Etenkin raporttien sisällöstä ja tulkinnasta on heikosti saatavilla dokumentaatiota. Käyttäjälle analyysin suorittaminen on erittäin helppoa selkeän web-käyttöliittymän avulla. Tulosten tulkitseminen raportista vaatii kuitenkin haittaohjelma-analyysin teorian tuntemusta. CuckooDroid-raportista saa mielestäni enemmän irti ja se on helppolukuisempi. Syvemmälle mentäessä on kuitenkin oltava tietoa APK-tiedostojen yleisestä toiminnasta Android-käyttöjärjestelmässä. Raportista saa kuitenkin nopeasti käsityksen tiedoston haitallisuudesta ja mahdollisista toiminnoista signaturejen ja VirusTotalin avulla.

Näkisinkin järjestelmän järkevimmäksi käyttötarkoitukseksi epäilyttävistä lähteistä tuotujen tiedostojen tarkastamisen haittaohjelmien varalta. Jos järjestelmä otetaan käyttöön isommassa ympäristössä, voidaan siihen kerätä omaa tietokantaa haittaohjelmista. Järjestelmää voidaan myös muokata tietyn tyyppisten tiedostojen analysoimiseen. Järjestelmän toteutus palvelimelle graafisella käyttöliittymällä voisi toimia paremmin etenkin, jos tarkoitus on muokata paljon analyysikoneita tai, jos halutaan käyttää Androidia emulaattorin avulla. Jatkossa voidaan järjestelmää myös muokata käyttötarkoituksen mukaan esimerkiksi kehittämällä tarkemmin kohdennettuja signatureja. Järjestelmä on sellaisenaan otettavissa käyttöön esimerkiksi RGCE-ympäristöön tarvittaessa.

Lähteet

Cuckoo. 2016. Cuckoo Sandbox Book. Introduction: Architecture. Viitattu 11.4.2016. http://docs.cuckoosandbox.org/en/latest/introduction/what/#architecture

Cuckoo. 2016. Cuckoo Sandbox Book. Introduction: What is Cuckoo?. Viitattu 11.4.2016. http://docs.cuckoosandbox.org/en/latest/introduction/what/

Cuckoo Droid. 2016. CuckooDroid Book. Guest Machine Architecture. Viitattu 26.4.2016. http://cuckoo-droid.readthedocs.org/en/latest/installation/guest_android_device/architecture/

Cuckoo Droid. 2016 CuckooDroid Book. Installation. Viitattu 26.4.2014. http://cuckoo-droid.readthedocs.io/en/latest/installation/

Dunham, K. 2013. A Fuzzy Future in Malware Research. ISSA. Viitattu 11.4.2016. https://c.ymcdn.com/sites/www.issa.org/resource/resmgr/journalpdfs/fuzzyhash-issa-journal0813.pdf

JYVSECTEC. 2016. JYVSECTEC:n verkkosivut. Viitattu 17.02.2016. http://jyvsectec.fi/fi/tietoa-meista/

JYVSECTEC. 2016. JYVSECTEC:n verkkosivut. Viitattu 17.02.2016. http://jyvsectec.fi/fi/kyberymparisto/

Ligh M., Adair S., Hartstein B. 2010. Malware Analyst's Cookbook and DVD : Tools and Techniques for Fighting Malicious Code. Hoboken. Wiley

Oktavianto D., Muhardianto I. 2013. Cuckoo Malware Analysis: Analyze malware using Cuckoo Sandbox. Birmingham. Packt Publishing Ltd.

Portnoy M., 2012. Virtualization Essentials. Sybex.

Sikorski M., Honig A. 2012. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. San Fransisco. No Starch Press.

Singh, S. Breaking the Sandbox. Viitattu 15.14.2016. https://www.exploit-db.com/docs/34591.pdf

Vashisht, S., Singh A. 2014. Turing Test in Reverse: New Sandbox-Evasion Techniques Seek Human Interaction. FireEye. Viitattu 15.4.2016. https://www.fireeye.com/blog/threat-research/2014/06/turing-test-in-reverse-new-sandbox-evasion-techniques-seek-human-interaction.html

Manual. VirtualBox. Viitattu 26.4.2016. https://www.virtualbox.org/manual/ch01.html

Liite 1. Yara sääntö

```
// Copyright (C) 2010-2014 Cuckoo Foundation.
// This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
// See the file 'docs/LICENSE' for copying permission.

rule vmdetect
{
    meta:
        author = "nex"
        description = "Possibly employs anti-virtualization techniques"

    strings:
        // Binary tricks
        $vmware = {56 4D 58 68}
        $virtualpc = {0F 3F 07 0B}
        $ssexy = {66 0F 70 ?? ?? 66 0F DB ?? ?? ?? ?? ?? 66 0F DB ?? ?? ?? ?? ?? 66 0F EF}
        $vmcheckdll = {45 C7 00 01}
        $redpill = {0F 01 0D 00 00 00 00 C3}

        // Random strings
        $vmware1 = "VMXh"
        $vmware2 = "Ven_VMware_" nocase
        $vmware3 = "Prod_VMware_Virtual_" nocase
        $vmware4 = "hgfs.sys" nocase
        $vmware5 = "mhgfs.sys" nocase
        $vmware6 = "prleth.sys" nocase
        $vmware7 = "prlfs.sys" nocase
        $vmware8 = "prlmouse.sys" nocase
        $vmware9 = "prlvideo.sys" nocase
        $vmware10 = "prl_pv32.sys" nocase
        $vmware11 = "vpc-s3.sys" nocase
        $vmware12 = "vmsrvc.sys" nocase
        $vmware13 = "vmx86.sys" nocase
        $vmware14 = "vmnet.sys" nocase
        $vmware15 = "vmicheartbeat" nocase
        $vmware16 = "vmicvss" nocase
        $vmware17 = "vmicshutdown" nocase
        $vmware18 = "vmicexchange" nocase
        $vmware19 = "vmdebug" nocase
        $vmware20 = "vmmouse" nocase
        $vmware21 = "vmtools" nocase
        $vmware22 = "VMMEMCTL" nocase
        $vmware23 = "vmx86" nocase
        $vmware24 = "vmware" nocase
        $virtualpc1 = "vpcbus" nocase
        $virtualpc2 = "vpc-s3" nocase
        $virtualpc3 = "vpcuhub" nocase
```

```
        $virtualpc4 = "msvmmouf" nocase
        $xen1 = "xenevtchn" nocase
        $xen2 = "xennet" nocase
        $xen3 = "xennet6" nocase
        $xen4 = "xensvc" nocase
        $xen5 = "xenvdb" nocase
        $xen6 = "XenVMM" nocase
        $virtualbox1 = "VBoxHook.dll" nocase
        $virtualbox2 = "VBoxService" nocase
        $virtualbox3 = "VBoxTray" nocase
        $virtualbox4 = "VBoxMouse" nocase
        $virtualbox5 = "VBoxGuest" nocase
        $virtualbox6 = "VBoxSF" nocase
        $virtualbox7 = "VBoxGuestAdditions" nocase
        $virtualbox8 = "VBOX HARDDISK"  nocase

        // MAC addresses
        $vmware_mac_1a = "00-05-69"
        $vmware_mac_1b = "00:05:69"
        $vmware_mac_1c = "000569"
        $vmware_mac_2a = "00-50-56"
        $vmware_mac_2b = "00:50:56"
        $vmware_mac_2c = "005056"
        $vmware_mac_3a = "00-0C-29" nocase
        $vmware_mac_3b = "00:0C:29" nocase
        $vmware_mac_3c = "000C29" nocase
        $vmware_mac_4a = "00-1C-14" nocase
        $vmware_mac_4b = "00:1C:14" nocase
        $vmware_mac_4c = "001C14" nocase
        $virtualbox_mac_1a = "08-00-27"
        $virtualbox_mac_1b = "08:00:27"
        $virtualbox_mac_1c = "080027"

    condition:
        any of them
}
```

Liite 2. Virtuaalikoneen luominen:

Luodaan kiintolevy:

$ VBoxManage createhd --filename /levyn/polku.vdi --size <koko_bitteinä>

Asetetaan käyttöjärjestelmän tyyppi:

$ VBoxManage createvm --name “koneen_nimi” --ostype "Linux" --register

Lisätään SATA kontrolleri ja liitetään dynaaminen levy.

*$ VBoxManage storagectl “koneen_nimi” --name "SATA Controller" --add sata *
> --controller IntelAHCI
*$ VBoxManage storageattach “koneen_nimi” --storagectl "SATA Controller" --port 0 *
> --device 0 --type hdd --medium /levyn/polku.vdi

Lisätään IDE kontrolleri ja liitetään DVD-asema, johon tuodaan asennusmedia (.iso)

$ VBoxManage storagectl “koneen_nimi” --name "IDE Controller" --add ide
*$ VBoxManage storageattach “koneen_nimi” --storagectl "IDE Controller" --port 0 *
> --device 0 --type dvddrive --medium “/polku/tiedostoon/.iso”

Määritetään järjestelmän asetukset:

$ VBoxManage modifyvm “ koneen_nimi” --ioapic on
$ VBoxManage modifyvm “koneen_nimi” --boot1 dvd --boot2 disk
$ VBoxManage modifyvm “koneen_nimi” --memory 256
$ VBoxManage modifyvm “koneen_nimi” --nic1 hostonly --hostonlyadapter1 vboxnet0
$ VBoxManage modifyvm “koneen_nimi” --vrde on

64-bittisille guesteille:

$VboxManage modifyvm ”koneen_nimi” --longmode on

Nyt työasema voidaan käynnistää:

$ VBoxHeadless --startvm “koneen_nimi”

Kun työaseman asennus on valmis voidaan boot-järjestyksessä asettaan kovalevy ensimmäiseksi:

$ VBoxManage modifyvm “koneen_nimi” --boot1 disk --boot2 dvd

Koneen tilaa voidaan hallita seuraavilla komennoilla:

VBoxManage controlvm “koneen_nimi” pause|resume|reset|poweroff|savestate

Liite 3. Signature

```
# Copyright (C) 2010-2015 Cuckoo Foundation.
# This file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org
# See the file 'docs/LICENSE' for copying permission.

from lib.cuckoo.common.abstracts import Signature

class CreatesExe(Signature):
    name = "creates_exe"
    description = "Creates a Windows executable on the filesystem"
    severity = 2
    categories = ["generic"]
    authors = ["Cuckoo Developers"]
    minimum = "0.5"

    # This is a signature template. It should be used as a skeleton for
    # creating custom signatures, therefore is disabled by default.
    # It doesn't verify whether a .exe is actually being created, but
    # it matches files being opened with any access type, including
    # read and attributes lookup.
    enabled = False

    def run(self):
        match = self.check_file(pattern=".*\\.exe$",
                                regex=True)
        if match:
            self.data.append({"file": match})
            return True
        return False
```