Hunajapurkkien käyttö tietoturvassa

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinna, Tietojenkäsittelyn koulutusohjelma

Kevät, 2017

Tomi Korpinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tietojenkäsittelyn koulutusohjelma
Visamäki, Hämeenlinna

Tekijä Tomi Korpinen **Vuosi** 2017

Työn nimi Hunajapurkkien käyttö tietoturvassa

Työn ohjaaja/t Erkki Laine

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutustua hunajapurkeiksi kutsuttuihin ansoihin, etsiä nykyaikana toimivia Hunajapurkki-ratkaisuja, ottaa käyttöön sopivia ohjelmistoja ja tutkia niiden keräämiä tuloksia.

Työn teoriaosassa käydään läpi aihetta käsittelevää kirjallisuutta ja kirjoittajien mielipiteitä teknologian hyötykäytöstä eri organisaatioiden tietoturvassa. Työn käytännön osuudessa asennetaan kolme erilaista hunajapurkkiohjelmistoa eri virtuaaliympäristöihin ja tutkitaan niillä saatuja tuloksia.

Työn tuloksena on selvitys Hunajapurkki-teknologian käyttöönoton ongelmista ja toimivat asennusohjeet joita voidaan käyttää uusien järjestelmien käyttöönotossa.

Avainsanat Honeypot, Tietoturva, Dionaea, Cowrie, Hunajapurkki

Sivut 33 sivua, joista liitteitä 7 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

Author Tomi Korpinen **Year** 2017

Subject Honeypots in information security

Supervisors Erkki Laine

ABSTRACT

The goal of this thesis was to research Honeypot technology, find honeypots which work in modern systems, install these different Honeypots and examine the data gathered by the systems.

The aim of the theory part was to research existing material and the writers opinions on incorporating honeypots to be a part of some organizations network security. The practical part of the thesis focuses on installing three honeypots in virtualization environments and examining the data gathered by the systems.

The end product is a report on Honeypots as a whole, the problems faced when installing the used systems and a set of instructions for installing the honeypots used in this thesis.

Keywords Honeypot, security, Dionaea, Cowrie, Hunajapurkki

Pages 33 pages including appendices 7 pages

SISÄLLYS

KÄSITELUETTELO

DMZ
Demilitarisoitu alue on tietotekniikassa sisä- ja ulkoverkon välissä oleva harmaa alue, jossa sijaitsevat esimerkiksi yrityksen julkiset palvelimet.

VPS
Virtuaalinen privaattipalvelin on palveluna myytävä virtuaalikone jota ylläpitäjä vuokraa asiakkaalle.

Python
Oliopohjainen ohjelmointikieli

MySQL
Suosittu avoimen lähdekoodin tietokanta

Hypervisor
Hypervisor eli Virtual machine monitor on tilanteesta riippuen esimerkiksi ohjelmisto, joka luo ja ajaa virtuaalikoneita. Hypervisoria käyttävää tietokonetta kutsutaan isäntäkoneeksi ja jokaista virtuaalikonetta kutsuaan vieraskoneeksi.

IDS
Intrusion Detection System eli tunkeilijan havaitsemisjärjestelmä on tietoverkkoon asetettava ohjelmisto joka havaitsee mahdolliset tunkeilijat.

1 JOHDANTO

Hunajapurkiksi kutsuttu järjestelmä on tietoverkossa oleva tietokone, johon on tarkoituksellisesti luotu heikkouksia, joilla yritetään saada verkon kimppuun hyökkäävien käyttäjien huomio. Järjestelmä kerää hyökkäyksistä tietoa ja tallentaa tämän datan ohjelmankohtaisella tavalla esimerkiksi lokeihin tai tietokantaan. Järjestelmän keräämällä tiedolla voidaan esimerkiksi tutkia miten hyökkääjät lähestyvät erilaisia tilanteita järjestelmiä hakkeroidessaan, millaisia työkaluja hakkerit käyttävät, tunnistaa uusia haittaohjelmia tai paikantaa hyökkääjän sijainti.

Sain opinnäytetyön aiheen yhdestä ideaseminaarista, joten työllä ei ole toimeksiantajaa. Työn tavoitteena on tutustua syvällisemmin hunajapurkkeihin, niiden toimintaperiaatteisiin, historiaan, nykyaikana toimiviin hunajapurkkeihin ja saada koottua toimiva hunajapurkki-järjestelmä, jonka kimppuun joku ulkopuolinen hyökkää. Toivon näiden hyökkäysten pohjalta saavani kerättyä tietoa erilaisista hyökkäysmetodeista. Järjestelmä tullaan ylläpitämään Amazonin pilvipalvelualustalla, jotta riski hyökkääjien pääsystä käyttäjän omaan tietoverkkoon saadaan nollattua.

Työssä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

Mikä on hunajapurkki?
Minkälaisia hyökkäyksiä järjestelmä kohtaa?
Mitä hyötyä hunajapurkeista on?

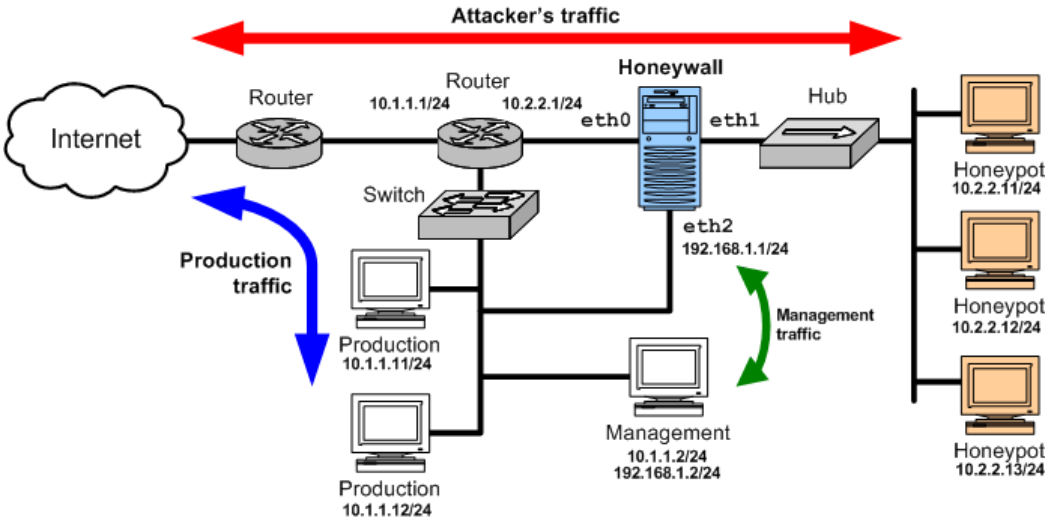
2 HUNAJAPURKKI

Hunajapurkilla on monia eri määrityksiä riippuen käyttäjän näkökulmasta, mutta käytän tässä työssä Spitznerin määritelmää ”Hunajapurkki on informaatiojärjestelmä, jonka arvo piilee sen luvattomassa tai laittomassa käytössä”.(Spitzner, 2003.) Toisin sanoen hunajapurkki voi olla esimerkiksi järjestelmä johon on pyritty luomaan tietoturva-aukkoja jotka mahdollistavat ulkopuoliselle hakkerille pääsyn resurssiin.

Hunajapurkin todellinen arvo pohjautuu sillä kerättyyn tietoon. Monitoroimalla järjestelmässä liikkuvaa dataa saadaan kerättyä tietoa, jota ei välttämättä ole mahdollista saada muilla tavoilla. Käymällä läpi saatua tietoa voidaan myös saada selville uusia tietoturva-aukkoja hyökkääjien käyttämiä menetelmiä tutkimalla.

Hunajapurkkien mahdollisten tietoturvariskien vähentämiseksi voidaan käyttää honeynettiä. Honeynet on tietoverkko, se koostuu useista hunajapurkeista, ne käyttävät eri käyttöjärjestelmiä ja ovat eri vuorovaikutustasoilla, jotta voidaan samanaikaisesti kerätä tietoa erilaisista hyökkäyksistä. (Spitzner, 2003.)

Honeynet on eristetty muusta tietoverkosta, jotta hyökkääjä voi käyttää hunajapurkkikoneita ilman vaaraa lähiverkon muille laitteille. (Kuva 3.) Yksi honeynetin avainelementeistä on toisen tason siltauslaite nimeltään Honeywall. Honeywallilla minimoidaan verkkoon kohdistuva riski ohjaamalla kaikki honeynetin liikenne sen läpi ja samalla saadaan kerättyä lisää tietoa analysointia varten.



Kuva 1. Honeynet eristettynä omaan aliverkkoon Honeywallin taakse (González, 2004.)

2.1 **Hunajapurkkien historia**

Hunajapurkkeja on käytetty jo useita vuosikymmeniä, mutta ensimmäinen aihetta sivuava kirjoitettu materiaali, joka oli saatavilla tavallisille ihmisille, on Clifford Stollin kirja ”The Cuckoo’s Egg”, joka julkaistiin vuonna 1990.

Kirjassa Stoll käy läpi tapahtumia kymmenen kuukauden ajalta vuosina 1986 ja 1987. Kirja ei ole kovinkaan tekninen, mutta siinä käydään läpi, kuinka astronomi Stoll työskennellessään Lawrence Berkeley-laboratoriossa huomasi, että hänen ylläpitämänsä järjestelmän kimppuun oli hyökätty.

Sen sijaan että Stoll olisi estänyt hyökkääjän pääsyn järjestelmään hän päätti tarkkailla hyökkääjän toimia ja saada selville kuka tämä hyökkääjä oli. Luomalla tekaistua materiaalia Stoll pystyi selvittämään, millaista tietoa hyökkääjä halusi kerätä ja samalla pidentämällä hyökkääjän järjestelmässä käyttämää aikaa hän pystyi helpottamaan hyökkääjän sijainnin paikantamista.

Vuonna 1990 Bill Cheswick julkaisi tietoturva-ammattilaisille suunnatun kirjoituksen ”An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied” joka tutki aihetta teknisemmin. Toisin kuin Stollin kirjassa tässä kirjoitelmassa Cheswick dokumentoi Berferd-järjestelmänsä rakentamisen, jonka hän halusi joutuvan hyökkäyksen kohteeksi. Berferd on myös ensimmäinen dokumentoitu tapaus järjestelmästä, jota voidaan kutsua hunajapurkiksi.

Vaikka ensimmäiset kirjoitukset julkaistiin 1990-luvun alussa, ensimmäinen julkinen hunajapurkkiratkaisu, Deception Toolkit (DTK) versio 0.1, julkaistiin vasta vuonna 1997. DTK oli Unix-käyttöjärjestelmälle asennettava kokoelma PERL-skriptejä ja C-koodia, jolla Cheswickin Berferd-järjestelmän lailla pyrittiin emuloimaan tunnettuja UNIX-järjestelmän tietoturva-aukkoja, jotka joutuessaan hyökkäyksen kohteeksi tallensivat tietoa hyökkääjien käyttäytymisestä ja toimista.

DTK:n jälkeen vuonna 1998 NAI julkaisi ensimmäisen kaupallisen hunajapurkki tuotteen: CyberCop Sting. Cyber Cop oli Windows NT-käyttöjärjestelmille asennettava ohjelmisto, jolla pystyttiin emuloimaan useita järjestelmiä samanaikaisesti. Tämä mahdollisti kokonaisten tietoverkkojen emuloinnin.

Samana vuonna Marty Roesch alkoi myös kehittää suurille yrityksille suunnattua hunajapurkkiratkaisua, joka sai nimekseen NetFacade. Roeschin työryhmä sai luotua järjestelmän, jolla pystyttiin yhdellä isäntäjärjestelmällä simuloimaan kokonainen C-luokan tietoverkko. NetFacade pystyi emuloimaan seitsemää eri käyttöjärjestelmää ja niihin erilaisia palveluita. NetFacade ei kuitenkaan tullut kovin suosituksi, mutta tämän projektin sivutuotteena oli Roeschin kehittämän avoimen lähdekoodin IDS-ohjelma Snort.

Vuonna 1999 luotiin Honeynet-project. Tämä on 30 tietoturva-ammattilaisen muodostama tutkimusryhmä, joka pyrkii tutkimaan blackhat-yhteisöä ja jakamaan keräämiään tuloksia koko turvallisuusyhteisölle. Vuonna 2001 ryhmä julkaisi tutkimusmateriaalinsa kirjassa ”Know Your Enemy”.

Vuosituhannen vaihteen jälkeen erilaisten ”matojen” määrä kasvoi huomattavasti ja nämä osoittautuivat hyvin tehokkaiksi ja niiden kyky levitä nopeasti internetin välityksellä oli myös yllättävä. Hunajapurkit olivat hyvä tapa kerätä tietoa näistä eri madoista.

Vuonna 2002 oli ensimmäinen dokumentoitu tapaus, jossa hunajapurkkiteknologian onnistui napata tuntematonta haavoittuvuutta hyväksikäyttävän ohjelman. Tämä tietoturva-aukko oli turvallisuusyhteisön tiedossa, mutta uskottiin, että haavoittuvuutta hyväksikäyttävää ohjelmaa ei ollut sillä hetkellä olemassa. Ohjelman tunnistamisen ja kiinniottamisen jälkeen pystyttiin demonstroimaan, että hunajapurkkien arvo ei ole vain tunnettujen hyökkäysten kiinniotossa, vaan myös tuntemattomien hyökkäysten havaitsemisessa ja nappaamisessa.

2.2 **Eri vuorovaikutustasot**

Hunajapurkit voidaan jakaa kolmeen ryhmään niiden vuorovaikutustason mukaan. Vuorovaikutuksella tarkoitetaan hyökkääjälle annettavaa tilaa toimia hunajapurkkijärjestelmän sisällä.

High Interaction eli korkean vuorovaikutuksen hunajapurkki on tavallinen tietokonejärjestelmä, kytkin tai reititin. Tällä järjestelmällä ei kuitenkaan ole tavanomaisia tehtäviä tietoverkossa, eikä sillä ole säännöllisiä käyttäjiä, joten järjestelmän ei tulisi pakollisten palveluiden lisäksi luoda minkäänlaista liikennettä tietoverkossa. Tällä pohjalla voidaan olettaa, että kaikki järjestelmällä tapahtuva toiminta on haitallista, joten kaikki liikenne tallennetaan lokeihin ja kaikki toiminta nauhoitetaan. (Holz & Provos, 2007.)

Tämän hunajapurkki tyypin heikkouksia ovat sen vaatiman ylläpidon määrä: koska järjestelmä tarjoaa hyökkääjälle suuremman alustan, millä hän voi toimia, vie myös saadun tiedon analysointi enemmän aikaa. Kokonaisen käyttöjärjestelmän käyttäminen mahdollistaa myös tilanteen, jossa hyökkääjä pystyy käyttämään hunajapurkkia muihin järjestelmiin hyökkäämiseen. (Holz & Provos, 2007.)

Low Interaction on aikaisempaan verrattuna matalan vuorovaikutuksen hunajapurkki. Se pyrkii emuloimaan palveluita, verkkopinoja tai muita oikean järjestelmän osia. Tällä hunajapurkilla pyritään tarjoamaan hyökkääjälle hyvin rajattu kokemus ja keräämään mahdollisimman paljon tietoa. Esimerkkinä tällaisesta hunajapurkista on järjestelmä, joka emuloi http-palvelinta ja pystyy vastaamaan yhtä tiedostoa koskeviin pyyntöihin. Hyökkääjän ja järjestelmän välisen vuorovaikutuksen tulisi olla tarpeeksi hyvä,

jotta hyökkääjää tai automatisoitua työkalua saadaan hämättyä. (Holz & Provos, 2007)

Tällaisen hunajapurkin ainoihin heikkouksiin voidaan laskea kerätyn tiedon laatu. Koska järjestelmä on vain simulaatio pienestä osasta oikeaa järjestelmää, ei hyökkääjä pysty tekemään paljon. Näin saadaan mahdolliset koko tietoverkkoa koskevat riskit mitätöityä. (Holz & Provos, 2007.)

Medium Interaction on keskitason vuorovaikutuksen hunajapurkki, joka pyrkii yhdistämään alhaisten ja korkeiden vuorovaikutustasojen hunajapurkkien hyödyt ja samalla vähentämään kyseisten järjestelmien haittapuolia. (Pathan, 2014.)

Näiden hunajapurkkien avainominaisuus on sovellusvirtualisointi. Tällainen hunajapurkki ei pyri täysin simuloimaan kokonaista toimivaa käyttöjärjestelmä ympäristöä eikä toteuttaa kaikkia sovellusprotokollan yksityiskohtia vaan pyrkii tarjoamaan tarpeeksi vastauksia, joita tunnetut tietoturva-aukkoja käyttävät ohjelmat odottavat ennen kuin ne yrittävät murtautua järjestelmään. (Pathan, 2014.)

2.3 **Hyödyt**

Hunajapurkkien hyöty on suurilta osin organisaatio kohtaista, mutta tähän kappaleeseen on listattu, jotain mahdollisia käyttötapauksia joissa näistä järjestelmistä on hyötyä.

Tiedon arvo
Suuret organisaatiot keräävät suuria määriä dataa päivittäin palomuurien lokeista, järjestelmälokeista ja eri IDS-lokeista. Valtavaa määrää tietoa voi olla hyvin hankala tutkia, ja suurin osa tästä kerätystä tiedosta on arvoltaan hyödytöntä taustakohinaa. Hunajapurkit keräävät hyvin vähän tietoa. Sen sijaan, että järjestelmä keräisi useita gigatavuja tietoa päivittäin, hunajapurkin keräämä data voidaan kiteyttää muutamaan megatavuun. Esimerkkinä tästä voidaan käyttää Honeynet projectia, joka saa keskimäärin kerättyä alle yhden megatavun tietoa päivässä. Datan vähäisestä määrästä huolimatta voidaan olettaa, että kerätty tieto on joko jonkinlainen hyökkäys, skannaus tai luotaus. (Pathan, 2014.)

Resurssit
Haaste, joka voi tulla eteen useimmille turvallisuusmekanismeille, on resurssien rajallisuus tai jopa niiden loppuminen. Resurssien loppuessa kesken turvallisuusmekanismit lopettavat toimintansa niiden saatavilla olevien resurssien ylikuormituksen takia. Esimerkiksi palomuuri voi epäonnistua, jos sen yhteyspöytä on täynnä, jolloin sen resurssit ovat loppuneet ja se ei pysty monitoroimaan yhteyksiä. Tai, jos keskitetty lokipalvelin ylikuormittuu, se ei ehkä saa kerättyä kaikkia tarjolla olevia lokeja, jolloin tärkeää dataa voi kadota. (Pathan, 2014.)

Koska hunajapurkit monitoroivat ja tallentavat vain vähän liikennettä, ei resurssien puuttumisen pitäisi olla ongelmana, toisin kuin IDS:t, joilla ongelmaksi voi muodostua koko verkon valvominen ja siitä aiheutuvat suuret määrät dataa lyhyessä ajassa. Hunajapurkki tarkkailee vain liikennettä, joka kohdistuu suoraan siihen, joten järjestelmä ei ylikuormitu. Sivuhyötynä resurssien vähäisestä käytöstä on huomattavan pieni rahallinen sijoitus hunajapurkkijärjestelmässä käytettävään laitteistoon. (Pathan, 2014.)

Yksinkertaisuus
Suurin etu hunajapurkeissa on niiden yksinkertaisuus. Vaikka jotkin hunajapurkit ovat monimutkaisia, ne kaikki toimivat silti samalla tavalla: jos joku ottaa yhteyttä järjestelmään, tulee tapaus tutkia. Tämä on hyvä asia, sillä mitä monimutkaisempi järjestelmä on, sitä suuremmalla todennäköisyydellä jokin tulee menemään pieleen. (Pathan, 2014.)

2.4 Haitat

Kuten kaikilla tietoturvateknologioilla, hunajapurkeillakin on myös haittoja. Näiden heikkouksien takia hunajapurkki ei voi korvata muita turvallisuusmekanismeja, mutta niistä on hyötyä valmiiden tietoturvamekanismien rinnalla.

Pieni näkökenttä
Hunajapurkki näkee vain siihen kohdistetut hyökkäykset. Kun hyökkääjä murtautuu tietoverkkoon ja hyökätessä useisiin järjestelmiin, ja kun hyökkääjä ei samalla käy hunajapurkin kimppuun ei järjestelmä tiedä näistä hyökkäyksistä mitään. Kaikki hunajapurkilla kerätty tieto on arvokasta, mutta jos siihen ei kuitenkaan kohdistu hyökkäyksiä, se ei saa kerättyä mitään tietoa, jolloin järjestelmä on arvoton. (Spitzner, 2003.)

Riski
Toisena heikkoutena on riski, jonka hunajapurkit voivat tuoda ympäristöön, jos hunajapurkin kimppuun hyökätään, voidaan sitä käyttää muiden koneiden kimppuun hyökkäämiseen. Jotkin hunajapurkit luovat suurempia riskejä ja jotkut pienempiä, riippuen siitä, emuloiko hunajapurkki vain joitain palveluita vai onko käytössä hunajapurkki, joka antaa hyökkääjälle kokonaisen käyttöjärjestelmän. (Spitzner 2003.)

Järjestelmän paljastuminen
Joissakin hunajapurkki järjestelmissä on ”sormenjälkiä”, joilla järjestelmän oikea identiteetti voi paljastua. Esimerkiksi, jos hunajapurkki on asetettu emuloimaan NT IIS web-palvelinta, mutta asennus on toteutettu huonosti, voi järjestelmässä olla tuntomerkkejä, joilla sen tunnistaa Solaris web -palvelimeksi. Nämä ristiriitaiset identiteetit voivat auttaa hyökkääjää tunnistamaan järjestelmän hunajapurkiksi. (Spitzner 2003.)

2.5 **Rooli tietoturvassa**

Hunajapurkit voidaan jakaa tuotanto- ja tutkimushunajapurkkeihin. Tutkimushunajapurkit tarjoavat alustan, jolla voidaan keskittyä etsimään uusia haittaohjelmia, tutkia hyökkääjien toimia askel askeleelta, kerätä tietoa olemassa olevista hyökkäysmetodeista ja ennakoida tulevia hyökkäyksiä. Tutkimushunajapurkit ovat myös käytettävyydeltään hyvin monimutkaisia, joten niitä käyttävät pääasiassa tutkijat, armeijat ja hallituksien organisaatiot. (Pathan, 2014.)

Tuotantohunajapurkkeja käytetään olemassa olevien tietoverkkojen tietoturvan parantamiseen. Tuotantohunajapurkeista saatava hyöty voidaan jakaa Bruce Schneierin luoman turvallisuusmallin mukaan kolmeen osa-alueeseen: torjunta, havaitseminen ja vastatoimi. (Spitzner, 2003.)

2.5.1 Torjunta

Haluttaessa estää hyökkääjien pääsy tietoverkkoon, voidaan käyttää monenlaisia työkaluja, esimerkiksi palomuureja, hyviä salasanakäytäntöjä ja digitaalisia sertifikaatteja. Näillä menetelmillä voidaan rajoittaa resursseihin pääsy tietyille henkilöille. Kriittisen datan ja salaisten dokumenttien kryptaaminen auttaa myös tietovuotojen estämisessä. (Spitzner, 2003.)

Voidaan siis ihmetellä, mitä hyötyä hunajapurkeista voi olla hyökkääjien estämisessä, kun ohjelmistoilla ei estetä ihmisten pääsyä järjestelmään, vaan päinvastoin yritetään kannustaa sitä? Lisäksi huonosti konfiguroitu hunajapurkki voi jopa altistaa järjestelmän ylimääräisille riskeille. (Spitzner, 2003.)

Asia on herättänyt keskustelua turvallisuusyhteisöissä, ja joidenkin ihmisten mielestä hunajapurkkien hyöty hyökkääjien torjunnassa syntyy hyökkääjien hämäämisessä ja hunajapurkin käyttämisessä pelotteena. Hyökkääjät eivät välttämättä tahdo hyökätä tietoverkon kimppuun, jos he tietäväthyökkäyksen kohteen olevan hunajapurkki, johon he voivat jäädä kiinni. (Spitzner, 2003.)

Vaikka hunajapurkin käyttäminen hämäyksenä tai pelotteena voi olla hyvä psykologinen ase, useimpien organisaatioiden on kuitenkin parempi käyttää resurssinsa tietoturva-aukkojen paikkaamiseen muilla keinoilla. Tällaisen hunajapurkin käyttö haavoittuvassa järjestelmässä on kyseenalaista, koska suurin osa hyökkääjistä ei käytä aikaa mahdollisten kohteiden analysointiin, vaan käyttää ns. haulikkoratkaisua, jossa he lähettävät mahdollisimman monta hyökkäystä eri laitteisiin nähdäkseen, mitkä niistä toimivat. (Spitzner, 2003.)

2.5.2 Havaitseminen

Torjunnan jälkeinen taso on luvattoman toiminnan havaitseminen ja siitä hälyttäminen. Ennemmin tai myöhemmin torjunta tulee epäonnistumaan. Esimerkkejä ovat: väärin konfiguroidut palomuurisäännöt, yksinkertaiset salasanat ja uudet tietoturva-aukot. Torjunta vain vähentää riskiä, mutta ei pysty poistamaan sitä. (Spitzner, 2003.)

Tietoturvassa on havainnoinnin parantamiseen jo useita teknologioita, näistä yhtenä esimerkkinä ovat erilaiset tunkeilijan havaitsemisjärjestelmät (IDS), jotka on suunniteltu vahtimaan tietoverkkoa ja havaitsemaan kaikki haitallinen toiminta. IDS monitoroi myös järjestelmälokeja, joista se etsii luvatonta toimintaa. (Spitzner, 2003.)

Torjunnassa hunajapurkkien hyöty on hyvin tapauskohtaista, mutta ne lisäävät huomattavasti enemmän arvoa havaitsemisessa. Reaalistaakseen tämän arvon pitää ensin käsitellä havaitsemisen kolme ongelmaa: hyökkäysten virheelliset esiintymät, hyökkäysten huomaamatta jääminen ja datan kokoaminen. (Spitzner, 2003.)

Yksittäisvirhe-esiintymät eivät ole ongelma, ja virheellisiä hälytyksiä tulee tapahtumaan. Ongelma ilmenee siinä vaiheessa, kun ylläpitäjä saa satoja tai jopa tuhansia hälytyksiä päivässä. Virheellisten hälytysten kasaantuessa on mahdollista, että hälytyksiä tarkkaileva henkilö jättää huomioimatta osan näistä, jolloin se järjestelmä, jonka tulisi suojata tietoverkkoa, muuttuu tehottomaksi. (Spitzner, 2003.)

IDS-järjestelmät ovat hyvä esimerkki virheellisten hälytysten tuottamista haasteista, sillä nämä tapaukset ovat niille hyvin tuttuja. Useimmat IDS-sensorit monitoroivat tietoverkkoa etsien sieltä epäilyttävää liikennettä. IDS:illä on käytössä tietokanta erilaisia haitallisiksi tunnistettuja signeerauksia. Huomatessaan epäilyttävää toimintaa IDS yrittää etsiä tietokannastaan täsmääviä signeerauksia. Kun IDS luulee löytäneensä osuman, se hälyttää asiasta järjestelmän ylläpitäjää. Ongelma tässä järjestelyssä on kuitenkin se, että IDS voi myös löytää tavallisesta tuotantoliikenteestä yhteensopivuuksia tietokannan signeerausten kanssa, jolloin järjestelmä luo virheellisiä hälytyksiä. (Spitzner, 2003.)

Ratkaisuna virheellisten hälytysten luomiselle on käskyttää järjestelmää olemaan ilmoittamatta validista tuotantoliikenteestä. Tämä on aikaa ja taitoa vaativa operaatio, jossa validia tuotantoliikennettä tulee verrata IDS:n tietokantaan ja kaikki haitalliset signeeraukset tulee joko muokata tai poistaa. Tällä tavalla saadaan vähennettyä virhe-esiintymien aiheuttamia hälytyksiä, mutta eliminoimalla suuria määriä signeerauksia, ongelmaksi voi ilmetä hyökkäysten huomaamatta jääminen. (Spitzner 2003.)

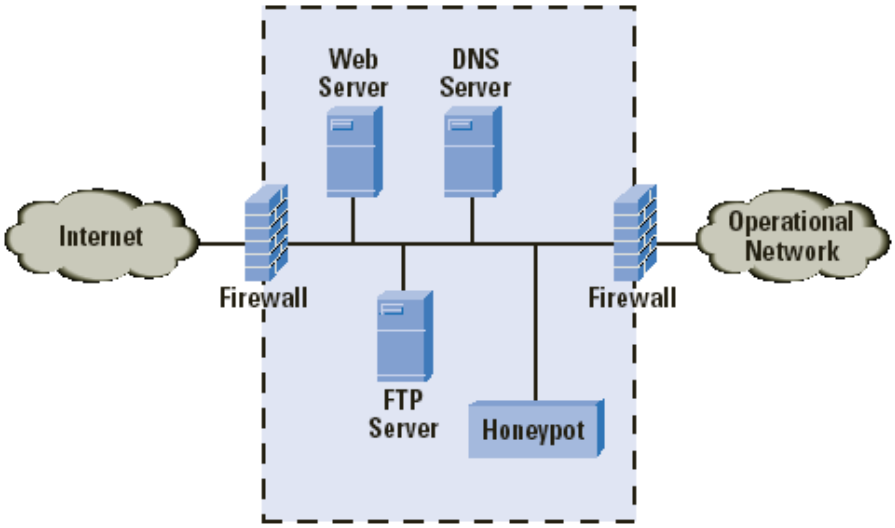
Jos järjestelmä voi aiheuttaa liikaa hälytyksiä, voi se tietenkin myös aiheuttaa liian vähän hälytyksiä. Riskinä on, että onnistunut hyökkäys voi tapahtua, mutta järjestelmä epäonnistuu siitä ilmoittamisessa. Riippumatta siitä

tunnistaako IDS-järjestelmä hyökkäykset signeerauksilla, protokollan varmentamisella tai jollain muulla metodologialla, ne voivat jättää huomioimatta uudet tai tuntemattomat hyökkäykset. (Spitzner, 2003.)

Kolmantena havaitsemisen haasteena on datan kerääntyminen. Teknologian kehittyessä syntyy koko ajan uusia tapoja kerätä dataa eri toiminnoista, ja moderni teknologia on jo äärimmäisen tehokas keräämään tietoa. IDS-järjestelmälokit ja ohjelmistolokit ovat hyviä luomaan gigatavuja dataa. Haasteeksi muodostuu tästä tietomäärästä arvokkaan datan kerääminen, jotta hyökkäysten paikantaminen ja estäminen ovat mahdollisia. (Spitzner, 2003.)

Yksinkertaisuutensa ansiosta hunajapurkki pystyy tehokkaasti vastaamaan näihin kolmeen ongelmaan. Hunajapurkeilla ei tulisi olla tuotantoliikennettä, joten ylimääräisiä virhe-esiintymiä ei tulisi syntyä. Hälytyksien puuttuminen ei myöskään ole ongelma, koska hunajapurkki ei joudu suodattamaan liikennettä signeerauksien tai muiden samankaltaisten hyökkäysten tunnistamismetodien avulla, vaan kaiken järjestelmään kohdistuneen liikenteen oletetaan olevan haitallista eli järjestelmää ei pysty helposti välttämään tai huijaamaan. Datan kerääminen ei myöskään muodostu ongelmaksi, koska verrattuna muihin ohjelmistoihin hunajapurkki tuottaa vähän dataa, joka helpottaa kerätyn tiedon analysointia. (Spitzner, 2003.)

Yksi mahdollinen ratkaisu on asettaa hunajapurkki lähiverkon demilitarisoidulle alueelle muiden palvelinten rinnalle: näin voidaan tarkkailla ulkopuolelta tulevaa liikennettä ilman, että tuotantoverkko vaarantuu. (Kuva 2.) DMZ-alueella kaikki yhteysyritykset olisivat eri porttien skannauksia ja yhteydenotot rinnakkaispalvelimilta osoittaisivat, että nämä järjestelmät ovat joutuneet hyökkäyksen kohteiksi ja niitä käytetään nyt muiden järjestelmien kimppuun hyökkäykseen. (Spitzner 2003.)



Kuva 2. DMZ:n sisälle asetettu hunajapurkki (Kellep n.d.)

2.5.3 Vastatoimi

Onnistuneen hyökkäyksen havaittua, tarvitaan kyky vastata siihen. Haaste näihin tapauksiin vastaamisessa on todisteiden keräys eli selvitys siitä mitä tapahtui ja milloin. Tämä on tärkeää paitsi mahdollisten hyökkääjien syyttämisessä myös tulevien hyökkäysten estämisessä. Kun organisaation järjestelmään on murtauduttu, tulee heidän selvittää, mihin järjestelmiin hyökkääjä on onnistunut murtautumaan, ovatko he luoneet takaovia tai logiikka pommeja, muokannet tai siepanneet arvokasta tietoa? Ovatko muut ihmiset samalla onnistuneet murtautumaan järjestelmään? (Spitzner 2003.

Hyökkääjän murtautuessa järjestelmään, heidän toimintansa jättävät todisteita, joita voidaan käyttää selvittämään, miten hyökkääjä pääsi järjestelmään sisälle, mitä he tekivät, kun järjestelmä oli heidän hallussaan, ja mahdollisesti, kuka hyökkääjä on? Ilman näitä todisteita organisaatiolla ei ole kykyä vastata samanlaisin hyökkäyksiin tehokkaasti. (Spitzner 2003.)

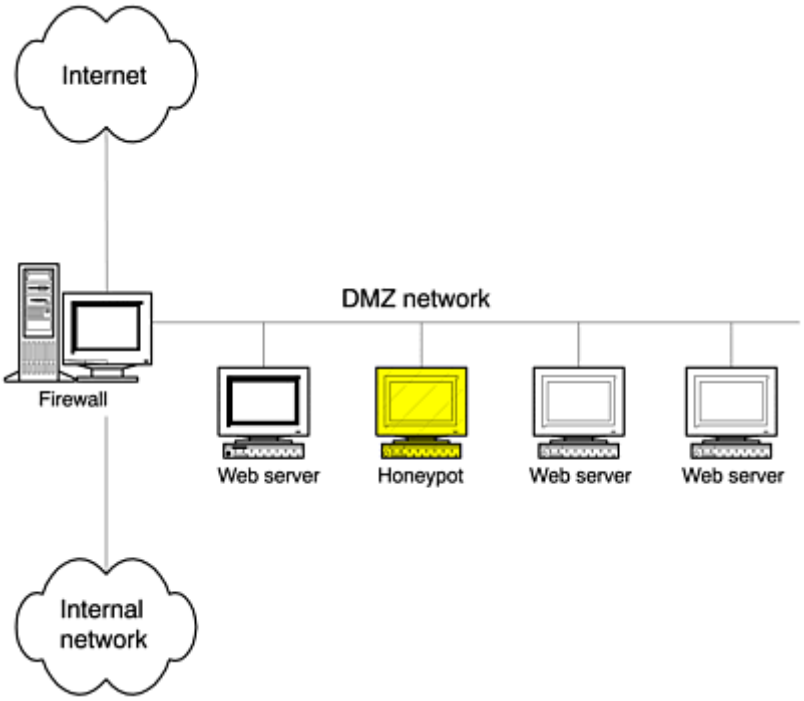
Vaikka hyökkääjä olisi ottanut askelia toimintansa piilottamiseksi, kuten järjestelmälokien muokkausta, nämä toimet voidaan silti jäljittää. Esimerkiksi on mahdollista askel askeleelta käydä läpi, mitä hyökkääjä teki katsomalla tiedon metadatasta sen MAC-aikoja. Useimmissa käyttöjärjestelmissä, jokainen tiedosto pitää yllä tietoa siitä, koska sitä on viimeksi muokattu, käytetty ja muunnettu. Selvittämällä mitä tiedostoja hyökkääjä on käynyt läpi, saadaan tietoa hyökkääjän tekemisistä. Tähän tarkoitukseen on luotu monia työkaluja, kuten esimerkiksi The Sleuth Kit, jolla voidaan muun muassa selvittää tapahtumien sarja pelkästään lukemalla MAC-aikoja. (Spitzner 2003.)

Mahdolliset todisteet voivat kuitenkin saastua hyvin nopeasti tavallisen tuotanto liikenteen luoman liikenteen takia. Tiedostoja uudelleen kirjoitetaan, eri prosessit sammuvat ja käynnistyvät, käyttäjät vaihtuvat ja näennäismuistin sisältö vaihtuu. Mitä enemmän toimintoja saastunut järjestelmä suorittaa sitä vaikeampaa todisteiden kerääminen on. (Spitzner 2003.)

Toisena ongelmana joillekin organisaatioille voi olla, että saastunut järjestelmä on yrityksen toiminnalle kriittinen ja sitä ei voi poistaa käytöstä. Organisaatio saattaa vian korjaamisen sijaan vain paikata aiemmin käytetyn tietoturva-aukon ja antaa järjestelmän jatkaa toimintaa, jolloin hyökkäyksen analysointi ja todisteiden kerääminen jäävät tekemättä. (Spitzner 2003.)

Hunajapurkit eivät ole organisaatiolle kriittisiä ja niillä ei ole tuotantoliikennettä, joten todisteiden saastuminen on vähäistä ja järjestelmän voi ottaa analysointiin ilman, että se vaikuttaa muuhun toimintaan. Hunajapurkin saastumisen mahdollistamiseksi voidaan järjestelmä naamioida esim. web-palvelimeksi. (Spitzner, 2003.)

Hunajapurkki voidaan asettaa olemassa olevien palvelimien sekaan organisaation DMZ:aan kuuntelemaan samaa HTTP-porttia muiden palvelimien kanssa, mutta konfiguroimaaan järjestelmän niin, että se ei palvele asiakkaita. Näin skannausyritykset osuvat kaikkiin neljään koneeseen, jolloin usean koneen saastuessa on myös todennäköistä, että hunajapurkki saastuu. (Kuva 2.) (Spitzner 2003.)



Kuva 3. Web palvelimeksi naamioitu hunajapurkki (Spitzner 2003.)

2.6 **Laillisuus**

Useimmat asiaa tutkivat kirjoitukset kertovat, että hunajapurkkien käytössä on lain puolesta kolme ongelmakohtaa, jotka eivät välttämättä haittaa teknologian käytössä, mutta joihin tulisi keskittyä. Nämä kirjoitukset myös keskittyvät hunajapurkkien tutkimiseen Yhdysvaltojen lain mukaan, joten siihen on keskitytty myös tässä työssä.

Yksinkertaisin ongelma on uskomus, että hyökkääjän houkutteleminen järjestelmän kimppuun hyökkäämiseen on mahdollinen syy haastaa hunajapurkin ylläpitäjä oikeuteen. Tämä ei kuitenkaan ole validi syy haastaa ketään oikeuteen kolmesta syystä: ansoitus on oikeudessa syyttäjää vastaan käytettävä puolustus, joten se ei ole syy haastaa muita oikeuteen, Hunajapurkin ylläpitäjän tulisi olla poliisiviranomainen, jotta tämä kelpaisi puolustukseksi ja vaikka aiemmat kohdat täyttyisivät, hunajapurkin tulisi houkutella ihminen toimimaan tavalla, jolla hän ei muuten toimisi, jota hunajapurkki ei toiminnallaan tee, vaan hyökkääjät toimivat omasta aloitteestaan. (Spitzner, 2010)

Toinen ja samalla monimutkaisin ongelma on yksityisyys. Hunajapurkki tarkkailee kaikkea järjestelmässä tapahtuvaa liikennettä, joten on mahdollista väittää kyseisen toiminnan rikkovan ainakin joitain yksityisyyttä turvaavia lakeja, kun järjestelmä tallentaa hyökkääjän käyttämiä tietoja, esimerkiksi hyökkääjän yrittämiä käyttäjätunnuksia ja hyökkääjän mahdollista viestintää muiden ihmisten kanssa. Yksityisyyttä koskevia lakeja on useita ja tilannetta monimutkaistaa hunajapurkin ja hyökkääjän geologinen sijainti. Kaksi helppoa tapaa vähentää näitä ongelmia on tarkentaa syy hunajapurkin käyttöön ja sitä millaista tietoa järjestelmällä kerätään. Hunajapurkin käytön syy on tärkeä, koska oman järjestelmäympäristön suojeluksi käytettävä hunajapurkki vapautuu yksityisyyttä koskevista rajoitteista. Hunajapurkilla kerätyn tiedon tyyppi taas vaikuttaa mahdollisten lainsäädännöllisten ongelmien määrään, jos hunajapurkki kerää vain pinnallista tietoa hyökkääjän toimista esimerkiksi IP-osoitteita ja tapahtuneen viestinnän ajankohtia eikä tallenna tapahtunutta viestintää, on ylläpitäjän tilanne parempi, koska viestinnän sisältöä koskevat yksityisyysongelmat ovat suurempia. (Spitzner, 2010)

Viimeinen ongelma on vastuu, joka syntyy, jos hyökkääjä käyttää hunajapurkkia muiden järjestelmien kimppuun hyökkäämiseen ja näin aiheuttaa vahinkoa. Argumentti on, jos hunajapurkin ylläpitäjä olisi ollut varovaisempi järjestelmänsä kanssa ei hyökkääjä olisi voinut aiheuttaa vahinkoa muille henkilöille. Vastaväitteenä tähän voisi olla se, että kaikissa teknologioissa on aina riski, että järjestelmä kaapataan ja sitä käytetään muiden vahingoittamiseksi. Järjestelmää asentaessa kannattaa kuitenkin miettiä, onko välttämättä tarpeellista käyttää korkeamman vuorovaikutustason hunajapurkkia, jolloin riski kaappaamiseen kasvaa. (Spitzner, 2010)

3 TOTEUTUS

Mahdollisia työhön käytettäviä ohjelmia etsiessä pyrittiin etsimään hunajapurkkeja, jotka keräävät tietoa eri alueista. Lopulta päädyttiin kolmeen hunajapurkkiin: Dionaea, Cowrie ja KFSensor. Näistä Dionaea ja Cowrie ovat Linux käyttöjärjestelmillä toimivia hunajapurkkeja ja KFSensor on maksullinen Windows hunajapurkki.

3.1 Dionaea-ohjelmiston ominaisuuksia

Dionaea on Markus Koetterin kehittämä pythonia käyttävä hunajapurkki, joka luotiin Google summer of code 2009-tapahtuman aikana Nepethes-nimisen hunajapurkin seuraajaksi. Dionaean tarkoitus on kerätä haittaohjelmia emuloimalla yleisiä järjestelmäpalveluita, joiden heikkouksia mahdollisten hyökkääjien toivotaan käyttävän hyväksi. Haittaohjelman havaittua Dionaea eristää sen analysoitavaksi. Ohjelman lopullinen tavoite on saada napattua kopio hyökkääjän käyttämästä haittaohjelmasta. (Dionaea n.d.)

Toimiakseen Dionaea käyttää seuraavia moduuleja:

Python-moduuli mahdollistaa pythonkääntäjän käyttämisen Dionaeassa ja sallii joidenkin Dionaean käyttämien skriptien hallinnan.
Curl-moduulia käytetään palvelimilta tiedostojen siirtämiseen ja lataamiseen. Sillä myös ladataan tiedostoja http:ta käyttäen ja lähetetään tiedostoja kolmansille osapuolille.
Emu-moduulilla havaitaan, profiloidaan ja tarvittaessa suoritetaan Shellkoodia.
Pcap-moduuli käyttää libpcap kirjastoa hylättyjen yhteysyritysten havaitsemiseen, jotta yhteyden hylätessään Dionaea voi silti saada tietoa yritys kokeiluista.

Hyökkääjien houkuttelemiseen Dionaea käyttää hyväkseen seuraavia protokollia: Black hole, EPMAP, FTP, Memache, Mirror, MQTT, nfq, PPTP, UPnP, SMB, HTTP & HTTPS, TFTP, MSSQL, MySQL ja VoIP. (Dionaea n.d.)

Tiedonkeräämiseen Dionaeassa on mahdollista käyttää tavallista tekstin lokitusta, mutta tämä voi johtaa hyvinkin pitkiin lokitiedostoihin, joissa on paljon ylimääräistä tietoa. Ratkaisuna tähän Dionaeassa on sisäinen kommunikaatiojärjestelmä, jota kutsutaan tapauksiksi. Jokaisella tapauksella on lähtöpiste, polku ja lista ominaisuuksia. Tapauksen luonnin jälkeen niitä käsittelee erilaiset ihandlerit eli tapauksienkäsittelijät, joiden ansiosta tieto saadaan muokattua käyttäjän haluamaan formaattiin. Dionaeassa käytettävissä olevat tapauksienkäsittelijät ovat: emuprofile, fail2ban, ftp, hpfeeds, log_db_sql, log_incident, log_json, log_sqlite, nfq, p0f, store, submit_http, submit_http_post, tftp_download, Virustotal (Dionaea n.d.)

Dionaea tuottaa suuria määriä tietoa, jonka läpikäymiseen voi saada kulumaan paljon aikaa. Tiedon analysointi prosessia saadaan kuitenkin nopeutettua asentamalla Dionaeaa käyttävälle hunajapurkki koneelle DionaeaFR niminen ohjelma, joka on luotu visualisoimaan Dionaean tuottamaa dataa. (Kuva 4.) Toimiakseen Dionaea ja DionaeaFR vaativat muutamia ohjelmia ja lisäosia näistä tärkeimpiä ovat: Python, cython, Django, Jquery, SqLite3 ja Kendo-UI

Kuva 4. Malli datan visualisoinnista DionaeaFR:llä (kuvakaappaus)

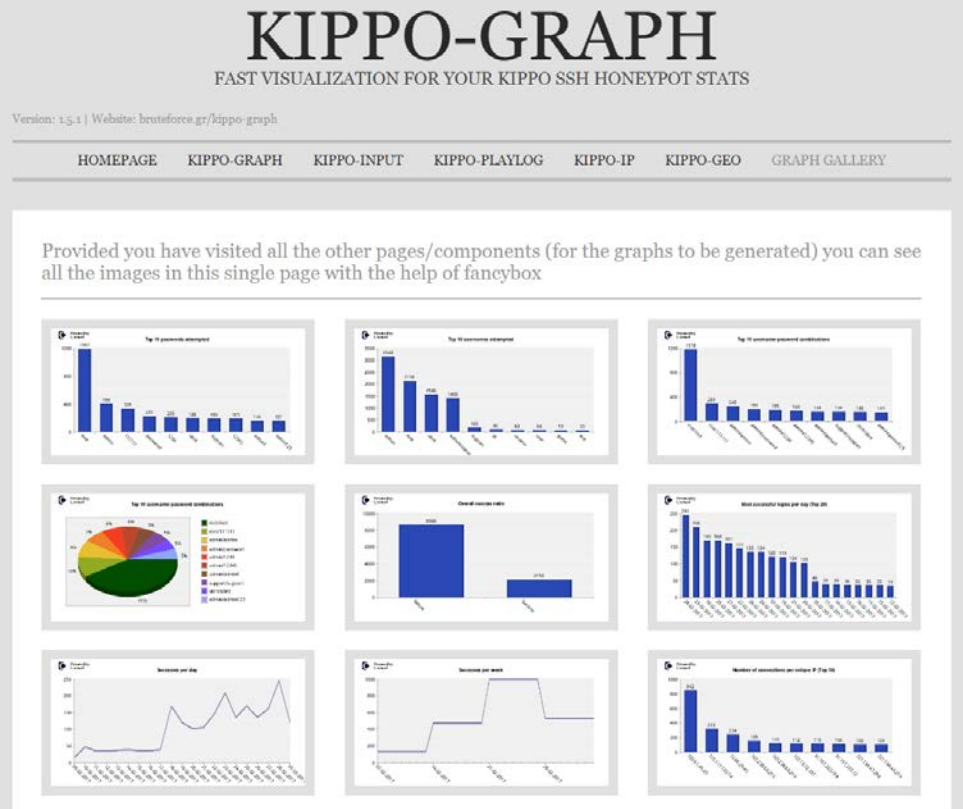
3.2 Cowrie-ohjelmiston ominaisuuksia

Cowrie on Pythonilla kirjoitettu kippo-nimiseen hunajapurkkiin pohjautuva SSH-hunajapurkki. Cowriella pyritään lokittamaan järjestelmään kohdistuvia hyökkäyksiä ja onnistuneen hyökkäyksen aikana suoritettavia komentorivi tapahtumia (Oosterhof, 2016.).

Hyökkääjiä hämätäkseen Cowrie pystyy emuloimaan tiedostojärjestelmää, joka on muodoltaan Debian 5.0-käyttöjärjestelmän kaltainen. Käyttäjä pystyy lisäämään valetiedostoja luotuun tiedostojärjestelmään, jotta hyökkääjä pystyy saamaan vastauksia käyttämilleen komennoille. Cowrie myös säilyttää kaikki tiedostot jotka hyökkääjä SSH-sessionsa aikana latasi järjestelmään (Oosterhof, 2016.).

Oletuksena Cowrie tuottaa järjestelmälokeja tavallisen tekstipohjaisen formaatin lisäksi myös Json-formaatissa. Tämä on paljon helpompi lukuista eikä sisällä ylimääräistä tekstiä. Tavallisen lokituksen lisäksi voidaan Cowrie käskyttää tallentamaan hyökkäyksistä saatua tietoa haluttuun tietokantapalveluun. Tietokannassa olevan tiedon voi uudelleen ohjata Kippo-

graph nimiseen ohjelmaan, joka visualisoi saadun tiedon ja asettaa tämän luettavaksi koneen ylläpitämälle verkkosivulle. (Kuva 5.) Cowriella ja Kippo-graphilla on useita vaatimuksia näistä tärkeimpiä ovat: Python, Twisted, cryptography, Php ja Apache (Oosterhof, 2016.).



Kuva 5. Cowriella kerätty tieto visualisoituna kippo-graphilla (kuvakaappaus)

Secure Shell

Käyttöjärjestelmissä on useita tapoja ottaa yhteyttä etänä järjestelmän komentoriviin. Yksi näistä tavoista on käyttää vanhoja Telnet-ohjelmia, jotka ovat saatavilla useimmilla tietoverkkoa käyttävillä käyttöjärjestelmillä. Telnetin käyttäminen etäyhteyden ottamiseen on kuitenkin vaarallista, koska kaikki käyttäjän Telnet session aikana lähettämä tieto on nähtävissä muilla lähiverkon koneilla, jolloin on mahdollista, että jokin kolmas osapuoli saa selville käyttäjän tietoja. (Krenz, 2006)

Etäyhteyksien tietoturvan parantamiseksi kehitettiin vuonna 1995 Secure Shell eli SSH, jonka tarkoitus on salata koneiden välinen liikenne. Tämän SSH-1:nä tunnetun protokollan kehitti Tatu Ylönen Helsingin Aalto yliopistossa. Tästä alkuperäisestä järjestelmästä löytyi kuitenkin vuosien aikana vikoja ja nykyinen SSH-2 standardi otettiin käyttöön vuonna 2006. (Rouse, 2016)

Salaamisen lisäksi SSH parantaa käyttäjien oikeuksien varmentamista, Mahdollistaa tiedostojen siirtämisen, X11 järjestelmän käyttämisen, porttien ohjaamisen ja muita turvallisuutta parantavia mekanismeja. (Krenz, 2006)

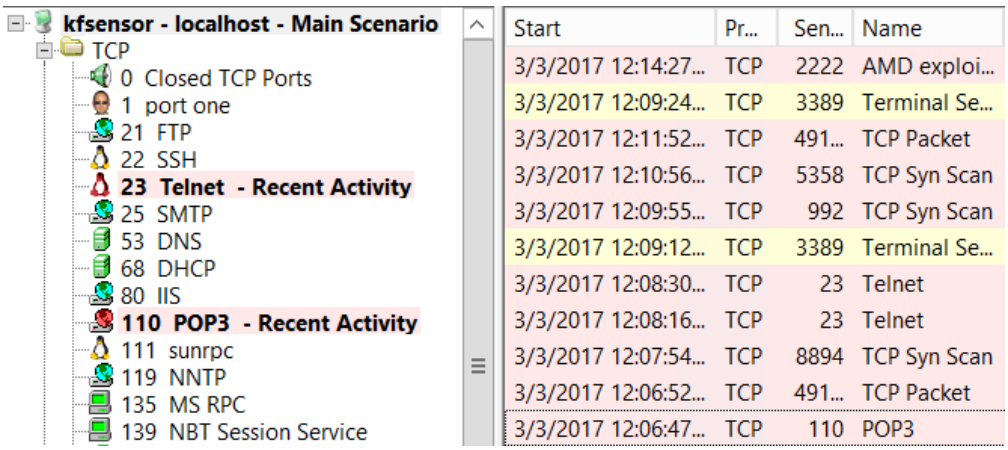
3.3 KFSensor-ohjelmiston ominaisuuksia

KFSensor on Keyfocus nimisen yrityksen vuonna 2003 julkaisema kaupallinen vuorovaikutukseltaan keskitason Windows hunajapurkki, joka toimii myös IDS:nä.

KFsensor on oletuksena konfiguroitu monitoroimaan ICMP:tä sekä kaikkia TCP ja UDP portteja. KFsensor voi vastata yhteyksiin usealla tavalla, Tavallisesta porttien kuuntelusta ja yksinkertaisista palveluista monimutkaisiin palvelusimulaatioihin. HTTP-protokollan simuloimiseen KFSensor emuloi Microsoftin web-palvelimen toimintaa ja vastaa valideihin ja invalideihin pyyntöihin. KFsensor pystyy myös ylläpitämään verkkosivua ja suorittamaan monimutkaisia toimintoja, kuten asiakkaan puolen välimuisti pyyntöjä vaikeuttaen hyökkääjän yrityksiä tunnistaa järjestelmä hunajapurkiksi. (Keyfocus, n.d.)

KFSensor simuloi järjestelmäpalveluita OSI-mallin sovellustasolla. Tämä antaa ohjelmalle täyden pääsyn Windowsin luontaisiin turvallisuusmekanismeihin ja verkkokirjastoihin tällä saadaan vähennettyä murtautumisen riskiä, kun järjestelmään ei tuoda omia mukautettuja ajureita ja IP pinoja. KfSensoria käyttävää järjestelmää voidaan kohdella tavallisena palvelimena verkossa ja monimutkaisia muutoksia reitittimiin ja palomuureihin ei tarvitse tehdä. (Keyfocus, n.d.)

Oletuksena KFSensor tallentaa kaiken interaktioista saadun tiedon ohjelman omiin lokeihin. Erilaisille tapahtumille voidaan asettaa oman väri koordinoidut vakavuus asteet. (Kuva 6.) Tämän lisäksi KFSensor tukee kerätyn tiedon muuttamisen raportiksi, jolloin KFSensor tallentaa kaiken tiedon tietokantaan ja pyrkii visualisoimaan tämän ylläpitämälleen verkkosivulle.



Kuva 6. KFsensorin käyttöliittymä (kuvakaappaus)

3.4 Järjestelmien ylläpito

Hunajapurkkien valitsemisen jälkeen ensimmäinen tehtävä oli etsiä alusta, johon järjestelmät voidaan asentaa. Aluksi Hunajapurkkeja yritettiin pyörittää fyysisillä laitteilla yksityisessä lähiverkossa, mutta tämä osoittautui hankalaksi vaihtoehdoksi, joten uusia ratkaisuja alettiin etsiä.

Muiden käyttäjien ratkaisuja tutkimalla päästiin lopulta siihen lopputulokseen, että hunajapurkki olisi parasta asentaa jonkin palveluntarjoajan virtuaalipalvelimelle. Eri VPS:stä tarvitsi ensin karsia pois ne, jotka eivät halunneet ylläpitää hunajapurkkeja omassa palvelussaan, jonka jälkeen hinta/laatu suhteen vertailun avulla päädyttiin Amazon AWS:ään.

AWS:ssä käyttöönotetut virtuaalikoneet sijaitsevat Singaporessa. Resursseja näille virtuaalikoneille on annettu AWS:n T2.micro instanssin mukaisesti 1 virtuaaliprosessori, 1 gigatavu keskusmuistia ja n. 8 gigatavua tallennustilaa. Käyttöjärjestelminä kahdessa ensimmäisessä koneessa toimivat Ubuntu 14.04 ja kolmannessa Windows Server 2012 R2.

3.4.1 Amazonin pilvipalvelu alusta AWS

AWS eli ”Amazon web service” on Amazonin ylläpitämä pilvipalvelualusta, josta asiakkaat voivat ostaa erilaisia pilvilaskenta palveluita. Amazonin palvelut toimivat 16 eri sijainnissa ympäri maailmaa. AWS tarjoaa yli 70 erilaista palvelua, kuten laskentatehoa, tallennustilaa, tietokantoja, mobiilikehitystyökaluja ja analytiikkaa.

Projektissa tarvitaan kolmea virtuaalikonetta ja yhtä erillistä tietokantaa, joten AWS palveluista käytetään Amazon Elastic Compute Cloud:ia eli Amazon EC2:ta ja Amazon Relational Database Serviceä eli RDS:ää.

Amazon tarjoaa käyttäjilleen ilmaista tasoa joistain sen palveluista. Tämä taso on hyvin rajattu, mutta sopiva työssä tarvittavien palvelinten pyörittämiseen. Ilmaisella tasolla yhtä EC2 instanssia voisi pyörittää kuukauden ilmaiseksi, mutta käytössä on kolme erillistä instanssia, joten ilmaisen tason rajat ylittyvät. (Kuva 7.)

Top Free Tier Services by Usage — View all

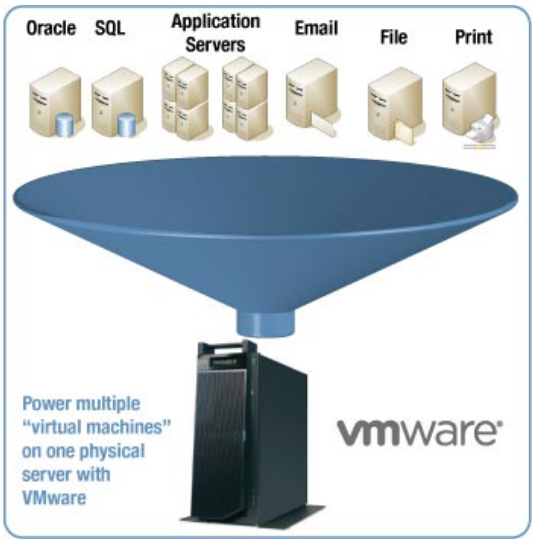
Service	Month-to-date usage/Free Tier limit	Forecasted month-end usage/Free Tier limit
EBS - I/Os	100.00% (2,000,000.00/2,000,000 IOs)	155.56% (3,111,111.11/2,000,000 IOs)
EC2 - Linux	78.00% (585.00/750 Hrs)	121.33% (910.00/750 Hrs)
EBS - Volumes	62.87% (18.86/30 GB-Mo)	97.80% (29.34/30 GB-Mo)
EC2 - Windows	37.87% (284.00/750 Hrs)	58.90% (441.78/750 Hrs)
KMS - Requests	0.05% (9.00/20,000 Requests)	0.07% (14.00/20,000 Requests)

Kuva 7. AWS:n arvioitu käyttö (kuvakaappaus)

3.4.2 Virtualisointi

Virtualisointi on tapa yksinkertaistaa IT-resurssien hallintaa ja kasvattaa niiden käyttöä keräämällä olemassa olevat teknologiaresurssit ja jakamalla ne uudelleen, näin parantaen yrityksen valmiutta vastata kasvavaan kysyntään.

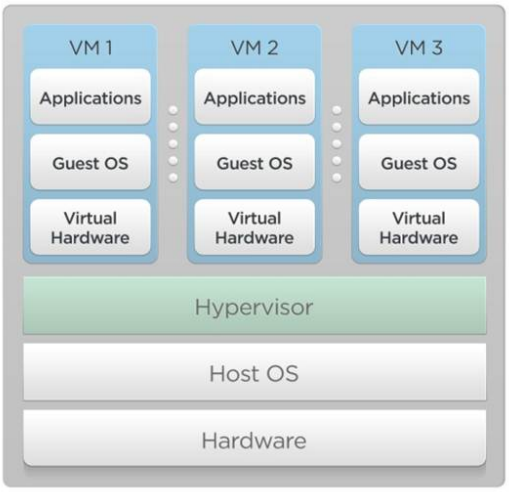
Palvelimissa ja tietoverkoissa virtualisointia käytetään ottamalla yksittäinen fyysinen laite ja muuntamalla se toimimaan kuin se olisi usea laite. (Kuva 8.) Näin saadaan kasvatettua hankittujen laitteiden hyötykäyttöä ja tehokkuutta samalla vähentäen kustannuksia ja tarvetta fyysisille laitteille. Tämä mahdollistaa myös useiden yksittäisten laitteiden resurssien yhdistämisen, jolloin saatavilla olevat resurssit voidaan esittää suurempana kokonaisuutena palvelimille ja ohjelmistoille.



Kuva 8. Palvelimien yhdistäminen virtualisoinnilla (Nasi n.d.)

Virtualisoinnilla on niin monia yleisiä käyttökohteita, että sen käyttöönotto voi olla organisaatioille sekavaa. Työn kannalta relevantteja ja teknologiaa helpoiten selittäviä tekniikkoja on kuitenkin datakeskuksiin kohdistuvat virtualisointitekniikat, joista tärkeimpiin kuuluu palvelin virtualisointi. (Golden, 2011)

Palvelinvirtualisoinnissa jonkin fyysisen laitteen resurssit virtualisoidaan loogisiksi varoiksi, joita käytetään koneelle asennettavien virtuaalikoneiden käyttämiseen. Usein virtualisointi toteutetaan erilaisilla Hypervisor -ohjelmilla, jotka luovat virtuaalikoneen esimerkiksi emuloimalla kokonaisen laitteistoympäristön. Tähän tyhjään virtuaalikoneeseen ladataan jokin käyttöjärjestelmä, joka pyytää isäntäjärjestelmältä resursseja. Hypervisor vastaanottaa nämä pyynnöt ja vastaa niihin tarjoamalla koneelle asetettuja resursseja. (Golden, 2011) (Kuva 9.)



Kuva 9. Esimerkki Hypervisorin toiminnasta jonkin olemassa olevan käyttöjärjestelmän päällä. (Cloudvortex n.d.)

Palvelinvirtualisointi voidaan vielä jakaa kolmeen osaan jotka ovat: käyttöjärjestelmä virtualisointi, laitteiston emulointi ja paravirtualisaatio. Tekniikat eroavat toisistaan siinä, miten virtuaalikoneelle annetaan resursseja ja millä tasolla virtualisointi tapahtuu, mutta perusteet ovat kuitenkin samanlaiset.

3.5 **Hunajapurkit Dionaea & Cowrie**

Ensimmäiselle käyttöönotetulle virtuaalikoneelle asennettiin kaksi hunajapurkkia: Dionaea ja Cowrie. Koneelle käyttöjärjestelmäksi valittiin Ubuntu 14.04 LTS, koska käytössä olevien ohjeiden mukaan tämä tulisi olemaan uusin Ubuntu-versio, jolla halutut hunajapurkit tulevat toimimaan.

3.5.1 Dionaea käyttöönotto

Dionaean asennusta on helpotettu viime aikoina, eikä koko ohjelmaa tarvitse enää itse lähteä pala palalta kasaamaan vaan ohjelmasta on luotu kokonainen asennuspaketti. (Liite 1.)

Dionaean asennuspakettia ei löydy oletusohjelmistokirjastosta, vaan sen hankkimiseksi tarvitsee Ubuntun oletuslistaan lisätä ”Honeypot PPA packagers”-tiimin yksityinen paketti arkisto, johon tämä tiimi lisää Dionaean uusimmat neutraalit versiot.

Dionaea toimii hyvin oletuskonfiguraatioilla. Muutamia muutoksia voi järjestelmään kuitenkin tehdä, kuten uusien tapahtuman käsittelijöiden lisääminen oletuksena olevien rinnalle ja lokien luonnin höllentäminen, jotta järjestelmä ei tuota ylimääräisiä varoituksia.

Dionaean tuottaman tiedon monimutkaisuuden takia asennetaan järjestelmään myös DionaeaFR visualisoimaan kerättyä tietoa. (Liite 2.).

3.5.2 Cowrie käyttöönotto

Cowrie-asennus joudutaan toteuttamaan manuaalisesti, joten kaikki vaadittavat ohjelmat asennetaan komentorivin kautta ja itse Cowrie kloonataan Githubista. Cowrien asennusta varten luodaan myös uusi Cowrie-niminen käyttäjä, jotta oikeat käyttäjäoikeudet syntyvät suoraan. (Liite 3.)

Cowrien asennusta varten voidaan myös konfiguroida pythonin virtuaaliympäristö, jotta mahdolliset riippuvuus ristiriidat eivät estä muista ohjelmistoja toimimasta. Virtuaaliympäristön vaatimat ohjelmat löytyvät Cowrien juurihakemistosta requirements.txt tiedoston sisältä. Nämä ohjelmat asennetaan pythonin omalla pip-nimisellä pakettienhallinta järjestelmällä.

Cowrie kansion juuresta löytyvä konfiguraatiotiedosto toimii oletusasetuksilla, mutta datan visualisoinnin takia kyseisestä tiedostosta otettiin käyttöön MySQL-moduuli, jonka avulla Cowrien pitäisi tallentaa hyökkäyksistä tuleva tieto lokien lisäksi myös paikalliseen MySQL tietokantaan. Konfiguraatioista muokattiin myös valepalvelimen oletusnimeä.

Cowrien juuren konfiguraatiotiedoston lisäksi löytyy ohjelmasta vielä kaksi sijaintia, joiden sisältöä on hyvä muokata käyttöönoton yhteydessä. Ensimmäinen on *data/userdb.txt* tiedosto, josta voidaan lisätä toimivia käyttäjiä joita hyökkääjät voivat käyttää valetiedostojärjestelmään kirjautuessaan (Kuva 10.). Toinen kiinnostava kansio on nimeltään *honeyfs/*. Tähän kansioon voidaan siirtää tiedostoja, jotka hyökkääjä tulee näkemään järjestelmään kirjautuessaan.

```
root:x:!root
root:x:!123456
root:x:*
richard:x:*
richard:x:fout
wbadmin:x:tuuli
```

Kuva 10. Malli userdb tiedoston sisällöstä (kuvakaappaus)

Cowrie on SSH hunajapurkki, joten järjestelmän oletus SSH-portti käytiin vaihtamassa /etc/sshd/sshd_config tiedostosta. Cowrie ei voi myöskään käyttää oletusporttia 22, koska ohjelma ei voi käyttää alhaisia portteja ilman pääkäyttäjän oikeuksia, joten portti 22 pitää ohjata uudelleen porttiin 2222 käyttämällä iptables-palomuuriohjelmaa.

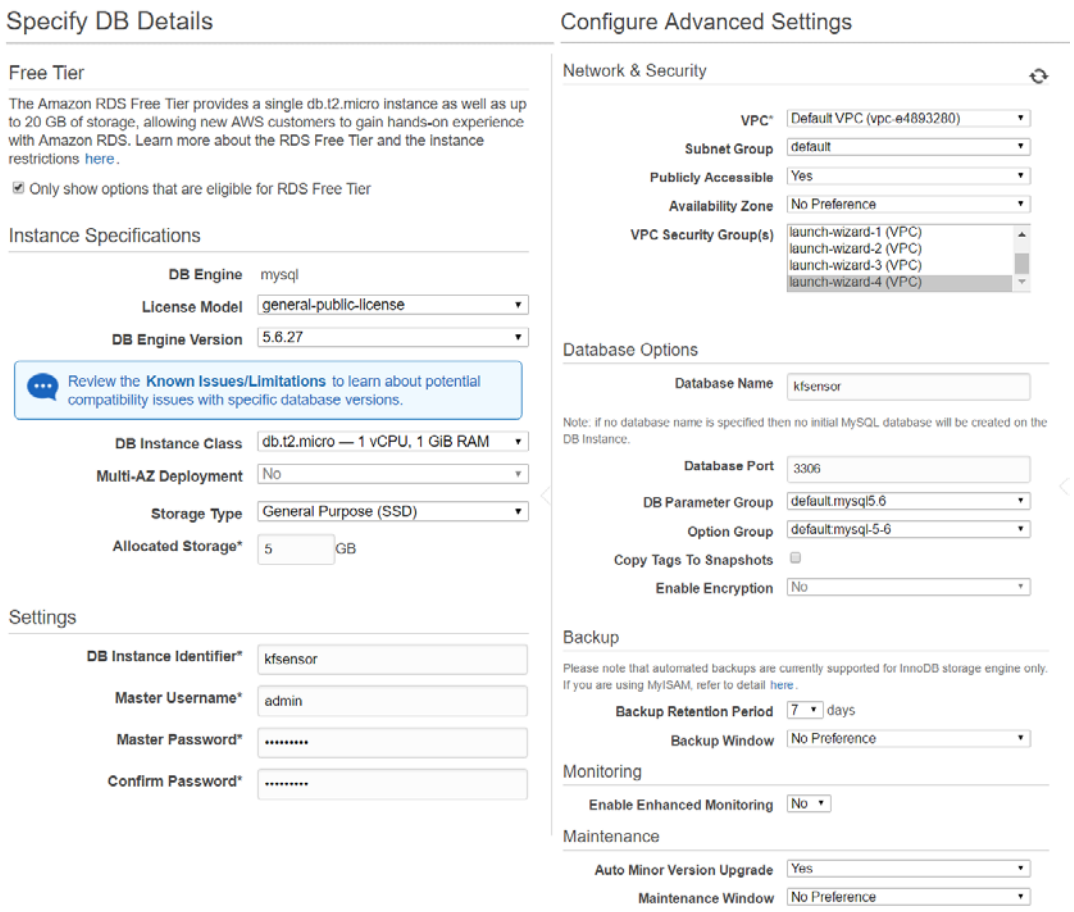
Cowrien tietojen visualisoinniksi asennettiin vielä kippo-graph, jonka asennuksen kulku löytyy liitteestä 4.

3.6 Hunajapurkki Kfsensor

KFsensorista asennetaan 30-päivän kokeiluversio, jossa on kuitenkin ohjelman kaikki ominaisuudet käytössä. Asennus on suoraviivainen ja käyttäjän tarvitsee vain hakea ja suorittaa ohjelman msi-tiedosto. KFsensorin ainoa vaatimus on WinPcap nimisen työkalun asennus, tätä tarvitaan kuitenkin vain, jos KFsensoria halutaan käyttää esimerkiksi Wireshark-ohjelman rinnalla.

Kfsensorin sisäisen raportointitoiminnon käyttämiseen tarvitaan, joko paikallinen tietokanta tai etätietokanta. Tietokanta on toteutettu Amazonin RDS palvelulla, joka ylläpitää toiminnon tarvitsemaa tietokantaa AWS pilvialustalla.

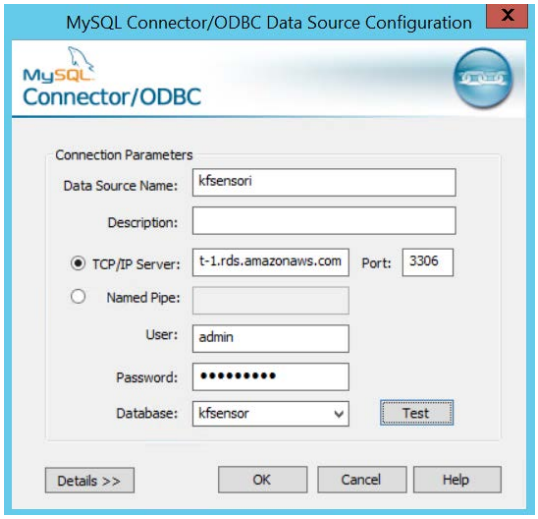
Tietokannan instanssin käyttöönotossa valitaan palvelun vaihtoehdoista sopiva. Kfsensor tukee vain MSSQL- ja MySQL-tietokantoja, joten työhön valittiin MySQL-tietokanta aikaisempien käyttökokemuksien pohjalta. Tietokantatyypin valitsemisen jälkeen muita pakollisia muutoksia olivat vain tietokantainstanssin nimeäminen, ylläpitäjätunnuksien nimeäminen ja salasanan asetus ja itse tietokannan nimeäminen (Kuva 11.).



Kuva 11. RDS-tietokannan asetukset (kuvakaappaus)

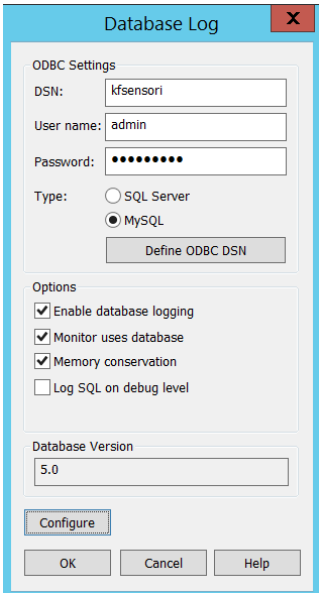
Tietokannan luomisen jälkeen tarvitsee KFsensor-koneelle asentaa ODBC (Open Database Connectivity) -ajurit, jotka mahdollistavat yhteyden ottaminen etänä toimivaan MySQL-tietokantaan.

Viimeisenä vaiheena, ennen yhteyden ottamista, on ODBC DSN:n (Data source name) määrittäminen KFsensorin sisällä. DSN voidaan määrittää KFsensorin polusta Settings -> Database log -> Define ODBC DSN, jossa ohjelmalle ilmoitetaan uuden datalähteen nimi, aiemmin luodun tietokannan osoite, Tietokannan pääkäyttäjätunnukset ja luodun tietokannan nimi (Kuva 12.).



Kuva 12. ODBC DSN:n asetusten määrittäminen. (kuvakaappaus)

DSN:n määrittämisen jälkeen ohjelmaa käskytetään tallentamaan kaikki uusi tieto käytössä olevaan tietokantaan ja monitoroimaan tätä tietokantaa, jotta ohjelma voi luoda ajankohtaisia raportteja (Kuva 13.).



Kuva 13. KFsensorin tietokannan viimeisten asetusten määrittäminen. (kuvakaappaus)

4 KERÄTTY TIETO

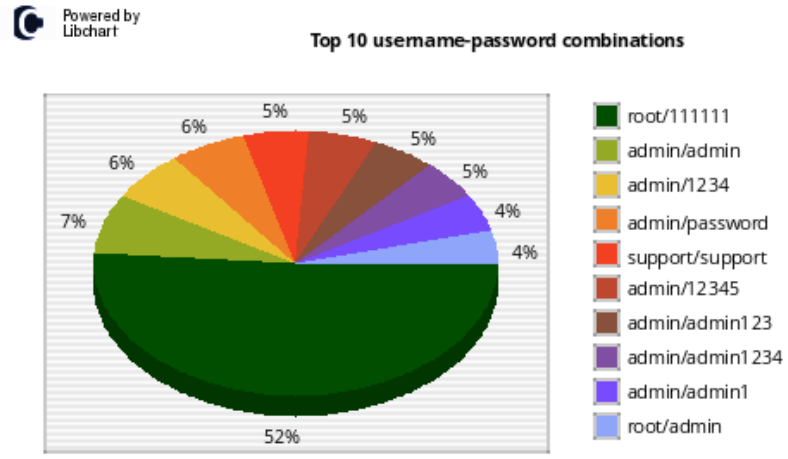
Lopuksi hunajapurkkeja ajavia koneita otetiin käyttöön alussa suunnitellun kahden sijaan kolme, koska Cowrien uusin versio ei täyttänyt MySQL-tietokantaa. Tämän takia uudelle Ubuntu-koneelle asennettiin Cowrien alkuperäinen versio Kippo, joka on toiminnallisuudeltaan hyvinkin samanlainen mutta joitain ominaisuuksia puuttuu.

Dionaea & Cowrie tulokset

Dionaea ja Cowrie asennettiin ensimmäiselle käyttöönotetulle hunajapurkille. Tämä järjestelmä otettiin käyttöön 5.2.2017 ja tuloksia tutkiessa se on ollut päällä n. 570 tuntia.

Järjestelmän käytössä oloaikana Dionaea havaitsi 51018 yhteyttä 11552 eri IP-osoitteesta. Suurin osa yhteyksistä hylättiin ja kirjattiin blackhole palvelun alle, mutta lähes kaikkiin Dionaean muista palveluista kohdistui myös yhteyksiä. DionaeaFR:n visualisoimaa tietoa löytyy liitteestä 5.

Käynnissä oloaikanaan Cowrie sai tuotettua 448 tekstilokitiedostoa ja 16 json lokitiedostoa. Cowrie osuuden teknisten ongelmien takia kerätyn datan visualisointi oli hankalampaa. Kerätyistä tiedoista hyökkääjien yrittämät käyttäjänimet ja salasanat saatiin python skriptillä siirrettyä MySQL -tietokantaan. 570 tunnin jälkeen Cowrie oli saanut 14112 kirjautumisyritystä. (Kuva 14.)



Kuva 14. Cowrien yhteysyritysten salasana yhdistelmien jakautuminen (kuvakaappaus)

Kippo tulokset

Kippo hunajapurkkia käyttävä virtuaalikone otettiin käyttöön 9.2.2017, kun huomattiin että Cowrie hunajapurkki ei alkanut täyttää MySQL-tietokantaa tiedolla. Järjestelmä on käyttöönotosta lähtien kerännyt tietoa n. 482 tuntia. Kippon visualisoimaa tietoa löytyy liitteestä 6.

Järjestelmä on kirjannut 10706 kirjautumisyritystä 743:sta erillisestä IP-osoitteesta. Yrityksistä 8588 onnistui ja 2188 epäonnistui kirjatumaan valetiedostojärjestelmään. Näistä onnistuneista yrityksistä saatiin kerättyä 802 komentoa, jotka voitiin jakaa 67 uniikkiin komentoon. Lopulta nämä komennot sisälsivät 13 yritystä ladata haitallisia tiedostoja valetiedostojärjestelmään. Kuvassa 15 on esimerkki ulkopuolisen käyttäjän komennoista ja epäonnistuneesta yrityksestä ladata järjestelmään tiedostoja.

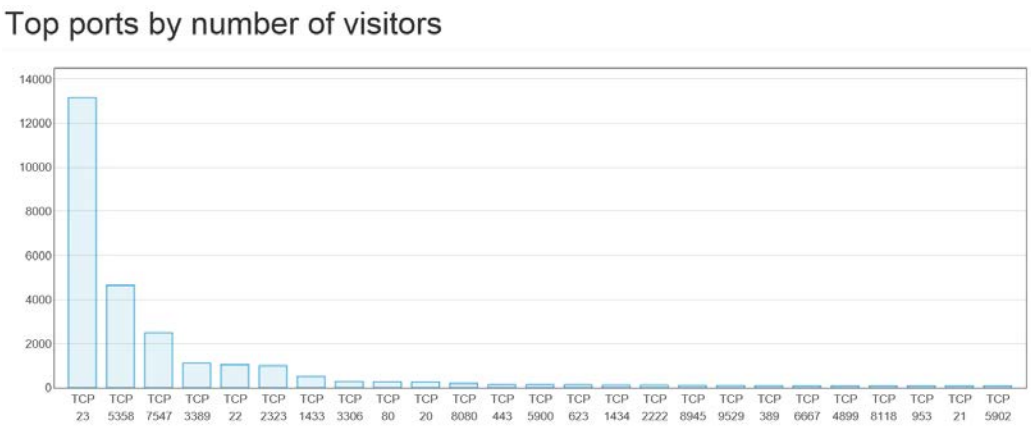
```
root@svr03:/$ service iptables stop
bash: service: command not found
root@svr03:/$ wget http://118.123.116.251:44244/Y8
Sorry, SSL not supported in this release
root@svr03:/$ chmod 0755 /root//Y8
chmod: cannot access /root//Y8: No such file or directory
root@svr03:/$ nohup /root//Y8 > /dev/null 2>&1 &
nohup: ignoring input and appending output to `nohup.out'
root@svr03:/$ chmod 777 /Y8
chmod: cannot access /Y8: No such file or directory
```

Kuva 15. Kippo:n tallentama komento sarja, jonka Kippo-graph onnistui visualisoimaan (kuvakaappaus)

KFsensor tulokset

Viimeisenä Kfsensoria ajava Windows 2012 palvelinkone otettiin käyttöön 6.2.2017 ja ehti olla käytössä n. 556 tuntia. Käynnissä oloaikana järjestelmä kirjasi sitä tutkineen 28427 käyttäjän, jotka aiheuttivat järjestelmässä 172809 tapahtumaa.

Suurin osa järjestelmän kirjanneista kävijöistä keskittyi Telnet-porttiin 23 (Kuva 16) ja hunajapurkinkirjanneista tapahtumista suurin osa kohdistui porttiin 3389 joka on Windowsin terminaalipalvelu, jota käytetään etätyöpöytäyhteyden avaamiseen. Muita KFSensorin luomia taulukoita löytyy liitteestä 7.



Kuva 16. KFsensorin visualisoima data järjestelmän vierailijoiden käyttämistä porteista

5 YHTEENVETO

Työn alkuperäinen ajatus oli tutustua hunajapurkkeihin ja niiden mahdollisiin käyttötarkoituksiin tietoturvassa ja lopulta ottaa käyttöön jokin hunajapurkki-järjestelmä. Aiheeseen olisi voinut tutustua vielä syvällisemmin, mutta työn edetessä on huomattu, että aiheeseen liittyvä kirjallisuus on hyvinkin vanhentunutta ja teknologia on enemmänkin mainittu vain jonkin suuremman aiheen sivussa. Mahdollisia hunajapurkkeja löytyi kyllä suuria määriä esimerkiksi Honeynet project:in nettisivulta, mutta suuri osa näistä oli vanhentuneita tai muuttaneet nimiään kehittäjien vaihtuessa. Vanhentuneiden ja itsekoottavien ohjelmien käytön ongelmat myös ilmenivät, kun ohjelmia asentaessa järjestelmät eivät tahtoneet toimia, koska ohjelman vaatimusohjelmien versiot olivat liian uusia ja jotkin toiminnot olivat ajan saatossa poistuneet käytöstä. Tämä aiheutti sen, että aikaa tuli käyttää paljon ohjelmien korjaamiseen, joka oli samalla turhauttavaa, mutta myös hyödyllistä, jos samanlaisia ongelmia ilmenee muissa tilanteissa tulevaisuudessa.

Ongelmista huolimatta uskon onnistuneeni vastaamaan tässä työssä esittämiini tutkimuskysymyksiin ainakin jollain tasolla, mutta olisin voinut miettiä käytännön osan toteutusta vielä lisää. Sain kerättyä n. kuukauden ajalta tietoa, mutta järjestelmien teknisten ongelmien ja joidenkin mahdollisten konfiguraatio virheiden takia osa tiedosta on arvoltaan heikohkoa.

Hunajapurkkeihin kohdistuneiden hyökkäyksien määrä oli oletettua suurempi ja tiedonkeräyksellä saatiin ainakin todistettua, että ulkomaailmassa on tahoja, jotka haluavat vahingoittaa tietojärjestelmiä niiden tiedonarvosta riippumatta. Vaikka suurimman osan projektin hyökkäyksistä olisi voinut torjua tavallisilla palomuuri käytännöillä saatiin ainakin tilastotietoa yleisimmistä hyökkäyksien kohteista.

Näin työn valmistuttua uskon pitäväni ainakin yhden näistä hunajapurkeista vielä toiminnassa, koska yksi kuukausi on hyvin lyhyt aika minkäänlaisen suuremman tietomäärän keräämiseksi ja tähän mennessä näyttää siltä, että mitä pidempään järjestelmät ovat pystyssä sitä suurempi määrä päivittäisiä yhteysyrityksiä niitä kohtaan suuntautuu. Uskon myös, että kyseisillä ohjelmistoilla on jokin paikka tietoverkkojen tietoturvassa, vaikka tämä ei olekaan niin suuri kuin jotkut olisivat toivoneet.

LÄHTEET

Cloudvortex, n.d.
Haettu 20.2.2017 osoitteesta
http://www.cloudvortex.com/cloud-hosting/web-applications/parallels/parallels-cloud-server

Dionaea, n.d.
Haettu 17.1.2017 osoitteesta
https://dionaea.readthedocs.io/en/latest/index.html

González Diego, 2004. Building a GenII Honeynet Gateway. Haettu 26.2.2017 osoitteesta
https://honeynet-es.org/papers/honeywall/

Golden Bernard (2011). Virtualization For Dummies, 3rd HP Special Edition. Hoboken: John Wiley & Sons, Inc.

Holz Thorsten ja Provos Niels, (2007). Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Boston: Addison-Wesley Professional

Kellep Charles, n.d. an overview of honeypots
Haettu 25.2.2017
https://www.blacksintechnology.net/an-overview-of-honeypots/

Keyfocus n.d.
http://www.keyfocus.net/kfsensor/help/
Haettu 1.3.2017

Krenz Mark, 2006.
Haettu 23.2.2017 osoitteesta
https://support.suso.com/supki/SSH_Tutorial_for_Linux

Nasi n.d. VMware Server Consolidation
Haettu 10.3.2017 osoitteesta
http://www.nasi.com/vmware_server-consolidation.php

Oosterhof Michael, 2016.
Haettu 26.2.2017
http://www.micheloosterhof.com/cowrie/

Pathan Al-Sakib Khan, (2014). The State of the Art in Intrusion Prevention and Detection. Boca Raton: Auerbach Publications.

Rouse Margaret, 2016.
Haettu 23.2.2017 osoitteesta
http://searchsecurity.techtarget.com/definition/Secure-Shell

Spitzner Lance, (2003). Honeypots: Tracking Hackers. Boston: Addison-Wesley Professional

Spitzner Lance, 2010 Honeypots: Are They Illegal?
https://www.symantec.com/connect/articles/honeypots-are-they-illegal
Haettu 26.2.2017 osoitteesta

Dionaea Asennus Liite 1

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:honeynet/nightly
sudo apt-get update
sudo apt-get install dionaea
sudo service dionaea start
```

DionaeaFR asennus Liite 2

apt-get install python-pip python-netaddr build-essential python-dev git
pip install Django==1.8
pip install pygeoip
pip install django-pagination
pip install django-tables2
pip install django-compressor
pip install django-htmlmin
pip install django-filter
cd /opt/
wget https://github.com/benjiec/django-tables2-simplefilter/archive/master.zip -O django-tables2-simplefilter.zip
unzip django-tables2-simplefilter.zip
mv django-tables2-simplefilter-master/ django-tables2-simplefilter/
cd django-tables2-simplefilter/
python setup.py install
cd /opt/
git clone https://github.com/bro/pysubnettree.git
cd pysubnettree/
python setup.py install
cd /opt/
wget http://nodejs.org/dist/v0.8.16/node-v0.8.16.tar.gz
tar xzvf node-v0.8.16.tar.gz
cd node-v0.8.16
./configure
make
make install
npm install -g less
npm install -g promise
cd /opt/
wget https://github.com/RootingPuntoEs/DionaeaFR/archive/master.zip -O DionaeaFR.zip
unzip DionaeaFR.zip
mv DionaeaFR-master/ DionaeaFR

cd /opt/
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz

wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
gunzip GeoLiteCity.dat.gz
gunzip GeoIP.dat.gz
mv GeoIP.dat DionaeaFR/DionaeaFR/static
mv GeoLiteCity.dat DionaeaFR/DionaeaFR/static
cp /opt/DionaeaFR/DionaeaFR/settings.py.dist /opt/DionaeaFR/DionaeaFR/settings.py

Vaihdetaan Sqlite tietokannan polku oikeaksi settings.py tiedostossa

sudo vim /opt/DionaeaFR/DionaeaFR/settings.py
mkdir /var/run/dionaeafr #for DionaeaFR's pid file
cd /opt/DionaeaFR/
python manage.py collectstatic
python manage.py runserver 0.0.0.0:8001

Alla olevassa tiedostossa rivillä 90 on vanhentunut merkintä, joka voi aiheuttaa virheen, kun palvelu yritetään käynnistää. Kyseinen sana tulee vaihtaa seuraavanlaiseksi

sudo vim /opt/DionaeaFR/Web/forms/connection.py
IpAddressField -> GenericIPAdressField

Cowrie asennus Liite 3

sudo apt-get install git python-virtualenv libmpfr-dev libssl-dev libmpc-dev libffi-dev build-essential libpython-dev python2.7-minimal
sudo adduser --disabled-password cowrie
sudo su - cowrie
git clone http://github.com/micheloosterhof/cowrie
cd cowrie
virtualenv cowrie-env
source cowrie-env/bin/activate
pip install -r requirements.txt
cp cowrie.cfg.dist cowrie.cfg
./start.sh cowrie-env

Kippo-graph Asennus Liite 4

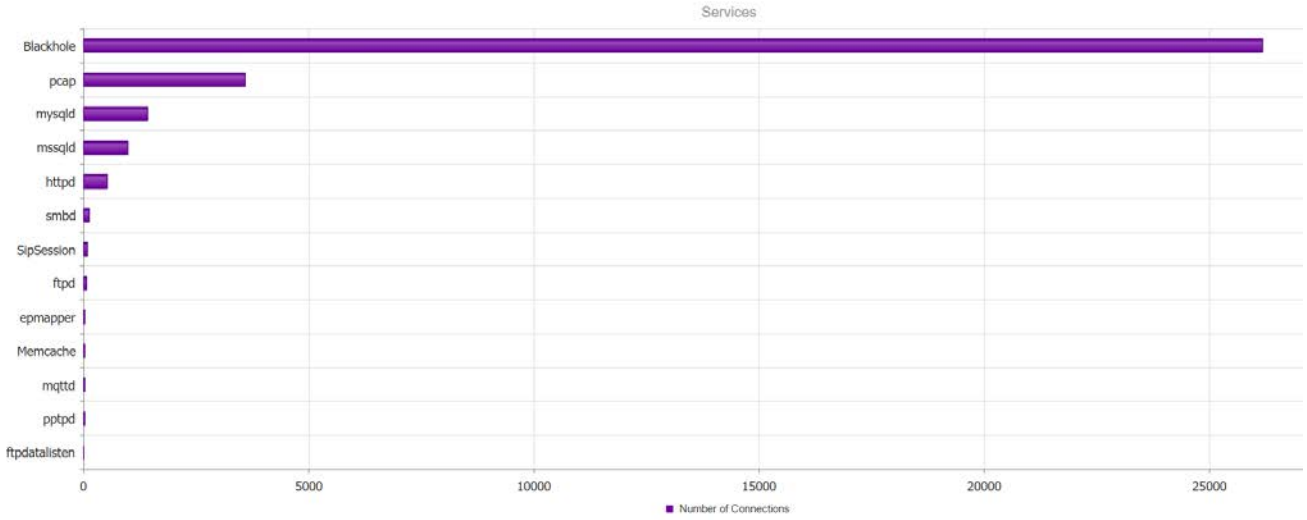
apt-get update && apt-get install -y libapache2-mod-php5 php5-mysql php5-gd php5-curl
sudo service apache2 restart
sudo git clone https://github.com/ikoniaris/kippo-graph.git
mv kippo-graph /var/www/html
cd /var/www/html
sudo tar zxvf kippo-graph

sudo mv kippo-graph- kippo-graph
cd kippo-graph
chmod 777 generated-graphs
cp config.php.dist config.php

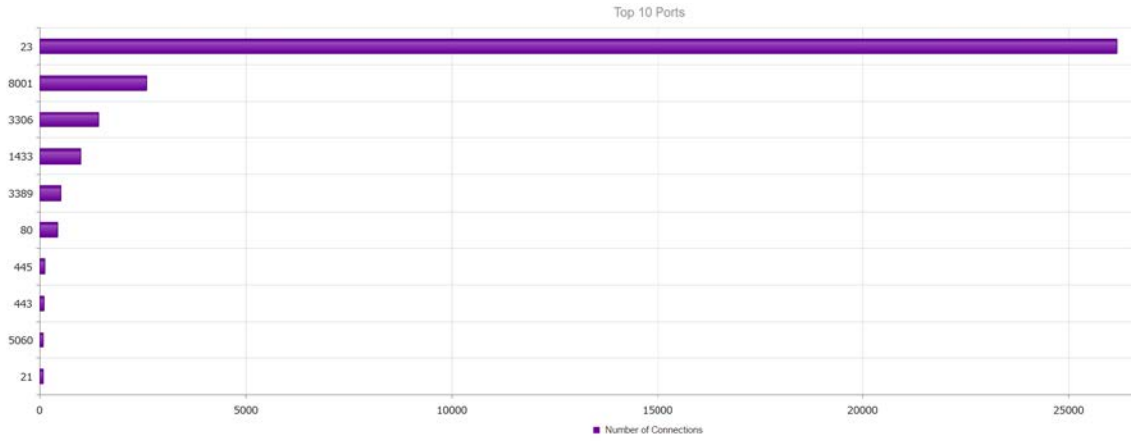
Lopuksi tuee vaihtaa config.php tiedoston arvot oikeiksi

sudo vim config.php

Dionaean visualisoitu tietoa Liite 5



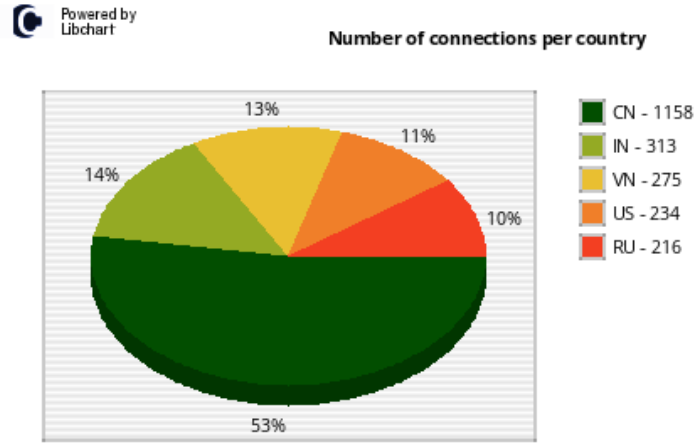
Kuva 17. Dionaea järjestelmän yhteyksien jakautuminen eri palveluihin (kuvakaappaus)



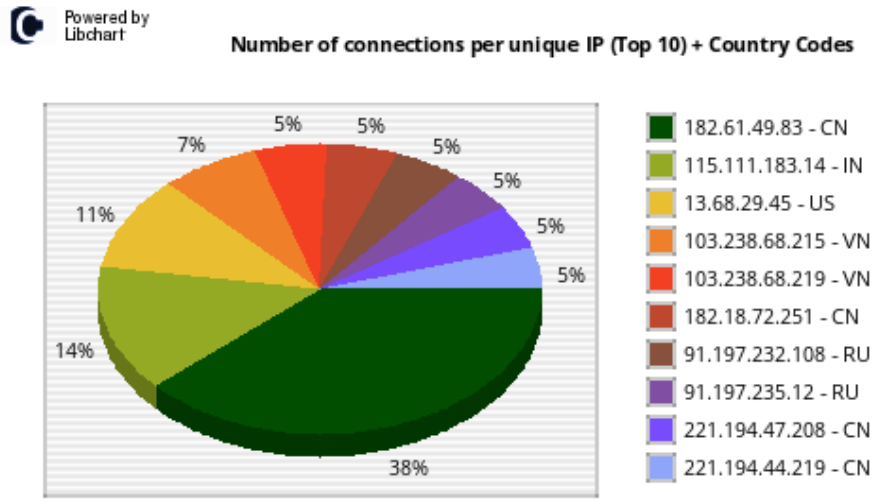
Kuva 18. Dionaean yhteyksien jakautuminen eri portteihin (kuvakaappaus)

Kuva 19. Dionanean yhteyksien lähteet kartalla. (kuvakaappaus)

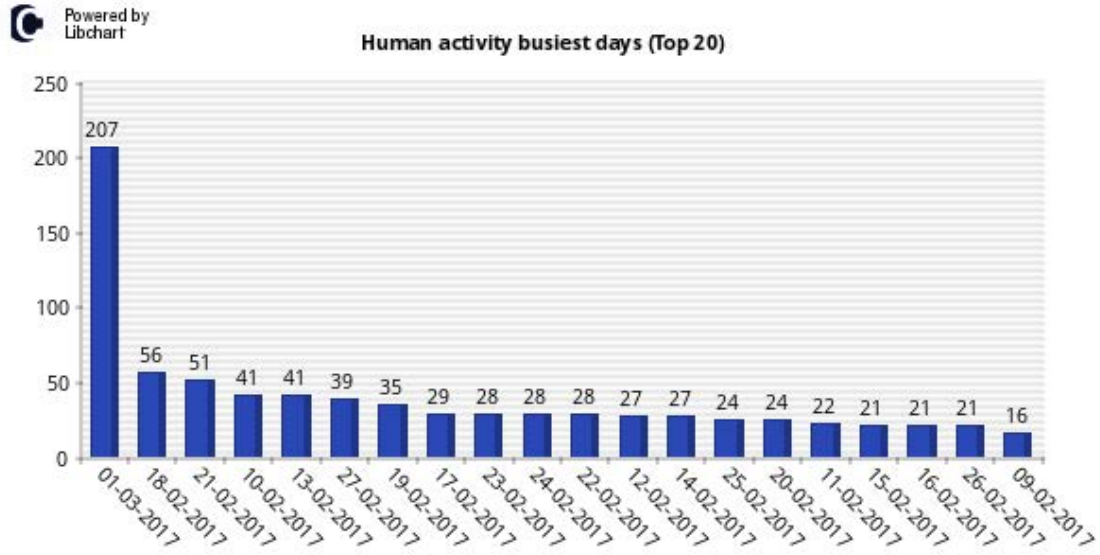
Kippon visualisoitua tietoa Liite 6



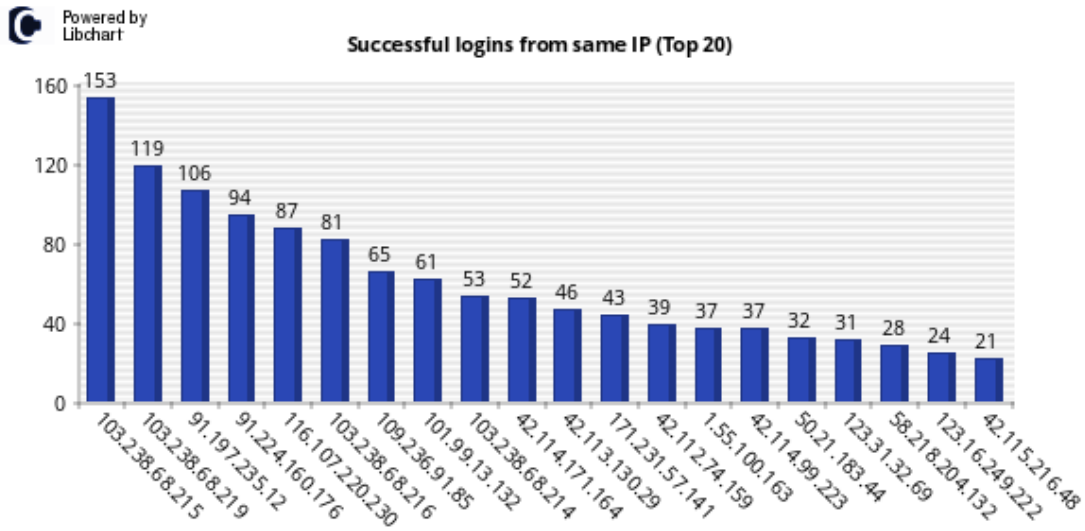
Kuva 20. Kippo yhteyksien jakautuminen eri maiden välille. (kuvakaappaus)



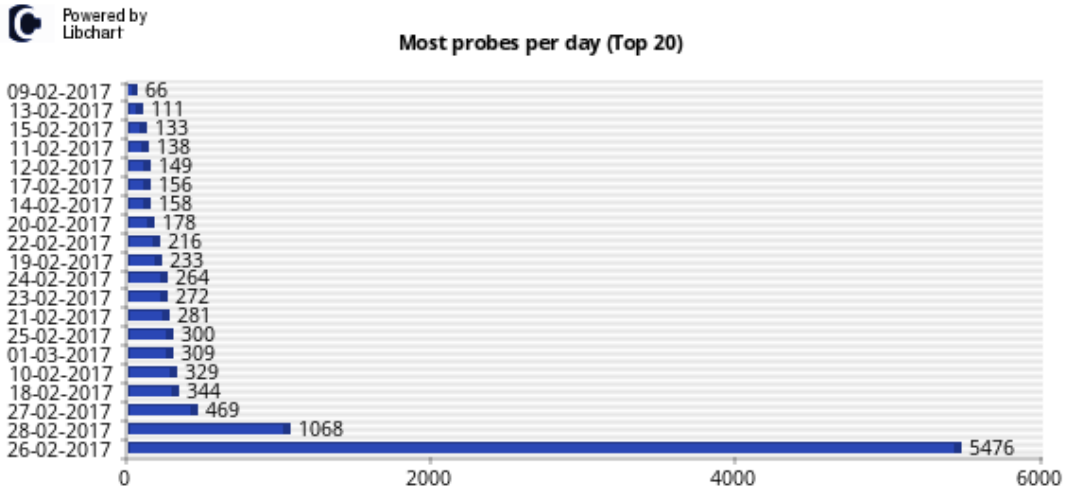
Kuva 21. Kippo yhteyksien jakautuminen eri IP-osoitteiden välille. (kuva-kaappaus)



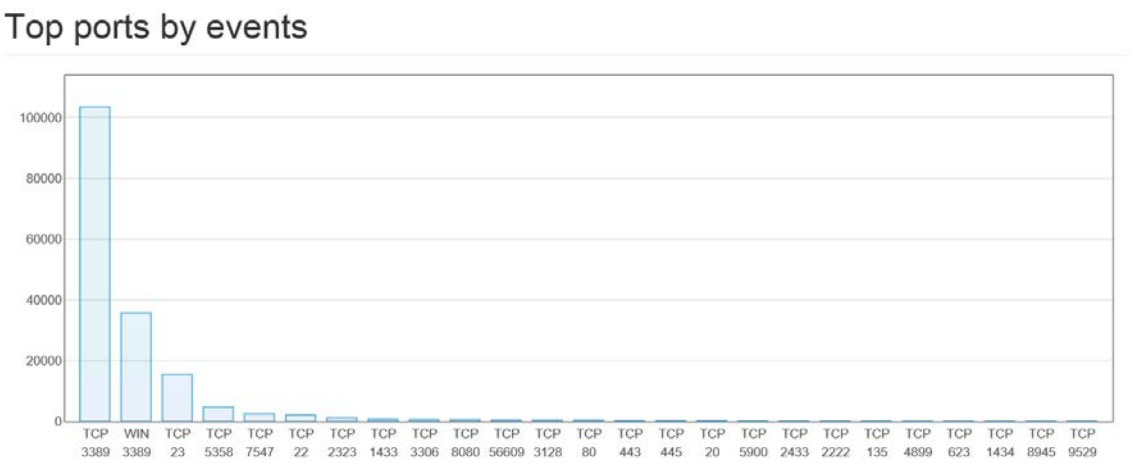
Kuva 22. ¨Top 20 "Ihmistoiminta" kippo koneella jaettuna päiville. (kuva-kaappaus)



Kuva 23. Top 20 Onnistuneet kirjautumisyritykset jaettuna IP-Osoitteille (kuvakaappaus)



Kuva 24. Top 20 päivät jolloin tapahtui eniten luotauksia kippo koneella (kuvakaappaus)



Kuva 25. KFSensorin tapahtumat jaettu porttikohtaisesti (kuvakaappaus)

Kuva 26. Kävijöiden luomat tapahtumat jaettuna eri IP-osoitteille. (kuvakaappaus)