Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2020

Nicolas Samáneh

TIETOTURVAUHKIEN HAVAITSEMINEN YARA-SÄÄNNÖILLÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2020 | 42 sivua

Nicolas Samáneh

TIETOTURVAUHKIEN HAVAITSEMINEN YARA-SÄÄNNÖILLÄ

Tietoturvauhkien havaitsemiseksi on yleisesti luonnehdittu riittävän virustorjuntaohjelman asentaminen. Perinteisten virustorjuntaohjelmien ennaltaehkäisevä toiminta perustuu uuden tiedoston tiivisteen vertaamiseen jo haitallisiksi todettujen tiedostojen tiivisteisiin. Useissa viimeaikaisissa kyberhyökkäyksissä käytetyt ennalta tuntemattomat haittaohjelmat ovat kasvattaneet tarvetta uhkien tehokkaampaan havainnointiin. Eräs uusimmista keinoista uhkien tunnistamiseen on YARA-sääntöjen käyttöönotto IT-ympäristössä. YARA-sääntöjen toiminta perustuu kuvauksien luomiseen haitallisista tiedostoista ja ne sisältävät merkkijonoja ja ehtoja. Kuvaukseksi riittää yksinkertaisimmillaan vain yksi sana, mutta kehittyneimmät YARA-säännöt sisältävät useita merkkijonoja ja monimutkaisia ehtolausekkeita.

Opinnäytetyön tavoitteena oli tutkia YARA-sääntöjä yhtenä keinona haittaohjelmien ennaltaehkäisevään tunnistamiseen ja luoda automatisoitu YARA-skannaus laitteeseen. Työn lähtökohtana toimi kuviteltu tilanne, jossa ennalta tuntematon haittaohjelma oli saatu asetettua yrityksen laitteelle. Haittaohjelmaksi valittiin Windows -toimialueen tiedusteluun tarkoitettu komentosarja.

Työ toteutettiin analysoimalla YARA-sääntöjen rakennetta, luomalla uusia sääntöjä ja optimoimalla sääntöjä virheellisten havaintojen minimoimiseksi. Työssä tutkittiin myös sääntöihin lisättäviä määritteitä ja niiden vaikutuksia YARA-työkalun tuottamiin tuloksiin. Opinnäytetyössä luotujen YARA-sääntöjen pohjalta kehitettiin ajastettu YARA-skannaus, jolla tutkimuksessa käytetty Windows-laite saatiin tarkastettua säännöllisin väliajoin. Skannauksen käyttämät YARA-säännöt luotiin työn tutkimusosuudessa analysoimalla tiedusteluun tarkoitettua haittaohjelmaa.

Opinnäytetyön tavoitteessa onnistuttiin ja lopputuloksena on toimiva YARA-skannaus, joka kerää tulokset skannausajankohdan perusteella nimettyyn tekstitiedostoon. Johtopäätöksenä todettiin, että YARA-säännöillä saadaan tehokkaasti luotua merkkejä toimialueen vaarantumisesta riippumatta siitä, onko uhka peräisin organisaation sisä- vai ulkopuolelta.

ASIASANAT:

kyberhyökkäys, tietoturva, haittaohjelma, virustorjuntaohjelma, YARA, forensiikka

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2020 | 42 pages

Nicolas Samáneh

IDENTIFYING CYBER SECURITY THREATS WITH YARA RULES

Cyber security threats have developed over the course of the 21st century forcing organizations to implement more and more secure solutions regarding their business. Upon detecting a compromise, the effectiveness of remedial actions often determines the future of the organization. Even more security solutions have emerged to help organizations secure their infrastructure and mitigate possible threats. One of the newest additions to the field of threat detection solutions are YARA rules. YARA rules are a way of identifying and classifying possible threats regarding an organization's domain.

The objectives of this thesis were to introduce YARA rules as a method of improving the effectiveness of proactive cyber security threat detection and to compose rules for detecting internal reconnaissance of a Windows Active Directory domain. Testing of the composed rules was carried out by using the YARA tool.

The research was carried out by examining the logic behind YARA rules and by creating rules for testing purposes. The created rules were tested and further developed to decrease the amount of false detections. A batch file intended for Active Directory reconnaissance was analysed and rules were created to detect certain command-line commands found inside the batch file.

The goal of the thesis to implement YARA rules as another layer of security was successful. As a result of the research, a scheduled YARA-scan was successfully implemented to a Windows computer. The scheduled scan utilized four different YARA rules based on the batch file analysis.

In conclusion, YARA provides an efficient addition to existing cyber security solutions, providing a method of identifying cyber security threats regardless of whether a threat has originated from inside or outside of an organization.

KEYWORDS:

cyber security, incident response, forensics, cyber attack, YARA, anti-virus

SISÄLTÖ

KUVAT

LYHENTEET

AD	Active Directory, Microsoft Windows -toimialueen kokonaisuus, joka sisältää tiedot käyttäjistä, resursseista ja laitteista.
DFIR	Digital Forensics and Incident Response, tietoturvaloukkauksen tutkinta.
Hash	Tiiviste eli tiedoston hajautusarvo. Esimerkiksi virustorjuntaohjelmat käyttävät tiedostojen tiivisteitä vertaamalla niitä haittaohjelmien tiivisteisiin.
IoC	Indicator of Compromise. Merkki saastumisesta.
IR	Incident Response eli tietoturvapoikkeaman hallinta. Tietoturvapoikkeamiin varautuminen, reagointi ja niistä toipuminen.
Komentojono	Kokonaisuus, jonka muodostaa ennalta peräkkäin kirjoitetut komennot.
Kybersota	Valtioiden välinen sodankäynti kyberavaruudessa.
PS	PowerShell. Windowsin komentotulkki.
Tietomurto	Tunkeutuminen suojattuun tietojärjestelmään tai suojatussa tietojärjestelmässä olevan tiedon oikeudeton tarkastelu.
Tietoturvaloukkaus	Tietojärjestelmään käsiksi pääseminen luvatta.
VT	VirusTotal. Haitallisten tiedostojen ja verkkotunnusten analysoinnin mahdollistava verkkosivusto.
YARA	Victor Alvarezin kehittämä sääntömuoto haittaohjelmien kuvaamiseen ja tunnistamiseen.

1 JOHDANTO

Suurissa toimialueissa tietomurron tutkintaan tarvitaan tehokkaita työkaluja tartunnan laajuuden ja kriittisyyden havaitsemiseen. Uudelle vuosikymmenelle siirryttäessä on varauduttava uudentyyppisiin uhkiin ja organisaatioiden on kehitettävä sekä implementoitava uusia keinoja tietoturvauhkien minimoimiseksi. Tietoturvauhka on käsitteenä todella laaja, tässä opinnäytetyössä tietoturvauhkana esitellään toimialueen tiedusteluun tarkoitettu komentosarja.

YARA-säännöt tarjoavat yhden ratkaisun uhkien tunnistamiseen. YARA-työkalu mahdollistaa koko toimialueen tarkastamisen tiettyjen uhkien varalta. YARA-sääntöjen syntaksi on yksinkertainen ja dynaamisesti muokattavissa tunnistamaan tiettyjä merkkijonoja tai heksadesimaalilukuja annetuista kohteista.

Erityisesti kohdennetuissa kyberhyökkäyksissä käytettyjen haittaohjelmien tunnistaminen on vaikeaa, sillä haittaohjelmien tiivisteitä ei välttämättä ole vielä lisätty virustorjuntaohjelmistojen tiivistetietokantaan. YARA-säännöillä voidaan luoda indikaattoreita tietystä haittaohjelmasta vain yhtä haitalliseksi todettua tiedostoa analysoimalla. Opinnäytetyön tutkimuksen taustana toimii hypoteettinen tilanne, jossa tiedustelukomentosarja on saatu organisaation laitteelle, mutta virustorjuntaohjelmien tiivistetietokannoista ei löydy tiedoston tiivistettä.

Opinnäytetyössä esitellään YARA-sääntöjen toimintaa ja hyödyntämistä mahdollisten tietoturvauhkien havaitsemiseen. Työssä esitellään osa YARA-työkalun eri toiminnoista ja työkalua suoritetaan Windowsin komentoriviltä. Opinnäytetyön aiheeksi valittiin YARA-säännöt, sillä tästä aiheesta ei löytynyt aikaisempia suomeksi kirjoitettuja opinnäytetöitä.

Työn päämäärinä olivat YARA-sääntöjen syntaksin ymmärtäminen, YARA-työkalun käytön opettelu ja ajastetun YARA-skannaustehtävän luominen. Työssä luotu ajastettu YARA-skannaus saatiin onnistuneesti määritettyä tarkistamaan laitteen PowerShell-komentohistoria ja käynnistyskansio. Ajastettu skannaus asetettiin suoriutumaan viikoittain ja tulokset saatiin kerättyä skannausajankohdan perusteella nimettyyn tekstitiedostoon.

Työssä käydään YARA-sääntöjen syntaksi läpi jakamalla esimerkkisääntö osiin ja esittelemällä säännön eri komponentit. Sääntöjen toiminnan ymmärtäminen ei vaadi aiempaa taustaa tietoturva-alalta. Sääntöihin lisätään teksti- sekä heksamerkkijonoja.

YARA-sääntöjä luovat ja jakavat niin tietoturva-alan ammattilaiset kuin tietoturvayhteisöjen jäsenet. Sääntöjä hyödyntäviä työkaluja kehitetään jatkuvasti lisää. GitHub-versionhallintasivusto mahdollistaa työkalujen jakamisen ja sisältää paljon oppaita YARA-sääntöjen käyttöön.

Opinnäytetyön luvussa 2 esitellään tietoturvauhan ja tietomurron määritteet. Luvussa 3 esitellään erilaisia haittaohjelmatyyppejä, niiden käyttötarkoituksia ja historiaa. Opinnäytetyön luku 4 keskittyy YARA-sääntöjen toiminnan ja luomisen esittelyyn. Luvussa 5 luodaan Active Directory -toimialueen korkean luokan käyttäjäryhmien tiedusteluun tarkoitettu tiedustelukomentosarja ja laaditaan YARA-sääntöjä sen tunnistamiseen. Sääntöjä testataan ja jatkokehitetään tunnistamaan analysoitu tietoturvauhka yhä tarkemmin.

Luvussa 6 luodaan ajastettu YARA-skannaus PowerShell-työkalun komentohistorian ja käynnistyskansion tarkistamiseksi. YARA-skannausta varten luodaan MS-DOS-komentojono, joka määritetään tulostamaan skannaustulokset mahdollisimman käyttäjäystävällisesti. Luvussa 7 esitellään YARA-sääntöjä hyödyntäviä tietoturvatyökaluja. Kaupalliset tietoturvatyökalut sisältävät tuhansia YARA-sääntöjä, jota päivitetään jatkuvasti.

2 TIETOTURVAUHAT

Ihmisellä on ollut läpi historian tarve puolustaa omaa reviiriään. Puolustettavaan reviiriin on sisältynyt yleisimmin tietoa, materiaa tai muita ihmisiä. Karkeasti sanottuna niin kauan kuin on ollut jotain puolustettavaa, on ollut myös uhkia. Ajan kuluessa ihmisen hallinnoimaan tietoon ja materiaan kohdistuneet uhat ovat luonnollisesti muuttuneet. Tämän päivän kontekstissa uhilla tarkoitetaan yleisimmin omaan kotiin, henkilökohtaisiin tietoihin tai -materiaan kohdistuvia uhkia.

Modernilla aikakaudella suuren osan niin ihmisten henkilökohtaisten tietojen kuin yrityksien sensitiivisten tietojen siirtyessä verkossa oleviin palveluihin tietoturva ja siihen kohdistuvat uhat ovat nousseet pinnalle yhä useammin. Käsitteet, kuten tietoturva, tietoturvauhka, haittaohjelma ja tiedustelu tulivat 2000-luvun aikana monille tutuiksi.

2.1 Tietoturvauhan määritelmä

Tietoturvauhka ei yksiselitteisesti tarkoita yritykseen hyökkäävää ulkopuolista tahoa. Tietoturvauhiksi voidaan luokitella esimerkiksi huonosti suojatut laitteet, tietojenkalasteluviestit ja virukset. Eri tahot määrittelevät tietoturvauhan eri tavalla. Kyberturvallisuuden sanaston mukaan tietoturvauhka on ”mahdollisesti toteutuva haitallinen tapahtuma tai kehityskulku, joka kohdistuu tietoturvaan ja toteutuessaan vaarantaa sen” (TSK 52 2018).

Tietoturvauhkien minimoimiseksi kehitetään jatkuvasti uusia keinoja. Kuten yksinkertaisin tapa oman kodin turvaamiseen on laadukas ulko-oven lukko, tietoturva-alalla vahvaa salasanaa voidaan pitää vastaavana varotoimenpiteenä. Muita tapoja uhkien minimoimiseen ovat esimerkiksi laitteiden ohjelmistopäivitysten asentaminen, käyttöoikeuksien hallinta ja tietoturvapoikkeamien havainnointi.

Yrityksen tietoturvan heikko ylläpito saattaa johtaa yrityksen joutumiseen tietomurron kohteeksi. Tietomurron paljastuessa käynnistyy tutkinta- ja palautumisprosessi. Hyökkäysten kehittyessä yhä monivaiheisimmiksi, aikainen havainnointi lisää yritysten kykyä tunnistaa mahdolliset uhat, niin sisäiset kuin ulkoisetkin. Myös yritysten yhä monimutkaisempi IT-infrastruktuuri vaikeuttaa poikkeamien aikaista havaitsemista ja tutkintaa.

2.2 Tietomurron määritelmä rikoslaissa

Suomen rikoslain 38. luku käsittää tieto- ja viestintärikokset. Luvun pykälät koskevat tietomurron tapauksessa niin tietomurron toteuttajaa kuin tietomurron kohdetta.

Tietomurrosta määrätään rikoslain 8. §:ssä (Rikoslaki 368/2015) seuraavasti:

> Joka käyttämällä hänelle kuulumatonta käyttäjätunnusta taikka turvajärjestelyn muuten murtamalla oikeudettomasti tunkeutuu tietojärjestelmään, jossa sähköisesti tai muulla vastaavalla teknisellä keinolla käsitellään, varastoidaan tai siirretään tietoja tai dataa, taikka sellaisen järjestelmän erikseen suojattuun osaan, on tuomittava tietomurrosta sakkoon tai vankeuteen enintään kahdeksi vuodeksi.
>
> Tietomurrosta tuomitaan myös se, joka tietojärjestelmään tai sen osaan tunkeutumatta
>
> 1) teknisen erikoislaitteen avulla tai
>
> 2) muuten teknisin keinoin turvajärjestelyn ohittaen, tietojärjestelmän haavoittuvuutta hyväksi käyttäen tai muuten ilmeisen vilpillisin keinoin
>
> oikeudettomasti ottaa selon 1 momentissa tarkoitetussa tietojärjestelmässä olevasta tiedosta tai datasta.
>
> Yritys on rangaistava.
>
> Tätä pykälää sovelletaan ainoastaan tekoon, josta ei ole muualla laissa säädetty ankarampaa tai yhtä ankaraa rangaistusta.

2.3 Tietomurrot maailmanlaajuisena ilmiönä - NotPetya

Maailmanlaajuisesti tunnetuimmat tietomurrot liittyvät kiristyshaittaohjelmien leviämiseen tietoteknisissä infrastruktuureissa. Kiristyshaittaohjelmahyökkäysten aiheuttamien vahinkojen määrä kasvaa, mitä suurempi organisaatio on kohteena. Kiristyshaittaohjelmien aiheuttamat rahalliset vahingot ovat vähintään kaksinkertaistuneet vuoden 2019 viimeisellä neljänneksellä. (Mathews 2020.)

Hyvänä esimerkkinä voidaan pitää vuoden 2017 hyökkäystä Ukrainan infrastruktuuria vastaan. Hyökkäyksessä käytetyn haittaohjelman nimeksi annettiin NotPetya. Kyberhyökkäys koostui kiristyshaittaohjelmasta, Mimikatz-nimisestä tunnistetietojen haravoimistyökalusta sekä Shadow Brokers-hakkeriryhmän käsiin saamasta NSA:n

kehittämästä ”EternalBlue” -haavoittuvuuden hyväksikäyttömenetelmästä. Microsoft oli julkaissut tietoturvapäivityksen haavoittuvuuden korjaamiseksi jo ennen hyökkäystä, mutta Mimikatz-työkalun avulla päivittämättömistä laitteista saatujen tunnistetietojen avulla NotPetya pääsi leviämään päivitettyihinkin laitteisiin. (Greenberg 2018.)

Haittaohjelman levitys aloitettiin Ukrainassa toimivien yritysten käyttämän M.E.Doc -tilitysohjelman päivityspalvelimen kautta. Vahinkoja koitui monille globaaleille yrityksille satojen miljoonien dollarien edestä. Yksi pahimmat vahingot kärsineistä yrityksistä oli tanskalainen rahtiyritys Maersk. (Greenberg 2018.)

Haittaohjelman nimeksi annettiin NotPetya, sillä haittaohjelma sisälsi Petyaa muistuttavan kiristyshaittaohjelman, mutta kiristyshaittaohjelma sisällytettiin vain tietoturvatutkijoiden hämäämiseksi. Haittaohjelman tarkoitus oli vain tehdä mahdollisimman monta tietoteknistä laitetta käyttökelvottomaksi. (Greenberg 2018.)

NotPetya-hyökkäystä luonnehditaan kybersodankäynnin sotatoimena Ukrainaa vastaan. Kyberhyökkäys aiheutti yhteensä yli 10 miljardin dollarin vahingot. (Greenberg 2018.)

2.4 Tietoturvapoikkeaman hallinta – Incident Response

Tietoturvapoikkeaman hallinta käsittää toimet poikkeamiin varautumiseen, reagointiin, vahinkojen rajoittamiseen ja niistä toipumiseen (TSK 52 2018). Tarkasti laadittu tietoturvapoikkeamasuunnitelma auttaa tehostamaan poikkeamien hallintaa ja vähentää organisaation mahdollisia rahallisia menetyksiä (Voigt 2018).

National Institute of Standards and Technology (NIST) on laatinut vuonna 2012 tietoturvapoikkeaman hallintaa varten julkisessa jaossa olevan oppaan Computer Security Incident Handling Guide (SP 800-61), joka määrittää muun muassa poikkeaman hallinnan eri vaiheet. Vaiheita ovat valmistautuminen (preparation), havainnointi ja analysointi (detection and analysis), rajaaminen, hävittäminen ja palautuminen (containment, eradication, and recovery) ja poikkeamasta palautumisen jälkeiset toimet (post-incident activity).

Tietoturvaloukkauksen tutkintaa luonnehditaan termillä Digital Forensics and Incident Response. DFIR-prosessissa keskitytään tietoturvaloukkausepäilyyn liittyvien todisteiden keruuseen ja analysointiin. Tärkeimpinä todisteina voidaan pitää niin päätelaitteilta kuin verkkoliikenteen mahdollistavilta laitteilta saatavat muistit ja lokitiedot. (TSK 52 2018.)

Tietoturvaloukkauksen tutkinta on monivaiheinen prosessi. Tietoturvaloukkauksen tapahtuessa tulisi yleisten periaatteiden mukaisesti olla tuhoamatta todisteita, kuten kotimurron sattuessa asukkaan tulee olla siivoamatta jälkiä. Tietomurron tapahtuessa vaarantunut laite tulee pitää käynnissä todisteiden menetyksen ehkäisemiseksi. Yleensä vaarantuneesta laitteesta luodaan muistivedos forensista tutkintaa varten. Mikäli organisaatio havaitsee tietyn laitteen vaarantuneen, ensimmäinen askel tietomurron leviämisen estämiseksi on laitteen kytkeminen pois verkosta.

3 HAITTAOHJELMAT

Haittaohjelmaa pidetään yleiskäsitteenä ei-toivotuille tietokoneohjelmille, jotka aiheuttavat suunnittelemattomia tapahtumia tietoteknisissä laitteissa (TSK 52 2018). Ensimmäisen konseptin tietokoneviruksesta loi unkarilaissyntyinen matemaatikko John von Neumann 1940-luvun lopulla. Ensimmäinen oikea haittaohjelma tunnistettiin kuitenkin vasta 1970-luvulla, kun ARPANET-verkossa havaittiin Creeperiksi nimetty haittaohjelma. (Matthews 2019.)

Haittaohjelmiksi luokitellaan niin kyberhyökkäyksissä käytetyt ohjelmat, joita ei ole aikaisemmin nähty luonnossa, kuin vakavuudeltaan alhaisimmalla tasolla olevat mainosohjelmat (Love 2018).

3.1 Haittaohjelmatyypit

Haittaohjelmia voidaan jaotella niiden alkuperän, tarkoituksen ja leviämismekaniikan kautta. Yleisimmät tavat haittaohjelmien leviämiseen ovat sähköpostien liitetiedostot ja internetselailu (Viestintävirasto 2015). Haittaohjelmatyyppejä on monia ja eri lähteet luokittelevat ne eri tavoin.

Tietokonevirus (Virus) on yksi haittaohjelmatyypeistä, vaikka termejä haittaohjelma ja virus käytetään usein tarkoittamaan samaa asiaa. Kuten biologiset viruksetkin, tietokonevirukset eivät leviä itsestään, vaan tarvitsevat ”kasvualustaksi” muita tietokoneohjelmia ja käyttäjän interaktiota levitäkseen. Tietokonevirukset asettavat haitallisen koodinsa muihin suoritettaviin tietokoneohjelmiin. Ennen internetiä virukset levisivät infektoituneiden levykkeiden avulla. Esimerkkejä eri virustyypeistä ovat makrovirukset sekä monimotoiset (polymorfiset) virukset. Makrovirukset leviävät infektoitujen Microsoft Office-liitetiedostojen avulla sähköpostilla. Polymorfiset virukset muuttavat lähdekoodia joka kerralla, kun virus kopioi itseään vaikeuttaakseen antivirusohjelman havainnointikykyä. (Fruhlinger 2019.)

Troijalainen (Trojan horse) on haittaohjelmatyyppi, jonka leviäminen edellyttää tietoteknisen laitteen käyttäjältä toimia. Troijalaiset naamioidaan usein viattomiksi ohjelmiksi, jotka suorittaessa ajavat tai asentavat käyttäjän huomaamatta haitallisia ohjelmia, komentoketjuja tai takaovia laitteeseen. Troijalaistyyppisiä haittaohjelmia on useita, mutta

ylivoimaisesti vakavin tyyppi on etäältä ohjattava troijalainen (engl. Remote Access Trojan), joka antaa haitantekijälle pääsyn ja etähallintamahdollisuudet uhrin laitteeseen. Yleisimmin RAT-haittaohjelmia käytetään kohteen vakoiluun, pankkitietojen kaappaamiseen tai lisähaittaohjelmien asentamiseen. (Heinzman 2019.)

Haittaohjelmatyypin nimi tulee antiikin Kreikan mytologiasta Troijan sodasta, jossa kreikkalaiset antoivat suuren puisen hevosen troijalaisille sovinnon eleeksi. Puisen hevosen sisällä oli kuitenkin kreikkalaisia sotilaita, jotka tulivat esiin yöllä avaten kaupungin portit muille kreikkalaisille sotilaille. (Rouse 2019.)

Mainosohjelma (Adware) kuuluu ei-toivottujen sovellusten kärkeen. Mainosohjelmien tarkoituksena on näyttää käyttäjälle mainoksia tai vaihtaa hakutuloksia tienaten rahaa tekijälleen. Laitteen tietoturvan kannalta mainosohjelmat eivät ole niin suuria uhkia kuin esimerkiksi Troijalaiset. (Constantin 2019.)

Mainosohjelmia levittävät usein kyberrikolliset rahoittaakseen haitallisempia haittaohjelmakampanjoitaan. Yleisin mainosohjelmatyyppi on ilmaisohjelman lisänä asennettava selainlisäosa tai työkalupalkki. Mainosohjelmat rikkovat myös usein käyttäjän tietosuojaa keräten arkaluontoista informaatiota käyttäjästä ja laitteesta. (Constantin 2019.)

Mato (Worm) on yksi tyypillisimmistä haittaohjelmista. Madot leviävät laitteesta toiseen käyttämällä hyväksi jotakin haavoittuvuutta järjestelmissä. Madot saattavat sisältää “haittakuormia” (engl. payload) laitteiden vahingoittamiseksi. Matojen haittakuormien yleinen tarkoitus on varastaa sensitiivistä informaatiota tai poistaa tiedostoja laitteista. Mato eroaa muista haittaohjelmista sen kyvyllään levitä laitteesta toiseen ilman käyttäjän interaktiota. Madot voivat myös monistaa itseään ja piilottaa itsestään duplikaatteja tietojärjestelmiin vaikeuttaen haittaohjelman tunnistusta ja tutkintaa. (Fruhlinger 2019.)

Kiristyshaittaohjelman (Ransomware) toiminta perustuu saastuneen laitteen sisällön salakirjoittamiseen eli kryptaamiseen estäen pääsyn tiedostoihin ilman uniikkia salausavainta. Salakirjoitustoimien valmistuttua käyttäjälle esitetään yleensä tekstitiedosto, joka sisältää lunnasvaatimuksen. (TSK 52 2018.)

Ensimmäinen kiristyshaittaohjelma tunnistettiin vuonna 1989. Kyseinen ransomware-ohjelma nimettiin AIDS Troijalaiseksi. Haittaohjelmaa levitettiin saastuneilla levykkeillä WHO:n AIDS-konferenssin kävijöille. Levykkeen sisältämä haitallinen koodi piilotti käyttäjiltä kansiot ja tiedostot ja vaati 189 dollaria lähetettynä panamalaiseen postilokeroon tietojen takaisinsaamiseksi. (KnowBe4 2020.)

Kiristyshaittaohjelmien aiheuttamat vahingot yrityksille ovat lähes tuplaantuneet vuoden 2019 toisen vuosineljänneksen aikana. Tutkijoiden mukaan keskimääräinen häiriöaika nousi ensimmäisellä vuosineljänneksellä 7,3 päivästä 9,6-päivään. (Davis 2019.)

Vakoiluohjelma (Spyware) on haittaohjelma, joka kerää tietojärjestelmästä tietoa salaa haitantekijälle. Ohjelman tarkoitus on kerätä mahdollisimman paljon tietoa järjestelmästä ja sen käyttäjästä. Tyypillisin tartuntatapa on jonkin ilmaisohjelman asentaminen, jonka mukana vakoiluohjelma asentuu laitteelle. Vakoiluohjelma ei replikoidu, toisin kuin madot. (Rouse 2019.)

Takaportti (Backdoor) avaa nimensä mukaisesti haitantekijälle luvattoman pääsyn tietojärjestelmään ilman käyttäjän tietämystä. Yleisimmin takaovia havaitaan palvelimissa, joiden käyttöjärjestelmä tai palvelu sisältää haavoittuvuuden, avaten mahdollisuudet takaportin asentamiseen. Takaportteja on havaittu tietojärjestelmissä niin haitantekijöiden kuin viranomaistahojen asentamana. Monet troijalaistyyppiset haittaohjelmat avaavat pääsyn laitteelle takaportin avulla. (Zetter 2014.)

3.2 Haittaohjelmien tunnistaminen

Haittaohjelmien ehkäisyä ja tunnistamista varten on kehitetty useita ohjelmia ja toimintatapoja. Tunnistamista ja poistoa varten yleisin tapa on virustorjuntaohjelman asentaminen.

Virustorjuntaohjelman tehtävä on estää haittaohjelmien tunkeutuminen järjestelmään ja niiden poistaminen. Virustorjuntaohjelmat käyttävät yleisesti kahta tapaa haittaohjelmien tunnistamiseksi: vertaamalla tiedostojen tiivisteitä eli hajautusarvoja (hash) virustorjuntaohjelman tietokantaan ja tekemällä heuristista eli toiminnallista analyysiä tiedoston suorituksesta.

Jokaisella virustorjuntaohjelmalla on oma tiivistetietokanta (engl. signature base), johon tiedostojen tiivisteitä verrataan. Tiivistetietokanta sisältää myös tunnusmerkkejä tunnetuista haittaohjelmista.

Virustorjuntaohjelmien yleisin proaktiivisen havainnoinnin keino on ajastettu täysi virusskannaus. Uuden tiedoston ilmaantuessa laitteelle virustorjuntaohjelma toimii proaktiivisesti vertaamalla tiedoston tiivistettä tunnettujen haittaohjelmien tiivisteisiin. Mikäli

tiedoston tiiviste ei vastaa tietokannassa olevaa tiivistettä, tiedoston suoritusta ei keskeytetä. Virustorjuntaohjelma siirtyy heuristisen analyysin vaiheeseen.

Monet virustorjuntaohjelmat ajavat laitteelle ladattavan tiedoston ensin “hiekkalaatikossa” (engl. sandbox) IoC-indikaattorien löytämiseksi. Sandbox-analyysissä virustorjuntaohjelma vertaa tiedoston suoritusketjua tunnettujen haittaohjelmien suoritusketjuihin. Uusimmat virustorjuntaohjelmat ovat optimoitu suorittamaan proaktiiviset toimenpiteet käyttäjän huomaamatta.

4 YARA-SÄÄNNÖT

YARA on VirusTotalin Victor Alvarezin kehittämä työkalu. Työkalu on tehty auttamaan tietoturvatutkijoita erilaisten haittaohjelmatyyppien tunnistamiseen ja kuvaamiseen. YARA mahdollistaa kuvauksien (sääntöjen) luomisen teksti- tai binääriperusteisesti. YARA on kehittäjänsä mukaan lyhenne sanoista *Yet Another Recursive Acronym* tai *Yet Another Ridiculous Acronym.* Työkalu tukee montaa alustaa, kuten Windows, MacOS ja Linux. (VirusTotal 2016.)

YARA-työkalua voidaan käyttää joko komentoriviltä tai Python-skriptillä YARA-Python lisäosalla. Työkalusta on luotu useita valmiita ja kaupallisia versioita laaja-alaisempaa skannausta varten. Sääntöjen syntaksi muistuttaa C-kieltä.

YARA-sääntöjen päteminen perustuu säännölle asetettujen parametrien vertaamiseen annettuihin kohteisiin. Yleisimmin tällä tarkoitetaan annettujen sääntöjen ”osumista” annettuihin kohteisiin. Verrattaviksi parametreiksi voidaan asettaa tekstipohjaisia merkkijonoja (string), heksadesimaaliarvoja tai säännöllisiä lausekkeita.

Sääntöjen luomiseen ei tarvita erityistyökaluja, vaan sääntöjen luominen onnistuu millä tahansa tekstinkäsittelyohjelmalla. Säännöt tulee tallentaa .yar-päätteiseen tekstitiedostoon. (VirusTotal 2019.)

Useat yrityksille suunnatut tietoturvaratkaisut hyödyntävät YARA-sääntöjen tehokkuutta uhkien tunnistamisessa normaalien virustorjuntaohjelmien toiminnallisuuksien lisänä.

4.1 Merkkijonot

Merkkijono (string) on yksinkertaisimmillaan ASCII-koodattu merkkikokoriippuvainen merkkijono. Merkkijonon nimi alkaa dollarisymbolilla. Nimi voi sisältää sarjan aakkosnumeerisia merkkejä sekä alaviivoja. Merkkijonon nimillä erotetaan verrattavat merkkijonot toisistaan ja nimiä käytetään sääntöjen condition-osuudessa määrittämään sääntöjen ehdot.

Kuva 1 esittelee YARA-sääntöjen syntaksin mukaisia merkkijonoja. Merkkijonojen lisäksi voidaan asettaa määritteitä merkkijonon tulkinnan muuttamiseksi. Nämä määritteet tulee asettaa merkkijonon perään välilyönneillä eroteltuna. Esimerkkejä määritteistä ovat

nocase, ascii, wide ja fullword. Aakkosnumeeriset merkkijonot tulee kirjoittaa lainausmerkeillä rajattuna. (VirusTotal 2019.)

```
$teksti_merkkijono = ”merkkijono”
$heksa_merkkijono = { 74 65 73 74 }
```

Kuva 1. Esimerkkejä merkkijonoista

Yksi merkkijonon perään asetettavista määritteistä on fullword-määrite, joka takaa verrattavan merkkijonon osuvan vain, jos merkkijono esiintyy kohdetiedostossa muiden kuin aakkosnumeeristen merkkien rajaamana. Esimerkiksi fullword-määritteen sisältävä merkkijono ”verkkosivu” vastaa kohdetiedostosta löytyvää merkkijonoa ”www.verkkosivu.com”, mutta ei vastaa merkkijonoa ”www.minunverkkosivu.com”. (VirusTotal 2019.)

4.2 Säännölliset lausekkeet

Regular Expression eli säännöllinen lauseke on yksinkertaisilla operaatioilla muodostettu lauseke, joka kuvaa merkkijonojen joukkoa tai säännöllistä relaatiota (Tieteen termipankki 2020).

YARA-säännöissä säännöllisiä lausekkeita ympäröi vinoviivat merkkijonoja ympäröivien lainausmerkkien sijaan, kuten Perl-ohjelmointikielessä. Säännöllisten lausekkeiden perään voi asettaa samoja määritteitä kuin merkkijonojenkin. YARA:n versiosta 2.0 lähtien YARA on käyttänyt omaa säännöllisten lausekkeiden moottoriaan. (VirusTotal 2019.)

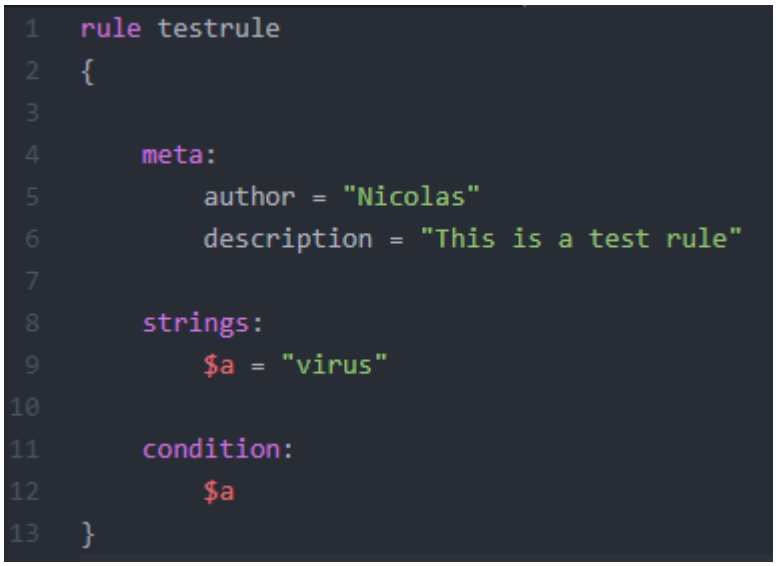
Yksinkertainen esimerkki säännöllisestä lausekkeesta YARA-säännössä on asteriskin käyttäminen. Asteriskin avulla säännön vertaaminen onnistuu kohdetiedoston sisältäessä YARA-säännön strings-osuuden merkkijonon merkeillä alkavan merkkijonon. Kuva 2 sisältää merkkijonon ja asteriskin. Vertaaminen onnistuu kohdetiedoston sisältäessä esimerkiksi merkkijonon ”testataan”, ”testit” tai ”testaus”.

```
$saannollinen = /test*/
```

Kuva 2. Säännöllistä lauseketta hyödyntävä merkkijono.

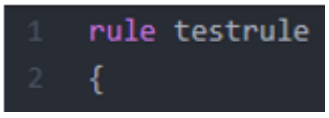
4.3 Sääntöjen rakenne

Säännöt koostuvat merkkijonoista (strings) ja ehdoista (condition) sekä vapaavalintaisesta meta-osuudesta. Ehto on ainoa tarpeellinen komponentti säännön toimintaan (VirusTotal 2019). Kuva 3 esittelee YARA-sääntöjen syntaksin mukaisesti laaditun säännön. Säännön eri komponenttien ymmärtämiseksi jaetaan sääntö osiin.



Kuva 3. Toimiva testisääntö.

YARA-sääntö alkaa rule-sanalla, jonka jälkeen säännölle annetaan tunniste (nimi). Säännön nimi seuraa C-kielen leksikaalista konventiota, nimi voi sisältää mitä vaan aakkosnumeerisia merkkejä, mutta ensimmäinen merkki ei voi olla numero. Säännön nimet ovat merkkikokoriippuvaisia ja rajoitettu 128 merkkiin. Itse sääntö alkaa aaltosulkeella (VirusTotal 2019). Esimerkissä säännön nimeksi on määritetty ”testrule” (Kuva 4).



Kuva 4. Säännölle annettu nimi.

Esimerkissä on säännön aloittavan aaltosulkeen jälkeen määritetty vapaavalintainen meta-osuus säännön tarkempaa kuvausta varten (Kuva 5). Testisäännössä annetut parametrit ovat ”author” – laatija sekä ”description” – kuvaus.

```
4       meta:
5           author = "Nicolas"
6           description = "This is a test rule"
```

Kuva 5. Meta-osuus.

Testisäännössä strings-osuuteen on sijoitettu yksi parametri $a, joka määrittää mitä merkkijonoa säännössä verrataan annettuun kohteeseen. Tässä säännössä on käytetty merkkijonoa ”virus”. Kuva 6 esittelee testisäännön strings-osuuden.

```
8       strings:
9           $a = "virus"
```

Kuva 6. Strings-osuus.

Pakollinen condition- eli ehto-osuus. Testisäännössä ehtona on täyttää strings-osuuden parametri $a. Kuva 7 esittelee testisäännön condition-osuuden.

```
11      condition:
12          $a
```

Kuva 7. Condition-osuus.

Sääntö päättyy aaltosulkeeseen. Sääntötiedostoon voidaan lisätä yhden tai useamman rivin kommentteja C-kielen tapaan (Kuva 8).

```
31  /*  Usean rivin kommentti
32      toinen rivi
33      kolmas rivi
34  */
35        condition: //yhden rivin kommentti
```

Kuva 8. Kommentointi.

4.4 YARA-työkalun suorittaminen

YARA-työkalua käytetään tässä opinnäytetyössä Windowsin komentoriviltä (cmd.exe). Työkalua käytetään syöttämällä YARA:n suoritettavan tietokoneohjelman nimi yara64.exe, jonka perään syötetään ohjelman suorituksen parametrit. Jos komentoriviin syöttää vain tietokoneohjelman nimen, YARA-työkalu tulostaa komentoriville virheilmoituksen puutteellisten parametrien vuoksi. Komentorivi 1 esittää yara64.exe-ohjelman tuottaman virheilmoituksen. Virheilmoituksen alle tulostuu opastus ohjelman käyttöön.

Komentorivi 1. Työkalun suorittaminen ilman parametreja.

```
D:\yara>yara64.exe
yara: wrong number of arguments
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Try `--help` for more options
```

Opastuksen mukaisesti YARA-ohjelman asetukset voidaan tulostaa komentoriville parametrilla `--help`. Komentorivi 2 esittää YARA-ohjelman tulosteen asetuskomennolla.

Komentorivi 2. YARA-ohjelman asetukset.

```
D:\yara>yara64.exe --help
YARA 3.11.0, the pattern matching swiss army knife.
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Mandatory arguments to long options are mandatory for short options too.

      --atom-quality-table=FILE        path to a file with the atom quality table
  -C, --compiled-rules                 load compiled rules
  -c, --count                          print only number of matches
  -d, --define=VAR=VALUE               define external variable
      --fail-on-warnings               fail on warnings
  -f, --fast-scan                      fast matching mode
  -h, --help                           show this help and exit
  -i, --identifier=IDENTIFIER          print only rules named IDENTIFIER
  -l, --max-rules=NUMBER               abort scanning after matching a NUMBER of rules
      --max-strings-per-rule=NUMBER    set maximum number of strings per rule (default=10000)
  -x, --module-data=MODULE=FILE        pass FILE's content as extra data to MODULE
  -n, --negate                         print only not satisfied rules (negate)
  -w, --no-warnings                    disable warnings
  -m, --print-meta                     print metadata
  -D, --print-module-data              print module data
  -e, --print-namespace                print rules' namespace
  -S, --print-stats                    print rules' statistics
  -s, --print-strings                  print matching strings
  -L, --print-string-length            print length of matched strings
  -g, --print-tags                     print tags
  -r, --recursive                      recursively search directories
  -k, --stack-size=SLOTS               set maximum stack size (default=16384)
  -t, --tag=TAG                        print only rules tagged as TAG
  -p, --threads=NUMBER                 use the specified NUMBER of threads to scan a directory
  -a, --timeout=SECONDS                abort scanning after the given number of SECONDS
  -v, --version                        show version information

Send bug reports and suggestions to: vmalvarez@virustotal.com.
```

YARA-ohjelman parametrit muokkaavat ohjelman suoritusta. Esimerkiksi m-parametrilla ohjelma tulostaa tulosteeseen onnistuneesti verratun säännön meta-osuuden. Ohjelma tulostaa osumien lukumäärän c-parametrilla.

Ohjelman onnistunutta suoritusta varten on komentoriville kirjoitettava ohjelman nimi ”yara64.exe”, sääntötiedoston polku sekä kohdetiedoston polku. Kohdetiedosto-osuus voi olla myös kansio tai prosessin id (pid)-arvo.

4.5 Tekstitiedoston luominen ja tarkistaminen

Tekstitiedoston luominen suoritetaan Windowsin komentoriviltä (cmd.exe) käyttäen echo-komentoa. Kirjoitetaan merkkijono ”this is a virus” tiedostoon testi.txt (Komentorivi 3).

Komentorivi 3. Tekstitiedoston luominen.

```
D:\yara>echo this is a virus > testi.txt
```

Tekstitiedoston tarkistamiseen suoritetaan YARA-työkalu (Komentorivi 4), syötetään sääntötiedosto ”rules.yar” ja juuri luotu kohde ”testi.txt”. Parametrilla m ohjelma tulostaa sääntöjen meta-osuuden. Parametri annetaan tulosteen selkeyttämiseksi.

Komentorivi 4. YARA-työkalun suorittaminen m-parametrilla.

```
D:\yara>yara64.exe rules.yar testi.txt -m
```

Annettujen parametrien ollessa ohjelman syntaksin mukaisesti oikein, ohjelma palauttaa komentoriville tulokset. Komentorivi 5 esittää ”testi.txt”-tiedoston tarkistamisen tulosteen. Tulosteen tarkistamisen jälkeen suoritetaan YARA-työkalu ilman m-parametria (Komentorivi 6).

Komentorivi 5. Tuloste testitiedoston tarkistamisesta.

```
testrule [author="Nicolas",description="This is a test rule"] testi.txt
```

Komentorivi 6. Testitiedoston tarkistus ilman parametreja.

```
D:\yara>yara64.exe rules.yar testi.txt
testrule testi.txt
```

4.6 Tulosten analysointi ja säännön optimointi

Vertaamalla tulosteita m-parametrilla ja ilman voidaan todeta, että m-parametrin määrittäminen ohjelman suoritukseen tuottaa käyttäjäystävällisemmän tulosteen.

Ilman parametreja ohjelma palauttaa ”osumat” muodossa:

```
{säännön nimi} {tiedostopolku}
```

m-parametrilla saatu tuloste:

```
{säännön nimi} [meta-osuus] {tiedostopolku}
```

Säännön tehokkuutta määrittäessä on hyvä tiedostaa, mitä merkkijonoja tarkistuksessa saattaa mennä ohi väärin luodun säännön seurauksena. Esimerkiksi luomalla kohdetiedostoon isoilla kirjaimilla kirjoitettu merkkijono voidaan testata säännön tehokkuutta tilanteessa, jossa kohdetiedosto sisältää saman merkkijonon eri kokoisena. Kirjoitetaan merkkijono ”This is a VIRUS” tiedostoon ”testi2.txt” (Komentorivi 7).

Komentorivi 7. Isoilla kirjaimilla kirjoitettu teksti.

```
D:\yara>echo This is a VIRUS > testi2.txt
```

Tarkistetaan juuri luotu tiedosto ”testi2.txt” (Komentorivi 8), joka sisältää isoilla kirjaimilla kirjoitetun merkkijonon ”VIRUS”. Tulosteen ollessa tyhjä voidaan todeta, että kohdetiedoston sisältö ei osunut aikaisemmin luotuun sääntöön.

Komentorivi 8. Muokatun tiedoston tarkistaminen.

```
D:\yara>yara64.exe rules.yar testi2.txt

D:\yara>
```

YARA-säännöissä tekstimerkkijonot ovat oletusarvoisesti merkkikokoriippuvaisia. Käyttämällä ”nocase”-määritettä merkkijonon jälkeen samalla rivillä tekstimerkkijonon vertaaminen ei ole merkkikokoriippuvainen. Kuvassa 9 monistetaan aikaisemmin luotu sääntö

”testrule” ja lisätään säännön strings-osuuden merkkijonoparametriin nocase-määrite halutun lopputuloksen saamiseksi. Monistettu sääntö tallennetaan nimellä ”testrule2”. Säännön meta-osuuteen päivitetään myös säännön funktiota tarkentava kuvaus. (Virus-Total 2019.)

```
rule testrule2
{
    meta:
        author = "Nicolas"
        description = "Case-insensitive string 'virus'"
    strings:
        $a = "virus" nocase
    condition:
        $a
}
```

Kuva 9. Muokattu sääntö.

Sääntötiedoston muokkauksen ja tallentamisen jälkeen testataan säännön toiminta. Suoritetaan YARA-työkalu päivitetyllä sääntötiedostolla ”rules.yar” ja kohdetiedostolla ”testi2.txt” (Komentorivi 9). Parametri m lisätään metatietojen tulostamiseksi.

Komentorivi 9. Tarkistaminen m-parametrilla.

```
D:\yara>yara64.exe rules.yar testi2.txt -m
testrule2 [author="Nicolas",description="Case-insensitive string \'virus\'"] testi2.txt
```

Saatu tuloste eroaa aiemmasta tulosteesta, sillä nocase-määritteellä vertaus onnistuu riippumatta merkkikoosta.

5 TOIMIALUEEN TIEDUSTELU

5.1 Sisäisen tiedustelun määritelmä

Active Directory -toimialueen tiedustelua Windowsin sisäänrakennetuilla komennoilla voidaan luonnehtia käytännön esimerkillä niin, että henkilö tarkastelee ja dokumentoi yrityksen toimistotiloissa näkyviä dokumentteja ja kokeilee yrityksen tietoteknisen osaston ovien lukkoja havainnoiden mahdollisten turvavarusteiden olemassaoloa. Tämänkaltainen toiminta on mahdollista kaikille yrityksen toimistotiloissa oleville henkilöille. Yrityksen tiloihin päässyt ulkopuolinen haitantekijä saattaisi toimia juuri näin tiedustellessaan tilojen fyysistä turvallisuutta.

Tässä konseptissa henkilöllä havainnollistetaan toimialueeseen kytkettynä olevaa laitetta, dokumenteilla toimialueen käyttäjäryhmiä ja ovien lukoilla mahdollisesti hyväksikäytettäviä konfiguraatiovirheitä.

Mikäli toimistotiloissa ei ole reaaliajassa valvontakameroita seuraavaa vartijaa tai muita henkilöitä näkemässä, tiedustelutoiminta saattaa jäädä huomaamatta ja todistusaineisto vähäiseksi.

Toimialueen tiedustelun mahdollistavia komentoja käyttävät yritysten IT-osastot toimialueen ylläpitoon. Esimerkiksi uuden ylläpitäjän aloittaessa IT-osastolla on Active Directory -toimialueeseen luotava uusi käyttäjä korkeat oikeudet omaavaan käyttäjäryhmään ja työntekijän työnkuvan vaihtuessa tulisi tehdä muokkauksia käyttäjäryhmiin.

Esimerkkinä ylläpitäjien käyttämästä toimialueen tiedustelun mahdollistavasta komennosta on net-komento, jolla voidaan hallita monia toimialueen konfiguraatioita. Net-komento sisältyy niin vanhoihin kuin uusimpiinkin Microsoft Windows -käyttöjärjestelmiin. Yksinkertaisimmillaan komennolla voidaan listata tiettyyn toimialueessa olevaan käyttäjäryhmään kuuluvat käyttäjätilit. (Fisher 2019.)

Toimialueen sisäisen tiedustelun havaitseminen ajoissa auttaa yrityksiä havaitsemaan mahdollisesti haitallisilla tarkoituksilla yrityksen toimialueessa olevia käyttäjiä ja saastuneita laitteita. Toimialueen tiedustelutoimien suorittaminen muodostaa yhden monista yrityksien tietoturvauhista.

5.2 Tiedustelukomentosarjan luominen ja sisältö

Tutkimusta varten luodaan .bat-päätteinen MS-DOS komentosarja. Komentosarjaan kirjoitetaan komentoja, joilla saadaan tiedusteltua Active Directory -toimialueen käyttäjätunnuksia ja korkeatasoisia käyttäjäryhmiä. Näitä komentoja käytetään aidoissa kyberhyökkäyksissä ja tietoturvatestauksissa. Komentosarja sisältää vain Windowsin sisäänrakennettuja komentoja. Sisäänrakennettujen komentojen käyttö vähentää toimialueeseen jo jalansijan saaneen hyökkääjän kiinnijäämisen riskiä haittaohjelmien siirtämiseen verrattuna. Komentosarjan suorituksesta pyritään luomaan mahdollisimman vähän henkilöstölle näkyvää materiaalia ja tulokset pyritään ohjaamaan sellaiseen kansioon, joka ei herätä käyttäjän epäilyksiä. Kuvassa 10 on tiedustelukomentosarjan sisältö avattu Atom-tekstinkäsittelyohjelmaan.

```
1  @echo off
2  mkdir C:\Microsoft\Windows\
3  2>NUL whoami /all > "C:\Microsoft\Windows\license.txt"
4  2>NUL net group "Domain Admins" /domain >> "C:\Microsoft\Windows\license.txt"
5  2>NUL net group "Enterprise Admins" /domain >> "C:\Microsoft\Windows\license.txt"
6  2>NUL net localgroup "Administrators" >> "C:\Microsoft\Windows\license.txt"
```

Kuva 10. Tiedustelukomentosarjan sisältö.

Tiedustelukomentosarjan ensimmäisen rivin komento ”@echo off” määrittää, ettei rivin jälkeisiä komentoja tulosteta komentorivin tulosteeseen (Laaksonen 2002).

Toisella rivillä oleva mkdir-komento luo kansion ”Windows” komennolle annettuun polkuun ”C:\Microsoft\”. Kansio tulee sisältämään komentosarjan tulosteet tekstitiedostona.

Komentosarjan rivien 3 - 6 alkuun määritetty ”2>NUL” estää komentoriviä tulostamasta mahdollisia virheilmoituksia komentokehotteeseen.

Komentojen tulosteiden ohjaus tiedostoon onnistuu >- ja >>- merkinnöillä. Rivillä 3 komennon jälkeen annettu merkki > korvaa mahdollisesti olemassa olevan tiedoston annetusta polusta. >>-merkintä kirjoittaa tulosteen tiedoston sisällön perään. Tulosteet ohjataan juuri luotuun kansioon tiedostoon ”license.txt”.

Rivin 3 komentoa whoami voidaan käyttää kirjautuneena olevan käyttäjän tietojen tulostamiseen komentoriville. Ilman parametreja komento tulostaa käyttäjän tiedot muodossa ”TOIMIALUE\käyttäjä”. Komentosarjassa annettu parametri /all tulostaa

suorituksenhetkisen käyttäjänimen, Security Identifier (SID) -arvon, käyttöoikeudet ja ryhmät, joihin kirjautuneena oleva käyttäjä kuuluu. (Microsoft 2017.)

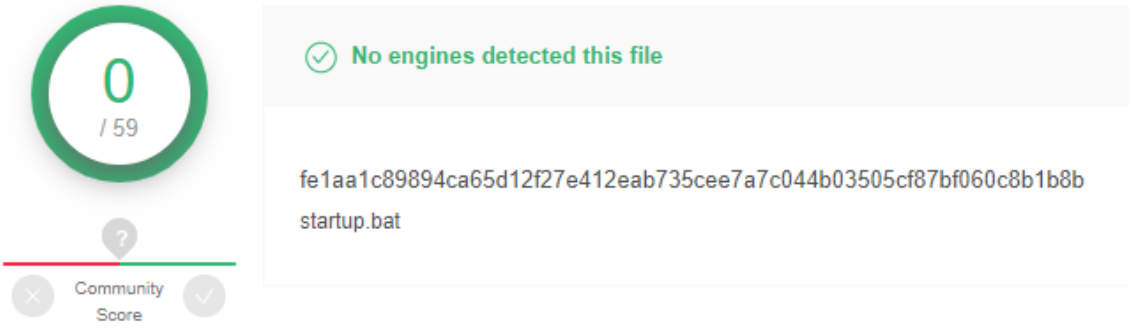
Rivien 4 ja 5 komennot tulostavat toimialueen korkeimmat oikeudet omaavien ryhmien Domain Admins ja Enterprise Admins sisältämät käyttäjätunnukset. Komentosarjan viimeinen komento tulostaa laitteen paikalliseen ylläpitoryhmään kuuluvat tunnukset.

Tiedustelukomentosarjan löytyessä muun kuin valtuutetun käyttäjän laitteelta, on syytä epäillä laitteen käyttäjän haitallisia aikomuksia tai laitteen tietoturvan vaarantuneen. Koska tiedustelukomentosarja kohdistuu yrityksen tietoturvaan (tässä tapauksessa korkean luokan ryhmien tiedusteluun Active Directory -toimialueessa) luokitellaan se tietoturvauhaksi.

5.3 Komentosarjan analysointi

Komentosarjan komentoja sisältyy monien haittaohjelmien suoritusketjuihin. Hyökkääjän saadessa jalansijan organisaatioon yleisesti ensimmäisiä toimia on sisäinen tiedustelu (Bey 2017). Tiedustelutoimiin voi myös liittyä esimerkiksi samassa verkossa olevien laitteiden haavoittuvuus- ja porttiskannausta. Ennen tiedustelukomentosarjan tarkempaa analyysiä tarkastellaan komentosarjan proaktiivista havaitsemista ajamalla virustarkistus tiedostoa kohden.

Komentosarja voidaan ladata ilmaiseen VirusTotal-palveluun, jossa tiedosto analysoidaan kymmenillä eri virustorjuntaohjelmistoilla. Kuvassa 11 tiedosto on ladattu VirusTotal-palveluun ja analysoitu 59 eri virustorjuntaohjelmistolla. Kuvassa näkyvä arvo 0/59 ja otsikko ”No engines detected this file” kertovat, että yksikään virustorjuntaohjelmista ei havainnut tiedostoa haitalliseksi. Otsikon alla näkyy tiedoston SHA-256-tiiviste ja tiivisteen alla tiedoston nimi.



Kuva 11. VirusTotal-palvelun tulokset.

Komentosarjan tiiviste ei siis vastaa tarkastuksen ajankohtana tunnettujen haittaohjelmien tiivisteitä. Toimialueessa ajettavien ajastettujen virustarkistusten myötä tiedostoa ei välttämättä havaittaisi.

5.4 YARA-sääntöjen luominen

Tiedustelukomentosarjan sisältämiä komentoja analysoimalla voidaan luoda komentosarjan tunnistava YARA-sääntö. Luodun säännön tehokkuutta voidaan arvioida analysoimalla YARA-työkalun tarkistuksen tuloksista saatavia mahdollisesti virheellisiä tuloksia. Määrittämällä säännölle vain yksi merkkijono, esimerkiksi ”whoami”, virheellisiä tuloksia voi syntyä monesta eri syystä.

Luodaan yksinkertainen sääntö ”whoami” -merkkijonon löytämiseen ja tarkastetaan hakemisto C:\Program Files\. Kuva 12 esittää juuri luodun säännön whoami_string, joka etsii merkkijonon ”whoami” täysipituisena. Suoritetaan YARA-työkalu hakemistoon C:\Program Files\ sääntötiedostolla ”reconrules.yar” ja ohjataan tulokset tekstitiedostoon ”whoamiresults.txt” (Komentorivi 10).

```
rule whoami_string
{
    meta:
        author = "Nicolas"
        description = "Matched whoami-string"

    strings:
        $a = "whoami" fullword

    condition:
        $a
}
```

Kuva 12. Whoami-merkkijonon sääntö.

Komentorivi 10. Hakemiston C:\Program Files\ tarkastaminen.

```
D:\yara>yara64.exe reconrules.yar -w -m -r "C:\Program Files\" 2>&1 | findstr /v /b "error"
 >> whoamiresults.txt
```

Suoritusparametrien jälkeinen findstr on Windowsin komentorivin komento, joka parametreilla /v ja /b poistaa tulosteesta ne rivit, jotka alkavat sanalla ”error”.

Haetaan tulokset komentorivin komennolla type, joka tulostaa tiedoston sisällön näytölle. Tulostetta käsitellään find-komennolla parametreilla /c ja /v, jotta komentorivi tulostaa vain tiedoston sisältämien rivien määrän (Komentorivi 11).

Komentorivi 11. YARA-työkalun tulosteen rivimäärä.

```
D:\yara>type whoamiresults.txt | find /c /v ""
11
```

Tiedoston rivimäärä 11 kertoo, että tarkistus tuotti paljon tuloksia vain C:\Program Files\-hakemiston tarkistamisesta. Tarkistuksen tulos voidaan analysoida tuottavan liikaa virheellisiä havaintoja.

Tiedustelukomentosarjan sisältäessä neljä eri komentoa, jotka tulostavat samaan tiedostoon, voidaan luoda oma YARA-sääntö jokaisen eri komennon tunnistamiseen. Kuvat 13, 14 ja 15 esittävät tiedustelukomentosarjan eri komentojen tunnistamista varten luodut säännöt.

```
rule da_recon
{

    meta:
        author = "Nicolas"
        description = "Found Domain Admin recon"

    strings:
        $a = "net group" fullword nocase
        $b = "Domain Admins" nocase
        $c = "Domain Administrators" nocase

    condition:
        $a and ($b or $c)
}
```

Kuva 13. Domain Admins tiedustelun sääntö.

```
rule ea_recon
{
    meta:
        author = "Nicolas"
        description = "Found Enterprise Admins recon"

    strings:
        $a = "net group" fullword nocase
        $b = "Enterprise Admins" nocase
        $c = "Enterprise Administrators" nocase

    condition:
        $a and ($b or $c)
}
```

Kuva 14. Enterprise Admins tiedustelun sääntö.

```
rule localadmin_recon
{

    meta:
        author = "Nicolas"
        description = "Found local admin recon"

    strings:
        $a = "net localgroup" fullword nocase
        $b = "Administrators" fullword nocase

    condition:
        $a and $b
}
```

Kuva 15. Paikallisen Administrators-ryhmän tiedustelun sääntö.

5.5 Sääntöjen testaaminen

Tiedustelukomentosarja kopioidaan käyttäjän Startup-kansioon Windowsin copy-komennolla (Komentorivi 12). Startup-kansiossa sijaitsevat tiedostot suoritetaan tietokoneen käynnistyessä.

Komentorivi 12. Komentosarjan kopioiminen.

```
D:\yara>copy startup.bat "C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\"
        1 file(s) copied.
```

Suoritetaan YARA-työkalu Startup-kansioon komentosarjan havaitsemiseksi (Komentorivi 13). Tulosteesta voidaan huomata jokaisen säännön toimivan tarkoituksenmukaisesti.

Komentorivi 13. YARA-työkalun suorittaminen Startup-hakemistoon.

```
D:\yara>yara64.exe reconrules.yar -w -r "C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
\Startup"
whoami_string C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\startup.bat
da_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\startup.bat
ea_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\startup.bat
localadmin_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\startup.bat
```

Tiedustelukomentosarja on onnistuneesti analysoitu ja analyysin pohjalta luotu YARA-sääntöjä sen eri osien tunnistamiseen. Koska komentosarja sisältää Windowsin omia komentoja, voi haitantekijä olla syöttänyt komennot erikseen komentokehotteeseen. Tässä tapauksessa voidaan tarkistaa esimerkiksi käyttäjän PowerShell-työkalun komentohistoria epäilyttävien komentojen varalta.

Syötetään PowerShell-työkalun komentoriville tiedustelukomentosarjan komennot yksi kerrallaan (Komentorivi 14). Komennot jäävät natiiviin tekstitiedostoon ”ConsoleHost_history.txt”.

Komentorivi 14. Tiedustelukomennot PowerShell-komentorivillä.

```
PS D:\yara\results> whoami /all > output.txt; net group "Domain Admins" /domain >> output.txt; net
 group "Enterprise Admins" /domain >> output.txt; net localgroup "Administrators">> output.txt
```

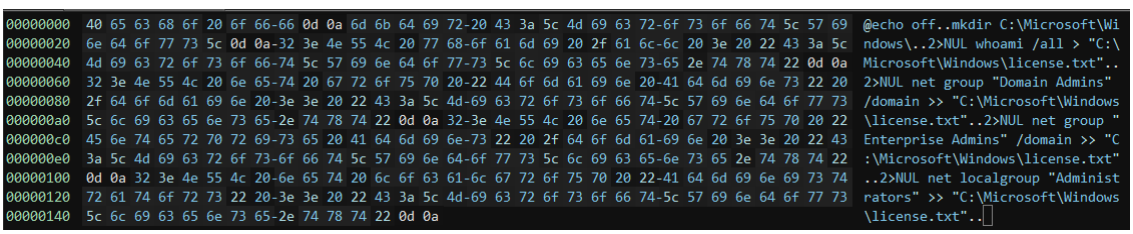
Tarkastetaan komentohistoria YARA-työkalulla (Komentorivi 15). Tulosteesta voidaan päätellä sääntöjen toimivan myös tilanteessa, jossa tiedustelukomennot on suoritettu PowerShell-komentorivillä erillisinä komentoina.

Komentorivi 15. PowerShell-komentohistorian tarkistaminen.

```
D:\yara>yara64.exe reconrules.yar -w -r "C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
ConsoleHost_history.txt"
whoami_string C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
da_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
ea_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
localadmin_recon C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
```

5.6 Heksapohjaisten sääntöjen luominen

Tiedustelukomentosarja voidaan avata Binary Ninja-ohjelmaan tiedoston heksadesimaaliarvojen tarkastelemiseksi (Kuva 16). Arvojen perusteella voidaan luoda heksapohjaiset YARA-säännöt jokaisesta komentosarjan eri komennosta.



Kuva 16. Binary Ninja-ohjelman näkymä.

YARA-sääntöjen muokkaaminen heksadesimaaliarvopohjaiseksi suoritetaan korvaamalla strings-osuuden merkkijonot eri komentojen heksadesimaaliarvoilla. Kuva 17 esittää whoami-komentoa vastaavaa sääntöä, joka on toteutettu käyttäen heksadesimaaliarvoja.

```
rule whoami_hex {

    meta:
      author = "Nicolas"
      description = "Matched whoami hex-string"

    strings:
      $a = {77 68 6f 61 6d 69}

    condition:
      $a
}
```

Kuva 17. Whoami-komennon heksasääntö.

Kuvat 18, 19 ja 20 esittävät tiedustelukomentosarjan muiden komentojen pohjalta tehtyjä YARA-sääntöjä. Sääntöjen strings-osuudet on toteutettu käyttäen heksadesimaaliarvoja.

```
rule da_recon_hex {

    meta:
      author = "Nicolas"
      description = "Matched net group Domain Admins hex-string"

    strings:
      $a = {6e 65 74 20 67 72 6f 75 70 20 22 44 6f 6d 61 69 6e 20 41 64 6d 69 6e 73 22}

    condition:
      $a
}
```

Kuva 18. Domain Admins-tiedustelun heksasääntö.

```
rule ea_recon_hex {

    meta:
      author = "Nicolas"
      description = "Matched net group Enterprise Admins hex-string"

    strings:
      $a = {6e 65 74 20 67 72 6f 75 70 20 22 45 6e 74 65 72 70 72 69 73 65 20 41 64 6d 69 6e 73 22}

    condition:
      $a
}
```

Kuva 19. Enterprise Admins-tiedustelun heksasääntö.

```
rule localadmin_recon_hex {

    meta:
      author = "Nicolas"
      description = "Matched net localgroup Administrators hex-string"

    strings:
      $a = {6e 65 74 20 6c 6f 63 61 6c 67 72 6f 75 70 20 22 41 64 6d 69 6e 69 73 74 72 61 74 6f 72 73 22}

    condition:
      $a
}
```

Kuva 20. Paikallisen Administrators-ryhmän tiedustelun heksasääntö.

Testataan luotuja sääntöjä suorittamalla YARA-työkalu startup.bat-tiedostoa kohden (Komentorivi 16). Työkalun suoritukseen asetetaan kohdetiedosto ja heksadesimaaliarvopohjaiset säännöt sisältävä sääntötiedosto. Tulosteesta voidaan päätellä sääntöjen toimivan halutulla tavalla.

Komentorivi 16. YARA-työkalun suorittaminen heksasäännöillä.

```
D:\yara>yara64.exe hexrules.yar startup.bat
whoami_hex startup.bat
da_recon_hex startup.bat
ea_recon_hex startup.bat
localadmin_recon_hex startup.bat
```

6 AJASTETTU YARA-SKANNAUS

Opinnäytetyön luvussa 5 olevan analyysin ja luotujen sääntöjen perusteella voidaan Windows-laitteelle luoda ajastettu tehtävä tiettyjen kohteiden tarkistamiseen YARA-työkalulla. Tarkistettaviksi kohteiksi määritetään laitteen PowerShell-työkalun komentohistoria sekä käynnistyskansio.

Ajastettua YARA-skannausta varten luodaan .bat-päätteinen komentojono (Kuva 21), joka määritetään Windowsin ajoitettuihin tehtäviin. Komentojono tulostaa kansioon D:\scanresults\ skannauksen tulokset .txt-tekstitiedostoon, joka nimetään päivämäärän mukaan. Komentojono tulostaa skannaustuloksiin päivämäärän, kellonajan sekä PowerShell-työkalun komentohistorian ja käynnistyskansion YARA-skannaustulokset. Komentojono tallennetaan nimellä ”yarascan.bat”.

```
@echo off
date /T > D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
time /T >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo. >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo Powershell history results >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo. >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
D:\yara\yara64.exe D:\yara\reconrules.yar -w -m -r "C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\PowerShell ^
\PSReadLine\ConsoleHost_history.txt" >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo. >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo Startup-folder results >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
echo. >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
D:\yara\yara64.exe D:\yara\reconrules.yar -w -m -r "C:\Users\Nicolas\AppData\Roaming\Microsoft\Windows\Start Menu ^
\Programs\Startup" >> D:\scanresults\scan_%date:~-4,4%%date:~-7,2%%date:~-10,2%.txt
```

Kuva 21. Komentojonon sisältö.

Suoritetaan yarascan.bat-komentojono komentoriviltä ja tarkistetaan, että tulokset sisältävä tekstitiedosto on määritetty oikein (Komentorivi 17). Komentoriviin syötetään ajettava tiedosto (yarascan.bat) sekä dir-komento kansion D:\scanresults\ sisällön tulostamiseksi. Tulosteesta voidaan päätellä komentojonon luoman tekstitiedoston nimen olevan haluttua muotoa.

Komentorivi 17. Komentojonon suoritus ja kansion sisältö.

```
D:\scanresults>D:\yara\yarascan\yarascan.bat & dir /B D:\scanresults\
scan_20200404.txt
```

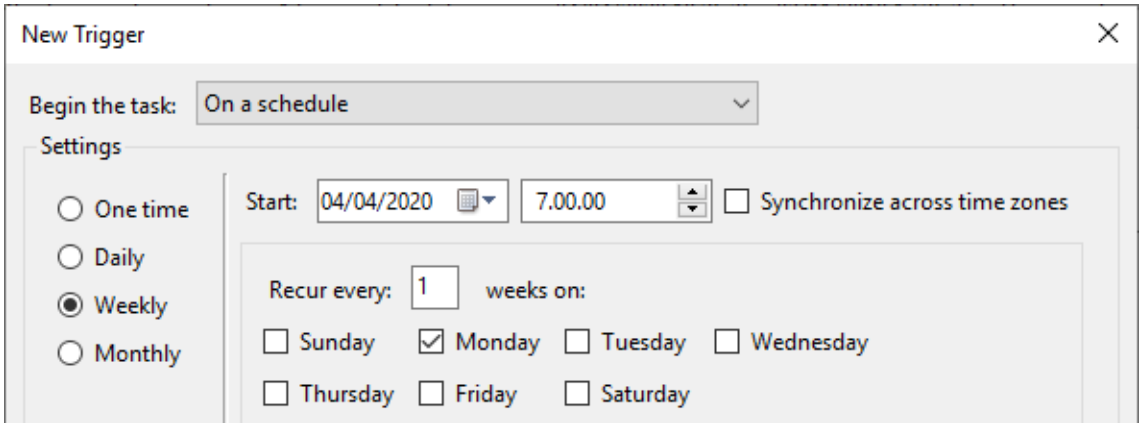
Skannauksen suorittamisen jälkeen avataan skannaustulokset sisältävä tekstitiedosto komentorivin komennolla type. Tämä komento tulostaa tiedoston sisällön komentorivin ikkunaan (Komentorivi 18).

Komentorivi 18. Skannaustulokset.

```
D:\scanresults>type scan_20200404.txt
04/04/2020
19.02

Powershell history results

whoami_string [author="Nicolas",description="Matched whoami-string"] C:\Users\Nicolas\AppData\Roaming\Microsoft\Wi
ndows\PowerShell\PSReadLine\ConsoleHost_history.txt
da_recon [author="Nicolas",description="Found Domain Admin recon"] C:\Users\Nicolas\AppData\Roaming\Microsoft\Wind
ows\PowerShell\PSReadLine\ConsoleHost_history.txt
ea_recon [author="Nicolas",description="Found Enterprise Admins recon"] C:\Users\Nicolas\AppData\Roaming\Microsoft
\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
localadmin_recon [author="Nicolas",description="Found local admin recon"] C:\Users\Nicolas\AppData\Roaming\Microso
ft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

Startup-folder results

whoami_string [author="Nicolas",description="Matched whoami-string"] C:\Users\Nicolas\AppData\Roaming\Microsoft\Wi
ndows\Start Menu\Programs\Startup\startup.bat
da_recon [author="Nicolas",description="Found Domain Admin recon"] C:\Users\Nicolas\AppData\Roaming\Microsoft\Wind
ows\Start Menu\Programs\Startup\startup.bat
ea_recon [author="Nicolas",description="Found Enterprise Admins recon"] C:\Users\Nicolas\AppData\Roaming\Microsoft
\Windows\Start Menu\Programs\Startup\startup.bat
localadmin_recon [author="Nicolas",description="Found local admin recon"] C:\Users\Nicolas\AppData\Roaming\Microso
ft\Windows\Start Menu\Programs\Startup\startup.bat
```

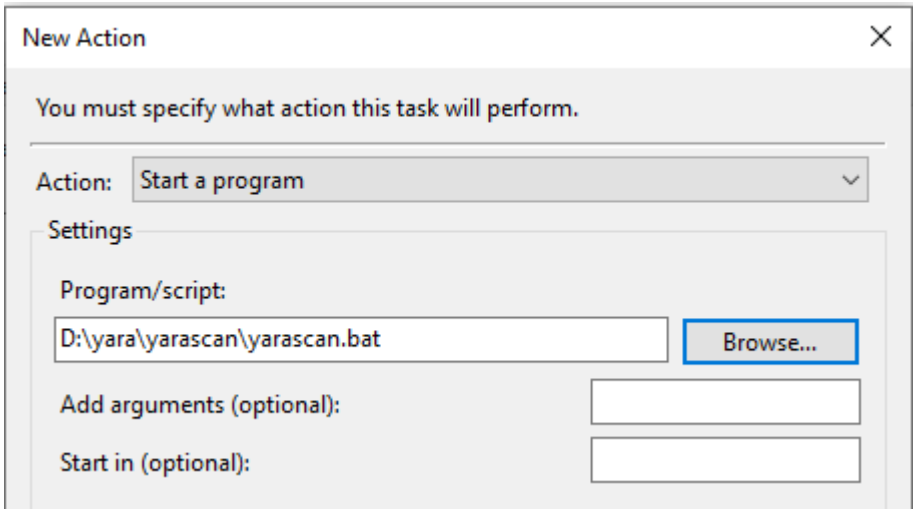
Tiedoston sisällön voidaan todeta olevan helppolukuinen, sillä tiedosto sisältää skannauksen ajankohdan ja skannaustulokset on otsikoitu.

Ajastetun tarkistuksen määritys tehdään Windowsin Tehtävien Ajastus -ohjelmassa. Ajastettu tehtävä määritetään suoriutumaan viikoittain, maanantaisin kello 07:00 (Kuva 22).



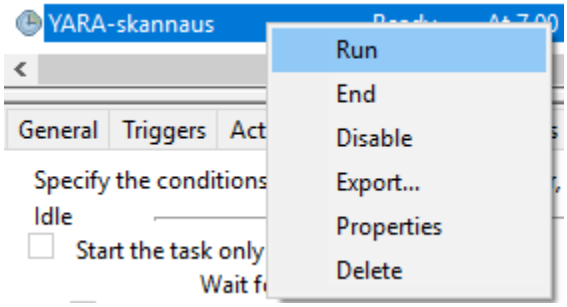
Kuva 22. Tehtävän suoritusajankohta.

Ajastettu tehtävä määritetään suorittamaan komentojono ”yarascan.bat” (Kuva 23).



Kuva 23. Komentojonon asettaminen.

Ajastettua YARA-skannausta voidaan testata Tehtävien Ajoitus-ohjelmassa myös manuaalisesti (Kuva 24).



Kuva 24. Tehtävän suorittaminen manuaalisesti.

7 YARA-SÄÄNNÖT TIETOTURVARATKAISUISSA

Tehokkaimpina YARA-sääntöjä hyödyntävistä tietoturvaratkaisuista on usein luonnehdittu saksalaisen tietoturvayrityksen Nextron Systemsin tietoturvatuotteet. Yrityksen vaikuttavin tietoturvatuote on nimeltään THOR. Tämä yritystasoinen tuote sisältää noin 11 000 YARA-sääntöä, IR-työkaluja sekä mahdollistaa tuhansien kohdelaitteiden tarkistuksen. Yrityksen ilmainen versio THOR-työkalusta on THOR Lite, joka sisältää kolme tuhatta sääntöä. Yritys tarjoaa myös avoimeen lähdekoodiin perustuvan, Python-ohjelmointikielellä kirjoitetun Loki-työkalun, joka sisältää samat kolme tuhatta sääntöä kuin THOR Lite. Loki-työkalu on kehitetty nopeaan IoC-indikaattorien havaitsemiseen.

Monet GitHub-versionhallintasivustosta saatavat avoimeen lähdekoodiin perustuvat tietokoneohjelmat hyödyntävät YARA-sääntöjä monin eri tavoin ja tietoturvatutkijat voivat ladata ilmaiseksi eri verkkosivustoilta sääntötiedostoja uusimpien haittaohjelmien tunnistamiseen.

YaraRules Project on verkkosivusto, joka on perustettu siksi, että tietoturvatutkijoille on yksi tietty säilö YARA-sääntöjen jakamiseen. Verkkosivuston jakamat säännöt ovat GNU-GPLv2-lisenssillä avoimia kaikille käyttäjille ja organisaatioille. Tämä talletuspaikka sisältää uusimpia ja hiotuimpia sääntöjä esimerkiksi viimeisimpien tietojenkalasteluviestien ja haitallisten liitetiedostojen havaitsemiseen. (YaraRules Project 2017.)

yarGen on tietoturva-asiantuntija Florian Rothin kehittämä automaattinen YARA-sääntögeneraattori. Tämän työkalun tarkoituksena on luoda kohdetiedostosta automaattisesti sen tunnistavia YARA-sääntöjä. Työkalun luomat säännöt vaativat kuitenkin muokkaamista, jotta lopputuloksena saadaan tehokas sääntö. Lopullinen toimiva sääntö saadaan luotua poistamalla epäoleelliset tai oletetusti liikaa virheellisiä havaintoja aiheuttavat merkkijonot säännöstä. (Roth 2018.)

yextend-ohjelma on kehitetty arkistoitujen kohteiden YARA-skannausta varten. Ohjelman on jakanut GitHub-verkkosivustolle tietoturvayritys Bayshore Networks. Ohjelma on kehitetty siksi, että haittaohjelmien havaitseminen monen tason kompressoidusta arkistosta on natiiville YARA-työkalulle hankalaa. (Bayshore Networks 2018.)

Muita YARA-sääntöjä hyödyntäviä työkaluja ovat esimerkiksi YARA-työkalun etäsuorittamisen PowerShell-komentoriviltä mahdollistava ”Invoke-Yara” sekä sähköpostivirtaa skannaava ”yaraMail”. (InQuest 2020.)

8 POHDINTA JA TULOKSET

2000-luvun alussa tietoturvauhkien havaitsemiseen koettiin riittävän ajan tasalla oleva virustorjuntaohjelma. Yritysten toiminnan siirtyessä verkkoon avautuu uudenlainen hyökkäysvektori ja huoli ulkopuolisen tahon pääsemisestä arkaluonteiseen materiaaliin kasvaa.

Viimeaikaisissa kyberhyökkäyksissä käytetyt haittaohjelmat on tehty tarkasti välttämään havaituksi tulemista. Virustorjuntaohjelmien valmistajille asettuu jatkuvasti paineita kovan kilpailun ja uhkien monipuolistumisen myötä. Tästä syystä harvat virustorjuntaohjelmia tuottavat yritykset avaavat julkisesti tuotteidensa konkreettisia toimintamalleja. YARA-sääntöjen ja sääntöyhteisön avoimuus ovat tuoneet tietoturvatutkijoille uusia tapoja kehittää puolustuskeinoja uusimpienkin uhkien tunnistamiseen.

Opinnäytetyöni aiheen valitsin sillä perusteella, että suomeksi julkaistuja tutkimustöitä ei YARA-säännöistä löytynyt. Työn lähtökohtana toimi oma kiinnostukseni YARA-sääntöjen käyttöön haittaohjelmien havaitsemiseen, uusien sääntöjen luomiseen sekä jatkokehitykseen. Koen YARA-sääntöjen monipuolisuuden ja internetistä löytyvien sääntöjä hyödyntävien työkalujen auttavan niin tietoturva-alan ammattilaisia kuin yritysten tietoturvavastaavia implementoimaan yhä tietoturvallisempia ratkaisuja IT-ympäristöihin.

Opinnäytetyön tutkimusosuuteen kuviteltiin tilanne, jossa yrityksen laitteelle on saatu ladattua tiedustelukomentosarja ja hyökkäystoimet jatkuvat. Tutkimustyö aloitettiin opettelemalla YARA-sääntöjen syntaksi ja YARA-työkalun käyttö. Tutkimustyö tehtiin luomalla tiedostoja, joita analysoimalla generoitiin YARA-sääntöjä niiden tunnistamiseen laitteesta.

YARA-sääntöjen syntaksi esiteltiin jakamalla esimerkkisääntö osiin, joita käsiteltiin yksi kerrallaan. Sääntöjen eri osien funktioiden ymmärtäminen loi pohjan YARA-työkalun käyttöön. Työkalujen tulosteiden analysoinnin tulokset pakottivat jatkokehittämään ja optimoimaan luotuja sääntöjä. Tuloksena saatiin kehitettyä entistä tarkempia sääntöjä.

Tutkimustyössä kehitettiin ajastettu YARA-skannaus laittelle. Ajastettu skannaus konfiguroitiin tarkastamaan PowerShell-työkalun komentohistoria ja käynnistyskansio tutkimuksessa analysoidun tiedustelukomentosarjan eri komentojen varalta. Tiedustelukomentosarja luotiin ennalta opittujen Active Directory -toimialueen tiedustelukomentojen pohjalta.

Työssä esiteltiin muutama YARA-sääntöjä hyödyntävä tietoturvaratkaisu, jotka osoittavat sääntöjen todellisen tehokkuuden ja monikäyttöisyyden. Sääntöjä hyödyntävät monet internetissä vapaasti jaettavat työkalut. Tutkimustyön aikana löydettiin uusia YARA-sääntökirjastoja.

YARA-sääntöjen monipuolisuus tarjoaa uusia mahdollisuuksia tietoturvauhkien havaitsemiseen. Esimerkiksi yhtä haitalliseksi todettua tiedostoa analysoimalla voidaan luoda tunnusmerkkejä sen havaitsemiseksi muualla IT-järjestelmissä. Sääntöjen helppo syntaksi ja niiden nopea testaaminen kasvattavat sääntöjen tehokkuutta huomattavasti. YARA:n internetistä löytyvä dokumentaatio auttaa ymmärtämään sääntöjen toimintaa ja monipuolisuutta.

Jatkokehityksenä opinnäytetyöni ulkopuolella aion kehittää työssä luotua ajastettua YARA-skannausta useiden muiden tietoturvauhkien havaitsemiseen. Tavoitteenani on monipuolistaa ajastettua YARA-skannausta esimerkiksi tunnistamaan laitteessa olevien ohjelmistojen haavoittuvat ohjelmistoversiot. Opinnäytetyön kirjottaminen oli mielenkiintoinen ja tietoturva-ammattitaitoani kehittävä prosessi.

LÄHTEET

Bey, T. 2017. Disrupting the cyber kill chain with Microsoft solutions. Viitattu 15.3.2020. https://www.bdo.com/digital/insights/cybersecurity/disrupting-cyber-kill-chain-microsoft-solutions

Bayshore Networks 2018. yextend. Viitattu 13.4.2020. https://github.com/BayshoreNetworks/yextend

Constantin, L. 2019. What is adware? How it works and how to protect against it. Viitattu 28.3.2020. https://www.csoonline.com/article/3406422/what-is-adware-how-it-works-and-how-to-protect-against-it.html

Davis, J. 2019. Ransomware Costs on the Rise, Causes Nearly 10 Days of Downtime. Viitattu 1.3.2020. https://healthitsecurity.com/news/ransomware-costs-on-the-rise-causes-nearly-10-days-of-downtime

Fisher, T. 2019. Net Command. Viitattu 14.3.2020. https://www.lifewire.com/net-command-2618094

Fruhlinger, J. 2019. What is a computer virus? How they spread and 5 signs you've been infected. Viitattu 27.3.2020. https://www.csoonline.com/article/3406446/what-is-a-computer-virus-how-they-spread-and-5-signs-youve-been-infected.html

Fruhlinger, J. 2019. What is a computer worm? How this self-spreading malware wreaks havoc. Viitattu 27.3.2020. https://www.csoonline.com/article/3429569/what-is-a-computer-worm-how-this-self-spreading-malware-wreaks-havoc.html

Greenberg, A. 2018. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Viitattu 13.3.2020. https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

Heinzman, A. 2019. What is RAT Malware, and Why Is It So Dangerous?. Viitattu 1.3.2020. https://www.howtogeek.com/410634/what-is-rat-malware-and-why-is-it-so-dangerous/

InQuest 2020. awesome-yara. Viitattu 13.4.2020. https://github.com/InQuest/awesome-yara#tools

KnowBe4 2020. What is Ransomware?. Viitattu 1.3.2020. https://www.knowbe4.com/ransomware

Laaksonen, A. 2002. Opasarkisto: MS-DOSin komentojonot: Komentojonot. Viitattu 14.3.2020. https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=msdkj

Love, J. 2018. Malware Types and Classifications. Viitattu 28.4.2020. https://www.lastline.com/blog/malware-types-and-classifications/

Mathews, L. 2020. Average Cost To Recover From Ransomware Skyrockets To Over $84,000 Viitattu 13.3.2020. https://www.forbes.com/sites/leemathews/2020/01/26/average-cost-to-recover-from-ransomware-skyrockets-to-over-84000/

Matthews, T. 2019. Creeper: The World's First Computer Virus. Viitattu 1.3.2020. https://www.exabeam.com/information-security/creeper-computer-virus/

Microsoft 2017. whoami. Viitattu 14.3.2020. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/whoami

National Institute of Standards and Technology 2012. Computer Security Incident Handling Guide. Viitattu 8.4.2020. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

Rikoslaki. 38. luku. 21.4.2015/578. Annettu Helsingissä 1.9.1995. Saatavilla sähköisesti: https://www.finlex.fi/fi/laki/ajantasa/1889/18890039001

Roth, F. 2018. yarGen. Viitattu 11.4.2020. https://github.com/Neo23x0/yarGen/blob/master/README.md

Rouse, M. 2019. Trojan horse (computing). Viitattu 1.3.2020. https://searchsecurity.techtarget.com/definition/Trojan-horse

Rouse, M. 2019. Spyware. Viitattu 3.5.2020. https://searchsecurity.techtarget.com/definition/spyware

Tieteen termipankki 2020. Viitattu 14.3.2020. http://tieteentermipankki.fi/wiki/Language_Technology:regular-expression

TSK 52. Sanastokeskus TSK. Viitattu 5.4.2020. http://www.tsk.fi/tiedostot/pdf/Kyberturvallisuuden_sanasto.pdf

Viestintävirasto 2015. Viitattu 29.02.2020. https://legacy.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2015/02/ttn201502101246.html

VirusTotal 2016. YARA in a nutshell. Viitattu 26.2.2020. https://virustotal.github.io/yara/

VirusTotal 2019. YARA Documentation. Viitattu 29.2.2020. https://yara.readthedocs.io/

Voigt, L. 2018. Incident Response Steps: 6 Tips for Responding to Security Incidents. Viitattu 5.4.2020. https://www.exabeam.com/incident-response/steps/

YaraRules Project 2017. YaraRules Project. Viitattu 11.4.2020. https://yararules.com/project/

Zetter, K. 2014. Hacker Lexicon: What is a Backdoor?. Viitattu 3.5.2020. https://www.wired.com/2014/12/hacker-lexicon-backdoor/