jamk.fi

Tietoturvauhkien virtuaalisen testaus-ympäristön automatisointi

Simo Tarkiainen

Opinnäytetyö
Huhtikuu 2021
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), Tieto- ja viestintätekniikka

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Tarkiainen, Simo	Opinnäytetyö, AMK	Huhtikuu 2021
	Sivumäärä 50	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: Kyllä

Työn nimi
Tietoturvauhkien virtuaalisen testausympäristön automatisointi

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Tero Kokkonen, Jarmo Nevala

Toimeksiantaja(t)
CYBERDI

Tiivistelmä

Tutkimuksen tarkoituksena oli automatisoida virtuaalisen ympäristön rakennus. Virtuaaliympäristön tarkoitus oli mahdollistaa haittaohjelmien tutkiminen, jota voidaan hyödyntää tietoturvakoulutuksessa ja muussa vastaavassa testaamisessa. Luotava virtuaaliympäristö oli tarkoitettu käytettäväksi koulutuksen lisäksi myös kotioloissa, minkä vuoksi sen asettamat vaatimukset isäntäkoneelle eivät saaneet olla liian suuret.

Tutkimuksen alussa tutkittiin erilaisia teknologioita ja palveluita, joiden avulla saataisiin automaattisesti luotua ympäristö ja sille tarvittavat ominaisuudet. Kun löydettiin tyydyttävä kokoonpano, keskityttiin ensimmäiseksi virtuaalikoneiden rakennuksen automatisointiin. Kun virtuaalikoneet saatiin automaattisesti rakennettua, lisättiin konfiguraationhallintatyökaluilla virtuaalikoneille ohjelmistoja, joilla mahdollistettiin tiedon kerääminen ja käsittely. Viimeiseksi suunniteltiin ympäristön eristys.

Tutkimuksen lopputulokseksi saatiin luotua toimintamalli, joka sisältää ohjeet eri palveluille virtuaaliympäristön luomista varten. Toimintamallissa virtuaaliympäristön rakennus on automatisoitu alusta alkaen, mikä tekee sen hyödyntämisestä tehokasta ja käyttäjäystävällistä. Virtuaaliympäristö koostui virtuaalikoneista, joilla oli kullakin omat roolinsa. Näiden roolien perusteella virtuaalikoneille asennettiin eri palvelut. Palvelut mahdollistivat lokien keräämisen, tallentamisen ja tutkimisen. Lokeista pystyttiin analysoimaan virtuaaliympäristössä tapahtuneita haittaohjelmien aiheuttamia muutoksia.

Tutkimuksessa kehitetty toimintamalli vastasi toimeksiantajan asettamiin vaatimuksiin ja tavoitteisiin. Tutkimuksessa luodulla toimintamallilla pystyttiin automaattisesti rakentamaan toimiva virtuaaliympäristö, joka tarjoaa työkalut tietoturvatestauksen aloittamista varten.

Avainsanat (asiasanat)

Tietoturva, automatisointi, virtualisointi

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Tarkiainen, Simo	Bachelor's thesis	April 2021
		Language of publication: Finnish
	Number of pages: 50	Permission for web publication: Yes

Title of publication

Automatization of cybersecurity testing environment

Degree programme

Information and Communications Technology

Supervisor(s)

Tero Kokkonen, Jarmo Nevala

Assigned by

CYBERDI

Abstract

The goal of the study was to automatize the creation of a virtual environment that can be utilized by the client in cybersecurity training and in other similar testing. The virtual environment was also intended to be used at home in addition to education, so the set requirements weren't allowed to be set too high.

In the beginning of the study research was done on different technologies and services which could be used to create the virtual environment and the necessary features for it. When a satisfying composition was conceived the focus was shifted onto the automatic building of virtual machines. When this part of the automation was finished, configuration management tools were used to install programs onto the virtual machines. These programs enabled the gathering and processing of data. The final part to be planned was the isolation of the virtual environment.

As a result of the study a procedure was constructed. This procedure contains the instructions for different automation programs which create the virtual environment. The creation of the virtual environment is fully automated from the beginning, making it user-friendly and efficient. The virtual environment consisted of multiple virtual machines, each of which having their own role. Different services and programs were installed on these virtual machines depending on these roles. The services enabled the collection, saving, and examination of logs. From these logs it was possible to analyze changes caused by malware and security incidents.

The procedure created in the study met the requirements and the goals of the client, as the procedure was able to automatically create a virtual environment, which provides the tools necessary for cybersecurity testing.

Keywords/tags (subjects)

Cybersecurity, automatization, virtualization

Miscellaneous (Confidential information)

Sisältö

Kuviot

Lyhenteet

SIEM Security Information and Event Management on ratkaisu, jolla valvotaan jonkin verkon tietokoneiden ja muiden laitteiden sisäisiä tapahtumia keräämällä niistä lokeja. (Anastasov & Davcev 2014.)

ELK Elastic-stack, johon kuuluu Elasticsearch, Logstash ja Kibana. (What is the ELK Stack? N.d.)

UFW Uncomplicated Firewall on graafinen käyttöliityymä Linuxin iptables-ohjelmalle. (Murray 2020.)

WinRM Windows Remote Management on Windowsin ominaisuus, joka mahdollistaa kommunikoinnin eri Windows-tietokoneiden välillä verkossa. (White, Coulter, Batchelor, Jacobs & Satran 2018.)

SSH Secure Shell -protokollaa käytetään etäyhteyden luomiseen kahden laitteen välillä epäluotettavassa verkossa. Se tarjoaa datan siirtoon autentikaatiota, eheyttä sekä luottamuksellisuutta. (IETF/RFC 4251:2006.)

AD DS Active Directory Domain Services on Microsoftin julkaisema palvelu, joka mahdollistaa toimialueen luomisen Windows-palvelimelle. (Petters 2020.)

JSON JavaScript Object Notation on kevyt ja helposti luettava tekstiformaatti. (ISO/IEC 21778:2017.)

DNS Domain Name System on protokolla, joka kartoittaa käyttäjäystävälliset verkkotunnukset IP-osoitteisiin. (What is DNS? N.d.)

1 Johdanto

1.1 Opinnäytetyön tausta ja tavoite

Kyberturvallisuus on tänä päivänä oleellinen osa sekä ihmisten että myös yritysten elämän ja toiminnan turvallisuutta. Kyberturvallisuuden päätarkoitus on säilyttää datan yksityisyys, saavutettavuus ja eheys digitaalisella alustalla (SFS-ISO/IEC 27032:2012). Yksityistä dataa uhkaavat tietoturvauhat tulevat monissa eri muodoissa. Kun uusia teknologioita ja ratkaisuja kehitetään niitä vastaan, kehittyvät myös haittaohjelmat samanaikaisesti. Näiden seikkojen vuoksi organisaatioiden tulee jatkuvasti panostaa tietoturvansa ylläpitoon ja päivittämiseen esimerkiksi palomuureilla, antivirus-ohjelmistoilla ja henkilökunnan koulutuksella. Mikäli näistä panostuksista huolimatta tietovuotoja kuitenkin tapahtuu, voivat seuraukset olla pahimmillaan hyvinkin vakavat.

Tietoturvan ylläpidon laiminlyönti mahdollistaa kyberrikollisten toimintaa. Kyberrikollisella tarkoitetaan henkilöitä, jotka tekevät rikoksia tai muita haitallisia toimia tietokonetta tai muuta digitaalista komponenttia apuna käyttäen. Kyberrikollisuuden tarkoitusperät ovat useimmiten taloudellisia, mutta tekojen taustalla olevat motiivit voivat olla myös poliittisia tai inhimillisiä. Kyberrikos voi olla esimerkiksi laittoman tai arkaluontoisen tiedon jakamista verkkosivuilla, jota voidaan mahdollisesti edelleen käyttää kyberrikosten tuottamiseen. Kyberrikollisuuteen kuuluu myös erilaisten haittaohjelmien luominen. Kyseiset haittaohjelmat saattavat olla hyvinkin monimutkaisia työkaluja, joita käytetään varsinaisten hyökkäysten tekemiseen johonkin kohteeseen. (Arief, Adzimi & Gross 2015.)

Kyberrikollisuuden lisääntyminen on seurausta yhteiskunnan digitalisoitumisesta. Digitalisoituminen tarkoittaa palveluiden ja tuotteiden siirtymistä digitaaliseen muotoon. Meneillään oleva maailmanlaajuinen pandemia on omalta osaltaan edelleen vahvistanut kyseistä kehitystä. Näin siksi, että yritykset ovat muuttuneen tilanteen vuoksi olleet pakotettuja muuttamaan toimintatapojaan ja siirtämään toimintaansa yhä enenevässä määrin verkkoon. Muutos on eri aloilla erilainen, mutta kaikilla

aloilla on samalla tavalla välttämätöntä panostaa tietoturvaan ja sen ammattilaisiin muutoksesta selviytyäkseen. (Almeida, Santos & Monteiro 2020.) Tämän vuoksi myös tietoturvakoulutukseen tultaneen lisäämään tulevaisuudessa resursseja ja uusia koulutustapoja.

Opinnäytetyössä on tavoitteena automatisoida virtuaaliympäristön rakennus, jota voidaan käyttää kyberturvallisuustestauksissa ja -koulutuksissa. Tarkoituksena on täten rakentaa automaattisesti virtuaalinen ympäristö, josta löytyy tarvittavat työkalut haittaohjelmien ja muiden vastaavien uhkien tutkimista varten. Kehitettävää prosessia, johon sisältyy virtuaaliympäristön rakennuksen eri vaiheet, voidaan kutsua myös toimintamalliksi. Kyseinen toimintamalli on suunnattu käytettäväksi opetuksessa, mutta myös sen kotikäyttö on asiasta kiinnostuneille mahdollista. Mahdollisen kotikäytön vuoksi laitteistolle asetetut vaatimukset eivät saa olla liian korkeat. Lisäksi kaikkien toimintamallissa käytettävien komponenttien tulee olla ilmaisia tai avointa lähdekoodia.

1.2 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii vuonna 2018 aloitettu CYBERDI-projekti, joka toteutetaan Jyväskylän ammattikorkeakoulun ja Poliisiammattikorkeakoulun yhteistyönä. Projektin rahoittaa Opetus- ja kulttuuriministeriö. (CYBERDI n.d.)

CYBERDI-projektilla on kolme päätavoitetta. Projektilla pyritään parantamaan yleistä tietoturvaosaamista terveydenhuollossa ja yrityksissä. Lisäksi projekti pyrkii sosiaalista mediaa apuna käyttäen parantamaan erityisesti kouluikäisten nuorten tietoisuutta kyberturvallisuuteen liittyvistä uhista. Projektin toisena tavoitteena on tehdä yhteistyötä muiden vastaavien kyberrikollisuutta estävien ja tutkivien organisaatioiden ja ammattilaisten kanssa kansainvälisessä mittakaavassa. Kolmantena tavoitteena on kyberrikollisuuden ehkäisy yhteistyössä Poliisin kanssa. Tähän pyritään nykypäivän teknologioita käyttämällä sekä myös uutta teknologiaa tuottamalla. (CYBERDI projektiesittely n.d.)

Opinnäytetyössä on osana CYBERDI-projektia tavoitteena luoda edellä mainittua uutta teknologiaa eli pyrkiä kehittämään uusi työkalu tietoturvauhkien tutkimista varten. Tavoitteena on työkalua hyödyntämällä pystyä testaamaan erilaisia haittaohjelmia sekä koulutuksessa että myös kotikäytössä.

2 Tutkimusasetelma

2.1 Tutkimuskysymys

Opinnäytetyössä tutkitaan, kuinka saadaan automaattisesti rakennettua virtuaalinen ympäristö tietoturvatestausta varten. Näin ollen opinnäytetyössä etsitään vastausta kysymykseen

- Kuinka saadaan automaattisesti rakennettua virtuaalinen ympäristö tietoturvauhkien testausta varten?

2.2 Tutkimusmetodologia

Opinnäytetyön tutkimusmetodologiaa valittaessa harkittiin useampaa eri menetelmää. Koska tavoitteena on luoda uusi ratkaisu olemassa olevaan ongelmaan sekä toiminnallinen että soveltava menetelmä olivat varteenotettavia vaihtoehtoja. Lopulta päädyttiin kuitenkin konstruktiiviseen metodiin, koska sen koettiin olevan tutkimusotteena tässä tapauksessa toimivin.

Konstruktiivisen tutkimusotteen tavoitteena on luoda uusi konstruktio jonkin oikean elämän ongelman tai tarpeen ratkaisemista varten. Tällöin konstruktiolla voidaan tarkoittaa lähes mitä tahansa kehitettyä uutta mallia, suunnitelmaa, rakennetta, tai muuta tuotetta. Konstruktion luomiseen käytetään jo olemassa olevaa teoriaa ja tietoa, ja näitä käyttämällä opitaan samalla uutta. (Lukka 2014, Mitä tarkoittaa.)

Konstruktiivinen tutkimusote soveltuu opinnäytetyön toteutukseen, koska toimeksiantajan tarve toimintamallille on oikeasta elämästä nouseva käytännön ongelma, johon pyritään vastaamaan luomalla uudenlainen konstruktio jo olemassa olevaa materiaalia hyödyntämällä. Konstruktiivisen metodin periaatteisiin kuuluu, että mikäli haluttua lopputuotetta ei jostakin syystä saavuteta, opitaan silti vanhasta tiedosta uutta. Tällöin voidaan tarkastella missä onnistuttiin ja epäonnistuttiin, ja toimia tulevaisuudessa tämän uuden tiedon varassa. Myös uuden oppiminen lähdemateriaalista on myös oleellinen osa konstruktiivista metodia. (Lukka 2014, Prosessi.)

Tämän opinnäytetyön kontekstissa konstruktiivinen tutkimusote tarkoittaa sitä, että halutun toimintamallin toteuttamiseksi hyödynnetään valmiita palveluita, ohjelmia ja malleja, joita on jo olemassa. Mikäli näissä valmiissa kokonaisuuksissa ilmenee puutteita, pyritään niitä täydentämään ja luomaan uusia tarpeen mukaan. Tämä voi kuitenkin osoittautua haasteelliseksi.

Yksi konstruktiivisen tutkimusotteen heikkouksista piileekin lähteiden ja muun valmiiden materiaalien saatavuudessa. Mikäli nämä eivät osoittaudu luotettaviksi, hyödyllisiksi tai muuten toimiviksi, tarkoittaa se suurempaa työmäärää tutkijalle. Saattaa myös olla, että tällöin joudutaan työskentelemään enemmän oman osaamisen ulkopuolella, mikä voi johtaa huonompaan lopputulokseen. Tällaisissa tapauksissa aikataulu voi myös venyä odotettua pidemmäksi, kuin mitä toimeksiantajan kanssa on sovittu.

Konstruktion luominen sisältää myös muita riskejä tai heikkouksia. Konstruktion luomiset ovat yleensä aikaa vieviä prosesseja, joihin sitoutuminen vaikuttaa olevan toimeksiantajille yleinen ongelma. Sitoutumattomuus saattaa tarkoittaa myös sitä, että ongelman alkuperäinen vakavuus on yliarvioitu, eikä sitä ole analysoitu tarpeeksi huolellisesti. Toimeksiantaja voi myös tuntea epävarmuutta siitä, että konstruktion myötä liikesalaisuuksia tai muuta arkaluontoista tietoa voi vuotaa ulkopuolisille, mikä voi johtaa hankalimmillaan koko projektin lopettamiseen. (Lukka 2014, Potentiaalisia riskejä.)

Konstruktiivista metodia käyttävillä, etenkin nuorilla tutkijoilla, saattaa olla haasteellista ylläpitää riittävää kommunikaatioita toimeksiantajan kanssa. Tutkimusprojekti voi vaatia tiivistä yhteistyötä molemmilta osapuolilta, ja mikäli tässä ei onnistuta, voi koko projekti raueta. (Lukka 2014, Potentiaalisia riskejä.) Kommunikaation puute voi tarkoittaa myös objektiivisuuden puutetta, mikäli tutkija ei hae työhön näkemyksiä toimeksiantajalta tai joltakin muulta taholta. Tämä saattaa johtaa helposti siihen, että lopputuote kuvastaa ainoastaan tutkijan näkemystä asiasta, jolloin se voi jäädä suppeaksi, puutteelliseksi tai vialliseksi.

Konstruktiivisella tutkimusotteella on myös monia vahvuuksia. Sen tapa kohdistaa huomio johonkin todellisen elämän ongelmaan on hyödyllistä sekä toimeksiantajalle että tutkijalle. Toimeksiantajan näkökulmasta tärkeintä on saada jokin sisäinen ongelma tarkasti analysoitavaksi. Tutkija puolestaan tuo ongelman ratkaisemiseksi uutta ulkopuolista tietoa ja osaamista ratkaisemista varten, jota hyödyntäen tutkimusta saadaan eteenpäin ja parhaimmillaan tutkittava ongelma ratkaistuksi. Tutkimusprosessin myötä tutkija puolestaan saa ainutkertaisen mahdollisuuden päästä kiinni organisaatioiden sisäisiin aiheisiin sekä tietoihin, jotka eivät muuten olisi saatavilla. (Lukka 2014, Potentiaalisia etuja.)

Konstruktiivisen metodin suhde sen lähdemateriaalin on myös yksi sen vahvuuksista. Metodi tuo lisää arvoa entiselle tiedolle hyödyntämällä sitä esimerkiksi käytännöllisen applikaation luomiseen, joka puolestaan auttaa edistämään tieteitä yleisellä tasolla (Lukka 2014, Potentiaalisia etuja). Laadukas lähdemateriaali nopeuttaa tutkimuksen etenemistä, kun tutkija voi pohjata tutkimustansa jo olemassa olevaan materiaaliin.

2.3 Tutkimuseettinen tarkastelu

Opinnäytetyössä seurataan Jyväskylän Ammattikorkeakoulun eettisiä periaatteita, ja noudatetaan tekijänoikeuslakeja käyttämällä oikeaoppisia lähdeviittauksia ja seuraamalla julkaisijoiden uudelleenjakeluohjeita. Uudelleenjakeluohjeet koskevat lähinnä

avoimen lähdekoodin ohjelmia, kun taas lähdeviitteet liittyvät erilaisiin tekstilähteisiin.

Työssä hyödynnetään ensisijaisesti tuotteita, jotka ovat avointa lähdekoodia tai muuten ilmaisessa jakelussa joko täys- tai kokeiluversioina. Työssä noudatetaan tuotteiden lisenssejä ja niissä määriteltyjä ohjeita.

Opinnäytetyön toteutus ei vaadi henkilötietojen tai muiden arkaluontoisten aineistojen keräämistä tai käsittelyä, eikä käytettyyn aineistoon liity näin ollen yksityisyyden loukkaamisen mahdollisuutta. Tämän vuoksi tutkimuksessa ei ole tarpeen tutkia ja soveltaa Euroopan Unionin yleistä tietosuoja- tai muita vastaavia asetuksia. Vastaavasti tutkimukselle ei ole myöskään tarvetta hakea tutkimuslupaa Jyväskylän ammattikorkeakoululta, koska opinnäytetyössä ei käsitellä oppilaitoksen henkilökuntaa tai opiskelijoita.

Opinnäytetyössä tuotettavaa toimintamallia voi mahdollisesti hyödyntää myös rikollisiin tarkoituksiin. Koska toimintamallin tarkoitus on tietoturvatestaus, voivat väärinkäyttäjät hyödyntää sitä omien haittaohjelmien ja niiden jälkien tutkimiseen ja sen myötä hyökkäysten kehitykseen. Varsinaisten hyökkäysten laukaisuun ympäristöstä ei kuitenkaan ole hyötyä.

Poikkeus edelliseen voi olla huolimaton käyttäjä, joka tahattomasti päästää jonkin haittaohjelman virtuaaliympäristöstä ulos isäntäkoneeseen ja sen myötä leviämään verkkoon. Tällainen tilanne saattaisi syntyä puutteellisesta ympäristön eristyksestä tai edistyneestä viruksesta, joka ymmärtää toimivansa virtuaaliympäristössä.

3 Käytetyt teknologiat

3.1 Teoria

3.1.1 Virtualisointi

Virtualisoinnilla tarkoitetaan teknologiaa, jolla jaetaan fyysisen tietokoneen resurssit useaksi loogiseksi osaksi. Näillä osilla ajetaan simuloituja ohjelmistoja tai laitteistoja, jotka ovat eristetty isäntäkoneen käyttöjärjestelmästä. Tämän avulla isäntäkoneen laitteistosta saadaan irti suurempi hyöty ja saadaan samalla eristettyä palvelut omiin instansseihinsa. Isäntäkoneella oleva virtualisointialusta hallinnoi virtuaalikoneita ja niille myönnettäviä resursseja, kuten muistia ja prosessoria, tarpeen mukaan. (Murphy n.d.)

Koska kokonaisen testiympäristön rakentaminen kotioloissa käyttämällä fyysisiä tietokoneita on epäkäytännöllistä tilankäytön ja kustannusten näkökulmasta, päätettiin projektissa hyödyntää toimintamallin rakentamisessa isännöityä virtualisointia. Näin toimittaessa saadaan luotua useampi virtuaalinen tietokone yhden fyysisen tietokoneen käyttöjärjestelmän päälle, jolloin ympäristö on mahdollista rakentaa ilman ylimääräistä investointia infrastruktuuriin. Virtualisointialustan asentaminen käyttöjärjestelmän päälle on myös huomattavasti helpompaa kuin sen asennus suoraan rautatasolle, mikä on oleellista käyttäjäystävällisyyden parantamisen kannalta. (Souppaya, Scarfone & Hoffman 2011, 10–12.)

3.1.2 Automatisointi

Automatisointi on prosessi, jossa luodaan järjestelmiä, jotka puolestaan suorittavat toistettavia prosesseja ilman ihmisen väliintuloa. Automatisoinnin luomiseen käytetään erilaisia ohjelmistoja ja ohjeita, malleja ja suunnitelmia, joissa ovat tarvittavat tiedot halutun lopputuloksen saamista varten. Yleensä kun manuaalisesti suoritetta-

vat prosessit automatisoidaan, saadaan alkuinvestoinnin jälkeen säästöjä työtunneissa. Rakennus ja muut rutiininomaiset toimenpiteet nopeutuvat, ja niissä tapahtuu myös vähemmän virheitä, kun ne suoritetaan johdonmukaisesti jokaisella kerralla. (IT Automation n.d.)

Edellä mainittujen seikkojen vuoksi toimintamallin rakentamisessa hyödynnetään automaatiota: tavoitteena on saada jokaisella kerralla luotettavasti samanlainen ympäristö palveluilla ja asetuksilla aikaiseksi. Tästä on erityisesti hyötyä, mikäli on tarve suorittaa ympäristön luominen useampaan kertaan. Huomattavan ajan säästön lisäksi automaatio madaltaa käyttäjältä vaadittavaa taitotasoa, kun käyttäjän ei tarvitse perehtyä syvemmin jokaisen teknologian käyttöön.

3.1.3 Tietoturvatiedon ja -tapahtumien hallintajärjestelmä

Tietoturvatiedon ja -tapahtumien hallintajärjestelmä, SIEM (Security Information and Event Management), on ratkaisu, jolla valvotaan jonkin verkon tietokoneiden ja muiden laitteiden sisäisiä tapahtumia keräämällä niistä lokeja. Kerätyt tiedot tallennetaan myöhempää tarkastelua varten, ja tarpeen mukaan voidaan myös luoda hälytyksiä valituille tapahtumille, kuten havaituille haittaohjelmille. Tämän avulla saadaan valvottua ympäristön laitteita yhdestä paikasta, helpottaen asiantuntijoiden työtä. (Anastasov & Davcev 2014.)

SIEM valittiin käytettäväksi, koska se mahdollistaa tietoturvauhkien ja niiden jälkien tutkimisen, mikä on toimintamallilla rakennettavan virtuaaliympäristön tarkoitus. Kun ympäristöön tuodaan tutkittavaa materiaalia, eli haittaohjelmia, voidaan lokeista nähdä niiden aiheuttamia tapahtumia. Näin mahdollistetaan niiden tarkastelu ja parempi ymmärtäminen.

3.1.4 Toimialue

Toimialue on palvelu, jossa yksi tai useampi palvelin hallitsee useampaa tietokonetta. Kun tietokoneita lisätään toimialueeseen, voidaan niillä paikallisten kirjautumistietojen sijasta kirjautua sisään toimialueen tunnuksilla. Näin käyttäjä pääsee käsiksi toimialueen tarjoamiin palveluihin ja resursseihin, joihin ei muuten olisi mahdollista päästä käsiksi. Ylläpitäjä pystyy hallitsemaan käytänteitä, joita toimialueeseen liitetyt tietokoneet ja käyttäjät noudattavat. Yleensä tietokoneet ja palvelimet ovat samassa verkossa, mutta käyttäjät pystyvät myös muodostamaan niihin etäyhteyden. (Hoffman 2017.)

Virtuaaliympäristöön lisätään toimialue, jotta saataisiin luotua monipuolisempia testausmahdollisuuksia. Näin tarjotaan edellytykset tutkia haittaohjelmien toimintaa toimialueessa, sen palveluissa ja sitä ylläpitävässä palvelimessa. Toimialueen tarjoama käyttäjänhallinta on erityisen hyödyllinen ominaisuus. Käyttäjänhallinnalla voidaan jakaa eri käyttäjille erilaisia oikeuksia, ja sitä kautta testaamaan näiden oikeuksien mahdollisia riskejä ja vuorovaikutusta haittaohjelmien kanssa.

3.2 Komponentit

3.2.1 Virtualbox

Virtualbox on Oracle Corporationin omistama ja jatkuvasti kehittämä avoimen lähdekoodin virtualisointialusta, jonka avulla pystytään ajamaan montaa eri käyttöjärjestelmää vieraana oman käyttöjärjestelmän alaisena. Käyttäjä pystyy luomaan uuden virtuaalikoneen helposti käyttöjärjestelmän levykuvasta. Prosessin aikana valitaan RAMin ja tallennustilan määrä, prosessorien määrä, sekä muut halutut asetukset. (VirtualBox 2019.)

Virtualbox valittiin käytettäväksi virtualisointialustaksi, koska se on tehokas, helppokäyttöinen ja ilmainen. Virtualbox toimii Windowsilla, Linuxilla, sekä muilla käyttöjär-

jestelmillä. Koska Virtualbox on avoimen lähdekoodin ohjelma, siinä esiintyvät ongelmat korjataan yleensä nopeasti, ja lisäksi uusia ominaisuuksia lisätään. Suurin kilpailija Virtualboxille oli VMWare, josta on monta eri versiota. (Vojnak, Đorđević, Timčenko & Štrbac 2019.) Osa VMWaren tuotteista on maksullisia, mutta myös maksuton ratkaisu löytyy VMWare Playerin muodossa. VMWaren tuotteita ei kuitenkaan päädytty käyttämään näihin liittyvien lisenssien vuoksi.

3.2.2 Vagrant

Vagrant on Hashicorpin julkaisema avoimen lähdekoodin työkalu, jolla hallinnoidaan virtuaalisia ympäristöjä. Vagrantin tarkoituksena on pystyä luomaan ja käynnistämään komentorivin kautta ennalta määritetty ympäristö automaattisesti. Tämä on erityisen hyödyllistä, kun halutaan luoda tarkoilla määrityksillä samanlainen ympäristö useampaan kertaan pienellä osanotolla, kuten esimerkiksi kehitysympäristön tarvetilanteessa. Koska vaaditut määritykset tehdään tekstitiedostoon, on niiden jakaminen ja muokkaaminen erityisen helppoa. (What is Vagrant? n.d.)

Vagrantin valmiita pakattuja ympäristöjä, eli Vagrant laatikoita (Vagrant Box), on saatavissa ilmaiseksi Vagrant Cloudissa, mistä niitä voi ladata ja lisätä omaan käyttöön. Mikäli ei haluta käyttää jonkun muun luomia ympäristöjä, niin omat virtuaalikoneet pystytään helposti pakkaamaan Vagrant-laatikoiksi, jotka sitten voi ottaa käyttöön ja halutessaan myös jakeluun.

Tekstitiedosto, johon määritykset tehdään, on nimeltään Vagrantfile. Kyseinen tiedosto perustuu Ruby-ohjelmointikieleen, ja sinne tehdään kaikki virtuaaliympäristön määritykset, kuten käytettävä virtualisointialusta, muonittajat ja monet muut asetukset, joilla voidaan tehdä halutut säädökset.

Vagrant valittiin sen helppolukuisen Vagrantfilen ja kätevien laatikoiden vuoksi. Näiden avulla saadaan helposti luotua ja muokattua ympäristöjä. Lisäksi se on ilmainen

ja toimii Windowsilla muiden käyttöjärjestelmien lisäksi. Vagrantilla saa helposti testattua monia eri käyttöjärjestelmiä julkisten laatikoiden avulla, mutta toisaalta nämä eivät kuitenkaan aina toimi, mikä voi häiritä käyttäjiä. Myös graafisen käyttöliittymän puute voi olla uusille käyttäjille haaste.

Toisena varteenotettavana vaihtoehtona projektissa käytettäväksi alustaksi oli Docker. Koska Docker käyttää isäntäkoneen käyttöjärjestelmää prosessien ajamiseen uuden luomisen sijasta, on se resurssien näkökulmasta nopeampi sekä kevyempi käyttää. Tämä voi osoittautua myös turvallisuusuhaksi, mikäli näitä prosesseja ei saada eristettyä huolellisesti isäntäkoneesta. (Šimec, Držanić & Lozić 2018.) Lisäksi toimintamallin luoma virtuaaliympäristö halutaan tehdä näennäisesti mahdollisimman tavalliseksi, mikä voi olla Dockerilla haastavaa.

3.2.3 Packer

Vagrantin lisäksi Hashicorp on julkaissut myös Packerin. Packerin käytön tarkoituksena on automatisoida käyttöjärjestelmän asentaminen virtuaalikoneeseen, pilveen tai jollekin muulle alustalle. Asennus tapahtuu käyttämällä JSON-tiedostoa, joka on jaettu pääasiassa kolmeen osaan: rakentajat (builders), muonittajat (provisioners) ja jälkikäsittelijät (post-processors).

Rakentajat määrittävät virtualisointiin liittyvät asiat, kuten mitä alustaa ja levykuvaa käytetään, sekä mitkä tiedostot ladataan käsiteltävälle virtuaalikoneelle. Jälkikäsittelijät puolestaan ajavat rakentajat-osiossa tietokoneelle ladatut tiedostot, ja antaa mahdollisuuden käyttää jotakin konfiguraationhallintasovellusta, kuten Chef, Salt, tai Ansible. Jälkikäsittelijät taasen muokkaavat Packerin tuotoksen jonkin muun virtualisointiohjelman formaattiin. Lopputuotteen pystyy esimerkiksi lataamaan suoraan Google Cloudiin pyörimään. (Lakhera 2019.)

Kun uutta käyttöjärjestelmää asennetaan, se kysyy liudan monia eri kysymyksiä, kuten esimerkiksi näppäimistö- ja kieliasetuksia. Packer vastaa näihin kysymyksiin automaattisesti hyödyntämällä käyttöjärjestelmän mukaan eri tiedostoja. Linuxiin perustuvia virtuaalikoneita kasattaessa Packer jakaa http:n kautta konfiguraatiotiedoston, josta se poimii vastaukset käyttöjärjestelmän asennuskysymyksiin. Windows käyttää samaan tarkoitukseen Autounattend.XML-tiedostoa, jossa käydään vastaavasti kaikki asennuksen vaatimat asetukset läpi.

Packer valittiin käytettäväksi, koska se mahdollistaa prosessin aloittamisen levykuvasta asti, jolloin edellä mainittu Vagrantin puute saadaan korjattua. Lisäksi molemmat, sekä Vagrant että Packer, ovat Hashicorpin tuotteita, minkä vuoksi yhteensopivuus ei muodostu haasteeksi. Packerin hyödyntäminen on välillä haasteellista johtuen siitä, että käyttöjärjestelmiä on monia erilaisia, eikä universaalia ratkaisua pystytä luomaan esimerkiksi kaikille Windowsin versioille. Mutta toimiessaan käyttöjärjestelmän koko asennus hoituu automaattisesti alusta loppuun Packerin ohjaamana.

Packerin käytön sijaan olisi projektissa vaihtoehtoisesti voinut harkita Windowsin ja Linuxin asennusta käyttäen niiden omia työkaluja. Kuitenkin, koska Packer perustuu näihin ja mahdollistaa automaatiota pidemmälle, ei niiden käytölle nähty tarvetta.

3.2.4 Kommunikaattorit

Koska Packer ja Vagrant eivät toimi virtualisointialustan kautta, tulee niiden käyttää jotakin toista kommunikointitapaa virtuaalikoneiden kanssa kommunikoimiseen. Tähän tarpeeseen sekä Packer että Vagrant käyttävät ohjelmia Secure Shell (SSH) ja Windows Remote Management (WinRM).

SSH-protokollaa käytetään etäyhteyden luomiseen kahden laitteen välillä epäluotettavassa verkossa. Se tarjoaa datan siirtoon autentikaatiota, eheyttä sekä luottamuksellisuutta. (IETF/RFC 4251:2006.) Sekä Vagrant että Packer käyttävät tätä teknologiaa oletuksena virtuaalikoneisiin yhdistämiseen.

WinRM-palvelu on Windowsin ominaisuus, joka perustuu WS-Management-protokollaan. Tämä protokolla määrittelee järjestelmille yhteisen tavan kommunikoida verkon välityksellä. Näin WinRM-palvelu pystyy siirtämään dataa Windowsia käyttävien tietokoneiden kesken. (White, Coulter, Batchelor, Jacobs & Satran 2018.) Kun Packer tai Vagrant pyrkivät yhdistämään Windowsia käyttävään virtuaalikoneeseen, suositellaan silloin WinRM-kommunikaattoria. Tämä johtuu siitä, että SSH ei toimi Windowsin kohdalla valmiiksi.

3.2.5 Ansible

Ansible on avoimen lähdekoodin automatisointiohjelma. Ansiblen pääasiallinen tarkoitus on hoitaa kaikki kohdetietokoneiden hallinnointi ja provisiointi, olivat ne sitten fyysisiä tai virtuaalisia. Ansiblen käyttö on yksinkertaista, sillä se ei vaadi agenttia eikä ylimääräisiä turvallisuussäädöksiä, kuten palomuurien porttien avaamista. Lisäksi Ansible käyttää YAML-merkintäkieltä, joka on ihmisen luettavissa, mikä tekee tiedostojen muokkaamisesta erittäin yksinkertaista. (How Ansible Works n.d.)

Ansiblen toiminta perustuu ns. pelikirjoihin (Playbooks), jotka on kirjoitettu YAML-merkintäkielellä. Pelikirjoissa määritellään vaiheittain kaikki automaattisesti suoritettavat askeleet, jotka halutaan ajaa hallinnoitavalla tietokoneella. Näitä askeleita kutsutaan moduuleiksi ja kullakin niillä on oma tarkoituksensa. (What is an Ansible playbook? N.d.)

Ansiblen kaltaisia muita kokoonpanohallintatyökaluja ovat esimerkiksi Chef, Salt, ja Puppet. Ansible valittiin käytettäväksi, koska sillä työskenteleminen on yksinkertaista pelikirjoja ja moduuleja hyödyntäen. Lisäksi kyseinen teknologia oli ennestään tutkijalle tuttu, mikä myös vaikutti päätökseen. Ansiblen haittapuoli on sen yhteensopivuus Windowsin kanssa. Tämä ei kuitenkaan vaikuttanut ylitsepääsemättömältä haasteelta, koska näiden Windowsia käyttävien virtuaalikoneiden on mahdollista hyödyntää eri teknologiaa.

3.2.6 Komentojonotiedostot

Komentojonotiedostot tai skriptit sisältävät tekstimuodossa olevia komentoja, jotka ajettaessa suoritetaan tietokoneen komentorivin kautta annetussa järjestyksessä. Kullakin käyttöjärjestelmällä on omanlaisensa tulkki, joka ajaa nämä komennot. Kirjoittamalla komennot tekstitiedostoon saadaan ne varastoitua helposti luettavaan ja muokattavaan tiedostoon myöhempää käyttö varten. (Shell script n.d.)

Komentojonotiedostojen valinta perustui Ansiblen puutteelliseen yhteensopivuuteen Windowsin kanssa. Jotta Ansiblella saataisiin suoritettua Windows-koneiden konfigurointi, tulisi se ajaa Linuxia käyttävältä koneelta, jonka vuoksi Linux käyttöjärjestelmänä tulisi isäntäkoneelle vaatimukseksi. Käyttämällä Powershell-skriptejä saadaan konfiguroitua kaikki virtuaaliympäristön Windows-koneet huolimatta isäntäkoneen käyttöjärjestelmästä. Koska Powershell on Windowsin sisäinen ohjelma, sitä ei tarvitse erikseen asentaa, ja se toimii luotettavasti. Myös Ansiblen kaltaiset ohjelmat voisivat toimia Windowsin kanssa, mutta Powershell toimii ilman laajempaa ylimääräistä konfiguraatiota, jonka vuoksi se valittiin.

3.2.7 ELK-stack

Yksi tapa luoda SIEM-palvelu on käyttää Elasticin julkaisemaa ELK-pinoa (ELK-stack). ELK-pinoon kuuluuvat Elasticsearch-, Logstash-, Kibana-, ja Beats-ohjelmat. SIEMin pystytykseen ei kuitenkaan tarvita Logstashia, vaan sen voi jättää toteutuksesta pois. (What is the ELK Stack? N.d.)

Elasticsearch on ilmainen hakumoottori, joka perustuu Apache Luceneen. Elasticsearch pystyy käsittelemään suuria määriä monenlaisia eri datatyyppejä. Kyseessä on siis palvelin, jolta saadaan JSON-pyynnöillä JSON-dataa. (Abueg 2020.)

ELK-pinon käyttöliittymä on Kibana, joka on myös avoimen lähdekoodin tuote. Kibana toimii erinomaisesti Elasticsearchin kanssa luomalla datasta monenlaisia graafisia visualisointeja, kuten pylväs- ja ympyrätaulukoita. Lisäksi Kibana antaa mahdollisuuden muokata ja jakaa dataa eri muodoissa. (Kibana n.d.)

Lisäksi ELK-pino käyttää Beatseja, jotka ovat pienikokoisia ja tehokkaita työkaluja. Beatseja käytetään lokien keräämiseen. Keräämisen jälkeen lokit lähetetään Elasticsearch-hakumoottorille säilytystä ja käsittelyä varten. Beatsejä on monenlaisia, ja niitä voi kuka tahansa kehittää, sillä ne perustuvat Go-ohjelmointikielen libbeat-ohjelmistokehykseen. SIEMin näkökulmasta Beatseista oleellisin on kuitenkin Auditbeat, joka seuraa käyttäjiin, prosesseihin sekä turvallisuuteen liittyviä tapahtumia kaikilla koneilla, joille se on asennettu. Lisäksi Windows-koneille asennetaan Winlogbeat. Winlogbeat poimii koneilta Windowsin tapahtumalokit, joiden avulla pystytään seuraamaan ympäristön muutoksia. (Berman 2020.)

ELK-pinon kanssa vastaaviin teknologioihin kuuluu esimerkiksi Splunk, joka on kaupallinen data-analyysialusta. Splunk toimii pitkälti vastaavalla tavalla kuin ELK-pino, mutta lisenssien vuoksi sen käyttö vaikeutuu ELKiin verrattuna, sillä ELK on ilmainen käyttää. (Son, S. & Kwon, Y. 2017.) Tämän vuoksi Splunk-palvelua ei hyödynnetty ympäristön SIEMinä.

3.2.8 Windows Active Directory Domain Services

Windows Active Directory Domain Services (AD DS) on Microsoftin julkaisema palvelu, joka mahdollistaa toimialueen luomisen Windows-palvelimelle. AD DS tarjoaa monta eri palvelua kuten käyttäjien autentikaatio, sääntöjen hallitseminen, laitteiden hallinta, asetusten jakaminen ja monia muita hyödyllisiä ominaisuuksia. (Petters 2020.) Näitä ominaisuuksia ja palveluita pystytään hyödyntämään monimuotoisten skenaarioiden luontiin erilaisia testejä varten.

Windows AD DS valittiin, koska se tekee tehtävänsä erinomaisesti ja muita varteenotettavia vaihtoehtoja ei ollut. Alun perin testattiin voisiko myös Linuxilla toimivan toimialueen luoda käyttämällä Sambaa, mutta tämä osoittautui sekä haasteelliseksi että tarpeettomaksi, koska sama saadaan aikaan helpommin käyttämällä Windowsin palvelua. AD DS saatiin Samban tavoin ilmaiseksi käyttämällä Windows-palvelimen evaluaatioversiota, joten Samban valitsemiselle ei jäänyt mitään pätevää syytä.

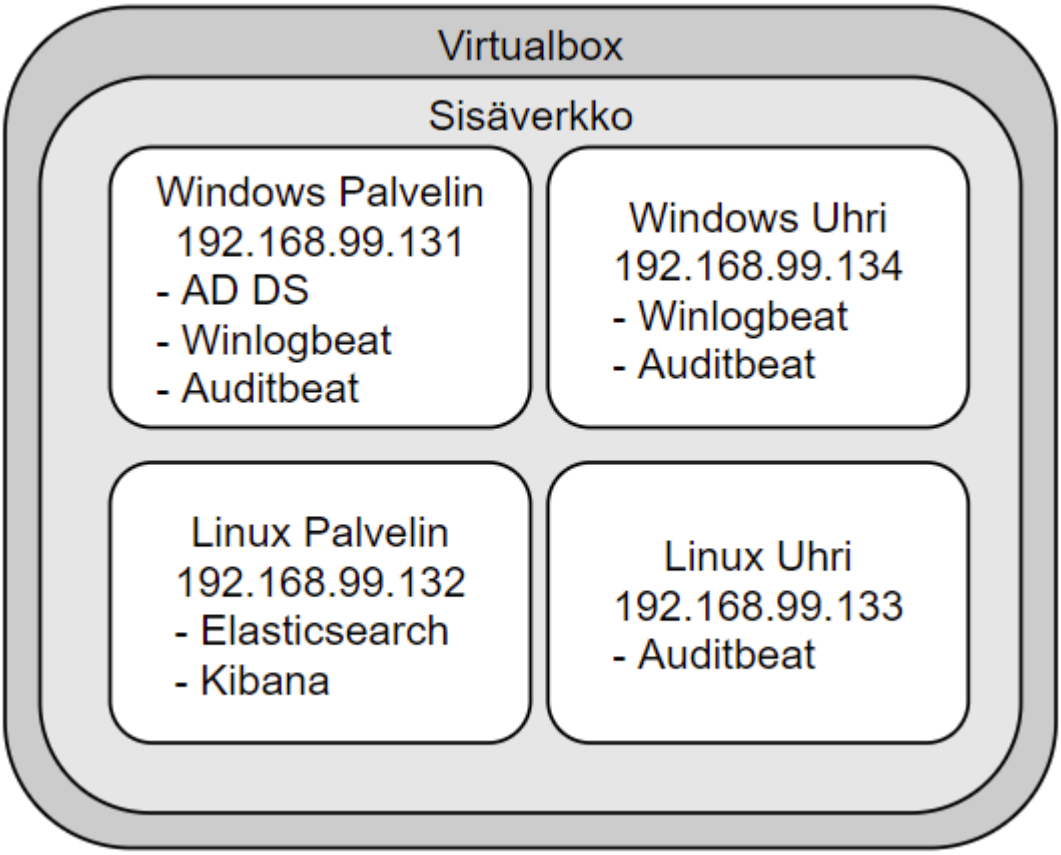
4 Toteutus

4.1 Suunnitelma

Teknologiavalintojen jälkeen suunniteltiin niiden sovellustapa sekä virtuaaliympäristön kokoonpano. Virtuaalikoneet jaettiin neljään eri rooliin, jotka olivat uhri- ja palvelinkoneet sekä Linuxille että Windowsille. Uhrikoneet nimettiin uhreiksi, koska ne suunniteltiin altistettavaksi haittaohjelmille ja hyökkäyksille. Uhrikoneiden käyttöjärjestelmiksi valittiin työpöytäversiot, jotta ne saatiin vaikuttamaan mahdollisimman tavallisilta työasemilta. Vastaavasti palvelinkoneilla valikoitiin palvelimille tarkoitettuja käyttöjärjestelmiä, jotta Linux-palvelimen kohdalla säästettäisiin keskusmuistia ja Windowsin kohdalla saataisiin tarvittavat palvelut asennettua.

Virtuaaliympäristön eristämiseksi isäntäkoneesta päätettiin virtuaalikoneiden jaetut kansiot poistaa ja verkkoyhteydet isäntäkoneeseen ja internetiin katkaista. Näin varmistettiin, että haittaohjelmat eivät voisi hyväksikäyttää näistä kumpaakaan leviämiseen. Jotta virtuaalikoneet voisivat kuitenkin kommunikoida keskenään, päätettiin luoda isäntäkoneesta eristetty sisäverkko, jota virtuaaliympäristö käytti. Tämä sisäverkko käytti osoiteavaruutta 192.168.99.0/24, sillä tämä oli toimeksiantajan virtuaalikoneille varattu osoiteavaruus. Palvelinten osoitteet sisäverkossa olivat 192.168.99.131 ja 192.168.99.132. Nämä osoitteet valittiin pysyvän samana virtuaa-

liympäristön kokoonpanosta riippumatta, jotta palveluihin saataisiin luotettavasti yhteys. Uhrien osoitteet päätettiin menevän palvelinten osoitteiden päälle, eli tavallisesti 192.168.99.133 ja 192.168.99.134. Tämä tehtiin siksi, että mikäli uhreja käynnistetään useampi kappale, niin osoitteita otetaan automaattisesti lisää käyttöön tarpeen mukaan. Esimerkiksi jos molempia uhreja käynnistettiin kaksi, tuli osoitteiksi 192.168.99.133–136. Kuviossa yksi näkyy virtuaaliympäristölle suunniteltu arkkitehtuuri.



Kuvio 1. Tavoitearkkitehtuuri

Tavoitearkkitehtuurin saavuttamiseksi suunniteltiin monivaiheinen prosessi, jossa hyödynnetään vaiheittain valittuja teknologioita virtuaaliympäristön rakentamista varten. Ensimmäinen vaihe on luoda Packerin avulla pakatut ympäristöt, joilla Vagrant sitten rakentaa virtuaalikoneet, jotka muodostavat virtuaaliympäristön. Samalla kun Vagrant rakentaa virtuaalikoneita, se asentaa niille tarvittavat palvelut, jotka mahdollistavat tietoturvatiedon keräämisen ja tutkimisen. Lopuksi ympäristön

eristykseen käytetään komentojonotiedostoa, jotta haittaohjelmien aiheuttamat uhat saataisiin minimoitua.

4.2 Laatikoiden luonti

Jotta ympäristön rakentaminen voidaan aloittaa, tulee ensimmäiseksi hankkia Vagrant-laatikot, joita käytetään virtuaalikoneiden luomiseen. Laatikoita on julkisessa jakelussa Vagrant Cloudissa ja niitä voidaan hyödyntää sellaisenaan virtuaaliympäristön rakentamisessa. Tässä työssä haluttiin kuitenkin tarjota käyttäjälle työkalut omien laatikoiden luomista varten. Hyödyntämällä näitä työkaluja käyttäjät voisivat luoda Vagrant laatikot omista käyttöjärjestelmien levykuvakkeista, poistaen tarpeen käyttää kolmannen osapuolen laatikoita Vagrant Cloudista.

Vagrant-laatikot voi luoda Vagrantin sisäänrakennetuilla komennoilla olemassa olevasta virtuaalikoneesta. Näin tehdessä käyttäjän tulee kuitenkin rakentaa virtuaalikoneet manuaalisesti käyttäen käyttöjärjestelmien levykuvakkeita, virtualisointialustaa ja Vagrantin komentoja. Tämä nostaa esille tarpeen ohjelmalle, jolla pystyttäisiin automatisoimaan tämä prosessi. Laatikoiden luomiseen voidaan käyttää Packer-ohjelmaa, joka on Vagrantin tavoin Hashicorpin tuote. Tämän vuoksi se on yhteensopiva Vagrant-laatikoiden kanssa, täydentäen tarpeen erinomaisesti. Jotta levykuvakkeista saatiin tehtyä tarvittavat laatikot, luotiin omat JSON-mallitiedostot kullekin ympäristön roolille sekä Linux- että Windows-käyttöjärjestelmille. Uhrien ja palvelinten JSON-tiedostojen luomiseen käytettiin valmiiksi olemassa olevia mallitiedostoja, jotka muokattiin tarpeen mukaan omaan sovellukseen käytettäväksi. Näin vältyttiin pitkältä prosessilta, jossa kaikki olisi tehty alusta alkaen.

Kunkin JSON-tiedoston kolme tärkeintä osaa ovat rakentajat (builders), muonittajat (provisioners) sekä jälkikäsittelijät (post-processors). Lisäksi jokaisen tiedoston yläosaan on lisätty muuttujat (variables)-osio, johon on listattu kaikki tiedostossa käytettävät muuttujat. Kuviossa kaksi näkyvät esimerkkinä Linux-uhrin muuttujat. Tätä kautta käyttäjä pystyy asettamaan oman levykuvakkeensa sijainnin iso_url-kenttään,

mahdollisesti vaatien myös tarkastussumman muutoksen. Muuttujat-osiot lisättiin tiedostojen yläosiin, jotta käyttäjien ei tarvitsisi selata tiedostoja läpi oleellisten kohtien löytämiseksi.

```
"description": "Packer template for building vagrant box of ubuntu bionic 18 desktop",
"variables": {
    "iso_url": "",
    "iso_checksum": "f295570badb09a606d97ddfc3421d7bf210b4a81c07ba81e9c040eda6ddea6a0",
    "keep_registered": "false",
    "iso_target_path": "isos/",
    "headless": "false",
    "vm_description": "Ubuntu AMD64 Bionic 18 Desktop",
    "vm_version": "1.0.0"
```

Kuvio 2. Linux-uhrin muuttujat

Kun Packer rakentaa virtuaalikonetta, se lukee siihen ohjeet JSON-tiedoston rakentajat-osiosta, joka näkyy kuviossa kolme. Siitä selviävät muun muassa virtualisointialusta, tallennustilan määrä, käytettävä levykuva ja sen tarkastussumma sekä http-kansion sijainti. Jotta Vagrant saataisiin myöhemmässä vaiheessa toimimaan, tulee Packerin asettaa SSH:n käyttäjätunnukseksi ja salasanaksi vagrant. Samassa kuviossa näkyvä boot_command puolestaan määrittää, mitä virtuaalikoneelle lähetetään automaattisena syötteenä heti käynnistymisen jälkeen. Tämä syöte käy läpi käyttöjärjestelmän asennukseen liittyviä kysymyksiä, kuten kielen ja aikavyöhykkeen valitsemisen.

```
"builders": [{
    "type": "virtualbox-iso",
    "guest_os_type": "Ubuntu_64",
    "export_opts": [
        "--manifest",
        "--vsys", "0",
        "--description", "{{user `vm_description`}}",
        "--version", "{{user `vm_version`}}"
    ],
    "http_directory": "packer_provisioning/linux/http",

    "iso_urls": [
        "{{ user `iso_url` }}",
        "https://releases.ubuntu.com/18.04/ubuntu-18.04.5-desktop-amd64.iso"
    ],
    "iso_checksum": "{{ user `iso_checksum` }}",
    "iso_target_path": "{{ user `iso_target_path` }}",

    "ssh_username": "vagrant",
    "ssh_password": "vagrant",
    "ssh_port": 22,
    "ssh_wait_timeout": "10000s",

    "disk_size": "40000",
    "guest_additions_path": "VBoxGuestAdditions_{{.Version}}.iso",
    "virtualbox_version_file": ".vbox_version",
    "vboxmanage": [
        ["modifyvm", "{{.Name}}", "--memory", "1024"],
        ["modifyvm", "{{.Name}}", "--vram", "64"],
        ["modifyvm", "{{.Name}}", "--cpus", "1"]
    ],
    "vm_name": "ubuntu_bionic_18_04_lts_desktop_packer_amd64_{{ uuid }}",
    "keep_registered": "{{user `keep_registered`}}",
    "headless": "{{user `headless`}}",

    "boot_wait": "8s",
    "boot_command": [
        "<esc><esc><f6><esc><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs><bs>
        " auto=true priority=critical noprompt ",
        " automatic-ubiquity ",
        " url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/desktop.cfg ",
        " keyboard-configuration/layoutcode=us ",
        " ubiquity/reboot=true ",
        " languagechooser/language-name=English ",
        " countrychooser/shortlist=IN ",
        " localechooser/supported-locales=en_US.UTF-8 ",
        " debian-installer/locale=en_US ",
        " netcfg/choose_interface=auto ",
        " boot=casper ",
        " initrd=/casper/initrd ",
        " quiet splash noprompt noshell ",
        " --- <wait>",
        "<enter><wait>"
  ],

    "shutdown_command": "echo 'packer' | sudo -S shutdown -P now"
}],
```

Kuvio 3. Linux-uhrin rakentajat

Kun Packer on asentanut virtuaalikoneen käyttöjärjestelmän, siirrytään muonittajat-osioon. Kyseisessä osiossa määritellään ajettavat skriptit sekä komennot, joilla ne ajetaan. Tämä näkyy kuviossa neljä.

```
"provisioners": [{
        "type": "shell",
        "script": "packer_provisioning/linux/scripts/setup.sh",
        "execute_command": "echo 'vagrant' | {{.Vars}} sudo -S -E bash '{{.Path}}'"
    },
            {
        "type": "shell",
        "script": "packer_provisioning/linux/scripts/vagrant.sh",
        "execute_command": "echo 'vagrant' | {{.Vars}} sudo -S -E bash '{{.Path}}'"
    },
    {
        "type": "shell",
        "script": "packer_provisioning/linux/scripts/cleanup.sh",
        "execute_command": "echo 'vagrant' | {{.Vars}} sudo -S -E bash '{{.Path}}'"
    }
],
```

Kuvio 4. Linux-uhrin muonittajat

Näiden skriptien avulla jokaiselle luotavalle Linuxia käyttävälle Vagrant-laatikolle tehdään SSH-oikeutettu käyttäjä ja Windowsia käytettäessä vastaava toimenpide tehdään WinRM:lle. Molemmissa vaihtoehdoissa on tarpeen siivota vielä turhia ohjelmia pois virtuaalikoneilta käyttämällä siihen tarkoitettuja skriptejä. Näin pienennetään luotavien laatikoiden kokoja.

Konstruktiivinen metodi osoittautui tässä kohtaa projektia erityisen hyödylliseksi, kun tarvittavat skriptit saatiin mallipohjien mukana. Mikäli niiden luominen ja testaaminen olisi aloitettu alusta, se olisi vaatinut runsaasti enemmän aikaa.

Kun virtuaalikoneet on käynnistetty, päivitetty ja konfiguroitu, voidaan siirtyä jälkikäsittelyvaiheeseen. Tässä vaiheessa Packer luo virtuaalikoneesta Vagrantille laatikon, joka asetetaan output-kansioon. Tämä prosessi on sama käyttöjärjestelmästä riippumatta kaikille luotaville virtuaalikoneille. Laatikon onnistuneen luomisen jälkeen virtuaalikone poistetaan, koska keep_input_artifact on asetettu epätodeksi, kuten kuviosta viisi näkyy.

```
"post-processors": [
    [{
        "type": "vagrant",
        "compression_level": 6,
        "keep_input_artifact": false,
        "output": "output/ubuntu-desktop.box"
    }]
]
```

Kuvio 5. Linux-uhrin jälkikäsittelijät

4.3 Ympäristön rakentaminen

Kun tarvittavat laatikot ovat haettu pilvipalvelusta tai luotu itse, voidaan siirtyä varsinaisen virtuaaliympäristön rakennukseen Vagrantin avulla. Ympäristön rakennukseen käytettiin Vagrantfile-nimistä tiedostoa, josta Vagrant saa kaiken tarvittavan tiedon. Kyseinen tiedosto luotiin alusta alkaen, mutta konstruktiivista metodia hyödynnettiin käyttämällä apuna erilaisia lyhyitä ohjeita oman tuotoksen tekemiseen, kuten virallisia dokumentaatioita.

Vagrantfile-tiedosto on jaettu osiin, joista kaksi ensimmäistä pyrkii parantamaan käyttökokemusta. Ensimmäinen näistä osista on kustomointiosio, joka näkyy kuviossa kuusi. Tämän osion kautta käyttäjät voivat määrittää käynnistettävät virtuaalikoneet ja niiden määrän, käytettävän laatikon sekä varattavan muistin. Ilman näitä muuttujilla tehtyjä määrittelyjä käyttäjät joutuisivat joka kerran muutoksia tehdessään etsimään tiedostosta kyseiset rivit, mutta käyttämällä muuttujia saadaan käyttökokemusta parannettua.

```
## Customization
        SIEM=1
                SiemBOX="Ubuntu/Server"
                SiemRAM=2560
        WinAD=1
                WinADBOX="Windows/Server"
                WinADRAM=4096
        LinVic=1
                LinVicBOX="Ubuntu/Victim"
                LinVicRAM=1024
        WinVic=1
                WinVicBOX="Windows/Victim"
                WinVicRAM=4096
## End of customization
```

Kuvio 6. Vagrantfilen muuttujat

Käyttökokemuksen parantamiseksi Vagrantfile-tiedostoon lisättiin myös yksinkertainen laskuri, joka pitää kirjaa allokoidun muistin määrästä (ks. kuvio seitsemän). Laskuri tulostaa yhteenlasketun allokoidun muistin määrän Vagranttia käyttäessä, minkä avulla liian pienen tai suuren muistin allokaation voi huomata heti ylösajon alkupäässä.

```
TotalRAM=0
if SIEM > 0
        TotalRAM=TotalRAM+SiemRAM
end
if WinAD > 0
        TotalRAM=TotalRAM+WinADRAM
end
if LinVic > 0
        TotalRAM=TotalRAM+(LinVic*LinVicRAM)
end
if WinVic > 0
        TotalRAM=TotalRAM+(WinVic+WinVicRAM)
end
puts "Total memory allocated: #{TotalRAM}MB"
```

Kuvio 7. Vagrantfilen muistilaskuri

Virtuaalikoneista jokainen sijaitsee omassa osiossaan, ja näistä virtuaalikoneista ensimmäisenä rakennettavana on SIEM. Jos näin ei toimita, muiden koneiden rakentaminen saattaa epäonnistua, koska ne eivät saa siihen SIEMiin yhteyttä. SIEMin osio Vagrantfilestä näkyy kuviossa kahdeksan.

```
## Elastic SIEM
if SIEM > 0
        config.vm.define "siem" do |si|
                si.vm.synced_folder ".", "/vagrant", disabled: true
                si.vm.communicator = "ssh"
                si.vm.box = SiemBOX
                si.vm.network "private_network", ip: "192.168.99.132",
                        virtualbox__intnet: "Danger network"
                si.vm.hostname = "siem"
                si.vm.provider "virtualbox" do |cfg|
                        cfg.name = "SIEM"
                        cfg.memory = SiemRAM
                end
                si.vm.provision "file", source: "./vagrant_provisioning", destination: "$HOME/"
                si.vm.provision :ansible_local do |ansible|
                        ansible.provisioning_path = "/home/vagrant"
                        ansible.playbook = "vagrant_provisioning/SIEM.yml"
                end
                si.vm.provision :ansible_local do |ansible|
                        ansible.provisioning_path = "/home/vagrant"
                        ansible.playbook = "vagrant_provisioning/FIREWALL.yml"
                end
                puts "SIEM enabled. \n"
        end
else
        puts "SIEM is not enabled. This will cause issues later on. \n"
end
## End of Elastic SIEM
```

Kuvio 8. Vagrantfile SIEMin rakennus

Vagrant käyttää yleensä tiedostojen siirtämiseen synkronoituja kansioita. Se ei ole kuitenkaan turvallisin vaihtoehto tiedonsiirtoon, sillä haittaohjelmat voivat käyttää kansioita hyödyksi isäntäkoneeseen pääsemiseksi. Tämä ongelma kierrettiin poistamalla synkronoidut kansiot ja lataamalla tarvittavat konfiguraatiotiedostot suoraan virtuaalikoneelle. Tätä keinoa voidaan hyödyntää kaikilla virtuaalikoneilla roolista ja käyttöjärjestelmästä riippumatta. Turvallisuuta pyrittiin parantamaan myös lisäämällä jokaiselle käynnistettävälle virtuaalikoneelle uusi verkkokortti, joka liitettiin uu-

teen Virtualboxin sisäiseen verkkoon. Tästä verkosta ei saa yhteyttä muihin verkkoihin, jolla estetään haittaohjelmien leviäminen paikalliseen verkkoon ja mahdollisesti myös internetiin.

Koska Ansible ei toimi Windowsilla, käytettiin Vagrantissa sen lokaalia versiota, Ansible_local. Ansible_local-muonittaja asentaa Ansiblen suoraan virtuaalikoneelle ja ajaa roolien mukaan määrätyt pelikirjat. Kuviossa yhdeksän näkyy esimerkkinä Elasticsearchin vaiheittainen asennus pelikirjan avulla SIEM-palvelimelle. Ensimmäiseksi ladataan ja puretaan tarvittava paketti verkosta, jotta saadaan tarvittavat tiedostot virtuaalikoneelle. Seuraavaksi tiedostoihin tehdään tarvittavat muutokset, jotta palveluun saadaan oikeat verkkoasetukset. Viimeiseksi palvelu käynnistetään, jolloin siihen pystytään yhdistämään IP-osoitteen kautta.

```
### ELASTICSEARCH


- name: Download Elasticsearch 7.9
  become: yes
  get_url:
    url: https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.9.0-amd64.deb
    dest: /etc/

- name: Extract Elasticsearch with dpkg
  become: yes
  command: sudo dpkg -i elasticsearch-7.9.0-amd64.deb
  args:
    chdir: /etc

- name: Configure Elasticsearch
  become: yes
  lineinfile:                                                    # Lineinfile-module is used to edit a line in a file.
    dest: "{{ item.dest }}"                                      # The file to be edited.
    regexp: "{{ item.regexp }}"                                  # The line to be replaced.
    line: "{{ item.line }}"                                      # What we are replacing the line with.
    backrefs: yes                                                # Enables variables.
  with_items:                                                    # What is being edited and how:
      # Naming the node
    - {dest: '/etc/elasticsearch/elasticsearch.yml', regexp: '#node.name: node-1', line: 'node.name: node-1'}
      # Allowing outside access
    - {dest: '/etc/elasticsearch/elasticsearch.yml', regexp: '#network.host: 192.168.0.1', line: 'network.host: 0.0.0.0'}
      # Setting the masternode. Note about regexp syntax: before square brackets, enter a \.
    - {dest: '/etc/elasticsearch/elasticsearch.yml', regexp: '#discovery.seed_hosts: \["host1", "host2"\]', line: 'discovery.seed_hosts: ["192.168.99.132"]'}
    - {dest: '/etc/elasticsearch/elasticsearch.yml', regexp: '#cluster.initial_master_nodes: \["node-1", "node-2"\]', line: 'cluster.initial_master_nodes: ["node-1

- name: Booting Elasticsearch
  become: yes
  service:
    name: elasticsearch
    state: started                                               # Start service
    enabled: yes                                                 # Enable service
```

Kuvio 9. Elasticsearchin asennus pelikirjalla

Rakennusjärjestyksessä seuraavana oli Windows-palvelin. Näin tosin toimitaan vain, mikäli käyttäjä on asettanut sen luotavaksi. Windows-palvelin ylläpitää virtuaaliym-

päristön toimialuetta, johon kaikki muut virtuaalikoneet lisätään, ja näin ollen se tulee käynnistää ennen muita. Mikäli järjestys olisi päinvastainen, muita virtuaalikoneita ei saataisi liitettyä automaattisesti toimialueeseen.

Koska Vagrant ei pysty sellaisenaan käyttämään SSH-yhteyttä Windows-koneisiin yhdistämiseen, vaihdettiin kommunikaattori niiden kohdalla WinRM:ään. Tämän lisäksi Vagrantille tuli antaa oikeat portit, jotta yhteyden muodostaminen onnistuisi. Windows-palvelimelle tehtiin vielä uniikki toimenpide, jossa WinRM:n todennustasoa laskettiin käyttämällä yksinkertaisempaa autentikaatiota. Jos näin ei olisi toimittu, rakennus olisi kaatunut toimialueen metsän luomiseen, sillä se olisi häirinnyt WinRM-yhteyttä. Kuviossa kymmenen näkyvät Windows-palvelimelle Vagrantfile-tiedostoon tehdyt määritykset.

```
## Windows AD
if WinAD > 0
        config.vm.define "win-ad" do |wa|
                wa.vm.synced_folder ".", "/vagrant", disabled: true
                wa.winrm.transport = :plaintext
                wa.winrm.basic_auth_only = true
                wa.vm.box = WinADBOX
                wa.vm.network :forwarded_port, guest: 3389, host: 3389, id: "rdp", auto_correct:true
                wa.vm.communicator = "winrm"
                wa.vm.guest = :windows
                wa.vm.network :forwarded_port, guest: 5985, host: 5985, id: "winrm", auto_correct:true
                wa.vm.provider "virtualbox" do |cfg|
                        cfg.name = "Windows AD"
                        cfg.memory = WinADRAM
                end
                wa.vm.synced_folder ".", "/vagrant", disabled: true
                wa.vm.network "private_network", ip: "192.168.99.131",
                        virtualbox__intnet: "Danger network"
                wa.vm.provision "shell",
                        path: "vagrant_provisioning/WINDOWS-VICTIM1.ps1"
                wa.vm.provision "shell",
                        path: "vagrant_provisioning/WINDOWS-AD1.ps1"
                wa.vm.provision :reload
                wa.vm.provision "shell",
                        path: "vagrant_provisioning/WINDOWS-AD2.ps1"
        end

        puts "Windows AD enabled. \n"
else
        puts "No Windows AD. \n"
end
## End of Windows AD
```

Kuvio 10. Vagrantfile Windows-palvelimen rakennus

Kuten aikaisemmin todettiin, Ansible ei toimi Windows-käyttöjärjestelmällä. Tämän vuoksi Vagrantilla muonitukseen käytettiin Powershell-ohjelmaa. Ensimmäiseksi ajettiin WINDOWS-VICTIM-1.ps1, jolla asennettiin Beatsit tiedonkeruuta varten. Kuviossa 11 näkyvät ajetut Powershell-komennot. Näiden avulla asennettiin AD-työkalut ja saatiin luotua Windows-toimialue ja metsä. Erityisen tärkeä osa komentoa on NoRebootOnCompletion, joka estää tietokoneen automaattisen sammumisen asennuksen jälkeen. Sammuminen aiheuttaisi häiriön Vagrantissa ja keskeyttäisi koko prosessin. Toisaalta palvelimen päivitys kuitenkin vaatii uudelleenkäynnistyksen. Tähän käytettiin Vagrantin reload-muonittajaa, joka käynnistää virtuaalikoneen uudestaan Vagrantin hallitsemana.

```
set-executionpolicy -executionpolicy unrestricted -force

# Set website security protocols
[Net.ServicePointManager]::SecurityProtocol = "Tls, Tls11, Tls12, Ssl3"

# Set the keyboard language to finnish
Set-WinUserLanguageList -languagelist en-FI -force -Confirm:$False

# Download & Install Wireshark silently
Invoke-WebRequest https://1.eu.dl.wireshark.org/win64/Wireshark-win64-3.4.0.exe -OutFile C:\
Wireshark.exe
C:\Wireshark.exe /S

# Install AD DC and mgmt tools
Add-WindowsFeature AD-Domain-Services -IncludeManagementTools -Confirm:$False

# Setup the AD forest with a safemode password, and stop the automatic reboot
Install-ADDSForest -DomainName dom.local -InstallDNS -safemodeadministratorpassword (
ConvertTo-SecureString -AsPlainText -String salaSana4444 -Force) -NoRebootOnCompletion -Confirm:
$False
```

Kuvio 11. Toimialueen asennus Powershell-skriptillä

Kun palvelimet olivat rakennettu, siirryttiin seuraavaksi uhrien rakentamiseen. Uhrien ominaisuutena on sekä Windowsin että Linuxin puolella skaalautuvuus. Skaalautuvuus on toteutettu for-luupilla. Skaalautuvuuden avulla virtuaalikoneita voidaan käynnistää useita kappaleita samanaikaisesti, ja samalla niiden IP-osoitteet ja nimet päivittyvät automaattisesti. Kuviossa 12 on Linux-uhrin osuus Vagrantfilestä.

```
## Linux Victim
## Attempting to boot the victim without SIEM will cause the setup to crash, because Elasticsearch cann
if LinVic > 0
        (1..LinVic).each do |i|
                config.vm.define "linux-victim-#{i}" do |lv|
                        lv.vm.box = LinVicBOX
                        lv.vm.synced_folder ".", "/vagrant", disabled: true
                        lv.vm.hostname = "linux-victim-#{i}"
                        lv.vm.provider "virtualbox" do |cfg|
                                cfg.name = "LINUX VICTIM #{i}"
                                cfg.memory = LinVicRAM
                        end
                        lv.vm.network "private_network", ip: "192.168.99.#{132+i}",
                                virtualbox__intnet: "Danger network"
                        lv.vm.provision "file", source: "./vagrant_provisioning", destination: "$HOME/"
                        lv.vm.provision :ansible_local do |ansible|
                                ansible.provisioning_path = "/home/vagrant"
                                ansible.playbook = "vagrant_provisioning/LINUX-VICTIM.yml"
                        end
                if WinAD > 0
                        lv.vm.provision :ansible_local do |ansible|
                                ansible.provisioning_path = "/home/vagrant"
                                ansible.playbook = "vagrant_provisioning/LINUX-JOIN.yml"
                        end
                end
        end

        if LinVic = 1
                puts "1 Linux Victim enabled \n"
        else
                puts "#{LinVic} Linux Victims enabled \n"
        end

end
else
        puts "No Linux Victims. \n"
end
## End of Linux Victim
```

Kuvio 12. Vagrantfile Linux-uhrin rakennus

Vagrantfileen tehtävä konfiguraatio Linux-uhrille on pitkälti samanlainen kuin palvelimen, ja erot löytyvätkin lähinnä muonitukseen käytettävistä pelikirjoista. Uhrille asennettiin Auditbeat, jolla lähetetään turvallisuustiedot SIEMille. Pelikirjan moduulit, jolla tämä tehdään, näkyvät kuviossa 13. Kun Auditbeat on ladattu ja purettu paketista, kirjoitetaan sen konfiguraatiotiedostoihin Kibanan ja Elasticsearchin IP-osoitteet ja portit, minkä jälkeen palvelu käynnistetään.

```
### AUDITBEAT

- name: Download Auditbeat 7.9        #.deb can be installed with https://docs.ansible.com/ansible/2.3/apt_module.html
  become: yes
  get_url:
    url: https://artifacts.elastic.co/downloads/beats/auditbeat/auditbeat-7.9.0-amd64.deb
    dest: /etc/

- name: Extract Auditbeat
  become: yes
  command: sudo dpkg -i auditbeat-7.9.0-amd64.deb
  args:
    chdir: /etc

- name: Configure Auditbeat
  become: yes
  lineinfile:                                                    # lineinfile-module is used to edit a
    dest: "{{ item.dest }}"                                      # The file to be edited.
    regexp: "{{ item.regexp }}"                                  # The line to be replaced.
    line: "{{ item.line }}"                                      # What we are replacing the line with
    backrefs: yes                                                # Enables variables.
  with_items:                                                    # What is being edited and how:
      # The Kibana to connect to
    - {dest: '/etc/auditbeat/auditbeat.yml', regexp: '#host: "localhost:5601"', line: '  host: "192.168.99.132:5601"'}
      # The Elasticsearch to connect to
    - {dest: '/etc/auditbeat/auditbeat.yml', regexp: "localhost:9200", line: "  hosts: 192.168.99.132:9200"}

- name: Start Auditbeat
  become: yes
  service:
    name: auditbeat
    state: started                                               # Start service
    enabled: yes                                                 # Enable service

- name: Setup Auditbeat
  become: yes
  command: auditbeat setup                                       # Module for running a command
```

Kuvio 13. Auditbeatin asennus pelikirjalla

Kun LINUX-VICTIM-pelikirja on ajettu, Vagrant tarkistaa onko Windows-palvelin valittu käynnistettäväksi. Mikäli se on valittu käynnistettäväksi, niin ajetaan vielä ylimääräinen pelikirja, jolla Linux-uhri liitetään Windows-palvelimen ylläpitämään toimialueeseen. Kuviossa 14 näkyvät LINUX_JOIN-pelikirjassa olevat Ansiblen moduulit, jotka tekevät Linux-uhrille tarvittavat muutokset toimialueeseen liittymistä varten. Ensimmäinen moduuli kartoittaa Windows-palvelimen ylläpitämän toimialueen isäntänimen sen IP-osoitteeseen, jotta Linux-uhri saa siihen yhteyden käyttämällä kyseistä nimeä. Tämän jälkeen Linuxin sources-tiedostoon lisätään tarvittavien pakettien lataamiseen vaadittavat arkistot. Viimeinen kuviossa näkyvä moduuli liittää virtuaalikoneen toimialueeseen.

```
- name: Edit the hosts-file                                          # Adds the domains address to hosts-file
  become: yes
  lineinfile:
    path: /etc/hosts
    state: present
    insertbefore: "# The following lines are desirable for IPv6 capable hosts"
    line: "192.168.99.131 dom.local"

- name: Edit sources                                                 # Deletes the word "last" from the sources-file
  become: yes
  lineinfile:
    path: /etc/apt/sources.list
    regexp: '^last'
    state: absent

- name: Edit sources 2                                               # Add new sources
  become: yes
  lineinfile:
    insertbefore: '^#last'
    dest: "{{ item.dest }}"
    line: "{{ item.line }}"
     backrefs: yes
  with_items:
    - {dest: '/etc/apt/sources.list', line: 'deb http://us.archive.ubuntu.com/ubuntu/ bionic universe'}
    - {dest: '/etc/apt/sources.list', line: 'deb http://us.archive.ubuntu.com/ubuntu/ bionic-updates universe'}


- name: Install packages                                             # Install new packages needed to connect to domain
  become: yes
  apt:
    state: present
    name:
      - realmd
      - libnss-sss
      - sssd
      - sssd-tools
      - adcli
      - samba-common-bin
      - oddjob
      - oddjob-mkhomedir
      - packagekit
      - python-pexpect            # this is for expect python module, use below
    update_cache: true


- name: Join the domain                                              # join the domain with given credentials
  become: yes
  expect:
    command: realm join -U vagrant dom.local
    responses:
      Password for *: vagrant
```

Kuvio 14. Linux-uhrin liittäminen toimialueeseen

Viimeinen käynnistettävä virtuaalikone on Windows-uhri. Sille tehtiin samanlainen WinRM-konfiguraatio kuin aikaisemmalle Windows-palvelimelle, ja samanlainen skaalautuvuus kuin Linux-uhrille. Windows-uhrin osuus Vagrantfilestä näkyy kuviossa 15.

```
## Windows Victim
## Attempting to boot the victim without SIEM will cause the setup to crash, because Elasticsearch cannot be found.
if WinVic > 0
        (1..WinVic).each do |j|
                config.vm.define "windows-victim-#{j}" do |wv|
                        wv.vm.synced_folder ".", "/vagrant", disabled: true
                        wv.vm.box = WinVicBOX
                        wv.vm.network :forwarded_port, guest: 3389, host: 3389, id: "rdp", auto_correct:true    # T
                        wv.vm.communicator = :winrm
                        wv.vm.guest = :windows
                        wv.vm.network :forwarded_port, guest: 5985, host: 5985, id: "winrm", auto_correct:true  # T
                        wv.winrm.retry_limit = 30
                        wv.winrm.retry_delay = 20
                        wv.vm.boot_timeout = 600
                        wv.vm.synced_folder ".", "/vagrant", disabled: true                                    # D
                        wv.vm.provider "virtualbox" do |cfg|
                                cfg.name = "WINDOWS VICTIM #{j}"
                                cfg.memory = WinVicRAM
                        end
                        wv.vm.network "private_network", ip: "192.168.99.#{132+LinVic+j}",              # Add a new
                                virtualbox__intnet: "Danger network"
                        wv.vm.provision "shell",
                          path: "vagrant_provisioning/WINDOWS-VICTIM1.ps1"
                if WinAD > 0
                        wv.vm.provision "shell",
                          path: "vagrant_provisioning/WINDOWS-VICTIM2.ps1"
                        wv.vm.provision :reload
                end
        end

        if WinVic = 1
                puts "1 Windows Victim enabled. \n"
        else
                puts "#{WinVic} Windows Victims enabled. "
        end

end
else
        puts "No Windows Victims. \n"
end
## End of Windows Victim
```

Kuvio 15. Vagrantfile Windows-uhrin rakennus

Windows-uhrille asennettiin Auditbeat kuten Linux-uhrillekin, ja lisäksi Winlogbeat, jonka asennus näkyy kuviossa 16. Winlogbeat- ja Auditbeat-palveluiden asentamisen vaiheet ovat samanlaiset Windows- ja Linux-uhreilla, mutta Windows-koneilla käytetään pelikirjan sijasta Powershell-skriptiä.

```
## Winlogbeat

# Download winlogbeat.zip
Invoke-WebRequest https://artifacts.elastic.co/downloads/beats/winlogbeat/winlogbeat-7.9.0-windows-x86_64.zip -OutFile C:\Winlogzip.zip

# Extract zip
expand-archive -path 'C:\Winlogzip.zip' -destinationpath 'C:\'

# Rename folder
Rename-Item -Path C:\winlogbeat-7.9.0-windows-x86_64 -NewName C:\Winlogbeat

# Configure Kibana and Elastic addresses
((Get-Content -path C:\Winlogbeat\winlogbeat.yml -Raw) -replace 'localhost:9200','192.168.99.132:9200') | Set-Content -Path C:\Winlogbeat\winlogbeat.yml
((Get-Content -path C:\Winlogbeat\winlogbeat.yml -Raw) -replace '#host: "localhost:5601"','host: "192.168.99.132:5601"') | Set-Content -Path C:\Winlogbeat\winlogbeat

# Service installation script
C:\Winlogbeat\install-service-winlogbeat.ps1

# Start Winlogbeat
start-service winlogbeat
```

Kuvio 16. Winlogbeatin asennus Powershellin avulla

Kun Windows-uhria liitetään toimialueeseen, tulee vaihtaa toisen verkkoadapterin DNS-osoitteeksi Windows-palvelimen käyttämä osoite. Tätä varten tulee selvittää sovittimen numero, joka saattaa myös olla vaihtuva. Kyseinen numero selvitetään kuvion 17 ensimmäisellä komennolla, joka tekee numerosta muuttujan, jota käytetään seuraavassa komennossa DNS-osoitteen muuttamiseksi. Tämän jälkeen luodaan pääsytiedot, jotka koostuvat nimestä ja salasanasta. Näitä pääsytietoja käytetään toimialueeseen liittymiseen.

```
### Windows 10 Domain configuration

## Joining

# Finds the Interface Index of the Ethernet 2 -adapter, and sets it as a variable
# If you have some other adapters than Ethernet, edit the Where-object -part
$IntIndex = $(Get-NetAdapter | Where-object {$_.Name -like "E*2" } | Select-Object -ExpandProperty InterfaceIndex)

# Sets the AD DC address as DNS for the aforementioned adapter
set-dnsclientserveraddress -interfaceindex $IntIndex -serveraddresses ("192.168.99.131")

# Create a login credential for the domain, you can also use your own
$domain = "dom.local"
$password = "vagrant" | ConvertTo-SecureString -asPlainText -Force
$username = "$domain\vagrant"
$credential = New-Object System.Management.Automation.PSCredential($username,$password)

# Join the domain
Add-Computer -DomainName $domain -Credential $credential
```

Kuvio 17. Windows-uhrin liittäminen toimialueeseen

4.4 Ympäristön eristys

Kun Vagrant on rakentanut virtuaaliympäristön, tulee se vielä eristää isäntäkoneesta. Jokaisella virtuaalikoneella on kaksi verkkokorttia, joista ensimmäinen on Vagrantin käyttämä. Tämän vuoksi kyseistä korttia ei voi poistaa Vagrantilla, vaan on käytettävä apuna Virtualboxia. Jotta tämä saataisiin automatisoitua, käytettiin Virtualboxin komentorivityökalua VBoxManage-skriptin tekemiseen (ks. kuvio 18). Näin saadaan irrotettua jokaisen käynnistetyn virtuaalikoneen ensimmäinen verkkokortti, jolloin virtuaalinen verkko ja isäntäkoneen verkko toimivat erillisinä toisistaan.

```
$siemCount = vboxmanage list vms | find """SIEM""" /c
$winACount = vboxmanage list vms | find """Windows AD""" /c
$linVCount = vboxmanage list vms | find """LINUX VICTIM """ /c
$winVCount = vboxmanage list vms | find """WINDOWS VICTIM""" /c

if ($siemCount -gt 0)
{
VBoxManage modifyvm "SIEM" --cableconnected1 off
vboxmanage controlvm "SIEM" nic1 null
}
elseif ($winACount -gt 0)
{
VBoxManage modifyvm "Windows AD" --cableconnected1 off
vboxmanage controlvm "Windows AD" nic1 null
}
elseif ($linVCount -gt 0)
{
1..$linVCount | % { VBoxManage modifyvm "LINUX VICTIM $_" --cableconnected1 off}
1..$linVCount | % { vboxmanage controlvm "LINUX VICTIM $_" nic1 null}
}
elseif ($winVCount -gt 0)
{
1..$winVCount | % { VBoxManage modifyvm "WINDOWS VICTIM $_" --cableconnected1 off}
1..$winVCount | % { vboxmanage controlvm "WINDOWS VICTIM $_" nic1 null}
}
else
{
echo "No machines detected!"
}
```

Kuvio 18. Ympäristön eristys Powershellillä

Kun eristysskripti on ajettu, virtuaaliympäristöllä ei siis enää ole verkkoyhteyttä isäntäkoneeseen eikä internettiin. Tämän varmistamiseksi virtuaaliympäristön sisältä yritettiin ottaa yhteyttä molempiin. Kuten kuviossa 19 näkyy, verkkoyhteydet katkaistiin onnistuneesti.

```
vagrant@linux-victim-1:~$ ping 192.168.0.100
connect: Network is unreachable
vagrant@linux-victim-1:~$ ping 8.8.8.8
connect: Network is unreachable
```

Kuvio 19. Epäonnistuneet pingit

Virtuaaliympäristön eristyksen toinen osa oli Virtualboxin jaettujen kansioiden poistaminen. Tämä tehtiin määrityksillä Vagrantfilessä, ja voidaan tarkistaa käyttämällä Virtualboxin komentorivityökalua. Kuten kuviossa 20 näkyy, jokaisen virtuaalikoneen jaetut kansiot ovat onnistuneesti poistettu.

```
D:\simulation-environment-master>vboxmanage showvminfo "Windows AD" | find "Shared"
Shared folders:<none>

D:\simulation-environment-master>vboxmanage showvminfo "LINUX VICTIM 1" | find "Shared"
Shared folders:<none>

D:\simulation-environment-master>vboxmanage showvminfo "WINDOWS VICTIM 1" | find "Shared"
Shared folders:<none>

D:\simulation-environment-master>vboxmanage showvminfo SIEM | find "Shared"
Shared folders:<none>
```

Kuvio 20. Jaettujen kansioiden poiston varmistus

Haittaohjelmat pystyvät hyödyntämään sekä verkkoyhteyksiä että jaettuja kansioita itsensä levittämistä varten, ja siksi näitä varten tuli tehdä varotoimia. Eristys ei valitettavasti ole kuitenkaan täysin aukoton. Mikäli haittaohjelmat ovat tarpeeksi kehittyneitä, voivat ne ennen muuta toimintaansa tutkia löytyykö tiedostoista virtualisoinnin jättämiä jälkiä tai muuta virtualisointiin viittaavaa.

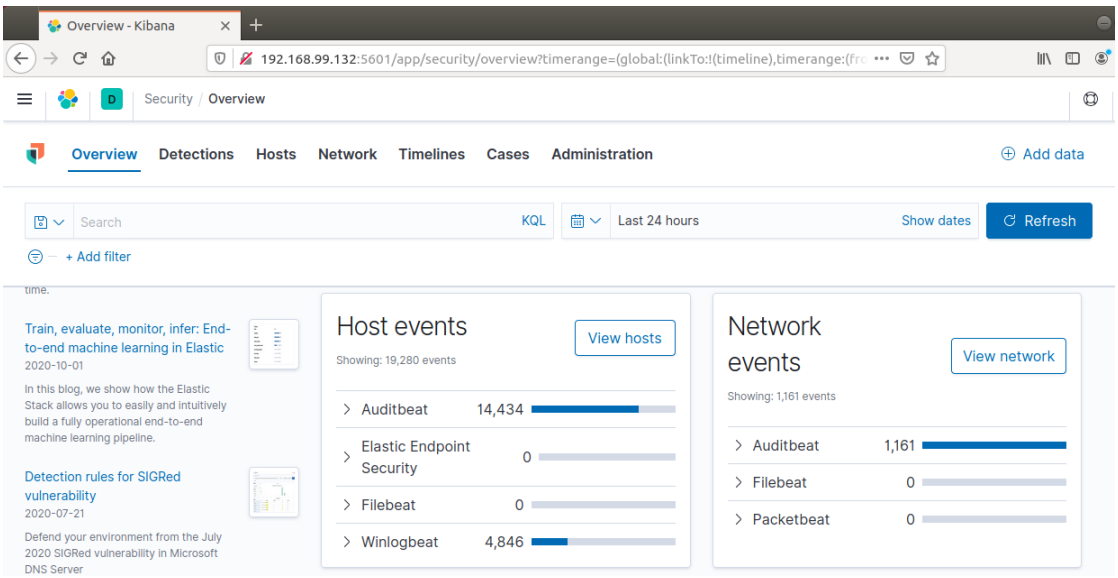
Mikäli haittaohjelma huomaa toimivansa virtuaaliympäristössä, se saattaa hyväksikäyttää virtualisointialustan arkkitehtuuria, ja sitä kautta päästä isäntäkoneeseen käsiksi. Nämä hyökkäykset kohdistuvat virtualisointialustojen elintärkeisiin turvallisuusominaisuuksiin, kuten virtuaalikoneiden eristykseen, virtuaalikoneiden ohjelmistopohjaisiin kommunikaatiokanaviin tai muihin hallintarajapintoihin. (Security aspects of Virtualization 2017, 34–37.)

Isäntäkoneeseen tunkeutumisen lisäksi haittaohjelmat saattavat myös virtualisoinnin havaittuaan lopettaa toimintansa. Tämä tarkoittaa tällöin, että virus ei tuota lainkaan hyökkäystä eikä näin ollen myöskään tuota minkäänlaisia jälkiä sellaisesta. Tämän ns. teeskentelyn vuoksi haittaohjelma voi vaikuttaa tavalliselta ohjelmalta ja päästä näin ollen virtuaaliympäristöstä ulos ja sitä kautta oikeaan ympäristöön ihmisen toimesta.

Vaikka edellä kuvatun kaltaiset hyökkäykset ovat ehkä harvinaisia, ovat ne kuitenkin huomioon otettava uhka tietoturvatestausta suunnitellessa ja tehdessä. Tämän vuoksi käyttäjän on tärkeää ymmärtää niin sanotun sokean testaamisen riskit ja suorittaa tarvittavat varotoimet tietoisesti riskialttiita testejä tehdessään. Näihin varotoimiin voi esimerkiksi kuulua testattavan haittaohjelman tietoihin tutustuminen ja asiaankuuluvien virtualisointialustan heikkouksien paikkaaminen.

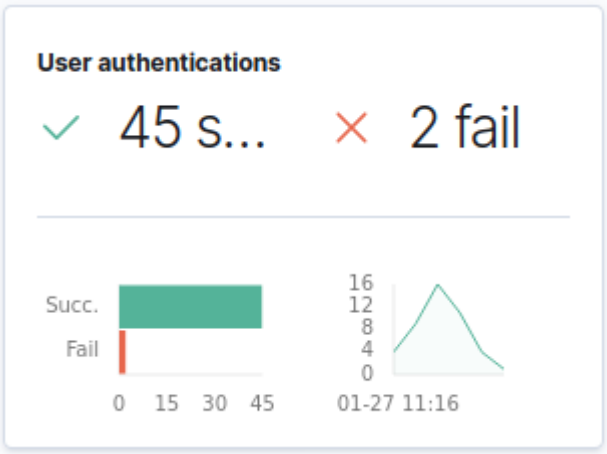
4.5 Luodut palvelut

Virtuaaliympäristön eristämisen jälkeen rakennettujen palvelimien palveluihin pääsee käsiksi ainoastaan sen sisältä käyttämällä sisäverkossa olevia IP-osoitteita. Käyttäjän näkökulmasta palveluista keskeisin on Kibana, joka näkyy kuviossa 21.



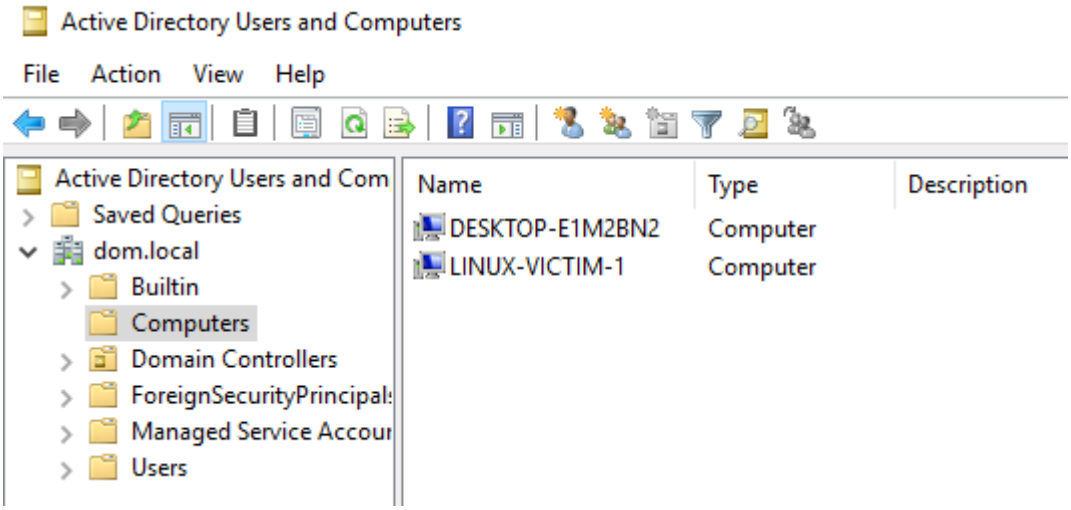
Kuvio 21. Kibanan SIEMin etusivu

Kibanan kautta päästään tarkastelemaan Beatsien keräämää tietoa, esimerkiksi ympäristössä tapahtuvia sisäänkirjautumisia. Kuten kuviossa 22 on nähtävissä, sekä onnistuneet että epäonnistuneet kirjautumiset kirjautuvat muistiin. Näin ylläpitäjät voivat huomata virtuaaliympäristössä tapahtuneet luvattomat kirjautumisyritykset, joita kyberrikolliset voivat yrittää tehdä. Tämä on yksi esimerkki SIEMin mahdollistamista monipuolisista valvontamahdollisuuksista.



Kuvio 22. Käyttäjien kirjautumiset ympäristössä

Toinen testauksen monipuolisuuden kannalta oleellinen osa virtuaaliympäristöä on Windows-toimialue, johon kaikki uhrikoneet liitetään. Kuviossa 23 varmistetaan, että Windows-palvelin on tietoinen uhrikoneista.



Kuvio 23. Toimialueeseen liitetyt tietokoneet

4.6 Ohjelmiston lisääminen

Jos käyttäjä haluaa lisätä ympäristön virtuaalikoneille ohjelmistoa, jota niille ei oletuksena asenneta automaattisesti, voi käyttäjä rakentaa ympäristön ja asentaa ohjelmat manuaalisesti verkon kautta ennen ympäristön eristystä. Jotta tätä ei tarvitse tehdä jokaisella rakennuskerralla erikseen, tulee käyttäjien kirjoittaa omat Powershell-skriptit tai Ansible-pelikirjat käyttöjärjestelmästä riippuen. Kuviossa 24 näkyy yksinkertainen pelikirja, jolla asennetaan ja käynnistetään Uncomplicated Firewall (UFW).

```
---
-
  connection: local
  gather_facts: false
  hosts: all
  tasks:

  - name: Install Firewall
    become: yes
    apt:
      state: present
      name: ufw

  - name: Enable Firewall
    become: yes
    command: ufw enable
```

Kuvio 24. UFW:n asennus pelikirjalla

Kun mukautetut tiedostot ovat valmiit, viedään ne vagrant_provisioning-kansioon, josta Linux-virtuaalikoneille viedään tiedostot Ansiblea varten. Windows-koneilla tämä ei ole pakollista, sillä niiden kohdalla voidaan käyttää myös suoraa tiedostopolkua. Luodut tiedostot ovat tästä huolimatta kannattavaa sijoittaa vagrant_provisioning-kansioon, jotta ne saadaan pidettyä samassa järjestyksessä muiden tiedostojen kanssa.

Kun uudet tiedostot ovat oikeassa kansiossa, tulee seuraavaksi lisätä Vagrantfileen halutun tietokoneen kohdalle uusi muonittaja-osio, jossa tiedostoa kutsutaan. Yksinkertaisin tapa tehdä tämä on kopioida aiempi muonittaja saman roolin asetuksista, jossa toinen konfiguraatiotiedosto on määritelty, ja vaihtaa sen nimi. Kuviossa 25 on nähtävillä lisätty FIREWALL-pelikirja, jota kutsutaan samalla tavalla kuin SIEM-pelikirjaa. Uusien muonittajien lisäämisen periaate on sama käyttöjärjestelmästä ja roolista riippumatta.

```
si.vm.provision :ansible_local do |ansible|
        ansible.provisioning_path = "/home/vagrant"
        ansible.playbook = "vagrant_provisioning/SIEM.yml"
end
si.vm.provision :ansible_local do |ansible|
        ansible.provisioning_path = "/home/vagrant"
        ansible.playbook = "vagrant_provisioning/FIREWALL.yml"
end
```

Kuvio 25. SIEMin muonittajat

Kun uudet tiedostot ovat kirjoitettu, viety oikeaan kansioon ja lisätty Vagrantfileen, virtuaaliympäristöä rakentaessa uudet konfiguraatiot suoritetaan annetuille rooleille. Näin käyttäjä kykenee lisäämään uusia palveluita ja muita testaukseen tarvittavia ohjelmia automaattiseen rakennukseen.

5 Tutkimustulokset

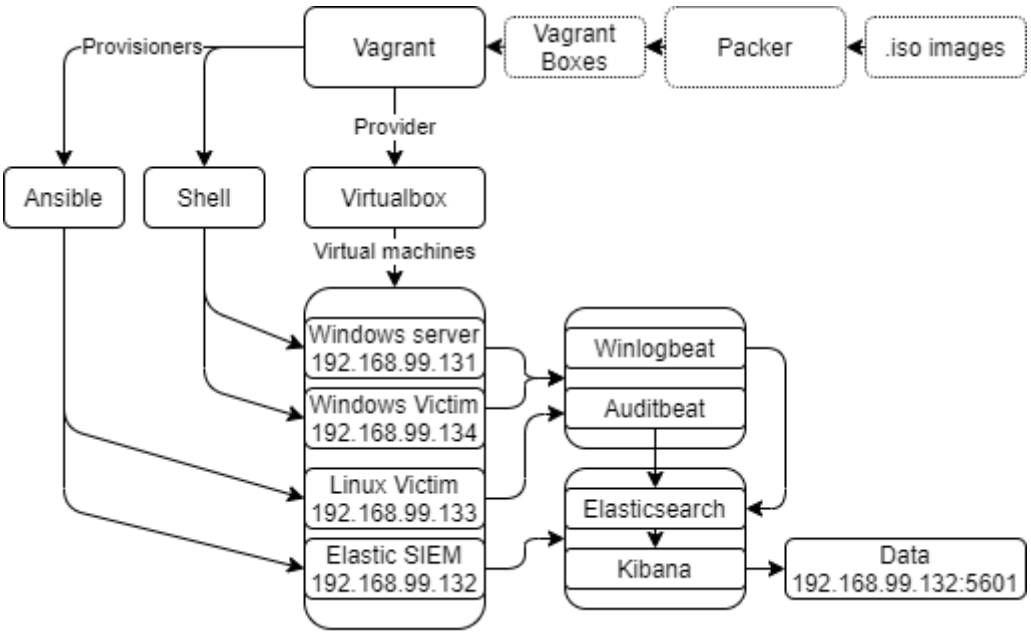
Tässä opinnäytetyössä oli tavoitteena automatisoida virtuaaliympäristön rakentaminen. Tätä virtuaaliympäristöä oli tarkoitus käyttää kyberturvallisuustestauksissa ja -koulutuksissa. Virtuaaliympäristöön tuli siksi asentaa työkalut haittaohjelmien tutkimista varten.

Tutkimuksessa onnistuttiin luomaan toimintamalli, joka vastaa edellä esiteltyihin tutkimuksessa asetettuihin tavoitteisiin. Toimintamallin luomiseen tarvittiin monien eri tuotteiden valintaa ja hyödyntämistä. Näiden tuotteiden versiot ja lisäksi käytetyn isäntäkoneen tiedot näkyvät kuviossa 26.

Isäntäkone RAM: 16 GB Käyttöjärjestelmä: Microsoft Windows 10 Pro
Palvelut Vagrant 2.2.9 Packer 1.6.5 Auditbeat 7.9 Winlogbeat 7.9 Elasticsearch 7.9 Kibana 7.9

Kuvio 26. Isäntäkoneen ja palveluiden tiedot

Hyödyntäen Packeria saatiin luotua Vagrant-laatikot käyttöjärjestelmien levykuvakkeista. Seuraavaksi Vagrant käytti näitä laatikoita virtuaaliympäristön rakennukseen Virtualboxille, joka toimi virtualisointialustana. Lopuksi virtuaalikoneille asennettiin Elastic-pinon palvelut käyttämällä Ansiblea ja Powershelliä. Näin saatiin automatisoitua virtuaaliympäristön rakennus levykuvakkeista asti. Tähän virtuaaliympäristöön asennettiin vaaditut työkalut, joilla pystyttiin tutkimaan erilaisten haittaohjelmien jättämiä jälkiä. Virtuaalikoneet eristettiin isäntäkoneen verkosta ja kansiojärjestelmistä, jotta virukset eivät pääsisi leviämään niiden kautta. Kuviossa 27 näkyvät käytetyt teknologiat, niiden järjestys ja yhteydet toisiinsa.



Kuvio 27. Vuokaavio

Tutkimuksen tavoitteena oli myös, että luotavaa toimintamallia voitaisiin käyttää opetuksen lisäksi myös kotikäytössä. Mahdollisen kotikäytön vuoksi virtuaaliympäristön tuli olla ilmainen ja keskusmuistin enimmäisvaatimus oli 16 GB. Tässä onnistuttiin valitsemalla riittävän kevyet palvelut ja ohjelmat, jotka olivat ilmaisia käyttää.

Tutkimuksen lopputulosta voidaan pitää luotettavana, sillä sekä eristyksen että palveluiden toiminnalle tehtiin onnistuneesti teknisiä testejä. Lisäksi, jotta työn tulokseen voisi luottaa, tulee toimintamallin lähdekoodi olemaan ilmaisessa jakelussa. Myös monet käytetyistä teknologioista pitävät lähdekoodiaan vapaassa jakelussa. Ainoa asia, jota ei voida pitää täysin luotettavana on virtuaaliympäristön eristys, sillä sitä on mahdotonta tehdä aivan täydelliseksi virtualisointialustan omien heikkouksien vuoksi.

Toimintamallin luomiseksi hyödynnettiin käytettyjen teknologioiden dokumentaatioita, joista saatiin tarvittavat tiedot teknologioiden käyttämiseksi ja mahdollisten ongelmien ratkaisemiseksi. Esimerkiksi Ansiblen moduuleita ja Vagrantin osia tutkittiin

tarvittaessa niiden virallisista dokumentaatioista. Osittain opinnäytetyössä sovellettiin myös kolmansien osapuolten julkaisemia prosesseja, joista jalostettiin opinnäytetyön tarkoitukseen soveltuvia ratkaisuja.

Tarkasteltaessa ilmaisia ratkaisuja vastaavanlaiseen tarpeeseen voidaan nähdä, että on jo olemassa erilaisia toteutuksia. Esimerkiksi Cuckoo Sandbox tarjoaa mahdollisuuden tutkia yksittäisten tiedostojen toimintaa eristetyssä ympäristössä (Cuckoo Automated Malware Analysis n.d.). Tähän palveluun ei kuitenkaan sisälly toimialue tai rakennus levykuvakkeesta asti, ja siksi ei vastaa tässä opinnäytetyössä toimintamallille asetettuihin tavoitteisiin ja tarpeisiin. Tämän vuoksi näin ollen voitaneen siis todeta, että tutkimuksessa luotu toimintamalli tuo toimeksiantajalle arvoa.

6 Yhteenveto

Opinnäytetyössä tutkittiin, kuinka saadaan automatisoitua virtuaalisen ympäristön rakennus tietoturvauhkien testausta varten. Opinnäytetyön tulos oli kehittää toimintamalli, jonka rakentamiseen käytettiin Virtualbox-, Packer-, Vagrant-, Ansible-, ja ELK-pino – työkaluja. Jokainen edellä mainittu teknologia suoritti rakennuksessa oman tehtävänsä. Prosessin lopputuotteena oli verkosta eristetty virtuaaliympäristö, joka sisälsi tarvittavat käyttövalmiit palvelut. Tutkimuksessa havaittiin, että kyseisiä tuotteita käyttämällä saatiin automatisoitua virtuaalisen ympäristön rakennus alusta loppuun, samalla asentaen palvelut tietoturvatestausta varten.

Tutkimusmenetelmäksi opinnäytetyössä valittiin konstruktiivinen metodi. Metodin voidaan katsoa soveltuneen tutkimukseen erityisen hyvin, koska työssä hyödynnettiin olemassa olevia lähteitä, palveluja ja muita ilmaiseksi saatavilla olevia materiaaleja. Käytetty metodi säästi myös tutkijan aikaa ja työtä: koska toimintamallin teettämisessä käytettiin hyödyksi edellä mainittuja valmiita malleja ja palveluja, saatiin osia tutkimuksesta tehtyä nopeammalla aikataululla.

Konstruktiivisen metodin mukaisesti tutkimuksessa luotiin aiempaan tietämykseen ja teoriaan perustaen uutta: tässä tapauksessa uusi toimintamalli. Uuden toimintamallin aikaansaamisen voidaan katsoa olleen tutkimuksen suurin arvo sekä yleisen hyödyllisyyden että tutkimuksen toimeksiantajan näkökulmasta.

Konstruktiivisen metodin heikkouksiin kuuluivat käytännön lähdemateriaalien, kuten valmiiden toimintamallien ja muiden projektien mahdollinen epäluotettavuus ja soveltumattomuus, mikä vaati tutkijalta materiaalien toiminnan ja hyödyllisyyden testaamista lähteitä valittaessa. Tämä työvaihe kannattaakin ottaa huomioon projektia aikataulutettaessa, kuten myös tässä tutkimuksessa tehtiin. Toisaalta tutkimuksessa havaittiin, että luotettavia ja vertaisarvioituja teoriapohjaisia lähteitä löytyi helposti ja monipuolisesti useilta eri sivustoilta, joiden avulla saatiin kasvatettua teoriapohjaista tietoperää.

Toisena konstruktiivisen tutkimusotteen haasteena voitaneen pitää riittävän vuorovaikutuksen ylläpitämistä tutkijan ja toimeksiantajan välillä tutkimusprosessin aikana. Opinnäytetyön aikana osapuolet olisivat voineet käydä enemmän keskustelua työn etenemisestä ja erilaisista toteutustavoista. Tutkimuksen alussa käytiin kattava keskustelu siihen liittyvistä tarpeista, vaatimuksista ja teknologioista, jonka vuoksi tavoitteessa kuitenkin pysyttiin. Tutkimuksessa olisi mahdollisesti kaivattu kolmannen osapuolen mielipidettä ja testausta, ja siksi objektiivisuuden puute vaikuttaa lopputuotteeseen. Kun kolmas osapuoli käyttää toimintamallia, löytyy varmasti parantamisen kohteita, joita tutkimuksen aikana ei onnistuttu löytämään.

Tutkimuksen luotettavuuden ja läpinäkyvyyden lisäämiseksi tutkimusprosessi pyrittiin kuvaamaan tarkasti, jolloin se on toistettavissa. Lisäksi sen lähdekoodi tulee olemaan ilmaisessa jakelussa, jolloin asiasta kiinnostuneet voivat perehtyä siihen. Toimintamallin luotettavuutta paransi teknilliset testit, joita tehtiin eristykselle ja palveluille, jotka olivat sen tärkeimpiä ominaisuuksia.

Opinnäytetyön lopputulosta voidaan pitää hyvänä, koska sille asetetut tavoitteet saavutettiin ilman, että virtuaaliympäristöstä olisi tarvinnut leikata oleellisia ominaisuuk-

sia pois. Erityisesti Vagrant-osuudesta onnistuttiin luomaan hyvin käyttäjäystävällinen ja helposti ymmärrettävä. Testausta ja sen myötä löytyvien mahdollisten ongelmien ratkaisua olisi voinut tutkimuksessa lisätä, vaikkakin se toisaalta olisi hyvin aikaa vievää. Myös teknologioiden tehokkaan käytön tutkimiseen olisi voitu käyttää enemmän aikaa, koska muutoksia näiden tiedostoihin jouduttiin tekemään kesken tutkimuksen. Lisäksi Packerin tiedostoja olisi voinut tutkimuksessa mahdollisesti jalostaa hieman pidemmälle.

Toimintamallin jatkokehityksessä tulisi ottaa huomioon kolmannen osapuolen testauksesta tullut palaute ja toimia sen mukaisesti. Näin saataisiin käsiteltyä objektiivisuuden puutetta ja korjattua yleisiä käyttäjien kohtaamia haasteita. Lisäksi voitaisiin kehittää tukea useammalle eri käyttöjärjestelmälle etenkin Packer-työkalun kohdalla, vaikkakaan tämä ei ole aivan välttämätöntä. Tärkeä kehityskohde olisi pyrkiä parantamaan virtuaaliympäristön eristystä isäntäkoneesta vakavampien virtualisointialustoja hyväksikäyttävien haittaohjelmien tutkimista varten. Näiden haittaohjelmien hillitseminen vaatii huomattavan määrän lisätutkimusta aiheesta, minkä vuoksi se rajattiin tästä tutkimuksesta pois.

Työn konkreettinen hyöty oli toimeksiantajalle uusi konstruktio, jota pystyy hyödyntämään koulutuksessa ja muussa tietoturvauhkiin liittyvissä testaamisissa. Voitaneen todeta, että uusi toimintamalli parantaa omalta osaltaan yleistä tietoturvaosaamista ja luo toimijoille sekä työpaikalla että kotioloissa mahdollisuuksia kehittyä kyberturvallisuusalan osaajina. Opinnäytetyössä tutkija kehitti taitojaan sekä tuttujen että uusien modernien teknologioiden parissa, oppien näiden toimintaperiaatteista ja hyödyntämisestä käytännön kontekstissa.

Lähteet

Abueg, R. 2020. Artikkeli Elasticsearchista Knowin verkkosivuilla. Viitattu 2.2.2021. https://www.knowi.com/blog/what-is-elastic-search/.

Almeida, F., Santos, J. & Monteiro, J. 2020. The Challenges and Opportunities in the Digitalization of Companies in a Post-COVID-19 World. IEEE Engineering Management Review, vol. 48, no. 3, 97-103. Viitattu 11.2.20.21. https://ieeexplore.ieee.org/document/9153093.

Anastasov, I. & Davcev, D. 2014. SIEM implementation for global and distributed environments. Konferenssijulkaisu. Viitattu. 15.2.2021. https://ieeexplore.ieee.org/document/6916651.

Arief, B., Adzimi, M. & Gross, T. 2015. Understanding Cybercrime from Its Stakeholders' Perspectives: Part 1—Attackers. IEEE Security & Privacy, 13, 1, 71-76. Viitattu 11.2.2021. https://ieeexplore.ieee.org/document/7031833.

Berman, D. 2020. Blogipostaus Beats-palvelusta Logz-verkkosivuilla. Viitattu 4.2.2021. https://logz.io/blog/beats-tutorial/.

Cuckoo Automated Malware Analysis. N.d. Cuckoo-työkalun esittely Cuckoosandbox.org-verkkosivuilla. Viitattu 28.3.2021. https://cuckoosandbox.org/.

CYBERDI projektiesittely. N.d. CYBERDI-projektin esittely JAMKin verkkosivuilla. Viitattu 17.2.2021. https://www.jamk.fi/fi/Tutkimus-ja-kehitys/projektit/CYBERDI/Projektiesittely/.

CYBERDI. N.d. Artikkeli CYBERDistä Jyvsectecin verkkosivuilla. Viitattu 15.1.2021. https://jyvsectec.fi/2018/10/cyberdi/.

Hoffman, C. 2017. Artikkeli toimialueista Howtogeek-verkkosivuilla. Viitattu 23.3.2021. https://www.howtogeek.com/194069/what-is-a-windows-domain-and-how-does-it-affect-my-pc/.

How Ansible Works. N.d. Artikkeli Ansiblesta Ansiblen verkkosivuilla. Viitattu 29.1.2021. https://www.ansible.com/overview/how-ansible-works.

IETF/RFC 4251:2006. SSH-protokollan arkkitehtuuri. Standardi. Internet Engineering Task Force. Viitattu 18.3.2021. https://tools.ietf.org/html/rfc4251.

ISO/IEC 21778:2017. The JSON data interchange syntax. Standardi. International Organization for Standardization. Viitattu 27.3.2021. https://www.iso.org/standard/71616.html.

IT Automation. N.d. Artikkeli automaatiosta VMWaren verkkosivuilla. Viitattu 22.1.2021. https://www.vmware.com/topics/glossary/content/it-automation.

Kibana. N.d. Artikkeli Kibanasta TectTargetin sivuilla. Viitattu 2.2.2021. https://searchitoperations.techtarget.com/definition/Kibana.

Lakhera, P. 2019. Artikkeli Packerista Mediumin verkkosivuilla. Viitattu 29.1.2021. https://medium.com/@devopslearning/100-days-of-devops-day-27-introduction-to-packer-d77089ecac01.

Lukka, K. 2014. Konstruktiivinen tutkimusote. Viitattu 20.2.2021. https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/.

Murphy, A. N.d. VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS. Viitattu 19.1.2021. https://www.f5.com/pdf/white-papers/virtualization-defined-wp.pdf.

Murray, A. 2020. UFW:n esittely Ubuntun verkkosivuilla. Viitattu 10.2.2021. https://wiki.ubuntu.com/UncomplicatedFirewall.

Petters, J. 2020. Artikkeli Windows AD DS -palvelusta. Viitattu 24.3.2021. https://www.varonis.com/blog/active-directory-domain-services/.

Security aspects of Virtualization. 2017. Analyysi virtualisoinnin turvallisuudesta. Raportti. Viitattu 19.2.2021. https://www.enisa.europa.eu/publications/security-aspects-of-virtualization/.

SFS-ISO/IEC 27032:2012. Menettelyohjeet kyberturvallisuuteen. Aihealueet: Informaatioteknologia, turvallisuustekniikat. Helsinki: Suomen Standardisoimisliitto SFS. Vahvistettu 16.7.2012. Viitattu 10.2.2021. https://janet.finna.fi, SFS Online.

Shell script. N.d. Artikkeli Komentojonotiedostoista SearchDataCenterin verkkosivuilla. Viitattu 30.1.2021. https://searchdatacenter.techtarget.com/definition/shell-script.

Šimec, A., Držanić, B & Lozić, D. 2018. Isolated Environment Tools for Software Development. Konferenssijulkaisu. Viitattu 28.2.2021. https://ieeexplore.ieee.org/document/8955312.

Son, S. & Kwon, Y. 2017. Performance of ELK stack and commercial system in security log analysis. Konferenssijulkaisu. Viitattu 1.3.2021. https://ieeexplore.ieee.org/document/8311756.

Souppaya, M., Scarfone, K. & Hoffman, P. 2011. Guide to Security for Full Virtualization Technologies. Opas virtualisoinnin turvallisuuteen. Viitattu 27.2.2021. https://www.nist.gov/publications/guide-security-full-virtualization-technologies.

What is an Ansible playbook? N.d. Artikkeli Ansiblen pelikirjoista Red Hatin verkkosivuilla. Viitattu 30.1.2021. https://www.redhat.com/en/topics/automation/what-is-an-ansible-playbook.

What is DNS? N.d. Artikkeli DNS-protokollasta Paloaltonetworks-verkkosivuilla. Viitattu 25.3.2021. https://www.paloaltonetworks.com/cyberpedia/what-is-dns.

What is the ELK Stack? N.d. ELK-pinon esittely. Viitattu 5.2.2021. https://www.elastic.co/what-is/elk-stack.

What is Vagrant? N.d. Artikkeli Vagrantista Opensourcen verkkosivuilla. Viitattu 29.1.2021. https://opensource.com/resources/vagrant.

White, S., Coulter, D., Batchelor, D., Jacobs, M. & Satran, M. 2018. Windows Remote Management. Tekninen dokumentaatio. Viitattu 19.3.2021. https://docs.microsoft.com/en-us/windows/win32/winrm/portal.

VirtualBox. 2019. Artikkeli VirtualBoxista Computerhopen verkkosivuilla. Viitattu 7.2.2021. https://www.computerhope.com/jargon/v/virtualbox.htm.

Vojnak, D., Đorđević, B., Timčenko, V. & Štrbac, S. 2019. Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation. Tutkimus virtualisointialustojen Virtualbox ja VMWare Workstation eroista. Viitattu 19.3.2021. https://ieeexplore.ieee.org/document/8971213.