

**Penetration Testing -harjoitusympäristön toteutus va-
pailla ohjelmilla -
Kali Linuxiin sisältyvällä Metasploitilla tunkeutuminen
Metasploitable 2 -koneeseen Vagrant-virtuaaliympäris-
tössä**

Toni Jääskeläinen



Tekijä(t) Toni Jääskeläinen	
Koulutusohjelma Tietojenkäsittely	
Opinnäytetyön otsikko Penetration Testing -harjoitusympäristön toteutus vapailla ohjelmilla - Kali Linuxiin sisältyvällä Metasploitilla tunkeutuminen Metasploitable 2 - koneeseen Vagrant-virtuaaliympäristössä	Sivu- ja liitesivumäärä 46 + 1
<p>Nykyaikaisista järjestelmistä on tullut niin monimutkaisia, että kaikkia mahdollisia haavoittuvuuksia voi olla vaikea löytää. Monimutkaisuus käy ilmi jo pelkästään tarjolla olevista erityyppisistä puolustusmekanismeista. Vaikka niiden tarkoitus on puolustaa järjestelmäympäristöä ja turvata tärkeät tiedot, voi määrittelyihin tulla virheitä ja näin ympäristöön voi jäädä haavoittuvuuksia monista turvatoimenpiteistä huolimatta. Tunkeutumistestaajat toimivat laillisesti hyökkääjän roolissa selvittämällä olemassa olevia haavoittuvuuksia ja yrittämällä tunkeutua niihin. He raportoivat löydöksensä ja antavat suosituksia haavoittuvuuk-sien paikkaamista varten. Olisi tärkeää olla olemassa ympäristö, jossa voi harjoitella tunkeutumistestausta turvallisesti.</p> <p>Aluksi teoriaosiossa käytiin läpi, mitä Penetration Testing eli tunkeutumistestaus on ja mitä siihen kuuluvat seitsemän vaihetta pitävät sisällään. Ympäristön toteutusta varten kerrottiin ensin vapaista ohjelmista ja virtualisoinnista. Tämän jälkeen verrattiin ympäristön toteutukseen soveltuvia työkaluja keskenään. Seuraavaksi verrattiin, minkälaisilla työkaluilla tunkeutuminen kannattaa toteuttaa. Teoriaosion lopussa verrattiin, minkälaiseen kohteeseen harjoitusympäristössä kannattaa tunkeutua.</p> <p>Työn tehtävänä oli selvittää, kuinka Penetration Testing -harjoitusympäristö toteutetaan vapailla ohjelmilla. Tavoitteena oli saada aikaan ympäristö, jossa voi turvallisesti harjoitella tunkeutumistestausta. Ympäristön piti myös olla helposti saatavilla ja hyödynnettävissä. Työn kohderyhmänä olivat Penetration Testing -aloittelijat, jotka hallitsevat Linuxin ja komentorivin käytön. Työ rajattiin käsittämään vapaat ohjelmat. Työhön valittiin virtuaaliympäristön mahdollistavaksi kokonaisuudeksi Vagrant, joka käyttää VirtualBoxia providerina. Tunkeutumistyökaluksi valittiin Kali Linuxiin normaalisti sisältyvä Metasploit Framework ja kohteeksi tarkoituksella haavoittuvaksi tehty Metasploitable 2.</p> <p>Työ oli toiminallinen opinnäytetyö ja siihen liittyvät tiedostot tuotettiin GitHubiin, opinnäytetyön raportti mukaan lukien. Virtuaalikoneiden boxit tuotettiin Vagrant Cloudiin osoitteeseen https://app.vagrantup.com/tonijaaskelainen/. Työssä pystytettiin kannettavaan tietokoneeseen virtuaalinen ympäristö ja testattiin sen toiminta.</p> <p>Työ tehtiin aikavälillä joulukuu 2015 – marraskuu 2017. Suurin osa työn alkuosasta tehtiin syksyllä 2016 ja työn loppuosasta syksyllä 2017.</p> <p>Projektin tuloksena syntyi tunkeutumistestaukseen sopiva virtualisoitu harjoitusympäristö, johon liittyvän Vagrantfilen voi ladata GitHubista osoitteesta https://github.com/tonijaaskelainen/beginnerpentest/ tai kopioida työn liiteosion. Vagrant haki työssä luodut base boxit onnistuneesti Vagrant Cloudista.</p>	
Asiasanat tunkeutumistestaus, vapaat ohjelmat, virtualisointi, Metasploit Framework, Metasploitable 2	

Sisällys

1	Johdanto	1
1.1	Projektin tausta	1
1.2	Projektin tavoite ja tehtävä	1
1.3	Tutkimuskysymykset	1
1.4	Kohderyhmä ja rajaus	2
1.5	Menetelmät ja työn rakenne	2
1.6	Käsitteet	2
2	Penetration Testing eli tunkeutumistestaus	3
2.1	Tunkeutumistestauksen merkitys ja hyödyt	3
2.2	Tunkeutumistestauksen standardi	4
2.2.1	Pre-engagement Interactions	5
2.2.2	Intelligence Gathering	6
2.2.3	Threat Modeling	7
2.2.4	Vulnerability Analysis	8
2.2.5	Exploitation	9
2.2.6	Post Exploitation	9
2.2.7	Reporting	10
3	Välineet ympäristön toteutukseen	10
3.1	Vapaat ohjelmat	10
3.2	Virtualisointi	12
3.2.1	VirtualBox	13
3.2.2	Vagrant	14
3.2.3	Packer	16
4	Välineet ympäristöön tunkeutumiseen	16
4.1	Kali Linux	17
4.1.1	Metasploit Framework	18
4.1.2	Burp Suite	19
5	Tunkeutumiskohteet	20
5.1	Metasploitable	20
5.2	Muut peruspalvelimet	21
5.2.1	Windows-palvelin	21
5.2.2	Linux-palvelin	22
6	Penetration Testing -harjoitusympäristön toteutus	22
6.1	Virtuaaliympäristön pystytys	23
6.1.1	Lataaminen	23
6.1.2	Asentaminen	26
6.1.3	Määrittely	27

6.2 Tunkeutumisen testaaminen	38
7 Pohdinta.....	39
Lähteet	40
Liitteet.....	47
Liite 1. Vagrantfile	47

1 Johdanto

Nykyaikaisista järjestelmistä on tullut niin monimutkaisia, että kaikkia mahdollisia haavoittuvuuksia voi olla vaikea löytää. Monimutkaisuus käy ilmi jo pelkästään tarjolla olevista erityyppisistä puolustusmekanismeista. Vaikka niiden tarkoitus on puolustaa järjestelmäympäristöä ja turvata tärkeät tiedot, voi määrittelyihin tulla virheitä ja näin ympäristöön voi jäädä haavoittuvuuksia monista turvatoimenpiteistä huolimatta. Tunkeutumistestaaajat toimivat laillisesti hyökkääjän roolissa selvittämällä olemassa olevia haavoittuvuuksia ja yrittämällä tunkeutua niihin. He raportoivat löydöksensä ja antavat suosituksia haavoittuvuuk-sien paikkaamista varten. Olisi tärkeää olla olemassa ympäristö, jossa voi harjoitella tunkeutumistestausta turvallisesti.

1.1 Projektin tausta

Projekti käynnistettiin opiskelijan opinnäytetyönä, koska opiskelijan mielestä aihealue on kiinnostava ja olisi hyvä saada aikaan ympäristö, jossa tunkeutumistestausta voi harjoitella turvallisesti.

1.2 Projektin tavoite ja tehtävä

Työn tehtävänä on selvittää, kuinka Penetration Testing -harjoitusympäristö toteutetaan vapailla ohjelmilla. Tavoitteena on saada aikaan ympäristö, jossa voi turvallisesti harjoitella tunkeutumistestausta. Tätä varten projektissa verrataan ympäristön toteutukseen ja tunkeutumiseen sopivia työkaluja ja kohteita keskenään. Ympäristön pitää myös olla helposti saatavilla ja hyödynnettävissä.

1.3 Tutkimuskysymykset

Seuraaviin kysymyksiin vastaaminen auttaa projektin tavoitteiden ja tehtävän saavuttamisessa. Kysymykset on jaoteltu pääkysymykseen ja alakysymyksiin.

Tutkimuksen pääkysymys:

- Kuinka vapailla ohjelmilla toteutetaan Penetration Testing -harjoitusympäristö?

Tutkimuksen alakysymykset:

- Millä välineillä Penetration Testing -harjoitusympäristö kannattaa toteuttaa?
- Millä välineillä tunkeutuminen Penetration Testing -harjoitusympäristöön kannattaa toteuttaa?
- Minkälaiseen kohteeseen kannattaa tunkeutua Penetration Testing -harjoitusympäristössä?

1.4 Kohderyhmä ja rajaus

Työn kohderyhmänä ovat Penetration Testing -aloittelijat, jotka hallitsevat Linuxin ja komentorivin käytön. Työ rajataan käsittämään vapaat ohjelmat. Työhön valitaan yksi virtuaaliympäristön mahdollistava kokonaisuus, yksi tunkeutumistyökalu ja yksi kohde, johon tunkeudutaan.

1.5 Menetelmät ja työn rakenne

Työ on toiminallinen opinnäytetyö ja siihen liittyvät tiedostot tuotetaan GitHubiin osoitteeseen <https://github.com/tonijaaskelainen/beginnerpentest>. Varsinainen opinnäytetyön raportti on saatavilla osoitteesta <https://github.com/tonijaaskelainen/beginnerpentest/blob/master/materiaalit/ONT.md>. Työssä pystytetään kannettavaan tietokoneeseen virtuaalinen ympäristö ja testataan sen toiminta.

Ensin työssä käydään läpi, mitä Penetration Testing on. Tämän jälkeen verrataan ympäristön toteutukseen soveltuvia työkaluja keskenään. Tämän yhteydessä kerrotaan vapaista ohjelmista ja virtualisoinnista. Seuraavaksi verrataan, minkälaisilla työkaluilla tunkeutuminen kannattaa toteuttaa. Sitten verrataan, minkälaiseen kohteeseen kannattaa tunkeutua Penetration Testing -harjoitusympäristössä. Kun nämä asiat ovat selvillä, niin ympäristö toteutetaan valitulla välineellä ja testataan valittuun kohteeseen tunkeutumista sitä varten valitulla työkalulla. Lopuksi pohditaan, miten työn tulokset vastaavat työn tavoitteeseen ja annetaan jatkokehitysehdotuksia.

1.6 Käsitteet

Seuraavaksi määritellään työssä käytettyjä käsitteitä.

Taulukko 1. Käsitteet ja niiden määritelmät

Käsite	Määritelmä
DoS	Denial of Service. Palvelunestohyökkäys. Palveluun kohdistuva hyökkäys, joka estää palvelun käyttämisen.
IDS	Intrusion Detection System. Tunkeutumisen havaitsemisjärjestelmä. Järjestelmä, joka yrittää havaita mahdollisen tunkeutujan.
IPS	Intrusion Prevention System. Tunkeutumisenestojärjestelmä. Järjestelmä, joka yrittää estää mahdollisen tunkeutumisen.
Kali Linux	Tunkeutumistestaukseen keskittynyt Linux-jakelu.
Metasploitable 2	Tarkoituksella haavoittuvaksi tehty kone.

Metasploit Framework	Monipuolinen tunkeutumistyökalu.
Palomuuuri	Laite tai ohjelmisto, joka suojaaa verkkoa suodattamalla tietoliikennettä tiettyjen sääntöjen mukaan.
Penetration Testing	Tunkeutumistestaus. Sallittu murtautuminen tietojärjestelmiin tietoturvan tilan kartoittamiseksi.
Pääsynhallinta	Tapa hallita pääsyä järjestelmään esimerkiksi käyttäjätunnuksilla ja salasanoilla.
Vagrant	Ohjelmisto, joka mahdollistaa helposti konfiguroitavien ja toistettavien järjestelmien luomisen yhtenäisellä tavalla.
Vapaat ohjelmat	Ohjelmat, joita on vapaus käyttää, kopioida, jakaa, tutkia, muuttaa ja parantaa.
Virtualisointi	Fyysisen version ohjelmallinen toteutus. Esimerkiksi virtualisoitu tietokone on virtuaalikone.

2 Penetration Testing eli tunkeutumistestaus

Tunkeutumistestauksella (engl. Penetration Testing) tarkoitetaan sallittua murtautumista tietojärjestelmiin ja IT-infrastruktuuriin, jotta saataisiin arvioitua tiettyjen puolustusmekanismien tehokkuus ja kartoitettua, onko ohjelmissa haavoittuvuuksia tai konfigurointivirheitä. Testauksen avulla on myös mahdollista selvittää, noudattavatko käyttäjät tietoturvaliikettä vai käyttäytyvätkö he riskialttiisti yrityksen kannalta. Tunkeutumistestaus suoritetaan yleensä manuaalisesti tai automaattisesti potentiaalisesti alttiisiin kohteisiin, kuten palvelimiin, päätelaitteisiin, langattomiin verkkoihin sekä tietoliikenne- ja mobiililaitteisiin. Jos tunkeutuminen onnistuu jonkin haavoittuvuuden kautta, niin testaajat saattavat yrittää peräkkäisiä hyökkäyksiä ja käyttöoikeuksien korottamista päästäkseen syvemmälle järjestelmiin ja saadakseen tarkemman kuvan järjestelmien tilasta. Kaikki löydetty haavoittuvuudet, joihin hyökkäys onnistui, esitellään yleensä IT- ja verkkojärjestelmien hallinnoijille strategisten päätösten tekoa ja parannuskeinojen priorisointia varten. Tunkeutumistestauksen perustarkoitus on mitata, kuinka käyttökelpoisia väärin konfiguroidut tai käyttäjien vaarantamat järjestelmät ovat hyökkääjille. Lisäksi testauksessa arvioidaan vastaavien tilanteiden aiheuttamat seuraukset resursseille tai toiminnoille. (Core Security 2015.)

2.1 Tunkeutumistestauksen merkitys ja hyödyt

Tunkeutumistestaus on tärkeää, koska tietomurrot ja palvelukatkokset ovat kalliita. Ne voivat taloudellisten kulujen lisäksi heikentää yrityksen mainetta ja vähentää asiakaskuntaa. Lisäksi on mahdotonta turvata kaikki mahdollinen tieto koko ajan. Perinteiset usean kerroksen puolustusmekanismit, kuten pääsynhallinta, salakirjoitus, IPS, IDS ja palomuurit

sekä uusien teknologioiden käyttöönotto ovat yhdessä tehneet järjestelmästä niin monimutkaisen, että kaikkien haavoittuvuuksien löytäminen ja eliminointi on entistä vaikeampaa. Tunkeutumistestaus auttaa identifioimaan ja priorisoimaan erilaisia turvallisuusriskejä arvioimalla yrityksen kykyä suojata hyödykkeitä sisäiseltä ja ulkoiselta väärinkäytöltä. Jotta voitaisiin varmistaa yhdenmukainen IT- ja verkkoturvallisuuden hallinta, pitäisi tunkeutumistestaus suorittaa säännöllisin väliajoin perusteellisesti. Näin voidaan tehokkaammin estää luvaton pääsy kriittisiin järjestelmiin ja niissä oleviin tietoihin. (Core Security 2015.)

Core Securityn (2015) mukaan tunkeutumistestaus pitäisi lisäksi suorittaa aina, kun

- verkkoinfrastruktuuria tai sovelluksia lisätään
- infrastruktuuriin tai sovelluksiin kohdistuu merkittäviä parannuksia/päivityksiä tai muutoksia
- uusia toimistotiloja otetaan käyttöön
- turvallisuuspaikkauksia otetaan käyttöön
- loppukäyttäjäpolitiikkaa muutetaan.

Tunkeutumistestauksen moniin hyötyihin kuuluu älykäs haavoittuvuuksien hallinta, verkon alhaallaolokulujen välttäminen, säännöstelyjen vaatimusten täyttäminen ja sakkojen välttäminen sekä yrityskuvan ja asiakaskunnan säilyttäminen. Älykäs haavoittuvuuksien hallinta tarkoittaa, että tunkeutumistestauksen avulla saadaan tarkkaa tietoa varsinaisista, hyökättävissä olevista haavoittuvuuksista ja näin pystytään identifioimaan kriittisimpien haavoittuvuuksien joukosta vähemmän merkittävät ja väärät hälytykset. Tämä mahdollistaa parannuskeinojen priorisoinnin ja turvallisuusresurssien tehokkaamman allokoinnin oikeaan aikaan oikeisiin paikkoihin. Verkon alhaallaolokulujen välttäminen tarkoittaa, että tunkeutumistestauksella pystytään ennakoivasti tunnistamaan ja osoittamaan riskejä ennen kuin hyökkäyksiä tai murtoja tapahtuu, ja näin vältetään taloudelliset riskit. Säännöstelyjen vaatimusten täyttäminen ja sakkojen välttäminen tarkoittaa, että tunkeutumistestaus auttaa yrityksiä osoittamaan, että ne noudattavat tiettyjä määräyksiä. Tunkeutumistestauksessa syntyvät tarkat raportit auttavat yrityksiä välttämään määräysten noudattamattomuudesta aiheutuvia sakkoja. Yrityskuvan ja asiakaskunnan säilyttäminen tarkoittaa, että tunkeutumistestaus auttaa yritystä säilyttämään sen maineen ja luotettavuuden, koska yksikin tietomurto voi karkottaa yrityksen nykyisiä asiakkaita ja estää uusien asiakkassuhteiden syntymistä. (Core Security 2015.)

2.2 Tunkeutumistestauksen standardi

Tunkeutumistestausta varten on olemassa lähes muodollinen standardi, jonka tarkoitus on tarjota yrityksille ja tunkeutumistestauksen tarjoajille yhteinen kieli. Nämä ovat täten kysei-

sen standardin kohderyhmänä. Standardin on tarkoitus sisältää minimilähtökohdat tunkeutumistestauksen suorittamista varten. Tämä tarkoittaa, että yritysten on tarkoitus pystyä vaatimaan tunkeutumistestaustyöltä tietyt lähtökohdat standardin avulla. Vastaavasti standardin on tarkoitus tarjota tunkeutumistestauksen tekijöille lähtökohdat huomioon otettavista asioista kartoituksesta raportointiin asti. Lisäksi standardin on myös tarkoitus sisältää korkeampia tasoja vaativampia turvallisuustarpeita varten. Nämä tasot on tarkoitus määritellä alakohtaisesti. Standardi tulee määrittelemään teknisen raportoinnin lisäksi myös mallin yritysjohdolle raportointia varten. (Penetration Testing Execution Standard 2015a.)

Tunkeutumistestauksen standardi määrittelee, mitä osa-alueita tunkeutumistestaukseen kuuluu. The Penetration Testing Execution Standardin (2014a) mukaan nämä ovat:

- Pre-engagement Interactions (työtä edeltävät toimenpiteet).
- Intelligence Gathering (tiedon kerääminen).
- Threat Modeling (uhkien mallintaminen).
- Vulnerability Analysis (haavoittuvuuksien analysointi).
- Exploitation (haavoittuvuuksiin hyökkääminen).
- Post Exploitation (hyökkäysten jälkeiset toimenpiteet).
- Reporting (raportointi).

Seuraavaksi käydään näitä osa-alueita läpi yksi kerrallaan. Osa-alueista pyritään antamaan kokonaiskuva.

2.2.1 Pre-engagement Interactions

Tämä osa-alue sisältää kysymyksiä, joihin kartoituksen kohteena olevan yrityksen on syytä vastata. Näin saadaan määritettyä tunkeutumistyön laajuus ja syitä, miksi ja mitä testejä yritys haluaa tunkeutumisyrityksen suorittavan. Työn laajuuden määrittelyssä on tarkoin otettava huomioon työn määrä, siihen käytettävä aika ja hinnoittelu. Asiakasyritys ei aina välttämättä tiedä, mitä asioita tarvitsee testata. Testaajan on siis pystyttävä antamaan opastusta, kun määritellään testattavia asioita. Lisäksi testaajan täytyy ymmärtää, että on eri asia testata yhtä kohdetta suurella intensiteetillä useiden aukkojen löytämistä varten ja montaa kohdetta vain, jotta löytyisi yksi tapa päästä järjestelmään sisään. Tunkeutumistestauksen tavoitteena on identifioida yritykselle haitallisia riskejä. Voi olla, että yrityksen tietoturvan taso ei ole riittävällä tasolla, ja tässä tapauksessa yritys hyötyy enemmän haavoittuvuuksien analysoinnista kuin varsinaisesta tunkeutumistestauksesta. Jos testaajalla on kokemusta testin suorittamisesta, niin yleensä testiin käytettävän ajan pystyy arvioimaan. Kaiken varalta tähän aikaan kannattaa lisätä 20 % yllätysten varalle. Jos työ valmistuu ennen 20 %:n lisäaikaa, niin testaajat voivat esimerkiksi syventyä vielä jonkin vaikean haavoittuvuuden selvittämiseen tai näyttää yrityksen tietoturva-asiantuntijoille, kuinka järjestelmiin päästiin sisään. Sen sijaan, jos osoittautuu, että työtehtävät ylittävät

sopimuksessa mainitun työn laajuuden, niin ne pitäisi kirjata Statement Of Work -dokumenttiin, joka molempien osapuolien on syytä allekirjoittaa ennen kuin lisätöitä voidaan aloittaa. Jotta rajat olisivat selkeät, niin työlle täytyy asettaa tarkat aloitus- ja lopetusajan kohdat. (Penetration Testing Execution Standard 2014b.)

Kysymykset on jaoteltu useampaan eri kategorioihin, joita ovat verkon tunkeutumistestaus, websovellusten tunkeutumistestaus, langattoman verkon tunkeutumistestaus, fyysinen tunkeutumistestaus ja sosiaalinen manipulointi. Lisäksi liiketoimintayksiköiden johtajille ja järjestelmänvalvojille on määritetty kysymyksiä. Esimerkiksi ennen verkkoon tunkeutumista on määriteltävä IP-osoitealueet ja toimialueet. On syytä varmistaa, että osoitteet todella kuuluvat kyseiselle yritykselle, jotta tunkeutuminen ei vahingossa tapahdu jonkin muun yrityksen verkkoon. Kolmansille osapuolille, kuten internet- ja pilvipalvelujen tarjoajille, on myös syytä ilmoittaa testauksesta, koska testaajien täytyy mahdollisesti noudattaa erityisiä sääntöjä kyseisiin kohteisiin hyökättäessä. Näin on myös eri maiden kohdalla, joten on hyvä ottaa selvää, missä maissa testattavat palvelimet ovat. Kaikki aiheet eivät välttämättä ole sopivia sosiaalista manipulointia varten, joten kysymyksillä saadaan selville sopivat aihealueet. Palvelunestohyökkäykset (DoS) voivat saada aikaan paljon vahinkoa, joten ne kannattaa suorittaa tuotantoympäristön kanssa identtisessä testiympäristössä. Hätätilanteiden varalle on myös kerättävä tarvittavat yhteystiedot. Yksi osa tunkeutumistestausta on testata yrityksen kykyä reagoida tietoturvaloukkauksiin ja jos järjestelmiin päästään sisälle ilman, että kukaan tietoturvahenkilö huomaa sitä, niin on löydetty valtava tietoturva-aukko. Tilanneraportteja pitäisi tehdä säännöllisin väliajoin. Kaikki arkaluontoinen viestintä pitäisi salakirjoittaa. Kaikkein arkaluontoisimpia tietoja välttämättä voi edes välittää kirjallisessa muodossa, eikä esimerkiksi henkilökohtaisia tietoja saa kerätä. (Penetration Testing Execution Standard 2014b.)

2.2.2 Intelligence Gathering

Tässä osion avulla kerättyjen tietojen pohjalta voidaan muodostaa hyökkäyssuunnitelma. Tiedon keräämisessä on kolme tasoa. Ensimmäisessä tasossa kerättävät tiedot mahdollista saada melkein kokonaan automatisoitujen työkalujen avulla. Toisessa tasossa kerättävät tiedot vaativat ensimmäisen tason automaattisten työkalujen lisäksi manuaalista analyysiä. Tasolla saadaan hyvä ymmärrys yrityksen liiketoiminnasta esimerkiksi organisaatiokaavion avulla. Kolmannella tasolla tarvitaan kaikkien alempien tasojen tietojen lisäksi paljon manuaalista analyysiä ja saadaan selville hyvin syvä ymmärrys esimerkiksi liiketoiminnalle tärkeistä asiakassuhteista. Tiedon keruun tasojen saavuttamiseksi on siis tarkoitus saada tiedustelun avulla mahdollisimman paljon tietoa, jota voidaan hyödyntää hyökkäyksissä. Tarvittavia tietoja löytyy yleensä myös julkisista lähteistä, joihin yritykset ja

työntekijät ovat antaneet tietojiaan. Nämä tiedot eivät kuitenkaan välttämättä ole ajan tasalla, ja ne voivat olla puutteellisia tai jopa vääriä. Aiemmassa vaiheessa määritellyt rajoitukset, aika ja tavoitteet määrittävät, kuinka paljon tietoa voidaan kerätä. Tiedon keruu voidaan suorittaa passiivisesti kolmansien osapuolten kautta, semi-passiivisesti ottamalla yhteyttä yrityksen julkisesti saataviin palveluihin ja aktiivisesti, jolloin tarkoituksena on saada selville esimerkiksi yritysverkon rakenne sekä palvelimissa olevat avoimet palvelut, hakemistot ja tiedostot. Vasta aktiivisen tiedonkeruun pitäisi vaikuttaa epäilyttävältä. (Penetration Testing Execution Standard 2014c.)

2.2.3 Threat Modeling

Tämä osio määrittelee, että uhkia mallinnettaessa täytyy säilyttää yhtenäinen tapa esittää uhat ja niiden kyky aiheuttaa haittaa sekä miten uhat vaikuttavat testattavaan yritykseen. Myös toistettavuus on tärkeää. Mallissa on neljä elementtiä: liiketoimintahyödykkeet, liiketoimintaprosessit, uhkien välittäjät/yhteisöt ja niiden kyvyt. Minimissään nämä kaikki pitäisi tunnistaa ja dokumentoida jokaisessa tunkeutumistestauksessa. Näiden lisäksi pitäisi ottaa huomioon "mitä jos" -tilanteet sen varalta, että hyödykkeitä menetettäisiin. Uhkien mallintaminen on kriittistä molemmille osapuolille, koska sen avulla yritys saa priorisoitua hyödykkeensä tärkeysjärjestykseen ja tunkeutumistestaajat pystyvät toimimaan mahdollisimman samankaltaisesti kuin hyökkääjä. Malli pitäisi luoda hyökkääjän näkökannalta, ja mallissa pitäisi hyödyntää tiedonkeruuvaiheessa saatuja tietoja. (Penetration Testing Execution Standard 2015b.)

Liiketoimintahyödykkeiden analyysissä saadaan selville, mitkä hyödykkeet ovat todennäköisimmin hyökkääjän kohteina ja mikä niiden arvo on yritykselle ja mikä vaikutus niiden menetyksellä olisi. Analyysin aikana saadaan myös organisatorista tietoa, kuten erilaisia käytäntöjä, suunnitelmia, tuotetietoja, markkinointitietoja, taloudellisia tietoja, työntekijöiden ja asiakkaiden tietoja ja teknisiä tietoja, joihin kuuluu esimerkiksi järjestelmien konfigurointitietoja ja käyttäjätunnuksiin liittyviä tietoja. Korotetuilla käyttäjätunnuksilla saadaan usein vaarannettua tietojärjestelmän turvallisuus. (Penetration Testing Execution Standard 2015b.)

Liiketoimintaprosessien analyysin avulla testaajat voivat saada selville, miten yritys saa rahaa, ja jos prosesseista löytyy virheitä, niin on mahdollista löytää keino, jolla yritys saadaan menettämään rahaa. On tarpeen erottaa kriittiset ja ei-kriittiset prosessit toisistaan, mutta on myös mahdollista, että useammat ei-kriittiset prosessit muodostavat yhdessä kriittisen virheen, jota voidaan myöhemmin hyödyntää tunkeutumistestauksessa. Analy-

soitaviin prosesseihin kuuluu teknisen infrastruktuurin, tiedon ja henkilöstön tukiprosesseja. Prosesseihin voi olla integroituna myös kolmas osapuoli. (Penetration Testing Execution Standard 2015b.)

Uhkien välittäjien/yhteisöjen analysoinnissa pyritään tunnistamaan sisäiset ja ulkoiset tekijät, joista voi koitua uhkaa yritykselle. Jollei tiettyä välittäjää pystytä tunnistamaan, niin yritetään löytää laajempi yhteisö, josta voisi aiheutua uhkia. Sisäisiä mahdollisesti uhkaa aiheuttavia tekijöitä ovat esimerkiksi työntekijät, johto, ylläpitäjät ja kehittäjät. Ulkoisia ovat taas esimerkiksi yrityskumppanit, kilpailijat, tavarantoimittajat ja rikolliset. (Penetration Testing Execution Standard 2015b.)

Kun uhkia välittävä yhteisö on tunnistettu, niin on aika selvittää sen kyvyt, joilla yrityksen turvallisuus pystytään vaarantamaan. Tämän selvittämiseksi täytyy analysoida, mitä työkaluja yhteisöllä on, miten he pääsevät hyödyntämään järjestelmissä olevia haavoittuvuuksia, heidän viestintämenetelmänsä ja miten he pääsevät hyödykkeisiin käsiksi. Hyökkääjien motivaatiot vaihtelevat jatkuvasti, joten motivaatioita pitäisi myös mallintaa. Mallia täydentää myös yrityksen vertaaminen vastaavan saman alan yrityksen kanssa. (Penetration Testing Execution Standard 2015b.)

2.2.4 Vulnerability Analysis

Haavoittuvuuksia testattaessa etsitään virheitä järjestelmistä ja sovelluksista, joista hyökkääjä voi saada hyötyä. Virheet voivat olla esimerkiksi järjestelmän konfigurointivirheitä tai ohjelmiston suunnitteluvirheitä. Testauksen syvyyden ja laajuuden pitäisi vastata tavoitteisiin. On olemassa aktiivista ja passiivista testausta. Aktiivisessa testauksessa ollaan suoraan tekemisissä testattavan komponentin kanssa, ja aktiivinen testaus voidaan suorittaa joko automaattisesti tai manuaalisesti. Aktiivisessa testauksessa käytetään esimerkiksi verkko-, palvelu- ja websovellusskannereita. Passiivisessa testauksessa voidaan analysoida tiedostoja kuvaavia tietoja (tekijä, yritys jne.), eli metadataa tai monitoroida tietoliikennettä. Testien tulokset pitäisi validoida esimerkiksi kokeilemalla manuaalisesti samoja yhteyksiä, joita on skannattu automaattisten työkalujen avulla. Kun haavoittuvuus on löydetty, niin täytyy tutkia, kuinka tarkka arvio haavoittuvuudesta on ja miten haavoittuvuutta pystyisi hyödyntämään. Tähän on olemassa apuna haavoittuvuustietokantoja, esimerkiksi CVE (Common Vulnerabilities and Exposures), ja tietokantoja, jotka listaavat, miten tunnettuihin haavoittuvuuksiin pystyy hyökkäämään. Näitä tietokantoja kutsutaan Exploit Databaseiksi. Julkisesti on saatavilla myös laitteiden ja ohjelmistojen manuaaleja sekä oletustunnuksia ja -salasanoja, joita tunkeutumistestaajan kannattaa kokeilla. (Penetration Testing Execution Standard 2014d.)

2.2.5 Exploitation

Tässä vaiheessa keskitytään pelkästään siihen, että päästään sisään järjestelmiin tai käsiiksi resursseihin. Nyt yritetään tunnistaa pääkohde, jonka kautta päästään sisään yritykseen sekä tunnistetaan korkea-arvoisia hyödykkeitä. Edellisen vaiheen kunnollinen suorittaminen helpottaa tätä vaihetta, koska valmiina olevasta listasta voidaan valita kohde, johon todennäköisimmin päästään sisään, ja jolla on korkein vaikutus yritykseen. (Penetration Testing Execution Standard 2014e.)

Yrityksellä voi olla käytössä useita erilaisia keinoja, joilla hyökkäykset pyritään estämään. Tällaiset tilanteet pitäisi pystyä kiertämään, mutta jollei siinä onnistuta, niin on aika harkita vaihtoehtoisia hyökkäystapoja. Näistä vastakeinoista olisi hyvä ottaa selvää, jos mahdollista, ennen kuin hyökkäys suoritetaan. Vastakeinoja ovat esimerkiksi virustentorjuntaohjelmistot, tunkeutumisenestojärjestelmät ja palomuurit. Näitä voidaan yrittää välttää tiedon koodauksella, pakkaamisella ja salauksella. On myös mahdollista injektoida prosessi, jolloin haitallinen koodi saadaan piiloon luotettavan prosessin sisään. Hyökkäyksiä olisi parasta pystyä suorittamaan suoraan RAM-muistissa, koska levyille kirjoitettaessa tiedot tulevat ilmi, kun levyt skannataan. Tiettyihin tilanteisiin täytyy räätälöidä itse tarvittavat hyökkäykset. Viimeinen vaihtoehto on yrittää löytää kehittyneillä menetelmillä uusia haavoittuvuuksia, joita ei ole vielä missään haavoittuvuustietokannassa. Näihin haavoittuvuuksiin hyökkäämistä kutsutaan nollapäivähyökkäykseksi. Jos ohjelman lähdekoodiin pääsee käsiiksi ja siitä löytyy virheitä, voi ohjelma olla altis nollapäivähyökkäyksille. Turvatoimia kiertämällä on myös mahdollista päästä laitteisiin fyysisesti käsiiksi. Hyökkäysvaiheen suurin haaste on päästä vähimmällä vaivalla yrityksen sisään, pysyä havaitsemattomana ja vaikuttaa suurimmalla mahdollisimmalla tavalla yrityksen liikevaihtoon. Edellisten vaiheiden suorittaminen hyvin helpottaa tätä vaihetta. (Penetration Testing Execution Standard 2014e.)

2.2.6 Post Exploitation

Kun laite on saatu haltuun, niin on aika määritellä laitteen arvo ja pitää laite tunkeutujan hallinnassa myöhempää käyttöä varten. Laitteen arvo määräytyy sen sisältävän arkaluontaisen tiedon lisäksi sen perusteella, kuinka hyödyllisesti laitteen kautta voidaan yhä edelleen vaarantaa yritysverkkoa. Tässä vaiheessa kuvataan tapoja, joilla voidaan tunnistaa arkaluontoinen tieto ja järjestelmän konfiguraatioasetukset, viestintäkanavat ja suhde muihin verkkolaitteisiin. Täytyy myös löytää vähintään yksi keino, jolla laitteeseen pääsee myöhemmin käsiiksi. Tässä vaiheessa täytyy myös noudattaa tiettyjä sääntöjä, joilla suo-

jellaan asiakasta (esimerkiksi kaikki järjestelmiin tehtyt muutokset dokumentoidaan ja pyritään palauttamaan alkuperäiseen tilaan) ja testaajaa (esimerkiksi yrityskäytäntöjen ja lakien noudattaminen). (Penetration Testing Execution Standard 2014f.)

Verkon määritysten analysoinnin avulla voidaan löytää lisää kohteita, joihin voi tunkeutua edellisten kautta. Kohteiden merkitys voi selvitä kohteeseen asennettujen ohjelmien ja palvelujen avulla. Arkaluontoista tietoa voidaan saada selville esimerkiksi näppäinpainalluksia tallentavien ohjelmien, näyttökuvien, tietoliikenteen monitoroinnin ja käyttäjän historiatietojen ja tiedostojen avulla. (Penetration Testing Execution Standard 2014f.)

2.2.7 Reporting

Lopullisessa tunkeutumistestausraportissa on kaksi pääosiota, joissa kerrotaan eri kohderyhmille testin tavoitteet, menetelmät ja tulokset. Toinen osa on tarkoitettu johdolle ja toinen sisältää testiin liittyvät tekniset yksityiskohdat. Johdolle raportoitaessa kuvataan löydetty asiat korkealla tasolla. Tämän osion olisi hyvä sisältää taustatiedot, testin yleistila, riskien arviointi, yleiset tulokset, yhteenveto suosituksista ja strateginen etenemissuunnitelma. Teknisen osion tulisi sisältää johdanto, osiot tiedon keruusta aktiivisesti ja passiivisesti sekä yritykseen ja henkilökuntaan liittyvät kerätyt tiedot. Lisäksi osioon pitäisi kuulua haavoittuvuusarviointi ja osio, jossa haavoittuvuuksien olemassaolo varmistetaan hyökkäämällä niihin. Lopuksi tekniseen osioon pitäisi sisällyttää hyökkäyksen jälkeen tehty toimenpiteet ja määritelmä riskeistä. Koko raportin lopuksi pitäisi antaa ohjeet turvallisuuden kehittämiseen ja hoitotoimenpiteisiin tulevaisuutta ajatellen. (Penetration Testing Execution Standard 2014g.)

3 Välineet ympäristön toteutukseen

Tässä luvussa verrataan ympäristön toteutukseen tarvittavia välineitä keskenään. Näistä välineistä otetaan yksi mukaan varsinaiseen toteutusvaiheeseen. Ympäristöstä tehdään virtuaalinen ja se toteutetaan vapailla ohjelmilla. Siksi luvun alussa käydään lyhyesti läpi, mitä vapaat ohjelmat ja virtualisointi tarkoittavat.

3.1 Vapaat ohjelmat

Free Software Foundation määrittelee vapaat ohjelmistot käyttäjän vapautta kunnioittaviksi ohjelmistoiksi. Kyseiset ohjelmistot eivät välttämättä ole ilmaisia, vaan vapaan ohjelmiston käyttäjällä on vapaus käyttää, kopioida, jakaa, tutkia, muuttaa ja parantaa vapaita ohjelmistoja haluamallaan tavalla. Jotta kaikki tämä olisi mahdollista, vapaista ohjelmis-

toista on tarjottava sekä suoritettava binääritiedosto että lähdekooditiedostot. Jos ohjelmaa on muokattu, niin siitä pitää tarjota myös kyseiset tiedostot. Vapaista ohjelmistoja voi myös käyttää kaupalliseen toimintaan. (Free Software Foundation 2016a/1996.)

Vapaiden ohjelmistojen tekijät eivät saa poistaa oikeuksia ohjelmistojen käyttäjiltä missään vaiheessa, kunhan käyttäjät eivät tee mitään väärää. Vapaiden ohjelmistojen oikeuksien suojaamista varten on kehitetty Copyleft, mutta vapaita ohjelmistoja voi olla myös ilman Copylefttiä. (Free Software Foundation 2016a/1996.) Copyleft on tekijänoikeusmenetelmä, jolla esimerkiksi ohjelmistot ja niiden muokatut ja laajennetut versiot saadaan vapaiksi. Näin vapaan ohjelmiston parannellusta versiosta ei voi tehdä omisteista ohjelmistoa (proprietary software). Omisteisten ohjelmistojen kanssa Copyrightia käytetään rajoittamaan käyttäjän vapautta ohjelmistoihin. Sitä vastoin Copyleft takaa, että käyttäjien vapaus ohjelmistoihin säilyy ennallaan. Vaikutus on siis vastakkainen ja tästä nimitys left. Copyleft-töihin täytyy kuitenkin merkitä tekijänoikeus Copyright-sanan tai -symbolin avulla, jotta vapaiden ohjelmistojen käyttäjien alkuperäinen vapaus säilyisi. Jos vapaat ohjelmistot olisivat tekijänoikeudettomia (public domain), niin niistä voitaisiin myös tehdä omistusoikeudellisia ja näin käyttäjiltä olisi riistetty vapautta. (Free Software Foundation 2015/1996.)

Vapaita ohjelmistoja varten on olemassa useita eri lisenssejä. Näistä uusin Free Software Foundationin kehittämä lisenssi on GNU General Public License (GPL) version 3. Tämä on suositeltu lisenssi useimpiin vapaisiin ohjelmistoihin. GNU GPL:stä on muita versioita eri tarpeisiin, kuten GNU Lesser General Public License (LGPL) version 3, joka sallii vapaiden ohjelmistojen linkittämisen ei-vapaisiin ohjelmamoduuleihin sekä AGPLv3 (GNU Affero General Public License version 3), joka sopii verkossa toimiville vapaille ohjelmistoille, koska lisenssi sallii käyttäjän saada verkossa toimivan ohjelmiston lähdekoodin itselleen. GNU All-Permissive License on tarkoitettu pienille tiedostoille. Muita GNU GPL:n kanssa yhteensopivia vapaita lisenssejä ovat muun muassa patentteihin kantaa ottava Apache License, Version 2.0, alkuperäisestä muokattu BSD-lisenssi sekä nk. MIT-lisenssi (Expat License ja X11 License). Vapaata dokumentaatiota varten on olemassa esimerkiksi lisenssi GNU Free Documentation License, eli GNU FDL. (Free Software Foundation 2016b/2014.)

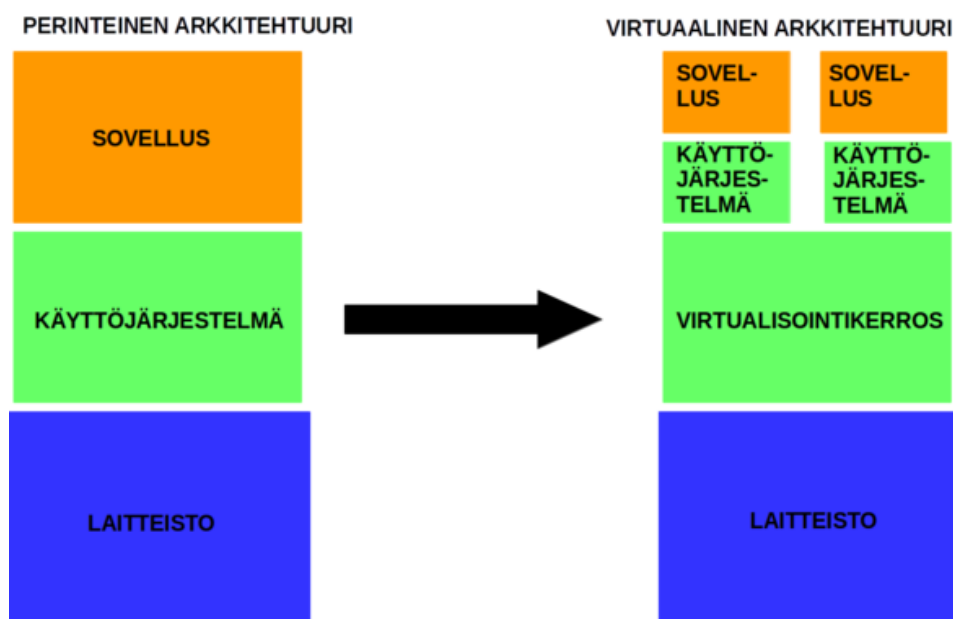
Avoimen lähdekoodin ohjelmistot (Open Source Software) ja vapaat ohjelmistot (Free Software) ovat määritelmiltään lähellä toisiaan, mutta ne välittävät eri merkityksen. Avointen ohjelmistojen määritelmä on vain hieman löysempi kuin vapaiden ohjelmistojen. Kuitenkin useimpien käyttäjien ilmeisin tulkinta avoimelle ohjelmistolle on, että avointa lähdekoodia pystyy katsomaan, vaikka tämä ei ole ainut avointen ohjelmistojen sallima asia.

Vapaiden ohjelmistojen nimestä olisi suoraan tarkoitus käydä ilmi, että käyttäjällä on lisäksi vapaus tehdä ohjelmistolla, mitä haluaa. Englannin kielen sana free ei kuitenkaan ole yksikäsitteinen ja siksi ohjelmistoja on yritetty kuvata myös ranskan-/espanjankielisellä libre-sanalla, mutta sitä ei ymmärretä esimerkiksi Intiassa. (Stallman 2016/2007.)

3.2 Virtualisointi

Virtualisointi tarkoittaa virtuaalisen version luomista jostain asiasta todellisen version sijaan. Virtualisoinnista on kyse esimerkiksi silloin, kun fyysinen kiintolevy jaetaan erillisiin loogisiin osioihin. (Rouse & Kirsch 2016/2006.) Virtuaalinen versio toteutetaan ohjelmallisesti eikä se siten ole fyysinen. Fyysisen kiintolevyn osiot kuitenkin näkyvät käyttöjärjestelmässä ikään kuin erillisinä kiintolevyinä.

Virtualisoinnin avulla tarvittavat resurssit abstrahoidaan virtualisoinnin alla olevalta todelliselta ohjelmistolta tai laitteistolta. Palvelinvirtualisoinnissa käytetään hypervisoriksi kutsuttua ohjelmistokerrosta emuloimaan laitteistoa. Kun virtualisoitu (vieras-)käyttöjärjestelmä kommunikoi tämän emuloidun laitteiston kanssa, niin se ei välttämättä tiedä ollenkaan, ettei kyseessä ole todellinen laitteisto. Virtualisoidun ympäristön suorituskyky ei ole täysin samaa luokkaa aidon laitteiston kanssa, mutta virtualisoitu ympäristö on joustava, koska se poistaa ohjelmistojen laitteistoriippuvuuden. (Rouse & Kirsch 2016/2006.) Kuva 1 havainnollistaa, miten virtualisoitu arkkitehtuuri eroaa virtualisointikerroksen avulla perinteisestä arkkitehtuurista. Kokonaan itsetehty kaksiulotteinen kuva 1 mukailee TechTargetin Rousen & Kirschin julkaiseman kolmiulotteisen kuvan ideaa.



Kuva 1. Perinteinen ja virtuaalinen arkkitehtuuri

Palvelinvirtualisointi on perinteisin tapa virtualisoida. Sen lisäksi virtualisointia käytetään nykyään tietoverkkojen, tallennustilan, datan, työasemien ja sovellusten yhteydessä. Palvelinvirtualisoinnin tarkoituksena on helpottaa resurssien käyttöä ja jakamista. Virtualisoinnin mahdollistavasta ohjelmistokerroksesta eli hypervisorista on olemassa kahta eri tyyppiä. Tyypin 1 hypervisor toimii suoraan laitteiston päällä ja tyypin 2 hypervisor tarvitsee toimiakseen käyttöjärjestelmän. Verkkovirtualisoinnissa kaista jaetaan itsenäisiin kanaviin ja näin saadaan muodostettua helpommin hallittavia osia verkosta. Tallennusvirtualisoinnissa useista verkkotallennuslaitteista muodostetaan ikään kuin yksi tallennusväline, jota hallitaan keskitetyn konsolin kautta. Datan virtualisoinnissa datan teknisiä tietoja, kuten sijainti ja tiedostomuoto, abstrahoidaan laajemman saavutettavuuden ja joustavuuden saavuttamiseksi. Työpöytävirtualisoinnissa palvelinkeskuksiin virtualisoidaan työasemia, joihin voi ottaa yhteyden ns. thin clientillä. Sovellusvirtualisoinnissa abstrahoidaan sovelluseros erilleen käyttöjärjestelmästä. Näin sovellus voidaan suorittaa kapseloidussa muodossa, ja tällöin esimerkiksi Windows-ohjelman voi saada toimimaan Linuxilla. (Rouse & Kirsch 2016/2006.)

3.2.1 VirtualBox

VirtualBox on Oraclen kehittämä virtualisointiin tarkoitettu tyypin 2 hypervisor, joka toimii Windows-, Mac-, Linux- ja Solaris-käyttöjärjestelmissä. VirtualBoxin avoin lähdekoodi on lisensoitu GNU General Public License versiolla 2. VirtualBox sopii kaikkialle: sulauteuista järjestelmistä työasemiin, palvelinkeskuksiin ja pilvipalveluihin. VirtualBoxiin voi luoda virtuaalikoneita yleisimpiä käyttöjärjestelmiä varten. Virtuaalikoneet sisältävät käyttöjärjestelmälle virtualisoidun laitteiston ominaisuudet, ja niihin voi asentaa Guest Additionseja suorituskyvyn nostoa ja lisäominaisuuksia varten. (Oracle Corporation 2016/2004.)

VirtualBoxin pääominaisuuksiin kuuluvat muun muassa siirrettävyys, Guest Additionsit, hyvä laitteistotuki, snapshotit, virtuaalikoneryhmät, puhdas modulaarinen arkkitehtuuri ja etäyhteys virtuaalikoneen näyttöön. VirtualBox ei myöskään vaadi, että fyysinen laitteisto tukisi virtualisointia, vaan useissa tapauksissa ohjelmistopohjainen virtualisointi riittää. (Oracle Corporation 2016/2004.)

Siirrettävyyden ansiosta VirtualBox ja sen virtuaalikoneet toimivat useissa eri käyttöjärjestelmissä. VirtualBox käyttää virtuaalikoneita varten OVF-tiedostomuotoa (Open Virtualization Format), ja se pystyy täten käyttämään myös muiden virtualisointisovellusten virtuaalikoneita, jotka ovat OVF-muodossa. VirtualBoxin virtuaalikoneita voi myös kloonata. Guest

Additionsien avulla virtuaalikoneet tukevat automaattista resoluution asettamista, kiihdytettyjä 3D-grafiikoita ja saumattomia ikkunoita. Niiden avulla voi käyttää myös jaettuja kansioita isäntäkoneen ja virtuaalikoneen välillä. (Oracle Corporation 2016/2004.)

Snapshottien avulla voidaan tallentaa virtuaalikoneiden tila ja näin voidaan luoda erilaisia konfiguraatioita tai palata johonkin toiseen tilaan, josta snapshot oli otettu. Esimerkiksi jonkin häiriön takia voidaan palata helposti toimivaan snapshottiin. Snapshotit sisältävät virtuaalikoneiden asetusten tilan, virtuaalikiintolevyjen tilan ja myös virtuaalikoneiden muistin tilan, jos snapshot on otettu käynnissä olevasta virtuaalikoneesta. Näin voidaan palata ohjelman suorituksessa suoraan siihen kohtaan, josta snapshot oli otettu. (Oracle Corporation 2016/2004.)

VirtualBoxiin on myös olemassa laajennuspaketti, joka tuo lisää toimintoja. Lisäksi VirtualBoxia pystyy käyttämään ohjelmallisesti sitä varten kehitetyn Software Development Kitin avulla. VirtualBoxin hallintaa varten on graafisen käyttöliittymän lisäksi olemassa kevennetty graafinen versio, komentorivityökalu ja etäyhteyspalvelin, joka vaatii että palvelinta käyttävään virtuaalikoneeseen on asennettu ja otettu käyttöön Virtual Remote Desktop Extension -lisäosa. (Oracle Corporation 2016/2004.)

3.2.2 Vagrant

Vagrant mahdollistaa helposti konfiguroitavien ja toistettavien sekä siirrettävien järjestelmien hallitsemisen yhtenäisellä tavalla. Koneet luodaan providerin avulla ja provisioner-työkalujen avulla koneisiin voidaan asentaa ohjelmistoja ja tehdä määrittäksiä. (sethvargo 2016a.) Vagrant tukee oletuksena VirtualBoxia, Dockeria ja Hyper-V:tä providereina. Tuki muille providereille onnistuu pluginien avulla. (sethvargo 2016b.) Provisioner-työkaluihin kuuluvat shell-skriptit ja konfiguraationhallintajärjestelmät. Niiden avulla ohjelmien asentamisesta tulee automatisoitua ja toistettavissa olevaa. (sethvargo 2016c.) Vagrant on HashiCorpin kehittämä ja toimii Windows-, Mac- ja Linux-käyttöjärjestelmissä (HashiCorp a).

Kehittäjille Vagrantin tarjoama hyöty on, että eri käyttöjärjestelmissä voidaan käyttää yhtenäiseksi konfiguroitua ympäristöä ja he voivat silti käyttää kehittämiseen haluamiaan editoreita ja työkaluja muokkaamalla valmista ympäristöä tarpeidensa mukaan. Ylläpitäjille Vagrant helpottaa hallintaskriptien testaamista paikallisesti ja mahdollistaa saman konfiguraation käytön ja työnkulun pilvipalveluissa. Suunnittelijat saavat Vagrantin avulla helposti käyttöönsä tarpeidensa mukaan toimivan ympäristön ja he voivat keskittyä ongelmien kor-

jaamisen sijaan suunnitteluun. (sethvargo 2016a.) Vagrant-ympäristön voi myös jakaa esimerkiksi tiimin jäsenille. Jakaminen onnistuu HTTP:n ja SSH:n lisäksi suoraan Vagrant-ympäristössä auki olevaan porttiin. Jakaminen vaatii ngrok-ohjelmiston toimiakseen. (sethvargo, pearkes & chrisroberts 2017.) Vagrant on lisensoitu MIT-lisenssillä (mitchellh, fgrehm & aqnouch 2016/2010).

Vagrantfilessa kuvataan projektiin liittyvät tiedot, kuten koneiden tyypit, niiden määrittelyt ja provisionerit. Jokaiseen projektiin kuuluu vain yksi Vagrantfile ja se on tarkoitus kommitoida versionhallintajärjestelmään, jotta muut projektin kehittäjät pääsevät koodiin käsiksi ja voivat suorittaa koodin Vagrantilla. Vagrantfilet ovat siirrettäviä Vagrantin tukemien alustojen välillä. Vagrantfile kirjoitetaan Ruby-kielellä, mutta Rubyä ei välttämättä tarvitse osata Vagrantfilen konfiguroimista varten, koska tiedosto koostuu pääasiassa yksinkertaisista muuttujien arvojen asettamisesta. Vagrant etsii Vagrantfilea oletusarvoisesti ensin työhakemistosta ja sen jälkeen tämän ylähakemistoista. Jos Vagrantfileja on useita, niin niissä olevat määrittelyt yhdistetään tietyssä järjestyksessä. (sethvargo 2016d.) Yhteen Vagrantfileen voi määrittellä useamman koneen. Näin voidaan mallintaa esimerkiksi palvelinympäristöä, jossa eri palvelut on eriytetty omiin koneisiinsa. (sethvargo & afeld 2017.)

Virtualisointiohjelmistot sisältävät usein komentorivityökaluja virtuaalikoneiden hallintaa varten. Vagrant käyttää näitä virtuaalikoneiden komentorivityökaluja sisäisesti ja pystyy ratkaisemaan automaattisesti yleisimmät virhetilanteet, joten ympäristöjen hallinta on helppoa. Erikoistapaukset saattavat kuitenkin vaatia custom-skriptien käyttöä. (sethvargo & mitchellh 2017.)

Vagrantin paketoituja koneita kutsutaan nimellä box. Boxeja saa hankittua julkisesta Vagrant-katalogista Vagrant Cloudista. Lisäksi Vagrant tukee avoimen lähdekoodin Bento-boxeja. Boxeja on tarjolla useille eri providereille ja boxit sisältävät useita eri käyttöjärjestelmiä. Boxeja on myös tehty erityistarkoituksiin, kuten LAMP-stackia varten. (sethvargo, vmelnik-ukraine & chrisroberts 2017.)

Vagrant tarjoaa korkean tason tietoverkkoasetuksia, jotka toimivat eri providerien välillä. Lisäksi on tarjolla erityisiä provider-kohtaisia tietoverkkoasetuksia. (sethvargo 2016e.) Vagrant Reload -plugin on GitHub-käyttäjän aidanns kehittämä plugin, joka tarjoaa mahdollisuuden tehdä reload eli ladata virtuaalikone uudelleen provision-vaiheen aikana (aidanns 2013a). Vagrantin oma komento `vagrant reload --provision` ei tee tätä, vaan silloin provision-vaihe alkaa kokonaan alusta (Keviv, kamito300 & sethvargo 2016). Vagrant Reload -plugin on lisensoitu MIT-lisenssillä (aidanns 2013b). GitHub-käyttäjä dotless-de on kehittänyt Vagrantille pluginin, jonka avulla voi ladata VirtualBoxin Guest Additionsit

automaattisesti. Tämä plugin on nimeltään vagrant-vbguest ja se on lisensoitu MIT-lisenssillä. (dotless-de 2017.)

3.2.3 Packer

Packer on HashiCorpin kehittämä ohjelmisto, jolla voidaan automatisoida virtuaalikoneiden luonti. Packerissa on valmiiksi tuki muutamalle virtualisointiohjelmistolle (esimerkiksi VirtualBox) ja pilvialustalle (esimerkiksi Microsoft Azure). Pluginien avulla Packeriin voi lisätä alustoja. (HashiCorp b.) Packerin tarkoitus on käyttää yhtä lähdekonfiguraatiota ja luoda sen avulla identtisiä koneimageja useille eri alustoille. Packer toimii yleisimmissä käyttöjärjestelmissä. (sethvargo 2017a.) Packerin kanssa voi hyödyntää konfiguraationhallintatyökaluja. Packer on nopea, vakaa ja hyvin testattavissa. Kaikki Packerin luomat imageet ovat identtisiä alustasta riippumatta. Tämä mahdollistaa Packer-imagejen siirrettävyyden kehitysympäristöstä aina tuotantoon asti, vaikka kehitys- ja tuotantoympäristöt käyttäisivät eri virtualisointiohjelmistoja. (sethvargo 2017b.)

Packerissa on käytössä oma terminologiansa. Builderit luovat koneen imagen yhdelle alustalle. Build on yksittäinen tehtävä, jonka avulla koneen image muodostuu. Yhden buildin tulosta kutsutaan artifactiksi. Post-Processorit käsittelevät aiemmin luotua artifactia ja luovat uuden artifactin. Provisionerit asentavat koneeseen ohjelmia, ennen kuin siitä tehdään image. Templatet ovat JSON-tiedostoja, jotka määrittävät buildin. (mwhooker, sethvargo & feministy 2017.) JSON-templateihin tehdään siis Packerin koneiden tarvittavat määrittelyt, hieman samaan tapaan kuin Vagrantfileen tehdään Vagrantin koneiden tarvittavat määrittelyt. Packeriin verrattuna Vagrant näyttää olevan korkeamman tason ohjelmisto, koska se tarjoaa työkaluja virtuaalikoneiden hallintaa varten (esimerkiksi käynnistys ja sammuttaminen).

Packer luomasta virtuaalikoneimagesta voi tehdä Vagrant-boxin Post-Processor-työkalujen avulla. Yleensä kaikki välissä muodostuneet artifactit poistuvat ja vain viimeisin artifact jää jäljelle Post-Processoreja käytettäessä. (sethvargo, mwhooker & Purple90 2017.)

Packer on lisensoitu Mozilla Public License versiolla 2 (mitchellh 2013).

4 Välineet ympäristöön tunkeutumiseen

Tässä luvussa verrataan ympäristön tunkeutumiseen tarvittavia välineitä keskenään. Näistä välineistä otetaan yksi mukaan varsinaiseen toteutusvaiheeseen.

4.1 Kali Linux

Kali Linux on Offensive Securityn kehittämä vapaa ja avoin tunkeutumistestaukseen keskittynyt Linux-jakelu. Se toimii 32- ja 64-bittisten laitteiden lisäksi ARM-laitteissa. (Kali Linux 2017.) Kali Linux perustuu Debian Linuxiin ja se on lisensoitu GNU GPL -lisenssillä, mutta Kalin non-free-osiossa on myös ei-vapaita ohjelmistoja (Offensive Security 2017a).

Kali on BackTrack Linuxista uudelleenkehitetty versio, joka sisältää useita satoja tunkeutumistestaukseen ja tietoturvaan liittyviä työkaluja. Kalin käyttäjän odotetaan osaavan käyttää Linuxia yleisesti, koska jakelu on tarkoitettu tunkeutumistestauksen ammattilaisille. (Offensive Security 2017b.) Täten Offensive Security (2017c) suosittelee Linux-aloittelijoille Debian-, Ubuntu- tai Mint-jakeluita.

Kali Linuxissa on muutamia muutoksia tavallisiin Linux-jakeluihin verrattuna. Kali on tarkoitettu pääkäyttäjän eli rootin käytettäväksi, koska useat tunkeutumistestauksessa käytettävät työkalut vaativat joka tapauksessa root-oikeudet toimiakseen. Lisäksi Kali estää tietoverkkopalveluiden automaattisen käynnistymisen koneen käynnistyessä uudelleen. (Offensive Security 2017c.) On kuitenkin mahdollista tehdä listoja, joissa sallitaan poikkeukset tähän käytäntöön (Offensive Security 2017d). Kali sisältää myös custom-ytimen ja minimaalisen määrän luotettuja pakettivarastoja (Offensive Security 2017c).

Kun Kalia käyttää Livenä (suoritetaan joltain tietovälineeltä ilman asennusta), niin voi valita Forensics Moden. Tämä tila ei jätä koneen kiintolevylle mitään jälkiä eikä toimenpiteitä tapahdu automaattisesti ilman käyttäjän toimia (Offensive Security 2017e).

Kalin työkaluja on jaettu metapaketteihin, joten tarvittaessa vain tiettyyn tehtävään soveltuvat työkalut voi asentaa metapaketin avulla. Esimerkiksi kali-linux-top10-metapaketista löytyy suositut aircrack-ng, burpsuite, hydra, john, maltego, metasploit-framework, nmap, sqlmap, wireshark ja zaproxy. (Offensive Security 2017f.)

Offensive Security (2017g) ryhmittelee työkalut käyttötarkoituksen mukaan seuraaviin ryhmiin:

- Exploitation Tools (haavoittuvuuksiin hyökkäävät työkalut).
- Forensics Tools (rikostekniset työkalut).
- Hardware Hacking (laitteiston hakkerointi).
- Information Gathering (tiedon kerääminen).
- Maintaining Access (sisäänpääsyn ylläpito).
- Password Attacks (salasanahyökkäykset).
- Reporting Tools (raportointityökalut).
- Reverse Engineering (takaisinmallinnus).

- Sniffing & Spoofing (tietoliikennepakettien kaappaaminen ja muokattujen pakettien lähettäminen).
- Stress Testing (kuormitustestaus).
- Vulnerability Analysis (haavoittuvuuksien analysointi).
- Web Applications (web-sovellukset).
- Wireless Attacks (langattomat hyökkäykset).

Työkaluilla voi olla useampi käyttötarkoitus. Kalin käytäntöihin ei kuulu yleisesti laittomiin tarkoituksiin (esim. DoS) käytettyjen työkalujen tarjoaminen oletuksena. (Offensive Security 2017h.)

4.1.1 Metasploit Framework

Metasploit Framework on Rapid7:n kehittämä Ruby-ohjelmointikieleen perustuva modulaarinen tunkeutumistestausalusta, joka sisältää työkaluja tietoturva- ja haavoittuvuuksien testaamista, tietoverkkojen listausta, hyökkäysten suorittamista ja havaituksi tulemisen välttämistä varten (Rapid7 a). Metasploitista on tarjolla viisi eri versiota. Avoimen lähdekoodin Metasploit Framework muodostaa pohjan Metasploitin muille kaupallisille versioille. (Rapid7 b.) Nämä ovat ominaisuuksien mukaan kasvavassa järjestyksessä listattuna Metasploit Community, Metasploit Express, Metasploit Ultimate ja Metasploit Pro (Rapid7 c). Metasploitin käyttöä varten on olemassa eettisen hakkeroinnin opas nimeltään Metasploit Unleashed (Offensive Security 2017i). Metasploit Framework on lisensoitu BSD-3-clauselissenssillä (bcook-r7 2017/2006), ja se toimii Windows-, Mac- ja Linux-käyttöjärjestelmissä (Cook 2017).

Metasploit Frameworkia käytetään yleisimmin MSFconsolen kautta (Rapid7 a). Muilla kaupallisilla versioilla Metasploitia pystyy käyttämään myös web-käyttöliittymän kautta (Rapid7 c).

Metasploit Frameworkin tärkeänä ydinkomponenttina ovat erilaiset moduulit. Moduulit ovat ohjelmistokomponentteja, jotka määrittävät Metasploitilla tehtävät toimenpiteet eli taskit. Metasploit Frameworkin moduulien tyypit ovat Auxiliary, Exploit, Payload, Post-Exploitation ja NOP generator. Auxiliary-moduulit eivät suorita hyökkäyskoodia, vaan ovat esimerkiksi haavoittuvuuksien skannereita. Exploit-moduulit suorittavat tiettyyn haavoittuvuuteen kohdennettuja komentoja. Payload-moduulit suorittavat haitallista koodia sen jälkeen, kun Exploit-moduulien avulla on päästy tunkeutumaan järjestelmään. Payload voi olla komentoikkuna tai Meterpreter. Meterpreter on kehittynyt payload, jolla voi luoda dynaamisesti uusia ominaisuuksia tarpeen mukaan. Post-Exploitation-moduuleilla kerätään lisää tietoa järjestelmästä. Niillä voidaan esimerkiksi listata kohteessa olevia palveluita ja ohjelmistoja. NOP generator -moduulit tuottavat satunnaisia tavuja, joita IDS- ja IPS-järjestelmät eivät

normaalisti tunnista. (Rapid7 a.) NOP on lyhenne sanasta no operation payload (Rapid7 b). Satunnaiset tavut eivät siis ole suoritettavaa koodia.

Toinen Metasploitin ydinkomponenteista on datastore. Kyseessä on taulu, johon nimetty arvoja, joilla voidaan määritellä, miten Metasploitin eri komponentit toimivat. Datastoreja on sekä globaaleja että moduulikohtaisia. (Rapid7 a.)

Metasploitissa voi koota kohteet ja tiedot omiksi kokonaisuuksiksi Workspaceihin. Esimerkiksi yrityksen aliverkot ja osastot voidaan jakaa omiin Workspaceihin. Workspaceihin voidaan tuoda, muokata ja niistä voidaan viedä dataa. Nykyisen kohteen tila raportoidaan Workspacesta automaattisesti. (Rapid7 d.) Myös tagien antaminen kohdekoneille on mahdollista (Rapid7 e).

Metasploit Frameworkissa on tuki PostgreSQL-tietokannalle. Tietokannan käyttö Metasploit Frameworkissa ei ole välttämätöntä, mutta tietokantaan saa talteen kohdekoneiden tiedot, hyökkäysten tulokset ja kaapatut tiedot. (Rapid7 f.) Metasploitia voi myös käyttää etänä ja sen voi asettaa toimimaan palveluna (Rapid7 g).

Muita Metasploitin käyttöön liittyviä käsitteitä ovat Bind Shell Payload, Discovery Scan, Exploit, Listener, Payload, Project, Reverse Shell Payload, Shellcode, Shell ja Vulnerability. Project tarkoittaa Metasploit Pro -versiossa samaa kuin Workspace Metasploit Frameworkissa. Discovery Scanin avulla etsitään kohdekoneet tietoverkosta. Shell tarkoittaa komentoikkunaa. Vulnerability on haavoittuvuus eli tietoturvaheikkous tai -vika. Exploit on ohjelma, joka antaa hyökkääjälle pääsyn kohdekoneelle hyödyntämällä siinä olevaa haavoittuvuutta. Payload on tunkeutumisen jälkeen kohdekoneella suoritettava koodi. Shellcode on kokoelma käskyjä, joita käytetään exploitin payloadina. Listener odottaa tulevaa yhteyttä hyökkäävällä tai kohteena olevalla koneella ja hallitsee yhteydenpidon koneiden välillä. Bind Shell Payload ottaa hyökkääjän koneelta yhteyttä kohdekoneen Listeneriin ja antaa hyökkääjälle komentoikkunan käyttöön. Reverse Shell Payload ottaa kohdekoneelta yhteyttä hyökkääjän koneella olevaan Listeneriin ja antaa hyökkääjälle komentoikkunan käyttöön. (Rapid7 h.)

4.1.2 Burp Suite

Burp Suite on web-sovelluksien turvallisuuden testaamiseen erikoistunut ohjelmisto (PortSwigger 2017a), johon on integroitu useita työkaluja (PortSwigger 2017b). Burp Suite on suljetun lähdekoodin ohjelmisto (PortSwigger 2017c), ja siitä on olemassa sekä ilmai-

nen että maksullinen Professional-versio (PortSwigger 2017d). Burp Suite on PortSwigge-
rin kehittämä ja toimii Windows-, Mac- ja Linux-käyttöjärjestelmissä (PortSwigger 2017e).
Lisäksi Burp Suite vaatii toimiakseen 64-bittisen Java Runtime Environmentin version 1.7
tai uudemman (PortSwigger 2017c).

Burpin työkaluja ovat Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, De-
coder, Comparer ja Extender. Näiden työkalujen välillä voi lähettää pyyntöjä, eli ne toimi-
vat tehokkaasti yhdessä. Target-työkalu mahdollistaa haavoittuvuuksien testaamisen ja
antaa tietoa kohdesovelluksista. Proxy-työkalu toimii välimiehenä sovelluksen ja selaimen
välillä ja sen avulla on mahdollista tarkastella ja muokata sekä tulevaa että lähtevää liiken-
nettä. Spider-työkalu etsii web-sovelluksesta sisältöä ja toiminnallisuutta. Scanner-työkalu
kuuluu vain Pro-versioon, ja se pystyy löytämään automaattisesti useita erityyppisiä haa-
voittuvuuksia. (PortSwigger 2017f.) Intruder-työkalun avulla voi suorittaa automatisoituja
kustomoituja hyökkäyksiä web-sovelluksiin (PortSwigger 2017d). Repeater-työkalun
avulla voidaan manipuloida yksittäisiä HTTP-pyyntöjä ja analysoida, miten web-sovellus
vastaa niihin. Sequencer-työkalu pystyy arvioimaan, kuinka satunnaisia ei-ennustettaviksi
tarkoitettut selaintiedot ovat. Decoder-työkalu pystyy koodaamaan tietoa toiseen muotoon
(encode) ja purkamaan koodatun tiedon alkuperäiseksi (decode). Comparer-työkalu tekee
visuaalisen vertailun minkä tahansa kahden tiedon välillä. Extender-työkalu mahdollistaa
laajennusten käytön Burpissa oman tai kolmannen osapuolen kirjoittaman koodin avulla.
(PortSwigger 2017f.)

5 Tunkeutumiskohteet

Tässä luvussa verrataan tunkeutumiskohteiksi sopivia järjestelmiä keskenään. Näistä koh-
teista otetaan yksi mukaan varsinaiseen toteutusvaiheeseen.

5.1 Metasploitable

Metasploitable 2 on tarkoituksella haavoittuvaksi tehty Ubuntu Linux -virtuaalikone. Metas-
ploitablen avulla voidaan havainnollistaa yleisiä haavoittuvuuksia ja testata tietoturvatyö-
kaluja. Metasploitable 2:een tunkeutumista varten on olemassa Metasploitin kehittäjän
Rapid7:n kirjoittama ohje nimeltään Metasploitable 2 Exploitability Guide. Oppaassa on
listattu useita turvallisuusvirheitä, joita Metasploitable 2 sisältää. (Rapid7 i.) Metasploitable
versio 2 on ladattavissa SourceForgesta ja se on lisensoitu sekä BSD- että GPLv2-lisens-
seillä (rapid7user 2015).

Metasploitablesta on myös julkaistu versio 3. Metasploitable 3 kasataan käyttämällä Packeria, Vagrantia, Vagrant Reload Pluginia ja VirtualBoxia. Kasaamisen voi suorittaa manuaalisesti, mutta siihen on myös tarjolla automaattiset skriptit: Windowsille PowerShell-muodossa ja Linuxille sh-muodossa. Sen sijaan Metasploitable 2 on valmis virtuaalikone. (Chen 2016.) Build-skripteistä käy ilmi, että Metasploitable 3 on Windows 2008 -palvelin. Metasploitable 3:sta on julkaistu myös Ubuntu Linux 14.04 -versio Vagrant Cloudiin (Sliim 2017). Metasploitable 3 on lisensoitu BSD-3-clause-lisenssillä (jbarnett-r7 2016).

Metasploitable 3 on tarkoitettu eritasoisille käyttäjille. Jokaiseen haavoittuvuuteen ei pysty tunketumaan Metasploitin avulla. Sen sijaan Metasploitable 2 on suunniteltu nimenomaan Metasploitia varten. Ympäristömuuttujan `MS3_DIFFICULTY` avulla voidaan säätää Metasploitable 3:n vaikeustasoa. Aidon yritystiedon sijaan Metasploitable 3 sisältää kuvia, joita on tarkoitus kerätä lipunryöstön tapaan. Kuvien dataa on voitu tarkoituksenmukaisesti monimutkaistaa, niihin on voitu asettaa tiukat käyttöoikeudet tai tiedoston ominaisuudet. Kuvat voivat myös olla sulautettuja. Kuvien löytäminen saattaa siis vaatia takaisinmallinnustaitoja. Lippujen ryöstössä harjoitellaan tunketumisen jälkeistä eli Post-Exploitation -vaihetta. Normaalisti tunkeutumistestauksessa hyödynnetään yhdeltä koneelta löytyneitä tietoja seuraavaan koneeseen tunkeutumista varten, joten Rapid7 aikoo tulevaisuudessa rakentaa kokonaisen verkon haavoittuvia koneita. (Chen 2016.)

5.2 Muut peruspalvelimet

Tässä luvussa kuvataan lyhyesti muita yleisesti käytössä olevia palvelimia. Yritysympäristössä tämä tarkoittaa Windows- ja Linux-palvelimia (Barker 2014).

5.2.1 Windows-palvelin

Barkerin (2014) mukaan Microsoftin kehittämällä Windows-käyttöjärjestelmillä ja -ohjelmistoilla oli vuonna 2014 75 %:n markkinaosuus ja täten ne on koettu standardivaihtoehdoksi yritysympäristössä. Windows-palvelinten vanhempien versioiden käyttäminen on tietoturvariski, koska elinkaaren lopun saavuttaneisiin palvelimiin ei enää tule päivityksiä. Windowsin lisenssit ovat myös kalliita. (Barker 2014.) Windows-palvelimista on saatavilla versioita kokeilukäyttöön tietyksi ajaksi Microsoft Evaluation Centeristä. Uusin versio on Windows Server 2016. (Microsoft 2017.) Windows koetaan käyttäjäystävällisemmäksi kuin Linux, koska Microsoft on tuottanut ohjelmistoja melkein 30 vuotta ja käyttäjät ovat omaksuneet tietyt asiat (Walker & Shepherd 2017a). Useimmat ohjelmat ovat yhteensopivia Windowsin kanssa (Walker & Shepherd 2017b), ja valitettavasti tämä pätee myös haittaohjelmiin (Walker & Shepherd 2017c).

5.2.2 Linux-palvelin

Yli 95 % supertietokoneista käyttää Linuxia, joten se on nopein ja vähiten resursseja käyttävä käyttöjärjestelmä. Linux-jakelut tarjoavat Microsoftin ohjelmia vastaavia vapaita ohjelmia, mutta Windows on tunnetumpi ja täten ylläpitäjien löytäminen Linux-palvelimille olla vaikeampaa ja kalliimpaa. (Barker 2014.) Linux koetaan turvallisemmaksi kuin Windows, koska Linuxissa käyttöoikeuksia rajoitetaan hyvin ja virheitä on mahdollista löytää avoimen lähdekoodin ansiosta (Walker & Shepherd 2017c).

Stroudin (2017) mukaan Canonicalin kehittämä Ubuntu Server voittaa vertailussa kaikki muut Linux-palvelinjakelut. Ubuntun uusin pitkään tuettu versio on 16.04 LTS (Long Term Support). LTS-versiot julkaistaan joka toinen vuosi ja niille tarjotaan tukea viideksi vuodeksi. (Stroud 2017.) Palvelimissa on tavallista käyttää pitkään tuettuja versioita. Ubuntu Server on vapaasti ladattavissa ja siitä on myös olemassa yhdeksän kuukauden ajan tuettuja versioita, joista uusin on 17.10 (Canonical 2017).

6 Penetration Testing -harjoitusympäristön toteutus

Tässä luvussa kuvataan Penetration Testing -harjoitusympäristön toteuttaminen. Ympäristö on virtuaalinen, toimii vapaiden ohjelmien avulla ja se on lisensoitu GPLv2-lisenssillä, joka on luettavissa GitHubista osoitteesta <https://github.com/tonijaaskelainen/beginnerpentest/blob/master/LICENSE>. Ensin ympäristö pystytetään ja sen jälkeen testataan ympäristöön tunkeutumista. Ympäristö pystytetään Vagrantin avulla ja Vagrantin provideriksi otetaan käyttöön VirtualBox. Hyökkäävä kone suorittaa Kali Linuxia ja tunkeutumistyökaluna toimii Metasploit Framework. Tunkeutumiskohteena toimii Metasploitable 2.

Projektin lisenssiksi valittiin GPLv2, koska työn tekijä halusi mahdollistaa työn jakamisen vapaasti muille käyttäjille ja kehittäjille. Ympäristöstä tehtiin virtuaalinen, koska haluttiin, että ympäristö ei ole sidonnainen yhteen laitteeseen, vaan että tunkeutumistestausta voisi harjoitella mahdollisimman monella koneella. Vagrant valittiin ympäristön perustaksi, koska se on vapaasti saatavilla, se toimii useilla eri alustoilla ja Vagrantfilet mahdollistavat koneiden määrittelyjen jakamisen kätevästi. VirtualBox valittiin Vagrantin provideriksi, koska Vagrantin käyttö vaatii jonkin providerin, VirtualBox on vapaasti saatavilla, toimii useilla eri alustoilla ja koska Vagrant tukee VirtualBoxia ilman plugineja. Kali Linux valittiin hyökkäyskoneeksi, koska se on suunniteltu tunkeutumistestausta varten ja täten siinä on paljon tarpeellisia työkaluja, mukaan lukien Metasploit Framework ja Burp Suite. Metas-

ploit Framework valittiin tunkeutumistyökaluksi, koska se on monipuolinen tunkeutumistestaukseen suunniteltu vapaa ohjelmisto. Kohteeksi valittiin Metasploitable 2, koska se on suunniteltu haavoittuvaksi ja on täten ideaalinen tunkeutumistestauksen harjoittelua varten.

6.1 Virtuaaliympäristön pystytys

Ympäristö pystytettiin SAMSUNG NPC300E5C -kannettavaan, jossa suoritettiin 64-bittistä Ubuntu Linux 16.04.3 LTS -käyttöjärjestelmää. Käytetty VirtualBoxin versio oli 5.1.30 ja Vagrantin versio 2.0.0. Lisäksi käytetty virtuaali-image oli Kali Linux light 64 bit VBox versio 2017.2 ja käytetty virtuaalikiintolevy sisälsi asennettuna Metasploitable 2.0.0 -järjestelmän, jonka palveluja suoritetaan tarkemmin sanoen 32-bittisessä Ubuntu Linux 8.04 Server -käyttöjärjestelmässä. Tämä näkyy kuvassa 2. Vasta myöhemmin kävi ilmi, että Kalin kevyt versio ei sisältänyt tarvittavia työkaluja, joten ne jouduttiin asentamaan erikseen.

```
msfadmin@metasploitable:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 8.04
Release:       8.04
Codename:      hardy
msfadmin@metasploitable:~$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
msfadmin@metasploitable:~$ _
```

Kuva 2. Metasploitablen järjestelmätiedot

6.1.1 Lataaminen

Järjestelmänvalvojan tilille kirjauduttiin ja ladattiin VirtualBox komennolla

```
wget http://download.virtualbox.org/virtualbox/5.1.30/virtualbox-5.1_5.1.30-118389~Ubuntu~xenial_amd64.deb
```

ja Vagrant komennolla

```
wget https://releases.hashicorp.com/vagrant/2.0.0/vagrant_2.0.0_x86_64.deb
```

Lisäksi tiedostojen eheyden varmistamiseksi ladattiin SHA256SUMS-tiedosto VirtualBoxille komennolla

```
wget http://download.virtualbox.org/virtualbox/5.1.30/SHA256SUMS
```

ja Vagrantille komennolla

```
wget https://releases.hashicorp.com/vagrant/2.0.0/va-  
grant_2.0.0_SHA256SUMS
```

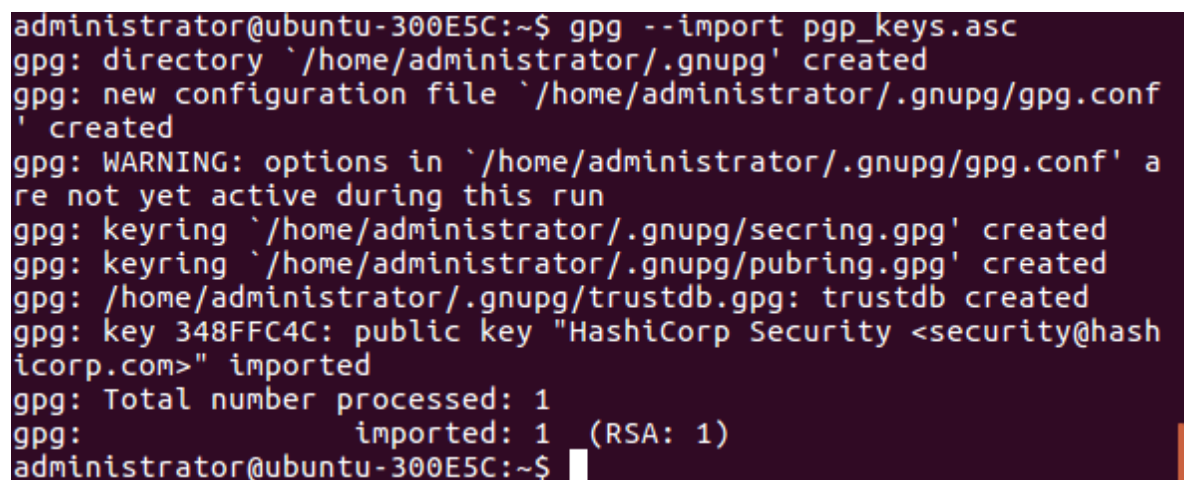
Hashicorp oli allekirjoittanut Vagrantin SHA256SUMS-tiedoston, joten seuraavaksi ladattiin allekirjoitustiedosto komennolla

```
wget https://releases.hashicorp.com/vagrant/2.0.0/va-  
grant_2.0.0_SHA256SUMS.sig
```

ja allekirjoitukseen käytetty julkinen avain komennolla

```
wget https://keybase.io/hashicorp/pgp_keys.asc
```

Tämän jälkeen avain importattiin kannettavalle. Kuvasta 3 käy ilmi "imported: 1", joten toimenpide onnistui.



```
administrator@ubuntu-300E5C:~$ gpg --import pgp_keys.asc  
gpg: directory `/home/administrator/.gnupg' created  
gpg: new configuration file `/home/administrator/.gnupg/gpg.conf'  
      ' created  
gpg: WARNING: options in `/home/administrator/.gnupg/gpg.conf' a  
re not yet active during this run  
gpg: keyring `/home/administrator/.gnupg/secring.gpg' created  
gpg: keyring `/home/administrator/.gnupg/pubring.gpg' created  
gpg: /home/administrator/.gnupg/trustdb.gpg: trustdb created  
gpg: key 348FFC4C: public key "HashiCorp Security <security@hash  
icorp.com>" imported  
gpg: Total number processed: 1  
gpg:             imported: 1 (RSA: 1)  
administrator@ubuntu-300E5C:~$
```

Kuva 3. Avaimen importtaus

Importtauksen jälkeen pystyttiin varmistamaan tarkistussummatiedoston eheys. Kuvassa 4 näkyy "Good signature", joten toimenpide onnistui.

```

administrator@ubuntu-300E5C:~$ gpg --verify vagrant_2.0.0_SHA256
SUMS.sig vagrant_2.0.0_SHA256SUMS
gpg: Signature made to 7. syyskuuta 2017 17.34.00 EEST using RS
A key ID 348FFC4C
gpg: Good signature from "HashiCorp Security <security@hashicorp
.com>"
gpg: WARNING: This key is not certified with a trusted signature
!
gpg:          There is no indication that the signature belongs
to the owner.
Primary key fingerprint: 91A6 E7F8 5D05 C656 30BE F189 5185 2D8
7 348F FC4C
administrator@ubuntu-300E5C:~$

```

Kuva 4. Tarkistussummatiedoston eheyden varmistaminen

Nyt itse asennuspaketin eheys voitiin varmistaa. Kuvasta 5 käy ilmi, että ladattu asennus-
paketti vagrant_2.0.0_x86_64.deb on OK.

```

administrator@ubuntu-300E5C:~$ shasum -a 256 -c vagrant_2.0.0_SH
A256SUMS
shasum: hashicorp-vagrant-2.0.0-1-x86_64.pkg.tar.xz:
hashicorp-vagrant-2.0.0-1-x86_64.pkg.tar.xz: FAILED open or read
shasum: vagrant_2.0.0_i686.deb: No such file or directory
vagrant_2.0.0_i686.deb: FAILED open or read
shasum: vagrant_2.0.0_i686.msi: No such file or directory
vagrant_2.0.0_i686.msi: FAILED open or read
shasum: vagrant_2.0.0_i686.rpm: No such file or directory
vagrant_2.0.0_i686.rpm: FAILED open or read
vagrant_2.0.0_x86_64.deb: OK
shasum: vagrant_2.0.0_x86_64.dmg: No such file or directory
vagrant_2.0.0_x86_64.dmg: FAILED open or read
shasum: vagrant_2.0.0_x86_64.msi: No such file or directory
vagrant_2.0.0_x86_64.msi: FAILED open or read
shasum: vagrant_2.0.0_x86_64.rpm: No such file or directory
vagrant_2.0.0_x86_64.rpm: FAILED open or read
shasum: WARNING: 7 listed files could not be read
administrator@ubuntu-300E5C:~$

```

Kuva 5. Asennuspaketin eheyden varmistaminen

VirtualBoxin tiedostot varmistettiin edellä mainitulla tavalla ilman allekirjoituksia, koska
niitä ei ollut saatavilla.

Ympäristöön tarvittiin vielä sekä hyökkäävä kone että kohteena oleva kone. Hyökkääväksi
koneeksi valittiin Offensive Securityn "Kali Linux light 64 bit VBox versio 2017.2". Tämä on
valmis VirtualBox-image, joka on lisäksi normaalia Kali Linuxia kevyempi.

Kali Linux ladattiin komennolla

```
wget https://images.offensive-security.com/virtual-images/kali-  
linux-light-2017.2-vbox-amd64.ova
```

ja selvitettiin tarkistussumma. Kuvassa 6 näkyvä tarkistussumma vastaa Offensive Securityn ilmoittamaa tarkistussummaa (Offensive Security 2017)), joten Kalin image on eheä.

```
administrator@ubuntu-300E5C:~$ sha256sum kali-linux-light-2017.2-  
vbox-amd64.ova  
a93dadeceec05da64b4aecb135ce84aaf2daecaa8d3b5577a4890c120a3a67fd0  
kali-linux-light-2017.2-vbox-amd64.ova  
administrator@ubuntu-300E5C:~$
```

Kuva 6. Kalin tarkistussumma

Metasploitablesta käytetty versio on 2.0.0 ja se ladattiin komennolla

```
wget https://downloads.sourceforge.net/project/metasploita-  
ble/Metasploitable2/metasploitable-linux-  
2.0.0.zip?r=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fmetasploit-  
able%2Ffiles%2FMetasploitable2%2F&ts=1509640578&use_mirror=netix
```

Tiedosto latautui nopeasti mirrorin kautta, mutta tiedostonimessä oli mirrorin takia ylimääräistä ja latauskomento ei pysähtynyt normaalisti, joten kun lataus oli valmis, niin `wget` piti pysäyttää näppäinyhdistelmällä Ctrl+C ja tiedosto nimettiin uudelleen `mv`-komennolla. Tämän jälkeen Metasploitable 2:n zip-paketti purettiin ja kuvasta 7 näkyy, että paketissa oli useampi eri tiedosto. Näistä `.vmdk`-muotoinen tiedosto on virtuaalinen kiintolevy.

```
administrator@ubuntu-300E5C:~$ unzip metasploitable-linux-2.0.0.  
zip  
Archive:  metasploitable-linux-2.0.0.zip  
creating: Metasploitable2-Linux/  
inflating: Metasploitable2-Linux/Metasploitable.nvram  
inflating: Metasploitable2-Linux/Metasploitable.vmdk  
inflating: Metasploitable2-Linux/Metasploitable.vmsd  
inflating: Metasploitable2-Linux/Metasploitable.vmx  
inflating: Metasploitable2-Linux/Metasploitable.vmx  
administrator@ubuntu-300E5C:~$
```

Kuva 7. zip-paketin purkaminen

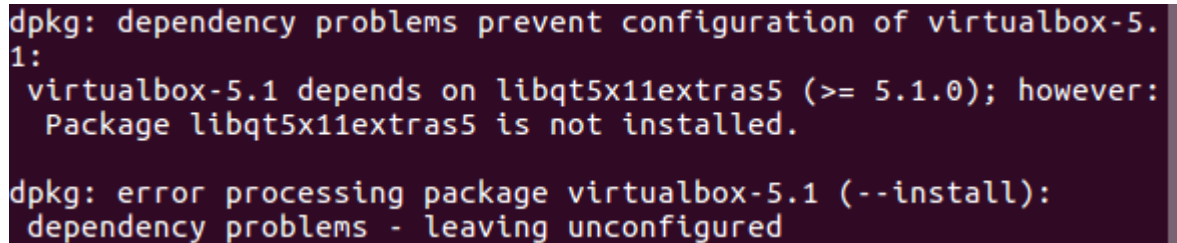
6.1.2 Asentaminen

VirtualBox asennettiin komennolla

```
sudo dpkg -i virtualbox-5.1_5.1.30-118389~Ubuntu~xenial_amd64.deb
```

Kuvasta 8 näkyy, että tämä asennus vaati kuitenkin toimiakseen paketin `libqt5x11extras5`, joten saatavilla olevien pakettien lista päivitettiin ja kyseinen paketti asennettiin komennolla

```
sudo apt-get update && sudo apt-get install libqt5x11extras5
```

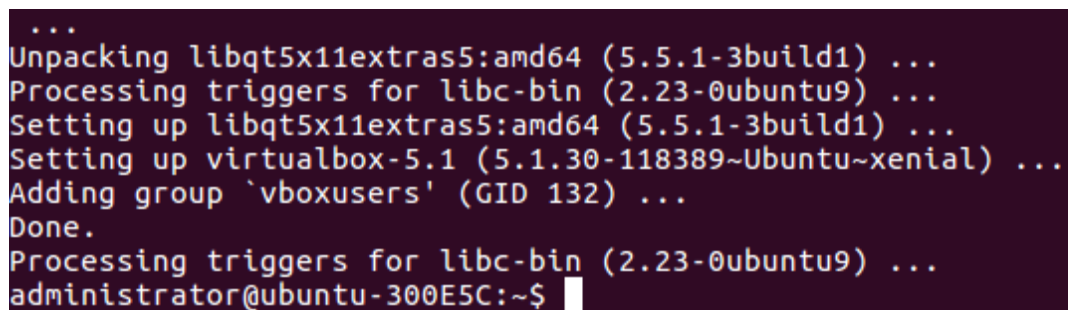


```
dpkg: dependency problems prevent configuration of virtualbox-5.1:
virtualbox-5.1 depends on libqt5x11extras5 (>= 5.1.0); however:
Package libqt5x11extras5 is not installed.

dpkg: error processing package virtualbox-5.1 (--install):
dependency problems - leaving unconfigured
```

Kuva 8. VirtualBoxin riippuvuusvirhe

Kuvasta 9 näkyy, että paketin onnistuneen asennuksen jälkeen VirtualBoxin asennus hoitui itsestään.



```
...
Unpacking libqt5x11extras5:amd64 (5.5.1-3build1) ...
Processing triggers for libc-bin (2.23-0ubuntu9) ...
Setting up libqt5x11extras5:amd64 (5.5.1-3build1) ...
Setting up virtualbox-5.1 (5.1.30-118389~Ubuntu~xenial) ...
Adding group `vboxusers' (GID 132) ...
Done.
Processing triggers for libc-bin (2.23-0ubuntu9) ...
administrator@ubuntu-300E5C:~$
```

Kuva 9. VirtualBoxin asennus onnistui

Vagrant asennettiin komennolla

```
sudo dpkg -i vagrant_2.0.0_x86_64.deb
```

Tämä asennus onnistui suoraan ilman ongelmia. Koneita ei tarvinnut varsinaisesti asentaa, koska käyttöjärjestelmät sisältyivät virtuaali-imageen ja -kiintolevyyn.

6.1.3 Määrittely

Järjestelmänvalvojan tilillä ladatut virtuaali-image ja -kiintolevy siirrettiin peruskäyttäjän puolelle ja tiedostojen omistajuus annettiin peruskäyttäjälle. Kuvasta 10 näkyy, että kyseessä oli Kalin .ova-muotoinen image ja Metasploitablen .vmdk-muotoinen virtuaalinen kiintolevy.

```
administrator@ubuntu-300E5C:~$ sudo mv *.ova Metasploitable2-Linux/*.vmdk /home/toni/
[sudo] password for administrator:
administrator@ubuntu-300E5C:~$ sudo chown toni:toni /home/toni/*.ova /home/toni/*.vmdk
administrator@ubuntu-300E5C:~$
```

Kuva 10. Tiedostojen siirto ja omistajuuden muokkaus

Järjestelmänvalvojan tililtä poistuttiin ja toimenpiteitä jatkettiin peruskäyttäjällä. Peruskäyttäjällä siirryttiin repositoryn hakemistoon komennolla

```
cd beginnerpentest/
```

ja asetettiin hakemisto Vagrantin käyttöä varten komennolla

```
vagrant init
```

Tämä loi hakemistoon Vagrantfilen. Myöhemmin oli tarkoitus muokata Vagrantfilea ja testata, miten Vagrantfile toimii boxien kanssa paikallisesti, ennen kuin ne lisätään Vagrant Cloudiin. Koneet oli ensin lisättävä VirtualBoxiin. Kun koneet ovat VirtualBoxissa ja niihin on tehty Vagrantin dokumentaation suositusten mukaiset määrittelyt, niin ne voi paketoida boxeiksi ja boxit voi julkaista Vagrant Cloudissa.

Kalin kohdalla kyseessä oli valmis virtuaali-image, joten sen pystyi importtaamaan suoraan VirtualBoxiin. Tämä käy ilmi kuvasta 11.

```
toni@ubuntu-300E5C:~/beginnerpentest$ VBoxManage import ../kali-linux-light-2017.2-vbox-amd64.ova
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Interpreting /home/toni/kali-linux-light-2017.2-vbox-amd64.ova..
OK.
```

Kuva 11. Kalin importtaus

Metasploitablen kohdalla piti luoda uusi virtuaalikone ja liittää siihen .vmdk-muotoinen virtuaalinen kiintolevy, johon Metasploitable 2 oli asennettu. Virtuaalikone luodaan kuvassa 12 ja kuvassa 13 luodaan SATA controller ja liitetään siihen Metasploitable 2:n kiintolevy.

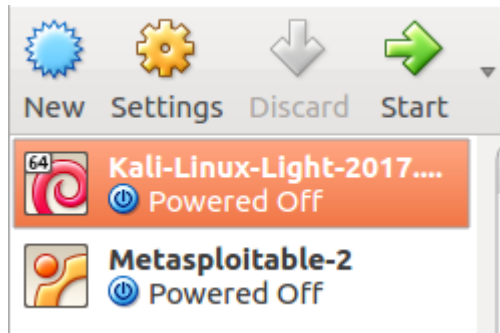

```
toni@ubuntu-300E5C:~/beginnerpentest$ VBoxManage createvm --name
"Metasploitable-2" --ostype "Ubuntu" --register
Virtual machine 'Metasploitable-2' is created and registered.
UUID: b52a8ef2-713a-4fd9-9e9e-7d17b081b351
Settings file: '/home/toni/VirtualBox VMs/Metasploitable-2/Metas
ploitable-2.vbox'
```

Kuva 12. Metasploitable 2 -koneen luonti

```
toni@ubuntu-300E5C:~/beginnerpentest$ VBoxManage storagectl Meta
sploitable-2 --name SATA --add sata --controller IntelAHCI
toni@ubuntu-300E5C:~/beginnerpentest$ VBoxManage storageattach M
etasploitable-2 --storagectl SATA --port 0 --device 0 --type hdd
--medium ../Metasploitable.vmdk
toni@ubuntu-300E5C:~/beginnerpentest$
```

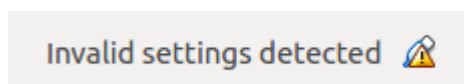
Kuva 13. Kiintolevyn lisääminen

VirtualBox käynnistettiin ja kuvassa 14 koneet näkyvät VirtualBoxin GUI:ssa.



Kuva 14. Virtuaalikoneet

Kali-imagen asennuksen jälkeen VirtualBox ilmoitti virheestä, joka näkyy kuvassa 15. Tuo virhe liittyi siihen, että imagessa oli aktivoitu oletuksena tuki USB 2.0:lle ja Pointing Deviceksi oli asetettu "USB Tablet". USB-tuki voitiin poistaa käytöstä kohdasta Settings... -> USB -> "Enable USB Controller", koska Vagrantin dokumentaatio suosittelee ylimääräisten lisälaitteiden poistamista käytöstä (HashiCorp c). Tämän perusteella Kalista poistettiin myös Audio-tuki kohdasta Settings... -> Audio -> "Enable Audio". Tuen poistamisen jälkeen Pointing Device piti vaihtaa kohdasta Settings... -> System -> Motherboard -> "Pointing Device:" -> "PS/2 Mouse".



Kuva 15. Kali-imagen virheelliset asetukset

Metasploitablen kohdalla asetuksista muutettiin kohdasta Settings... -> System -> Motherboard -> "Base Memory:" arvoksi 512 MB ja alempaa kohdasta "Boot Order" jätettiin aktiiviseksi vain "Hard Disk" ja vielä alempaa kohdasta "Extended Features:" asetettiin "Hardware Clock in UTC Time". Kohdasta Settings... -> Display -> "Video Memory:" asetettiin arvo 12 MB. Audio poistettiin käytöstä myös Metasploitablesta.

Vagrantin dokumentaatio suosittelee myös asentamaan koneisiin VirtualBoxin Guest Additionsit (HashiCorp d) ennen paketointia boxeiksi, joten ne yritettiin asentaa seuraavaksi. Tätä varten Metasploitableen luontiin optinen asema kohdasta Settings... -> Storage ja Storage Treen alapuolelta +-merkin kohdalta "Adds new storage controller." -> "Add IDE Controller" ja sen jälkeen kohdasta "Controller: IDE" painettiin +-merkistä "Adds optical drive." ja valittiin "Choose disk". VirtualBoxin mukana tullut tiedosto löytyi polusta `/usr/share/virtualbox/VBoxGuestAdditions.iso`. Lisäksi kohdasta "Controller SATA" Attributes-osiosta asetettiin "Port Count:" arvoon 1.

Koneet käynnistettiin VirtualBoxin ikkunasta valitsemalla Start. Kalissa Guest Additionseista oli jo asennettu aiempi versio, mutta Metasploitable 2:ssa niitä ei ollut. Kuvassa 16 näkyy, että komennot eivät tulostaneet mitään.

```
msfadmin@metasploitable:~$ dpkg -l | grep virtualbox-guest
msfadmin@metasploitable:~$ lsmod | grep vboxguest
msfadmin@metasploitable:~$ _
```

Kuva 16. Ei Guest Additionseja

Kuvassa 17 näkyy, että Guest Additionsit yritettiin asentaa Metasploitableen, mutta asennuksessa tuli virhe.

```
msfadmin@metasploitable:~$ sudo mount /dev/sr0 /media/cdrom
mount: block device /dev/scd0 is write-protected, mounting read-only
msfadmin@metasploitable:~$ sudo sh /media/cdrom/VBoxLinuxAdditions.run
Verifying archive integrity... All good.
Uncompressing VirtualBox 5.1.30 Guest Additions for Linux.....
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
vboxadd.sh: Starting the VirtualBox Guest Additions.
Failed to set up service vboxadd, please check the log file
/var/log/VBoxGuestAdditions.log for details.
msfadmin@metasploitable:~$ _
```

Kuva 17. Guest Additions -virhe

Kuvassa 18 näkyy virhelokien sisällöt. Asennus ei löytänyt Linux-ytimen lähdetiedostoja, jotka olisi pitänyt ladata. Metasploitable on yhdistetty oletuksena NAT-verkkoon, joten lataaminen on mahdollista (ping 8.8.8.8 toimii). Vasta Vagrantin boxit yritetään asettaa Host-only-verkkoon, josta ei ole yhteyttä Internetiin.

```
vboxadd.sh: Starting the VirtualBox Guest Additions.
Failed to set up service vboxadd, please check the log file
/var/log/VBoxGuestAdditions.log for details.
msfadmin@metasploitable:~$ less /var/log/VBoxGuestAdditions.log

vboxadd.sh: failed: Look at /var/log/vboxadd-install.log to find out what went w
rong.
vboxadd.sh: failed: Look at /var/log/vboxadd-install.log to find out what went w
rong.
vboxadd.sh: failed: modprobe vboxguest failed.
msfadmin@metasploitable:~$ less /var/log/vboxadd-install.log
/tmp/vbox.0/Makefile.include.header:112: *** Error: unable to find the sources o
f your current Linux kernel. Specify KERN_DIR=<directory> and run Make again. S
top.
Creating user for the Guest Additions.
Creating udev rule for the Guest Additions kernel module.
```

Kuva 18. Virhelokien sisältö

Pakettien lista yritettiin päivittää komennolla

```
sudo apt-get update
```

ja kuvasta 19 näkyy, että listan päivittämisessä oli ongelmia, eikä asennuspakettien lähdetä voitu varmistaa.

```
Err http://us.archive.ubuntu.com hardy-backports/main Sources
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy-backports/restricted Sources
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy-backports/universe Sources
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy-backports/multiverse Sources
404 Not Found [IP: 91.189.91.23 80]
Fetched 198B in 1s (118B/s)
msfadmin@metasploitable:~$ sudo apt-get install linux-headers-$(uname -r) build-essential dkms
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  linux-headers-2.6.24-16
The following NEW packages will be installed:
  build-essential dkms linux-headers-2.6.24-16 linux-headers-2.6.24-16-server
0 upgraded, 4 newly installed, 0 to remove and 139 not upgraded.
Need to get 8835kB of archives.
After this operation, 68.8MB of additional disk space will be used.
Do you want to continue [Y/n]?
WARNING: The following packages cannot be authenticated!
  build-essential linux-headers-2.6.24-16 linux-headers-2.6.24-16-server dkms
Install these packages without verification [y/N]? y
```

Kuva 19. Paketit Metasploitablelle

Kuvasta 20 näkyy, että paketit yritettiin kuitenkin asentaa, mutta niitä ei pystytty lataamaan. Seuraavaksi kokeiltiin ehdotettua komentoa

```
sudo apt-get install --fix-missing
```

joka toimi, mutta asentaminen päättyi silti samaan virheeseen kuin aiemminkin.

```
Err http://us.archive.ubuntu.com hardy/main build-essential 11.3ubuntu1
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy/main linux-headers-2.6.24-16 2.6.24-16.30
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy/main linux-headers-2.6.24-16-server 2.6.2
4-16.30
404 Not Found [IP: 91.189.91.23 80]
Err http://us.archive.ubuntu.com hardy-backports/universe dkms 2.0.20.4-0ubuntu1
~hardy1
404 Not Found [IP: 91.189.91.23 80]
Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/b/build-essential/
build-essential_11.3ubuntu1_i386.deb 404 Not Found [IP: 91.189.91.23 80]
Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/l/linux/linux-head
ers-2.6.24-16_2.6.24-16.30_all.deb 404 Not Found [IP: 91.189.91.23 80]
Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/main/l/linux/linux-head
ers-2.6.24-16-server_2.6.24-16.30_i386.deb 404 Not Found [IP: 91.189.91.23 80]
Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/d/dkms/dkms_2.
0.20.4-0ubuntu1~hardy1_all.deb 404 Not Found [IP: 91.189.91.23 80]
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis
sing?
msfadmin@metasploitable:~$ _
```

Kuva 20. Pakettien asennusvirhe

Tästä nousi epäily, että kyseiset repositoryt eivät enää toimineet, koska Metasploitablen käyttämältä Ubuntu-versiolta on loppunut tuki. Ubuntun dokumentaatiosta selvisi, miten End-Of-Life-järjestelmien pakettilistaus voidaan muuttaa (jorgemorais 2015). Tarvittava tiedosto avattiin komennolla

```
sudo nano /etc/apt/sources.list
```

ja tiedoston loppuun kirjoitettiin sisällöksi

```
# EOL
deb http://old-releases.ubuntu.com/ubuntu/ hardy main restricted
universe multiverse
deb http://old-releases.ubuntu.com/ubuntu/ hardy-updates main re-
stricted universe multiverse
deb http://old-releases.ubuntu.com/ubuntu/ hardy-security main re-
stricted universe multiverse
```

```
deb http://old-releases.ubuntu.com/ubuntu/ hardy-backports main  
restricted universe multiverse
```

Tämän jälkeen pakettien lista päivitettiin ja asennus onnistui jopa ilman autentikointivirheitä. Kuvassa 21 näkyy, kuinka Guest Additionsit asentuivat vihdoin. Lisäksi .iso-tiedosto poistettiin optisesta asemasta ja Metasploitablelle asetettiin VirtualBoxissa kohdassa Settings... -> General -> Advanced -> "Shared Clipboard:" arvoksi Bidirectional. Kone käynnistettiin uudelleen komennolla

```
sudo shutdown -r now
```

ja komento

```
lsmod | grep vboxguest
```

näytti, että kyseinen moduuli oli ladattu uudelleenkäynnistuksen jälkeen.

```
msfadmin@metasploitable:~$ sudo sh /media/cdrom/VBoxLinuxAdditions.run  
Verifying archive integrity... All good.  
Uncompressing VirtualBox 5.1.30 Guest Additions for Linux.....  
VirtualBox Guest Additions installer  
Copying additional installer modules ...  
Installing additional modules ...  
vboxadd.sh: Starting the VirtualBox Guest Additions.  
  
You may need to restart the Window System (or just restart the guest system)  
to enable the Guest Additions.  
  
msfadmin@metasploitable:~$ sudo umount /media/  
cdrom/ cdrom0/ floppy/ floppy0/  
msfadmin@metasploitable:~$ sudo umount /media/cdrom  
msfadmin@metasploitable:~$ _
```

Kuva 21. Guest Additionsien asennus

Kalissa tiedostoon `/etc/ssh/sshd_config` kirjoitettiin rivi `PermitRootLogin yes`, jotta Kalin oletuskäyttäjällä (root) pääsee kirjautumaan SSH:n kautta. Normaalisti tämä käytäntö ei todellakaan ole tietoturvan kannalta hyvä vaihtoehto.

Lisäksi riviltä `UseDNS no` poistettiin kommenttimerkki `#`. UseDNS-määrittely tehtiin myös Metasploitable 2:lle. Lisäksi Metasploitablen käyttäjälle piti asettaa mahdollisuus korottaa rootiksi ilman salasanaa. Metasploitablen oletuskäyttäjän msfadmin huomattiin kuuluvan admin-ryhmään, joten määrittely tiedostoon `/etc/sudoers` oli helppoa komennolla

```
sudo visudo
```

koska tiedosto sisälsi valmiiksi admin-ryhmää koskevan rivin. Kuvassa 22 näkyy, että riville lisättiin `NOPASSWD:`.

```
# Members of the admin group may gain root privileges
/admin ALL=(ALL) NOPASSWD: ALL

"/etc/sudoers.tmp" 23 lines, 480 characters written
msfadmin@metasploitable:~$ groups
msfadmin adm dialout cdrom floppy audio dip video plugdev fuse lpadmin admin sam
bashare
msfadmin@metasploitable:~$ _
```

Kuva 22. root-oikeudet ilman salasanaa

Suosituksena on luoda vagrant-käyttäjä ja antaa sille salasana vagrant, mutta molemmissa koneissa oli tunnetut käyttäjät/salasanat (msfadmin/msfadmin sekä root/toor), joten käyttäjän luonnille ei nähty tarvetta. Lisäksi rootin salasanan olisi hyvä olla vagrant. Salasana jätettiin kuitenkin oletuksiksi, eli Kalissa toor ja Metasploitablessa rootilla ei ole salasanaa. Näin rootiksi voi korottautua helposti ja nopeasti antamalla msfadmin-käyttäjällä komennon

```
sudo su -
```

kun sudo ei enää vaadi salasanaa eikä rootillakaan ole salasanaa. Tämä näkyy kuvassa 23.

```
msfadmin@metasploitable:~$ sudo su -
root@metasploitable:~# whoami
root
root@metasploitable:~# pwd
/root
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:~# exit
logout
msfadmin@metasploitable:~$ _
```

Kuva 23. rootiksi korottautuminen

Tunnistautumista varten Kalissa luontiin SSH-RSA-avainpari rootille. Metasploitablessa tämä avainpari oli jo olemassa. Avainparin generointi näkyy kuvassa 24.

```

root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:oz8nZdkTMiQ3DipV5A4MWPq2CI3QBnB/n8eryu3nhhA root@kali
The key's randomart image is:
+---[RSA 2048]---+
|+ .00 000|
|+.0 0..0 +|
|. + ..+..* .|
|. + ...E+ 0+ .|
|0 . 0. .S 0= .|
|. 0 ... 0+.0|
|. . . . +. .|
|. . 0+.+|
| . oo=B.|
+-----[SHA256]-----+
root@kali:~#

```

Kuva 24. Private-keyn generointi

Metasploitablessa tiedoston `~/.ssh/authorized_keys` oikeuksia piti muuttaa niin, että vain omistajalla on luku- ja kirjoitusoikeus. Tämä tapahtui komennolla

```
chmod 0600 ~/.ssh/authorized_keys
```

Vasta nyt huomattiin, ettei Kali Linuxin Light-versiossa ole kuin muutama työkalu ja kevyt graafinen käyttöliittymä (Xfce). Täyden Kali-version .iso-tiedoston lataaminen tai siihen kuuluvien kaikkien metapakettien asentaminen olisi ollut tässä vaiheessa turhaa, joten vain työn kannalta tarvittavat työkalut asennettiin. Asennus käy ilmi kuvasta 25.

```

root@kali:~# apt-get update && apt-get install metasploit-framework postgresql

```

Kuva 25. Pakettien lisäys Kalin Light-versioon

SSH-palvelin piti vielä asettaa käynnistymään koneen käynnistyessä. Kalissa tämä oli normaalisti estetty. Lakhanin (2016) kirjoittamien ohjeiden perusteella suoritettiin kuvassa 26 näkyvät komennot

```
root@kali:~# update-rc.d -f ssh remove
root@kali:~# update-rc.d -f ssh defaults
root@kali:~# update-rc.d -f ssh enable 2 3 4 5
root@kali:~#
```

Kuva 26. SSH-palvelimen käynnistysasetukset

Kali käynnistettiin uudelleen ja komento

```
service ssh status
```

näytti, että SSH-palvelin on vihdoinkin käynnissä.

Koneet sammutettiin ja valmiit virtuaalikoneet paketoitiin boxeiksi komennolla

```
vagrant package box --base VIRTUAALIKONEEN_NIMI --output BO-
XIN_NIMI.box
```

jossa `VIRTUAALIKONEEN_NIMI` on VirtualBoxiin määritellyn virtuaalikoneen nimi ja `BO-`
`XIN_NIMI` on boxille haluttu nimi.

Kun koneet oli paketoitu, niin koneet määriteltiin Vagrantfileen ja niihin viitattiin paikallisesti koneella olevien boxien nimillä. Näin koneita pystyi testaamaan ennen niiden julkaisua Vagrant Cloudiin. Vagrantfileen lisättiin vielä koneiden MAC-osoitteet (HashiCorp d), jotka sai VirtualBoxista selville seuraavasti: Settings... -> Network -> Adapter 1 -> Advanced -> "MAC Address:".

Koneet käynnistettiin komennolla

```
vagrant up KONEEN_NIMI
```

jossa `KONEEN_NIMI` oli Vagrantfileen määritelty koneen nimi kohdassa

```
config.vm.define
```

Vastaavasti koneisiin otettiin yhteys komennolla

```
vagrant ssh KONEEN_NIMI
```


Nämän jälkeen koneet sammutettiin komennolla

```
vagrant halt KONEEN_NIMI
```

Metasploitableen sai onnistuneesti yhteyden, mutta Kaliin ei, vaikka GUI:n kautta varmistettuna SSH-palvelin oli päällä. Vagrant ei jostain syystä saanut Host-only-verkkoa määritettyä Kaliin Vagrantfilen perusteella, vaikka vastaava tilanne onnistui Metasploitablelle. Kuvasta 27 näkyy, Metasploitablen verkkoasetukset. `10.0.2.15/24` on NAT-verkko, jonka Vagrant vaatii toimiakseen. `192.168.56.101/24` on Vagrantfileissa asetettu Host-only-verkko. Lisäksi kuvan terminaalista käy ilmi, että Metasploitableen on päästy sisään isäntäkäyttöjärjestelmän kautta.

```
msfadmin@metasploitable:~$ ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:36:26:90
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe36:2690/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:451 errors:0 dropped:0 overruns:0 frame:0
          TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53328 (52.0 KB)  TX bytes:46654 (45.5 KB)
          Base address:0xd010 Memory:f0000000-f0020000

eth1      Link encap:Ethernet  HWaddr 08:00:27:c3:c3:0c
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec3:c30c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3114 (3.0 KB)  TX bytes:3708 (3.6 KB)
          Base address:0xd040 Memory:f0820000-f0840000
```

Kuva 27. Metasploitablen verkkoasetukset

Host-only-verkon määrittely näytti aiheuttavan myös sen, ettei Kaliin päässyt sisään SSH:n kautta. Tämän takia Kali jouduttiin jättämään pelkästään Vagrantin oletusverkkoon `10.0.2.15/24` NAT-tilaan. VirtualBoxin Guest Additionsit päivitettiin Kalissa vastaamaan käytössä olevaa VirtualBox-versiota, mutta siitäkään ei ollut apua. Kyseisessä tapauksessa VirtualBoxin aiemmat Guest Additionsit piti ensin poistaa kirjoittamalla asennus-skriptin perään argumentiksi `uninstall`. Vagrantfilesta kommentoitiin pois kohta

```
kali_conf.vm.network "private_network", ip: "192.168.56.102"
```

Poiston jälkeen Kaliin sai yhteyden SSH:n kautta, joten boxit voitiin lisätä Vagrant Cloudiin. Vagrant Cloudiin luotu käyttäjäprofiili (<https://vagrantcloud.com> -> "Create an Ac-

count") on tonijaaskelainen eli boxit löytyvät osoitteesta <https://app.vagrantup.com/tonijaaskelainen/>. Koneet lisättiin painikkeesta "Create a new Vagrant box". Koneiden nimi ja lyhyt kuvaus kirjoitettiin niille varattuihin kohtiin ja jatkettiin painamalla "Create box". Koneille annettiin versio ja laajempi kuvaus ja painettiin "Create a version". Koneille lisättiin provideriksi virtualbox painikkeesta "Add a provider" ja File Hosting -kohtaan jätettiin "Upload to Vagrant Cloud" ja painettiin "Continue to upload", minkä jälkeen boxit lisättiin "Upload File -kohdasta. Kohdasta Versions valittiin luotu boxin versio ja painettiin Release... ja sitten "Release version", jotta boxit saatiin julkaistua.

Kun koneet olivat Vagrant Cloudissa, niin Vagrantfile voitiin päivittää hakemaan koneiden tiedot suoraan Vagrant Cloudista. Esimerkiksi Kalille kyseinen kohta oli

```
config.vm.define "kali" do |kali_conf|  
  kali_conf.vm.box = "tonijaaskelainen/kali"
```

Vagrantfile validoitiin komennolla

```
vagrant validate
```

Paikalliset koneiden määrittelyt tuhottiin Vagrantista komennolla

```
vagrant destroy KONEEN_NIMI
```

ja boxit poistettiin kokonaan Vagrantista komennolla

```
vagrant box remove BOXIN_NIMI.box
```

Lisäksi VirtualBoxissa olleet virtuaalikoneet, joista boxit luotiin, poistettiin.

Kun Vagrantfile oli määritelty hakemaan koneiden tiedot suoraan Vagrant Cloudista, niin niiden toiminta voitiin testata normaaliin tapaan käynnistämällä (`vagrant up KONEEN_NIMI`) ja niihin sai otettua yhteyden normaaliin tapaan (`vagrant ssh KONEEN_NIMI`). Koneet toimivat normaalisti, joten seuraavaksi voitiin aloittaa tunkeutumisen testaaminen.

6.2 Tunkeutumisen testaaminen

7 Pohdinta

Projektissa oli tarkoitus saada selville, miten tunkeutumistestaukseen soveltuva harjoitusympäristö toteutetaan ja millä välineillä ja kohteilla toteutus kannattaa suorittaa. Lisäksi tarkoitus oli tunkeutua toteutettuun järjestelmään. Projektissa saatiin selville toteutukseen sopivat välineet ja harjoitusympäristö saatiin toteutettua. Työssä päädyttiin käyttämään virtualisointia, Vagrantia ja vapaita ohjelmia, jotta ympäristö olisi helposti jaettavissa ja muiden käytettävissä. Ympäristö on julkisesti saatavilla, joten käytännössä kuka tahansa voi hyödyntää sitä ja harjoitella eettistä hakkerointia turvallisesti.

Ympäristöstä voisi jatkokehittää laajemman version, jossa olisi Linux-palvelimen lisäksi myös Windows-palvelin. Metasploitable 2:n voisi jättää aloittelijoita varten ja kokeneet voisivat tunketutua Linux- ja Windows-palvelimiin. Vaihtoehtoisesti myös Metasploitable 3:n voisi ottaa mukaan ja säätää sen vaikeustasoa tarpeen mukaan ympäristömuuttujan `MS3_DIFFICULTY` avulla. Työssä kävi vasta myöhään ilmi, ettei Kalin Light-versio ei sisältänyt kuin muutaman työkalun, joten Kaliin voisi päivittää tarpeen tullen tarvittavat työkalut ja GUI:n voisi poistaa tilan säästämiseksi. Kalissa normaalisti olevat työkalut löytyvät metapaketista `kali-linux-full`.

Projektissa opittiin pystyttämään tarvittava ympäristö vapailla ohjelmilla. Lisäksi työn hallinnointia varten opittiin käyttämään Gitiä ja GitHubia. Työn suurin haaste oli todella laaja teoriaosuus ja siihen liittyvien laadukkaiden lähteiden löytäminen. Työhön otettiin liian helposti viitteitä ohjelmistojen kotisivuilta ja dokumentaatiosta tieteellisten artikkelien sijaan. Toinen haaste oli saada Kali-kone toimimaan Vagrantissa oikein. Vagrant ei jostain syystä saanut Host-only-verkkoa määriteltyä Kaliin Vagrantfilen perusteella, vaikka vastaava tilanne onnistui Metasploitablelle. Tämän takia Kali jouduttiin jättämään pelkästään Vagrantin oletusverkkoon `10.0.2.15/24` NAT-tilaan.

Lähteet

aidanns 2013a. Vagrant Reload Provisioner. Luettavissa: <https://github.com/aidanns/vagrant-reload>. Luettu: 26.10.2017.

aidanns 2013b. MIT License. Luettavissa: <https://github.com/aidanns/vagrant-reload/blob/master/LICENSE.txt>. Luettu: 26.10.2017.

Barker, D. 2014. Windows vs Linux: Which OS is best for your business? Luettavissa: <http://www.techradar.com/news/software/operating-systems/windows-vs-linux-which-os-is-best-for-your-business-1265775>. Luettu: 24.10.2017.

bcook-r7 2017/2006. LICENSE. Luettavissa: <https://github.com/rapid7/metasploit-framework/blob/master/LICENSE>. Luettu: 23.10.2017.

Canonical Ltd. 2017. Download Ubuntu Server. Luettavissa: <https://www.ubuntu.com/download/server>. Luettu: 25.10.2017.

Chen, W. 2016. Metasploitable3: An Intentionally Vulnerable Machine for Exploit Testing. Luettavissa: <https://blog.rapid7.com/2016/11/15/test-your-might-with-the-shiny-new-metasploitable3/>. Luettu: 25.10.2017.

Cook, B. 2017. Nightly Installers. Luettavissa: <https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers>. Luettu: 24.10.2017.

Core Security 2015. Penetration Testing Overview. Luettavissa: <http://www.coresecurity.com/penetration-testing-overview>. Luettu: 5.12.2015.

(Huomautus oponoijaa varten: Tarkistettu 13.9.2016, että tieto löytyy Internet Archive Wayback Machinesta <http://web.archive.org/web/20160104132108/http://www.coresecurity.com/penetration-testing-overview>)

dotless-de 2017. vagrant-vbguest. Luettavissa: <https://github.com/dotless-de/vagrant-vbguest>. Luettu: 3.11.2017.

Free Software Foundation, Inc. 2015/1996. What is Copyleft? Luettavissa: <https://www.gnu.org/copyleft/copyleft.html>. Luettu: 4.10.2016.

Free Software Foundation, Inc. 2016a/1996. What is free software? Luettavissa: <https://www.gnu.org/philosophy/free-sw.html>. Luettu: 4.10.2016.

Free Software Foundation, Inc. 2016b/2014. Various Licenses and Comments about Them. Luettavissa: <https://www.gnu.org/licenses/license-list.html>. Luettu: 5.10.2016.

HashiCorp a. Development Environments Made Easy. Luettavissa: <https://www.vagrantup.com/>. Luettu: 24.10.2017.

HashiCorp b. Build Automated Machine Images. Luettavissa: <https://www.packer.io/>. Luettu: 26.10.2017.

HashiCorp c. Creating a Base Box. Luettavissa: <https://www.vagrantup.com/docs/boxes/base.html>. Luettu: 6.11.2017.

HashiCorp d. Creating a Base Box - VirtualBox Provider. Luettavissa: <https://www.vagrantup.com/docs/virtualbox/boxes.html>. Luettu: 6.11.2017.

jbarnett-r7 2016. LICENSE. Luettavissa: <https://github.com/rapid7/metasploit-table3/blob/master/LICENSE>. Luettu: 26.10.2017.

jorgemorais 2015. EOLUpgrades. Luettavissa: <https://help.ubuntu.com/community/EOLUpgrades>. Luettu: 9.11.2017.

Kali Linux 2017. Our Most Advanced Penetration Testing Distribution, Ever. Luettavissa: <https://www.kali.org/>. Luettu: 16.10.2017.

Keviv, kamito300 & sethvargo 2016. Reload. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/cli/reload.html.md>. Luettu: 27.10.2017.

Lakhani, A. 2016. Enable SSH on Kali Linux Enable SSH on Kali Linux. Luettavissa: <http://www.drchaos.com/enable-ssh-on-kali-linux-enable-ssh-on-kali-linux/>. Luettu: 9.11.2017.

Microsoft 2017. Windows Server Evaluations. Luettavissa: <https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server>. Luettu: 25.10.2017.

mitchellh 2013. Mozilla Public License Version 2.0. Luettavissa: <https://github.com/hashicorp/packer/blob/master/LICENSE>. Luettu: 26.10.2017.

mitchellh, fgrehm & aqnouch 2016/2010. The MIT License. Luettavissa: <https://github.com/mitchellh/vagrant/blob/master/LICENSE>. Luettu: 21.11.2016.

mwhooker, sethvargo & feminist 2017. Packer Terminology. Luettavissa: <https://github.com/hashicorp/packer/blob/master/website/source/docs/basics/terminology.html.md>. Luettu: 27.10.2017.

Offensive Security 2017a. Kali Linux Open Source Policy. Luettavissa: <https://docs.kali.org/policy/kali-linux-open-source-policy>. Luettu: 18.10.2017.

Offensive Security 2017b. What is Kali Linux ? Luettavissa: <https://docs.kali.org/introduction/what-is-kali-linux>. Luettu: 18.10.2017.

Offensive Security 2017c. Should I Use Kali Linux? Luettavissa: <https://docs.kali.org/introduction/should-i-use-kali-linux>. Luettu: 18.10.2017.

Offensive Security 2017d. Kali Network Service Policies. Luettavissa: <https://docs.kali.org/policy/kali-linux-network-service-policies>. Luettu: 18.10.2017.

Offensive Security 2017e. Kali Linux Forensics Mode. Luettavissa: <https://docs.kali.org/general-use/kali-linux-forensics-mode>. Luettu: 18.10.2017.

Offensive Security 2017f. Kali Linux Metapackages. Luettavissa: <https://tools.kali.org/kali-metapackages>. Luettu: 18.10.2017.

Offensive Security 2017g. Kali Linux Tools Listing. Luettavissa: <https://tools.kali.org/tools-listing>. Luettu: 18.10.2017.

Offensive Security 2017h. Kali Linux Tools Policy. Luettavissa: <https://docs.kali.org/policy/penetration-testing-tools-policy>. Luettu: 18.10.2017.

Offensive Security 2017i. Metasploit Unleashed. Luettavissa: <https://www.offensive-security.com/metasploit-unleashed/>. Luettu: 19.10.2017.

Offensive Security 2017j. Kali Linux Downloads - Virtual Images. Luettavissa: <https://www.offensive-security.com/kali-linux-vmware-virtualbox-image-download/>. Luettu: 2.11.2017.

Oracle Corporation 2016/2004. Chapter 1. First steps. Luettavissa: <https://www.virtual-box.org/manual/ch01.html>. Luettu: 9.11.2016.

Penetration Testing Execution Standard, The 2014a. Main Page. Luettavissa: http://www.pentest-standard.org/index.php/Main_Page. Luettu: 5.12.2015.

Penetration Testing Execution Standard, The 2014b. Pre-engagement. Luettavissa: <http://www.pentest-standard.org/index.php/Pre-engagement>. Luettu: 14.9.2016.

Penetration Testing Execution Standard, The 2014c. Intelligence Gathering. Luettavissa: http://www.pentest-standard.org/index.php/Intelligence_Gathering. Luettu: 16.9.2016.

Penetration Testing Execution Standard, The 2014d. Vulnerability Analysis. Luettavissa: http://www.pentest-standard.org/index.php/Vulnerability_Analysis. Luettu: 19.9.2016.

Penetration Testing Execution Standard, The 2014e. Exploitation. Luettavissa: <http://www.pentest-standard.org/index.php/Exploitation>. Luettu: 20.9.2016.

Penetration Testing Execution Standard, The 2014f. Post Exploitation. Luettavissa: http://www.pentest-standard.org/index.php/Post_Exploitation. Luettu: 20.9.2016.

Penetration Testing Execution Standard, The 2014g. Reporting. Luettavissa: <http://www.pentest-standard.org/index.php/Reporting>. Luettu: 20.9.2016.

Penetration Testing Execution Standard, The 2015a. FAQ. Luettavissa: <http://www.pentest-standard.org/index.php/FAQ>. Luettu 13.9.2016.

Penetration Testing Execution Standard, The 2015b. Threat Modeling. Luettavissa: http://www.pentest-standard.org/index.php/Threat_Modeling. Luettu: 16.9.2016.

PortSwigger Ltd. 2017a. How to Use Burp Suite. Luettavissa: https://support.portswigger.net/customer/portal/articles/1783101-Using%20Burp_Using%20Burp%20Suite.html. Luettu: 23.10.2017.

PortSwigger Ltd. 2017b. Burp Suite Tools. Luettavissa: https://support.portswigger.net/customer/portal/articles/1783109-Using%20Burp_Burp%20Tools.html. Luettu: 23.10.2017.

PortSwigger Ltd. 2017c. Burp Suite Software. Luettavissa: <https://support.portswigger.net/customer/portal/articles/1855756-burp-suite-software>. Luettu: 23.10.2017.

PortSwigger Ltd. 2017d. Burp Suite Editions. Luettavissa: <https://portswigger.net/burp>. Luettu: 24.10.2017.

PortSwigger Ltd. 2017e. Download Burp Suite Free Edition. Luettavissa: <https://portswigger.net/burp/freedownload>. Luettu: 24.10.2017.

PortSwigger Ltd. 2017f. Burp Tools. Luettavissa: https://portswigger.net/burp/help/suite_burptools.html. Luettu: 24.10.2017.

Rapid7 a. Metasploit Framework. Luettavissa: <https://metasploit.help.rapid7.com/docs/msf-overview>. Luettu: 21.10.2017.

Rapid7 b. Getting Started. Luettavissa: <https://metasploit.help.rapid7.com/docs/getting-started>. Luettu: 20.10.2017.

Rapid7 c. Comparing Product Editions. Luettavissa: <https://metasploit.help.rapid7.com/docs/comparing-product-editions>. Luettu: 20.10.2017.

Rapid7 d. Managing Workspaces. Luettavissa: <https://metasploit.help.rapid7.com/docs/managing-workspaces>. Luettu: 22.10.2017.

Rapid7 e. Tagging hosts in msfconsole. Luettavissa: <https://metasploit.help.rapid7.com/docs/tagging-hosts-in-msfconsole>. Luettu: 22.10.2017.

Rapid7 f. Managing the Database. Luettavissa: <https://metasploit.help.rapid7.com/docs/managing-the-database>. Luettu: 22.10.2017.

Rapid7 g. Running Metasploit Remotely. Luettavissa: <https://metasploit.help.rapid7.com/docs/running-metasploit-remotely>. Luettu: 22.10.2017.

Rapid7 h. Metasploit Basics. Luettavissa: <https://metasploit.help.rapid7.com/docs/metasploit-basics>. Luettu: 20.10.2017.

Rapid7 i. Metasploitable 2 Exploitability Guide. Luettavissa: <https://metasploit.help.rapid7.com/v1.1/docs/metasploitable-2-exploitability-guide>. Luettu: 25.10.2017.

rapid7user 2015. Metasploitable. Luettavissa: <https://sourceforge.net/projects/metasploitable/>. Luettu: 25.10.2017.

Rouse, M. & Kirsch, B. 2016/2006. What is virtualization? Luettavissa: <http://searchserver-virtualization.techtarget.com/definition/virtualization>. Luettu: 27.10.2016.

sethvargo 2016a. Why Vagrant? Luettavissa: <https://github.com/mitchellh/vagrant/blob/master/website/source/docs/why-vagrant/index.html.md>. Luettu: 22.11.2016.

sethvargo 2016b. Providers. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/providers/index.html.md>. Luettu: 15.10.2017.

sethvargo 2016c. Provisioning. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/provisioning/index.html.md>. Luettu: 15.10.2017.

sethvargo 2016d. Vagrantfile. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/vagrantfile/index.html.md>. Luettu: 2.10.2017.

sethvargo 2016e. Networking. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/networking/index.html.md>. Luettu: 15.10.2017.

sethvargo 2017a. Introduction to Packer. Luettavissa: <https://www.packer.io/intro/index.html>. Luettu: 27.10.2017.

sethvargo 2017b. Why Use Packer? Luettavissa: <https://github.com/hashicorp/packer/blob/master/website/source/intro/why.html.md>. Luettu: 26.10.2017.

sethvargo & afeld 2017. Multi-Machine. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/multi-machine/index.html.md>. Luettu: 15.10.2017.

sethvargo & mitchellh 2017. Vagrant vs. CLI Tools. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/intro/vs/cli-tools.md>. Luettu: 2.10.2017.

sethvargo, mwhooker & Purple90 2017. Vagrant Boxes. Luettavissa: <https://github.com/hashicorp/packer/blob/master/website/source/intro/getting-started/vagrant.html.md>. Luettu: 27.10.2017.

sethvargo, pearkes & chrisroberts 2017. Vagrant Share. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/share/index.html.md>. Luettu: 13.10.2017.

sethvargo, vmelnik-ukraine & chrisroberts 2017. Boxes. Luettavissa: <https://github.com/hashicorp/vagrant/blob/master/website/source/docs/boxes.html.md>. Luettu: 13.10.2017.

Sliim 2017. Sliim/metasploitable3-trusty Vagrant box. Luettavissa: <https://app.vagrantup.com/Sliim/boxes/metasploitable3-trusty>. Luettu: 25.10.2017.

Stallman, R. 2016/2007. Why Open Source misses the point of Free Software. Luettavissa: <https://www.gnu.org/philosophy/open-source-misses-the-point.html>. Luettu: 5.10.2016.

Stroud, F. 2017. Top 10 Linux Server Distributions of 2017. Luettavissa: <https://www.serverwatch.com/columns/slideshows/top-10-linux-server-distributions.html>. Luettu: 25.10.2017.

Walker, D. & Shepherd, A. 2017a. Windows vs Linux: what's the best operating system? Luettavissa: <http://www.itpro.co.uk/operating-systems/24841/windows-vs-linux-whats-the-best-operating-system-4/page/0/2>. Luettu: 25.10.2017.

Walker, D. & Shepherd, A. 2017b. Windows vs Linux: what's the best operating system? Luettavissa: <http://www.itpro.co.uk/operating-systems/24841/windows-vs-linux-whats-the-best-operating-system-4>. Luettu: 25.10.2017.

Walker, D. & Shepherd, A. 2017c. Windows vs Linux: what's the best operating system? Luettavissa: <http://www.itpro.co.uk/operating-systems/24841/windows-vs-linux-whats-the-best-operating-system-4/page/0/1>. Luettu: 25.10.2017.

Liitteet

Liite 1. Vagrantfile

```
Vagrant.configure("2") do |config|

  config.vm.define "kali" do |kali_conf|
    kali_conf.vm.box = "tonijaaskelainen/kali"

    kali_conf.vm.provider "virtualbox" do |vb|
      vb.name = "Kali-Linux-Light-2017.2-amd64"
      #vb.gui = true
      #vb.customize ["modifyvm", :id, "--FLAG", "VALUE"]
    end

    kali_conf.vm.base_mac = "080027D7153E"
    #kali_conf.vm.network "private_network", ip: "192.168.56.102"
    # This did not work

    kali_conf.ssh.username = "root"
    kali_conf.ssh.password = "toor"

    #kali_conf.vm.provision "shell",
      #inline: "apt-get update && apt-get install -y kali-linux-
full" # This should install the tools that are found in a normal
Kali Linux installation but some of them require confirmation by
user
    end

    config.vm.define "ms2" do |ms2_conf|
      ms2_conf.vm.box = "tonijaaskelainen/ms2"

      ms2_conf.vm.provider "virtualbox" do |vb|
        vb.name = "Metasploitable-2"
        #vb.customize ["modifyvm", :id, "--FLAG", "VALUE"]
      end

      ms2_conf.vm.base_mac = "080027362690"
      ms2_conf.vm.network "private_network", ip: "192.168.56.101"

      ms2_conf.ssh.username = "msfadmin"
      ms2_conf.ssh.password = "msfadmin"
    end
  end
end
```