Tampereen yliopisto

Antti Ruhala

BACKDOORIN NAAMIOINTI HYÖKKÄYKSEN ERI VAIHEISSA

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Huhtikuu 2019

TIIVISTELMÄ

Antti Ruhala: Backdoorin naamiointi hyökkäyksen eri vaiheissa
Kandidaatintyö
Tampereen yliopisto
Tietotekniikka
Huhtikuu 2019

Tässä kandidaatintyössä selvitetään erilaisia backdooreille ominaisia tapoja välttää hyökkäyksen paljastumista. Kandidaatintyö on tehty kirjallisuusselvityksenä. Työn aineistona toimivat sekä haavoittuvuustestauksen opetusmateriaaliksi luodut teokset sekä aiheen tieteelliset artikkelit. Hyökkäys on jaettu haittaohjelman generointiin sekä toimintaan kohteen hallinnan saavuttamisen jälkeen. Toimintaa kohteessa tarkastellaan ensin yhden kohdejärjestelmän osalta, jonka jälkeen selvitetään usean yhteyksissä olevan järjestelmän hallinnan hyödyt. Kandidaatintyö käsittelee esimerkeissään Metasploit työkalukehyksen tarjoaman Meterpreter service -työkalun ratkaisuja puolustusmekanismien luomiin ongelmiin.

Backdoorien toimintoja tarkastellaan erityisesti naamiointi- sekä harhautuskyvyn tehostamiseen keskittyen. Esiintuodut toiminnot luovat joko suoraan parempaa naamiointikykyä, tai mahdollistavat toisen tapahtuman, jolla on selvä vaikutus hyökkäyksen piilouttamiselle.

Selvityksen tuloksena saadun tiedon perusteella järjestelmään tuodulla backdoorilla on erittäin hyvät mahdollisuudet muokata ympäröivää käyttöjärjestelmää siten, ettei hyökkäyksen saavuttamat muutokset ole ilmiselviä. Hyökkääjä voi manipuloida tunnettuja puolustuksen mekanismeja saavutettuaan tarpeeksi korkeat oikeudet käyttöjärjestelmään. Backdoorin toiminnan pitäminen osana välimuistia piilottaa itse backdoorin useimmilta puolustuskeinoilta, mutta luo ongelmia välimuistin pyyhkiytymisen yhteydessä. Hyökkäyksen säilyvyyden takaamiseksi on useita ratkaisuja sekä itse kohdejärjestelmässä, mutta myös usean yhteydessä toisiinsa olevan järjestelmän hallinnassa. Usean hyökkäyskohteen hallintaa voidaan helpottaa erillisellä Armitage-käyttöliittymällä, joka tuo myös uusia etuja hyökkäyksen tehostamiseen ja tiimityöhön.

Avainsanat: Backdoor, Metasploit, Meterpreter, Payload, Msfvenom, salaus, naamiointi, harhautus.

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

API	engl. Application programming interface, ohjelmointirajapinta.
MACE	engl. Modified-Accessed-Created-Entry, tiedostoon liitetyt luonnin ja käytön metatiedot.
MS	Microsoft, tietotekniikan alan yritys.
NAT	engl. Network Address Translation, sisäverkon IP-osoitteiden piilottaminen yhden julkisen IP-osoitteen taakse.
PID	engl. Process identifier, käynnissä olevan järjestelmäprosessin yksilöivä tunniste.
p0f	engl. Passive Operating system Fingerprinter, Tietoliikennepakettien analyysin työkalu.
SSH	engl. Secure Shell, käyttöjärjestelmän etäkäyttöön suunniteltu protokolla.

1. JOHDANTO

Internetin ja viestiyhteyksien siirtyessä osaksi jokaisen päivittäistä toimintaa erilaisten toisilleen verkon kautta kommunikoivien laitteiden määrä maailmassa kasvaa radikaalisti. Tämä älykkäiden laitteiden yleistyminen voidaan nähdä toimintaa tehostavana tai helpottavana ilmiönä, mutta se lisää myös tietoturvallisuuden kannalta riskejä ja haavoittuvuuksia kohteisiin, joissa niitä ei ennen olisi voinut olla. Käyttöjärjestelmien lisääntyminen jokapäiväisiin laitteisiin sekä näiden toimiminen verkkoyhteydellä mahdollistaa hyökkääjälle backdoor-hyökkäyksen toteutuksen.

Backdoor tarkoittaa järjestelmään luotua yhteyttä, jonka avulla hyökkääjä voi käyttää järjestelmää ilman sen alkuperäisen käyttäjän lupaa ja tietämystä. Backdoor-haavoittuvuus voidaan luoda jo laitteen kehityksen aikana joko tahallisesti tai tahattomasti, mutta yleisesti käytettyjen nykypäiväisten tietokonekäyttöjärjestelmien backdoor-haavoittuvuudet syntyvät järjestelmään tuotujen ohjelmistojen ja prosessien toiminnasta. Tämän kandidaatintyön tutkimusongelmana on selvittää haavoittuvuustestaajan käyttämän backdoor-työkalun naamioinnin tapojen mahdollisuus hyökkäyksen eri vaiheissa. Kandidaatintyössä esitetään ensin yleisesti kaikille backdoor-hyökkäyksille ominaisia piirteitä, jonka jälkeen siirrytään käyttämään *Metasploit: Meterpreter servicen* toimintaa esimerkkinä erään tunnetun backdoorin toteutuksesta. Kyseisen backdoorin tarjoamia hyökkäyksen piilottamisen keinoja tarkastellaan hyökkäyksen eri vaiheissa. Meterpreterin kehityksellisen laajuuden vuoksi kandidaatintyössä tarkasteltuja piilotus- ja hämäysperiaatteita voidaan ajatella sopivaksi soveltaa yleisesti muillakin backdoorsovelluksilla luotuihin hyökkäyksiin.

Luvussa 2 esitellään ohjelmistopohjaisten backdoorien yleisiä piirteitä sekä yhteyden muodostamisen tapoja hyökkääjän ja haittaohjelman välillä. Konkretian lisäämiseksi aiheeseen luvussa 3 tuodaan esille myöhemmin työssä esimerkkinä käytettävä Meterpreter service sekä sitä käsittelevän työkalukehyksen rakenne. Kun kandidaatintyössä esimerkkinä tarkasteltava työkalu on esitelty, päästään luvussa 4 paneutumaan tapoihin, joilla siirrettävä haittaohjelma voidaan naamioida salauksen avulla, ja mitä salauksen onnistumiseksi tulee ottaa huomioon. Luvussa 5 käsitellään hyökkäyksenaktiivivaiheen aikaisia toimia huomaamattomuuden parantamiseksi sekä jälkien piiloittamiseksi. Yhden järjestelmän sisällä toimimiseen tutustumisen jälkeen kandidaatintyössä siirrytään tarkastelemaan useiden kohdelaitteiden muodostamia

verkkokokonaisuuksia. Luvussa 6 tutkitaan kohdejärjestelmän sisäverkossa levittäytymistä sekä siihen liittyviä hyötyjä ja mahdollisuuksia. Viimeisimpänä luvussa 7 esitellään usean backdoorkohteen hallinnan helpottamista hallintatyökalulla ja esitetään sen tuomia etuja hyökkäyksen automatisointiin sekä ryhmätyöskentelyyn.

2. OHJELMISTOPOHJAISET BACKDOORIT

Yleisesti backdoor-haavoittuvuus tarkoittaa järjestelmän vikaa tai ominaisuutta, minkä ansiosta ulkopuolinen käyttäjälle tuntematon toimija voi tarkkailla tai muokata käyttäjän järjestelmän toimintaa. Backdoor-hyökkäyksiä voidaan toteuttaa monella eri tavalla niiden alustoista riippuen, ja niiden syntyminen ei aina ole tarkoituksenmukaista. Erilaisia backdoor-haavoittuvuuksia voi syntyä myös virheellisestä suunnittelusta tai kehittäjän tarkoituksenmukaisesta toiminnasta esimerkiksi vakoilu- tai viranomaistoimintaa varten. Tarkastelun alle jäävät nyt vain ulkopuolisen hyökkääjän asennuttamat ohjelmistopohjaiset backdoor-sovellukset.

Tietoturva-alan haavoittuvuuksien, kuten backdoorien, kuvailemisessa käytetään termejä haavoittuvuus sekä *payload*. Payload on yleisesti tietoliikennetekniikassa käytetty termi, jolla viitataan tiedonsiirrossa siirrettäväksi tarkoitettuun hyödylliseen dataan eroteltuna tiedonsiirtoon vaadittavista osoitetiedoista ja muista asetustiedoista, joiden avulla haluttu hyödyllinen data pyritään saattamaan kohteeseensa. Terminä haavoittuvuus viittaa hyökkäyksen mahdollistavaan vikaan, joka toimii suojakuorena tai mahdollistajana yhdelle tai usemmalle payloadille, jotka sisältävät vaarallisen datan. Yksi haavoittuvuus, kuten virheellisesti luotettavaksi todettu muokattu sovellustiedosto, voi sisältää useita eri toiminnallisuuksia sisältäviä payloadeja. [4, 7] Sovelluspohjaisten backdoorien tapauksessa käyttäjän tietokoneelle asennettava ohjelmisto toimii haavoittuvuutena, ja sovellukseen piilotettu haitallinen backdoor-koodi on kyseisen haavoittuvuuden payload.

Tiedonsiirtoa vaativan hyökkäyksen tavat muodostaa yhteys hyökkääjän ja kohteen välille voidaan jakaa kahteen kategoriaan yhteydenoton suuntaan perustuen. Vahingollinen sovellus kantaa mukanaan joko *Reverse shell*- tai *Bind shell* -payloadeja, jotka toimivat yhteyden muodostuksen mekanismeina. [7]

Reverse shellissä yhteys hyökkääjän palvelimeen muodostetaan kohteesta lähtien [7]. Yhteyden ottaminen kohteesta kuuntelevalle hyökkääjän palvelimelle ei jää oletusasetuksilla toimivaan palomuuriin, sillä ulospäin lähtevät yhteydet sallitaan, koska yleisesti järjestelmään asennetut ohjelmistot ovat asennuksen jälkeen luotettuja [7].

Bind shell on tapa, jossa payload kiinnittyy kohdetietokoneen porttiin, johon hyökkääjän palvelin myöhemmin muodostaa yhteyden. Tällainen yhteydenmuodostuskeino on vähemmän käytetty, sillä palomuurit on usein asetettu toimimaan pääsylistojen avulla, jolloin ulkoa päin tulevan tuntemattoman lähteen lähettämät paketit suodatetaan. [7]

Backdoor-payloadeja voidaan yrittää piilottaa ohjelmistojen koodiin *loogisina pommeina*, jolloin vain tietyt harvinaiset käyttötapaukset laukaisevat haitallisen koodin [12]. Vaihtoehtoisesti payload voidaan rakentaa *itsemuokkaantuvalla koodilla*, jossa koodia silmämääräisesti tai automaatiolla tarkastellessa jotkin osat haitallisesta koodista eivät näy. Eräs hyvä tapa löytää backdooreja on tarkastella joidenkin tunnettujen tiedostosijaintien esiintymistä ohjelmiston lähdekoodissa [12]. Itsemuokkaantuvassa koodissa jotkin merkkijonotyyppiset muuttujat on voitu piilottaa esimerkiksi ohjelmointikielen yleisesti tarjoamien salausalgoritmien taakse, jotka avataan vasta prosessin käynnistyksen yhteydessä. Tunnetut payloadit yritetään usein salata kokonaisuuksina, jotteivat niiden lähdekoodin tuntevat puolustusmenetelmät huomaisi niitä. Suurten kokonaisuuksien salaaminen luo kuitenkin dataan selviä alueita, joissa *entropia* eli datan epäjärjestys on merkittävästi suurempaa verrattuna ympäröivään dataansa. Suuri määrä entropiaa datassa viittaa usein joko kompressointiin tai salattuun dataan. [12] Nykyisillä analyysiin perustuvilla puolustuskeinoilla on erityisesti vaikeuksia löytää kehitysvaiheessa ohjelmistoon lisättyjä loogisia pommeja [1].

Backdoor voidaan tuoda järjestelmään myös vaiheistetusti esimerkiksi luomalla ohjelmistoon kuollutta koodia, jota hyödyntävä backdoor tuodaan vasta myöhemmän päivityksen aikana. [12] Näin backdoorin tarvitsema haitallinen koodi saadaan jaettua useaan eri ohjelmiston lähdekoodin osioon, jolloin erityisesti silmämääräinen tarkastelu vaikeutuu. Osa tuodusta backdoorin koodista saattaa säästyä puhdistukselta, jos hyökkäyksen jokin osa paljastuu. Kehityksen aikana järjestelmään tarkoituksella lisättyä haavoittuvuutta kutsutaan *elinkaarihyökkäykseksi (life cycle attack*) [1]. Tuotua ja asetettua backdoor-koodia voidaan käyttää hyödyksi toisen hyökkäyksen osana [5]. Jos jokin toinen samaan järjestelmään suunniteltu hyökkäys estetään ja siihen liittyvät tiedostot poistetaan, voidaan ne luoda uudelleen aikaisemmin saavutetun backdoorin avulla. [12]

Backdoor-hyökkäyksen mahdollistaneen sovelluksen kehittäjän maineen menetys on usein suurempi haitta kuin itse haavoittuvuuden aiheuttamat vauriot. Backdoor-haavoittuvuus voidaan nähdä vahinkona, mutta hyökkääjän pääsy osaksi kehittäjän palvelun tuotantoketjua tai lähdekoodia lopullisessa julkaisussa lasketaan usein epäpätevyydeksi. [12]

Useat nykypäiväiset sensorityyppiset laitteet kuten mittarit ja kamerat ovat erittäin haavoittuvia hyökkäyksille. Monesti teollisuuskäyttöön suunnitellut laitteet jätetään konfiguroimatta ennen käyttöönottoa. Tehtaalta lähtenyt verkkoon kytketty kamera saattaa olla oletuksena käytettävissä ilman salasanaa, pelkällä käyttäjätunnuksella, tai suoraan laitteen julkisista tiedoista löytyvillä tunnuksilla. [10] Haavoittuvuuksien

esiintymistä selvittävässä tutkimuksessa [10] tutkijat löysivät 218 testatusta teollisuuden sensorilaitteesta yhteensä 95 backdoor-haavoittuvuutta, kun tutkimuksen alla olivat sekä rauta- että ohjelmistotason haavoittuvuudet.

3. METASPLOIT: METERPRETER SERVICE

Metasploit framework on avoimen lähdekoodin haavoittuvuustestaukseen luotu työkalukehys, jonka tarkoituksena on auttaa informaation keräämisessä haavoittuvuuksien esiintymisen suhteen [2, 7]. Metasploit tarjoaa suuren määrän eri haavoittuvuuksia käsitteleviä työkaluja, joita pyritään pitämään ajantasalla vastaamaan kyseisiin haavoittuvuuksiin tehtyjä korjauksia.

Metasploit frameworkin työkalut on jaettu erilaisiin moduuleihin, joiden yhteistoiminnalla voidaan toteuttaa kokonaisia hyökkäyksiä lähtien kohdejärjestelmän tarkastelusta ja päättyen järjestelmän murtoon ja haittaohjelmien asennukseen. Nimellisesti mainittuina moduulit on jaettu *Exploit*, *Auxilary* ja *Post-Exploitation* moduuleihin. *Exploit* moduuli sisältää sarjan työkaluja, joilla payloadeja voidaan siirtää esimerkiksi web-applikaatioiden kautta kohteen tietokoneelle [7]. *Auxilary* moduuli pitää sisällään erilaisia haavoittuvuusskannauksen ja denial of service -tyyppisiä hyökkäyksiä. Backdoor-hyökkäykset sekä muut saavutetun murron jälkeen hyödynnettävissä olevat haavoittuvuustestauksen työkalut on sijoitettu *Post-Exploitation* moduuliin.

Meterpreter service on eräs Metasploitin tarjoamista backdoor-tyyppisistä hyökkäyksistä, jonka useat Metasploitin muut hyökkäykset mahdollistavat. Esimerkiksi edellä mainitut *Reverse shell*- ja *Bind shell* -payloadit tarjoavat yhteyden muodostuksen jälkeen hyökkääjälle Meterpreter shellin, eli Meterpreter servicen komentorivin. [7]

Useissa tietokonejärjestelmissä uuden komentorivisyötettä kuuntelevan prosessin aloittaminen ajatellaan merkkinä haitallisesta toiminnasta. Sekä Windows- että UNIX-järjestelmiin on kehitetty suojaohjelmia, jotka valvovat tällaisen toiminnan esiintymistä. Komentoriviä kuuntelevan prosessin luonti myös luo uuden prosessin järjestelmän prosessilistaukseen, mikä altistaa hyökkääjän näkyvyydelle. [8] Meterpreterin toiminnassa on otettu huomioon edellä mainittu hyökkääjän prosessin näkyvyyden ongelma, ja tätä sekä muita piiloutumisen tapoja käsitellään tarkemmin tämän kandidaatintyön haittaohjelman naamiointia käsittelevissä kappaleissa.

Meterpreter on laajennuksilla rakentuva hyökkäystyokalu, joka on itsessään hyvin pienikokoinen. Hyökkäyksen edetessä erimuotoisia laajennuksia ladataan hyökkääjän tarpeiden mukaan yhteydessä olevalta hyökkääjän palvelimelta, kunhan yhteys on onnistuneesti muodostettu.

Meterpreter on Metasploitin tapaan open source -kehityksen tulosta, ja tarjoaa laajan Ruby-pohjaisen API:n (Application programming interface) jatkokehitystä varten.

Laajennettavissa olevan API:n toiminnot on koottu esimerkiksi rubydoc verkkosivulle [9]. Meterpreterille on mahdollista kirjoittaa omia laajennuksia millä tahansa ohjelmointikielellä, joka tukee koodin jakamista shared objecteina. Tämä mahdollistaa ylemmän tason ohjelmointikielien tuen, eikä kehittäjän tarvitse käyttää esimerkiksi Assembly -ohjelmointikieltä jatkokehityksessä. [8]

Meterpreterin käyttämien pakettien lähettämiselle on luotu oma spesifikaatio. Jotta pakettien laajentaminen olisi helppoa, eikä niiden rakennetta tarvitsisi muuttaa lähettämisen vuoksi, on Meterpreterin pakettien lähettäminen toteutettu Length-Type-Value rakenteeseen perustuvalla protokollalla. Näin satunnaisen kaltaisia sisältöjä voidaan lähettää ilman että niitä pilkkovan koodin tarvitsee tuntea datan formaattia. Erilaiset rakenteen tunnisteet lisätään näin ollen osaksi Value-osiota. [8] Eräs hyöty Length-Type-Value formaatista on, että se mahdollistaa datan merkitsemisen tietyille kanaville, jolloin usea kohdejärjestelmän prosessi voi keskustella hyökkääjäkoneen kanssa. [11]

4. METERPRETER PAYLOADIN MUODOSTUS

Payload viittaa hyökkääjän luomaan haitalliseen koodiin, joka pyritään saamaan haluttuun kohteeseen haavoittuvuutta hyödyntäen. Useiden backdoorien lähdekoodit rakentuvat osista, joita on käytetty muissakin jo tunnetuissa backdooreissa. Esimerkiksi Meterprerin koodi jää nykypäiväisimpien antivirusohjelmistojen suodatukseen, mutta tätä riskiä voidaan vähentää käyttämällä salausta payloadin ympärillä sekä muistin hyödyntämisellä haittaohjelman toiminnan alustana [2]. Tunnistettuja uhkia ja niiden piirteitä jaetaan aktiivisesti puolustuskeinojen kehittäjien kesken [5].

Metasploit framework tarjoaa useita payloadien salausalgoritmejä, joista eräs toimivaksi todettu on *Shikata ga nai* (Engl. it cannot be helped). Shikata ga nai on polymorphinen enkooderi, eli sen tulos vaihtuu joka kerta kun se ajetaan. On mahdollista, että antivirus tunnistaa erään enkoodauksen tuloksen mutta ei toista. Hyökkääjälle on suositeltavaa kokeilla antiviruksen tunnistuskykyä omalla palvelimellaan ennen haavoittuvuustestauksen aloittamista. [6]

Salauksen jälkeen payload voidaan liittää osaksi ajettavaa exe-tiedostoa. Metasploit framework on yhdistänyt *Msfencoden* ja *Msfpayloadin* yhdeksi työkaluksi nimeltä *Msfvenom*, jolloin payloadin salaaminen ja liittäminen osaksi alustasovellusta voidaan toteuttaa yhdellä työkalulla helposti [4]. Käytetty exe-tiedosto voi olla mikä tahansa, jonka hyökkääjä ajattelee kohteen avaavan. Exe-tiedostoon liittämisen lisäksi payload on mahdollista lähettää yksinään yksinkertaisen verkkosivuston kautta, jonka IP-osoite vastaa hyökkääjän palvelimen ylläpitämää serveriä [6]. *Msfpaylodin* tuottamia tiedostoja voidaan käyttää myös osana itsekehitettyjä C, Ruby, Java sekä Perl scriptejä työkalun hyvän ulostuloformaattituen ansiosta [2].

Verkkosivujen lisäksi yleisiä tapoja saada payload-tiedostoja kohdejärjestelmään ovat esimerkiksi luottavalta lähteeltä kuulostava sähköpostiviesti, maasta poimittu muistitikku tai social engineering, jossa kohde luottaa hyökkääjään siinä määrin, että hän suostuu itse asentamaan payloadin tietokoneeseensa hyökkääjän opastuksen alaisena. Aiemmin avattu payloadin liittäminen osaksi tunnettua ja luotettua ohjelmistoa salattuna voidaan ajatella troijalaisten kategoriaan haitallisten ohjelmakoodien luokittelussa [2].

Kun payload on onnistuneesti saatu kohteen tietokoneelle, on kohteen vielä avattava saastutettu tiedosto, jolloin *Reverse shell-* tai *Bind shell* -payload aktivoituu ja yhteys muodostuu hyökkääjän tietokoneelle. M. Baggetin tutkimuksessa [2] huomattiin, että haittaohjelman muuttaminen esimerkiksi MS Paintin troijalaiseksi vähensi sen

tunnistavien antivirusten määrää merkittävästi. Troijalaisen salaaminen joillain muilla kuin *Msfencoden* oletusparametreilla teki tunnistuksen mahdollisuuksista lähes olemattomia, etenkin jos hyökkääjä tietää mitä tunnettuja bittijonoja tuotoksessa tulee välttää.

Backdoorien kehittäjät voivat käytännössä luoda oman laajennuksensa Meterpreterin valmiin rakenteen päälle juuri haluamaansa tarkoitukseen. Meterpreteriin on kuitenkin valmiiksi rakennettu yleisimmät hyökkääjän tarpeisiin sopivat työkalut kohteen salakuunteluun, tiedostorakenteiden muutamiseen sekä yhteyksien luontiin ja backdoorohjelmiston siirtämiseen toisille sisäverkon tietokoneille. [6, 8]

Meterpreterin rakenne vaihtelee hieman hyökkäyksen kohteen käyttöjärjestelmän mukaisesti. Hyökkäyksen alussa kohteen käyttöjärjestelmän selvittämiseen voidaan käyttää kohteen verkkoon lähettämiin paketteihin jättämiä käyttöjärjestelmäominaisia piirteitä [4]. Metasploitista löytyy useita työkaluja, joilla kohteen rakennetta ja toimintaa voidaan tarkastella ulkoapäin. Pakettien ja käyttöjärjestelmien toisiinsa kytkentään löytyy esimerkkinä *Passive Operating system Fingerprinter* -työkalu (p0f) [4]. Samankaltaista tunnistusta voidaan toteuttaa tunnetummalla *Nmap*-työkalulla, jonka toiminta perustuu kohteeseen lähetettyihin ja sieltä vastauksena tulleisiin paketteihin, kun taas edeltävä *p0f* tarkastelee paketteja sivustakatselijan roolista.

5. HAITTAOHJELMAN NAAMIOINTI

Haitallista toimintaa tarkkaillaan monen eri tahon toimesta. Hyökkäyksen piilottamiseksi on otettava huomioon niin järjestelmän automatisoidut puolustuskeinot, kuin järjestelmää hallinnoiva käyttäjä. Siinä missä edeltävä luku keskittyi esivalmisteluihin, joilla haittaohjelma saadaan kohdejärjestelmään, keskitytään tässä luvussa hyökkäyksen aikaisiin ja jälkeisiin toimiin.

5.1 Todisteiden ja todistajien siivous

Hyökkäyksen aikana tehdyt toimet jättävät jälkensä. Käyttöjärjestelmät valvovat omia tapahtumiaan erilaisilla lokitiedostoilla, ja jotkin ulkopuoliset ohjelmat kuten antivirus ja muut turvapalvelut saattavat pyrkiä tunnistamaan sekä estämään vaarallisiksi todettuja toimia. Meterpreter toimii täysin tietokoneen välimuistissa, ellei hyökkääjä tee säilymisen (persistence) mahdollistavia toimenpiteitä, jotka kirjoittavat tietokoneen levylle. Vaikka toiminta tapahtuukin lähinnä välimuistissa, syntyy käyttöjärjestelmän lokitietoja hyökkäyksen alussa tapahtuneesta prosessin avaamisesta, muutoksista toisiin tiedostoihin sekä hyökkääjän ja kohteen välillä tapahtuneesta tietoliikenteestä. Hyökkääjän kannalta olisi ideaalia, jos hyökkäyksen kohteena toiminut järjestelmä voitaisiin tyhjentää kokonaan ja uudelleenrakentaa, jolloin kaikki jäljet hyökkäyksestä katoaisivat [6]. Usein on kuitenkin riittävää, että hyökkäyksen kohteen tapahtumavirta sotketaan siihen pisteeseen, ettei jälkeenpäin tehty tarkastelu paljasta hyökkäyksen todellista laajuutta [6]. Mikäli tapahtumavirran jättämistä näkyviin halutaan välttää, voidaan kohdejärjestelmä kopioida kokonaisuudessaan. Tiedostoja tarkasteleva hyökkäys voidaan toteuttaa kokonaan järjestelmän kopioon. [11] Tämän kaltainen varmuuskopiointi antaa hyökkääjälle aikaa avata mahdollisia salauksia kohdejärjestelmässä.

Windows käyttöjärjestelmien lokitiedot tallennetaan tunnettuun paikkaan, ja lokijärjestelmän toiminta on tunnettua. Meterpreteriin on kehitetty *event_manager*-työkalu, jonka tehtävänä on antaa hyökkääjälle helppo tapa muokata Windows tapahtumalokia tai puhdistaa se kokonaan. Hyökkäyksen aikana on suositeltavaa pitää kirjaa omista muutoksista kohdejärjestelmään, jotta muutosten piilottaminen olisi helpompaa jälkeenpäin [6]. Suurten aikakatkojen esiintyminen tapahtumalokeissa näyttää epäilyttävältä ulkoiselle tarkastelijalle. Usein hyökkäyksestä jää informaatiota jäljelle jälkisiivouksesta huolimatta, minkä perusteella tutkiva osapuoli pystyy toteamaan,

että jotain on tapahtunut. Tiedon rajallisuuden takia on kuitenkin usein vaikeaa sanoa tarkasti, mitä järjestelmälle on todellisuudessa tehty. [6]

Antivirusohjelmistot keräävät ja tarkkailevat hyökkääjän toimintaa. Tämän ongelman kohtaamiseen on luotu useita työkaluja. Yleisimpien antivirusohjelmistojen prosessien tappamiseen on luotu oma komentonsa *killav*, mutta prosesseja voidaan tappaa myös niiden PID (Process identifier) perusteella. [8] Kohdejärjestelmän tunnetut puolustusmekanismit kuten Windows palomuuri voidaan joko listata tai tappaa kokonaan. Meterpreter tunnistaaa käytetäänkö käyttöjärjestelmää osana virtuaalikonetta, joka voi viitata hyökkääjän toiminnan tarkasteluun suoraan käyttäjän toimesta eli siihen, että hyökkäystä osattiin odottaa. Eräs yleinen haittaohjelmien analyysin keino on antaa haittaohjelman toimia vapaasti suljetussa ympäristössä ja tutkia miten se toimii. Tällaista tutkimusta kutsutaan *dynaamiseksi analyysiksi*, verraten *staattiseen analyysiin*, jossa haittaohjelman koodia tutkitaan käynnistämättä itse ohjelmaa. [5]

5.2 Ympäristöön sulautuminen

Haittaohjelman aktivointi ja hyökkääjän yhteyden muodostaminen vaativat aktiivista kuuntelevaa prosessia. Tällaisen prosessin jättäminen näkyväksi tarkastelulle mahdollistaa hyökkäyksen paljastumisen.

Meterpreter payload piilottaa käyttämänsä prosessin sen kätkevän sovelluksen prosessin sisälle. Haittaohjelman prosessi käynnistyy isäntäprosessinsa mukana. [8] Käyttäjä näkee vain tarkoituksenmukaisesti käynnistämänsä prosessin, tarkastellessaan aktiivisia prosesseja tehtävienhallinnasta. Haittaohjelman on jopa mahdollista liikkua eri prosessien välillä, esimerkiksi osaksi explorer.exe -prosessia. [6] Tällöin alkuperäisen prosessin sammuttaminen ei vaikuta hyökkäykseen. Useat antivirusohjelmistot eivät pääse tarkastelemaan muistin sisäistä toimintaa, joten muistin sisällä toimiva haittaohjelma saattaa välttyä aktiivisiltakin tunnistusyrityksiltä [2].

Hyökkäyksen aikainen toiminta on keskitetty kokonaan tietokoneen muistiin, eikä levylle kirjoiteta mitään [8]. Näin haittaohjelma pysyy paremmin piilossa, mutta ongelmia syntyy esimerkiksi tietokoneen sammuttamisen ja muunlaisen muistin pyyhkiytymisen yhteydessä.

Meterpreterin toteutus mahdollistaa SSH-yhteyden vaativan porttiohjauksen (port forwarding) asettamisen, jolla hyökkääjä voi luoda SSH-yhteyksiä hyökkäyskohteesta toisiin koneisiin, joihin ei hyökkääjän koneelta olisi ollut suoraa näköyhteyttä. Meterperter

tarjoaa tuen kustomisoidulle pakettien salaukselle, ja oletuksena käyttää laajennuspakettiensa lataamiseen XOR-salausta. Vaikka XOR ei ole sinällään turvallinen, antaa se hyvän suojan pakettien piilottamiselle puolustusjärjestelmiltä, jotka etsivät tiettyjä tunnettuja kaavoja lähetetystä liikenteestä. [8]

Eräs backdoorien löytämisen keino on ”koputella” tietokoneen portteja löytääkseen niitä kuunteleva backdoor. Tätä vastaan voidaan taistella vaihtamalla käytettyä porttia tietyin väliajoin tai asettamalla tietynlainen pakettien sekvenssi, jolla kuunteleva backdoori aktivoituu. [3] R. Bestak et al. nostavat esille tutkimuksessaan [3] cd00r-ohjelman. Kyseinen backdoor kuuntelee valittua TCP-porttia ja aktivoituu vasta tietyn TCP SYN -pakettisekvenssin jälkeen. Tutkijat esittävät julkaisussaan koneoppimisen käyttämistä löytämään tämän kaltaisia harvinaisia tapahtumia tietoliikennevirrasta.

5.3 Käyttäjän harhautus

Vaikka hyökkääjä onnistuisi välttämään järjestelmän omat puolustusmekanismit, on silti mahdollista, että hyökkäys epäonnistuu tarkkaavaisen käyttäjän takia. Hyökkäyksen jälkeiset muutokset voivat olla syntyneiden lokitapahtumien lisäksi järjestelmään tuotuja uusia tiedostoja. Joissain tapauksissa lokitapahtumiakaan ei ole järkevää piilottaa, sillä tapahtumien seuraukset saattavat olla niin ilmiselviä, ettei lokitiedon poistamisella ole haluttua vaikutusta. Näissä tapauksissa tapahtuma voidaan piilottaa mahdollisen toisen käyttäjän tekemäksi.

Lokitietojen muokkaamiseen käytetyn *event_manager* lisäosan rinnalle on luotu hyödyllinen *timestomp*-työkalu, jonka käyttötarkoitus on lokitietojen muokkaamisen tapainen, mutta kohteena on tällä kertaa tiedostojen MACE-metatiedot (Modified-Accessed-Created-Entry) [11]. *Timestomp*-työkalulla on mahdollista muuttaa esimerkiksi tiedoston luonti-, muokkaus- ja viimeisin käyttöpäiväattribuutteja. [6, 11] Käyttöjärjestelmän syvyyksistä löytyvä tiedosto, jota on viimeksi käytetty useita vuosia sitten, ei herätä yhtä suuria epäilyksiä kuin viimepäivinä syntynyt.

Tiedoston metatietojen muokkauksen lisäksi voidaan myös tiedoston luonnin ajaksi vaihtaa aktiivista käyttäjää, jolloin saavutetaan mahdollisten lisäoikeuksien rinnalla myös turvallisuutta siinä mielessä, että aktiviteettien toteuttaja on käyttäjän silmissä joku toinen kuin tarkasteleva käyttäjä itse. Kerberos todennusprotokolla on nykyaikaisten Windows- sekä muutamien UNIX-järjestelmien oletusprotokolla käyttäjien todentamiseen ja erottamiseen toisistaan. Käyttäjien oikeudet sekä identiteetit ovat liitettyinä käyttäjille osoitettuihin tunnisteisiin (token), jotka luodaan käyttäjää kohti käyttäjän kirjautuessa sisään. Kerberos tunniste on voimassa käyttöjärjestelmässä tietyn ajan kirjautumisen

jälkeen [6]. Käyttäjän oikeellisuuden tarkastaminen kertaalleen, ja myöhempien saman käyttäjän toimien hyväksyminen aikaisemman tarkastuksen perusteella lisää tunnistautumisprotokollien tehokkuutta [4]. Hyökkääjä voi kasvattaa oikeuksiaan käyttöjärjestelmässä kaappaamalla sisäänkirjautuneen järjestelmänvalvojan tunnisteen [6, 11]. Tämän kaltainen hyökkäys kuitenkin vaatii, että kohdetietokoneelle on kirjauduttu sisään kyseisillä tunnuksilla tunnisteen vanhentumisajan loppumista lyhyemmän ajanjakson aikana. [6].

6. LEVITTÄYTYMINEN JA PALAUTUMINEN POISTOYRITYKSISTÄ

Laajojen kokonaisuuksien haltuunotto backdoorien avulla vaatii hyökkäyksen laajentamista yhdestä kohdejärjestelmästä siihen yhteyksissä oleviin järjestelmiin. Laitteiden verkosto koostuu useista erilaisista laitteista, joiden välillä liikkuminen saattaa vaatia useiden eri haavoittuvuuksien hyväksikäyttöä. Useiden laitteiden haltuunotto luo hyökkäykselle sekä mahdollisuuksia palautumiselle että laajuutta hyökkääjän kohdenäkyvyysalueelle.

6.1 Liikkuminen sisäverkossa

Jotkin kohdejärjestelmät voivat olla piilossa alkuperäisiltä kohteen etsinnän skannauksilta palomuurin tai NAT (Network Address Translation) avulla luoduissa sisäverkoissa [4]. Voi olla, että alkuperäinen hyökkäyksen kohde ei ole näkyvissä suoraa julkisesti, vaan vaatii ensin jonkin toisen sisäverkkoon yhteydessä olevan järjestelmän hallinnan. [6, 11] Palomuurin ohittamiseksi voi usein riittää pelkkä MAC-osoitteen väärentäminen vastaamaan palomuurin pääsylistan sisältöä [4]. Sisäverkossa liikkuminen on kohdejärjestelmän tutkimisen ja hallinnan kannalta kriittistä, mutta siitä on myös piiloutumisen ja säilyvyyden kannalta tärkeitä hyötyjä. Metasploit tarjoaa sarjan porttiskannauksen työkaluja, joilla hyökkääjä voi etsiä tapoja liikkua sisäverkon järjestelmien välillä [4, 6, 11].

Järjestelmien välillä liikkumiselle on useita erilaisia toimivia ratkaisuja, mutta yleisiä haavoittuvuuksia ovat vanhentuneet SSH-palvelimet (Secure Shell), MS SQL -server (Microsoft SQL) sekä Pass the Hash haavoittuvuus. Tämän kandidaatintyön aiheen rajauksen mukaisesti haavoittuvuuksien käsittely jätetään vähäiseksi, mutta valveutuneen käyttäjän tulisi ainakin ymmärtää, mihin haavoittuvuudet perustuvat.

SSH on yleisesti turvallinen tiedonsiirtoprotokolla, mutta sen vanhentuneista päivittämättömistä versioista on löydetty haavoittuvuuksia [6]. Vanhentuneita SSH-palvelimia voi löytyä vanhoista tietokoneista, joita ei ole välttämättä liitetty osaksi internettiä useaan vuoteen turvallisuussyistä.

MS SQL liittyvät haavoittuvuudet perustuvat SSH tapaan päivittämättömiin ja konfiguroimattomiin asennuksiin. Monet käyttäjät eivät ole tietoisia järjestelmissään pyörivistä MS SQL -servereistä, sillä ohjelmisto asennetaan osana yleisempiä ohjelmia.

Eräs kyseisen serverin asentava ohjelmisto on ohjelmistokehityksen työkalu Microsoft Visual Studio. [6]

Sisäverkon tietokoneilla on usein yksi ja sama järjestelmänvalvoja. Tämä järjestelmänvalvoja saattaa olla korjauksiin tai yleiseen hallinnointiin tarkoitettu tunnus, jolla on suuremmat oikeudet kuin arkipäiväkäyttöön tarkoitetuilla tunnuksilla. Pass the Hash on saman järjestelmänvalvojan tunnuksen käyttöön perustuva haavoittuvuus [4, 6]. Tunnuksen salasana ja käyttäjänimi säilötään järjestelmään hash-salattuna [4]. Hashin murtaminen on ajallisesti vaikeaa, sillä se on alusta lähtien suunniteltu yksisuuntaiseksi salaukseksi. Aikaisemmin esiin tuotuun Windows järjestelmien yleisesti käyttämään Kerberos tunnistautumisprotokollaan on liitetty ominaisuudeksi mahdollisuus NTLM (New Technology LAN Manager) tunnistautumiselle [6]. NTLM mahdollistaa tunnistautumisen pelkällä tunnuksen salasanan hashillä [4, 11], jolloin tietokoneelta kaivettua järjestelmänvalvojan salattua käyttäjätunnusta voidaan käyttää yhteyden avaamiseksi toiselle saman verkon tietokoneelle [6].

6.2 Komentoasemien luonti useaan saman verkon koneeseen

Sisäverkon usean tietokoneen hallinta on jatkuvalle backdoor-hyökkäykselle tärkeää. Hyökkäyksen aikana tapahtunut paljastuminen johtaa usein kohdejärjestelmän poistamiseen verkosta tai sammuttamiseen. Mikäli kohde puhdistetaan hyökkäystä edeltävään tilaan, olisi hyökkäys aloitettava taas alusta lähtien uusien sisäänpääsykeinojen selvittämisestä. Ongelmaa voidaan helpottaa luomalla useita komentoasemia hyökättävään verkostoon. Useat komentoasemat estävät yhtä järjestelmää muodostumasta pullonkaulaksi yhteyksille, ja niiden kautta voidaan uudelleenrakentaa paljastuneet yhteydet hyökkäykselle kriittisiin paljastuneisiin kohteisiin.

Sisäverkon laitteiden haltuunottoa voidaan toteuttaa aikaisemman kappaleen järjestelmien välisellä liikkumisella, tai vaihtoehtoisesti alkuperäisen järjestelmän haavoittuvuutta voidaan mahdollisuuksien mukaan soveltaa muihin verkoston tietokoneisiin. Eräs hyökkäyksen puolustustarkoituksellinen kartoituksen keino on tutkia verkon tietoliikennettä, ja selvittää mikä järjestelmä on yhteydessä ulkopuoliseen hyökkääjään. [12] Hyökkääjän tietoliikenne näkyisi tarkastelussa epänormaalina, muilta tietokoneilta puuttuvana tietoliikenteenä. [12] Tätä tarkastelua voidaan hidastaa luomalla

monimutkaisia rakenteita, joissa useat järjestelmät ovat yhteydessä hyökkääjään joko suoraan tai toisten sisäverkon laitteiden kautta. Näin tehdessä hyökkääjän yhteys ei näy yksittäisen kohteen tietoliikenteenä, ja hyökkäyksen toteutus voidaan jakaa usean eri kohteen vastuualueiksi, jos usealla tietokoneella on yhteys johonkin suurempaan kohdealueeseen.

6.3 Palautuminen uudelleenkäynnistyksestä

Meterpreter on suunniteltu toimimaan käyttöjärjestelmän välimuisissa. Täysin muistiin keskittyvä toiminta piilottaa haittaohjelman käytetyiltä puolustusmekanismeilta, mutta tuo mukanaan merkittävän ongelman hyökkäyksen säilyvyydelle. Mikäli kohteeseen ei ole liitettävissä toista järjestelmää, jossa hyökkäystä voidaan pitää elossa kohteen sammuttamisen ajan, on hyökkääjän keksittävä tapa luoda hyökkäyksen avaava yhteys uudestaan kohteen uudelleenkäynnistyksen yhteyteen. Looginen ratkaisu ongelmaan on asettaa yhteyden avaus skriptinä osaksi kohteen käynnistuksen operaatioita.

Pysyvyys (*Persistence*) voidaan toteuttaa asettamalla kohdejärjestelmä luomaan yhteyskutsuja tietyin väliajoin, tai asettamaan *Bind Shellin* portti kuuntelutilaan. [6, 11] Pysyvyyden rakentaminen kohdejärjestelmään tekee hyökkäyksestä selvästi huomiota herättävämmän. Älykäs tarkastalija saattaa huomata yhteyden avauksen jokaisella uudelleenkäynnistyksellä, jolloin aikaisemmin saavutetut muutokset saattavat vaarantua. On tärkeää huomata, että jos pysyvyyden mahdollistavat operaatiot unohdetaan poistaa hyökkäyksen lopuksi, jää järjestelmä toisille avoimeksi. Mikäli toinen hyökkääjä onnistuu selvittämään kuuntelevan portin tai hyökkääjän kutsuttavan IP-osoitteen, on hänellä suora yhteys kohdejärjestelmään.

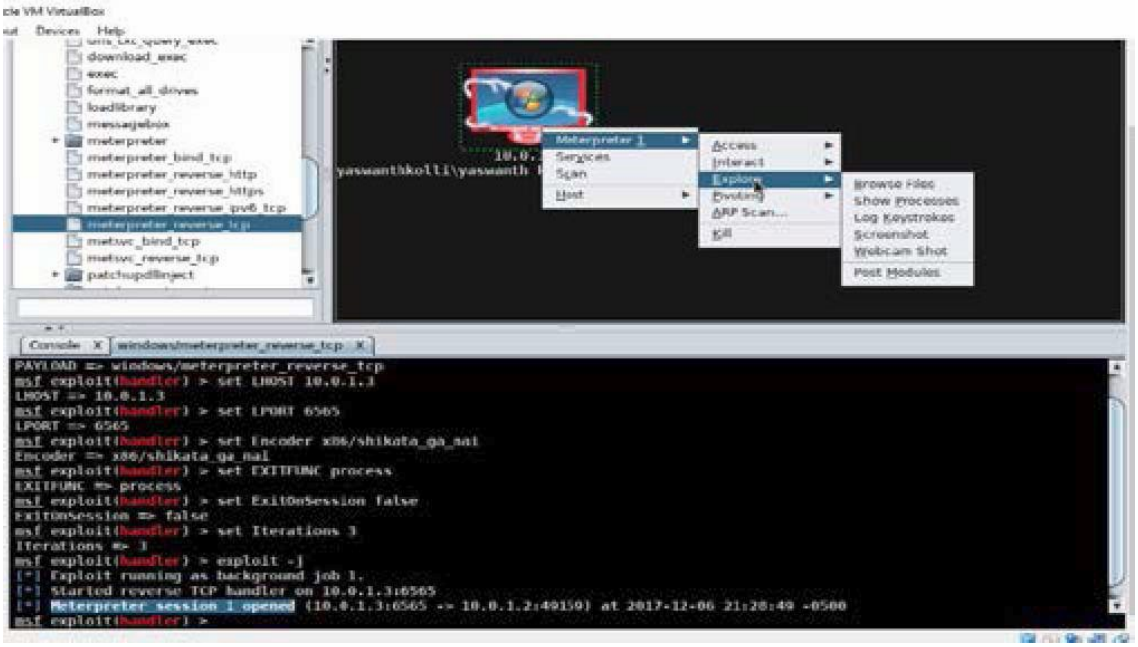
7. BACKDOOR-VERKOSTON HALLINTA

Kun hyökkäyksen kohteiden konstellaation koko kasvaa, voi hyökkääjän olla yksin tai edes ryhmänä työskennellessä vaikeaa hallinnoida kaikkien kohdelaitteiden erinnäisiä suhteita toisiinsa nähden. Ongelmaa varten on kehitetty hallinnointijärjestelmiä, joista tässä kandidaatintyössä tutustumme Metasploit frameworkille ominaiseen *Armitage* hallinnointijärjestelmään.

Armitage on Metasploit frameworkin päälle rakennettu graafinen käyttöliittymä [7, 6]. Se on rakennettu helpottamaan Metasploitin käyttöä ja asentuu Kali Linux käyttöjärjestelmän normaalissa asennuksessa. [7] Kali Linux on Debian pohjainen Linux–distribuutio johon on kerätty laaja valikoima haavoittuvuustestauksen työkaluja, joista Metasploit framework on vain yksi kokoelma.

Armitage tarjoaa dynaamisen käyttöliittymän, jonka avulla hyökkääjä voi valita kohteen ja siihen kohdistettavan toteutuskelpoisen Metasploit frameworkin hyökkäysmoduulin [7]. Tämä toimintatapa mahdollistaa nopean työskentelyn, eikä vaadi hyökkääjää toteuttamaan hyökkäyksiä vain komentoriviä käyttäen, vaikkakin tämä on mahdollista *Armitagen* ohella. Kohdejärjestelmien verkkojen visualisoinnissa useiden kohteiden väliset yhteydet kuvataan *Armitagessa* tietokoneiden välille vedettyinä nuolina [4].

Armitagen käyttöliittymää on esitelty seuraavassa kuvassa 1. Kuvassa esiintyvän käyttöliittymän alareunassa hyökkääjällä on käytettävissään perinteinen Metasploit-komentorivi, joka toimii samalla tavalla kuin ilman Armitage-laajennusta. Komentorivin yläpuolella hyökkääjälle esitetään löydetyt ja jo hallinnassa olevat kohteet. Kohteen valitsemalla hyökkääjän on mahdollista valita toteutuskelpoisista hyökkäysmoduuleista.



Kuva 1. *Armitagen käyttöliittymä [7] © 2018 IEEE.*

Hyökkääjät voivat toteuttaa ryhmätyöskentelyä jakamalla *Armitage*-istunnon keskenään. Tämä mahdollistaa saadun tiedon nopean jakamisen hyökkääjien kesken sekä työskentelyn samaa kohdetta vastaan vain yhden hyökkääjän avaaman yhteyden kautta. [7] Yhteyksien määrän minimoiminen vähentää hyökkäyksen paljastumisen riskiä, mutta voidaan myös ajatella, että tämän jälkeen koko operaatio on yhden yhteyden pysyvyyden varassa.

Joitain hyökkäyksen osioita on myös mahdollista automatisoida, ja *Armitage* tarjoaa hyökkääjän käyttöön tapahtuminen automatisaatiota erilaisten bottien avulla. [7] Backdoorin avulla on mahdollista tehdä operaatioita, jotka ovat hyvin samanlaisia jokaisella hyökkäyskerralla, kuten lokitietojen poistaminen tai salasanatiedostojen kopiointi tunnetusta lähteestä. Periaatteena olisi hyvä, että automatisoidaan kaikki mikä ei vaadi ihmisen käsittelyä, jos voidaan olla varmoja, että suunnitellut proseduurit toimivat varmasti kohdeympäristössä.

8. YHTEENVETO

Kandidaatintyössä selvitettiin kirjallisuusselvityksenä ohjelmistopohjaisen backdoorin piilottamisen keinoja toimittaessa suositussa ja tunnetussa käyttöjärjestelmässä. Yleistyneet käyttöjärjestelmät motivoivat hyökkääjiä analysoimaan järjestelmien toimintaperiaatteita. Suuren suosion saanut käyttöjärjestelmä takaa uhrien määrän, mikäli järjestelmästä löytyy hyödynnettävä haavoittuvuus. Vanhojen, haavoittuneiden tunnistusprotokollien käyttäminen vaarantaa uudet järjestelmät tunnetuille hyökkäyksille.

Kerätyn tiedon perusteella onnistuneesti järjestelmään tuodulla backdoor-ohjelmalla on erittäin hyvät mahdollisuudet piiloutua tunnettujen puolustusmekanismien sekä käyttäjän näkymättömiin. Useat hyökkääjien käyttämät toiminnot ovat kehittäjien tiedossa, mutta puolustautuminen vaatii usein merkittävää päivitystä nykyiseen järjestelmään. Vanhat järjestelmien versiot tuovat hyökkääjälle mahdollisuuden levitä kohteen sisäverkossa. Haittaohjelmien ja puolustusmekanismien kehitys on jatkuvaa taistelua, mutta nykyisellään etenkin onnistuneesti asennettu backdoor on selvästi puolustuskeinoja edellä. Työssä esille tuoduista piiloutumisen keinoista toimivimmat ovat haittaohjelman salauksen lisäksi toiminnan keskittäminen tietokoneen välimuistiin sekä puolustus- ja kirjanpitoprosessien manipulaatio. Loki- ja tiedostotapahtumien muokkaus vaikeuttaa jälkeenpäin tehtyä tutkintaa merkittävästi. Haittaohjelman toiminta välimuistissa on aktiivisenkin antiviruksen tavoittamattomissa, sillä antivirusten tarkastelualue keskittyy usein vain levylle tallennettuihin ohjelmistoihin. Käyttäjän, eikä niinkään järjestelmän, harhautukseen löytyneistä keinoista merkittävimpiä ovat tiedostojen metatietojen muokkaus sekä toisen käyttäjän tekemiksi merkityt muutokset.

Työtä olisi voinut vielä jatkaa luomalla käytännön harjoitusympäristön, jossa työssä esiteltyjä ratkaisuja oltaisiin kokeiltu, mutta näin työstä olisi tullut lähes kaksinkertaisesti pidempi. Harjoitusympäristön sekä siinä tehtyjen toimien kuvaaminen olisi tuonut työhön merkittävän määrän kuvia, jolloin kandidaatintyön pituus olisi riistäytynyt hallitsemattomaksi, ellei käsitellyn materiaalin määrää olisi vähennetty. Ohjelmistopohjaisten backdoorien kannalta käsittelyn kattavuus muodostui hyväksi, mutta esimerkiksi fyysisen laitteiston backdoorien jättäminen tarkastelun ulkopuolelle jättää joitain piiloutumisen keinoja käsittelemättä.

LÄHTEET

[1] H. Agrawal, J Alberi, L, Bahler, W. Conner, J. Micallef, A. Virodov, S. R. Snyder, Preventing Insider Malware Threats Using Program Analysis Techniques, MILCOM 210 Military communications conference, 2010, 6 p.

[2] M. Bagget, Effectiveness of Antivirus in Detecting Metasploit Payloads, SANS Reading Room, 2008, 61 p. Saatavissa: https://www.sans.org/reading-room/whitepapers/casestudies/effectiveness-antivirus-detecting-metasploit-payloads-2134

[3] R. Bestak, L Kencl, L. E. Li, J. Widmer, H. Yin, Networking 2012, 11th International IFIP TC 6 Networking Conference, Prague Czech Republic, 2012, 421 p. Saatavissa: https://link-springer-com.libproxy.tuni.fi/content/pdf/10.1007%2F978-3-642-30054-7.pdf

[4] P. Bramwell, Hands-On Penetration Testing on Windows, Packt Publishing Ltd, Birmingham UK, 2018, 424 p

[5] G. Johanse, Digital Forensics and Incident Response, Packt Publishing Ltd., Birminghamn UK, 2017, 303 p.

[6] D. Kennedy, J. O'Gorman, D. Kearns, M. Aharoni, Metasploit: The Penetration Tester's Guide, No Startch Press Inc, San Francisco, 2011, 299 p.

[7] Y. Kolli, T. K. Mohd, A. Y. Javaid, Remote Desktop Backdoor Implementation with Reverse TCP Payload using Open Source Tools for Instructional Use, The university of Toledo, Toledo, OH, USA, 2018, 6 p.

[8] M. Miller, Metasploit's Meterpreter, 2004, 68 p. Saatavissa: http://www.hick.org/~mmiller/#papers

[9] RubyDoc: Class: MSF::Sessions::Meterpreter, Verkkosivu. Saatavissa: (viitattu 14.2.2019) https://www.rubydoc.info/github/rapid7/metasploit-framework/Msf/Sessions/Meterpreter

[10] L. Sha, Fu. Xiao, W. Chen, J. Sun, IIOT-SIDefenfer: Detecting and defense against the sensitive information leakage in industry IoT, Springer Science & Business Media New York, 2017, 88 p. Saatavissa: https://link-springer-com.libproxy.tuni.fi/content/pdf/10.1007%2Fs11280-017-0459-8.pdf

[11] D. Teixeira, A. Singh, M. Agarwal, Metasploit Penetration Testing Cookbook Third Edition, Packt Publishing Ltd., Birmingham UK, 2018, 388 p.

[12] C. Wysopal, C. Eng, Static Detection of Application Backdoors, Veracode inc. Burlington, MA USA, 7 p.