MAANPUOLUSTUSKORKEAKOULU

WINDOWS 10- JA DEBIAN 10 -KÄYTTÖJÄRJESTELMIEN TIETO-TURVALLISUUS TAKTISISSA PÄÄTELAITTEISSA

Pro gradu

Yliluutnantti
Jose Mäntylä

Sotatieteiden maisterikurssi 9
Kenttätykistöopintosuunta

Huhtikuu 2020

MAANPUOLUSTUSKORKEAKOULU

Kurssi	Linja
Sotatieteiden maisterikurssi 9	**Kenttätykistöopintosuunta**
Tekijä **Yliluutnantti Jose Mäntylä**	
Tutkielman nimi **Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus taktisissa päätelaitteissa**	
Oppiaine johon työ liittyy Sotatekniikka	Säilytyspaikka MPKK:n kurssikirjasto
Aika Huhtikuu 2020	Tekstisivuja 78 Liitesivuja 2

TIIVISTELMÄ

Tutkimuksen tehtävänä on selvittää Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus taktisissa päätelaitteissa. Tutkimuksessa selvitetään, mitkä ovat käyttöjärjestelmien tietoturvallisuuden peruspilarit lähtien liikkeelle käyttöjärjestelmien rakennetasolta ylöspäin päättyen kovennuksiin, joita käytetään muun muassa korkean tietoturvastandardin taktisissa päätelaitteissa. Tutkimus toteutetaan laadullisen tutkimuksen mukaisella kirjallisuusanalyysilla, Debian 10 -käyttöjärjestelmän teknisellä kokeella sekä käyttäen haavoittuvuuksien CVSS -analysointimenetelmää.

Yksikään käyttöjärjestelmä, kuten Windows 10- ja Debian 10 -käyttöjärjestelmät, eivät ole täysin turvallisia ja kaikkiin kyetään vaikuttamaan jollain jopa tuntemattomalla hyökkäysmenetelmällä. Yksittäiset laitteet, teot tai säännöt eivät yksinään riitä suojaamaan näitä käyttöjärjestelmiä kyberhyökkäyksiltä tai virheiltä. Hyökkääjä voi yrittää tunkeutua järjestelmään tiedon saamiseksi tai asentaa sisäverkkoon leviävän ja toimintaa haittaavaan haittaohjelman. Hyökkäysvektorina voidaan käyttää hyväksi järjestelmän haavoittuvuutta.

Välttääkseen tätä, taktisten päätelaitteiden käyttöjärjestelmät pyritään tekemään tietoturvalliseksi monitasoisella suojauksella sisältäen muun muassa niihin tehtävät tietoturvakovennukset, joiden tehtävänä on pienentää mahdollista hyökkäyspinta-alaa. Tavoitteena on, että järjestelmän suojaus sisältäisi monta kerrosta, jolloin yhden pettäessä on hyökkääjällä vielä monta edessä. Windows- ja Debian -käyttöjärjestelmien kehittyessä niiden kompleksisuus on lisääntynyt, jonka johdosta Windows 10- ja Debian 10 -käyttöjärjestelmiin tarvitaan aikaisempiin versioihin verrattuna yhä enemmän kovennuksia.

Ymmärtääkseen, mitä käyttöjärjestelmästä pitää koventaa on tiedettävä, millaisia haavoittuvuuksia järjestelmässä on, miten sen tietoturvallisuus vaarantuu sekä miten kyberhyökkäykset siihen toteutetaan. Ymmärryksen kasvun myötä haavoittuvuuksien vaikuttavuutta voidaan lieventää omin toimenpitein sekä kehittää kyberhyökkäyksien havainnointi- ja reagointimenetelmiä. Analysoinnin tuloksilla pitää olla vaikutusta myös siihen, kuinka esimerkiksi taktisia päätelaitteita kovennetaan ja mitä ohjelmia niihin asennetaan.

AVAINSANAT

tietoturva, tietoturvallisuus, käyttöjärjestelmä, rakenne, Windows 10 -käyttöjärjestelmä, Debian 10 -käyttöjärjestelmä, Puskurin ylivuoto, tietoturvaominaisuudet, hyökkäyspinta-ala, haavoittuvuudet, tekninen koe, CVSS -analysointimenetelmä

WINDOWS 10- JA DEBIAN 10 -KÄYTTÖJÄRJESTELMIEN TIETO-TURVALLISUUS TAKTISISSA PÄÄTELAITTEISSA

SISÄLLYSLUETTELO

WINDOWS 10- JA DEBIAN 10 -KÄYTTÖJÄRJESTELMIEN TIETOTURVALLISUUS TAKTISISSA PÄÄTELAITTEISSA

1. JOHDANTO

Tutkimuksen tarkoituksena on selvittää Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus taktisissa päätelaitteissa. Tutkimuksessa selvitetään, mitkä ovat käyttöjärjestelmien tietoturvallisuuden peruspilarit lähtien liikkeelle käyttöjärjestelmien rakennetasolta ylöspäin päättyen kovennuksiin, joita käytetään muun muassa korkean tietoturvastandardin taktisissa päätelaitteissa. Lisäksi tutkimuksessa vertaillaan käyttöjärjestelmien haavoittuvuuksien määrää ja laatua, sekä miten tietoturva on kehittynyt käyttöjärjestelmien aiemmista versioista.

M18 -johtamisjärjestelmäpäivityksen myötä Puolustusvoimien taktiset päätelaitteet päivitetään Debian -käyttöjärjestelmään (MATI2). Aiemmin käytössä olleissa MATI -päätelaitteissa oli käyttöjärjestelmänä Windows 7. Miksi laitteissa siirryttiin Debian -käyttöjärjestelmään Windows 10 sijaan ja miten tähän osaltaan vaikutti tai olisi pitänyt vaikuttaa valitun käyttöjärjestelmän tietoturva? Tutkimuksen ollessa julkinen Puolustusvoimien käytössä olevaa Debian -versiota ei mainita tutkimusraportissa.

1.1. Tutkimusintressi

Aihe on ajankohtainen ja tutkijalle myös mielenkiintoinen. Puolustusvoimien koulutuksessa ollaan siirtymässä taktisissa päätelaitteissa Debian -käyttöjärjestelmään. Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvaeroista taktisissa päätelaitteissa ei ole tehty Maanpuolustuskorkeakoululla laajamittaista tutkimusta. Tutkijan aiempi kandidaattivaiheen tutkimus liittyi kyberaseiden vaikutukseen kriittisissä tietojärjestelmissä, pro gradu -tutkimus tietoturvallisuuden aihepiiristä toimii siten jatkumona aiempaan tutkimukseen.

1.2. Tutkimustilanne

Käyttöjärjestelmien tietoturvallisuudesta on tehty kaksi pro gradu -tutkielmaa Maanpuolustuskorkeakoululla.

Ensimmäinen pro gradu -tutkielma on yliluutnantti Lauri Maskuliinin vuonna 2018 valmistunut ”BYOD-laitteiden tietoturva”, joka käsittelee rauhan ajan työssä käytettäviä päätelaitteita ja niiden tietoturvallisuutta.

Toinen pro gradu -tutkielma on yliluutnantti Ville Rantamäen vuonna 2018 valmistunut ”Taisteluosastoon kohdistuva kyberuhka”, jossa tarkastellaan, minkä tasoisen uhan potentiaalinen verkkohyökkääjä kykenisi aiheuttamaan todennäköisimmässä ja vaarallisimmassa tapauksessa hyökkäämällä taisteluosastossa sijaitsevaa reititintä, työasemaa tai tiedostopalvelinta vastaan.

1.3. Tutkimustehtävä ja tutkimuksen rakenne

Tutkimuksen tehtävänä on selvittää Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus taktisissa päätelaitteissa. Tutkimuksessa selvitetään mihin taktisten päätelaitteiden käyttöjärjestelmien tietoturvallisuus perustuu, minkä tyyppisiä uhkia ne kohtaavat ja miten Windows 10- ja Debian 10-käyttöjärjestelmät pyrkivät näitä uhkia torjumaan.

Tutkimuksen pääkysymys on:

Miten Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus toteutuu taktisissa päätelaitteissa?

Tutkimuksen alakysymykset ovat:

1. Mitkä ovat käyttöjärjestelmien tietoturvallisuuden peruspilarit?
2. Millaisia Windows 10- ja Debian 10 -käyttöjärjestelmät ovat tietoturvaominaisuuksiltaan?
3. Millaisia tietoturvakovennuksia taktisten päätelaitteiden Windows 10- ja Debian 10 -käyttöjärjestelmiin voidaan tehdä?
4. Millaisia haavoittuvuuksia Windows 10- ja Debian 10 -käyttöjärjestelmissä on?

1.4. Tutkimusmenetelmät

Tutkimus toteutetaan laadullisen tutkimuksen mukaisella kirjallisuusanalyysilla, Debian 10 -käyttöjärjestelmän teknisellä kokeella sekä käyttäen haavoittuvuuksien CVSS -analysointimenetelmää.

Kirjallisuusanalyysilla luodaan aiempiin tutkimuksiin tukeutuen teoriapohja muulle tutkimukselle ja esimerkiksi keskeiset käsitteet kyetään määrittämään sen avulla. Aineiston sisältöä analysoidaan, jonka pohjalta lukijalle muodostetaan käyttöjärjestelmien tietoturvallisuudesta ja siihen liittyvistä ilmiöstä riittävä ymmärrys.

Debian 10 -käyttöjärjestelmän tietoturvaominaisuuksien tutkimiseksi toteutettiin tekninen koe. Koe suoritettiin, koska kyseisen pääversion mukana tulevista paketeista ja tietoturvaominaisuuksista on vain vähän tietoa saatavilla muissa tutkimuksissa tai verkkolähteissä. Osasta saatavilla olevasta tiedosta ei myöskään voidaan erotella, mitkä ovat esimerkiksi Debian 9 -käyttöjärjestelmän tietoturvaominaisuuksia.

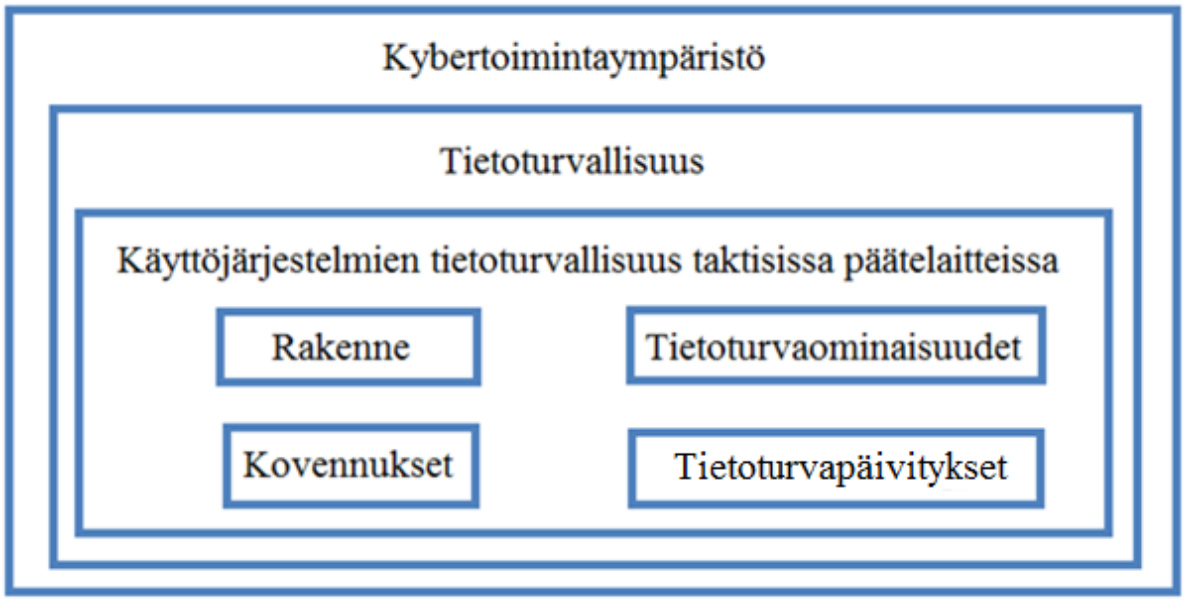
Haavoittuvuuksien laatua tutkimuksessa käsiteltävissä käyttöjärjestelmissä tutkitaan CVSS -analyysimenetelmällä. Analysointimenetelmällä tuetaan kirjallisuusanalyysin ja teknisen kokeen johtopäätöksiä. Taktisille päätelaitteille luodaan geneerinen haavoittuvuus, sekä Windows 10- ja Debian 10 -käyttöjärjestelmien haavoittuvuuksien määrää ja laatua vertaillaan taulukoilla.

1.5. Näkökulma, rajaukset ja viitekehys

Työ rajataan käsittelemään Windows 10 1903- ja Debian 10 ”Buster” -pääversioita (major). TUVE-koneissa aloitettiin vuonna 2019 siirtyminen Windows 10 -käyttöjärjestelmään ja se on myöskin laajamittaisesti käytössä oleva käyttöjärjestelmä. Toisaalta taas Microsoft markkinoi pilvipohjaista Windows 10-käyttöjärjestelmää (Windows Virtual Desktop) yhä laajemmin yrityskäyttöön. Onkin mahdollista, että käyttäjäversiota (päätelaite) kehitetään tulevaisuudessa samaan suuntaan, jolloin sen käyttäminen kovennetuissa taktisissa päätelaitteissa ja TUVE-koneissa saattaisi olla tietoturvan kannalta hyvin haastavaa. Tällöin kasvaa paine siirtyä käyttämään Debianin kaltaisia täysin isäntälaitepohjaisia käyttöjärjestelmiä.

Perusteluna Debian 10 -version käsittelylle on, että tulevaisuudessa Puolustusvoimat ja muut viranomaiset tulevat mahdollisesti siirtymään tietoturvakovennetuissa päätelaitteissa johonkin sen alaversioista (minor), jolloin sen tietoturvaa täytyy tutkia ennakkoon.

Tutkimuksessa rajataan peruskäyttäjän toimet ulkopuolelle käyttöjärjestelmän tietoturvallisuuden parantamiseksi. Näitä toimia ovat esimerkiksi palomuurin tai virusohjelman asentaminen jälkikäteen jo käytössä olevaan käyttöjärjestelmään. Tällä rajauksella pyritään siihen, että tutkimuksessa käsiteltäisiin tietoturvakovennettuja käyttöjärjestelmiä sellaisenaan, jolloin niitä kyetään objektiivisesti vertailemaan.



Tutkimuksen näkökulmana toimii käyttöjärjestelmän tietoturva taktisissa päätelaitteissa. Tietoturvan muodostumista käsitellään rakenteen, tietoturvaominaisuuksien, kovennuksien ja tietoturvapäivitysnopeuden kautta. Ylemmällä tasolla on tietoturvallisuus ja kybertoimintaympäristö.

Tietoturvallisuus tarkoittaa tässä työssä: tiedon saatavuutta, luottamuksellisuutta, eheyttä, kiistämättömyyttä sekä käyttäjän tai hänen roolin tunnistusta ja todennusta.

1.6. Lähdemateriaalin esittely ja lähdekritiikki

Tutkielman keskeisin lähderyhmä ovat aiemmat tutkimukset, oppaat ja muu kirjallisuus, verkkojulkaisut sekä tietoturvasivustot. Käyttöjärjestelmien tutkimuksessa kaksi päälähdettä on William Stallingsin teos ”*Operating System”* 9.painos sekä Andrew Tanenbaumin teos ”*Modern operating systems”* 4.painos. Teokset sisältävät paljon tietoa käyttöjärjestelmien rakenteesta sekä niiden tietoturvasta. Operating System -teoksen sisältö on paikoin vanhentunutta. Muun muassa oppaan määritys modernista käyttöjärjestelmistä on jo auttamatta vanhentunut, mutta tästä huolimatta kyseisessä teoksessa on paljon käyttökelpoista sisältöä muun muassa käyttöjärjestelmän rakenteesta.

Tietoturvallisuuden osalta tutkimuksen päälähteenä toimii Michael E. Whitmanin teos ”Principles of information security” 4.painos vuodelta 2011. Teoksessa määritellään laajasti, mitä tietoturvallisuus ja sen uhat ovat.

Modernien Windows -käyttöjärjestelmien turvallisuutta pohditaan laadukkaasti Ian Broderickin 2019 tekemässä Aalto-yliopiston diplomityössä: ”Relevance of Security Features Introduced in Modern Windows OS”. Toinen hyvin ajantasainen Windows 10 -käyttöjärjestelmää käsittelevä lähde on Michael G. Solomin: ”Security Strategies in Windows Platforms and Applications” vuodelta 2019.

Yliluutnantti Lauri Maskuliinin vuonna 2018 valmistunut pro gradu -tutkielma ”BYOD-laitteiden tietoturva” käsitteli töihin tuotavien omien laitteiden tietoturvallisuutta. Maskuliinin tutkielma on rajattu normaaliolojen päätelaitteisiin eikä esimerkiksi poikkeusoloissa käytettäviin taktisiin päätelaitteisiin. Tutkielma itsessään ei toimi suoraan lähteenä vaan lähinnä siinä käytetyt menetelmät, kuten käyttöjärjestelmien CVSS-analysointimenetelmä.

Muita tutkimuksessa käytettyjä lähteitä ovat muun muassa valmistajien omat verkkosivut sekä käyttöjärjestelmien tietoturvaa käsittelevät ja haavoittuvuuksia tilastoivat verkkosivut. Pääosa kirjallisesta lähdemateriaalista on englanninkielistä, joten käännöstyö on muodostanut suurimman osan työstä.

Päälähteinä käytettäviä oppaita voidaan pitää laadukkaina lähteinä niiden tietotekniikan alalla saaman vastaanoton ja arvostuksen johdosta. Niitä käytetäänkin aina varmentamaan muiden lähteiden tuloksia. Internet-lähteistä saatavia lähdeaineistoja (tutkimukset) joudutaan tarkastelemaan lähdekriittisemmin, mutta ne sisältävät usein ajanmukaisempaa tietoa kuin alan oppaat.

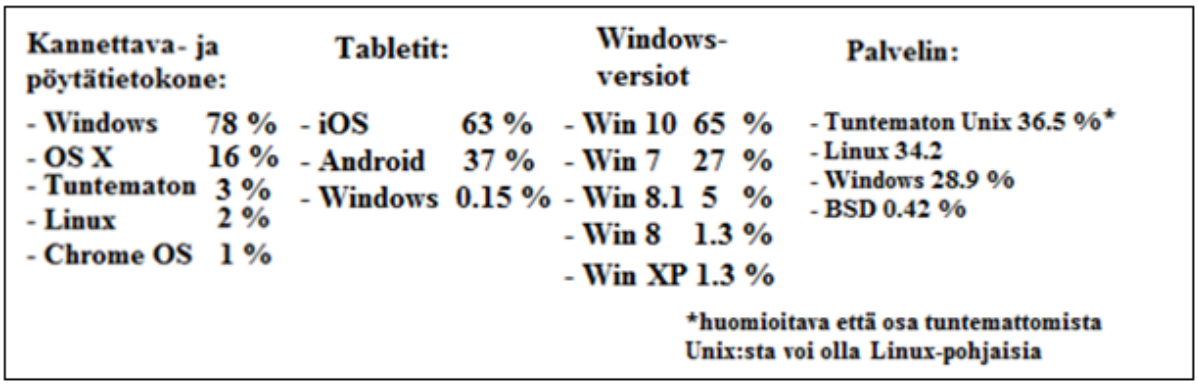
Tutkimuksen sisällöllisenä ohjaajana (2.ohjaaja) ja teknisenä asiantuntijana on toiminut Veli-Matti Visuri. Hänen roolinsa esimerkiksi käännöstyössä ja substanssiosaajana on ollut tutkimuksen onnistumisen kannalta merkittävä.

2. KÄYTTÖJÄRJESTELMÄN TIETOTURVA

2.1. Käyttöjärjestelmä

Käyttöjärjestelmä on tietokoneen keskeisin ohjelmisto toimien rajapintana laitteiston ja sovellusten välillä ja mahdollistaen täten niiden toiminnan tietokoneessa [1, s.8–9]. Käyttöjärjestelmä käynnistyy, kun tietokone laitetaan päälle, latautuen tietokoneen keskusmuistiin ja alkaen käsitellä koneella saatavissa olevia resursseja mahdollisimman taloudellisella tavalla maksimaalisen suorituskyvyn saavuttamiseksi. [2, s.11–12; 3]

Yleisimmät käyttöjärjestelmät eri päätelaitteille ovat [4, 5]:



Kuva 1 Yleisimmät käyttöjärjestelmät eri päätelaitteille [4, 5, suomennos ylil. Jose Mäntylä]

Tietokoneiden käyttöjärjestelmät ovat sovellusohjelmien käytön mahdollistavia monimutkaisia ohjelmistoja. Niiden monimutkaisuus johtuu osittain haasteesta pyrkiä vastaamaan vaikeisiin ja joissakin tapauksissa keskenään kilpaileviin tavoitteisiin: käytettävyys, tehokkuus ja konfigurointikyky. Viisi tärkeintä käyttöjärjestelmien kehittymisen ansiosta aiheutunutta kehitysaskelta ovat prosessit, muistinhallinta, tietoturvallisuus, prosessien vuoronnus sekä resurssienhallinta. [1, s.83]

Käyttöjärjestelmän tärkeimmät tehtävät ovat tehtävänhallinta, laitteistonhallinta, tiedostojärjestelmä, muistinhallinta, verkonhallinta, käyttöoikeuksien hallinta ja graafinen käyttöliittymä. [1, s.8–9; 2, s.11–12; 3]

Graafisen ja ei-graafisen käyttöliittymän tehtävänä on toimia rajapintana, jolla käyttäjä kykenee hallitsemaan käyttöjärjestelmää. Käyttöliittymä voi olla pelkkä komentorivi tai käyttäjälle visuaalinen graafinen käyttöliittymä, jossa käyttäjälle on näkyvissä komentoja sisältäviä ikoneita ja visuaalisia indikaattoreita. Käyttöjärjestelmien palveluista tiedon saatavuutta on eniten parantanut graafisten käyttöliittymien integroiminen osaksi käyttöjärjestelmiä. [6, s.405–416; 7, s.8–9]

Käyttöjärjestelmä voidaan kuvata sotilaallisen järjestelmäopin pohjalta. Järjestelmänä sille asetetaan tiettyjä tavoitteita. Sen pitää pystyä suorittamaan haluttu suorituskyky itsenäisesti (esim. johtamissovelluksen ja ammunnanhallintasovelluksen suorittaminen), kyetä kommunikoimaan muihin järjestelmiin (esim. verkkoprotokollat, laiteajurit) sekä sillä pitää olla mahdollisuus liittyä toiseen järjestelmään (esim. Linux WSL -rajapinta, Windows -koneet, Linux -emulointi). Käyttöjärjestelmä liittyy olennaisena osana tietokoneeseen (järjestelmien järjestelmä). Ilman sitä tietokone kyllä käynnistyy, mutta sillä voi tehdä vain hyvin vähän asioita.

Vuosien saatossa käyttöjärjestelmien rakenteet ja käytettävyys ovat asteittain kehittyneet. Viime vuosina on esitelty paljon uusia suunnitteluelementtejä, niin uusissa kuin päivityksinä vanhoissa käyttöjärjestelmissä, muuttaen merkittävästi käyttöjärjestelmien luonnetta. [1, s.92–97]

Modernit käyttöjärjestelmät, kuten Windows 10 ja Debian 10, vastaavat laitteisto-, sovellus- ja tietoturvakehityksen tarpeisiin. Laitteistokehityksen johdosta, moniydinprosessorit, prosessoreiden laskentanopeus, verkon siirtonopeudet, muistien kapasiteetti ja monimuotoisuus ovat lisääntyneet. Sovelluskehitys (multimediasovellukset, internet-yhteys, pääte- ja palvelinohjelmat) on aiheuttanut muutoksia käyttöjärjestelmissä. Internet-yhteyksien johdosta käyttöjärjestelmiin kohdistuneet tietoturvauhat (virukset, haittaohjelmat ja penetrointitekniikat) ovat myös merkittävästi vaikuttaneet suunnitteluun. [1, s.92–97]

Nopeasti muuttuneet käyttöjärjestelmien vaatimukset eivät pelkästään ole vaatineet muutostöitä ja parannuksia olemassa oleviin arkkitehtuureihin, vaan uusia tapoja organisoida kokonaisia käyttöjärjestelmiä. [1, s.92–97]:

Moderneissa käyttöjärjestelmissä yleistyneitä toiminnollisuuksia ovat muun muassa [9, s.21; 10]:

- Virtualisointi: yhdestä fyysisestä tietokoneesta luodaan useita virtualisoituja tietokoneita virtualisointiohjelmistojen Hypervisor-ohjelmiston avulla. Virtualisoidut koneet käyttävät fyysisen koneen tietojenkäsittelyresursseja, kuten prosessoria, muistia ja tallennustilaa. [8]
- Moniprosessoritietokoneiden käyttämissä käyttöjärjestelmissä yhtenäinen muistin käsittely on haastavaa. Tämän takia näissä käyttöjärjestelmissä (esimerkiksi Linux) on yleistynyt NUMA-muistiarkkitehtuuri (Non-Uniform Memory Access). NUMA-järjestelmässä muisti jaetaan muistisolmuihin. Keskusmuistin eri osiin on erilainen viive ja kaistanleveys riippuen siitä, mistä prosessoriytimestä muistia käsitellään, joka on sen suuri

ero symmetrisiin moniprosessorijärjestelmiin. Niissä järjestelmissä muistiviive on sama riippumatta käytettävästä suorittimesta.

2.2. Tietoturva

Tietoturva pitää sisällään järjestelyitä, joilla pyritään varmistamaan ja turvaamaan tiedon saatavuus, eheys ja luottamuksellisuus (CIA-malli) säilytyksen, prosessoinnin ja lähetyksen aikana. Tietoturvaa voidaan parantaa hyvien tietoturvakäytäntöjen, opetuksen, koulutuksen, tietoisuuden ja teknologian avulla. Käyttöjärjestelmiin liittyen tietoturvan kolmijakoa voidaan täydentää käyttäjän tunnistuksella ja todennuksella, joita käsitellään myöhemmin tässä tutkimuksessa. Nämä ovat tietoturvan toteutumisen kulmakiviä, joita käyttöjärjestelmä pyrkii suojelemaan kyberhyökkääjiltä ja salakuuntelulta. [1; 6, s.596; 11, s.8–14; 12, s.20]

Luottamuksellisuus on tiedon ominaisuus, joka ilmentää sitä, että tieto on vain sen käyttöön oikeutettujen käytettävissä eikä se paljastu muille. Luottamuksellisuuden säilyttämisen tärkeimpiä keinoja käyttöjärjestelmässä ovat pääsyn rajoittaminen ja datan salaaminen, jolloin tiedon omistaja voi määrittää, kenellä on oikeus käsitellä tiettyä tietoa. Tilanteita, joissa tiedon luottamuksellisuus menetetään ovat muun muassa tietokone-, käyttäjätieto- ja sähköpostivarkaus. [6, s.596; 13]

Eheys on tiedon ominaisuus, joka ilmentää sitä, ettei esimerkiksi käyttöjärjestelmään tallennettua tietoa kyetä omistajan tietämättä muuttamaan tai poistamaan. Väärän tiedon tallentaminen pyritään myös estämään. Tieto ei myöskään saa muuttua tahattomasti. Jos tiedon eheys vaarantuu, niin muutokset pitäisi pystyä todentamaan jälkikäteen. Tiivistäen eheä tieto on sisäisesti ristiriidatonta. [6, s.596; 13]

Tiedon saatavuus ilmentää sitä, miten tieto on hyödynnettävissä käyttöjärjestelmässä haluttuna aikana ja vaaditulla tavalla. Suurin uhkakuva tiedon saatavuuden kannalta käyttöjärjestelmissä on se, että jokin tekijä kuten esimerkiksi sovellusvirheet tai kyberhyökkäys estäisi järjestelmän käytön kokonaan. Tiedon saatavuuteen liittyy myös termi käytettävyys, jolla kuvataan järjestelmän helppokäyttöisyyttä sekä sitä, miten se soveltuu suunniteltuun tarkoitukseensa. [6, s.596; 13]

Tietoturvallisuudesta suurin osa toteutetaan käyttäjien turvallisilla toimintatavoilla. Onkin arvioitu, että tietoturvasta vain 20 prosenttia kyetään toteuttamaan tekniikalla (muun muassa:

käyttöjärjestelmävalinnat, palomuurit, virustorjunta ja pääsynvalvonta) ja 80 prosenttia on kaikkea muuta, mitä tietoturvaan liittyy kuten käyttäjien ohjeistamista sekä sitä, kenellä on pääsy mihinkin tietoon. [14]

Tiedon saatavuutta ei saa liikaa haitata eheyden ja luottamuksellisuuden varmistamisella. Tällöin esimerkiksi käyttäjä saattaa sammuttaa liian tiukkoja sääntöjä sisältävän palomuurin, jotta tietokone yhdistyy lähiverkkoon. [14]

2.3. Käyttöjärjestelmän tietoturva

On monia tapoja vaarantaa tietokoneen tietoturva. Usein menetelmät eivät ole erityisen edistyneitä, vaan heikko salasana, yksinkertainen huijaus tai haavoittuvuuden hyväksikäyttö riittää hyökkääjälle. On myös monia esimerkkejä, joissa järjestelmien tietoturva ei riitä, kun käyttäjä kadottaa luottamuksellista tietoa sisältävän muistitikun tai kiintolevyä ei tyhjennetä oikeaoppisesti. [6, s.599]

Käyttöjärjestelmän tietoturvan kannalta ei ole olennaista esimerkiksi keskittyä verkkohyökkäyksiin. Olennaista on keskittyä hyökkäysvektoreihin, joissa käyttöjärjestelmä on hyökkäyksen kohteena sekä on merkittävässä roolissa tietoturvakäytänteiden vahvistamisessa tai niiden heikentämisessä. [6, s.599]

Käyttöjärjestelmien tietoturvaa pohtiessa on ymmärrettävä, että teoriassa saattaisi olla mahdollista tehdä täysin tietoturvallinen ja vähän virheitä sisältävä käyttöjärjestelmä, mutta silloin siitä jouduttaisiin kompromissina tekemään hyvin yksinkertainen, käytettävyydeltään rajattu ja niukasti toimintoja sisältävä järjestelmä. Pelkkä käyttöjärjestelmän suunnittelu ei riittäisi, vaan myös käytettävä laitteisto täytyisi suunnitella ja valmistaa täysin tietoturvavaatimusten pohjalta. Kaiken kompleksisuuden hallinta täysin tietoturvallisen järjestelmän rakentamiseksi olisikin äärimmäisen vaikeaa, ellei jopa mahdotonta. [6, s.600–601]

Puolustusvoimien kaltaisten viranomaisten kannalta tietoturvallisuus on tärkeämpi kriteeri kuin uudet ja käytettävyyttä parantavat toiminnot, mutta kaupallisesti marginaalisten käyttöryhmien tarpeet eivät toimi riittävänä kannustimena tämänkaltaisten käyttöjärjestelmien kehittämiseen ja ylläpitoon. [6, s.600–601]

Monet maat kehittelevätkin omia yleisesti käytössä olevia käyttöjärjestelmiä turvallisempia vaihtoehtoja. Tästä esimerkkinä voidaan pitää Venäjän asevoimien tarvetta varten kehitetty Astra Linux. Astra Linux on Debian -käyttöjärjestelmän pohjalta tietoturvalliseksi suunniteltu

käyttöjärjestelmä, jolla voi Venäjän asevoimissa käsitellä jopa kaikista arkaluontoisimmaksi määritettyä tietoa. [15, 16]

2.4. Tietoturva päätelaitteissa

Päätelaitetasolla tietoturvallisuus koostuu käyttöjärjestelmien tietoturvaominaisuuksista ja turvaohjelmistoista. Niiden tehtävä on muun muassa [17, s.21]:

- eriyttää eri ohjelmat ja niiden suorittaminen toisistaan
- eriyttää eri ohjelmien käsittelemät tiedot toisistaan
- salata päätelaitteelle tallennetut tiedot (tai päätelaitteen massamuistit)
- havaita, mikäli tietoja on yritetty käsitellä tai käsitelty oikeudettomasti
- rajoittaa, käyttäjien toimia järjestelmässä kuten ohjelmien asennuksia ja turva-asetusten muutoksia sekä osioida kiintolevy virtuaalisiin osiin, joihin tietyillä käyttäjillä on vain pääsy
- rajoittaa pääkäyttäjien toimia järjestelmässä
- havaita tai estää haittaohjelmien toimintaa esimerkiksi käyttäjältä vaadittavien vahvistuksien avulla
- estää tai havaita väärinkäyttö tai tietojen oikeudeton käyttö esimerkiksi lokitusten tai turva-asetusten avulla.

Päätelaitteen tietoturvallisuutta voidaan heikentää merkittävästi käyttäjän tahallisilla tai tahattomilla toimilla esimerkiksi nostamalla käyttäjän oikeuksia haavoittuvuuden avulla. Korotetuilla oikeuksilla käyttäjän on mahdollista esimerkiksi asentaa sovelluksia muistakin kuin hyväksytyistä lähteistä. Tietoturvallisuus saattaa samalla heiketä myös muilla tavoin esimerkiksi mahdollistamalla uusia verkkoa kuuntelevia tai verkkoon tietoja lähettäviä palveluita. [17, s.23]

Päätelaitteissa saattaa olla myös käyttöjärjestelmän tietoturvallisuutta heikentäviä ominaisuuksia, kuten avoimia palveluita tai haavoittuvuuksia. Näitä voi olla esimerkiksi hallinta-, valvonta- ja tukipalveluissa sekä mahdollisissa laite- ja ohjelmistovalmistajien pilvi-, tunnistus- ja tallennuspalveluissa. Päätelaitteiden ja niihin liittyvien palveluiden käyttöön liittyvät riskit kartoitetaan auditoinneilla. Tietoturvallisuuden heikentyminen vaikuttaa päätelaitteilla käytettävien palveluiden luottamuksellisuuteen, eheyteen ja saatavuuteen. [17, s.23]

2.5. Tunkeutujat ja haitalliset ohjelmat

Käyttöjärjestelmiin kohdistuu lukemattomia uhkia, joista tutkimuksen viitekehys huomioiden merkittävimpiä ovat tunkeutuminen, haittaohjelmat, sisäpiirihyökkäykset sekä hyökkäyksen mahdollistavat haavoittuvuudet. [1, 6]

2.5.1 Tunkeutujat

Käyttöjärjestelmiin tunkeutujat voidaan kategorioida motivaatioiden mukaisesti: varkaisiin, haktivisteihin, vandaaleihin, terroristeihin, valtiollisiin tahoihin, vakoilijoihin ja kiristäjiin. Tunkeutujalla on yleensä tavoitteena saada haltuun käyttöjärjestelmä tai vain kasvattaa omia käyttöoikeuksiaan käyttöjärjestelmässä. [1, s.658–659; 6 s.599; 18, s. 657–661]

Tunkeutumisen motiivina voi olla esimerkiksi raha, tieto, järjestelmän lamauttaminen tai laitteen hyväksi käyttäminen muiden järjestelmien lamauttamiseksi. Yksinkertaisimmat hyökkäykset käyttävät hyväksi järjestelmien ja ohjelmien haavoittuvuuksia, jotka mahdollistavat haittaohjelman suorittamisen. Haittaohjelma voi esimerkiksi avata takaoven, eli keinon ohittaa järjestelmän normaalin todentamisen tai salauksen tunkeutujalle. Takaovi voi syntyä myös esimerkiksi tahallisesti heikennetystä salausalgoritmin parametristä. Tunkeutumisessa voidaan käyttää hyväksi myös sovelluksien puskurin ylivuotovirhettä sekä heikosti suojattuja tai paljastuneita kirjautumistietoja. [1, s.658–659; 6, s.599; 18 s. 657–661]

2.5.2 Haittaohjelmat

Haittaohjelmia ovat esimerkiksi virukset, madot, troijalaiset sekä näiden yhdistelmät. Haittaohjelmat pyrkivät sisään järjestelmään tehden siellä käyttäjän tietämättä jotain, mikä voidaan olettaa haitalliseksi käyttäjälle tai järjestelmälle kuten vuotamalla tietoa tai aiheuttamalla muuten ei-toivottuja tapahtumia. [1, s.31; 19, s.3; 20, s.137; 21]

Käyttöjärjestelmän tietoturvallisuuden kannalta kriittisimmät haittaohjelmat kohdistuvat järjestelmäsovelluksiin sekä käyttöjärjestelmän ydintason ohjelmiin. Monimutkaisimpia haittaohjelmia ovat monimuotoiset ja yhdistelmämadot, jotka voivat käyttää montaa eri hyökkäysvektoria hyödyksi haavoittuvuuksien löytämiseksi sekä tunkeutumisen toteuttamiseksi. [1, s.659; 6, s.660–662; 12, s.91]

Kohdennetut haittaohjelmahyökkäykset (APT) ovat monivaiheisia kyberhyökkäyksiä, jotka kohdistuvat tiettyyn rajattuun kohdeympäristöön tai -järjestelmään ja jotka tehdään haittaohjel-

mien sekä muiden toimintojen avulla. Tavoitteena voi olla saada kohteesta kriittistä tietoa haltuun, muuttaa kohteen toimintaa tai aiheuttaa jopa fyysistä vahinkoa. Tekijöinä ovat yleensä hyvin rahoitetut tahot, kuten esimerkiksi valtiot tai organisoidut rikolliset. [11, s.31; 22, s.22]

APT alkaa usein kattavalla kohdetiedustelulla, jota seuraa jopa yksilöllisesti suunniteltu haittaohjelmahyökkäys, jolla pyritään sisään järjestelmään. Päästyään sisään järjestelmään kyberhyökkääjät pyrkivät etenemään ja tarvittaessa kasvattamaan käyttöoikeuksiaan kohdejärjestelmässä päästäkseen tavoitteeseensa. Kyberhyökkääjät eivät välttämättä halua, että heidän toimintansa huomattaisiin, joten kyberhyökkäyksen päästessä tavoitteeseen sen jäljet voidaan lopuksi poistaa käyttöjärjestelmästä. [11, s.31; 22, s.22]

Haittaohjelmat, kuten käynnistyshaittaohjelmat voidaan jakaa kolmeen kategoriaan toimintatasosta riippuen:

Käyttäjätaso (User space, taso 3): käyttäjätason haittaohjelmat ovat kaikkein yleisimpiä ja helpoimpia toteuttaa, mutta samalla ne ovat myös helpoiten torjuttavissa. Niiden tavoitteena voi olla saada tai muokata käyttäjätason ohjelmien ja jaettujen kirjastojen tietoa. [23, 24]

Ydintaso (Kernel space, taso 0): Ydintason haittaohjelmat ovat kaikkein kriittisimpiä haittaohjelmia, koska ne toimivat ytimen muistiavaruudessa. Ydintasolla toimiva haittaohjelma voi vaikuttaa suoraan käyttöjärjestelmän ja ajureiden toimintaan, ja siten esimerkiksi piilottaa itsensä tiedostojärjestelmästä ja ohittaa tietoturvaohjelmistot. [19, s.36; 23]

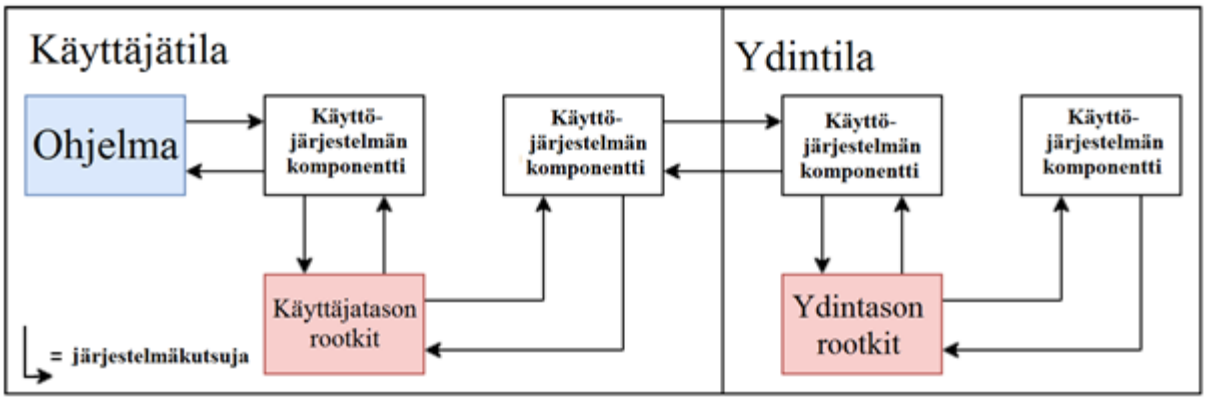
Virtualisointitaso (taso -1): laiteohjelmistotason haittaohjelmat toimivat järjestelmän alhaisimmalla tasolla, jossa virtuaalimonitori Hypervisor suorittaa virtuaalikoneita. Saastuneen käyttöjärjestelmän ydin ei havaitse, että se ei ole yhteydessä oikeaan laitteistoon vaan haittaohjelman muuttamaan ympäristöön. [23]

Ylemmällä tasolla toimivat haittaohjelmien torjuntaohjelmat eivät kykene havaitsemaan omaa tasoa alempien tasojen haittaohjelmia, joten tavoitteena on, että haittaohjelmatorjunta olisi mahdollisimman alhaisella tasolla. [23]

Käyttöjärjestelmät ovat haavoittuvaisimmillaan käynnistyksen aikana, koska tällöin suojaustoiminnot eivät ole vielä ehtineet käynnistyä. Rootkitit ja bootkit -käynnistyshaittaohjelmat yrittävät hyväksikäyttää tätä järjestelmän vaihetta. [23, 25]

Rootkitit ovat haittaohjelmia, jotka piiloutuvat käyttäjältä ja/tai käyttöjärjestelmältä. Tavoitteena Rootkitillä voi olla saada käyttöjärjestelmä osittain tai kokonaan käyttöön ilman, että

käyttäjä havaitsee sitä. Päästyään järjestelmään ne pyrkivät piilottamaan itsensä. Tähän pääs-tääkseen ydintason rootkitit saattavat toimia ajureina tai käyttäjätason rootkitit pelkästään pii-lottavat toiminnallisuuttaan sekä mahdollisesti yrittävät estää sen poistamisen. Tähän rootkitit pyrkivät esimerkiksi pääsemällä kiinni käyttöjärjestelmän API-kutsuihin (inline hooking). Ta-voitteena Rootkitillä voi olla luoda takaovi muille haittaohjelmille tai etähallinnalle. [19, s.51–52; 23; 26, s. 10]



Kuva 2 Käyttäjä- ja ydintilan rootkitit [26, s.12, suomennos ylil. Jose Mäntylä]

Bootkitit ovat haittaohjelmia, jotka ottavat kohteeksi kiintolevyn ensimmäisen sektorin (master boot record eli MBR), jolta BIOS yrittää etsiä käyttöjärjestelmän latauskoodia. Bootkit -haitta-ohjelmat pyrkivät kontrolloimaan kaikkia käyttöjärjestelmän käynnistysvaiheita, tavoitteena muokata järjestelmäkoodia ja/tai ajureita. Niiden havaitseminen on vaikeaa, koska bootkit-hait-taohjelmat ladataan järjestelmään ennen haittaohjelmatorjunnan käynnistymistä, ja sen kom-ponentit sijaitsevat käyttöjärjestelmän tiedostojärjestelmän ulkopuolella. Tavoitteena hyökkää-jällä on esimerkiksi aiheuttaa järjestelmän epävakautta, vakoilla käyttäjää tai jopa estää järjes-telmän käyttö ilman lunnaiden maksua (ransomware). [25]

Bootkit -tartuntojen vaikuttavuus on moderneissa järjestelmissä poistunut UEFI:n yleistyessä BIOS:n seuraajana. Tämä on mahdollistanut emolevyjen turvatoimintojen kuten Secure boot -teknologian (ks. luku 2.4.7) käyttöönoton. [25]

2.5.3 Sisäpiirihyökkäykset

Tunkeutuminen saattaa tulla myös organisaation sisältä. Tähän on hyvin iso riski esimerkiksi isoissa organisaatioissa. Tämänkaltaisen hyökkäyksen kykenee toteuttamaan esimerkiksi asen-nuspalvelimen ohjelmoija tai käyttöjärjestelmiä päätelaitteeseen asentava asentaja. Näitä ”sisä-piirihyökkäyksiä” (insider attacks) ovat muun muassa loogiset pommit (logic bombs) ja taka-ovet (back doors). [6, s.657–660]

Looginen pommi on haittaohjelma, joka suoriutuu itsekseen, kun tietty tai tietyt ehdot täyttyvät. Tällainen haittaohjelma saattaisi olla esimerkiksi asennettuna taktiseen päätelaitteeseen ja se aktivoituisi, kun jokin päätelaitteen johtamisohjelmista saisi tärkeän viestin ja siten tyhjentäisi kiintolevyn tai salaisi tärkeitä tiedostoja. Looginen pommi saattaisi kohdistua myös paikkatietosovelluksiin alkaen vääristää päätelaitteen GPS:n paikkatietoa tietyn päivämäärän jälkeen tietyllä alueella. [6, s.657–658; 27, s.1]

Käyttöjärjestelmän haavoittuvuus saattaa syntyä ohjelmointivirheistä, yhteensopivuusongelmista sekä konfiguraatiovirheistä, mutta tahallisesti kyse on takaovesta. Sisäpiirihyökkäyksessä järjestelmään voidaan asentaa takaovi tai se voidaan tahallisesti jättää järjestelmään. Järjestelmään voidaan avata takaovi myös siihen kykenevällä haittaohjelmalla. [19, s.3; 20, s.137; 21]

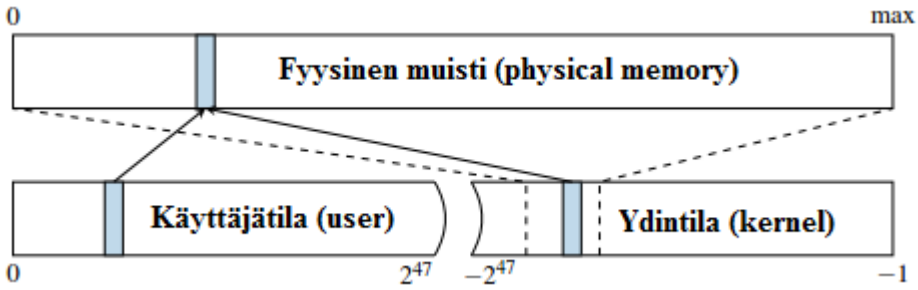
Takaovi voidaan luoda esimerkiksi tahallaan heikennetyn salausavaimen avulla. Takaoven avulla ohjelmoija voisi tehdä esimerkiksi kirjautumisen tietyllä käyttäjänimellä mahdolliseksi ilman salasanaa. Yksi keino havaita takaovia on lähdekoodikatselmoinnit, joissa lähdekoodi tarkastetaan muiden toimesta. [6, s.658–659]

2.5.4 Meltdown- ja Spectre -hyökkäykset

Meltdown- ja Spectre ovat universaaleja ja järjestelmäriippumattomia hyökkäyksiä. Ne hyväksikäyttävät prosessoreiden ominaisuutta, jossa suorituskykyä pyritään tehostamaan ennakoimalla käskyjen suoritusta (branch prediction). Ennakoinnissa käskysarjan haarautuessa prosessori pyrkii ennakoimaan todennäköisimmän polun ja valmistelemaan sen suorituksia, samalla hakien koodin tarvitsemat muistiviittaukset valmiiksi. Ominaisuus mahdollistaa kuitenkin haavoittuvuuden, jonka avulla hyökkääjä kykenee ohittamaan prosessoreiden tietoturvan kannalta kriittisiä muistinsuojauksia. [28, 29, 30]

Käyttäen hyväksi muistiviittauksien käsittelyssä käytettävän oikeustarkastuksen puutetta sekä ennakoivaa suoritusta, mahdollistuu Meltdown-hyökkäys. Hyökkäyksen avulla kyetään rikkomaan eri muistiosoiteavaruuksiin eristämisen (käyttäjä- ja ydintason) tuoma suoja, jolloin hyökkääjällä on ohjelmointirajapinnan kautta pääsy käyttöjärjestelmän ytimeen. [28, 29, 30]

Meltdown -hyökkäys mahdollistaa ytimen muistin ja muiden sovelluksien muistin lukemisen tarkastelemalla, luettiinko muistialue välimuistiin ennakoivan suorituksen aikana, mahdollisesti paljastaen muistista esimerkiksi salasanoja, salausavaimia tai muita luottamuksellisia tietoja. [28, 29, 31]



Kuva 3 Fyysinen osoite (sininen) johon on pääsy sekä käyttäjä- että ydintilalla ennakoivan suorituksen johdosta [29, s.5]

Meltdown -haavoittuvuudet johtuvat suureksi osaksi laitteistotason ongelmista. Tämä tuottaakin suurimman haasteen niiden paikkaamisessa. Esimerkiksi tietyt prosessorit yhdistettynä päivittämättömään käyttöjärjestelmään muodostavat yhdessä tämän kriittisen haavoittuvuuden. Suurimmat riskit Meltdown-haavoittuvuutta käyttäville hyökkäyksille on virtuaalipalvelimissa (esimerkiksi konesaleissa) ja monikäyttäjäpäätelaitteissa (esimerkiksi TUVE), joissa kyberhyökkääjä voi saada monen käyttäjän tiedot haltuunsa yhdeltä laitteelta. [29, 31]

Spectre -hyökkäys rikkoo sovelluksien muistialueiden välisen eristyksen tarjoten hyökkääjälle mahdollisuuden saada niistä luottamuksellista tietoa. Hyökkääjä pyrkii ajamaan kohdejärjestelmässä täsmälleen samanlaista komentoa kuin jossakin kohteeksi valitussa prosessissa, kyeten tällöin vaikuttamaan myös kohdeprosessin ennakoivaan suorittamiseen. [28, 30, 31]

Ennakoivan suorituksen aikana käsitellään muistialueita, joihin ohjelma ei tavallisen suorituksen aikana menisi. Hyökkääjä kykenee hyökkäyksen avulla lukemaan luottamuksellista muistia tarkastelemalla muistinkäsittelyn ajoituksia tai käyttäen muita olemassa olevia sivukanavia. [28, 30, 31]

Spectre -haavoittuvuutta on vaikeampi hyödyntää kuin Meltdown -haavoittuvuutta, mutta toisaalta taas toteutuessaan ongelman rajoittaminen on haasteellisempaa. Hyökkääjän haasteena on se, että ennakoivan suorituksen yksityiskohtainen toteutus vaihtelee eri prosessorimalleissa. [28, 30, 31]

Tärkein keino paikata Spectre -haavoittuvuuksia ovat ohjelmistopäivitykset, joilla on kyetty estämään esimerkiksi sivukanavien käyttö (Windows 10 Out of Band -päivitys ja Linux Kernel KPTI). Monesti tämänkaltaisilla päivityksillä on ollut negatiivista vaikutusta suorituskykyyn riippuen käytetystä prosessorista ja työkuorman luonteesta. Ne ovat lisäksi aiheuttaneet uusia haavoittuvuuksia. Käyttöjärjestelmien (mukaan lukien Windows- ja Debian) suojaamisen lisäksi tietokoneen emolevy pitää päivittää uudella laiteohjelmistopäivityksellä. [29, 30, 31]

Tietokoneen Meltdown- ja Spectre- lievennysten tila ja sitä kautta käyttöjärjestelmän haavoittuvuus voidaan tarkistaa siihen tarkoitetuilla ohjelmilla, tai esimerkiksi Windows 10:ssä siihen Microsoftin julkaisemalla PowerShell-komentosarjalla. Muita keinoja lieventää riskejä on käyttää luotettuja ja tarkastettuja sovelluksia (jotka on käännetty ja linkitetty Meltdown ja Spectre mitigoituja kirjastoja vasten), suojata sensitiiviset ohjelmat sekä ajaa tuntemattomia ohjelmia tietoturvatestaukseen tarkoitetussa virtuaaliympäristössä eli hiekkalaatikossa. [29, 32]

2.5.5 Puskurin ylivuoto

Keskus- ja virtuaalimuisti ovat käyttöjärjestelmän haavoittuvaisimmat järjestelmäresurssit tietoturvauhille. Olennainen tämänkaltaisten uhkien torjuntakeino on estää luvaton pääsyoikeus prosessien muistialueeseen. Prosessit ovat keskusmuistiin ladattuja ohjelmia, joiden suoritusta käyttöjärjestelmä hallitsee. [1, s.662; 33; 34, s.7; 35, s.36]

Jos prosessi ei ole ilmoittanut muistialuettaan jaettavaksi, toisilla prosesseilla ei saisi olla pääsyä muistialueella oleviin tietoihin. Prosessin ilmoittaessa osan muistialueesta jaettavaksi tiettyjen prosessien kanssa, käyttöjärjestelmän tietoturvapalvelun tehtävä on pitää huolta siitä, että vain näillä prosesseilla on siihen pääsyoikeus. [1, s.662; 33; 34, s.7]

Puskuri on väliaikainen muistialue, jonka ohjelma varaa käsitelläkseen tietoa palasina. Puskureita käyttämällä pyritään tehostamaan suorituskykyä ja/tai minimoimaan sovelluksen muistinvarausta. Puskurin ylivuoto on virhetilanne, jossa prosessin indeksi ei enää osoita sille varatulle muistialueelle, jolloin ohjelma kirjoittaa puskuriksi varatun muistin ulkopuolelle. [1; 34, s.7; 35, s.35–37; 36, s.23; 37]

Puskurin ylivuotovirheet ovat yleinen tietoturvaongelmia aiheuttava yksittäinen syy käyttöjärjestelmissä. Ylivuotovirheet kohdistuvat erityisesti matalan tason ohjelmointikieliin, jotka kääntyvät suoraan prosessorin kääntäjän (assembler) avulla binääriluvuiksi. Näitä ovat esimerkiksi C ja C++ -ohjelmointikielillä kirjoitetut ohjelmat. [1, s.662–663; 36, s.23; 37]

Suurin osa Windows 10:n ytimen suljetusta lähdekoodista on kirjoitettu C-ohjelmointikielellä ja pieni osa Assemblyllä. Muut komponentit siitä ylöspäin on kirjoitettu C++ ja C# -ohjelmointikielillä. Debianin Linux-ydin on kirjoitettu C-kielellä ja kokonaisuudessaan Debian 10 -käyttöjärjestelmästä noin 38 prosenttia on kirjoitettu C-ohjelmointikielellä ja 25 prosenttia C++-ohjelmointikielellä. [38, 39, 40]

Matalan tason ohjelmointikielillä saavutetaan mahdollisimman optimaalinen järjestelmäresurssien käyttö, verrattuna esimerkiksi järjestelmäresursseja enemmän vaativiin ja muistiturvallisempiin; tulkattavaan Python- sekä virtuaalikoneessa ajettavaan Java -ohjelmointikieliin. C ja C++ -ohjelmointikielet eivät sisällä taulukon raja-arvojen tarkastusominaisuutta ilman että sitä erikseen koodataan. Tämän lisäksi käyttäjä kykenee matalan tason ohjelmointikielillä käsittelemään käyttöjärjestelmän rajoitteissa suoraan muistia, jonka johdosta näillä ohjelmointikielillä kirjoitetut ohjelmistot ovat erityisen alttiita määrittelemättömille tiloille kuten ylivuodoille. [6, 35, 41]

Puskurin ylivuotojen sijainneista ja koosta riippuen niitä ei välttämättä havaita. Tällöin muistissa oleva data saattaa esimerkiksi korruptoitua, tai siinä esiintyy odottamattomia tiloja sekä prosessit päättyvät epänormaalisti. [1, s.662–663; 35, s.35–37; 36, s.23; 37]

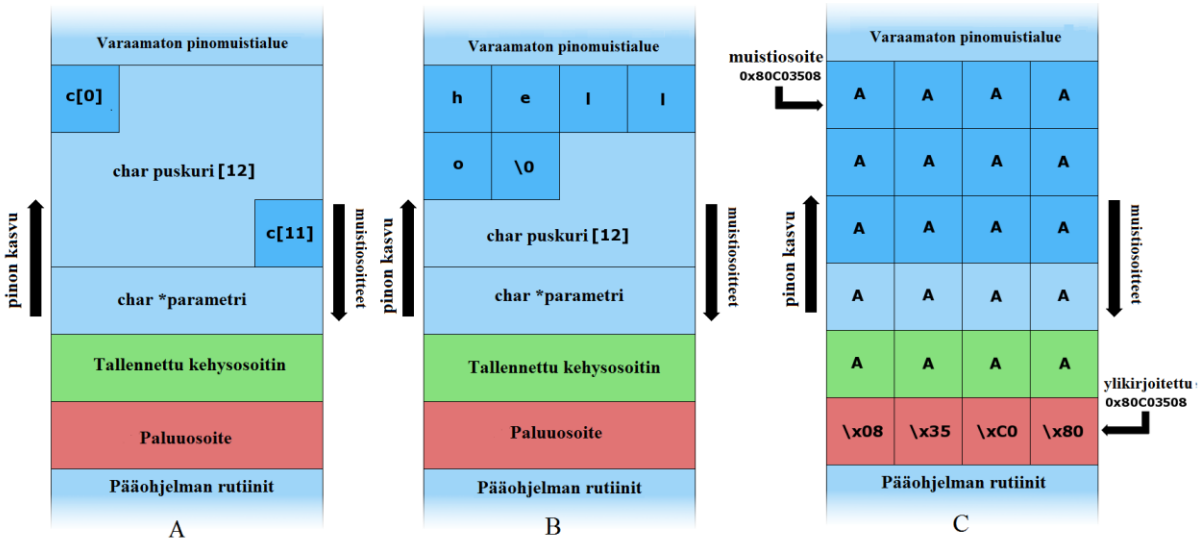
Paljastaakseen puskurin ylivuotovirhehaavoittuvuuden hyökkääjä pyrkii [1, s.665]:

- tunnistamaan puskurin ylivuotovirheen aiheuttaman haavoittuvuuden käyttöjärjestelmässä suoritettavasta sovelluksesta, käyttäen hyväksi järjestelmästä hankittua dataa
- jäljittämällä sovelluksen alkuperäislähteen
- käyttäen hyödyksi työkaluja, jotka generoivat syötteitä sovelluksille tunnistaakseen mahdollisesti haavoittuvaiset funktiot (Fuzzing)
- ymmärtämään miten ja millaisia puskureita ohjelma käyttää

Seuraavassa koodissa ja kuvassa 2 esitellään puskurin ylivuotovirhettä. Koodi ottaa argumentin komentoriviltä ja kopioi sen paikalliseen pinomuuttujaan ”puskuri”, funktiossa ”ylivuotava_funktio”, joka toimii oikein kaikilla alle 12 merkin mittaisilla argumenteilla (kuten voit nähdä kuvan 4 B kuviossa alapuolella). Kaikki yli 11 merkin mittaiset argumentit aiheuttavat pinon korruption, koska tällöin ylimääräinen data valuu sille kuulumattomalle muistialueelle. [42]

```
#include <string.h>

void ylivuotava_funktio(char* parametri)

(
        // varataan kaksitoista merkkiä pitkä puskuri

        char puskuri[12];

        // kopioidaan puskuriin argumenttitaulukosta arvo tarkamastamatta
        // mahtuuko se puskuriin

        strcpy(puskuri, parametri);

        // tässä kohti funktion paluuosoitin voidaan ylikirjoittaa
```

```
int main(int argumenttimäärä, char **argumentit)
{

// viitataan argumenttitaulukkoon indeksillä yksi, eli komentoriviohjel-
missa ensimmäinen käyttäjän syöttämä argumentti ja kutsutaan sillä yli-
vuotavaa funktiota

ylivuotava_funktio(argumentit[1]);

// poistutaan pääohjelmasta
   return 0;
}
```



Kuva 4 Puskurin ylivuotovirhe [42, suomennos ylil. Jose Mäntylä]

Ylhäällä olevasta kuvasta C näkyy, kun yli 11 -merkkinen argumentti syötetään funktiolle ”ylivuotava_funktio” joka ylikirjoittaa paikallispinon dataan (local stack data), tallennetun kehysosoittimen (frame pointer), ja tärkeimpänä funktion paluuosoitteen (return address). Kun ”ylivuotava_funktio” palaa, ohjelma hyppää hyökkääjän ylikirjoittamaan paluuosoitteeseen. Näin hyökkääjä on ylikirjoittanut paluuosoitteen hyväksikäyttäen puskurin ylivuotoa, ja täten voi hypätä haluamaansa funktioon ohjelmassa. [42]

Puskurin ylivuodon hyväksikäyttö voi mahdollistaa shellcode-hyökkäyksen (haitallinen käsky- tai komentosarja) alustaan ja haluttuihin funktioihin. Jos kohdeohjelmalla on pääkäyttäjäoikeudet, hyökkääjä saattaa saada ne haltuunsa. [42]

Käyttöjärjestelmät ja sovellukset suojataan puskurin ylivuotovirheiltä seuraavilla keinoilla [1, s.662–670; 43; 44]:

- käyttöjärjestelmän ja sovelluksien tietoturvallinen koodaaminen saavutetaan laadunvarmistuksilla, joilla tarkastetaan koodin yhteensopivuus asetettujen tietoturvavaatimusten kanssa
- käyttämällä lähdekoodin staattiseen analysointiin tarkistusohjelmia (linter), jotka varoittavat ohjelmointivirheistä, tyylivirheistä, epäilyttävistä rakenteista, usein toistuvista koodirimpsuista (copypaste detection) sekä korkeasta syklomaattisesta kompleksisuudesta eli itsenäisten suorituspolkujen määrästä (cyclomatic complexity)
- käyttämällä yleisesti turvalliseksi arvioituja koodikirjastoja ohjelmointityön pohjana
- päivittämällä haavoittuvien sovelluksien muistinhallinnan suojauksia muuttamalla muistialueiden sijaintia ja ominaisuuksia, tai tunnistamalla ennakkoon ylivuotohyökkäyksen kohteena oleva puskuri
- estämällä ohjelmia ja käyttöjärjestelmiä käyttämästä muistialueiden osia, jotka pysyvät reservinä
- satunnaistamalla keskeisten tietorakenteiden sijaintia osoiteavaruudessa jokaisessa prosessissa, jolloin hyökkääjä ei kykene ennustamaan niitä (ASLR)
- asettamalla prosessin kriittisten muistialueiden väliin suoja-alueita (guard pages), joihin hyökkääjä pyytää oikeutta, jolloin suojaus kaataa prosessin ennen ylivuotovirheen toteutumista
- käyttämällä tietojen suorittamisen estämistoimintoa (DEP), joka valvoo ohjelmia ja varmistaa, että ne käyttävät järjestelmämuistia turvallisesti. Jos tietojen suorittamisen estämistoiminta havaitsee, että jokin tietokoneen ohjelma käyttää muistia väärällä tavalla, se sulkee ohjelman ja ilmoittaa havainnosta. Toiminto estää monien ylivuotovirhettä hyväksikäyttävien haittaohjelmien toiminnan ja muita vastaavia turvallisuusuhkia.

2.6. Vastakeinot ulkoisia uhkia vastaan

Yksittäiset laitteet, teot tai säännöt eivät yksinään riitä suojaamaan järjestelmää kyberhyökkäyksiltä tai virheiltä. Välttääkseen tätä modernit käyttöjärjestelmät pyritään tekemään tietoturvalliseksi monitasoisella suojauksella (defense in depth). Tavoitteena on, että järjestelmän suojaus sisältäisi monta kerrosta, jolloin yhden pettäessä on hyökkääjällä vielä monta edessä. [6, s.684; 45, s.37]

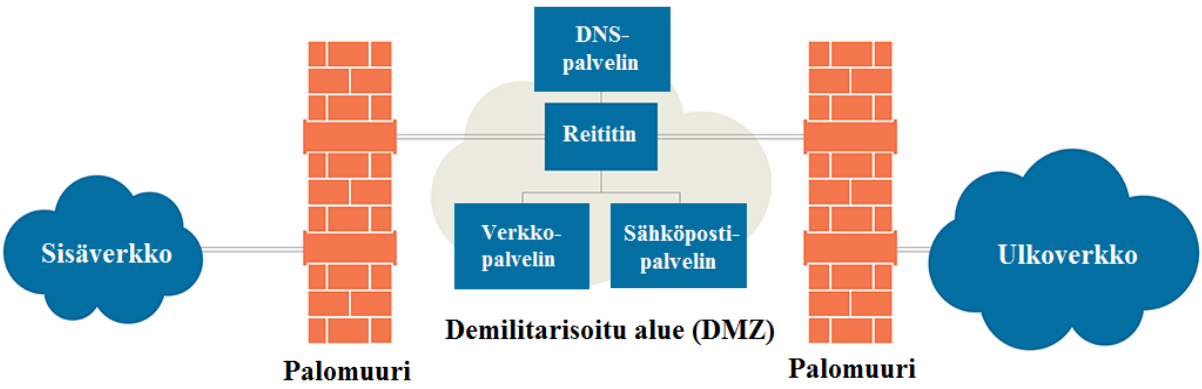
Ylläpitääkseen tietoturvallisuutta käyttöjärjestelmissä, on tärkeää ymmärtää, miten niitä vastaan voidaan hyökätä. Mutta vielä tärkeämpää on ymmärtää, miten ne voidaan suojata esimerkiksi tiedon luottamuksellisuuden menettämiseltä. Seuraavissa alaluvuissa esitellään näitä suojausmenetelmiä. [45, s.37]

2.6.1 Palomuuri

Palomuuri on verkkolaite tai ohjelmisto, jonka tehtävänä on suojata järjestelmää säätelemällä verkkoporttien ja sovelluksien läpi liikkuvaa dataa sekä rajata sisäverkon näkyvyyttä ulkoverkkoon. Palomuureja luokitellaan prosessointitavan, aikakauden ja rakenteen perusteella eri kategorioihin. [1, s.344–351; 46]

Palomuurin tavoitteena on rajoittaa liikennettä verkkosegmenttien väliltä ennalta määriteltyjen sääntöjen mukaisesti, joilla suojataan DMZ:n palveluita tai sisäverkon järjestelmiä ja päätelaitteita haitalliselta liikenteeltä. Palomuurilla voidaan myös yrittää estää luottamuksellisen informaation vuotaminen ulos sisäverkosta. [1; 6, s.685]

Palomuureilla kyetään tekemään demilitarisoituja alueita eli DMZ:tä. DMZ on konsepti, jossa fyysinen tai looginen aliverkko muodostetaan turvattomaan ulkoverkkoon kuten esimerkiksi internettiin. Tämä sisäverkosta erillinen verkkoalue, jonka laitteet eivät pysty muodostamaan suoria yhteyksiä sisäverkon laitteisiin, sisältää ulkoverkon palveluita tarjoavat laitteet kuten verkko-, sähköposti-, DNS-palvelimet (nimipalvelujärjestelmä). Tämän ansiosta sisäverkko on suojattu, vaikka esimerkiksi sähköpostipalvelimeen onnistuttaisiin hyökkäämään. Demilitarisoitu alue voidaan toteuttaa monella eri tavalla, kuten yhdellä tai kahdella palomuurilla sekä niiden yhdistelmällä. [47, s.296–297; 48, 49]



Kuva 5 Kahden palomuurin DMZ-alue [49, suomentanut ylil. Jose Mäntylä]

2.6.2 Haittaohjelmien torjunta

Palomuurit ja tunkeilijan havaitsemis- ja estojärjestelmät yrittävät pitää tunkeutujat pois tietokoneelta, mutta ne saattavat epäonnistua monella tapaa. Seuraavana suojaustasossa tunkeutujia vastassa ovat haittaohjelmien torjuntaohjelmat (virustorjunta), jotka ovat integroituna käyttöjärjestelmiin tai erikseen asennettavia. Torjuntaohjelmat pyrkivät havaitsemaan haittaohjelman käyttöjärjestelmästä haittaohjelmatarkastuksilla ja analysoimalla ohjelmien toimintaa sekä eheyttä. [6, s.687]

Haittaohjelmatarkastuksessa torjuntaohjelmat käyttävät syöttiohjelmaa (goat file), johon tavoitetilassa haittaohjelma tartuttaa koodiansa. Tätä haitallista koodia verrataan olemassa olevaan torjuntaohjelman tietokantaan, josta pyritään löytämään vastaavuus jo olemassa olevaan haittaohjelmaan ja näin paljastamaan tunkeutuja. Suojatakseen koko käyttöjärjestelmän torjuntaohjelman pitää kyetä tarkastamaan myös muun muassa käynnistyslohko (MBR). [6, s.687–690; 50]

Haasteena sormenjälkiin perustuvassa havaitsemisessa on, että torjuntaohjelman tietokantaa pitää kyetä päivittämään riittävän usein, jotta torjuntaohjelma kykenee tunnistamaan uusia haittaohjelmia riittävän pienellä viiveellä. [6, s.687–690; 50]

Heuristinen analyysi käyttää hyväksi algoritmeja verratakseen tunnettujen haittaohjelmien tunnuspiirteitä löydettyihin mahdollisiin uusin uhkiin. Heuristisen analyysin avulla torjuntaohjelma kykenee havaitsemaan haittaohjelmia, joita ei ole vielä löydetty sekä havaitsemaan olemassa olevia haittaohjelmia, jotka ovat pyrkineet piiloutumaan tai muokkautumaan. [6, s.687–690; 50]

Torjuntaohjelmien toinen toimintatapa haittaohjelmia vastaan, on toimia taustalla analysoiden järjestelmäkutsuista haittaohjelmaan viittaavaa käytöstä (behavioral checking). Hälyttävää on, jos kohde pyrkii suorittamaan epäilyttävää sovellusta tai komentosarjaa (muun muassa poistamaan tai muokkaaman suuria määriä tiedostoja, ylikirjoittamaan käyttöjärjestelmän käynnistyssektoria, muuttamaan muiden ohjelmien sekä käyttöjärjestelmä -asetuksia), jotka viittaavat kohteen olevan haittaohjelma. [6, s.691; 50]

Torjuntaohjelmien kolmas toimintatapa haittaohjelmia vastaan, on tiedostojen eheystarkastukset (integrity checking). Tarkastettuaan kiintolevyn ensimmäistä kertaa ja todettuaan sen puhtaaksi, torjuntaohjelma lisää tarkistussumman ohjelmiin, jotta se voi myöhemmin tarkastaa, että

ne ovat säilyneet eheänä. Jos myöhemmässä tarkastuksessa tarkistussumma on muuttunut, herää torjuntaohjelmalla epäily haittaohjelmasta. [6, s.691]

Jatkotoimenpiteenä muissa analysoinneissa haittaohjelmahälytyksen tuottanut ohjelma voidaan siirtää tarkempaan dynaamiseen analysointiin, jossa ollaan ohjelmatiedoston sijaan kiinnostuneita siitä, mitä haittaohjelma tekee ajon aikana. Epäilty haittaohjelma käynnistetään hiekkalaatikossa (tietokone tai -verkko, nykyisin usein virtuaalinen), jossa sen käytöstä eli vaikutusta ympäristöön seurataan. Dynaamista analysointia käytetään, koska esimerkiksi kryptattujen, polymorfisten tai tahallisesti monimutkaistettujen (obfuscation) haittaohjelmien takaisinmallinnus on usein haastavaa. [19, s.21]

Muistissa pysyvällä haittaohjelmalla (memory-resident) on kyky säilyä tietokoneen muistissa ja näin ollen ajossa vielä suorittamisen jälkeen. Muistissa pysyvä takaovi saattaa odottaa käskyä, tiedostovirus (file infecting virus) odottaa isäntäohjelman suoritusta tai mato lähetellä sähköposteja. Jotkin muistissa pysyvät haittaohjelmat kykenevät tarkastelemaan järjestelmäkutsuja ja havaitsemaan torjuntaohjelman tarkastusyritykset, jolloin niiden toiminnan estäminen on haastavaa. [6, s.687–690; 51]

Muistissa pysyvät haittaohjelmat kyetään torjumaan nykyaikaisilla torjuntaohjelmilla, jotka kykenevät toimimaan ilman käyttöjärjestelmää, koska ne sisältävät tarvittavat virtuaaliajurit kiintolevyn tiedostojen lukemiseen. [6, s.690–691]

2.6.3 Käyttäjän tunnistaminen ja todentaminen

Palveluiden turvallinen käyttäminen ja turvallinen asiointi käyttöjärjestelmässä edellyttää kaikkien käyttäjien tunnistamista ja todentamista kirjautuessaan. Tunnistamisella pyritään varmentamaan, kuka käyttäjä on ja todentamisella todentamaan, kuka käyttäjä väittää olevansa. Jos käyttöjärjestelmä ei pysty tunnistamaan ja todentamaan käyttäjää, se ei pysty kontrolloimaan mihin tiedostoihin ja resursseihin heillä on pääsy. Joten käyttöjärjestelmät eivät anna tähän mahdollisuutta. Tavoitetilassa todentamatonta käyttäjää ei päästetä järjestelmään. [6, s.626; 52; 53, s. 67]

Todennustavat pohjautuvat johonkin, mitä käyttäjä tietää, jotain mitä hän on tai johonkin, mitä hänellä on. Nykyaikaiset käyttöjärjestelmät sisältävät monia erilaisia käyttäjän todennustapoja, joista käytetyimpiä ovat käyttäjätunnuksen ja salasanan yhdistelmä. Muita yleisesti käytössä olevia keinoja käyttäjän todentamiseen kirjautumisen yhteydessä ovat muun muassa toimikortti- ja biometrinen todennus. [6, s.626–637; 52; 53, s. 68–69]

2.6.4 Käyttöoikeuksien hallinta

Käyttöoikeuksien hallinnan tehtävänä on antaa käyttöjärjestelmän käyttäjille tai rooleille tietyn tasoisia oikeuksia palveluihin ja järjestelmäresursseihin. Yleistynyt tasojako käyttöjärjestelmissä toteutetaan roolipohjaisilla (RBAC) käyttäjä- ja ylläpitäjätasoilla, mutta laajempiakin roolijakoja on olemassa. Käynnistettävien prosessien oikeudet riippuvat käyttäjätasosta. [1, s.651–653; 6, s.605–608]

Käyttöoikeuksien hallinnan avulla käyttäjille kyetään antamaan käyttäjän tai roolin mukainen pääsy tietokantoihin sekä luku- tai kirjoitusoikeus sen tiedostoihin. Seuraavassa kaaviossa on esitelty malli tiedostojen käyttöoikeuksista. Tässä mallissa esimerkiksi käyttäjä 1 omistaa tiedoston 1. Hänellä on myös oikeus lukea ja kirjoittaa siihen. [1, s.672]

	Tiedosto 1	**Tiedosto 2**	**Tiedosto 3**	**Tiedosto 4**
Käyttäjä- tai Rooli A	Omistaa Lukea Kirjoittaa		Omistaa Lukea Kirjoittaa	
Käyttäjä- tai Rooli B	Lukea	Omistaa Lukea Kirjoittaa	Kirjoittaa	Lukea
Käyttäjä- tai Rooli C	Lukea Kirjoittaa	Lukea		Omistaa Lukea Kirjoittaa

Taulukko 1 Tiedostojen käyttöoikeudet [1, s.671, suomennos ylil. Jose Mäntylä]

2.6.5 Salaus, salasanat ja suolaus

Salaaminen on keskeinen tiedon suojaamisen keino, jossa matemaattisia menetelmiä kuten algoritmeja käyttäen alkuperäinen data salataan luottamukselliseen muotoon, jotta luvaton käyttäjä ei kykene lukemaan sitä. Käyttöjärjestelmä käyttää salausta esimerkiksi tallennettujen salasanojen, tiedostojen ja levyosioiden sekä tiedonsiirron salaamiseksi. [6, s.619–620; 54, s.25; 55, s.58; 56]

Salausmenetelmät jaetaan symmetrisiin ja epäsymmetrisiin menetelmiin sen mukaan, käytetäänkö salaamisessa yhtä salaista avainta vai yksityisen ja julkisen avaimen sisältävää avainparia. [6, s.622; 55, s.16–17]

Symmetrisessä järjestelmässä lähettäjällä ja vastaanottajalla on yhteinen avain, jonka avulla he voivat sekä salata että purkaa viestejä. Yleinen käytössä oleva symmetrinen salausalgoritmi on AES. [6, s.622; 55, s.16–17]

Epäsymmetrisessä menetelmässä salaaminen ja purkaminen toteutetaan eri avaimilla. Toinen avaimista pidetään julkisena ja toinen yksityisenä. Salaaminen tapahtuu esimerkiksi vastaanottajan julkisen avaimen avulla, jonka lähettäjä saa tietoonsa ennen salausta vastaanottajalta tai muodostettuna identiteettiin liittyvistä tiedoista (Identity Based Encryption). Vastaanottaja avaa salauksen omalla yksityisellä avaimellaan. Yleinen käytössä oleva epäsymmetrinen salausalgoritmi on esimerkiksi RSA. [6, s.622; 55, s.16–17]

Salausavaimet pystytään purkamaan riittävällä ajalla ja laskentateholla, mutta hyvin toteutetussa salauksessa perinteisten supertietokoneiden laskentateholla puhutaan silti jopa miljoonista vuosista. Esimerkiksi AES-256 salauksessa kyberhyökkääjän täytyy testata 2^256 eri vaihtoehtoa, joten maailman tehokkaimmalla supertietokoneella (IBM Summit 148,6 petafloppia/s) menisi noin (2^256) / (148,6 x 2^50 (liukuluku) x 365 x 24 x 60 x 60) = 2,18 x 10^52 vuotta avata salaus. [54, s.25; 55, s.58]

Tulevaisuuden salausta purkavilla kvanttikoneilla nykyiset salausavaimet, erityisesti epäsymmetriset, kyetään todennäköisesti purkamaan huomattavasti nopeammin kuin nykyisillä traditionaalisilla tietokoneilla. Tämän takia maailmalla onkin kehitteillä monia kvanttiresistenttejä algoritmeja. [54, s.25; 55, s.58]

Käyttöjärjestelmä tarvitsee salausta esimerkiksi tallennettujen salasanojen salaamiseksi. Niitä ei saa löytyä järjestelmästä selväkielisinä, vaan ne tallennetaan tiivisteiden ja suolauksen avulla. Näillä menetelmillä suojattujen salasanojen väärinkäyttöriski pienenee tietomurtojen yhteydessä. [54, s.51; 18, s.687–688]

Tiivistefunktiot ovat tietojenkäsittelyssä algoritmeja, jolla voidaan laskea datasta kiinteän mittainen tiiviste esimerkiksi salasanoihin tai ohjelmien allekirjoitusvarmenteihin. Salasanojen salaamisen tarkoitettuja tiivistefunktiota ovat muun muassa PBKDF2, bcrypt, scrypt, SHA-2 ja SHA-3. Ohjelmien varmentamisessa laskettavista tiivisteistä puhutaan usein tarkastussummina. Varmentamiseen käytettäviä tiivistefunktioita ovat muun muassa SHA-256 sekä CRC-32. [57; 58]

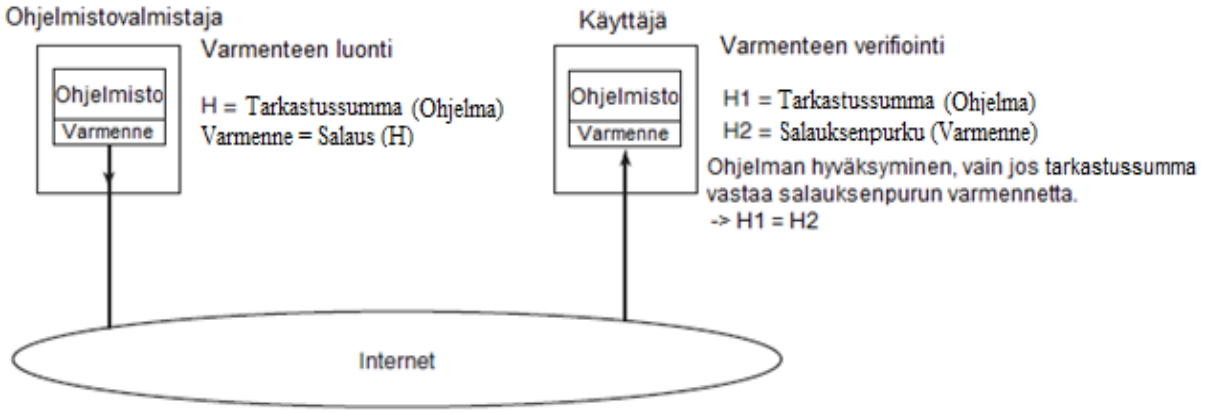
Syöte	Tiiviste
Käyttöjärjestelmä	30ac023607c5fce8f9388e942e0aa24dbaa8bd3bd0980911c452cbbaf1816f0f
Käyttöjärjestelmät	7cddf73f598171a5623389c9ceaf15791187e0834f5ec6dec10d16acca77531d
Käyttöjärjestelmän	3cbc845be7d2dd89cf5e8f49a8768bbd289194aa7ddd2482a777935eca6c6d1a

Taulukko 2 SHA-256 tiivistefunktiolla lasketut esimerkkitiivisteet [59]

Merkittävä lisä salasanatiivisteiden turvallisuuden lisäämiseksi on niin sanotun suolan (Salt) käyttö. Suolauksessa käytetään yleensä julkista satunnaisesti muodostettua, ainutkertaista sekä käyttäjäkohtaista merkkijonoa, jota käytetään algoritmin parametrina salasanan lisäksi. Näin ollen suolan muuttaminen muuttaa myös salasanatiivistettä. Suolaus vaikeuttaa salasanojen selvittämistä, sillä vaikka salasanaa vastaava tiiviste olisi tiedossa, täytyisi myös tietää millä löydetty salasanatiiviste on suolattu. Menetelmää käytetään, koska yleisille sanoille on avoimia tiivistetietokantoja, kuten esimerkiksi crackstation.net, jossa on miljoonien käytettyjen salasanojen tiivistetietokanta. [18, s.687–688; 52, s.51; 57]

Tietoturvan kannalta on olennaisen tärkeää, että käyttöjärjestelmien asennuspaketit ohjelmineen ja ajureineen sekä päivitykset voidaan todeta alkuperäisiksi. Eheyden varmentamiseen käytetään tiivistefunktiolla laskettavia tarkastussummia (tiiviste) ja epäsymmetrisiin julkisiin avaimiin perustuvia allekirjoitusvarmenteita. [6, s.694; 18, s.687–688]

Allekirjoittaakseen ohjelman ohjelmiston valmistaja laskee tarkastussumman (tiiviste), jonka jälkeen valmistaja allekirjoittaa (salaa) tarkastussumman omalla yksityisellä avaimellaan. Asennusvaiheessa tarkastussumma lasketaan ja allekirjoitusvarmenne avataan valmistajan julkisella avaimella, jonka jälkeen valmistajan julkista avainta verrataan laskettuun tarkastussummaan. Jos ne täsmäävät toisiinsa, ohjelma hyväksytään alkuperäisenä. [6, s.694]



Kuva 6 Allekirjoitusvarmenteet [6, s.694, suomennos ylil. Jose Mäntylä]

2.6.6 Käynnistyksen suojaaminen

Tietokoneen emolevy sisältää laiteohjelmistojen (firmware) sekä käyttöjärjestelmän keskusmuistiin lataavan ja käynnistävän BIOS:n (Basic Input-Output System) tai uudemman UEFI:n (Unified Extensible Firmware Interface). Niiden tehtävänä on käynnistää käyttöjärjestelmä ja antaa mahdollisuus määrittää laitteiston perusasetukset. [60]

UEFI kehitettiin korvaamaan 16-bittisille järjestelmille alunperin suunniteltu BIOS-ohjelmisto. Tavoitteena oli mahdollistaa 64-bittisten sovellusten ajo käynnistysohjelmistoissa ja nopeuttaa käynnistystä. UEFI:n yhteydessä voidaan myös käyttää Secure Boot -protokollaa, jolloin varmentamattomista lähteistä tulevien ohjelmien suorittaminen vaikeutuu. [26, s.4; 60]

Secure boot -protokollan tavoitteena on estää käynnistystiedostojen muokkaaminen ja sitä kautta BIOS / UEFI:n haavoittuvuuksia hyväksikäyttävien haittaohjelmien ja murtautumistyökalujen pääseminen käsiksi käyttöjärjestelmän käynnistystiedostoihin. Secure Boot:n ollessa päällä sallitaan ainoastaan varmennetut ja laitteisto-ohjelmistojen kanssa yhteensopivien käyttöjärjestelmien käynnistystiedostojen ajaminen tietokoneelle. [26, s.4; 60]

TPM -turvapiiri (Trusted Platform Module) on tietokoneen emolevyyn integroitu piiri, jossa säilytetään työasemalevyjen avaamiseen tarvittavat salausavaimet, ja jotka se vapauttaa vain, jos työaseman tiiviste (hash) vastaa tietokantaan tallennettuja tarkisteita. Esimerkiksi alaluvussa 3.1.4 Informaation salaus käsiteltävä Bitlocker käyttää TPM-piiriä lukitsemalla salausavaimet sen sisään. TPM -piiri on vaatimuksena Measured Boot -protokollan käytölle. [61]

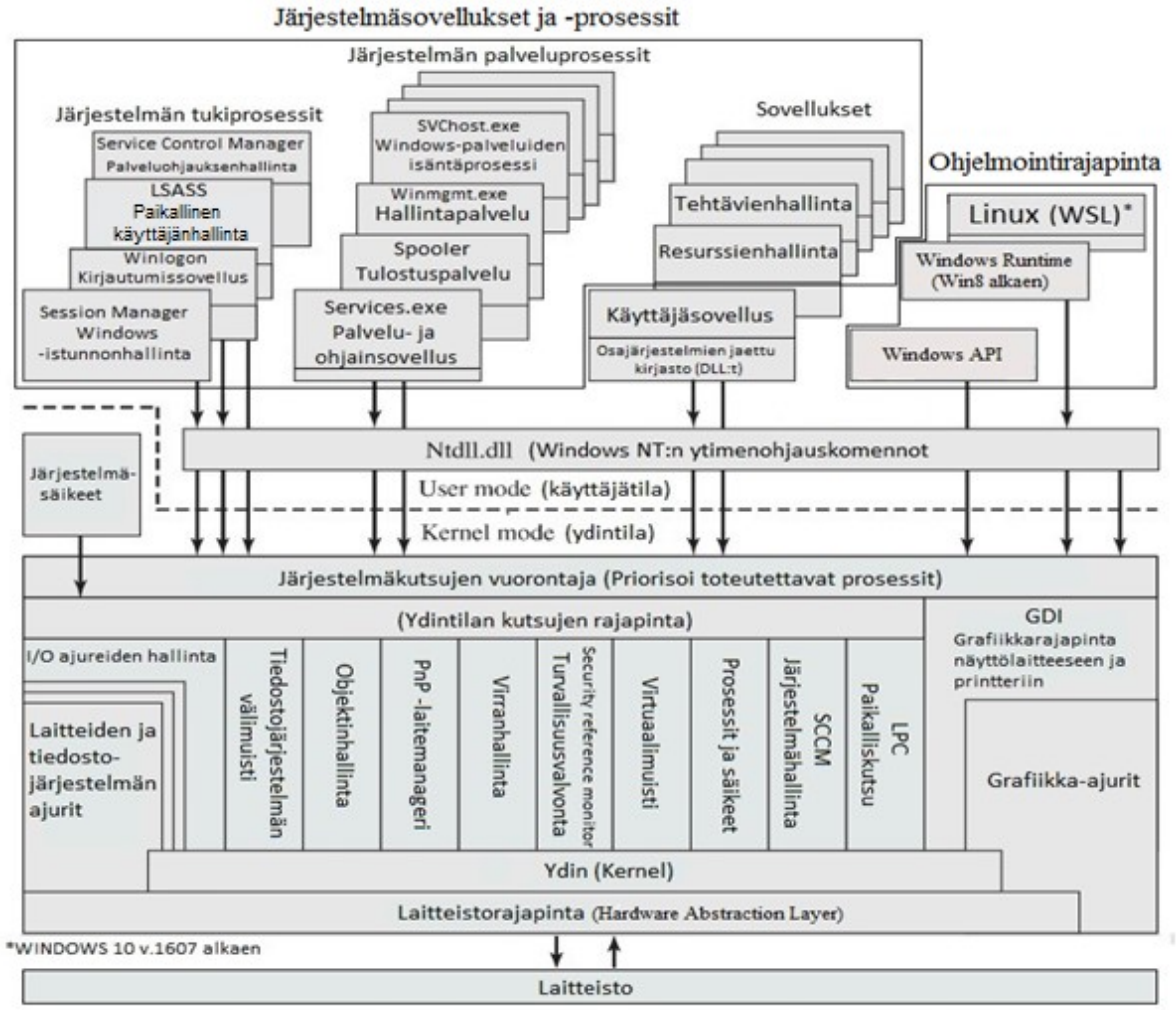
2.6.7 Ytimen rakenne

Käyttöjärjestelmän ydin toimii käyttöjärjestelmän peruskivenä käsitellen prosesseja ja keskeytyksiä omassa suojatussa muistiavaruudessaan (kernel space). Ydin tarjoaa toimintonsa palveluina (services), jotka koostuvat erillisistä komponenteista (components). Komponentit kykenevät kommunikoimaan keskenään rajapintojen (interfaces) avulla. Sovellusohjelmat käyttävät järjestelmäkutsuja kommunikoidakseen ytimen kanssa. [2; 3, s.11–12; 62]

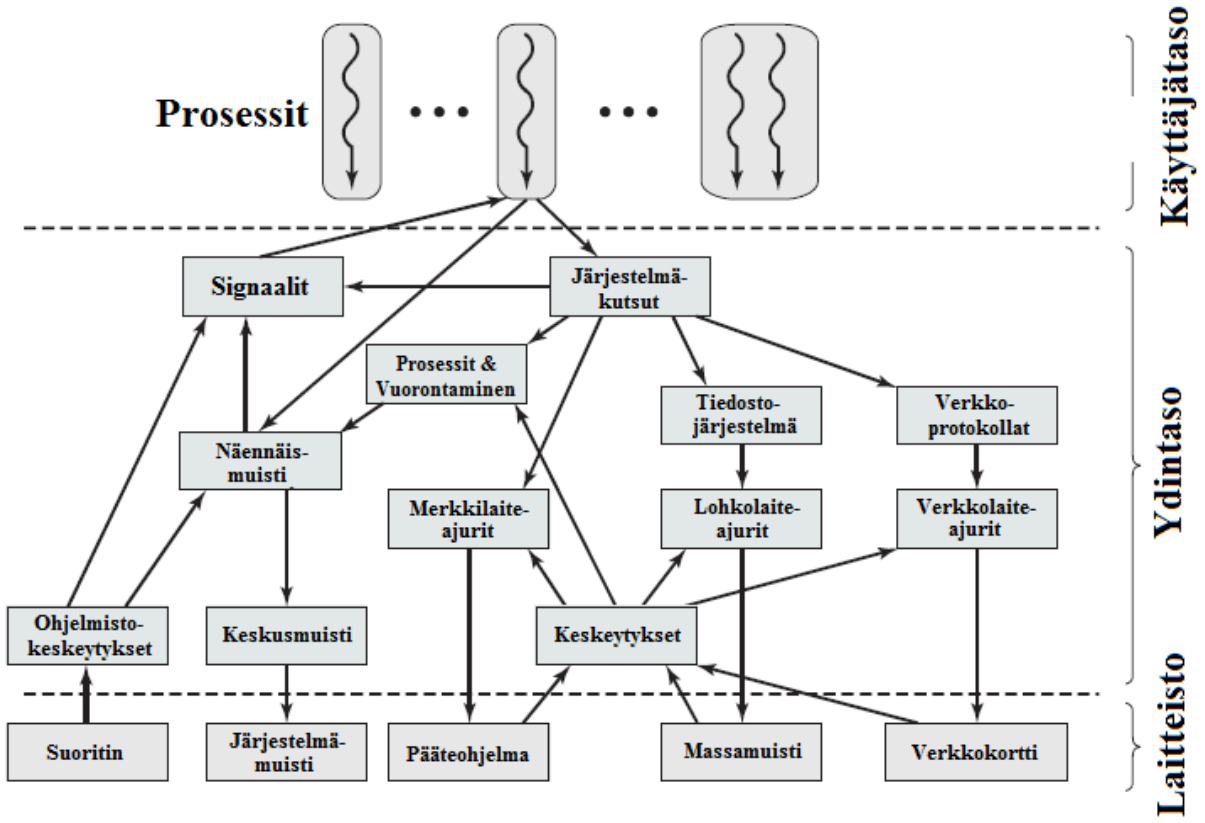
Käyttöjärjestelmän ytimen tuottamia tärkeitä palveluita järjestelmän toimintakyvylle ovat vuoronnus, tehtävänhallinta, laitteistonhallinta, tiedostojärjestelmä, muistinhallinta, verkonhallinta, käyttöoikeuksien hallinta, graafinen- ja/tai komentorivikäyttöliittymä. [2; 3, s.11–12; 62]

Käyttöjärjestelmän ytimet voidaan jaotella monoliittisiin ytimiin, joissa kaikki toiminnot ovat samassa muistiavaruudessa (esimerkkinä Debian) sekä mikroytimellisiin (esimerkkinä Google Fuchsia), joissa tietoturvallisemmin käyttöjärjestelmämoduulit on eristetty ytimestä ja toisistaan jakamalla ne eri muistiavaruuteen nopeuden kustannuksella. Windows -käyttöjärjestelmien ydin voidaan katsoa kuuluvan näiden mono- ja mikroytimillisten tyyppien hybrideiksi. [1, s.92–94; 6, s.62–72]

Kuvassa 6 on kuvattu Windows 10 -käyttöjärjestelmän ytimen arkkitehtuuri ja keskeiset prosessit. Windows (NT) ytimen tehtävänä on toteuttaa kaikki itsenäisen toiminnan kannalta oleelliset tehtävät. [1, s.102; 6, s.864–871]



Kuva 7 Windows 10 -käyttöjärjestelmän ytimen arkkitehtuuri ja keskeiset prosessit [1, s.102, suomennos ylil. Jose Mäntylä]



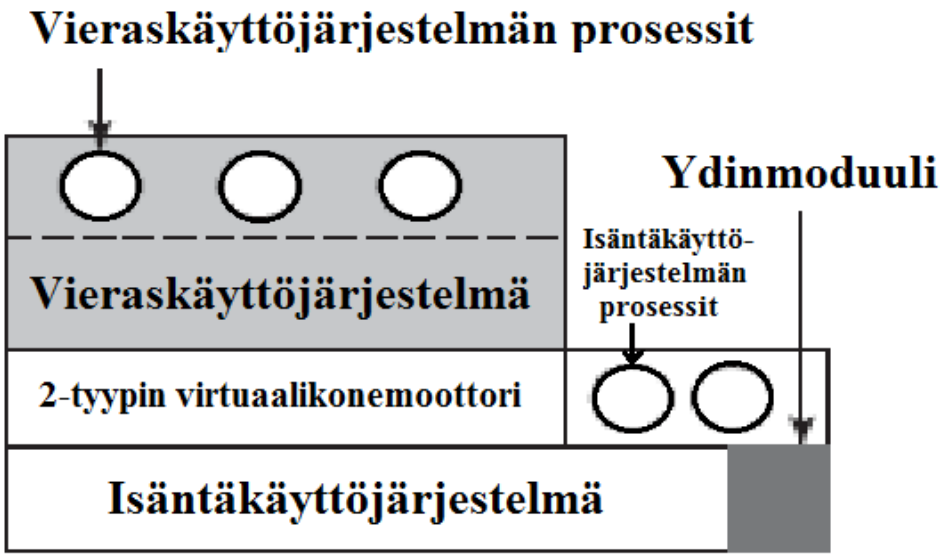
Kuva 8 Debianin Linux -ytimen arkkitehtuuri [1, s.116, suomennos ylil. Jose Mäntylä]

2.6.8 Virtualisointi

Virtualisoinnilla järjestelmään voidaan luoda monta virtuaalikonetta, jotka kaikki käyttävät järjestelmän laitteistoa ja joilla voidaan ajaa jopa montaa eri käyttöjärjestelmää samanaikaisesti. Virtualisoinnilla mahdollistetaan yhteensopivuustilojen käyttö, jotta esimerkiksi vanhojen Windows-versioiden ohjelmia kyetään ajamaan Windows 10 -käyttöjärjestelmässä. [6, s.472–473]

Moderneissa käyttöjärjestelmissä käytetään 2-tyypin virtuaalikonemoottoreita (hypervisor), jotka käyttävät isäntäkoneen käyttöjärjestelmää ja sen tiedostojärjestelmää luodakseen prosesseja, tallentaakseen tiedostoja ja niin edelleen. Erona 1- ja 2-tyypin virtuaalikonemoottoreilla on, että 1-tyypin ”natiivit” virtuaalikonemoottorit käyttävät suoraan laitteistoa, eivätkä ne tarvitse käyttöjärjestelmää laitteiston ja niiden välillä. [6, s.71–72 & 472–473]

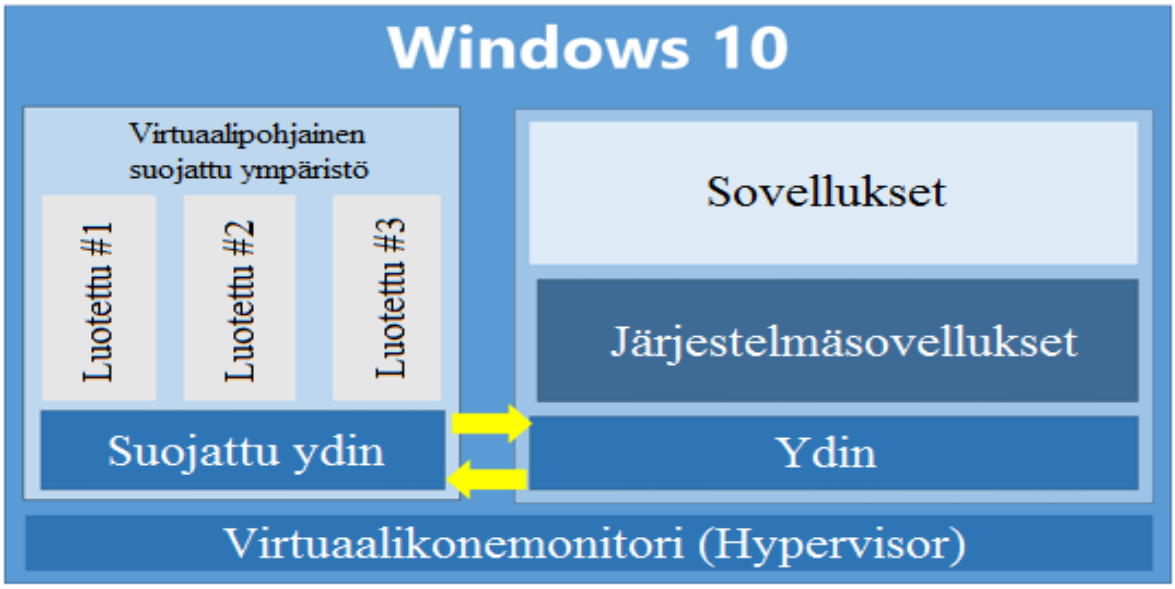
2-tyypin virtuaalikonemoottoreilla mahdollistetaan ympäristöjen eristäminen toisistaan ja yhden virtuaalikoneen kaatuessa ei kaadeta muita virtuaalikoneita. Kuvassa 9 on esitetty 2-tyypin virtuaalikonemoottori [6, s.71–72 & 472–473]



Kuva 9 2-tyypin virtuaalikonemoottori [6, s.71, suomennos ylil. Jose Mäntylä]

Virtualisointia käytetään hyväksi muun muassa muistin suojaamisessa. Prosessit eivät saa kirjoittaa toisten virtuaalikoneiden (esimerkiksi vieraskäyttöjärjestelmä) tai isäntäkäyttöjärjestelmän muistialueille. Joten käyttöjärjestelmissä käytetään yleisesti virtuaalimuistia, jossa prosessi näkee virtuaalisen osoiteavaruuden fyysisen muistin sijasta ja käyttöjärjestelmä huolehtii virtuaalisosoitteen muuttamisesta fyysiseksi osoitteeksi. Käytetyn muistialueen uudelleenallokointi ei myöskään saa paljastaa vanhaa dataa (object reuse protection). [63, s.19]

Kuvassa 10 on Windows 10 -käyttöjärjestelmän virtualisointi esimerkkinä käyttöjärjestelmien toteuttamasta virtualisoinnista. Virtuaalipohjainen suojattu ympäristö on jaettu eri muistiavaruuteen järjestelmän muista osista.



Kuva 10 Windows 10-käyttöjärjestelmän virtualisointi [64, suomennos ylil. Jose Mäntylä]

Turvallisuuden kannalta virtuaalikoneella pitää olla käytössään täysin virtualisoidut resurssit. Kaikki siirräntäpyynnöt (I/O instructions) pitää ohjata ohjelmointikielen tulkin (kuten Bochs) kautta ja tehdä täysin, mitä siirräntäpyynnössä vaaditaan. [6, s.475]

Virtuaalikoneiden avulla voidaan lisätä järjestelmän tietoturvaa esimerkiksi [6, s.475; 65, s.28–31]:

- tutkimalla haittaohjelmia virtuaalisessa ”hiekkalaatikossa” (ks. 2.4.3 Haittaohjelmien torjunta) ja kehittämällä parempia menetelmiä niiden torjumiseksi
- siirtämällä sovellukset omiin virtuaalikoneisiinsa, jolloin niihin voidaan tehdä käyttötarkoituksen perusteella tiukemmat tietoturvasäännöt
- palauttamalla virtuaalikoneet varmuuskopioista tietomurron jälkeen murtoa edeltävään tilaan ja tehdä virtuaalikoneeseen tarvittavat muutokset, joilla estetään hyökkäyksen toistaminen
- analysoimalla ohjelmien ajon aikaista käyttäytymistä suorittamalla se turvallisesti virtuaalikoneessa, joka tallettaa ohjelman toiminnan lokiin. Loki analysoidaan, jolloin voidaan havaita ohjelmointivirheet.

Virtualisointi usein parantaa, mutta joskus myös heikentää tietoturvaa. Virtuaaliympäristöt ovat kaikkien muiden ympäristöjen tavoin haavoittuvia ja niiden ominaisuuksia voidaan käyttää väärin. [65, s.28–31]

Mahdollisia virtualisoinnin aiheuttamia riskejä tietoturvalle ovat esimerkiksi [65, s.28–31]:

- leikepöydän jakaminen isäntäkoneen ja virtuaalikoneiden kesken, mitä mahdollinen haittaohjelma voi käyttää väärin
- virtuaalikoneen valvonta -ohjelmista voi löytyä suunnittelu- tai ohjelmointivirheitä. Virtuaalikoneessa ajettava haittaohjelma voi virhettä hyväksikäyttäen mahdollisesti kaapata isäntäkoneen hallintaansa
- kopioiduista virtuaalikoneista voidaan saada haltuun kertakäyttöisiksi tarkoitettuja salausavaimia
- jos isäntäkoneelle tehdään varmuuskopio virtuaalikoneesta, siitä voidaan saada selville salasanoja ja salausavaimia

2.7. Käyttöjärjestelmän toimet tietoturvan ylläpitoon

Käyttöjärjestelmä- ja ohjelmapäivitysten jälkeen verkkorikolliset ja muut yhteisöt alkavat etsiä uudesta ohjelmakoodista hyväksikäyttömenetelmää. Tavoitetilassa hyväksikäyttömenetelmän löytää yhteisö, jolla ei ole halua käyttää menetelmää kyberrikollisuuden tai -sodankäynnin menetelmänä, vaan hyväksikäyttömenetelmä välitetään alkuperäisen ohjelmiston tekijöille, jotka paikkaavat tarvittaessa haavoittuvuuden päivityksellä. [6, s.974; 66]

Oleellinen osa käyttöjärjestelmän tietoturvallisuutta on haavoittuvuudet korjaavien päivitysten asentaminen järjestelmään manuaalisesti tai automaattisesti. Jos käyttöjärjestelmää ja siihen asennettuja ohjelmistoja ei päivitetä, suurenee ajan myötä todennäköisyys haavoittuvuuden löytämiselle ja sen johdosta hyökkäyspinta-ala kasvaa. Oleellinen osa päivitysketjua on niiden jalkauttaminen käyttäjille asti. [6, s.974; 66; 67, s.2]

Ironista haavoittuvuuksien julkaisussa on, että usein verkkohyökkääjät käyttävät hyväksi jo julkaistuja haavoittuvuuksia takaisinmallintamalla (reverse engineering) niitä välineenä kehittää hyökkäysmenetelmä. [6, s.974]

2.8. Testaus ja auditointi

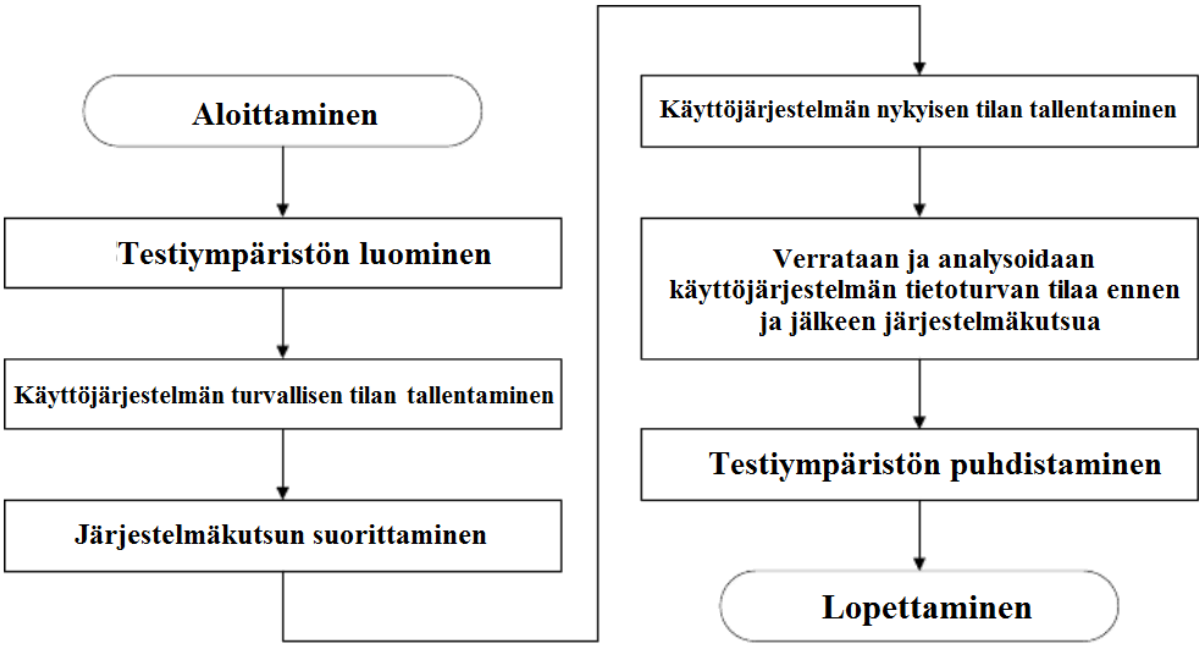
On yleisesti hyväksyttyä, että käyttöjärjestelmät ja niiden tietoturvallisuus on merkittävin tekijä koko järjestelmän tietoturvallisuudelle. Tämän takia uhkamallinnus, tietoturvatestaaminen ja auditointi on hyvin tärkeää käyttöjärjestelmän valinnassa. Auditointiin on olemassa valmiita sertifikaatteja: Näistä tunnetuimpia ovat Common Criteria (CC) ja FIPS 140-2. [68; 69, s.88; 70]

Common Criteria on vuonna 2005 julkaistu kansainvälinen tietoturvastandardi, jota käytetään sertifiointijärjestelmänä muun muassa käyttöjärjestelmien kyberturvallisuuden auditointiin. Common Criteria -sertifikaatti toimii vakuutena sille, että käyttöjärjestelmän tietoturvallisuus on toteutettu ja arvioitu perusteellisilla sekä standardoiduilla menetelmillä. Windows 10-käyttöjärjestelmällä ja esimerkiksi Debian-pohjaisella Ubuntu-jakelulla on CC-sertifikaatti, mutta Debian -käyttöjärjestelmillä ei. Debian-projekti on perustellut sertifioimattomuutta sillä, että ne maksavat liikaa. [69, s.88; 70]

FIPS 140-2 on Yhdysvaltojen hallinnon tietoturvallisuusstandardi, ohjelmistojen ja laitteistojen kryptograafisille moduuleille, jonka myöntää Yhdysvaltojen Standardi- ja teknologia-instituutti (NIST). Standardilla evaluoidaan 11 eri salauksen suunnittelun ja toteutuksen aluetta, joita arvioidaan akkreditoiduissa laboratorioissa. Jokaisen alueen arviointi jakautuu neljään tasoon tietoturvattomammasta tietoturvallisimpaan. FIPS 140-2 standarditasoa arvioimalla voidaan vertailla käyttöjärjestelmissä käytettävien salauksien tietoturvallisuutta. [71, s.18–19]

Windows 1903 ei ole vuoden 2019 loppuun mennessä saanut FIPS 140-2 sertifikaattia, mutta useampi sen käyttämistä salausmenetelmistä ja aiemmista pääversioista on. Windows 10 -käyttöjärjestelmä voidaan konfiguroida erilliseen FIPS 140-2 -hyväksynnän saaneeseen moodiin (FIPS-mode) eli tehdasasetuksilla se ei täytä sertifikaatin kriteereitä. Debian 10 -käyttöjärjestelmällä ei ole myöskään FIPS 140-2 sertifikaattia [71, s.18–19; 72]

Sertifikaatit eivät yksin takaa käyttöjärjestelmien tietoturvallisuutta. Niiden puuttuminen aiheuttaa vielä suuremman tarpeen testaamiselle, jotta voidaan todeta käyttöjärjestelmän soveltuvuus tietoturvakovennuttuihin järjestelmiin. Laajojen ohjelmistojen kuten käyttöjärjestelmien tietoturvatestaamisessa hyödynnetään automaatiota, koska manuaaliset tarkastukset vaativat liikaa resursseja. Ne tarkastavat muun muassa käyttöjärjestelmän konfiguraatiota, dokumentaation johdonmukaisuutta, tunkeutumisen havaitsemista, järjestelmäkutsujen ja komentojen suoritusta, tietoturvakäytänteitä ja -tavoitteita sekä -auditointitapoja. [68, s.116–123]



Kuva 11 Järjestelmäkutsujen testaaminen [68, s.119]

3. WINDOWS 10- JA DEBIAN 10 -KÄYTTÖJÄRJESTELMIEN TIETOTURVAOMINAISUUDET

Tietoturvaominaisuuksien tarkoitus on suojata käyttöjärjestelmää väärinkäytöiltä hyökkääjälle edullisella tavalla. Käyttöjärjestelmä ei kykene toimimaan tietoturvallisesti ilman kykyä todentaa käyttäjiä ja prosesseja. Tämän johdosta se ei kykenisi kontrolloimaan niiden pääsyä tietokantoihin, tai tekemään lokia niiden toiminnasta. Muilla tietoturvaominaisuuksilla, kuten palomuureilla, virustorjunnalla ja salauksella, pyritään käyttöjärjestelmissä ratkaisemaan tämä ongelma. [1, s.657–695; 18, s. 684–704; 73, s. 657–701]

3.1. Windows 10 -käyttöjärjestelmän tietoturvaominaisuudet

Windows 10 -käyttöjärjestelmä on Microsoft Windows NT -perheeseen kuuluva vuonna 2015 julkaistu Windows 8.1 -käyttöjärjestelmän seuraaja. Windows 10 -käyttöjärjestelmä on moderni käyttöjärjestelmä, joka monien muiden käyttöjärjestelmien tavoin soveltuu monille erityyppisille päätelaitteille. Sovellusrajapintana näissä kaikissa toimii Universaali Windows-ympäristö (UWP), jolloin sovellusten ydinrakenne on sama laitetyypistä riippumatta. Edelleen on kuitenkin mahdollista käyttää vanhempaa Win32 -rajapintaa. [1, s.101; 2; 73; 74]

Microsoft on siirtymässä Windows 10 alkaen niin sanottuun *Rolling release* -päivitysmalliin, jossa uusien Windows-käyttöjärjestelmien (7, 8, 8.1, 10) sijasta Microsoft päivittää Windows 10 -käyttöjärjestelmän käyttäjäversioita jatkuvasti uusilla pää- ja alaversioilla. Uusia yrityskäyttöön tarkoitettuja Enterprise-versioita sekä palvelinkäyttöön tuotettavia Windows Server-käyttöjärjestelmiä tuotetaan edelleen. Osa tässä luvussa esiteltävistä ominaisuuksista on käytössä vain Windows 10 Enterprise 5 (E5) -käyttöjärjestelmäversioissa. [1, s.101; 2; 73; 74]

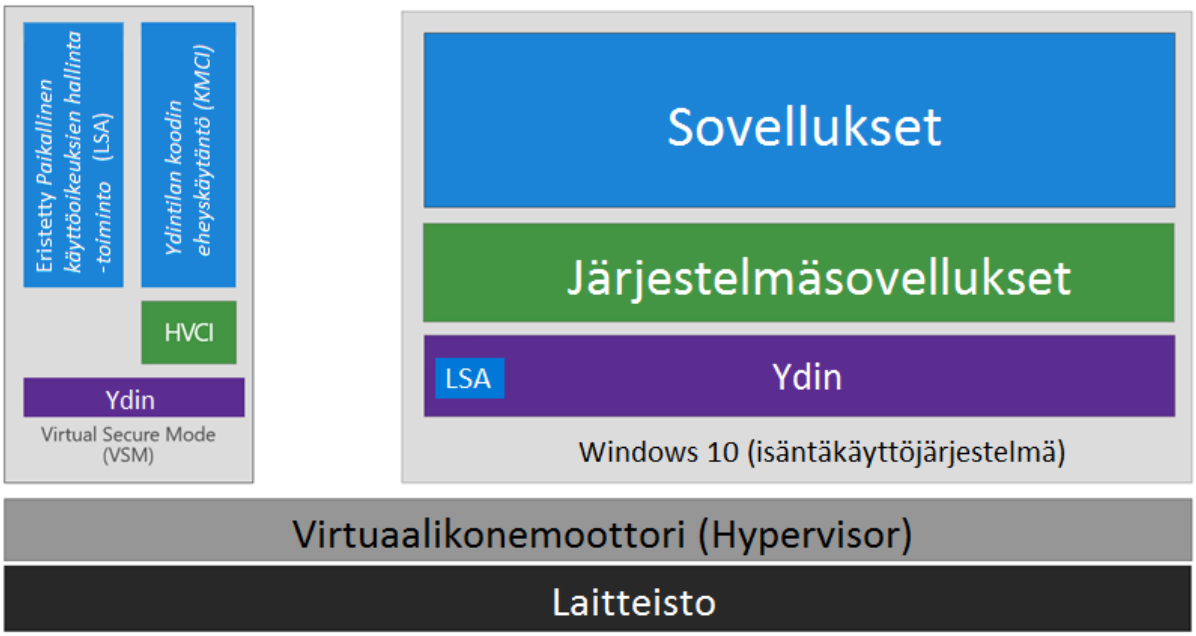
3.1.1 Hyökkäyspinta-alan pienentäminen

Windows 10 -käyttöjärjestelmän tärkeimpiä keinoja pienentää hyökkäyspinta-alaa on virtualisointi ja siihen pohjautuvat suojaukset, laitevartija sekä prosessin suojaus. [22, 26, 67, 75, 76, 77, 78, 79]

Windows 10 -käyttöjärjestelmä käyttää laitesuojaukseen virtuaalipohjaista muistinsuojausta. Virtuaalipohjainen suojaus (virtualization-based security, VBS) luo eristetyn muistialueen muusta käyttöjärjestelmän muistiavaruudesta (ks. kuva 9, s. 30). Tavoitteena ominaisuudella on, että suojatun alueen sovelluksia ei kyetä muokkaamaan muistialueen ulkopuolelta, mikä luo

järjestelmä- ja tietoturvasovelluksille suojauksen monilta haavoittuvuuksilta. Suojatussa virtuaalipohjaisessa ympäristössä toimii oma suojattu ydin (secure kernel), jolla se suorittaa suojattuja sovelluksia. [22, s. 24; 75]

Hypervisor Enforced Code Integrity (HVCI) on VBS:ään liittyvä muistin eheyskäytäntö, jonka tavoitteena on estää haitallisen tai varmentamattoman koodin injektio ja suorittaminen ytimessä. Toiminto käyttää hyväksi laitteistovirtualisointia (hardware virtualization) ja virtuaalikonemonitoria (hypervisor), jotta tavoitetilassa kohdeympäristö olisi suojattu haittaohjelmien hyökkäyksiltä (Virtual Secure Mode, VSM). Suojatussa virtuaalisoidussa ympäristössä toimii myös ydintilan koodin eheyskäytännöstä huolehtiva (KMCI) ja *Eristetty paikallinen käyttöoikeuksien hallinta* -toiminto (LSA, ks. 3.1.3 Identiteetin suojaus). [76; 78, s. 17; 79]



Kuva 12 Virtuaalipohjainen suojaus (VBS) ja HVCI [79, suomennos ylil. Jose Mäntylä]

Laitevartija (Device Guard) on yhdistelmä monia laitteisto- ja ohjelmistotoimintoja, joilla pyritään kontrolloimaan, mitä ohjelmia Windows 10 -käyttöjärjestelmässä voi suorittaa. Laitevartijan tavoitteena on erityisesti estää kohdistetut haittaohjelmahyökkäykset (APT-hyökkäys), joita käsiteltiin alaluvussa 2.5.2 Haittaohjelmat. [22, s.21–23]

Laitevartija käyttää koodin eheyskäytäntöjä (Code integrity policy) rajoittaakseen, mitä sovelluksia voidaan ajaa Windows -käyttöjärjestelmän käyttäjä- ja ydintasoilla. Se myöskin käyttää hyväksi virtualisointia suojatakseen itsensä järjestelmävalvojan oikeudet omaavan hyökkääjän toteuttamalta poiskytkemiseltä. Näitä toimintoja voidaan hallita organisaation omilla ryhmäkäytännöillä. [67, 77]

Windows 10-käyttöjärjestelmän kaikilla järjestelmänvalvojaoikeuksien omaavilla prosesseilla on oikeudet virheenjäljitykseen (debug privilege). Nämä mahdollistavat niille oikeuden pyytää oikeuden kaikkiin prosesseihin järjestelmässä, ja lukea niiden muistin tietoa. Ohjelmat voidaan suojata näiltä pyynnöiltä käyttämällä niihin prosessinsuojausta (protected processes) [26, s.19]

Prosessinsuojauksesta Windows-käyttöjärjestelmissä vastaa PPL-toiminto (Protected Processes Light). PPL-toiminto antaa ohjelmille eritasoisia suojauksia digitaalisilla allekirjoituksilla. Erityisesti koodin allekirjoitusvarmenne sertifikaatti (code-signing certificate) määrittää, millä suojaustasolla ohjelma voidaan suorittaa. Suojaustaso määrittää, miten ohjelman oikeuksia rajoitetaan ja mitä sovelluslaajennuksia (DLL) se voi ladata. [26, s.19]

3.1.2 Muistin suojaus

Windows 10 -käyttöjärjestelmä suojaa muistin eheyttä hallintavirran suojauksella, Data-alueen suorituksen esto -toiminnolla, osoiteavaruuden satunnaistamistekniikalla sekä Strukturoidun poikkeuskäsittelyn ylikirjoitussuojalla. [22, 26, 80, 81]

Hallintavirran suojaus (Control Flow Guard eli CFG) on Windows 10 hyökkäysten estoon tarkoitettu suojaus, jonka tavoitteena on lieventää, tai jopa estää järjestelmän muistiin kohdistuvia hyökkäyksiä. Alun perin CFG-suojaus kehitettiin Use After Free -hyökkäyksien (UAF) torjumiseksi. Tämänkaltaiset hyökkäykset käyttävät hyväksi järjestelmien tapoja käsitellä muistia, jossa verkkohyökkääjillä on mahdollisuus suorittaa komentoja sisäänkirjautuneen käyttäjän (pahimmillaan järjestelmänvalvojan) oikeuksilla. CFG:tä kehitettiin ajan myötä torjumaan myös puskurin ylivuotovirheitä. [22, s.20; 26, s.33]

CFG tarkastelee ja rajoittaa ohjelmien antamia käskyjä kahdessa eri vaiheessa: analysointi- ja toimeenpanovaiheessa. Analysointivaiheessa ohjelman käskyt analysoidaan ennakkoon, jotta hallintavirrasta saadaan muodostettua kuva. Ohjelman, kuten viruksen yrittäessä puskurin ylivuotohyökkäystä, toiminta estetään toimeenpanovaiheessa. [22, s.20; 26, s.33]

Data-alueen suorituksen esto -toiminnon (Data Execution Prevention, DEP) avulla voi Windows 10 -käyttöjärjestelmä merkitä tietyt sivut (pages) muistista ei-suoritettaviksi. Kun ohjelmakäskyjä ei voida suorittaa tämän kaltaisesta muistiosoitteesta, puskurin ylivuotohyökkäyksien tekemisestä tulee vaikeampaa. Ohjelmiston yrittäessä suorittaa koodia suojatulta alueelta se aiheuttaa virheen. Kutsuttava prosessi lopetetaan, mikäli tätä virheen aiheuttamaa poikkeamaa ei käsitellä asianmukaisesti. [80, s.23]

Osoiteavaruuksien satunnaistamistekniikan (Address Space Layout Randomization, ASLR) avulla Windows 10 -käyttöjärjestelmän komponenttien osoiteavaruudesta satunnaistetaan kriittiset osoittimet (pointer), kuten esimerkiksi perusosoittimet (base pointer), pino- ja kekomuistiosoitteet sekä funktio-osoittimet. Tällä estetään tehokkaasti puskurin ylivuotohyökkäykset suoraan käyttöjärjestelmän käyttämiin muistiosoitteisiin. Satunnaisuudesta johtuen esimerkiksi haittaohjelmien on vaikeampi löytää hyökkäyksen tavoitteena oleva kohta muistista. [80, s.23]

Strukturoidun poikkeuskäsittelyn ylikirjoitussuoja (Structured Exception Handler Overwrite Protection, SEHOP) on toiminto, joka estää hyökkääjiä muokkaamasta sovelluksien poikkeuskäsittelytietueita. Se on yleinen menetelmä puskurin ylivuotohyökkäyksien toteuttamisessa. [81]

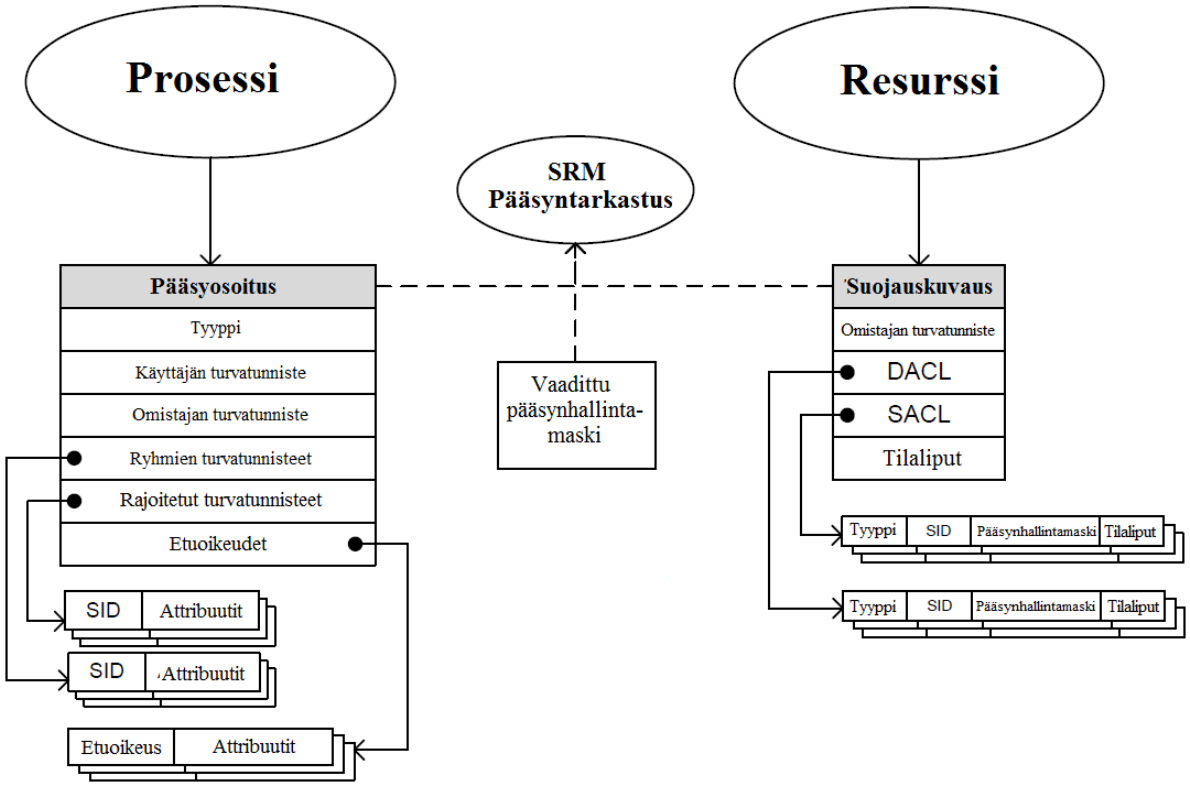
3.1.3 Uhkan vastustaminen

Windows 10 -käyttöjärjestelmässä on monien aiempien Windows-versioiden tapaan palomuuri. Windows Defender -palomuurilla pyritään estämään liikennettä, joka ei ole sallittua, käyttäen pääsynhallintalistan perusteella suodattavaa tilatonta pakettisuodatusta (Stateful Firewall). Säännöillä voidaan rajoittaa porttien ja ohjelmien käyttöä eri verkkoympäristöissä. Windows 10 -käyttöjärjestelmään liittyy myös uusia verkkopohjaisen *Advanced Threat Protection* -suojauksen toimintoja, mutta niitä ei käsitellä tässä tutkimuksessa, koska ne eivät paranna julkisesta verkosta irtiolevien taktisten päätelaitteiden tietoturvallisuutta. [82, s.251]

3.1.4 Identiteetin suojaus

Merkittävä osa Windows 10-käyttöjärjestelmän tietoturvallisuudesta perustuu identiteetin suojaamiseen. Identiteetin suojaamiseen kuuluu muun muassa käyttäjän tunnistaminen ja todentaminen sekä pääsynhallinta. [1, 63, 67, 77, 79, 80, 83]

Kirjautuessaan Windows 10-käyttöjärjestelmään käytetään käyttäjänimen ja salasanan yhdistelmää, pin-koodia tai Microsoftin mukaan turvallisempia vaihtoehtoja: Windows Hello:n biometristä tunnistetta (kuten kasvoja, iiristä tai sormenjälkeä) tai fyysistä turva-avainta (toimikortti/laite). Kirjautumisen onnistuessa Windows luo käyttäjälle prosessin, jonka prosessiobjektiin (process object) liitetään pääsyosoitin (access token). [1, s.686–687]



Kuva 13 Windows 10 -käyttöjärjestelmän resurssien suojauksen tärkeimmät tietorakenteet [63, s.26]

Windows 10 -käyttöjärjestelmän resurssien suojaukseen kuuluu [1, s.688; 63, s.26–28; 80, s.17–22]:

- pääsyosoitus, joka pitää sisällään tiedon siitä, kuka käyttäjä on, mihin ryhmiin hän kuuluu ja mitä valtuuksia hänellä on. Prosessin tai säikeen oikeudet on myös tallennettu pääsyosoitukseen.
- turvatunniste (Security identifier, SID) joka on yksilöllinen turvatunniste, johon käyttöjärjestelmä sisäisesti viittaa myöntäessään oikeuksia
- käyttäjän turvatunniste (User SID), jonka avulla Windows tunnistaa käyttäjän (käyttäjätunnuksen) muista toimialueen käyttäjistä. Ihmiskäyttäjien lisäksi Windows antaa tietyille prosesseille ja palveluille omat turvatunnisteet.
- omistajan turvatunniste (Owner SID), joka määrittää kuka omistaa tietueen
- ryhmän turvatunniste (Group SID), joka määrittää mihin ryhmään käyttäjä kuuluu. Jokaisella ryhmällä on oma uniikki ryhmätunniste.
- etuoikeus (privilege), joka tarkoittaa oikeutta suorittaa käyttöjärjestelmään kohdistuvia toimintoja, joita ovat mm. tiedostojen ja hakemistojen varmistaminen, ohjelmien vian etsintä, tietoturvatapahtumien luominen ja järjestelmän sammuttaminen

- suojauskahva (Security Descriptor, SD), joka on suojattavien objektien kuvaus. Se sisältää muun muassa objektin omistajan SID:n, oletusryhmän, järjestelmän- ja normaalin pääsynhallintalistan sekä erinäisiä tilalippuja (Flags).
- käyttöjärjestelmän pääsynhallintaprosessi (Security Reference Monitor, SRM), joka on ydintilassa toimiva prosessi. SRM tekee päätöksen siitä, annetaanko prosessille oikeus käsitellä pyytämäänsä objektia.
- pääsynhallintalista (access control list, ACL), joka sisältää kahden tyypin pääsynhallintamäärityksiä: 1. Harkinnanvarainen pääsynhallintalista (Disrectionary Access Control List, DACL) perustuu identiteettiin valtuuksineen ja sääntöineen, jotka määrittelevät käyttäjien ja käyttäjäryhmien oikeudet käyttöjärjestelmässä 2. Järjestelmänpääsynhallintalista (System Access Control List, SACL) määrittelee pääsyn suojattuihin objekteihin. Kun prosessi esimerkiksi yrittää pääsyä suojattuun objektiin, käyttöjärjestelmä vertaa pääsyosoitinta olion DACL:iin ja sen jälkeen SACL:iin.
- pääsynhallintatietue (Access Control List Entry, ACE), joka määrittelee prosessin ja säikeet ja niiden oikeudet olioihin. Se on liitetty pääsynhallintalistaan.
- tilaliput (Flags), joita käytetään määrittelemään pääsynvalvontatietueen käyttäytyminen periytymisen suhteen.
- pääsynhallintamaski (Access Mask), joka määrittelee lopulta oikeudet tai auditointiasetukset olioille. Se sisältää muun muassa yleiset oikeudet sekä standardioikeudet. Yleisiä oikeuksia ovat: oikeus lukea, kirjoittaa, suorittaa tai tehdä nämä kaikki objektiin. Standardioikeuksia ovat: oikeus poistaa objekti, lukea objektin suojauskuvaus, muuttaa objektin oikeuksia tai omistaja sekä oikeus käyttää objektia synkronointiin.

Käyttäjätilin valvonta (User Account Control eli UAC) on Windows-käyttöjärjestelmien turvaominaisuus, jolla yritetään estää se, että ohjelma saa ilman järjestelmänvalvojan hyväksyntää järjestelmänvalvojaoikeudet. Vaikka käyttäjä olisi kirjautunut järjestelmänvalvojaoikeuksilla, ohjelmat joutuvat kysymään järjestelmänvalvojaoikeuksia erillisellä ikkunalla, jossa varmistetaan käyttäjältä, että ohjelman saa suorittaa laajennetuin valtuuksin. Ominaisuuden johdosta haittaohjelmat eivät lähtökohtaisesti kykene tekemään vain järjestelmänvalvojalle sallittuja asioita käyttäjän tietämättä. UAC ei vaikuta palveluprosesseihin (service), koska ne suoritetaan System (järjestelmäsovelluksien hallinnointi), LocalSystem (täydet oikeudet käyttöjärjestelmään), Network Service (etäkäyttö) käyttäjänä. [63, s.22; 80, s.22–23]

Credential Guard on Windows 10 -käyttöjärjestelmän virtuaalipohjaiseen suojaukseen (VBS) kuuluva teknologia, joka pyrkii suojaamaan *Paikallista käyttöoikeuksien hallintaa* (LSA) tunnistetietojen varastamiselta ja hyväksikäytöltä (Pass-the-hash, PTH). Toiminto käyttää virtuaalipohjaista suojausta ja eristää ohjelmien tunnistetiedot niin, että vain etuoikeutettu järjestelmäohjelmisto voi niitä käyttää. Käyttäjien todennukseen kuuluu oleellisesti todennusprotokollat: NT Lan Manager (NTLM) ja Kerberos. [67, 77, 79, 83]

3.1.5 Tiedon suojaus

Windows 10 -käyttöjärjestelmä käyttää tiedon suojaamiseen tiedostojärjestelmiä, laitesalausta ja tietojen suojausta. [22, 53, 84, 85, 86, 87, 88, 89]

Windows 10 -käyttöjärjestelmään sisältyvä ReFS-tiedostojärjestelmä (Resilient File System) on suunniteltu NTFS-tiedostojärjestemän pohjalta, tavoitteena maksimoida tietojen saatavuus, skaalauksen tehokkuus suurissa tiedostojärjestelmissä ja tiedostojen kloonaamisen nopeuttaminen virtuaalikoneen tarkastuspisteitä yhdistämällä (block cloning). Yksi ReFS:n haittapuolista on, että sitä ei voida käyttää samassa levyosiossa, jossa Windows -käyttöjärjestelmä sijaitsee, mutta sitä voidaan käyttää muissa massamuistin osioissa. [84, 85]

ReFS asettaa tarkistussumman meta- ja tiedostotietoihin. Lukiessaan, kirjoittaessaan ja skannatessaan (scrubber) tiedostoa, ReFS tutkii tarkistussumman varmistaakseen sen olevan oikein ja havaitsee näin mahdollisen tiedostokorruption. Tämän jälkeen ReFS korjaa korruptoituneen tiedoston varmuuskopiolla tai sellaisen puuttuessa poistaa kyseisen tiedoston. [84, 85]

Windows 10 -käyttöjärjestelmässä kiintolevyn tiedot voidaan salata salausmenetelmillä kuten Bitlockerilla (Windows 10 Pro, -Enterprise ja -Education) tai vaihtoehtoisesti Windowsin omalla laitesalauksella (device encryption). Levysalaus parantaa tiedon luottamuksellisuutta pyrkimällä estämään tiedon vuotamisen ulos järjestelmästä sekä luvattomilta muutoksilta, joita esimerkiksi laitteistotason haittaohjelma saattaa tiedon eheyden kustannuksella tehdä. [22, s.17; 86]

Bitlockerilla voidaan salata kokonaisia massamuistin loogisia osioita (volumes), jolloin valitusta loogisesta levystä salataan käytössä olevat osat, tai halutessa koko levy. Bitlockerilla voi salata myös yksittäisiä tiedostoja ja kansioita. Salaus on käytössä kaikilla käyttäjillä ja se voidaan ottaa käyttöön jo asennusvaiheessa. Ainoastaan järjestelmänvalvoja voi kytkeä Bitlocker

salauksen päälle tai pois. Bitlocker tallentaa salausavaimet TPM-suojapiiriin, jos se on määritetty ryhmäkäytänteissä (group policy) toimimaan näin sekä järjestelmässä on TPM-suojapiiri. [22, s.17; 86]

Yksittäisiä tiedostoja ja kansioita voidaan salata Windows 10 -käyttöjärjestelmän omalla salausohjelmalla Encrypting File System -salauksella eli EFS:llä. Yksi suurimmista haitoista verrattunna Bitlockeriin on, että salatut tiedostot ovat käyttäjätilikohtaisia, jolloin muut käyttäjät eivät voi avata tai muokata tiedostoja. Lisäksi datan avaamiseksi toisella tietokoneella tarvitaan isäntäkoneen EFS -sertifikaatti ja salausavain. Bitlockerista poiketen EFS tallentaa salausavaimet käyttöjärjestelmän tietokantaan. Tämä kasvattaa riskiä niiden hyväksikäytölle. [87, 88]

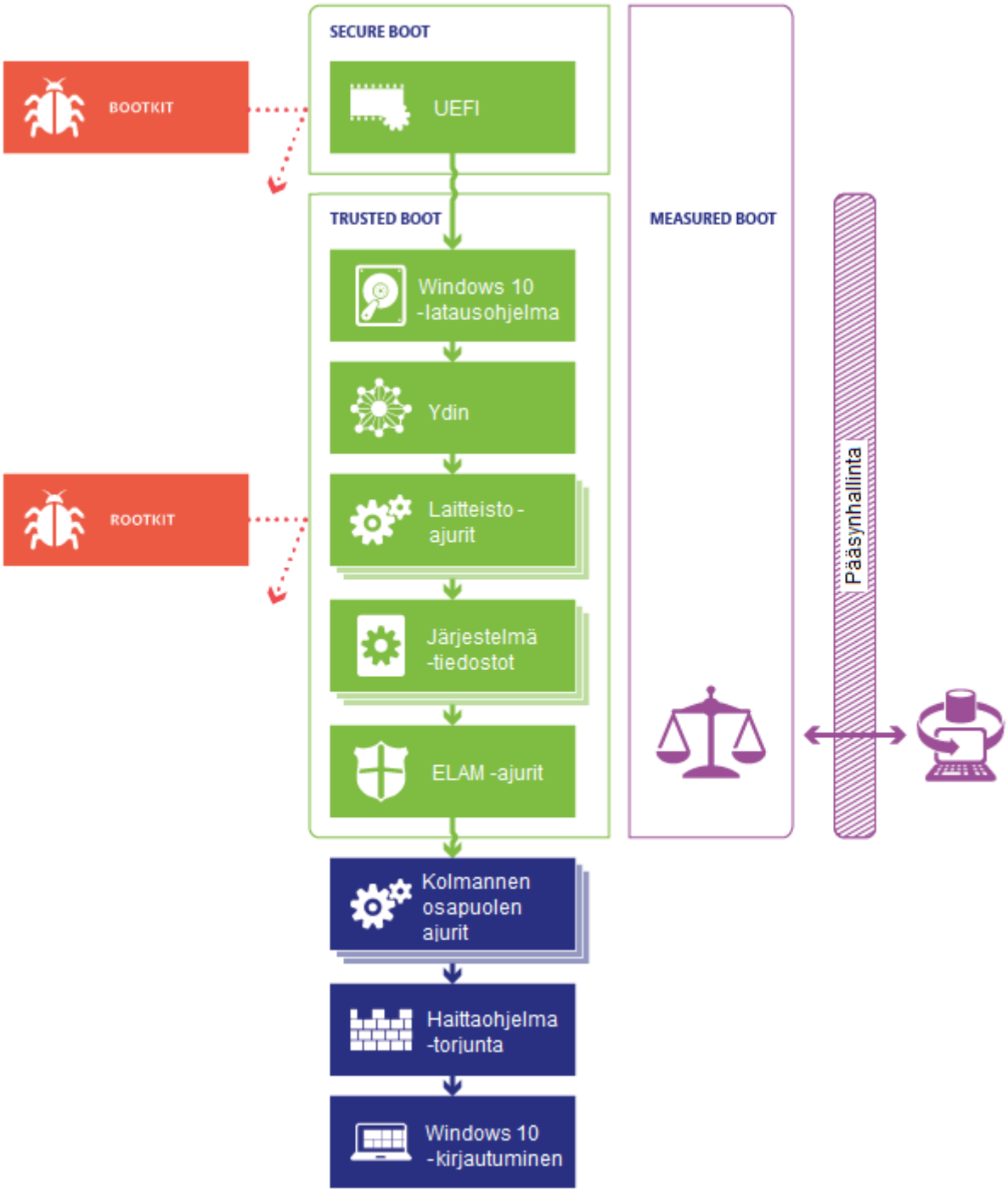
EFS käyttää oletuksena salaamiseen symmetrisen AES-256 -salauksen, epäsymmetrisen ECC-salauksen sekä SHA-tiivistefunktion yhdistelmää. Vaihtoehtona on myös 3DES-salaus. Bitlocker sisältää myös AES-256 -salauksen, joten EFS- ja Bitlocker-salauksen korkeimmat suojatasot ei olennaisesti eroa toisistaan. [87, 88]

Windows Information Protection (WIP) on Windows 10-käyttöjärjestelmään integroitu tietoturvatoiminto, joka hallitsee järjestelmän sovelluksia suojatakseen päätelaitteella olevaa tietoa. WIP:lle luodaan säännöt, jonka mukaan WIP salaa tietyt sensitiiviset tiedostot automaattisesti EFS:llä. Järjestelmänvalvoja voi määrittää, millä ohjelmilla on oikeus lukea ja kopioida tätä dataa. [89]

3.1.6 Tunkeutumisen havaitseminen ja vastatoimet

Aiemmat Windows 10 -käyttöjärjestelmän alaluvut käsittelivät toimia hyökkäyksen estämiseksi ennakkoon. Windows 10 -käyttöjärjestelmä pyrkii käynnistyksen aikaisilla suojauksilla ja Windows Defender -haittaohjelman torjuntaohjelmalla havaitsemaan ja torjumaan jo toteutuneen hyökkäyksen. [6, 22, 26, 61, 64, 90, 91, 92]

Windows 10 -käyttöjärjestelmällä on neljä tietoturvaominaisuutta suojautua ennen käynnistystä tapahtuvaa muokkausta, tai käynnistyksen aikaisten rootkittien ja käynnistyshaittaohjelmien (bootkit) tartunnalta. Näitä ovat: Secure Boot (ks. luku 2.4.7), Trusted boot (tarkastaa järjestelmän komponenttien eheyden ennen ajoa), Early Launch Anti-Malware (tarkastaa kaikki ajurit ennen niiden ajoa) sekä Measured Boot (tietokoneen laitteisto tallentaa lokiin käynnistysprosessin tiedot, joita Windows vertaa luotettavien palvelimien tietokantoihin. [26, 61]



Kuva 14 Windows 10 -käyttöjärjestelmän käynnistysprosessin suojaus [61, suomennos ylil. Jose Mäntylä]

ELAM -tietoturvatoiminto esiteltiin Windows 8 -käyttöjärjestelmässä. Toiminto mahdollistaa Windows 10 -haittaohjelmatorjunnan jo käyttöjärjestelmän käynnistyksen varhaisessa vaiheessa. ELAM on käytännössä ajuri, jonka Windows 10 käynnistää ennen muita ajureita. ELAM käynnistyy ennen tiedostojärjestelmäajureita, jolloin se ei kykene käyttämään tiedostojärjestelmän tiedostoja ilman omaa tiedostojärjestelmäajuria. ELAM -kuitenkin tarvitsee oman haittaohjelmien tunnistetietopankin tunnistaakseen haittaohjelmia, joten se ajetaan ELAM:n kanssa käynnistyksessä ja suljetaan tarpeen jälkeen. [26, s.7–8]

Haittaohjelmien torjunnassa yksi Windows 10 -käyttöjärjestelmän merkittävimmistä tietoturvaominaisuuksista on Windows Defender, joka on sen sisäänrakennettu haittaohjelmien torjuntaohjelma. Windows Defenderin pyrkimyksenä on antaa reaaliaikainen suojaus haittaohjelmilta käyttöjärjestelmän käytön aikana. Torjuntaohjelma kykenee tarkastelemaan ydintason toimintaa ja havaitsemaan tiedostojärjestelmästä löytyvät haittaohjelmat sekä käyttäytymisanalyysillä havaitsemaan haittaohjelmiin viittaavaa toimintaa. [6, s.974; 90]

Microsoftin suurin kilpailuetu suhteessa kaupallisiin Windows -haittaohjelmien torjuntaohjelmien valmistajiin on, että se saa todella paljon analysointitietoa Windows 10 -käyttäjiltään. Torjuntaohjelman uhkamäärityksiä ja uhkien havaitsemismenetelmiä päivitetäänkin päivittäin. [6, s.974; 90]

Windows 10 -käyttöjärjestelmän Defender käyttää aiempien versioiden tapaan Microsoft Malware Protection -moottoria (MMPE). Samaa haittaohjelmien torjuntamoottoria (malware engine) ja tunnisteita käytetään myös muissa Microsoftin haittaohjelmantorjunta -ohjelmissa. [6, 22]

Aikaisempiin Windows-versioiden haittaohjelman torjuntaohjelmiin verrattuna Windows 10 -käyttöjärjestelmän Windows Defender [22, s.14; 91]:

- havaitsee paremmin nollapäivähaavoittuvuuksia (ei yleisessä tiedossa tai olemassa olevaa korjausta) hyväksikäyttäviä haittaohjelmahyökkäyksiä
- sisältää pilvipohjaisia tarkkuutta tehostavia ominaisuuksia
- kykenee havaitsemaan muistissa pysyviä haittaohjelmia (ks. 2.6.2 Haittaohjelmien torjunta).

	Windows 10 (2019)	Windows 8.1 (2016)
Suojaus nollapäivä haittaohjelmahyökkäyksiltä	100 % (331)	88.1 % (163)
Viimeisen neljän viikon aikana havaitut haittaohjelmat	100 % (20428)	99.8 % (13681)
Vaikutus suorituskykyyn (teollisuuslaitekeskiarvo)	10.6 %	15.2 %
Väärien sovellushälytyksien määrä kuukaudessa	1	3

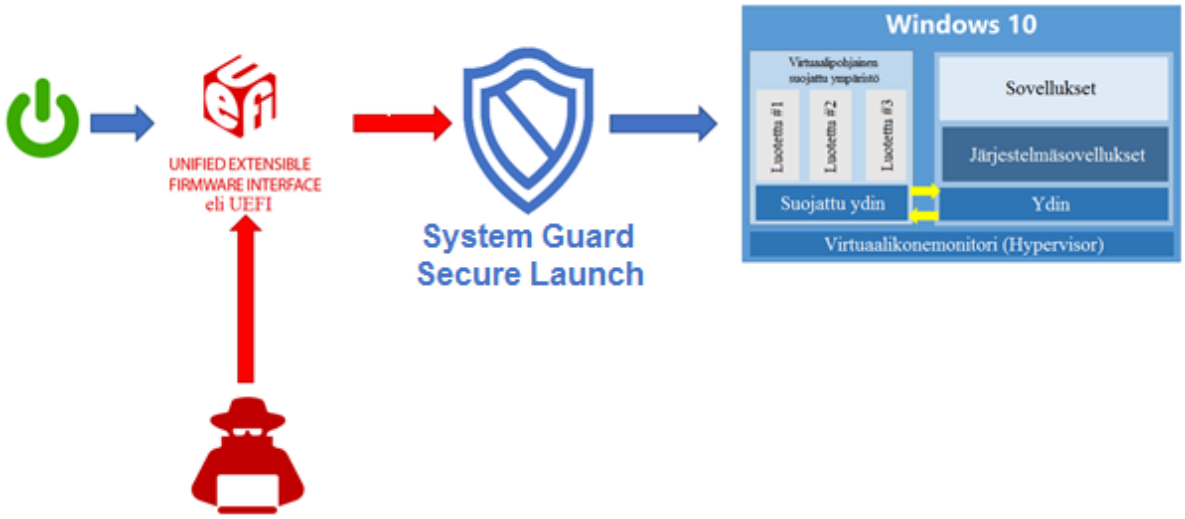
Taulukko 3 Windows 10- ja 8.1 -käyttöjärjestelmien haittaohjelman torjuntaohjelmien vertailu [91, 93]

Windows Defender Application Guard suojaa järjestelmän edistyneiltä hyökkäyksiltä käyttäen laitteistopohjaista eristystä. Tavoitteena on eristää esimerkiksi epäluotettavat verkkosivut ja tiedostot Windows Hypervisorin tuottamaan hiekkalaatikkoon, jossa niihin tehdään dynaaminen analyysi (ks. alaluku 2.4.3 Haittaohjelmien torjunta). [92]

Windows Defender System Guard:n tavoitteena on suojata Windows 10 -käyttöjärjestelmän kriittisiä resursseja (Windowsin autentintikointipino, biometrinen tunnistepino ja niin edelleen)

suojaamalla ja ylläpitämällä järjestelmän eheyttä jo ennen käyttöjärjestelmän käynnistymistä. [64]

Käynnistyksen aikaisen valvonnan toteuttaa *Dynaaminen luotettavuuden mittaava palvelu* (DRTM). Palvelu pyrkii torjumaan käynnistysten aikaisia uhkia (rootkitit, bootkit, lunnasohjelmat…) lisäämällä epäluotettavan tai haittaohjelmalla saastutetun UEFI:n suojatun käynnistyksen (secure boot, ks. luku 2.4.7) perään lisävaiheen: System Guard Secure Launch). Suojatun käynnistyksen jälkeen DRTM-palvelu siirtää System Guard Secure Launch -suojauksen avulla järjestelmän luotettuun tilaan (trusted state), jossa se pakottaa suorittimille tunnettua ja luotettavaa koodia ennen käyttöjärjestelmän käynnistymistä. [64]



Kuva 15 DRTM ja System Guard Secure Launch [64, suomennos ylil. Jose Mäntylä]

3.2. Debian -käyttöjärjestelmän tietoturvaominaisuudet

Debian 10 -käyttöjärjestelmä on Linux -ytimen (versio 4.19) päälle rakennettu käyttöjärjestelmä, jonka ensimmäinen versio julkaistiin vuonna 1993. Debian -käyttöjärjestelmä on Linux-puun merkittävä päähaara, jonka alla on suuri määrä Debianin dpkg-pakettienhallintaa käyttäviä jakeluhaaroja kuten esimerkiksi Ubuntu -käyttöjärjestelmä. Debian 10 eli ”Buster” sisältää yli 59 000 pakettia. Sen ”vakaa-versio” (stable) 10.0 julkaistiin heinäkuussa 2019 ja sen tuki (LTS) jatkuu vuoteen 2024 asti. [94, 95]

Mielekkään Debian 10- käyttöjärjestelmän ja vanhojen Debian -versioiden vertailun vuoksi toteutettiin tekninen koe. Testikoneelle asennettiin perusasetuksilla Debian 10 -pääversion viimeisin alaversio 10.3 (DVD). Asennuksen jälkeen käyttöjärjestelmästä tarkastettiin päätekomennolla ”apt list --installed”, mitä sovelluspaketteja asennuksen mukana käyttöjärjestelmään on asennettu. Tällä varmistettiin muun muassa se, että käyttöjärjestelmän asennuksen mukana

on asentunut tietoturvatyökaluja ja niiden tuottamia tietoturvaominaisuuksia, joita tutkimuksessa käsitellään.

Sovelluspakettien tutkimisen lisäksi testikoneessa ajettiin Python 3:lla ”kconfig-hardened-check”-työkalu, jolla saatiin tietoon asennuksen mukana tulevan 4.19 -version Linux-ytimen tietoturvaominaisuudet. Listan tietoturvaominaisuuksista osa esiintyy jo aiemmissa Debian -versioissa. Debian 10 -käyttöjärjestelmän ytimen tietoturvaominaisuudet on esitelty liitteessä 1.

Debian 10 -käyttöjärjestelmän tietoturvallisuutta on kehitetty aikaisemmasta 9-versiosta. Uuteen 10 -pääversioon ei ole lisätty tietoturvavaikutukseltaan kriittisiä ominaisuuksia, mutta monia vaikutukseltaan suuria kylläkin. [95; liite 1]

Uusia Debian 10 -pääversiossa käyttöönotettuja tietoturvaominaisuuksia ovat muun muassa [95; liite 1]:

- jakelupaketin Linux-ydin on päivitetty 4.9 versiosta 4.19 versioon (voidaan päivittää 5.4 versioon manuaalisesti)
- debian 10 -käyttöjärjestelmä voidaan asentaa päätelaitteelle, jossa on Secure Boot aktivoituna
- ytimen kekohyökkäyksien (kernel heap attacks) torjumista varten kehitetty ”Slab_Freelist_Hardened” -toiminnollisuus
- puskurinylivuotovirheiden havaitsemiseen tarkoitettu FORTIFY_SOURCE -kääntäjämakro (GCC ja GLIBC:lle) (ks. 3.2.3 Muistin eheys)
- ohjelmien turvallisuusprofiilien luomiseen tarkoitettu AppArmor -moduuli (ks. 3.2.4 Identiteetin suojaus)
- tietorakenteiden korruption tarkastamiseen tarkoitettu työkalu ”CHECK_ON_DATA_CORRUPTION” (ks. 3.2.5 Informaation suojaus)
- linux-ytimen moduulien autenttisuuden varmentamiseen tarkoitettu ”CONFIG_MODULE_SIG”-parametri
- Pakettihallintatyökalua (APT) on kovennettu niin, että se kykenee ajamaan ladattavia paketteja hiekkalaatikossa (Seccomp-hiekkalaatikossa), joka rajoittaa sallittuja järjestelmäkutsuja ja tekee ansoja mahdollisia haittaohjelmia vastaan
- myöhemmin tutkimuksessa käsiteltävä LUKS-levynsalaus käyttää uutta LUKS2 -formaattia, joka tuo metatietoon redundanssin, korruption havaitsemisen sekä konfiguroitavat PBKDF-algoritmit.

- autentikoinnin salaukselle (Authenticated Encryption, AE) tuki, joka pyrkii takaamaan samanaikaisesti käyttöjärjestelmän datan luottamuksellisuutta ja autenttisuutta
- NFTables -palomuuriohjelmisto on korvannut vanhat erilliset palomuuriohjelmat kuten Iptablesin. Uusia ominaisuuksia ovat muun muassa: atominen tuki (joustavampi sääntöjen korvaaminen), interaktiivinen konsoli, skriptituki, reaaliaikainen palomuurisääntöjen muutosten seuranta. [96]

Debiania asennettaessa voidaan valita useita valmiita konfiguraatioita. Näiden avulla voidaan asentaa vain tarvittavat osat käyttöjärjestelmästä. Tämä osaltaan vähentää hyökkäyspinta-alaa ja parantaa järjestelmän ylläpidettävyyttä. [80, s.34]

Debianin Linux-ydin itsessään sisältää monia suojausmenetelmiä, jotka suojaavat ytimen kohdistuvilta uhilta, joita aiheuttavat ohjelmointivirheet ja haavoittuvuuksien hyväksikäyttöyritykset. Ytimellä itsessään on keinoja, joilla se pyrkii havaitsemaan esimerkiksi hyökkäysyritykset. [97]

3.2.1 Hyökkäyspinta-alan pienentäminen

Debianin tärkein keino vähentää tietoturvauhkia ja hyökkäyspinta-alaa on estää keinot, joilla Linux-ydintä voidaan käyttää suorituksien uudelleenohjaamiseen. Tähän Linux-ydin pyrkii esimerkiksi rajoittamalla ohjelmistorajapintoja käyttäjätasolla tehden ydintilan ohjelmistorajapintojen virheellisestä käytöstä vaikeampaa sekä pienentämällä kirjoitettavaa ydintilan muistia. [97]

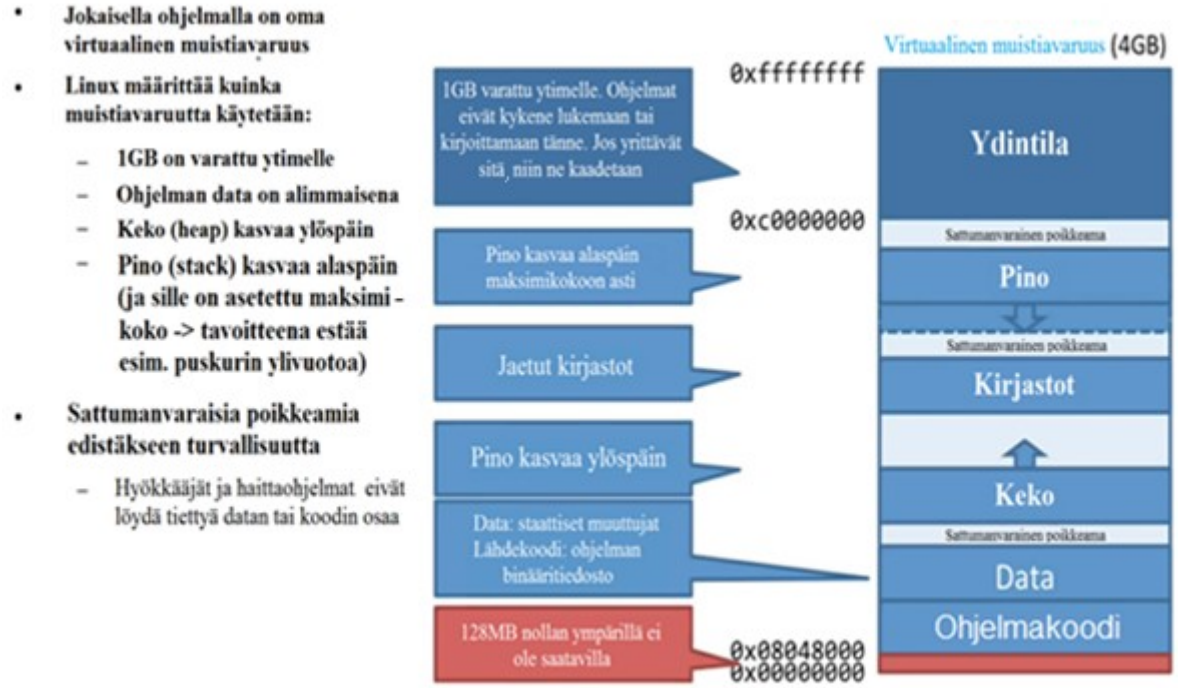
Muita Debian 10 -käyttöjärjestelmän ja Linux-ytimen keinoja vähentää hyökkäyspinta-alaa ovat [97, 98]:

- toiminnot Config_Strict_Kernel_RWX ja -Module_RWX, jotka estävät ytimen koodin ja kirjoitussuojatun datan ylikirjoittamisen
- tiedostojen suorittamisen estäminen ilman, että ne muutetaan käyttäjän toimesta erikseen suoritettavaksi
- pyrkiä estämään Linux-ytimen toimintaansa tarvittavien funktio-osoittimien (function pointers) ja sen muuttujien (kuvaajat/taulukot, tiedostorakenteita jne.) ylikirjoittamisen
- erottamalla ytimen ja käyttäjän muistiavaruus toisistaan (ks. 3.2.3 Muistin eheys)
- rajoittamalla prosessien järjestelmäkutsujen pääsyä (seccomp) ytimen käsittelyyn vain tietyistä kohdista (kernel entry points)

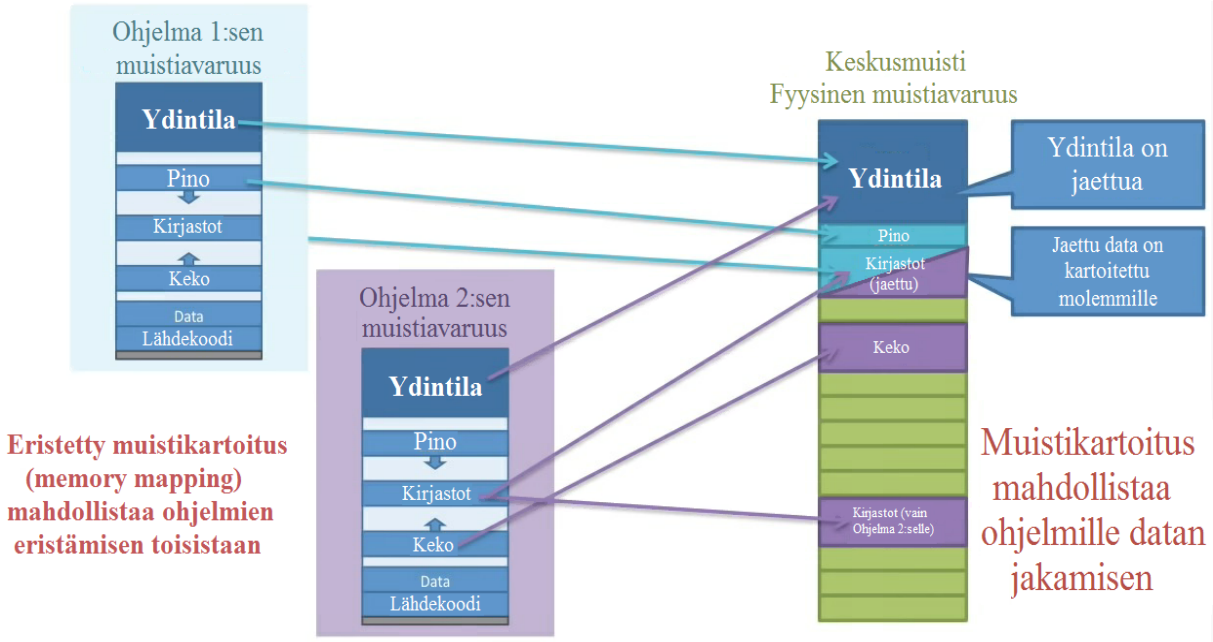
– rajoittamalla käyttäjien kykyä ladata ydinmoduuleita, mikä mahdollistaisi kyberhyökkääjälle kyvyn laajentaa järjestelmän hyökkäyspinta-alaa huomattavasti

3.2.2 Muistin suojaaminen

Debian 10 -käyttöjärjestelmän Linux-ydin sisältää Windows 10 -käyttöjärjestelmän tavoin monia muistinsuojaustoiminnollisuuksia. Se tukee muun muassa *Osoiteavaruuden satunnaistamistekniikkaa* (ASLR) ja muistin virtualisointia, joissa ohjelmille eristetään oma virtuaalinen muistiavaruus sekä puskurin ylivuotoa torjuva SSP-laajennus sekä FORTIFY_SOURCE. Osoiteavaruuden satunnaistamistekniikkaa ja virtualisointia esitellään kuvissa 16 ja 17. [80, s.34, 97, 99, 100, 101, 102]



Kuva 16 Ohjelman muistiavaruus [101, suom. ylil. Jose Mäntylä]



Kuva 17 Ohjelmien eristetty muistiavaruus [101, suom. ylil. Jose Mäntylä]

Debianin suojaukset estävät, että ydintila ei koskaan suorita ohjelmiaan käyttäjätilan muistissa (ks. kuvat 16 ja 17). Suojauksien johdosta ydin ei myöskään yritä päästä käsiksi käyttäjätilan muistiin ilman poikkeuslupaa. Näiden sääntöjen noudattamisen mahdollistaa Debian 10 -käyttöjärjestelmän tuki muun muassa suorittimien laitteistopohjaiseen SMAP-suojausominaisuuteen (Supervisor Mode Access Prevention), jonka avulla pääkäyttäjäoikeuden omaavat ohjelmat asettavat ”ansoja” (traps) käyttäjätilan muistiin. Tällä vaikeutetaan haittaohjelmien kykyä huijata ydintä käyttämään käskyjä tai dataa käyttäjätilan muistiavaruudessa. [97]

Debian 10 -käyttöjärjestelmän mukana tulee versio 8 GNU -projektin kääntäjien kokoelmasta (GNU Compiler Collection, GCC), joka toimii kääntäjänä muun muassa C ja C++ -kieliin. Kokoelman mukana on SSP-laajennus (Stack Smashing Protector), joka pyrkii havaitsemaan pinon ylivuodot asettamalla satunnaisen arvon omaavan suojamuuttujan (Stackguard) pinoon. Arvon muuttuessa suoritus keskeytetään ja tiedot tallennetaan lokiin. [99; 100, s.70–71]

Muutoksena Debian 9:stä Debian 10 -pääversioon GCC-kokoelmassa on nyt otettu käyttöön ”FORTIFY_SOURCE”-kääntäjämakro, joka pyrkii havaitsemaan tietyntyyppisiä puskurin ylivuotovirheitä. Makro tarkastaa puskurin ylivuodon riskiä aiheuttavien funktioiden kuten esimerkiksi aiemmin tutkimuksessa (ks. alaluku 2.5.5 Puskurin ylivuoto) käsitellyn ”strcpy”-funktion ajoa. Makro laskee kopioitavana olevien bittien määrää siten, etteivät ne ylitä puskuria. [102; liite 1]

3.2.3 Uhkan vastustaminen

Debian 10 -käyttöjärjestelmässä on Linux-jakelupakettien palomuuri. Palomuuri rajoittaa Netfilter -rajapinnan kautta kulkevaa tietoliikennettä rajoittamalla auki olevia portteja. Asennuksen jälkeen kaikki portit ovat auki, mutta pääkäyttäjä kykenee NFTables -palomuuriohjelmistolla luomaan palomuurisääntöjä verkkoliikenteen hallitsemiseksi. Linux-palomuurilla ei kuitenkaan voida asettaa ohjelmasääntöjä. [96, 103]

3.2.4 Identiteetin suojaus

Debianin Linux-ytimen Linux Security Modules (LSM) on käyttöliittymä, jonka avulla mahdollistetaan useampia pääsynhallintamalleja [80, s.34]

Debian-käyttäjät saavat uniikin käyttäjätunnuksen UID:n (User ID). UID on kokonaisluku 0 ja 65535 väliltä. Tiedostot, prosessit ja muut järjestelmäresurssit on merkattu tietyille käyttäjille. Käyttäjätasoja on kolme. Lähtökohtaisesti tiedoston omistaja on persoona, joka teki kyseisen tiedoston. On olemassa kuitenkin keino vaihtaa omistajuutta. Järjestelmänvalvojana voidaan käyttäjät organisoida tietyn tai tiettyjen ryhmätunnuksien eli GID:den alle (Group ID). [6, s.798–799]

Kirjaimellisesti Debian -käyttöjärjestelmässä ei rajoiteta käyttäjien oikeuksia vaan tiedoston luontihetkellä tiedoston oikeuksia. Ne määrittävät omistajan, muiden omistajaryhmään kuuluvien käyttäjien sekä muiden käyttäjien oikeudet tiedostoon. Oikeudet voidaan jakaa kolmeen ryhmään: oikeus lukea (r), oikeus kirjoittaa (w) ja ohjelmatiedostoille oikeus suorittaa (x). Ohjelma voidaan suorittaa myös muilla kuin kirjautuneen käyttäjän tunnuksilla. [6, s.799]

Binääri	Symboolit	Tiedosto-oikeudet
111000000	rwx------	Omistaja voi lukea, kirjoittaa ja suorittaa
111111000	rwxrwx---	Omistaja ja ryhmä voi lukea, kirjoittaa ja suorittaa
110100000	rw-r-----	Omistaja voi lukea ja kirjoittaa; ryhmä voi lukea
110100100	rw-r--r--	Omistaja voi lukea ja kirjoittaa; kaikki muut lukea
111101101	rwxr-xr-x	Omistaja voi tehdä kaikkea, muut voivat lukea ja suorittaa
000000000	---------	Kenelläkään ei oikeuksia tiedostoon
000000111	------rwx	Ulkopuolisilla on vain oikeus lukea, kirjoittaa ja suorittaa

Taulukko 4 Esimerkki Linuxin tiedostosuojauksesta [6, s.799, suomennos ylil. Jose Mäntylä]

AppArmor on uusi Linux-jakelupakettien kuten Debian 10:n suojautumiskeino haittaohjelmia vastaan. Se on oletuksena päällä Debian 10 -pääversiossa. AppArmor on LSM-käyttöliittymän päälle rakennettu pakollinen pääsynhallintajärjestelmä (mandatory access control, MAC), jolla ohjelmille luodaan oma turvallisuusprofiili. Käytännössä Linux-ydin tekee kyselyn AppArmor:lle ennen järjestelmäkutsujen suorittamista kysyäkseen, että onko prosessilla oikeus suorittaa kyseinen toiminto. Tätä mekanismia käyttäen AppArmor rajoittaa ohjelmien toimintoja ja resursseja. [104]

3.2.5 Tiedon suojaus

Tiedon luottamuksellisuuden suojaamiseen Debian 10 -käyttöjärjestelmällä on käytössään muun muassa LUKS laitteistoriippumaton standardi sekä tiedon eheyden suojaamiseen muun muassa ”CHECK_DATA_CORRUPTION” -toiminnollisuus. [105; 106, 107, liite 1]

LUKS (Linux Unified Key Setup) on monissa Linux-jakelupaketeissa kuten Debian 10:ssä käytetty laitteistoriippumaton standardi, joka sisältää muun muassa *cryptsetup*-levynsalaus- ja *cryptomount*-salauksenpurku -työkalut. LUKS:n *cryptsetup* levynsalaustyökalua voidaan käyttää jo käyttöjärjestelmän asennusvaiheessa salattujen levyosioiden luomiseen. [105]

LUKS:in toiminnan mahdollistaa Linuxin DMCrypt -ydinmoduuli. Moduuli mahdollistaa myös monien muiden muistityyppien kuin massamuistien salaamisen. LUKS mahdollistaa eri tiivistefunktioiden ja avaimien erikoon käytön, mutta suositeltavaa on käyttää vakioasetuksellista kahden AES-512 salausalgoritmin yhdistelmää (XTS-moodi), joka on hyvin suorituskykyinen ja tietoturvallinen vaihtoehto. [105]

LUKS:in suurimpia etuja on, että se on (versio 2.3) yhteensopiva Bitlocker-salauksen kanssa, jolloin Debian 10 -päätelaitteilla kyetään avaamaan ja kirjoittamaan Bitlocker-salattujen massamuistien ja muiden tiedontallenusvälineiden dataa (BITLK-formaatti). [105, 106]

Erillisten tiedostojen salaamiseen voidaan käyttää Debian 10 -käyttöjärjestelmän mukana tulevaa OpenSSL-työkalua. OpenSSL-työkalulla tiedostot voidaan salata ja purkaa käyttäen esimerkiksi AES-256 -algoritmia. [108]

“CHECK_DATA_CORRUPTION” -toiminnollisuuden avulla Linux-ydin tarkastaa ja suojaa tietorakenteita korruptiolta. Kun tietorakenteiden korruptio havaitaan, sen aiheuttanut ohjelma kaadetaan tai siitä annetaan varoitus käyttäjälle ja muille ohjelmille, joka on tietoturvallisempi vaihtoehto. [107, liite 1]

3.2.6 Tunkeutumisen havaitseminen ja vastatoimet

Kaikki käyttöjärjestelmät, kuten myös Debian 10, ovat alttiita haittaohjelmille, mutta on huomioitava, että tietokoneiden eri Linux-jakelupaketeissa ja niiden versioissa leviäviä haittaohjelmia on merkittävästi vähemmän kuin huomattavasti suuremman markkinaosuuden omaavissa Windows -käyttöjärjestelmissä. Tyypillisesti haittaohjelman torjuntaohjelmia asennetaan postipalvelimille tai palvelinkoneille etsimään ja torjumaan Windows-haittaohjelmia. [109, s.1; 110]

Linux-jakelupakettien (mukaan lukien Debian 10) mukana tuleva avoimen lähdekoodin ClamAV -haittaohjelmien torjuntaohjelma on suunniteltu havaitsemaan sekä torjumaan troijalaisia, viruksia sekä muita haittaohjelmia ja uhkia. ClamAV käyttää tähän eri menetelmiä kuten sähköposti- ja verkkoskannausta. Päätelaiteturvallisuus on myös yksi ClamAV:n tavoitteista ja yksi sen keinoista tähän on reaaliaikainen suojaus, joka estää pääsyn tiedostoihin ennen kuin ne on tarkastettu. [109, s.3; 110]

4. KÄYTTÖJÄRJESTELMIEN TIETOTURVAKOVENNUKSET

4.1. Kovennukset

Käyttöjärjestelmien kehittäjät kohtaavat dilemman päättäessään, mitä toiminnollisuuksia tehdasasetuksilla käyttöjärjestelmässä on päällä. Haluna voi olla esitellä kaikkia mahdollisia ominaisuuksia, markkinoida käyttöjärjestelmää turvallisena ja asentamalla vain välttämättömät toiminnallisuudet, tai jotain siltä väliltä. Käyttöjärjestelmät voivatkin olla alttiita tietoturvan vaarantumiselle alkuperäisillä asetuksillaan. Koventamalla taktisien päätelaitteiden käyttöjärjestelmiä voidaan huomattavasti vähentää hyökkäyspinta-alaa. [45, s.541; 67]

Itse käyttöjärjestelmän koventamisen tavoitteena on turvata käyttöjärjestelmä itsessään, mihin kaikki palvelut ja sovellukset tukeutuvat. Suurimmassa osassa käyttöjärjestelmiä alkuperäisasetuksissa usein suositaan käytettävyyttä tietoturvan kustannuksella. Poikkeuksen tekevät pelkästään tietoturvallisuuden näkökulmasta suunnitellut käyttöjärjestelmät kuten esimerkiksi luvussa 2.3 käsitelty Astra Linux tai Arch Linux. Sen lisäksi jokaisella organisaatiolla on omat käyttötarpeet, joiden perusteella luodaan erilaiset profiilit asetuksineen. Tämän takia kovennukset täytyy suunnitella organisaatio- ja käyttökohdekohtaisesti. [1, s.681; 67]

Taktisten päätelaitteiden käyttöjärjestelmien koventamiseen on olemassa valmiita tietokantoja, jotka sisältävät eri järjestelmiin tehtävät suosituskovennukset. *Center for Internet Security* (CIS) on tietoturvaorganisaatio, joka on julkaissut kovennustoimenpideoppaita eri käyttöjärjestelmiin CIS Benchmark -sivustolla. [111]

Sivustolla valtionhallinnot, yritykset, teollisuus ja oppilaitokset päättävät konsensus-periaatteella käyttöjärjestelmiin tehtävät kovennukset, jotka julkaistaan CIS -oppaina. Sivustolta saa ostettua CIS -oppaan mukaisesti kovennettuja valmiita käyttöjärjestelmien levykuvia (muun muassa Debian 9) tai ladattua vaihtoehtoisesti suosituskovennuksia asentavia paikkaustyökaluja. [111]

CIS -oppaissa suositeltavia kovennustoimenpiteitä on määrällisesti sitä enemmän, mitä enemmän käyttöjärjestelmän hyökkäyspinta-alaa kaventavia toimenpiteitä on. Kovennukset lajitellaan sen perusteella, onko niillä vaikutusta lopullisen CIS Benchmark -pisteytykseen. Pisteytystä arvioidaan käyttöjärjestelmissä erikseen ajettavilla CIS -työkaluilla. Seuraavassa taulukossa on vertailtu korkean tietoturvatason (taso 2) päätelaitteiden käyttöjärjestelmiin tehtävien suosituskovennustoimenpiteiden määrä. [111, 112, 113]

Käyttöjärjestelmä	Microsoft Windows 8.1	Microsoft Windows 10 E5 1903	Debian 9	Debian 10
Vaikutus pisteytykseen	326	445	187	212
Ei vaikutusta pisteytykseen	5	1	30	22
Yhteensä	331	446	217	234

Taulukko 5 CIS -oppaan suosituskovennustoimenpiteet korkean tietoturvatason (tason 2) päätelaitteille [112, 113]

CIS -oppaan suosituskovennustoimenpiteiden määrästä taulukko 5 voidaan havaita, että käyttöjärjestelmien koventamiseen vaaditaan sekä Windows 10- ja Debian 10 -käyttöjärjestelmiin aikaisempia versiota enemmän asetusmuutoksia ja muita kovennustoimenpiteitä. Debian 10 -käyttöjärjestelmään tarvitaan noin puolet vähemmän asetusmuutoksia kuin Windows 10 -käyttöjärjestelmään. Kovennusmäärän näkökulmasta Debian 10 -käyttöjärjestelmä on tietoturvallisempi tehdasasetuksilla, kuin Windows10 -käyttöjärjestelmä.

4.1.1 Asennustiedosto

Järjestelmän tietoturvallisuus pitää ottaa huomioon jo asennusvaiheessa, ja myös ennen sitä. Käyttöjärjestelmä on päivittämättömänä ja asennustiedostomuodossa aina helppo kohde, jos hyökkääjä pääsee siihen käsiksi. Asennuspalvelin asennustiedostoineen tulisi olla omassa suojatussa verkossa, tai vaihtoehtoisesti sellaista ei käytetä ja päätelaitteet asennetaan ulkoiselta tiedontallennusvälineeltä. [1, s.682]

Muita menetelmiä suojata järjestelmää asennusvaiheessa on [1, s.682; 45; 67]:

- sallia käyttöjärjestelmän asennuksen käynnistyminen vain tietyltä palvelimelta, tai ottaa verkkokäynnistystapa aiemmin mainituista syistä kokonaan pois päältä
- salaamalla BIOS / UEFI omalla salasanalla
- sallimalla vain luotettujen ajureiden asentaminen, koska laiteajureilla on ydintason oikeudet ja tästä johtuen haitallinen laiteajuri kykenee ohittamaan monia turvallisuustoimintoja
- päivittämällä käyttöjärjestelmä turvalliseksi todettuun versioon, johon kaikki tarvittavat kovennukset ovat jo tiedossa ja saatavilla
- asentamalla esimerkiksi Linux-pohjaisiin käyttöjärjestelmiin tietoturvakovennettu ydin (Linux-hardened)

4.1.2 Käyttöjärjestelmän tietoturvapäivitysnopeus

Vähentääkseen kyberhyökkäyksen riskejä taktisten päätelaitteiden sovelluksiin ja käyttöjärjestelmään pitää asentaa tietoturva-auditoituja päivityksiä riittävän nopeassa aikakehyksessä. Tähän vaikuttaa erityisesti järjestelmään ja informaatioon kohdistuva haavoittuvuuksien riski, jota niissä käsitellään sekä säilötään. Päivitystuen loppuessa pitää sovelluksen tai käyttöjärjestelmän käytöstä luopua kovennetuissa järjestelmissä. [67, s.2; 114]

Käyttöjärjestelmä- ja sovellushaavoittuvuuden julkaisemisen jälkeen, haavoittuvuutta hyväksikäyttäviä haittaohjelmia ja komentosarjoja aletaan työstää heti, sekä niitä julkaistaan jopa tunneissa. Niistä osaa ei tuoteta hyökkäysmielessä [114, s.2]

Päivityksien tietoturvatestauksen ja asennuksen suositeltu aikakehys eri CVSS -kriiteeristön mukaisissa vakavuusluokissa [114, s.2]:

- kriittiset haavoittuvuudet = 48 tunnin sisällä päivityksen julkaisusta
- korkean riskin haavoittuvuudet = kahden viikon sisällä päivityksen julkaisusta
- keski- ja matalan tason haavoittuvuudet = kuukauden sisällä päivityksen julkaisusta

Väliaikaisena korjauksena ennen päivityksen saatavuutta voidaan haavoittuvuuden sisältämiä toiminnollisuuksia kytkeä pois päältä. Vaihtoehtoisesti voidaan palomuurilla sekä muilla asetuksilla rajoittaa tai estää kyberhyökkääjän pääsy haavoittuvaan palveluun. [114, s.3]

4.1.3 Tarpeettomat palvelut ja ohjelmat

Kaikki käyttöjärjestelmiin asennettavat sovellukset eivät alkuperäisillä asetuksillaan ole välttämättä turvallisia. Asennettuna saattaa olla sovelluksia ja aktivoituna paljon toimintoja, joita sotilaallisessa käyttöympäristössä ei tarvita. Sisäänrakennettuna turvaominaisuudet saattavat olla pois päältä tai asetettuna alhaiselle turvatasolle. Hyvänä esimerkkinä voidaan pitää PDF-lukuohjelmia ja monia selaimia, joissa JavaScriptin suorittaminen on oletuksena sallittua. [1, s.683; 67]

Riskien alentamiseksi jo asennusvaiheessa poistetaan ja deaktivoidaan tarpeettomat tai haavoittuvuuksia sisältävät sovellukset kovennetusta käyttöjärjestelmästä, aktivoidaan kaikki mahdolliset tietoturvaominaisuudet asennetuissa sovelluksissa sekä deaktivoidaan asennettujen sovelluksien ominaisuudet, joita ei tarvita. [1, s.683; 45, s.541; 67]

Kyberhyökkääjä saattaa saada haittaohjelman taktisiin päätelaitteisiin esimerkiksi ulkoisten muistilähteiden välityksellä. Haittaohjelma pyrkii käyttämään hyödyksi asennettujen ohjelmien haavoittuvuuksia, eikä sitä välttämättä tarvitse asentaa ohjelmamuodossa käyttöjärjestelmään. [1, s.683; 67, s. 2–3; 115]

Riskin vähentämiseksi kovennetuissa käyttöjärjestelmissä käytetään *Application Whitelisting* -tietoturvamallia, jossa sallitaan vain listan mukaisten luotettujen ohjelmien ajo käyttöjärjestelmässä, ja kaikkien muiden sovelluksien ajo estetään. Tietoturvamallin noudattamisen lisäksi kovennettuihin järjestelmiin pitää asentaa vain valmistajan digitaalisen allekirjoitusvarmenteen (ks. alaluku 2.4.6) omaavia ohjelmia. [1, s.683; 67, s. 2–3; 115]

4.1.4 Käyttöoikeuksien hallinnan ja käyttäjän todentamisen tiukentaminen

Windows 10- ja Debian 10 -käyttöjärjestelmät säilövät käyttäjien kirjautumiseen vaadittavia käyttäjätietoja kuten esimerkiksi salasanoja. Windows 10 -käyttöjärjestelmässä tietokannan nimi on SAM, eli *Security Accounts Manager*. Tietokannan avulla käyttäjät voivat kirjautua järjestelmään, vaikka kirjautumispalvelu ei olisi saatavilla. [67, s.6]

Vaikka tämä toiminnallisuus on tiedon saatavuuden kannalta ihanteellista, kyberhyökkääjät kykenevät hyväksikäyttämään tätä toiminnollisuutta saadakseen pahimmillaan jopa toimialueen järjestelmävalvoja (*domain administrator*) -oikeudet käsiinsä. Vähentääkseen riskiä käyttöjärjestelmä konfiguroidaan niin, että kirjautumistietokantaan tallennetaan vain edeltävät kirjautumistiedot tai estetään kokonaan toiminnallisuuden käyttö. [67, s.6]

5. KÄYTTÖJÄRJESTELMIEN HAAVOITTUVUUDET JA CVSS -ANALYSOINTIMENETELMÄ

5.1. CVE-tunniste ja CVSS -analyysi

Ohjelmisto, laitteisto ja laiteohjelmistojen haavoittuvuudet aiheuttavat kriittisen riskin käyttöjärjestelmälle, mutta huono käyttöjärjestelmä voi olla riski hyvälle ohjelmalle. Haavoittuvuuksia tarkasteltaessa vastaan tulee tunniste CVE (*Common Vulnerabilities and Exposures)*. Kyseessä on yleinen haavoittuvuuksien hallintamenetelmä, joka sisältää 15 viime vuoden aikana julkaistut tunnetut haavoittuvuudet. CVE-tunnisteen voi hakea yksilöimään löytämäänsä haavoittuvuutta riippumatta siitä, onko hakija esimerkiksi yksityinen henkilö tai suuryritys. [116, 117, 118]

CVE-tunniste muodostuu kolmesta osasta: alkuosa, vuosiluku ja juokseva numero eli esimerkiksi CVE-2018-8174 (*Double kill* -haavoittuvuus). [117, 119]

CVSS -analyysi (Common Vulnerability Scoring System) on CVE-tietokannan haavoittuvuuksien vakavuuden arviointiin käytettävä avoin, standardoitu ja kaupallisista tavoitteista riippumaton pisteytysjärjestelmä. Analyysin avulla pisteytetään haavoittuvuuksien ominaisuuksia ajan ja käyttäjäympäristön suhteen. Sen avulla organisaatio pystyy muodostamaan kuvan haavoittuvuuden vakavuudesta. Vakavuustarkastelun avulla pystytään priorisoimaan ja kohdentamaan toimet haavoittuvuuksien aiheuttamien tietoturvauhkien paikkaamiseksi. Viimeisin versio pisteytysjärjestelmästä on versio 3.1. [117, 116]

CVSS-pisteiden vakavuusasteikko [120, s.16]:

- 0: Ei uhkaa
- 0.1-3.9: Matala
- 4.0-6.9: Keskitaso
- 7.0-8.9: Korkea
- 9.0-10.0: Kriittinen

Lauri Maskulinin ”BYOD-laitteiden tietoturva” ja Ville Rantamäen "Taisteluosastoon kohdistuva kyberuhkat” pro gradu -tutkielmissa tutkittiin CVSS:n 3.0-versiota. Muutokset uuteen 3.1 versioon ovat [121]:

- CVSS-analyysi mittaa nyt mahdollisten organisaatio -ja toimintaympäristökohtaisten riskien sijaan haavoittuvuuksien vakavuuttta. Perusteluna tähän oli, että vakavuutta

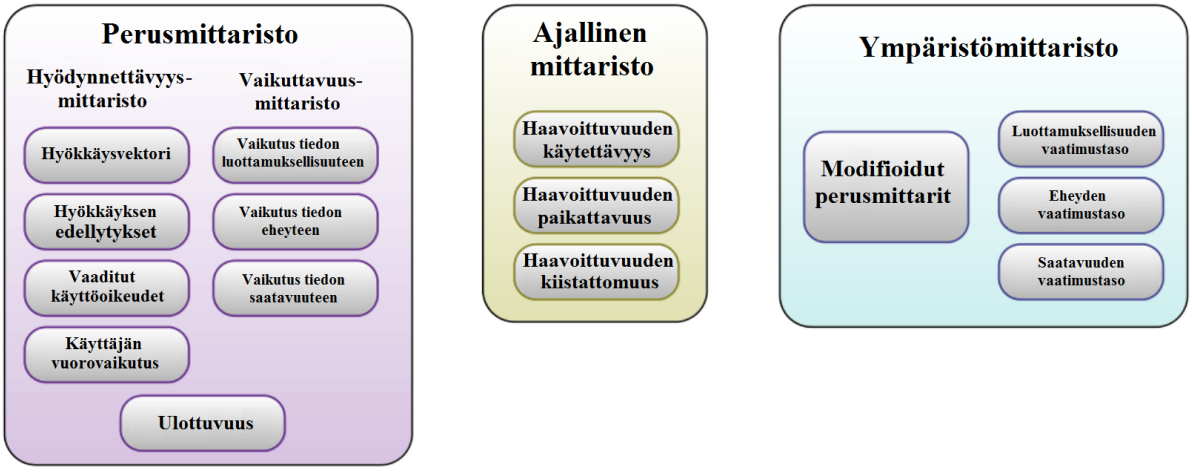
voidaan käyttää osana riskianalyysia, mutta riskin määrittäminen täysin vaatii aina ulkoisten tekijöiden huomioonottamista.

- vektoreiden kuvauksia on muutettu suuntaan, jossa ne ovat selkeämpiä analyysin käyttäjille sekä vältetään viittauksia OSI-malliin
- muutoksia pisteytyksessä ja laskukaavoissa
- CVSS-analyysia voidaan nyt laajentaa eri organisaatioiden, kuten vaikkapa Puolustusvoimien tekemillä omilla standardoiduilla mittareilla. Tällöin voidaan ottaa esimerkiksi taktisiin päätelaitteisiin tai taisteluosaston runkoverkkoon kohdistuvat tekijät huomioon haavoittuvuuksien vakavuuden analyysissä.

5.1.1 CVSS-mittaristot

Lopullinen CVSS -pisteytys perustuu kolmeen eri pääosa-alueeseen sekä organisaation omiin siihen liitettäviin mittaristoihin. [120]:

- Perusmittaristo (base) kuvaa haavoittuvuuden luonteenomaisia piirteitä sekä perusominaisuuksia, jotka ovat muuttumattomia ajan ja toimintaympäristön suhteen.
- Ajallinen mittaristo (temporal) kuvaa haavoittuvuuden aikasidonnaisia piirteitä, jotka pitävät paikkaansa vain haavoittuvuuden arviointihetkellä.
- Ympäristömittaristo (environmental) kuvaa haavoittuvuuden piirteitä, jotka ovat merkittäviä ja uniikkeja tietyssä toimintaympäristössä.



Kuva 18 CVSS -arviointikriteeristön mittarit [120, suomennos ylil. Jose Mäntylä]

Hyökkäysvektorin -arvo (HV) määräytyy sen mukaan, mistä haavoittuvuutta voidaan hyväksikäyttää. Etäisyyden kasvaessa hyökkäysvektoripiste kasvaa. Arvot ovat seuraavat [120, s.5–6]:

- Verkko (V): haavoittuvuutta voidaan hyväksikäyttää ulkoverkosta (esim. Internet)

- Rinnakkainen (R): haavoittuvuutta voidaan hyväksikäyttää jaetun fyysisen (esim. Bluetooth) tai loogisen verkon (esim. sisäverkko) yli
- Paikallinen (P): haavoittuvuutta voidaan hyväksikäyttää vain paikallisesti (esim. näppäimistö ja takaovi)
- Fyysinen (F): haavoittuvuutta voidaan hyväksikäyttää vain fyysisellä pääsyllä kohdejärjestelmään

Hyökkäyksen edellytykset -arvo (HE) määräytyy sen mukaan, kuinka paljon kyberhyökkääjän kontrolloimattomia edellytyksiä haavoittuvuuden hyväksikäyttö vaatii. Mitä alhaisempi vaatimustaso on, sitä enemmän HE-pisteitä se saa. Arvot ovat seuraavat [120, s.6–7]:

- Matala (M): erikoisehtoja järjestelmään pääsyyn tai kykyä olosuhteiden lieventämiseen ei ole. Hyökkääjä kykenee toistamaan hyökkäyksen haavoittuvaan järjestelmään.
- Korkea (K): Hyökkäyksen onnistumiselle on ehtoja, joita hyökkääjä ei kykene kontrolloimaan ilman huomattavaa resurssien käyttöä. Hyökkääjän tarvitsee esimerkiksi kerätä haavoittuvuuden sisältävästä järjestelmästä tietoa tai valmistella kohdejärjestelmää parantaakseen hyökkäyskeinon toimintavarmuutta.

Vaaditut käyttöoikeudet -arvo (VK) määräytyy sen mukaan, minkä tasoiset pääsyoikeudet kyberhyökkääjä tarvitsee ennen kuin voi hyväksikäyttää haavoittuvuutta. Jos pääsyoikeuksia ei tarvita, sitä suurempi VK-arvo on. Arvot ovat seuraavat [120, s.7–8]:

- Ei oikeuksia (E): hyökkääjä ei tarvitse pääsyoikeutta haavoittuvaan kohdejärjestelmään toteuttaakseen kyberhyökkäystä
- Matala (M): hyökkääjä tarvitsee käyttäjätason oikeudet kohdejärjestelmään
- Korkea (K): hyökkääjä tarvitsee järjestelmänvalvojatason oikeudet kohdejärjestelmään

Käyttäjän vuorovaikutus -arvo (KV) määräytyy sen mukaan, kuinka paljon kohdejärjestelmän käyttäjää tarvitaan haavoittuvuutta hyväksikäyttävän kyberhyökkäyksen onnistumiseen. Mitä vähemmän käyttäjän vuorovaikutusta haavoittuvuuden hyväksikäyttöön tarvitaan, sitä suurempi KV-arvo on. Arvot ovat seuraavat [120, s. 8–9]:

- Ei tarvetta (E): ei tarvetta käyttäjän vuorovaikutukselle
- Tarvitaan (T): käyttäjän vuorovaikutusta, kuten esimerkiksi jonkun toiminnon suorittamista, haavoittuvuuden hyväksikäyttöön tarvitaan

Ulottuvuus -arvo (U) määräytyy sen mukaan, kyetäänkö haavoittuvuuden hyväksikäytöllä pääsemään käsiksi järjestelmän sisällä oleviin toisiin kohteisiin. U-arvo on sitä suurempi, mitä laajempi ulottuvuus järjestelmän muihin kohteisin on. Arvot ovat seuraavat [120, s.8–9]

- Ei ulottuvuutta (E): haavoittuvuuden hyväksikäytöllä päästään käsiksi vain kohdeohjelmaan ja sen resursseihin.
- Jatkaminen (J): haavoittuvuuden hyväksikäytöllä päästään käsiksi kohdejärjestelmän muihin resursseihin, jotka ovat alkuperäisen kohdeohjelman pääsyn ulkopuolella. Hyökkäystä kyetään mahdollisesti jatkamaan verkon muihin järjestelmiin.

Kaikki vaikutusarvot ovat sitä suurempia, mitä isompi vaikutus tiedon luottamuksellisuuteen, eheyteen tai saatavuuteen on [120, s.10–11].

Vaikutus tiedon luottamuksellisuuteen -arvo (VTL) määräytyy sen mukaan, kuinka vakavasti haavoittuvuuden sisältävän ohjelmiston säilyttämä tieto vaarantuu, kun haavoittuvuutta hyväksikäytetään kyberhyökkäyksessä. Arvot ovat seuraavat [120, s.10]:

- Ei vaikutusta (E): Ei vaikutusta tiedon luottamuksellisuuteen
- Osittainen (O): Jonkinasteinen tiedon luottamuksellisuuden menetys. Hyökkääjällä on osittainen pääsy rajoitettuun tietoon.
- Korkea (K): tiedon luottamuksellisuus menetetään täysin. Hyökkääjä saa kohdeohjelman kaiken tiedon haltuunsa.

Vaikutus tiedon eheyteen -arvo (VTE) määräytyy sen mukaan, kuinka vakavasti haavoittuvuuden sisältävän ohjelmiston tiedon eheys vaarantuu, kun haavoittuvuutta hyväksikäytetään kyberhyökkäyksessä. Arvot ovat seuraavat [120, s.10]:

- Ei vaikutusta (E): ei vaikutusta tiedon eheyteen
- Osittainen (O): tiedon manipulointi on mahdollista, mutta hyökkääjällä ei ole kontrollia muokkauksesta aiheutuviin seuraamuksiin
- Korkea (K): tiedon eheys tai sen suojaus menetetään täysin. Hyökkääjä kykenee manipuloimaan kohdeohjelmassa säilytettävää tietoa.

Vaikutus tiedon saatavuuteen -arvo (VTS) määräytyy sen mukaan, kuinka paljon haavoittuvuuden hyväksikäytöllä on vaikutusta kohdeohjelman tiedon ja palveluiden saatavuuteen. Arvot ovat seuraavat [120, s.10–11]:

- Ei vaikutusta (E): ei vaikutusta tiedon saatavuuteen
- Osittainen (O): kohdejärjestelmän suorituskyvyn ja/tai resurssien saatavuudessa on katkoksia. Kyberhyökkääjän kyetessä hyödyntämään haavoittuvuutta uudelleen, ei hän kykene täysin estämään kohdeohjelman toimintaa.

- Korkea (K): tiedon saatavuus menetetään täysin ja kyberhyökkääjä kykenee estämään käyttäjän pääsyn kohdeohjelman resursseihin. Vaikutusta voidaan ylläpitää tai se voi olla säilyvä kyberhyökkäyksen jälkeenkin.

Haavoittuvuuden käytettävyys -arvo (HK) määräytyy sen mukaan, kuinka todennäköisesti haavoittuvuutta kyetään hyväksikäyttämään kyberhyökkäyksessä. HK-arvo kasvaa hyväksikäyttöön soveltuvien menetelmien kypsyyden kasvaessa. Arvot ovat seuraavat [120, s.11]:

- Ei tiedossa (ET): haittaohjelmakoodia ei vielä saatavilla tai hyväksikäyttö mahdollisuus on vai teoreettista
- Konseptitodistus (Proof-of-Concept) (K): konseptin todistava haittaohjelma tai skripti on saatavilla, mutta hyväksikäyttömenetelmä ei ole vielä käytännöllinen suurimpaan osaan järjestelmistä. Hyväksikäyttömenetelmää joudutaan muokkaamaan kohdeympäristön mukaan.
- Toimiva (T): toimivaa hyväksikäyttömenetelmä on yleisesti saatavilla ja se toimii suurimpaan osaan kohdejärjestelmistä
- Korkea (K): toimivaa autonominen hyväksikäyttömenetelmä on olemassa tai sellaista ei tarvita ja hyväksikäyttömenetelmän yksityistietoja on yleisesti saatavilla.
- Ei määritetty (EM): haavoittuvuuden käytöstä ei ole riittävästi määritettyä tietoa, joten sillä ei ole vaikutusta kokonaisvaikuttavuuden arviontiin.

Haavoittuvuuden paikattavuus -arvo (HP) määräytyy sen mukaan, onko haavoittuvuuteen korjausta ja minkä tasoinen korjaus on. Tyypillisesti haavoittuvuuteen ei ole pysyvää korjausta, kun se julkaistaan, joten sitä paikataan pikakorjauksilla (hotfix) ja toiminnollisuuksien poiskytkennällä. Mitä tehottomampia korjaukset ovat, sitä suuremmat pisteet haavoittuvuus saa. Arvot ovat seuraavat [120, s.12]:

- Ei tiedossa (ET): haavoittuvuuden paikattavuudesta ei ole tietoa, joten tällä arvolla ei ole vaikutusta kokonaisvaikuttavuuden arviointiin
- Pysyvä valmistajan korjaus (LVK): haavoittuvuuden paikkaamiseen on saatavilla pysyvä valmistajan korjaus
- Väliaikainen valmistajan korjaus (VVK): haavoittuvuuden paikkaamiseen on saatavilla väliaikainen valmistajan korjaus
- Epävirallinen korjaus (EK): haavoittuvuuden paikkaamiseen ei ole olemassa valmistajan korjausta
- Ei saatavilla (ES): haavoittuvuuteen ei ole olemassa paikkausmenetelmää

Haavoittuvuuden kiistattomuus -arvo (HKM) mittaa haavoittuvuuden ja siitä julkaistujen tietojen kiistattomuutta. Joskus pelkästään haavoittuvuuden olemassaolo julkaistaan ilman tarkempaa tietoa siitä. Mitä varmistetumpi haavoittuvuuden olemassaolo on valmistajan tai muun luotettavan lähteen toimesta, sitä suuremmat pisteet se saa. Arvot ovat seuraavat [120, s.13]:

- Ei tiedossa (ET): haavoittuvuuden olemassaolosta ei ole riittävästi tietoa saatavilla
- Tuntematon (T): Vaikutuksista on raportteja, jotka viittaavat haavoittuvuuden olemassaoloon, mutta tarkka aiheuttaja on tuntematon tai raporttien arviot aiheuttajasta eroavat tosistaan
- Merkittävä (M): merkittävää tietoa haavoittuvuudesta on julkaistu, mutta tutkijoilla ei ole täyttä varmuutta alkuperäisestä aiheuttajasta tai pääsyä lähdekoodiin varmistaakseen täysin sen toimivuutta
- Vahvistettu (V): haavoittuvuudesta on julkaistu yksityiskohtaisia raportteja tai toimivan kopion tekeminen siitä on mahdollista

Ympäristömittariston avulla voidaan ottaa toimintaympäristön vaikutus pisteytyksessä huomioon. Arvoja ovat tietoturvallisuuden vaatimustaso (TV) sekä modifioidut perusmittarit. Tietoturvallisuuden vaatimuksessa arvioidaan luottamuksellisuuden, eheyden ja saatavuuden kohdeorganisaatiokohtaista painokerrointa vaikuttavuusmittariston arvojen pisteytykseen. Vaikuttavuusmittarien arvojen ja tietoturvallisuuden vaatimustason painokertoimen avulla lasketaan modifioitu tulo tietoturvan osa-alueille. Mitä suurempi osa-alueen tärkeys kohdeorganisaatiolle on sitä suuremman pisteen se saa. Arvot ovat seuraavat [120, s.14]

- Ei tiedossa (ET): kohdeorganisaation tietoturvallisuuden osa-aluepainotusta haavoittuvuuden pisteytykseen ei kyetä määrittämään
- Matala (M): tiedon luottamuksellisuuden / eheyden / saatavuuden menetyksellä on todennäköisesti vähäinen haittavaikutus kohdeorganisaatiolle
- Keskitaso (KT): tiedon luottamuksellisuuden / eheyden / saatavuuden menetyksellä on todennäköisesti merkittävä haittavaikutus kohdeorganisaatiolle
- Korkea (K): tiedon luottamuksellisuuden / eheyden / saatavuuden menetyksellä on todennäköisesti kriittinen haittavaikutus kohdeorganisaatiolle

Modifioiduilla perusmittareilla voidaan toimintaympäristöanalyysin vaikutuksesta korvata perusmittarin arvoja. Modifioitujen perusmittarien (hyödynnettävyys ja vaikutus) eteen tulee näissä arvoissa ”modifioitu”. [120, s.15]

5.1.2 Geneerisen haavoittuvuuden vakavuusanalyysi

Haavoittuvuuden vakavuusanalyysilla voidaan analysoida haavoittuvuuden vaikutusta taktisten päätelaitteiden käyttöjärjestelmiin. Taktisiin päätelaitteisiin asti ulottuvan haavoittuvuuden hyväksikäyttö olisi hyökkääjälle haastavaa. Onnistunut hyökkäys vaatisi mahdollisesti kattavaa kohdetiedustelua sekä varsinaisen hyökkäyksen ulottamista monien suojattujen verkkokerroksien ylitse saastuttamalla reitittimiä, ja ohittamalla palomuureja päästääkseen päätelaitetasalle. Todennäköisempää onnistumisen kannalta onkin, että hyökkäys vaatisi fyysisen pääsyn jollekin kohdelaitteelle, josta kyberhyökkäystä kyetään levittämään. [122, s.55]

Taktiset päätelaitteet ovat johto-organisaatioiden toiminnan kannalta tärkeimpiä suunnittelutyökaluja, joihin suunnatun kyberhyökkäyksen tavoitteena voi olla saada, muuttaa tai estää operaatioiden toteuttamiseen tarvittavaa tietoa vaikuttamalla taistelunjohto- ja tietojärjestelmiin. [122, s.55]

Vastatakseen päätutkimuskysymykseen: ”Miten Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvallisuus toteutuu taktisissa päätelaitteissa” joudutaan yleisesti Windows 10- ja Debian 10 -käyttöjärjestelmien lisäksi analysoimaan, millaisia haavoittuvuuksia erityisesti taktisten päätelaitteiden Windows 10- ja Debian 10 -käyttöjärjestelmissä on. Tämän takia tutkimusta varten luodaan erityisesti taktiseen päätelaitteen käyttöjärjestelmään kohdistuva geneerinen haavoittuvuus, jonka hyväksikäyttö avaa takaoven hyökkääjälle. Takaovea hyväksikäyttämällä hyökkääjä saa osittaisen kontrollin sisäverkon päätelaitteisiin ja kykenee jatkamaan kyberhyökkäystä asentamalla järjestelmään tiedustelu-haittaohjelman.

Haavoittuvuuden arvot ovat seuraavat:

Hyökkäysvektorin -arvoksi (HV) määritetään Rinnakkainen (R), koska taktiset päätelaitteet ovat verkon kauimmassa reunassa, jolloin niihin asti pääseminen julkisesta ulkoverkosta käsin on todella epätodennäköistä. Riittämättömän kybervalvonnan puuttuessa sisäverkon sisällä hyökkäyksen jatkaminen ja takaoven asentaminen muihin taktisisiin päätelaitteisiin voisi olla mahdollista. [122, s.55]

Hyökkäyksen edellytykset -arvoksi (HE) määritetään Korkea (K), koska kovennettujen taktisten päätelaitteiden hyväksikäyttö vaatisi todennäköisesti hyökkääjältä edistynyttä kohdejärjestelmän tiedustelua sekä tekijöitä, joita se ei kykenisi kontrolloimaan ilman huomattavaa resurssien käyttöä. [122, s.55]

Vaaditut käyttöoikeudet -arvoksi (VK) määritetään Korkea (K), koska kohdejärjestelmän hyökkäyspinta-alaa on pienennetty kovennuksilla, jolloin onnistunut haavoittuvuuden hyväksikäyttö sekä hyökkäyksen jatkaminen haittaohjelman asentamisella, edellyttäisi järjestelmänvalvojatason oikeuksia. [122, s.58]

Käyttäjän vuorovaikutus -arvoksi (KV) määritetään Ei tarvetta (E), koska jos se vaatisi toimenpiteitä käyttäjältä eikä olisi itsenäinen, se ei kykenisi toimimaan huomaamattomasti ja havaittaisiin ennen leviämistä. [122, s.55]

Ulottuvuus -arvoksi (U) määritetään Jatkaminen (J), koska kyseisessä kohdejärjestelmän sisäverkossa päätelaitteiden käyttöjärjestelmät ovat versioltaan identtisiä, joten hyökkäystä voitaisiin jatkaa kohteena olevan verkon muihin taktisiin päätelaitteisiin. Lisäksi haitatakseen tai tiedustellakseen kohdeorganisaation toimintaa mahdollisimman laajasti, hyökkäyksen suunnittelija haluaisi vaikuttaa mahdollisimman suureen määrään päätelaitteita, jolloin haavoittuvuuden hyväksikäytöllä pitää kyetä jatkamaan hyökkäystä myös muihin sisäverkon taktisiin päätelaitteisiin. [122, s.58]

Vaikutukset tiedon luottamuksellisuuteen, eheyteen ja saatavuuteen ovat kaikki Korkeat (K). Hyökkääjä saa takaovea käyttäen kohdejärjestelmästä kaiken haluamansa tiedon haltuunsa. Takaoven avulla hän kykenee manipuloimaan kohdejärjestelmän tietoa ja mahdollisesti levittämään vääristynyttä tietoa verkon muihin vastaaviin samoja ohjelmia käyttäviin päätelaitteisiin. Käyttäjän pääsy kohdejärjestelmän tietoihin voidaan estää, mutta saatavuuden täydellinen menettäminen toimisi hälytyskellona vastatoimenpiteille, joten jatkaakseen hyökkäystä mahdollisimman pitkään huomaamattomasti, tämän kaltaista vaikutusta ei ennen tavoitteeseen pääsyä todennäköisesti haluttaisi. [122, s.59]

Haavoittuvuuden käytettävyys -arvoksi (HK) määritetään Korkea (K), koska toimiakseen autonomisesti kohdejärjestelmässä ja jatkaakseen haavoittuvuuden hyväksikäyttöä muihin verkon päätelaitteisiin, sen täytyy olla teknisesti edistynyt. [122, s.59]

Haavoittuvuuden paikattavuus -arvoksi (HP) määritetään Ei tiedossa (ET), koska haavoittuvuuden olemassa olosta ei ole riittävästi tietoa saatavilla ja hyökkäys kyettäisiin mahdollisesti toteuttamaan käyttäen monia eri haavoittuvuuksia. [122, s.59]

Haavoittuvuuden kiistattomuus -arvoksi (HKM) määritetään Tuntematon (T), koska ollessaan uusi haavoittuvuus, siitä ei ole vielä olemassa kiistatonta tietoa. Se ei myöskään toimi aina kaavamaisesti, joten kerättyjen ilmoitusten sisältö saattaa vaihdella. [122, s.60]

Tietoturvallisuusvaatimukset taktisille päätelaitteille eivät ole toiminnan kannalta kaikista kriittisimpiä, koska kaikista kriittisintä tietoa varmuuskopioidaan muualle sekä toimittaminen vastaanottajalle toteutetaan usein perinteisillä menetelmillä sähköisten sijasta. Tietoturvallisuuden vaatimustason (TV) arvot haavoittuvuudelle ovat seuraavat [122, s.57]:

- Tiedon luottamuksellisuus saa arvon Keskitaso (KT), koska esimerkiksi johtamis- ja tilannetieto eivät saa vuotaa ennen toteutettavaa operaatiota. Vuotava tieto ei kuitenkaan ole kaikista kriittisintä, koska jaettavaa tietoa rajataan käyttäjien tarpeiden mukaan.
- Tiedon eheys saa arvon Keskitaso (KT), koska aiemmin mainittujen tietojen muokkaaminen lisäisi riskiä toiminnan epäonnistumiselle, mutta tietoa varmennetaan perinteisillä menetelmillä.
- Tiedon saatavuus saa arvon Keskitaso (KT), koska haavoittuvuuden hyväksikäytöllä on vaikutusta esimerkiksi johtamis- ja tilannetiedon välittämiseen, mutta kohdeorganisaatio voi turvautua myös varamenetelmiin.

Mittari	Lyhenne	Arvot				
Hyökkäysvektori	HV	Verkko (V)	**Rinnakkainen (R)**	Paikallinen (P)	Fyysinen (F)	
Hyökkäyksen edellytykset	HE	Matala (M)	**Korkea (K)**			
Vaaditut käyttöoikeudet	VK	Ei oikeuksia (E)	Matala (M)	**Korkea (K)**		
Käyttäjän vuorovaikutus	KV	**Ei tarvetta (E)**	Tarvitaan (T)			
Ulottuvuus	U	Ei ulottuvuutta (E)	**Jatkaminen (J)**			
Vaikutus tiedon luottamuksellisuuteen	VTL	Ei vaikutusta (E)	Osittainen (O)	**Korkea (K)**		
Vaikutus tiedon eheyteen	VTE	Ei vaikutusta (E)	Osittainen (O)	**Korkea (K)**		
Vaikutus tiedon saatavuuteen	VTS	Ei vaikutusta (E)	Osittainen (O)	**Korkea (K)**		
Haavoittuvuuden käytettävyys	HK	Ei tiedossa (E)	Konseptitodistus (K)	Toimiva (T)	**Korkea (K)**	Ei määritetty (EM)
Haavoittuvuuden paikattavuus	HP	**Ei tiedossa (E)**	Pysyvä valmistajan korjaus (PVK)	Väliaikainen valmistajan korjaus (VVK)	Epävirallinen korjaus (EK)	Ei saatavilla (ES)
Haavoittuvuuden kiistattomuus	HKM	Ei tiedossa (E)	**Tuntematon (T)**	Merkittävä (M)	Vahvistettu (V)	
Luottamuksellisuuden vaatimustaso	LV	Ei tiedossa (E)	Matala (M)	**Keskitaso (KT)**	Korkea (K)	
Eheyden vaatimustaso	EV	Ei tiedossa (E)	Matala (M)	**Keskitaso (KT)**	Korkea (K)	
Saatavuuden vaatimustaso	SV	Ei tiedossa (E)	Matala (M)	**Keskitaso (KT)**	Korkea (K)	

Taulukko 6 Geneerisen haavoittuvuuden CVSS-arvot

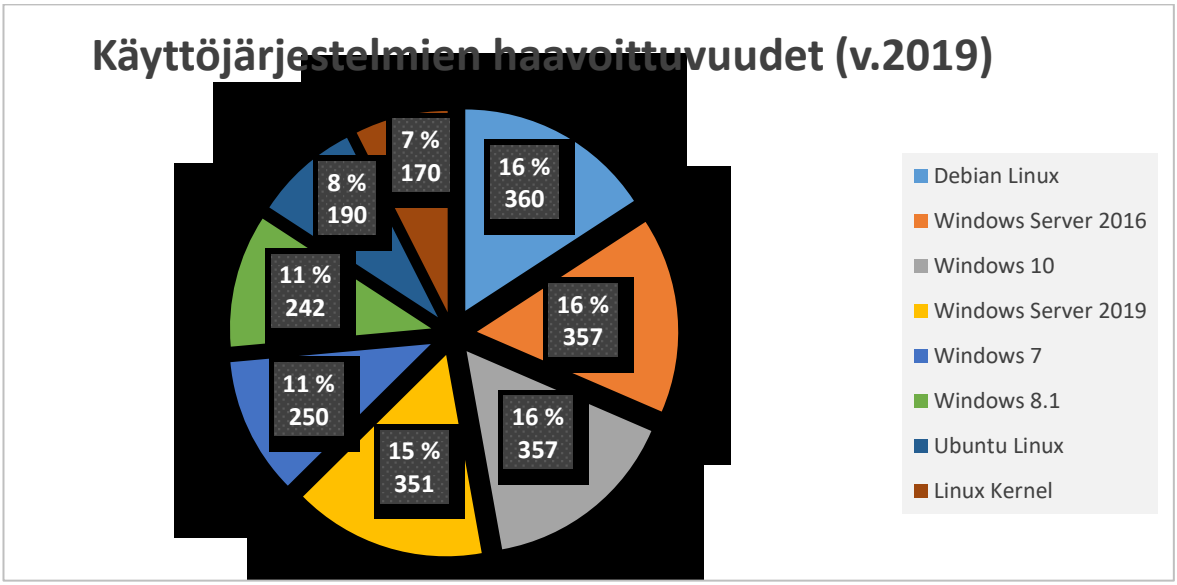
Analyysin tulokset ovat seuraavat:

- Perusmittaristo: 7.6 (korkea)
- Ajallinen mittaristo: 7.0 (korkea)
- Ympäristömittaristo: 7.1 (korkea)

Geneerisen haavoittuvuuden CVSS-piste on 7.2 ja vakavuusluokaltaan se on korkea.

5.2. Windows 10 ja Debian -käyttöjärjestelmien haavoittuvuudet

Haavoittuvuuksien määrää ja vakavuutta tarkastelemalla voidaan osaltaan tehdä päätelmiä käyttöjärjestelmien tietoturvallisuudesta. Seuraavassa diagrammissa on esillä vuoden 2019 haavoittuvuudet. Kaikista Debian -käyttöjärjestelmistä on vuonna 2019 löytynyt hieman enemmän haavoittuvuuksia muihin vertailtaviin käyttöjärjestelmiin verrattuna. Diagrammia analysoidessa on kuitenkin otettava huomioon helposti esiintyvä virhe siitä, että sivusto listaa tähän lukuun kaikkien eri Debian-versioiden haavoittuvuudet, eikä pelkästään esimerkiksi Debian 10 -käyttöjärjestelmää. Windows 10 -käyttöjärjestelmän alle ei tässä diagrammissa ole listattu muiden Windows -jakeluversioiden haavoittuvuuksia. Alaluvun myöhemmin esiintyvissä tilastoissa tämä on huomioitu.

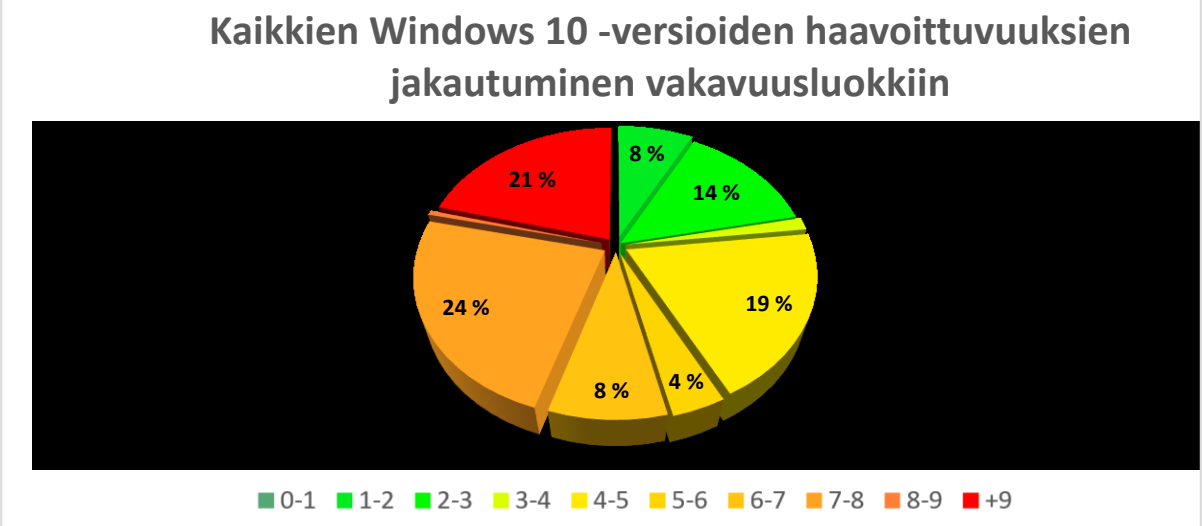


Kuva 19 Käyttöjärjestelmien haavoittuvuudet vuonna 2019 [116]

Vertailusta saadaan tarkempi vertailemalla eri versioita käyttöjärjestelmistä. Seuraavalla sivulla vertaillaan Windows 10 1903 -pääpäivityksen haavoittuvuuksien tyyppiä ja määrä suhteessa kaikkiin Windows 10-versiohin, sekä Debian 10 -version haavoittuvuuksien tyyppiä ja määrää suhteessa kaikkiin Debian-versioihin.

Painotettu keskiarvo on laskettu kaavalla [116]:

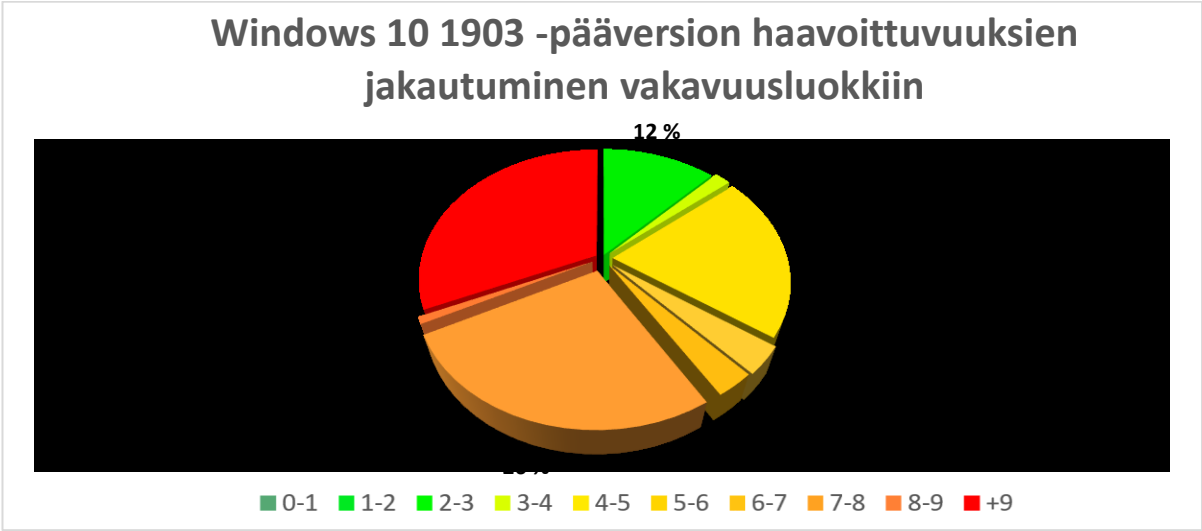
(Summa (vakavuusluokan yläarvo eli esim. 1-2 -> 2 * vakavuusluokan haavoittuvuuksien määrä)) / haavoittuvuuksien summa



Kuva 20 Kaikkien Windows 10 -versioiden haavoittuvuuksien jakautuminen vakavuusluokkiin (26.3.2020) [116]

0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	+9
0	85	156	16	215	43	94	264	6	232

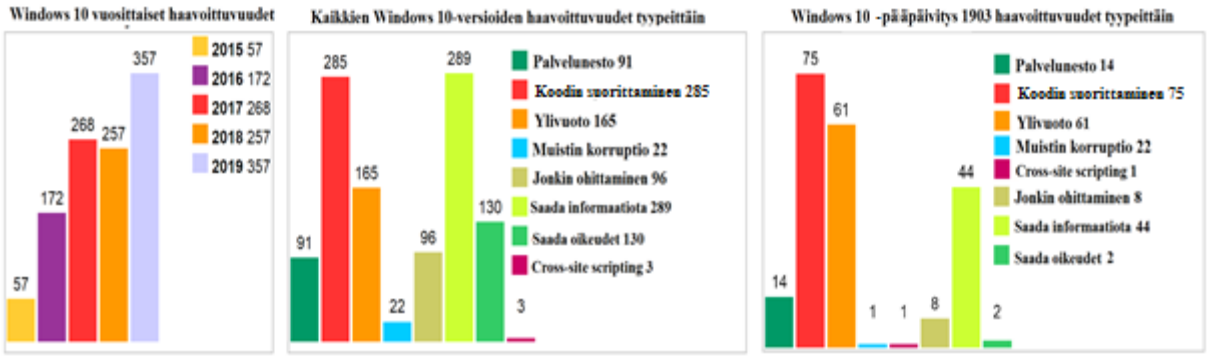
Painotettu keskiarvo 6.3



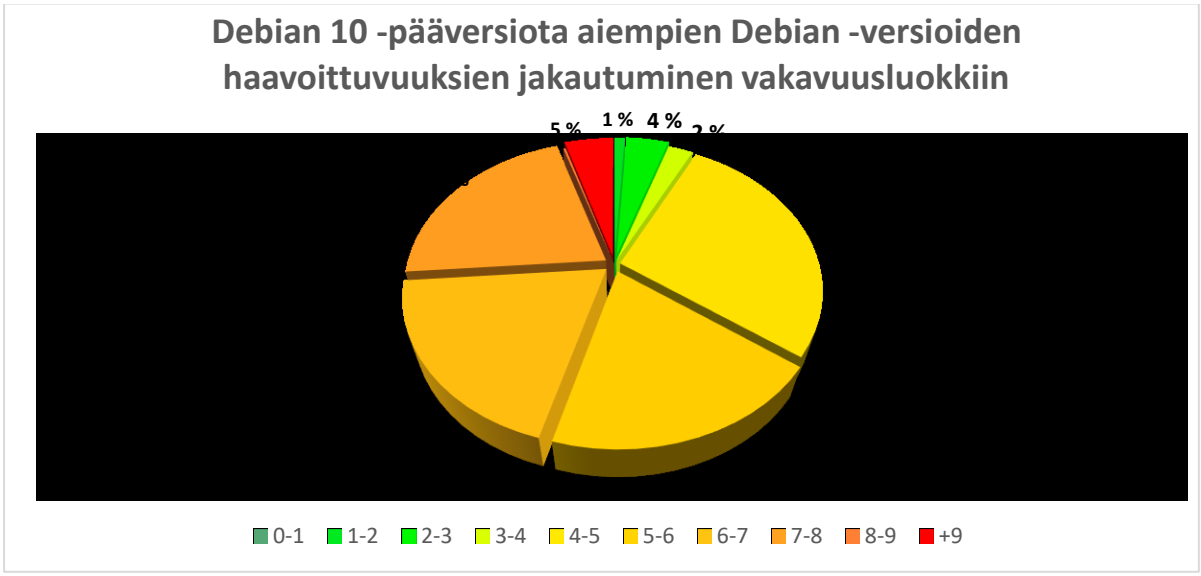
Kuva 21 Windows 10 1903 -pääversion haavoittuvuuksien jakautuminen vakavuusluokkiin (26.3.2020) [116]

0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	+9
		29	4	48	8	7	61	2	72

Painotettu keskiarvo: 7.1



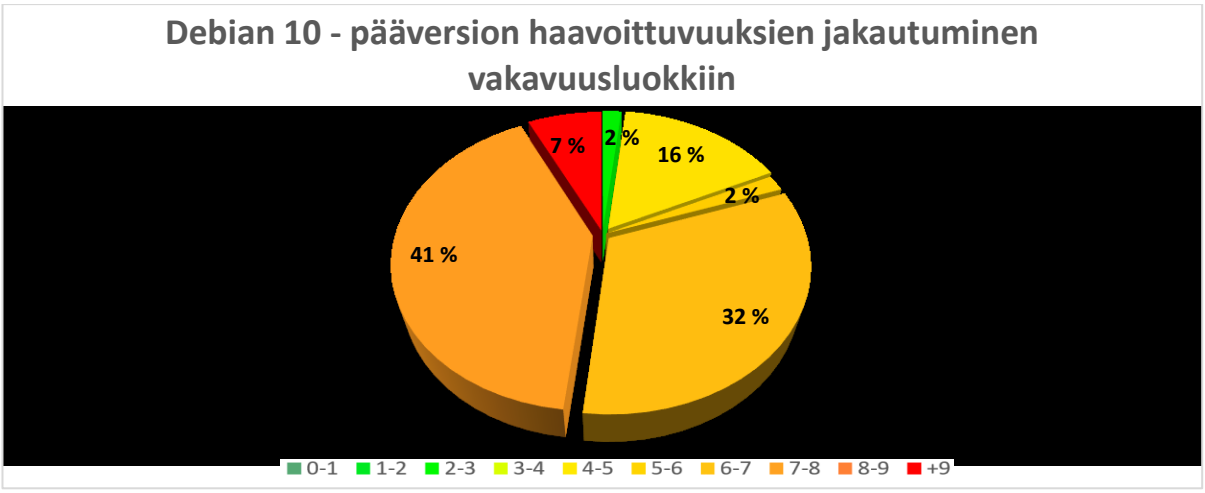
Kuva 22 Windows vuosittaiset ja sen eri versioiden haavoittuvuudet (26.3.2020) [116]



Kuva 23 Debian 10 -pääversiota aiempien Debian -versioiden haavoittuvuuksien jakautuminen vakavuus-luokkiin (26.3.2020) [116]

0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	+9
	31	109	67	852	605	569	625	7	136

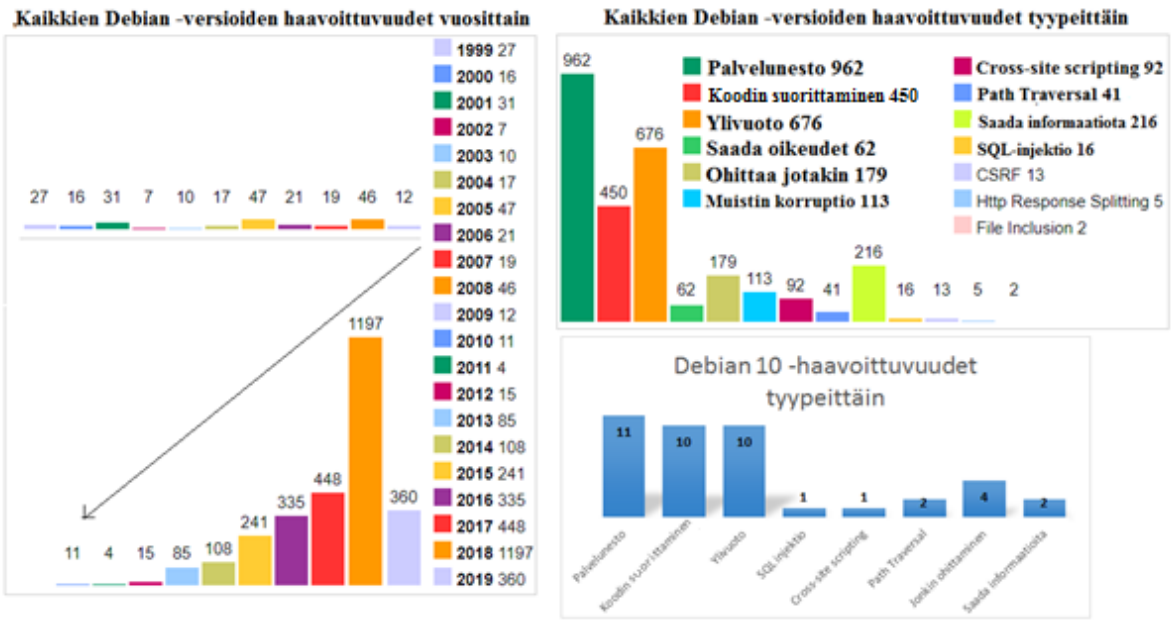
Painotettu keskiarvo: 6.5



Kuva 24 Debian 10 -pääversion haavoittuvuuksien jakautuminen vakavuusluokkiin (26.3.2020) [116]

0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9+
0	0	1	0	9	1	18	23	0	4

Painotettu keksiarvo 7.2



Kuva 25 Debian -versioiden haavoittuvuudet (26.3.2020) [116]

Tilastoista voidaan nähdä, että Windows 10 -käyttöjärjestelmästä on löytynyt määrällisesti enemmän vakavia haavoittuvuuksia kuin Debian 10 -käyttöjärjestelmästä. Käyttöjärjestelmien painotetut keskiarvot ovat lähestulkoon samoja. Molempien käyttöjärjestelmien painotetut keskiarvot ovat kuitenkin nousseet merkittävästi koko ajan keskiarvosta, mikä tarkoittaa sitä, että löydetyt haavoittuvuudet ovat yhä vakavampia. palvelunesto, koodin suorittaminen, ylivuoto ja informaation vuodon mahdollistavat haavoittuvuudet ovat molemmissa käyttöjärjestelmissä tilastojen mukaan yleisimpien haavoittuvuuksien joukossa.

6. JOHTOPÄÄTÖKSET

6.1. Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturva

6.1.1 Käyttöjärjestelmien tietoturvallisuuden peruspilarit

Käyttöjärjestelmän tietoturvallisuuden peruspilareista puhuttaessa on olennaista määrittää, mikä liittyy käyttöjärjestelmiin. Olennaista on keskittyä tietoturvaominaisuuksiin, jotka sisältyvät käyttöjärjestelmään ja missä käyttöjärjestelmä on merkittävässä roolissa tietoturvakäytänteiden vahvistamisessa tai niiden heikentämisessä sekä hyökkäysvektoreihin, joissa käyttöjärjestelmä on hyökkäyksen kohteena.

Usein nämä hyökkäysmenetelmät eivät ole erityisen edistyneitä, vaan heikko salasana, yksinkertainen huijaus tai haavoittuvuuden hyväksikäyttö riittää hyökkääjälle. On myös monia esimerkkejä, joissa järjestelmien tietoturva ei riitä, kun käyttäjä kadottaa luottamuksellista tietoa sisältävän muistitikun tai kiintolevyä ei tyhjennetä oikeaoppisesti.

On mahdollista tehdä hyvin tietoturvallinen ja vähän virheitä sisältävä käyttöjärjestelmä. Tällöin kuitenkin jouduttaisiin kompromissina tekemään hyvin yksinkertainen, käytettävyydeltään rajattu ja niukasti toimintoja sisältävä järjestelmä. Pelkkä käyttöjärjestelmän suunnittelu ei riittäisi, vaan myös käytettävä laitteisto täytyisi suunnitella ja valmistaa täysin tietoturvavaatimusten pohjalta. Kaiken kompleksisuuden hallinta täysin tietoturvallisen järjestelmän rakentamiseksi olisikin äärimmäisen vaikeaa, ellei jopa mahdotonta.

Tietoturvallisuudesta suurin osa toteutetaan käyttäjien turvallisilla toimintatavoilla. Onkin arvioitu, että tietoturvasta vain 20 prosenttia kyetään toteuttamaan tekniikalla (muun muassa: käyttöjärjestelmävalinnat, palomuurit, virustorjunta ja pääsynvalvonta) ja 80 prosenttia on kaikkea muuta, mitä tietoturvaan liittyy, kuten käyttäjien ohjeistamista sekä sitä, kenellä on pääsy mihinkin tietoon. Päätelaitetasolla tietoturvallisuus koostuu käyttöjärjestelmien tietoturvaominaisuuksista ja turvaohjelmistoista.

Käyttöjärjestelmän haavoittuvuus saattaa syntyä ohjelmointivirheistä, yhteensopivuusongelmista sekä konfiguraatiovirheistä. Yksinkertaisimmat käyttöjärjestelmiin kohdistuvat hyökkäykset käyttävät hyväkseen järjestelmien ja ohjelmien haavoittuvuuksia, jotka mahdollistavat esimerkiksi haittaohjelman suorittamisen.

Haittaohjelmia ovat esimerkiksi virukset, madot, troijalaiset sekä näiden yhdistelmät. Ne tekevät käyttäjän tietämättä jotain, mikä voidaan olettaa haitalliseksi käyttäjälle tai järjestelmälle, kuten vuotamalla tietoa tai aiheuttamalla muuten ei-toivottuja tapahtumia. Haittaohjelma voi esimerkiksi avata takaoven eli keinon ohittaa järjestelmän normaalin todentamisen tai salauksen tunkeutujalle. Takaovi voi syntyä myös esimerkiksi tahallisesti heikennetystä salausalgoritmin parametreista (sisäpiirihyökkäys).

Tunkeutumisessa voidaan käyttää hyväksi myös sovelluksien puskurin ylivuotovirhettä sekä heikosti suojattuja, tai paljastuneita kirjautumistietoja. Käyttöjärjestelmän tietoturvallisuuden kannalta kriittisimmät hyökkäykset kohdistuvat järjestelmäsovelluksiin sekä käyttöjärjestelmän ydintason ohjelmiin.

Tietoturvasertifikaatit (CC ja FIPS140-2) helpottavat tietoturvallisuuden auditointia. Sertifikaattien johdosta käyttöjärjestelmän kaikkia tietoturvaominaisuuksia ei välttämättä tarvitse testata ja evaluoida niin hyvin kuin käyttöjärjestelmää, jossa niitä ei ole. Tällä voi olla suurikin vaikutus käyttöjärjestelmähankkeen elinjaksokustannuksiin varsinkin suunnittelu- ja kehittämisvaiheessa.

Tietoturvasertifikaatteja on monilla käyttöjärjestelmillä. Windows 1903-pääpäivitys on muun muassa evaluoitu ja se on saanut CC-tietoturvasertifikaatin (kuten myös Ubuntu), mutta Debian 10 -käyttöjärjestelmä ei ole. Perusteluna sertifioimattomuudelle on ollut, että ne maksavat liikaa. Tähän Debian-projektilla ei ole ollut rahaa.

Käyttöjärjestelmän kompleksisuuden ja koon lisäksi tietoturvaan vaikuttavia tekijöitä ovat esimerkiksi muutokset, ja saatavilla olevat resurssit. Laajempi koodin katselmointi, tai hitaammat muutokset ja pidempi aika katselmoida koodia vähentävät riskiä virheille.

Ylläpitääkseen tietoturvallisuutta käyttöjärjestelmissä, on tärkeää myös ymmärtää, miten niitä vastaan voidaan hyökätä. Mutta vielä tärkeämpää on ymmärtää, miten ne voidaan suojata esimerkiksi tiedon luottamuksellisuuden menettämiseltä.

Yksittäiset laitteet, teot tai säännöt eivät yksinään riitä suojaamaan järjestelmää kyberhyökkäyksiltä tai virheiltä. Välttääkseen tätä, modernit käyttöjärjestelmät pyritään tekemään tietoturvalliseksi monitasoisella suojauksella (defense in depth). Tavoitteena on, että järjestelmän suojaus sisältäisi monta kerrosta, jolloin yhden pettäessä on hyökkääjällä vielä monta edessä. Suojauksen tärkeimpiä kerroksia ovat käyttöjärjestelmän ytimen turvallisuus, käynnistyksen suojaaminen, virtualisointi, haittaohjelmien torjunta, käyttäjän tunnistaminen ja todentaminen, käyttöoikeuksien hallinta, palomuuri sekä ohjelmien ja kiintolevyn salaus.

Käyttöjärjestelmien tietoturvallisuuden peruspilarit ovat:

- monitasoinen suojaus
- käyttöjärjestelmien ja siihen asennettavien ohjelmien saamat tietoturvasertifikaatit
- tietoturvapäivitykset
- digitaaliset varmenteet

6.1.2 Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvaominaisuudet

Käyttöjärjestelmän ytimet voidaan jaotella monoliittisiin ytimiin, joissa kaikki toiminnot ovat samassa muistiavaruudessa (esimerkkinä Debian) sekä mikroytimellisiin (esimerkkinä Google Fuchsia), joissa tietoturvallisemmin käyttöjärjestelmämoduulit on eristetty ytimestä ja toisistaan jakamalla ne eri muistiavaruuteen. Eristämisen johdosta järjestelmän nopeus hidastuu, verrattuna siihen, että kaikki moduulit olisivat samassa muistiavaruudessa. Windows -käyttöjärjestelmien ydin voidaan katsoa kuuluvan näiden mono- ja mikroytimillisten tyyppien hybridiksi.

Windows 10 -käyttöjärjestelmässä muistiavaruus on jaettu käyttäjä- ja ydintiloihin sekä virtualisoinnilla suojattuun ydintilaan (VBS). Tietoturvallisuudeltaan tämä hybridivaihtoehto on huonompi kuin käyttöjärjestelmän moduulit toisistaan täysin eristävä mikroytimillinen rakenne, mutta parempi kuin, että kaikki käyttöjärjestelmän moduulit ovat samassa muistiavaruudessa (monoliittinen ydin kuten esimerkiksi Debianin Linux-ydin).

Windows 10 -käyttöjärjestelmän toimet tietoturvan ylläpitoon:

- hyökkäyspinta-alaa pienennetään: virtualisoinnilla ja siihen pohjautuvilla suojauksilla (VBS), sekä organisaation omien ryhmäkäytäntöjen avulla (*Code Integrity Policy*) konfiguroitavilla laitesuojauksella (*Device Guard*) ja prosessisuojauksilla (PPL)

- muistia suojataan: *hallintavirran suojauksella* (CFG), *Data-alueen suorituksen esto* -toiminnolla (DEP), *Osoiteavaruuden satunnaistamistekniikalla* (ASLR) sekä *Struktuuroidun poikkeuskäsittelyn ylikirjoitussuojalla* (SEHOP)
- uhkaa vastustetaan: *Windows Defender palomuurilla*, jolla estetään pakettisuodatuksen avulla liikennettä, joka ei ole sallittua.
- identiteettiä suojataan: Käyttäjän tunnistamiseen ja todentamiseen (*Credential Guard*) liittyvillä toiminnoilla sekä pääsynhallinnalla (UAC)
- tietoa suojataan: käyttämällä Microsoftin omaa ReFS-tiedostojärjestelmää, EFS- tai Bitlocker -laitesalausta, digitaalisilla varmenteilla sekä WIP-tietoturvatoiminnolla
- tunkeutuminen havaitaan ja vastatoimet tehdään käyttäen käynnistyssuojauksia (*Secure-*, *Trusted-* ja *Measured Boot* sekä ELAM -tietoturvatoimintoa) sekä *Windows Defender* -haittaohjelmien torjuntaohjelmalla.

Debian 10 -käyttöjärjestelmän toimet tietoturvan ylläpitoon:

- hyökkäyspinta-alaa pienennetään: rajoittamalla ohjelmistorajapintoja käyttäjätasolla tehden ydintilan ohjelmistorajapintojen virheellisestä käytöstä vaikeampaa, pienentämällä kirjoitettavan ydintilan muistia, estämällä ytimen koodin ja kirjoitussuojatun datan ylikirjoittaminen (*Config_Strict_Kernel_* ja -*Module_RWX*), estämällä tiedostojen suorittaminen ilman, että ne erikseen muutetaan käyttäjän toimesta suoritettavaksi, erottamalla ytimen- ja käyttäjän muistiavaruus toisistaan sekä rajoittamalla prosessien järjestelmäkutsujen pääsyä (seccomp) ytimen käsittelyyn sekä käyttäjän kykyä ladata ydinmoduuleita.
- muistia suojataan: *Osoiteavaruuden satunnaistamistekniikalla* (ASLR) ja muistin virtualisoinnilla, puskurin ylivuotoa torjuvilla SSP-laajennuksella sekä FORTIFY_SOURCE -kääntäjämakrolla (GCC ja GLIBC)
- uhkaa vastustetaan: Linux -jakelupakettien palomuurilla, jolla kyetään rajoittamaan Netfilter-rajapinnan kautta kulkevaa tietoliikennettä rajoittamalla auki olevia portteja.
- identiteettiä suojataan: Linux Securite Modules -käyttöliittymällä (LSM), jonka päälle rakennetuilla toiminnoilla, kuten AppArmor:lla, mahdollistetaan pakollisen pääsynhallinnan (MAC) käyttö ja turvallisuusprofiilien luonti ohjelmille.
- tiedon luottamuksellisuutta suojataan LUKS laitteistoriippumattomalla standardilla sekä eheyttä ”CHECK_DATA_CORRUPTION” -toiminnollisuudella ja digitaalisesti varmennetuilla asennuspaketeilla.
- tunkeutuminen havaitaan ja vastatoimet tehdään käyttäen avoimen lähdekoodin ClamAV-haittaohjelmien torjuntaohjelmaa.

Molemmissa käyttöjärjestelmissä käytetään muun muassa virtualisointia ja muita hyökkäyspinta-alaa vähentäviä toimintoja kuten muistin suojauksia. Windows 10 -käyttöjärjestelmässä suojauksia on määrällisesti hieman enemmän, mutta Debian 10 -käyttöjärjestelmässä on monia hyödyllisiä ominaisuuksia kuten se, että tiedostot täytyy erikseen tehdä käyttäjän toimesta päätteellä suoritettavaksi. Kyseinen ominaisuus rajoittaa haittaohjelmien toimintamahdollisuuksia järjestelmässä huomattavasti.

Windows 10 -käyttöjärjestelmässä tämän kaltaista suojausta ei ole, ja esimerkiksi haittaohjelmat voivat suorittaa käyttäjätasolla itsensä ilman käyttäjän toimia tai havaintoa. Hyökkäyspinta-alan ja muistin suojauksen osalta Windows 10- ja Debian 10 -käyttöjärjestelmät ovat saman tasoiset.

Uhkan vastustamiseen Windows 10- ja Debian 10 -käyttöjärjestelmissä ei ole muita tietoturvatoimintoja kuin palomuuri. Debian 10 -käyttöjärjestelmän palomuuri sallii perusasetuksilla kaiken liikenteen porttien läpi, joten sääntöjen asettaminen on järjestelmän tietoturvallisuuden turvaamisen kannalta välttämätöntä. *Windows Defender* -palomuurilla kyetään porttien lisäksi rajoittamaan ohjelmien liikennettä. Se myöskin rajoittaa liikennettä heti asennuksen jälkeen.

Identiteetin suojaus on tärkeä osa Windows 10- ja Debian 10 -käyttöjärjestelmien tietoturvaa. Molemmissa käyttöjärjestelmissä on Pakollinen pääsynhallinta -järjestelmiä, joilla kyetään määrittämään käyttäjien ja ohjelmien oikeudet tiedostoihin, toimintoihin sekä resursseihin. Identiteetin suojauksen osalta Windows 10- ja Debian 10 -käyttöjärjestelmät ovat saman tasoiset.

Windows -käyttöjärjestelmän ensisijaisen osion NTFS -tiedostojärjestelmä tai Debianin ext4 -tiedostojärjestelmät eivät suojaa tiedon luottamuksellisuutta, mikäli dataa käsitellään käyttöjärjestelmien ulkopuolella, avaten ne esimerkiksi toisella tietokoneella. Windows 10- ja Debian 10 -käyttöjärjestelmien tiedostojärjestelmät ovatkin täysin suojaamattomia ilman levynsalausta. Tiedon saatavuutta sekä eheyttä voidaan kuitenkin parantaa Windows 10 -käyttöjärjestelmässä käyttäen muissa kuin ensisijaisissa osioissa ReFS -tiedostojärjestelmää.

Windows 10- ja Debian 10 -käyttöjärjestelmät käyttävät tiedon luottamuksellisuuden suojaamiseen levynsalaustyökaluja sekä eheyden suojaamiseen eheystarkastuksia sekä asennuspaketin digitaalista varmentamista. Koska merkittävä eroa ei ole salauksen ja eheystarkastuksien tietoturvalisäyksen osalta, Windows 10- ja Debian 10 -käyttöjärjestelmät ovat saman tasoiset.

Tunkeutumisen havaitsemiseen ja torjumiseen Windows 10 -käyttöjärjestelmässä on enemmän toimintoja kuin Debian 10 -käyttöjärjestelmässä. Näillä toiminnoilla kyetään suojaamaan myös järjestelmän käynnistystä. Debian 10 -käyttöjärjestelmässä oleva Linux -jakelupakettien ClamAV -haittaohjelman torjuntaohjelma on myös tehty ensisijaisesti torjumaan Windows -haittaohjelmia. Windows 10 -päätelaitteet saisivat merkittävän lisän haittaohjelmasuojaukseen verkkopohjaisesta ATP -suojauksesta, mutta taktisten päätelaitteiden tasolla tätä ei voida pitää oletusarvoisena.

Merkittävin ero Windows 10- ja Debian 10- käyttöjärjestelmissä on se, että Debian 10 -käyttöjärjestelmä perustuu avoimeen lähdekoodiin ja Windows 10 -käyttöjärjestelmä suljettuun. Avoimesta Debianista löydetään todennäköisemmin muun muassa virheitä ja haavoittuvuuksia, mikä pelkkänä tilastollisena vertailuna näyttää huonolta Debianin kannalta, tehden suljettujen järjestelmien käytöstä näennäisesti turvallisempaa.

Etuna Debianin avoimen lähdekoodin käytöstä on se, että sitä kyetään tarkastelemaan systemaattisesti kehitettäessä ja julkaisun jälkeen. Kuka tahansa voi tarkastaa koodia, jolloin esimerkiksi takaporttien piilottaminen ohjelmiin on hyvin vaikeaa ja virheiden löytäminen helppoa.

Tietoturvan kannalta on suotuisaa, että käytettävä käyttöjärjestelmä perustuu avoimeen lähdekoodiin. Se ei kuitenkaan takaa virheettömyyttä ja tietoturvallisuutta täysin, joten esimerkiksi Debianilla on monia tietoturvaominaisuuksia, jotka edistävät pääsyä tähän tavoitteeseen.

Kaikki käyttöjärjestelmät, niin kuin Windows 10- ja Debian 10 -käyttöjärjestelmät, ovat alttiita haittaohjelmille. Windows 10 -käyttöjärjestelmällä, niin kuin muillakin Windows -käyttöjärjestelmillä, on kuitenkin työpöytäkäytössä huomattavasti suurempi käyttäjäkunta. Tämän takia ne ovatkin huomattavasti kannattavampi kohde rikollisille, joten siihen on tehty huomattavasti suurempi määrä haittaohjelmia ja muita hyökkäysmenetelmiä verrattuna esimerkiksi Debian 10 -käyttöjärjestelmään.

6.1.3 Tietoturvakovennukset taktisten päätelaitteiden Windows 10- ja Debian 10 -käyttöjärjestelmiin

Käyttöjärjestelmien kehittäjät kohtaavat dilemman päättäessään, mitä toiminnollisuuksia tehdasasetuksilla käyttöjärjestelmässä on päällä. Haluna voi olla kilpailla ominaisuuksilla ja niiden määrällä, markkinoida käyttöjärjestelmää turvallisena ja asentamalla vain välttämättömät toiminnallisuudet tai jotain siltä väliltä. Käyttöjärjestelmät voivatkin olla alttiita tietoturvan vaarantumiselle alkuperäisillä asetuksillaan.

Koventamalla taktisien päätelaitteiden käyttöjärjestelmiä voidaan huomattavasti vähentää hyökkäyspinta-alaa. Tärkeimpiä toimenpiteitä ovat:

- käyttöjärjestelmän ja ohjelmien asennuspalvelimen suojaaminen
- asennuspaketin ja asennettavien ohjelmien autenttisuuden varmistaminen
- käyttöjärjestelmän tietoturvallisuuden parantaminen asennusvaiheessa jo asennusvaiheessa esimerkiksi asentamalla kovennettu Linux -ydin
- poistamalla tarpeettomat palvelut ja ohjelmat
- tekemällä esimerkiksi CIS Benchmark -oppaan suosittelemat kovennukset manuaalisesti tai ajamalla ne tekevä työkalu
- varmistamalla riittävän nopea tietoturva-auditoitujen päivityksien asentaminen.

Windows- ja Debian -käyttöjärjestelmien kehittyessä niiden kompleksisuus on samalla lisääntynyt, jonka johdosta Windows 10- ja Debian 10 -käyttöjärjestelmiin tarvitaan aikaisempiin versioihin verrattuna yhä enemmän kovennuksia.

Eroja on myös näiden käyttöjärjestelmien välillä. Windows 10 -käyttöjärjestelmään tarvitaankin kaksi kertaa enemmän kovennuksia Debian 10 -käyttöjärjestelmään verrattuna, joten pelkästään suosituskovennuksien määrää tarkastellen Debian 10 -käyttöjärjestelmää voitaisiin tämän valossa pitää tietoturvallisempana.

6.1.4 Windows 10- ja Debian 10 -käyttöjärjestelmien haavoittuvuudet

Laadukas menetelmä käyttöjärjestelmien haavoittuvuuksien analysointiin on CVSS -analyysi (Common Vulnerability Scoring System), joka on CVE-tietokannan haavoittuvuuksien vakavuuden arviointiin käytettävä avoin, standardoitu ja kaupallisista tavoitteista riippumaton pisteytysjärjestelmä. Analyysin avulla pisteytetään haavoittuvuuksien ominaisuuksia ajan ja käyttäjäympäristön suhteen, minkä avulla organisaatio pystyy muodostamaan kuvan haavoittuvuuden vakavuudesta. Vakavuustarkastelun avulla pystytään priorisoimaan ja kohdentamaan toimet haavoittuvuuksien aiheuttamien tietoturvauhkien paikkaamiseksi.

Jokaisen haavoittuvuuden analysointi alusta alkaen CVSS -analyysillä ei ole tehokasta. Analyysia tulisi laajentaa Puolustusvoimien omilla standardoiduilla mittareilla erikseen verkon eri tasoille, kuten esimerkiksi reitittimiin ja taktiseen päätelaitteisiin. Tämän ansiosta eri järjestelmiin kohdistuvat tekijät voitaisiin kertoimena ottaa huomioon haavoittuvuuksien vakavuuden analyysiin lopullisissa pisteissä.

Debian 10 -käyttöjärjestelmässä on oletuksena päällä enemmän tietoturvaominaisuuksia kuin esimerkiksi Ubuntussa tai monissa muissa Linux-jakelupaketeissa, mutta silti se sisältää eniten haavoittuvuuksia. Löydettyjen haavoittuvuuksien määrä ei aina korreloi heikomman tietoturvallisuuden kanssa. Suurempi merkitys onkin haavoittuvuuksien vakavuuskeskiarvolla. Vakavuuskeskiarvo indikoi, kuinka helposti haavoittuvuutta voidaan hyväksikäyttää.

Käyttöjärjestelmien haavoittuvuuksien määrä ei myöskään kerro käyttöjärjestelmän raportoimattomista tai vielä tuntemattomista haavoittuvuuksista sekä eroista ohjelmien laadussa. Se indikoi testaukselle kohdennetuista resursseista, testaajien määrästä sekä Debian 10:n tapauksessa myös sovelluksien suuresta määrästä (58 000).

Pelkän haavoittuvuuden sijasta kyberhyökkäyksien onnistumiseen vaikuttaa enemmän se, kuinka paljon resursseja ja kykyä kyberhyökkääjällä on. Kyberhyökkäyksen onnistuminen johtuu usein huonoista ohjelmista, mutta joskus myös käyttöjärjestelmien tietoturvaominaisuuksien ja tehtyjen kovennuksien puutteista.

Suurin hyöty haavoittuvuuksien analysoimisessa on ymmärryksen kasvu siitä, miten käyttöjärjestelmien ja sen ohjelmien tietoturvallisuus vaarantuu sekä miten kyberhyökkäykset toteutetaan. Ymmärryksen kasvun myötä haavoittuvuuksien vaikuttavuutta voidaan lieventää omin toimenpitein sekä kehittää kyberhyökkäyksien havainnointi- ja reagointimenetelmiä. Analysoinnin tuloksilla pitää olla vaikutusta myös siihen, kuinka esimerkiksi taktisia päätelaitteita kovennetaan, ja mitä ohjelmia niihin asennetaan.

Yksikään käyttöjärjestelmä, kuten Windows 10- ja Debian 10 -käyttöjärjestelmät, eivät ole täysin turvallisia ja kaikkiin kyetään vaikuttamaan jollain jopa tuntemattomalla hyökkäysmenetelmällä. Mielestäni tärkeintä käyttöjärjestelmän valintaprosessissa on tukeutua valmiisiin sertifikaatteihin, varmentaa ja paikata tietoturva-auditoinneilla niiden puutteita, koventaa tietoturva halutulle tasolle sekä käyttö- ja ylläpitovaiheessa päivittää käyttöjärjestelmää ja muita ohjelmia.

Yleisimmät haavoittuvuudet Windows 10- ja Debian 10 -käyttöjärjestelmiin mahdollistavat koodin suorittamisen, puskurin ylivuodon tai palveluneston kohdejärjestelmässä. Windows 10 -käyttöjärjestelmässä näiden lisäksi yleisiä ovat muistin korruption (muistia kyetään muokkaamaan ilman annettua työtehtävää) ja informaation saamisen aiheuttavat haavoittuvuudet. Yleisimpien haavoittuvuuksien jälkeen Debian 10 -käyttöjärjestelmässä seuraavaksi eniten on jonkun ohittamisen tai informaation saamisen aiheuttavia haavoittuvuuksia.

6.2. Validiteetti ja luotettavuus

Tutkimuksessa on käytetty paljon eri tason lähteitä. Tutkimuksissa on jouduttu hyödyntämään myös Microsoftin omia sekä Debian ja Linux-yhteisön julkaisuja. Tämänkaltaisissa lähteissä riskinä voi olla, että ne sisältävät markkinointimateriaaliksi suunnattua tietoa, jolloin se on subjektiivista. Subjektiivisuutta on pyritty vähentämään vertaamalla tietoa mahdollisimman moniin eri lähteisiin, kuten luotettaviin ja tietoturva-alalla arvostettuihin oppaisiin.

Tietoturva-aiheiseen kirjallisuusanalyysiin on vaikea löytää vielä ajallisesti valideja lähteitä, koska käyttöjärjestelmiä päivitetään ja ne muuttuvat paljon jopa puolen vuoden aikakehyksessä. Hyvänä esimerkkinä tutkimuksessa nousi esille määritelmä ”moderni käyttöjärjestelmä”. Modernia käyttöjärjestelmää ja niiden jo 80-luvulla yleistyneitä ominaisuuksia kuvailtiin esimerkiksi Stallingsin juuri päivitetyssä *Operating System - Internals and design principles* -oppaan 9. painoksessa moderneina.

CVSS -analyysin tutkimusmenetelmän käyttämisessä suurimpana ongelmana on, että tehdyt tulkinnat ovat osin subjektiivisia. Subjektiivisuutta voitaisiin vähentää, jos analyysin tekijöitä olisi riittävästi, jolloin suuremmasta otannasta saataisiin laskettua keskiarvo.

6.3. Tutkimustyö

6.3.1 Tutkimusprosessi

Tutkimustyö aloitettiin syyskuussa 2018 kirjallisuusanalyysin aineistohaulla ja samanaikaisesti kirjoitettiin tutkimussuunnitelmaa. Tutkimussuunnitelma oli valmis joulukuussa 2018. Tällöin tutkimusmenetelminä oli kirjallisuusanalyysi sekä asiantuntijahaastattelu. Tutkimussuunnitelmaseminaarin jälkeen alkoi varsinaisen tutkielman kirjoitus.

Molemmat ohjaajat vaihtuivat vuoden 2019 aikana, joka aiheutti tutkimustyössä suunnan muutosta. Asiantuntijahaastattelusta luovuttiin, koska arveltiin että siitä saatu sisältö olisi ollut liian subjektiivista. Samoihin aikoihin tutkimusmenetelmäksi lisättiin CVSS-analyysimenetelmä.

Vuoden 2020 alussa eteen tuli ongelma, jossa kirjallisuusanalyysin avulla ei saatu tarpeeksi tietoa Debian 10 -käyttöjärjestelmän tietoturvaominaisuuksista ja paketeista sekä näiden eroista Debian 9 -käyttöjärjestelmään. Tutkimukseen lisättiin tekninen koe, jolla selvitettiin Debian 10 -käyttöjärjestelmään asentuvat paketit ja ytimen tietoturvaominaisuudet. Tutkimustyö valmistui huhtikuun alussa vuonna 2020.

6.3.2 Jatkotutkimukset

Käyttöjärjestelmistä ja erityisesti niiden tietoturvallisuudesta on tehty todella vähän julkista tutkimusta Suomessa. Usein tehdyt tutkimukset ovat englanninkielisiä, ja keskittyvät tutkimaan käyttöjärjestelmiin kohdistuvia hyökkäysmenetelmiä. Monesti tietoturvatutkimuksella on taloudellisia intressejä, ja niitä ei välttämättä julkaista julkiseen jakoon.

Tutkimustuloksien kaikille saatavaksi jalkauttamisen kannalta haasteena Maanpuolustuskorkeakoululla sekä muissa Puolustusvoimien tutkimuslaitoksissa on, että jos tutkimuksessa käsitellään hyvin spesifisti esimerkiksi operatiivisia järjestelmiä, tutkimusta ei voida enää pitää julkisena.

Mahdollisia jatkotutkimusaiheita ovat:

CVSS-analyysimenetelmän ja Puolustusvoimien mittarin integrointi, jonka tavoitteena olisi tutkia haavoittuvuuksien vakavuutta eri järjestelmätasoilla ja laatia CVSS-analyysiin Puolustusvoimien omat integroitavat CVSS-mittarit. Puolustusvoimien mittareita voitaisiin hyödyntää haavoittuvuuksien vakavuuden arvioinnissa.

Tietoturvalliset Linux -jakelupaketit - Qubes, Tails, BlackArch ja Arch Linux, jonka tavoitteena olisi tutkia tietoturvallisiksi mainostettujen Linux-jakelupakettien todellista tietoturvallisuutta ja verrata niitä esimerkiksi Puolustusvoimilla jo käytössä oleviin Linux-jakelupakettien tietoturvallisuuteen. Tutkimustuloksena olisi analyysi siitä, että olisiko joku Linux-jakelupaketeista parempi vaihtoehto tietoturvallisuus näkökulmasta, verrattuna jo käytössä oleviin.

Käyttöjärjestelmien tietoturva-auditointi Puolustusvoimissa, jonka tavoitteena olisi tutkia, kuinka Puolustusvoimille käyttöön tulevien käyttöjärjestelmien tietoturvaa auditoidaan ja tarvitseeko auditointia kehittää. Tutkimuksella voitaisiin kehittää käyttöön otettavien käyttöjärjestelmien tietoturvallisuutta.

Astra Linux -käyttöjärjestelmän käyttö Venäjän asevoimissa, jonka tavoitteena olisi tutkia miksi Venäjän asevoimat siirtyivät käyttämään Astra Linux -käyttöjärjestelmiä, millaisia tietoturvaominaisuuksia Astra Linux -käyttöjärjestelmissä on, sekä miten haavoittuvaisia Astra Linux -käyttöjärjestelmät ovat.

Debian 11-käyttöjärjestelmän ominaisuudet ja tietoturvallisuus, jonka tavoitteena olisi tutkia vielä julkaisemattoman Debian 11 -käyttöjärjestelmän ominaisuuksia sekä tietoturvallisuutta. Tutkimuksen avulla voitaisiin esimerkiksi arvioida Debian 11 -käyttöjärjestelmän soveltuvuutta Puolustusvoimien päätelaitteisiin.

Taktisten päätelaitteiden käyttöjärjestelmien tietoturvan ylläpito, jonka tavoitteena olisi tutkia miten taktisten päätelaitteiden käyttöjärjestelmien tietoturvallisuutta ylläpidetään esimerkiksi päivittämisellä, ja miten tietoturvallisuuden ylläpitoa voidaan kehittää rauhan- ja sodan aikana. Tutkimusta voitaisiin hyödyntää esimerkiksi ohjesäännöissä ja koulutusmateriaaleissa.

LÄHTEET

[1] William Stallings. Operating System - Internals and design principles. 9.painos. Prentice Hall. 2018. ISBN 978-1-292-21429-0

[2] Haikala I., Järvinen H-M. Käyttöjärjestelmät. 2. painos. Helsinki: Talentum. 2004. 246 s. ISBN 952-14-0851-0.

[3] Ubuntu. Mikä käyttöjärjestelmä on? [internetsivu]. [viitattu 17.10.2018]. Saatavissa: http://ubuntu.fi/mika-kayttojarjestelma-on/

[4] Statcounter. Operating System Market Share [internetsivu]. [viitattu 15.2.2020]. Saatavissa: http://gs.statcounter.com/os-market-share

[5] W3Techs. Usage of operating systems for websites. 2020. [internetsivu]. [viitattu 27.1.2020]. Saatavissa: https://w3techs.com/technologies/overview/operating_system

[6] Andrew S. Tanenbaum. Modern operating systems. 4.painos. Pearson. 2015. ISBN 978-0-13-230998-1

[7] Ammaar A. Mufti. Developing a Natural, Intuitive Graphical User Interface for Mobile devices. Tampereen yliopisto. 2016

[8] Citrix. Mitä virtualisointi on?. [internetsivu]. [viitattu 22.1.2020]. Saatavissa: https://www.citrix.fi/glossary/what-is-virtualization.html

[9] Tomi Leppänen. Linuxin vuorontimen soveltuvuus ja säätäminen pelikäyttöön. Itä-Suomen Yliopisto, 2018. Saatavissa: https://epublications.uef.fi/pub/urn_nbn_fi_uef-20180818/urn_nbn_fi_uef-20180818.pdf

[10] Michael Kerrisk. Numa -overview of Non-Uniform Memory Architecture. [internetsivu]. [viitattu 26.02.2020]. Saatavissa: http://man7.org/linux/man-pages/man7/numa.7.html

[11] Sanastokeskus TSK ry. Kyberturvallisuuden sanasto. Helsinki. 2018. ISBN 978-952-5608-49-6. Saatavissa: https://www.tsk.fi/tiedostot/pdf/Kyberturvallisuuden_sanasto.pdf

[12] Michael E. Whitman & Herbert J. Mattord. Principles of information security. 6.painos. Cengage learning. 2017. ISBN 978-1-337-10206-3

[13] Sanastokeskus TSK ry. Tietoturvan termitalkoot. [internetsivu]. Saatavissa: http://www.tsk.fi/tsk/termitalkoot/fi/

[14] Y-lehti. Tietoturvasta huolehtiminen on elinehto. 2010. [internetsivu]. [viitattu 8.1.2020]. Saatavissa: https://www.y-lehti.fi/arkisto/artikkeli/3192/Tietoturvasta+huolehtiminen+on+elinehto+

[15] Securityaffairs. Russian military plans to replace Windows with Astra Linux. [internetsivu]. [viitattu 22.1.2020]. Saatavissa: https://securityaffairs.co/wordpress/86407/security/astra-linux-russia-army.html

[16] Astra Linux. Astra Linux Special edition. [internetsivu]. [viitattu 22.1.2020]. Saatavissa: https://astralinux.ru/en/products/astra-linux-special-edition/

[17] Valtiovarainministeriö. Päätelaitteiden tietoturvaohje. Helsinki, 2013. ISBN 978-952-251-520-9. Saatavissa: https://www.vahtiohje.fi/c/document_library/get_file?uuid=b1064d7a-83e5-4246-be9a-a8a84c8caaa0&groupId=10229

[18] Abraham Silberschatz. Operating system concepts. 9.painos. Courier-Kendallville 2013. 944s. ISBN 978-1-118-06333-0

[19] Tomi Karppinen. Haittaohjelmat ja niiden analyysi. Pro Gradu. Jyväskylä, 2014. Jyväskylän yliopisto. Saatavissa: https://jyx.jyu.fi/bitstream/handle/12345-6789/43832/1/URN%3ANBN%3Afi%3Ajyu-201406242136.pdf

[20] Michael Sikorski & Andrew Honig. Practical Malware Analysis The Hands-On Guide to Dissecting Malicious Software. San Francisco: No Starch Press. 2012. ISBN 978-1-59327-290-6

[21] Savolainen, N. Esikuntapanssariajoneuvojen tietoliikennelaitteiden tietoturvan ratkaisuvaihtoehtoja. Pro Gradu. Helsinki, 2015. Maanpuolustuskorkeakoulu

[22] ESET. Microsoft Windows 10 Security and Privacy. [verkkojulkaisu]. Saatavissa: https://www.welivesecurity.com/wp-content/uploads/2016/06/windows-10-security-privacy.pdf

[23] Mawarebytes. Rootkit. [internetsivu]. [viitattu 16.10.2019]. Saatavissa: https://blog.malwarebytes.com/detections/rootkit/

[24] Malwaretech. Inline Hooking for programmers. [internetsivu]. [viitattu 17.10.2019]. Saatavissa: https://www.malwaretech.com/2015/01/inline-hooking-for-programmers-part-1.html

[25] Malwarebytes. Bootkit. [internetsivu]. [viitattu 16.10.2019]. Saatavissa: https://blog.malwarebytes.com/detections/bootkit/

[26] Broderick Aquilino. Relevance of security features introduced in modern Windows OS. Aalto yliopisto, Helsinki. 2019. Saatavissa: https://aaltodoc.aalto.fi/bitstream/handle/123456789/38990/master_Aquilino_Broderick_2019.pdf?sequence=1&isAllowed=y

[27] TriggerScope: Towards detecting logic bombs in android applications. Northeastern University. Santa Barbara, 2016. [verkkojulkaisu]. Saatavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7546513

[28] Kyberturvallisuuskeskus. Meltdown- ja Spectre -hyökkäykset hyödyntävät prosessorien ongelmia. 2018. [internetsivu]. [viitattu 26.2.2019]. Saatavissa: https://www.kyberturvallisuuskeskus.fi/fi/meltdown-ja-spectre-hyokkaykset-hyodyntavat-prosessorien-ongelmia

[29] Moritz Lipp, et al. Meltdown: Reading Kernel Memory from User Space. [verkkojulkaisu]. Saatavissa: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-lipp.pdf

[30] Paul Kocher, et al. Spectre Attacks: Exploiting Speculative Execution. [verkkojulkaisu]. Saatavissa: https://spectreattack.com/spectre.pdf

[31] Meltdownattack. Meltdown and Spectre vulnerabilities in modern computers leak passwords and sensitive data. [verkkojulkaisu]. Saatavissa: https://meltdownattack.com/

[32] Kernel. Spectre Side Channels. 2020. [internetsivu]. [viitattu 24.1.2020]. Saatavissa: https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/spectre.html

[33] Elias Levy. Smashing the stack for fun and profit. 1996. [verkkojulkaisu]. Saatavissa: http://www-inst.eecs.berkeley.edu/~cs161/fa08/papers/stack_smashing.pdf

[34] Khalid Alharbi & Xiadong Lin. Preventing stack buffer overflow attacks. 2013. [verkkojulkaisu]. Saatavissa: https://patentimages.storage.googleapis.com/93/24/fa/a696e4843967ca/US9251373.pdf

[35] Seacord, R. Secure Coding in C and C++. Addison-Wesley, 2013. ISBN-13 978-0-321-82213-0

[36] NIST. Glossary of key information security terms. 2013. [verkkojulkaisu]. Saatavissa: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=913810

[37] Peter Bright. How security flaws work: The buffer overflow. 2015. [internetsivu]. [viitattu 10.1.2019]. Saatavissa: https://arstechnica.com/information-technology/2015/08/how-security-flaws-work-the-buffer-overflow/

[38] Daniel Trejo. After All These Years, the World is Still Powered by C Programming. [internetsivu]. [viitattu 23.1.2020]. Saatavissa: https://www.toptal.com/c/after-all-these-years-the-world-is-still-powered-by-c-programming

[39] Vincent Lextrait. The Programming Languages Beacon. 2016. [internetsivu]. [viitattu 23.1.2020]. Saatavissa: https://www.mentofacturing.com/Vincent/implementations.html

[40] Debian. Statistics. [internetsivu]. [viitattu 23.1.2020]. Saatavissa: https://sources.debian.org/stats/

[41] Computer Science. Computer Programming Languages. [internetsivu]. [viitattu 23.09.2019]. Saatavissa: https://www.computerscience.org/resources/computer-programming-languages/

[42] Wikipedia. Stack Buffer Overflow. [internetsivu]. [viitattu 26.1.2020]. Saatavissa: https://en.wikipedia.org/wiki/Stack_buffer_overflow

[43] Jason Andres & Steven Winterfeld. The basics of information security: understanding the fundamentals of infoSec in theory and practice. 2.painos. 2014. ISBN 978-0-12-800744-0

[44] Dell. Mitä on tietojen suorittamisen estäminen. 2019. [internetsivu]. [viitattu 24.1.2020]. Saatavissa: https://www.dell.com/support/article/fi/fi/fibsdt1/sln288643/mit%C3%A4-on-tietojen-suorittamisen-est%C3%A4minen?lang=fi

[45] Michael G. Solomon. Security Strategies In Windows Platforms and Applications. Jones & Bartlett Learning, LLC, 2019. Saatavissa: https://ebookcentral-proquest-com.mpenvoy.csc.fi/lib/ndul/detail.action?docID=5904926

[46] CISCO. What is Firewall?. [internetsivu]. [viitattu 11.09.2019]. Saatavissa: https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html

[47] Steven Splaine. Testing Web Security: Assessing the Security of Web Sites and applications. Wiley Publishing, Inc. 2002. ISBN 978-0471232810

[48] Valtiovarainministeriö. Mikä on sisäverkko. [internetsivu]. [viitattu 25.1.2020]. Saatavissa: https://www.vahtiohje.fi/web/guest/2.-mika-on-sisaverkko

[49] TechTarget. DMZ (networking). [internetsivu]. [viitattu 25.1.2020]. Saatavissa: https://searchsecurity.techtarget.com/definition/DMZ

[50] Techtarget. Antivirus software. 2017. [internetsivu]. [viitattu 3.1.2020]. Saatavissa: https://searchsecurity.techtarget.com/definition/antivirus-software

[51] Trend Micro. Memory resident. [internetsivu]. [viitattu 25.1.2020]. Saatavissa: https://www.trendmicro.com/vinfo/us/security/definition/memory-resident

[52] STEK. Tietojen suojaaminen ja tunnistaminen. 2020. [internetsivu]. [viitattu 9.1.2020]. Saatavissa: https://stek.fi/termit/autentikointi

[53] Charles Pfleeger, et al. Security in Computing. 5.painos. Prentice Hall. 2015. ISBN 978-0-13-408504-3

[54] Valtiovarainministeriö. VAHTI 2/2015 Ohje salauskäytännöistä. 2015, Helsinki. ISBN 978-952-251-726-5. Saatavissa: https://www.vahtiohje.fi/web/guest/2/2015-ohje-salauskaytannoista

[55] Liikenne- ja viestintäministeriö. Sähköisen viestinnän salaus- ja suojausmenetelmät. 2018. [verkkojulkaisu]. Saatavissa: https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/160614/LVM_02_2018_Sahkoisen_viestinnan%20salaus_ja_suojaus.pdf?sequence=1&isAllowed=y

[56] Surveillance Self-Defense. What should I know About Encryption? 2018. [internetsivu]. [viitattu 06.03.2020]. Saatavissa: https://ssd.eff.org/en/module/what-should-i-know-about-encryption

[57] Viestintävirasto. Ylläpitäjä: Tiivistä ja suolaa salasanat. 2014. [internetsivu]. [viitattu 03.03.2020]. Saatavissa: https://legacy.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2014/12/ttn201412220912.html

[58] Kyberturvallisuuskeskus. SHA-1-tiivistefunktio on lopullisesti murrettu. 2020. [internetsivu]. [viitattu 04.03.2020]. Saatavissa: https://www.kyberturvallisuuskeskus.fi/fi/ajankohtaista/sha-1-tiivistefunktio-lopullisesti-murrettu

[59] Md5hashing. Sha256. [internetsivu]. [viitattu 04.03.2020]. Saatavissa: https://md5hashing.net/hash/sha256/

[60] Mikrobitti. UEFI-käynnistysohjelmisto. 2017. [verkkojulkaisu]. Saatavissa: https://www.mikrobitti.fi/neuvot/tiesitko-etta-koneessasi-on-uusi-kaynnistysohjelmisto-nain-saadat-uefi-asetukset-oikein/5a944995-96d0-39bc-bb9d-8ce4dce253ab

[61] Microsoft. Secure the Windows 10 boot process. 2018. [internetsivu]. [viitattu 11.10.2019]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/information-protection/secure-the-windows-10-boot-process

[62] The Linux Information Project. Kernel Definition. [internetsivu]. [viitattu 03.03.2020]. Saatavisssa: www.linfo.org/kernel.htm

[63] Janne Kautiainen. Järjestelmien siirtäminen uuteen käyttöjärjestelmäympäristöön. Diplomityö. Lappeenrannan teknillinen yliopisto, 2010. Saatavissa: https://lut-pub.lut.fi/bitstream/handle/10024/63014/nbnfi-fe201208176312.pdf?sequence=2&isAllowed=y

[64] Microsoft. Windows Defender System Guard. [internetsivu]. [viitattu 29.1.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-system-guard/how-hardware-based-root-of-trust-helps-protect-windows

[65] Juha Kalliomäki. Suorituskyvyn turvaaminen virtualisoidussa ympäristössä. Pro Gradu-tutkielma. Helsingin yliopisto, 2012. Saatavissa: https://helda.helsinki.fi/bitstream/handle/10138/38134/Juho_Kalliomki_Gradu_ja_tiivistelm.pdf

[66] Zunesis. Why It's Important to Install Windows updates and patches? [internetsivu]. [viitattu 27.1.2020]. Saatavissa: https://www.zunesis.com/why-install-windows-updates/

[67] Australian Cyber Security Centre. Hardening Microsoft Windows 10 version 1709 workstations. 2019. [verkkojulkaisu]. Saatavissa: https://www.cyber.gov.au/publications

[68] Gaoshou Zhai, et al. Security Testing for Operating System and Its System Calls. [verkkojulkaisu]. Saatavissa: https://www.researchgate.net/publication/221318278_Security_Testing_for_Operating_System_and_Its_System_Calls

[69] Anssi Kärkkäinen. A Cyber Security architecture for military networks usinga a cognitive network approach. Maanpuolustuskorkeakoulu, 2013. Saatavissa: https://www.doria.fi/bitstream/handle/10024/92640/Y2642_K%c3%a4rkk%c3%a4inenAP_YEK56.pdf?sequence=2&isAllowed=y

[70] Common Criteria. Common Criteria for information technology security evaluation. 2017. [verkkojulkaisu]. Saatavissa: https://www.commoncriteriaportal.org/cc/

[71] Maria Tulensalo. Common Criteria IT security standard in product development process. Aalto-yliopisto, 2010. Saatavissa: http://lib.tkk.fi/Dipl/2010/urn100312.pdf

[72] Microsoft. FIPS 140-2 Validation. 2019. [viitattu 16.2.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation

[73] Terry Myerson. Announcing Windows 10. [internetsivu]. [viitattu 15.11.2018]. Saatavissa: https://blogs.windows.com/windowsexperience/2014/09/30/announcing-windows-10/

[74] Tim Danton & Matti Kiianmies. Windows 10 käyttäjänopas. Porvoo: Bookwell Oy. 2015. 145 s. ISBN 978-952-321-138-4

[75] Microsoft. Virtualization-based Security (VBS). 2017. [internetsivu]. [viitattu 11.12.2019]. Saatavissa: https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-vbs

[76] Microsoft. Memory integrity. 2019. [internetsivu]. [viitattu 11.12.2019]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/threat-protection/device-guard/memory-integrity

[77] Dell. Windows 10 Enterprise Security: Credential Guard ja Device Guard. [internetsivu]. [viitattu 22.08.2019]. Saatavissa: https://www.dell.com/support/article/fi/fi/fidhs1/sln304974/windows-10-enterprise-security-tunniste-tietojen-guard-ja-device-guard?lang=fi

[78] Rafal Wojtczuk. Analysis of the attack surface of Windows 10 Virtualization-based-secucurity. Blackhat, 2016. [verkkojulkaisu]. Saatavissa: https://www.blackhat.com/docs/us-16/materials/us-16-Wojtczuk-Analysis-Of-The-Attack-Surface-Of-Windows-10-Virtualization-Based-Security.pdf

[79] Shrikant Joshi. Virtualization Based Security (VBS) and Hypervisor Enforced Code Integrity (HVCI) for Olympia users!. Microsoft, 2018. [internetsivu]. [viitattu 13.03.2020]. Saatavissa: https://techcommunity.microsoft.com/t5/windows-insider-program/virtualization-based-security-vbs-and-hypervisor-enforced-code/m-p/240571#

[80] Jari Piippola. Käyttöjärjestelmien tietoturvaratkaisut. Pro Gradu-tutkielma. Oulun yliopisto, 2014. Saatavissa: http://jultika.oulu.fi/files/nbnfioulu-201405281537.pdf

[81] Microsoft. How to enable Structured Exception Handling Overwrite Protection (SEHOP) in Windows operating systems. [internetsivu]. [viitattu 02.02.2020] Saatavissa: https://support.microsoft.com/en-in/help/956607/how-to-enable-structured-exception-handling-overwrite-protection-sehop

[82] Chuck Easttom. Computer Security Fundamentals. Pearson IT Certification, 2019. ISBN 978-0-13-577477-9

[83] Microsoft. Protect derived domain credentials with Windows Defender Credential Guard. [intenetsivu]. [viitattu 03.02.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard

[84] If-Koubou. Mikä on ReFS (Resilient File System) Windowsissa. [internetsivu]. [viitattu 29.1.2020]. Saatavissa: https://fi.if-koubou.com/articles/how-to/what-is-refs-the-resilient-file-system-on-windows.html

[85] Microsoft. Resilient File System (ReFS) overview. [internetsivu]. [viitattu 29.1.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows-server/storage/refs/refs-overview

[86] Microsoft. Device encryption in Windows 10. [internetsivu]. [viitattu 23.09.2019]. Saatavissa: https://support.microsoft.com/en-us/help/4502379/windows-10-device-encryption

[87] TheWindowsClub. Encrypting File System (EFS) on Windows 10 explained. [internetsivu]. [viitattu 03.02.2020]. Saatavissa: https://www.thewindowsclub.com/encrypting-file-system-efs-windows-10

[88] Microsoft. File Encryption. [internsivu]. [viitattu 03.02.2020]. Saatavissa: https://docs.microsoft.com/fi-fi/windows/win32/fileio/file-encryption

[89] Microsoft. Protect your enterpise data using Windows Information Protection (WIP). [internetsivu]. [viitattu 03.02.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/information-protection/windows-information-protection/protect-enterprise-data-using-wip

[90] Microsoft. Windows Defender Antivirus. [internetsivu]. [viitattu 06.06.2019]. Saatavissa: https://www.microsoft.com/fi-fi/windows/comprehensive-security

[91] AV-test. Windows 10 - Windows Defender Antivirus 4.18. 2019. [internetsivu]. [viitattu 13.03.2020]. Saatavissa: https://www.av-test.org/en/antivirus/business-windows-client/windows-10/december-2019/microsoft-windows-defender-antivirus-4.18-195015/

[92] Microsoft. Windows Defender Application Guard overview. [internetsivu]. [viitattu 29.1.2020]. Saatavissa: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-guard/wd-app-guard-overview

[93] AV-test. Windows 8.1 - System Center Endpoint Protection. 2016. [internetsivu]. [viitattu 13.03.2020]. Saatavissa: https://www.av-test.org/en/antivirus/business-windows-client/windows-8/october-2016/microsoft-system-center-endpoint-protection-4.9-164174/

[94] Debian. Tietoa Debianista. [internetsivu]. [viitattu 10.12.2018]. Saatavissa: https://www.debian.org/intro/about#what

[95] Debian. What's new in Debian 10. [internetsivu]. [viitattu 18.02.2020]. Saatavissa: https://www.debian.org/releases/stable/amd64/release-notes/ch-whats-new.html

[96] Linux. NFTAbles. 2019. [internetsivu]. [viitattu 25.03.2020]. Saatavissa: https://www.linux.fi/wiki/NFTables

[97] Kernel. Kernel Self-Protection. [internetsivu]. [viitattu 03.02.2020]. Saatavissa: https://www.kernel.org/doc/html/latest/security/self-protection.html#

[98] Jake Edge. A seccomp overview. LWN, 2015. [internetsivu]. [viitattu 22.02.2020]. Saatavissa: https://lwn.net/Articles/656307/

[99] Zhilong Wang , et al. To detect stack buffer overflow with polymorphic canaries. 2018. [verkkojulkaisu]. Saatavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8416487

[100] Robert C. Seacord. Secure Coding in C and C++. Addison-Wesley Professional, 2013. ISBN-13 978-0321822130

[101] David Black-Schaffer. Virtual Memory: 9 Memory Protection. [näyttöesitys]. Saatavissa: http://bit.ly/2Shoq9z

[102] Siddharth Sharma. Enhance application security with FORTIFY_SOURCE. [internetsivu]. [viitattu 01.03.2020]. Saatavissa: https://access.redhat.com/blogs/766093/posts/1976213

[103] Debian. DebianFIrewall. [internetsivu]. [viitattu 02.03.2020]. Saatavissa: https://wiki.debian.org/DebianFirewall

[104] Debian. Introduction to AppArmor. [internetsivu]. [viitattu 23.02.2020]. Saatavissa: https://debian-handbook.info/browse/stable/sect.apparmor.html

[105] Clemens Fruhwirth. LUKS1 On-Disk Format Specification. [verkkojulkaisu]. Saatavissa: http://cdn.kernel.org/pub/linux/utils/cryptsetup/LUKS_docs/on-disk-format.pdf

[106] Milan Broz. Cryptsetup 2.3.0 Release notes. 2019. [internetsivu]. [viitattu 15.02.2020]. Saatavissa: https://gitlab.com/cryptsetup/cryptsetup/-/blob/master/docs/v2.3.0-ReleaseNotes

[107] Patchwork. Provide toggle for BUG on data corruption. [internetsivu]. [viitattu 01.03.2020]. Saatavissa: https://patchwork.kernel.org/patch/9286627/

[108] OpenSSL. OpenSSL Overview. [internetsivu]. [viitattu 02.03.2020]. Saatavissa: https://wiki.openssl.org/index.php/OpenSSL_Overview

[109] Muhammad Vandestra. List Anti Rootkit & AntiVirus for Ubuntu, Linux & BSD. Dragon Promedia Ebook Publisher, 2018. ISBN 978-0463932834

[110] Clamav. Introduction. [internetsivu]. [viitattu 23.02.2020]. Saatavissa: https://www.clamav.net/documents/introduction

[111] CIS Security. CIS Benchmarks FAQ. [internetsivu]. [viitattu 23.02.2020]. Saatavissa: https://www.cisecurity.org/cis-benchmarks/cis-benchmarks-faq/

[112] CIS Benchmarks. Securing Debian Linux. [verkkojulkaisu]. Saatavissa: https://www.cisecurity.org/benchmark/debian_linux/

[113] CIS Benchmarks. Securing Microsoft Windows Desktop. [verkkojulkaisu]. Saatavissa: https://www.cisecurity.org/benchmark/microsoft_windows_desktop/

[114] Australian Cyber Security Centre. Asessing security vulnerabilities and applying patches. 2019 [verkkojulkaisu] Saatavissa: https://www.cyber.gov.au/publications

[115] Application Whitelisting. Tampereen teknillinen yliopisto. 2013. [verkkojulkaisu]. [viitattu 22.08.2019]. Saatavissa: https://wiki.tut.fi/Tietoturva/Tutkielmat/Application-Whitelisting

[116] CVE Details. Current CVSS Score Distribution For All Vulnerabilities. [internetsivu]. [viitattu 07.02.2020]. Saatavissa: https://www.cvedetails.com

[117] First. Common vulnerability scoring system v3.0: Specification document. [verkkojulkaisu]. Saatavissa: https://www.first.org/cvss/cvss-v30-specification-v1.8.pdf

[118] Nixu. CVE – haavoittuvuusmaailman kolme tärkeää kirjainta. [internetsivu]. [viitattu 30.05.2019]. Saatavissa: https://www.nixu.com/fi/blog/cve-haavoittuvuusmaailman-kolme-tarkeaa-kirjainta

[119] NVD. Common Vulnerabilities and exposures. [internetsivu]. [viitattu 30.05.2019]. Saatavissa: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-8174

[120] First. Common vulnerability scoring system v3.1: Specification document. [verkkojulkaisu]. Saatavissa: https://www.first.org/cvss/v3.1/specification-document

[121] Hugo Santos. Measuring the severity of vulnerabilities: changes in CVSS 3.1. Incibecert. [internetsivu]. [viitattu 24.02.2020]. Saatavissa: https://www.incibecert.es/en/blog/measuring-severity-vulnerabilities-changes-cvss-31

[122] Ville Rantamäki. Taisteluosastoon kohdistuva kyberuhka. Pro Gradu. Helsinki, 2018. Maanpuolustuskorkeakoulu

DEBIAN 10 -KÄYTTÖJÄRJESTELMÄN TEKNINEN KOE

Debian 10.3 -alaversion Linux-ytimen tietoturvaominaisuudet:

```
root@A8-5E-45-15-26-BD:/home/jose/Asiakirjat/kconfig-hardened-check-master# ./kconfig-hardened-check.py -c debian-buster.config
[+] Trying to detect architecture in "config_files/distros/debian-buster.config"...
[+] Detected architecture: X86_64
[+] Checking "config_files/distros/debian-buster.config" against hardening preferences...
                option name                | desired val | decision |       reason       |   check result
=========================================================================================================================
CONFIG_BUG                                 |      y      |defconfig |  self_protection   |   OK
CONFIG_STRICT_KERNEL_RWX                   |      y      |defconfig |  self_protection   |   OK
CONFIG_STACKPROTECTOR_STRONG               |      y      |defconfig |  self_protection   |   OK
CONFIG_SLUB_DEBUG                          |      y      |defconfig |  self_protection   |   OK
CONFIG_STRICT_MODULE_RWX                   |      y      |defconfig |  self_protection   |   OK
CONFIG_MICROCODE                           |      y      |defconfig |  self_protection   |   OK
CONFIG_RETPOLINE                           |      y      |defconfig |  self_protection   |   OK
CONFIG_X86_SMAP                            |      y      |defconfig |  self_protection   |   OK
CONFIG_X86_UMIP                            |      y      |defconfig |  self_protection   |   OK: CONFIG_X86_INTEL_UMIP "y"
CONFIG_IOMMU_SUPPORT                       |      y      |defconfig |  self_protection   |   OK
CONFIG_SYN_COOKIES                         |      y      |defconfig |  self_protection   |   OK
CONFIG_PAGE_TABLE_ISOLATION                |      y      |defconfig |  self_protection   |   OK
CONFIG_RANDOMIZE_MEMORY                    |      y      |defconfig |  self_protection   |   OK
CONFIG_INTEL_IOMMU                         |      y      |defconfig |  self_protection   |   OK
CONFIG_AMD_IOMMU                           |      y      |defconfig |  self_protection   |   OK
CONFIG_VMAP_STACK                          |      y      |defconfig |  self_protection   |   OK
CONFIG_RANDOMIZE_BASE                      |      y      |defconfig |  self_protection   |   OK
CONFIG_THREAD_INFO_IN_TASK                 |      y      |defconfig |  self_protection   |   OK
CONFIG_BUG_ON_DATA_CORRUPTION              |      y      |   kspp   |  self_protection   |   OK
CONFIG_DEBUG_WX                            |      y      |   kspp   |  self_protection   |   OK
CONFIG_SCHED_STACK_END_CHECK               |      y      |   kspp   |  self_protection   |   OK
CONFIG_SLAB_FREELIST_HARDENED              |      y      |   kspp   |  self_protection   |   OK
CONFIG_SLAB_FREELIST_RANDOM                |      y      |   kspp   |  self_protection   |   OK
CONFIG_SHUFFLE_PAGE_ALLOCATOR              |      y      |   kspp   |  self_protection   |   FAIL: not found
CONFIG_FORTIFY_SOURCE                      |      y      |   kspp   |  self_protection   |   OK
CONFIG_GCC_PLUGINS                         |      y      |   kspp   |  self_protection   |   FAIL: not found
CONFIG_GCC_PLUGIN_RANDSTRUCT               |      y      |   kspp   |  self_protection   |   FAIL: not found
CONFIG_GCC_PLUGIN_LATENT_ENTROPY           |      y      |   kspp   |  self_protection   |   FAIL: not found
CONFIG_DEBUG_LIST                          |      y      |   kspp   |  self_protection   |   OK
CONFIG_DEBUG_SG                            |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_DEBUG_CREDENTIALS                   |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_DEBUG_NOTIFIERS                     |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_HARDENED_USERCOPY                   |      y      |   kspp   |  self_protection   |   OK
CONFIG_HARDENED_USERCOPY_FALLBACK          | is not set  |   kspp   |  self_protection   |   OK
CONFIG_MODULE_SIG                          |      y      |   kspp   |  self_protection   |   OK
CONFIG_MODULE_SIG_ALL                      |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_MODULE_SIG_SHA512                   |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_MODULE_SIG_FORCE                    |      y      |   kspp   |  self_protection   |   FAIL: "is not set"
CONFIG_DEFAULT_MMAP_MIN_ADDR               |    65536    |   kspp   |  self_protection   |   OK
CONFIG_REFCOUNT_FULL                       |      y      |   kspp   |  self_protection   |   OK
CONFIG_INIT_STACK_ALL                      |      y      |  clipos  |  self_protection   |   FAIL: not found
CONFIG_INIT_ON_ALLOC_DEFAULT_ON            |      y      |  clipos  |  self_protection   |   FAIL: not found
CONFIG_INIT_ON_FREE_DEFAULT_ON             |      y      |  clipos  |  self_protection   |   OK: CONFIG_PAGE_POISONING "y"
CONFIG_SECURITY_DMESG_RESTRICT             |      y      |  clipos  |  self_protection   |   OK
CONFIG_DEBUG_VIRTUAL                       |      y      |  clipos  |  self_protection   |   FAIL: "is not set"
CONFIG_STATIC_USERMODEHELPER               |      y      |  clipos  |  self_protection   |   FAIL: "is not set"
CONFIG_SLAB_MERGE_DEFAULT                  | is not set  |  clipos  |  self_protection   |   FAIL: "y"
CONFIG_GCC_PLUGIN_RANDSTRUCT_PERFORMANCE   | is not set  |  clipos  |  self_protection   |   FAIL: CONFIG_GCC_PLUGIN_RANDSTRUCT is needed
CONFIG_GCC_PLUGIN_STACKLEAK                |      y      |  clipos  |  self_protection   |   FAIL: not found
CONFIG_STACKLEAK_METRICS                   | is not set  |  clipos  |  self_protection   |   FAIL: CONFIG_GCC_PLUGIN_STACKLEAK is needed
CONFIG_STACKLEAK_RUNTIME_DISABLE           | is not set  |  clipos  |  self_protection   |   FAIL: CONFIG_GCC_PLUGIN_STACKLEAK is needed
CONFIG_RANDOM_TRUST_CPU                    | is not set  |  clipos  |  self_protection   |   FAIL: "y"
CONFIG_INTEL_IOMMU_SVM                     |      y      |  clipos  |  self_protection   |   OK
CONFIG_INTEL_IOMMU_DEFAULT_ON              |      y      |  clipos  |  self_protection   |   FAIL: "is not set"
CONFIG_SLUB_DEBUG_ON                       |      y      |    my    |  self_protection   |   FAIL: "is not set"
CONFIG_RESET_ATTACK_MITIGATION             |      y      |    my    |  self_protection   |   FAIL: "is not set"
CONFIG_AMD_IOMMU_V2                        |      y      |    my    |  self_protection   |   OK
CONFIG_SECURITY                            |      y      |defconfig |  security_policy   |   OK
CONFIG_SECURITY_WRITABLE_HOOKS             | is not set  |defconfig |  security_policy   |   OK: not found
CONFIG_SECURITY_YAMA                       |      y      |   kspp   |  security_policy   |   OK
CONFIG_SECURITY_LOADPIN                    |      y      |    my    |  security_policy   |   FAIL: "is not set"
CONFIG_SECURITY_LOCKDOWN_LSM               |      y      |    my    |  security_policy   |   FAIL: not found
CONFIG_SECURITY_LOCKDOWN_LSM_EARLY         |      y      |    my    |  security_policy   |   FAIL: not found
CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY|      y      |    my    |  security_policy   |   FAIL: not found
CONFIG_SECURITY_SAFESETID                  |      y      |    my    |  security_policy   |   FAIL: not found
CONFIG_SECCOMP                             |      y      |defconfig | cut attack surface |   OK
```

option name	desired val	decision	reason	check result
CONFIG_SECCOMP_FILTER	y	defconfig	cut_attack_surface	OK
CONFIG_STRICT_DEVMEM	y	defconfig	cut_attack_surface	OK
CONFIG_MODULES	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_DEVMEM	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_IO_STRICT_DEVMEM	y	kspp	cut_attack_surface	OK
CONFIG_ACPI_CUSTOM_METHOD	is not set	kspp	cut_attack_surface	OK
CONFIG_COMPAT_BRK	is not set	kspp	cut_attack_surface	OK
CONFIG_DEVKMEM	is not set	kspp	cut_attack_surface	OK
CONFIG_COMPAT_VDSO	is not set	kspp	cut_attack_surface	OK
CONFIG_BINFMT_MISC	is not set	kspp	cut_attack_surface	FAIL: "m"
CONFIG_INET_DIAG	is not set	kspp	cut_attack_surface	FAIL: "m"
CONFIG_KEXEC	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_PROC_KCORE	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_LEGACY_PTYS	is not set	kspp	cut_attack_surface	OK
CONFIG_HIBERNATION	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_LEGACY_VSYSCALL_NONE	y	kspp	cut_attack_surface	OK
CONFIG_IA32_EMULATION	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_X86_X32	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_MODIFY_LDT_SYSCALL	is not set	kspp	cut_attack_surface	FAIL: "y"
CONFIG_X86_PTDUMP	is not set	grsecurity	cut_attack_surface	OK
CONFIG_ZSMALLOC_STAT	is not set	grsecurity	cut_attack_surface	OK
CONFIG_PAGE_OWNER	is not set	grsecurity	cut_attack_surface	OK
CONFIG_DEBUG_KMEMLEAK	is not set	grsecurity	cut_attack_surface	OK
CONFIG_BINFMT_AOUT	is not set	grsecurity	cut_attack_surface	OK: not fou
CONFIG_KPROBES	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_UPROBES	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_GENERIC_TRACER	is not set	grsecurity	cut_attack_surface	FAIL: "y" nd
CONFIG_PROC_VMCORE	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_PROC_PAGE_MONITOR	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_USELIB	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_CHECKPOINT_RESTORE	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_USERFAULTFD	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_HWPOISON_INJECT	is not set	grsecurity	cut_attack_surface	FAIL: "m"
CONFIG_MEM_SOFT_DIRTY	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_DEVPORT	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_DEBUG_FS	is not set	grsecurity	cut_attack_surface	FAIL: "y"
CONFIG_NOTIFIER_ERROR_INJECTION	is not set	grsecurity	cut_attack_surface	FAIL: "m"
CONFIG_ACPI_TABLE_UPGRADE	is not set	lockdown	cut_attack_surface	FAIL: "y"
CONFIG_ACPI_APEI_EINJ	is not set	lockdown	cut_attack_surface	OK
CONFIG_PROFILING	is not set	lockdown	cut_attack_surface	FAIL: "y"
CONFIG_BPF_SYSCALL	is not set	lockdown	cut_attack_surface	FAIL: "y"
CONFIG_MMIOTRACE_TEST	is not set	lockdown	cut_attack_surface	OK
CONFIG_KSM	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_KALLSYMS	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_X86_VSYSCALL_EMULATION	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_MAGIC_SYSRQ	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_KEXEC_FILE	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_USER_NS	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_LDISC_AUTOLOAD	is not set	clipos	cut_attack_surface	FAIL: "y"
CONFIG_MMIOTRACE	is not set	my	cut_attack_surface	FAIL: "y"
CONFIG_LIVEPATCH	is not set	my	cut_attack_surface	FAIL: "y"
CONFIG_IP_DCCP	is not set	my	cut_attack_surface	FAIL: "m"
CONFIG_IP_SCTP	is not set	my	cut_attack_surface	FAIL: "m"
CONFIG_FTRACE	is not set	my	cut_attack_surface	FAIL: "y"
CONFIG_BPF_JIT	is not set	my	cut_attack_surface	FAIL: "y"
CONFIG_VIDEO_VIVID	is not set	my	cut_attack_surface	FAIL: "m"
CONFIG_ARCH_MMAP_RND_BITS	32	clipos	userspace_hardening	FAIL: "28"

```
[+] config check is finished: 'OK' - 54 / 'FAIL' - 69
root@A8-5E-45-15-26-BD:/home/jose/Asiakirjat/kconfig-hardened-check-master#
```