Juha Moisio

Selaimen sormenjälkitunnistamisen torjunta käyttöjärjestelmäavusteisella virtualisoinnilla

Tietotekniikan pro gradu -tutkielma

4. lokakuuta 2017

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Juha Moisio

Yhteystiedot: juha.pa.moisio@student.jyu.fi

Ohjaaja: Ville Tirronen

Työn nimi: Selaimen sormenjälkitunnistamisen torjunta käyttöjärjestelmäavusteisella virtualisoinnilla

Title in English: Preventing browser fingerprinting using operating system level virtualization

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikan maisterinohjelma

Sivumäärä: 62+9

Tiivistelmä: Selaimen sormenjälkitunnistaminen mahdollistaa käyttäjien seurannan käyttäjien yksityisyyttä luokkaavasti. Tutkielmassa selvitetään voidaanko käyttöjärjestelmäavusteisilla virtualisointiteknologioilla vastata selaimen sormenjälkitunnistamisen torjunnan asettamiin haasteisiin. Tutkielmassa havaittiin neljä eri haastetta selaimen sormenjälkitunnistamisen torjunnalle. Tutkielma keskittyy Linux-kernelin tukemiin käyttöjärjestelmäavusteisiin virtualisointiteknologiohin, jotka mahdollistavat ohjelmistokonttien toteuttamisen.

Avainsanat: ohjelmistokontti, virtualisointi, yksityisyys, selaimen sormenjälkitunnistaminen, pro gradu -tutkielmat

Abstract: Browser fingerprinting enables tracking of users in a way that violates users' privacy. In this thesis, operating system level virtualisation technologies are observed to overcome the the challenges of browser fingerprinting prevention. Four different challenges were detected in this thesis for preventing browser fingerprinting. The focus of the thesis is on the operating system level virtualisation technologies that are supported by the Linux-kernel and which enables the creation of software containers.

Keywords: container, virtualization, privacy, browser fingerprinting, Master's Theses

Termiluettelo

DNS	(Domain Name System) on protokolla numeeristen IP-osoitteiden ja sanallisten verkkotunnusten väliseen muunnokseen Internetissä.
DNT	(Do Not Track) on HTTP-protokollan valinnainen otsaketietue, jolla www-asiakas voi ilmaista tahtonsa käyttäjänseurannan toteuttamiselle.
Entropia	on muuttujan sisältämän informaation määrän ilmaisemiseen käytetty suure.
Eväste	on www-selaimen HTTP-protokollalla tallettama ja lähettämä tilallista tietoa sisältävä tietue.
HTTP	(Hypertext Transfer Protocol) on protokolla www-asiakkaan ja www-palvelun väliseen tiedonsiirtoon.
IP	(Internet Protocol) on protokolla Internetin toteuttamiseen.
JSON	(JavaScript Object Notation) on standardoitu oliopohjaisen tiedon esitystapa tiedon käsittelyyn.
LXD	on ohjelma, joka lisää taustapalvelut ohjelmistokonttien hallintaan ja jakeluun.
LXC	on ohjelma, joka mahdollistaa Linux-kernelin käyttöjärjestelmäavusteisen virtualisoinnin käyttämisen käyttäjätasolla.
NPAPI	(Netscape Plugin Application Programming Interface) on selaimissa käytetty ohjelmointirajapinta liitännäisten toteuttamiseen.
POSIX	(Portable Operating System Interface for uniX) on standardi käyttöjärjestelmän toteuttamiselle.
SHA	(Secure Hash Algorithm) on algoritmi tiivisteen laskemiseen.
TCP	(Transmission Control Protocol) on protokolla tiedonsiirtoyhteyksien luontiin internetverkoissa.
WWW	(World Wide Web) on menetelmä nimettyjen resurssien, kuten dokumenttien, jakamiseen ja viittaamiseen Internetissä.

Kuviot

Taulukot

Sisältö

1 Johdanto

Euroopan parlamentin ja neuvoston laatimassa sähköisen viestinnän tietosuojadirektiivissä on säädetty henkilötietojen käsittelystä ja yksityisyyden suojasta sähköisen viestinnän alalla (European Communities 2002). Sähköisen viestinnän tietosuojadirektiivissä edellytetään käyttäjän tietoisuutta sähköisten henkilötietojen keräykselle ja käyttäjän toimien seuraamiselle (ks. European Communities 2002, pykälä 24). Menetelmiltä, kuten evästeiden käytöltä, edellytetään käyttäjien tietoisuutta ja hyväksyntää (ks. European Communities 2002, pykälä 25). Direktiivin tulkintakäytännöstä poiketen www-palvelut joko kysyvät käyttäjiltään luvan evästeiden käytölle tai vaihtoehtoisesti tarjoavat lisätietoa evästeiden käytöstä (Viestintävirasto 2017). Nykyisestä direktiivin tulkintakäytännöstä voi kuitenkin saada sen vaikutelman, että pelkästään evästeitä käytetään käyttäjien tunnistamiseen ja seurantaan.

Tutkimusaiheena oleva selaimen sormenjälkitunnisteen käyttäminen verkkokäyttäjien seurantamenetelmänä ei edellytä evästeiden käyttämistä (Eckersley 2010). Selaimen sormenjälkitunnisteeseen perustuva seurantamenetelmä on tullut www-käyttäjien yleiseen tietoisuuteen menetelmän tehokkuutta testaavien www-palveluiden myötä. Www-palvelut, kuten Panoptclicks [1] ja AmIUnique [2] mahdollistavat oman selaimensa kokoonpanon sormenjälkitunnisteen yksilöllisyyden mittaamisen. Sekä Panoptclicksin että AmIUniquen keräämää tietoa on hyödynnetty tutkimuksissa, jotka ovat osoittaneet selaimen sormenjälkitunnistamisen mahdollistavan käyttäjien laajamittaisen seurannan (Eckersley 2010; Laperdrix, Rudametkin ja Baudry 2016).

Ajankohtaisena tutkimusaiheena selaimen sormenjälkitunnistamissa on torjuntamenetelmien kehittäminen. Eri tutkimusten osalta on esitetty torjuntamenetelmiä, joiden avulla selaimen sormenjälkitunnistaminen voidaan torjua käyttäjän selaimessa (Baumann ym. 2016). Ongelmana esitetyille torjuntamenetelmille on niiden puutteellisuus. Yksikään käytettävissä olevista torjuntamenetelmistä ei pysty torjumaan kaikkia tunnettuja selaimen sormenjälkitunnistamiseen käytettyjä menetelmiä (Luangmaneerote, Zaluska ja Carr 2016).

1. Panoptclicks on käytettävissä www-osoitteessa `https://panopticlick.eff.org/`
2. AmIUnique on käytettävissä www-osoitteessa `https://amiunique.org/`.

Tutkielman aiheena on selvittää, voidaanko selaimen sormenjälkitunnistamista torjua käyttöjärjestelmäavusteisen virtualisoinnin avulla. Selaimen sormenjälkitunnisteen käyttäminen käyttäjien yksilöimiseen perustuu modernien selainten välittämään tietoon käyttäjän selaimen ja suoritusympäristön kokoonpanosta. Käyttöjärjestelmäavusteisella virtualisointimenetelmillä voidaan prosessin suoritusympäristö eriyttää siten, ettei prosessilla ole pääsyä käyttöjärjestelmän varsinaisen suoritusympäristön resursseihin. Tutkielmassa testataan oletusta, voidaanko selaimen suoritusympäristön eriyttämisellä estää selaimen sormenjälkitunnisteen yksilöitävyys piilottamalla käyttöjärjestelmän kokoonpanotietoja käyttöjärjestelmäavusteisten virtualisointiteknologioiden avulla.

Luvussa 2 kerrotaan mitkä www-selainten ominaisuudet mahdollistavat yksilöllisen sormenjälkitunnisteen luomisen. Luvussa 3 määritetään selaimen sormenjälkitunnistamisen torjuntamallit, kuvataan kolmen eri aiemmin esitetyn torjuntamenetelmän toimintaperiaatteet ja johdetaan neljä eri torjunnan asettamaa haastetta. Luvussa 4 pohditaan mahdollisia ratkaisuja selaimen sormenjälkitunnistamisen estämiselle. Luvussa 5 kuvataan käyttöjärjestelmäavusteinen virtualisoinnin periaate. Luvussa 6 esitellään Linux-kernelin tukemat ominaisuudet käyttöjärjestelmäavusteisen virtualisoinnin toteuttamiseen. Luvussa 7 testataan käyttöjärjestelmäavusteisen virtualisoinnin käyttämistä selaimen sormenjälkitunnistamisen torjunnassa ja analysoidaan toteutetun testin tulokset.

2 Käyttäjän seuranta selaimen sormenjälkitunnisteella

Käyttäjän tunnistaminen on olennainen mekanismi nykyisiä www-palveluja, jotka tarjoavat käyttäjilleen henkilökohtaisia ja samalla tietoturvallisia palveluja. Käyttäjän todentaminen toteutetaan usein käyttäjän antamalla vahvistuksella omasta henkilöllisyydestään (Blakemore, Redol ja Correia 2016). Käyttäjän todentamiseen voidaan käyttää eri tunnistautumismenetelmiä, kuten käyttämällä käyttäjän tietämää käyttäjätunnusta ja salasana, käyttäjän omaavaa biometrisistä tunnistetta tai käyttäjän hallussa pitämää turvallisuustunnistetta (Klaper 2015). Edellä mainituille tunnistautumismenetelmille yhteistä on, että käyttäjän tunnistaminen tapahtuu käyttäjän toimesta. Käyttäjän tunnistamista on mahdollista toteuttaa myös epäsuorasti ilman, että käyttäjä on aktiivisena toimijana tunnistamisprosessissa (Blakemore, Redol ja Correia 2016).

Torres, Jonker ja Mauw (2015) luokittelevat verkkokäyttäjien seurannan kahden eri tyyppiseen. Ensimmäistä seurannan tyyppiä he pitävät www-palvelun sisäisenä seurantana, joka kohdistuu yksittäisen www-palvelun omien käyttäjien seurantaan. Tällä seurannan muodolla he katsovat olevan hyötyä www-palvelulle ja sen käyttäjille. Sisäistä käyttäjien seurantaa voidaan käyttää www-palvelun tietoturvan parantamiseen havaitsemalla väärinkäytöksiä käyttäjien toimissa. Liu ja Wang (2016) ovat osoittaneet, että sormenjälkitunnistetta voidaan käyttää anonyymiutta vahvistavia menetelmiä käyttävien verkkohyökkääjien jäljittämiseen. Unger ym. (2013) ovet selvittäneet sormenjälkitunnisteen käyttämistä HTTP-istunnon kaappauksen havaitsemisesssa.

Toista seurannan muotoa Torres, Jonker ja Mauw (2015) pitävät kolmannen osapuolen teettämänä seurantana, joka mahdollistaa verkkokäyttäjien laajamittaisen seurannan. Sen avulla sama käyttäjä voidaan tunnistaa usean eri verkkopalvelun välillä (Nikiforakis ym. 2013). Useiden kaupallisten palveluiden on havaittu tarjoavan palveluja käyttäjien laajamittaiseen seurantaan (Nikiforakis ym. 2013; Acar ym. 2013). Laajamittainen käyttäjienseuranta mahdollistaa käyttäjien verkkokäyttäytymisen profiloinnin, joka on käyttäjien yksityisyydensuojaa loukkaavaa. Seuraavassa luvussa kuvataan käyttäjien epäsuoratunnistaminen, jota voidaan käyttää käyttäjien laajamittaiseen seurantaan.

2.1 Epäsuora käyttäjän tunnistaminen

Epäsuora käyttäjän tunnistaminen voidaan toteuttaa tilallisesti tallentamalla yksilöllinen tunniste käyttäjän laitteelle tai tilattomasti keräämällä käyttäjän laitteesta sellaista tietoa, joka riittää laitteen yksilöimiseen (Blakemore, Redol ja Correia 2016). Käyttäjän epäsuoraa tunnistamista voidaan käyttää käyttäjän henkilöllisyyden vahvistuksen tukena tai väärinkäytösten havaitsemisessa (Blakemore, Redol ja Correia 2016). Epäsuoraa tunnistamista käytetään kuitenkin myös käyttäjän yksityisyyttä loukkaavasti käyttäjien toimien seuraamiseen (Nikiforakis ym. 2013).

Yleinen menetelmä verkkokäyttäjien epäsuoraan tunnistamiseen on käyttää selaimen evästeitä (Nikiforakis ym. 2013). Alkujaan evästeet kehitettiin tilallisten istuntojen toteuttamiseen muutoin tilattoman HTTP-protokolla päälle (Kristol ja Montulli 1997). Tilallisten www-palvelujen toteuttamisen lisäksi, evästeitä, erityisesti kolmannenosapuolen asettamia evästeitä, voidaan käyttää käyttäjien seurantaan (Nikiforakis ym. 2013). Seurantaevästeiden avulla käyttäjän laitteelle voidaan tallettaa yksilöllinen tunniste, jolla käyttäjä voidaan tunnistaa (Broenink 2012).

Evästeiden käyttöä käyttäjien pysyvänä tunnisteena estää se, että käyttäjät voivat itse poistaa evästeitä ja vaikuttaa niiden luomiseen (Nikiforakis ym. 2013). Lisäksi käyttäjä voi käyttää selainten sisältämää yksityisyyttä parantavaa selaustilaa, jossa evästeiden tallentamista on rajoitettu (Zhao ja Liu 2015). Nikiforakis ym. (2013) arvioivat, että evästeiden heikko säilyvyys on johtanut muiden käyttäjien tunnistamis- ja seurantamenetelmien kehittämiseen.

Käyttäjän epäsuora tunnistaminen käyttäjän laitteen avulla kutsutaan laitteen sormenjälkitunnistamiseksi (Eckersley 2010). Laitteen sormenjälkitunnistamista voidaan toteuttaa verkon yli ja selaimen välityksellä (Acar ym. 2013). Menetelmä ei vaadi tiedon tallentamista käyttäjän laitteelle (Boda ym. 2012), ja se voidaan tehdä tilattomasti (Acar ym. 2013). Sormenjälkitunnisteen luominen ei vaadi muutoksia käyttäjän selaimeen (Broenink 2012) eikä käyttäjän suostumusta (Saito ym. 2016). Sormenjälkitunnisteen muodostaminen selaimen välityksellä perustuu selaimen välittämään tietoon käyttäjän selaimen ja suoritusympäristön kokoonpanosta (Laperdrix, Rudametkin ja Baudry 2016).

2.2 Laitetunnistaminen selaimen välityksellä

Laitetunnistamisen mahdollisuutta laitetta yksilöivän tiedon perusteella on selvitetty eri tutkimusten osalta. Kohno, Broido ja Claffy (2005) osoittivat, kuinka laitteen yksilöivän tunnisteen luomiseen voidaan käyttää laitteen kellon ajan poikkeamaa. Lukas, Fridrich ja Goljan (2006) tutkivat digitaalikameroiden tunnistamista käyttämällä tunnisteena kameran kennon aiheuttamaa kohinaa. Eckersley (2010) mittasi laitetunnistamisen mahdollisuutta käyttämällä tunnisteena selaimen välittämää tietoa käyttäjän kokoonpanosta. Tutkimus toteutettiin avoimena käyttäjätestinä, jossa testikäyttäjät pystyivät mittaamaan oman selaimensa yksilöitävyyden.

Eckersley (2010) tekemä tutkimus osoitti selainten olevan alttiita laitetunnistamiselle. Testissä mitatut tiedot käyttäjien selaimien välityksellä olivat riittäviä käyttäjiä yksilöivien sormenjälkitunnisteiden muodostamiseen. Kolme vuotta tutkimuksen jälkeen Acar ym. (2013) esittivät selaimen sormenjälkitunnistamisen olevan ”todellinen ja kasvava ongelma, joka ansaitsee päättäjien ja tutkimusyhteisön huomion”. He mittasivat kehittämällään menetelmällä tunnettujen sormenjälkitunnistemenetelmien käytön levinneisyyttä suosittujen www-sivustojen joukossa.

Www-selaimet pyrkivät turvaamaan käyttäjien yksityisyyttä. Useisiin www-selaimiin on tullut selaustila, joka tarjoaa käyttäjilleen parempaa yksityisyyden suojaa minimoimalla session aikaisten tietojen tallentamista (Zhao ja Liu 2015). Selaimet tukevat myös DNT-otsaketietueen käyttämistä. DNT-otsaketietueen avulla, käyttäjä voi välittää tiedon www-palveluille, ettei hän halua tulla seuratuksi.

Selainten yksityisen selaustilan ja DNT-otsaketietueen tehokkuutta käyttäjien yksityisyydensuojan turvaamisena on kuitenkin kritisoitu. Acar ym. (2013) mukaan selainten yksityinen selaustila ei estä sormenjälkitunnisteen luomista. Lisäksi DNT-otsaketietueen valinnaisuudesta johtuen useat www-palvelut eivät kunnioita sen käyttämistä (Acar ym. 2013). DNT-otsaketietueen sisältämää arvoa voidaan käyttää myös käyttäjää yksilöivänä tietona sormenjälkitunnisteen luomisessa, vaikka sen entropia on pieni (Laperdrix, Rudametkin ja Baudry 2016).

EU:n sähköisen viestinnän tietosuojadirektiivi edellyttää käyttäjien tietoisuutta ja hyväk-

syntää käyttäjien toimien seuraamiselle sähköisen viestinnän alalla (European Communities 2002, ks. pykälä 25). Acar ym. (2013) ovat kuitenkin osoittaneet, kuinka useat suositut www-palvelut käyttävät käyttäjien seurannan mahdollistavia laitteen sormenjälkitunnistamismenetelmiä, mutta eivät ilmoita niiden käytöstä.

Seuraavassa luvussa on lueteltu tekijät, jotka vaikuttavat sormenjälkitunnisteen luomiseen. Sen jälkeen kuvataan selaimen välityksellä saatavilla olevat käyttäjää yksilöivät tiedot ja niiden lähteet. Saatavilla olevat käyttäjää yksilöivät tiedot on kuvattu tarkemmin tiedon lähdettä kuvaavassa alaluvussa.

2.3 Sormenjälkitunnisteeseen vaikuttavat tekijät

Käyttäjän erottaminen muista käyttäjistä edellyttää käyttäjää yksilöivältä tiedolta riittävän suurta informaation määrää (Eckersley 2010). Informaation määrää voidaan ilmaista määrällisesti entropian avulla (Laperdrix, Rudametkin ja Baudry 2016). Entropiaa voi pitää suureena, joka kuvaa alarajan bittien lukumäärälle, joka tarvitaan muuttujan sisältämän informaation tallentamiseen (Vajapeyam 2014). Entropian voi laskea diskreetille satunnaismuuttujalle laskemalla summan muuttujan saamien arvojen todennäköisyyksien ilmaisemiseen tarvittavien bittien lukumäärille (Vajapeyam 2014). Satunnaismuuttujan X entropia H saadaan Shannonin entropian laskukaavalla

$$H(X) = -\sum_{i=1}^{n} P(X_i) \log_2 P(X_i),$$

missä X:n saa yksittäisiä arvoja $x_1, x_2, ..., x_n$ ja $P(X)$ on arvojen todennäköisyysfunktio (Laperdrix, Rudametkin ja Baudry 2016). Jotta entropia on vertailukelpoinen muiden otoskokojen suhteen, se voidaan normalisoida kaavalla

$$\frac{H(X)}{\log_2(N)},$$

missä N on otoksen koko (Laperdrix, Rudametkin ja Baudry 2016).

Eckersley (2010) sai yksilöitävyyden asteeksi 84,3 %, kun käyttäjien selaimista kerätyn tie-

don entropia oli 18,1 bittiä. Eckersley (2010) arvioi yleisesti käyttäjien yksilöitävyyteen riittävän 15-20 bitin entropia.

Tiedon soveltuvuutta sormenjälkitunnisteen luomiseen voidaan arvioida entropian lisäksi tiedon muuttumattomuudella (Broenink 2012). Mikäli sormenjälkitunnisteen halutaan pysyvän vakaana, tulee kerättyjen tietojen säilyä samoina (Broenink 2012). Muuttujat, kuten dynaaminen IP-osoite ja paikkakoordinaatit, tekevät sormenjälkitunnisteesta epävakaan, mikä vaikeuttaa käyttäjän tunnistamista pitemmällä aikavälillä (Blakemore, Redol ja Correia 2016).

Sormenjälkitunnisteen muuttumista voi kiertää joko käyttämällä vakaina pysyviä tietoja, joiden voidaan olettaa pysyvän samoina pitemmän aikavälin tai tunnistamalla muutokset sormenjälkitunnisteissa (Broenink 2012). Muutoksia sormenjälkitunnisteessa voidaan jäljittää heuristisella päättelyllä, mutta tämä voi johtaa vääriin tuloksiin (Eckersley 2010; Broenink 2012).

2.4 Selaimen välityksellä saatavat käyttäjää yksilöivät tiedot

Käyttäjän selaimesta ja suoritusympäristöstä saatava tieto, kuten näytön koko ja kieli, mahdollistavat www-palvelun käyttäjäkohtaisen personoinnin ja käytettävyyden parantamisen. Samalla tietoa käyttäjän kokoonpanosta voidaan käyttää selaimen sormenjälkitunnisteen luomiseen (Broenink 2012). Sormenjälkitunnisteeseen muodostamiseen tarvitaan riittävä määrä käyttäjää yksilöivää tietoa, jotta tunniste on yksilöllinen ja jotta käyttäjä on mahdollista erottaa muista käyttäjistä (Eckersley 2010).

Selaimen välityksellä saatavilla olevat käyttäjää yksilöivät tiedot voidaan jakaa eri kategorioihin (Nikiforakis ym. 2013). Kategoriat ja niiden sisältämiä käyttäjää yksilöiviä tietoja on lueteltu taulukossa 1. Taulukko ei sisällä kaikkia mahdollisia selaimen välityksellä saatavilla olevien tietoja. Se auttaa hahmottamaan kokonaiskuvaa siitä, mitä tietoa www-palvelulla on saatavilla käyttäjistä.

Www-palvelu voi kerätä tietoa käyttäjästä selaimen välityksellä joko passiivisesti tai aktiivisesti (Broenink 2012). Passiivinen tiedonkeräys perustuu yhteyden väliseen tiedonvälitykseen www-palvelun ja käyttäjän selaimen välillä (Broenink 2012). Tiedonsiirtoproto-

Kategoria	**Saatavilla olevia tietoja**
Selaimen kokoonpanotiedot	Asennetut liitännäiset, selaimen tukemat tiedostomuodot.
Selaimen asetukset	Käyttäjän aikavyöhyke, käyttäjän kieli, DNT-otsaketietueen arvo, liitännäinen käytössä, evästeet käytössä, välityspalvelimen asetukset.
Selaimen tiedot	Selaimen alusta ja versio.
Käyttöjärjestelmän kokoonpanotiedot	Käyttöjärjestelmän alusta ja versio, asennetut fontit.
Laitteiston kokoonpanotiedot	Näytön resoluutio, laitteistoajurit, laitteistomuuttujat.
Verkon kokoonpanotiedot	IP-osoite, TCP/IP-parametrit.

Taulukko 1. Selaimen välityksellä saatavien käyttäjää yksilöivien tietojen kategoriat (Broenink 2012).

kollana selaimen ja www-palvelun välillä käytetään HTTP-protokollaa, jossa selain lähettää HTTP-asiakkaana pyyntöjä www-palvelun HTTP-palvelimelle. Passiivisesti tietoa voidaan kerätä HTTP-pyynnön sisältämistä otsaketietueista (Torres, Jonker ja Mauw 2015). HTTP-otsaketietueiden sisältämää käyttäjää yksilöivää tietoa on eritelty tarkemmin alaluvussa 2.4.3.

Passiivisesti tietoa voidaan kerätä muutoinkin kuin HTTP-protokollan sisältämistä otsaketietueista. Kohno, Broido ja Claffy (2005) käyttävät laitteen sormenjälkitunnistamiseen kuljetuskerrosta ajan poikkeaman mittaamiseen keräämällä tietoa TCP-protokollan otsaketietueista. Mahdollisia muita TCP/IP-protokollapinon sisältämiä tiedonlähteitä passiiviseen tiedon keräykseen on tarkasteltu TCP/IP-viitemallin avulla taulukossa 2. TCP/IP-viitemalli jakaa

TCP/IP-viitemallin kerrokset	Saatavilla olevat tiedot
Sovelluskerros	HTTP-pyynnön sisältämä otsaketietueet, kuten käyttäjän suoritusympäristön kokoonpanoa kuvaava User-Agent -otsaketietue.
Kuljetuskerros	TCP-pakettien otsaketietueet, kuten laitteiston kellonaika Timestamp-otsaketietueesta.
Verkkokerros	IP-paketin otsaketietueet, kuten käyttäjän IP-osoite.
Verkkoyhteyskerros	ARP-protokolla välitetyt tiedot, kuten käyttäjän MAC-osoite Ethernet-verkossa.

Taulukko 2. Passiivinen tiedonkeräys TCP/IP-viitemallilla (Braden 1989) tarkasteltuna.

TCP/IP-protokollapinon kerroksittain, missä alemman kerroksen protokolla paketoi ylemmän kerroksen tiedon (Braden 1989).

Aktiivinen tiedonkerääminen toteutetaan passiivisesta poiketen asiakaspuolella www-palvelun suorituksen aikana pyytämällä kerättäviä tietoja käyttäjän selaimelta (Broenink 2012). Tietojen pyytäminen toteutetaan asiakaspuolella ajettavien skriptien avulla (Torres, Jonker ja Mauw 2015). Asiakaspuolen tiedonkeräysskriptit voidaan toteuttaa selainten yleisesti tukemalla JavaScripti-ohjelmointikielellä, jolloin tiedonlähteinä toimii selaimen JavaScript-suoritusympäristö. Selaimen JavaScript-suoritusympäristö tarjoaa joukon ohjelmointirajapintoja, jotka välittävät tietoa käyttäjän selaimen ja suoritusympäristön kokoonpanosta (Laperdrix, Rudametkin ja Baudry 2016).

Selainten JavaScript-ohjelmointirajapintojen lisäksi asiakaspuolen skriptien käytössä on selaimen liitännäisten tarjoamat rajapinnat, jotka lisäävät yksilöivän tiedonlähteitä käyttäjän suoritusympäristöön ja kokoonpanoon tai lisäävät selainten sisältämien rajapintojen ilmoittavien tietojen tarkkuutta (Laperdrix, Rudametkin ja Baudry 2016). Selaimen ohjelmointirajapintoja on tarkasteltu tarkemmin alaluvussa 2.4.1 ja selaimen liitännäisiä alaluvussa 2.4.2.

2.4.1 Selaimen JavaScript-ohjelmointirajapinnat

Yksilöllisen sormenjälkitunnisteen luomisen kannalta on olennaista löytää muuttujia, joiden arvot erottavat käyttäjän muista www-palvelun käyttäjistä (Acar ym. 2013). Selaimet tarjoavat lukuisia eri ohjelmointirajapintoja www-sovellusten toteuttamiseen. Mozilla-organisaation ylläpitämä ohjelmistokehittäjille suunnattu dokumentaatio selainten tarjoamista rajapinnoista sisältää noin 700 eri ohjelmointirajapintoja käsittelevää dokumentaatiota.[1] Selainten sisältämien ohjelmointirajapintojen tarjoamia muuttujia, jotka antavat tietoa käyttäjän kokoonpanosta, voidaan käyttää yksilöllisen sormenjälkitunnisteen luomiseen (Acar ym. 2013).

Laperdrix, Rudametkin ja Baudry (2016) selvittivät 17 eri käyttäjää yksilöivän muuttujan käyttämistä sormenjälkitunnisteen luomiseen. Tutkimuksessa hyödynnettiin hiljattain havaittuja selaimen laitteistotunnistamiseen soveltuvia menetelmiä, jotka perustuvat selainten Canvas- ja WebGL-ohjelmointirajapintojen käyttöön. Testissä kerätyistä tunnisteista yksitoista oli kerättävissä selainten JavaScript- ohjelmointirajapintojen kautta, viisi HTTP-pyynnön otsaketietueista ja yksi selaimen Flash-liitännäisen kautta. Testissä käytetyt selainten JavaScript-ohjelmointirajapintojen kautta kerätyt muuttujat ja Laperdrix, Rudametkin ja Baudry (2016) mittaamat muuttujien sisältämien tietojen normalisoidut entropiat on kuvattu taulukossa 3. Taulukkoon on lisäksi liitetty muuttujien ohjelmointirajapinnat, josta muuttuja on saatavilla. Tieto muuttujien lähdeohjelmointirajapinnoista perustuu Mozilla-organisaation ylläpitämään ohjelmointirajapintojen dokumentaatioon.

2.4.2 Selaimen liitännäiset

Sormenjälkitunnisteen yksilöllisyyttä voidaan kasvattaa hyödyntämällä selaimen liitännäisiä. Selaimen liitännäiset avaavat uusia tiedonlähteitä käyttäjän suoritusympäristöön tai lisäävät selainten JavaScript-ohjelmointirajapintojen ilmoittavien tietojen tarkkuutta (Laperdrix, Rudametkin ja Baudry 2016). Eckersley (2010) hyödynsi Flash- ja Java-liitännäisiä käyttäjän laitteelle asennettujen fonttien selvittämiseen. Asennettujen fonttien listauksen entropia oli testin toiseksi suurin, mikä nosti käyttäjien yksilöitävyyttä. Laperdrix, Rudametkin ja

1. Dokumentaatio selaimien ohjelmointirajapinnoista on saatavilla www-osoitteessa `https://developer.mozilla.org/en-US/docs/Web/API`

Muuttuja	**Ohjelmointirajapinta**	**Entropia**
Aikavyöhyke	Date	0,198
Mainostenesto käytössä	Document	0,059
Canvas-elementin renderöintitiedot	HTMLCanvasElement	0,491
DNT-käyttäjäasetus	Navigator	0,056
Käyttöjärjestelmän alusta	Navigator	0,137
Evästeet sallittu	Navigator	0,015
Lista selaimen-liitännäisistä	NavigatorPlugins	0,656
Näytön resoluutio ja värisyvyys	Screen	0,290
Local-storage sallittu	Web Storage	0,024
Session-storage sallittu	Web Storage	0,024
Näytönohjaimen valmistaja	WebGL	0,127
Näytönohjaimen malli	WebGL	0,202

Taulukko 3. JavaScript-ohjelmointirajapintojen sisältämät tiedot selaimen sormenjälkitunnistamiseen (Laperdrix, Rudametkin ja Baudry 2016).

Baudry (2016) mittasivat asennettujen fonttien listauksen normalisoiduksi entropiaksi 0,497. Normalisoitu entropia oli 0,548, kun mukaan ei lasketa mobiilikäyttäjiä, joilla fonttien listausta ei ollut saatavilla Flash-liitännäisen tuen puuttumisen vuoksi. Tämä oli testin kolmanneksi merkittävin muuttuja, kun vertaillaan muuttujien entropioiden suuruutta.

Laitetunnistaminen selaimen liitännäisiä käyttämällä vaatii, että käyttäjä on asentanut kyseisen liitännäisen ja sallinut sen käytön sivustolla. Liitännäisten käyttäminen sormenjälkitunnisteen luomiseen edellyttää lisäksi käyttöjärjestelmän sekä selaimen tukea liitännäiselle. Flash- ja Java-liitännäiset eivät ole tuettuja useimmissa mobiiliselaimissa (Laperdrix, Rudametkin ja Baudry 2016). Liitännäisten käyttö on myös vähentynyt, johon on osaltaan vaikuttanut perinteisen liitännäisten mahdollistavan NPAPI-rajapinnan tuen poistuminen sekä Chrome- että Firefox-selaimista (Smedberg 2015).[2]

2. Tutkielman kirjoittamisen hetkellä Firefox-selaimen NPAPI-rajapinnan tuki päättyi selaimen version 52 julkaisun myötä. Version muutosloki on katsottavissa www-osoitteesta `https://www.mozilla.org/firefox/52.0/releasenotes/`.

HTTP-otsaketietue	Entropia	Kuvaus
User-Agent	0,580	Sisältää selaimen alustan ja version tiedot, sekä käyttöjärjestelmäkohtaista tietoa.
Accept	0,082	Ilmoittaa selaimen tukemat tietotyypit.
Accept-Language	0,351	Ilmoittaa selaimen tukeman kielen.
Accept-Encoding	0,091	Ilmoittaa selaimen tukemat enkoodauksen muodot.
Lista HTTP-otsaketietueista	0,249	Kuvaa HTTP-pyynnön sisältämät otsaketietueiden nimet ja otsaketietueiden keskinäisen järjestyksen.

Taulukko 4. HTTP-otsaketietueiden sisältämät tiedot selaimen sormenjälkitunnistamiseen (Laperdrix, Rudametkin ja Baudry 2016).

Vaikka liitännäisiä ei olisi käytettävissä, se ei estä laitteistotunnistamista selaimen välityksellä niin mobiili- kuin työpöytäselaimilla, kun käytettävissä on tarpeeksi käyttäjää yksilöivää tietoa (Laperdrix, Rudametkin ja Baudry 2016). Fifield ja Egelman (2015) ja Saito ym. (2016) ovat myös osoittaneet kuinka asennettujen fonttien listausta voidaan selvittää ilman selaimen liitännäisiä.

2.4.3 Selaimen lähettämät HTTP-otsaketietueet

HTTP-asiakas voi lähettää HTTP-pyyntöjen mukana lisätietoja pyynnöstä ja itsestään HTTP-pyynnön otsaketietueiden mukana HTTP-palvelulle (Fielding ym. 1999). Selaimen lähettämiä tietoja HTTP-pyynnön mukana voidaan käyttää selaimen ominaisuuksien tunnistamisessa ja selainkohtaisessa vianetsinnässä (Broenink 2012).

HTTP-protokollan mukaisesti HTTP-pyyntö koostuu pyyntörivistä, joka sisältää metodin tyypin, pyydetyn resurssin polun ja käytetyn HTTP-protokollan version, yhdestä tai useammasta otsaketietueesta ja valinnaisesta viestikentästä (Fielding ym. 1999). Oletuksena selaimet lähettävät otsaketietueissa User-Agent- ja Accept-otsaketietueet, jotka kertovan selai-

men alustan ja version tiedot, sekä ilmoittavat selaimen tukemista ominaisuuksista (Broenink 2012).

Selaimen lähettämiä HTTP-otsaketietueita ja niiden sisältämiä tietoja voidaan tarkastella käyttäen Wireshark-pakettianalysaattoria [3]. Firefox-selaimen lähettämän GET-pyynnön sisältämät tiedot on nähtävissä kuviosta 1. Firefox lähettää oletuksena User-agent ja kolme eri Accept-alkuista otsaketietuetta. Tietueiden sisältämät tiedot on kuvattu taulukossa 4. Näiden lisäksi, Firefox lähettää Connection-ja Upgrade-Insecure-Requests -otsaketietueet. Connection-otsaketietueella HTTP-asiakas voi ilmoittaa lisätietoja HTTP-pyynnön yhteystyypistä (Fielding ym. 1999). Upgrade-Insecure-Requests -otsaketietueella HTTP-asiakas ilmoittaa tukevansa Upgrade-Insecure-Requests -mekanismia, jonka avulla www-palvelu voi ohjata pyynnöt ei-salaamattomat HTTP-pyynnöt käyttämään salausta (Calzavara, Rabitti ja Bugliesi 2016).

```
GET / HTTP/1.1
Host: www.jyu.fi
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:53.0)
        Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Kuvio 1. Firefox-selaimen lähettämät otsaketiedot www.jyu.fi -sivuston GET-pyynnössä.

HTTP-otsaketietueiden paljastamaa tietoa käyttäjän kokoonpanosta voidaan käyttää sormenjälkitunnisteen luomisessa käyttäjän tunnistamiseen (Broenink 2012). Laperdrix, Rudametkin ja Baudry (2016) käyttivät teettämässään testissä viittä eri HTTP-otsaketietuetta sormenjälkitunnisteen luomiseen. Tunnisteiden sisältämien tietojen lisäksi, he käyttivät yksilöllisenä tunnisteena listausta otsaketietueiden nimistä sekä tietueiden keskinäistä järjestystä. Laperdrix, Rudametkin ja Baudry (2016) käyttämät HTTP-otsaketietueet, niiden sisältämien tietojen kuvaukset, sekä heidän mittaamat tietojen normalisoidut entropiat on kuvattu taulukossa 4.

3. Wireshark on saatavilla www-osoitteesta `https://www.wireshark.org/`

3 Selaimen sormenjälkitunnistamisen torjunta

Selaimen sormenjälkitunnistamisen torjuntaan liittyy haasteita, joita tässä tutkielmassa on tunnistettu. Luvussa 7 selvitetään voidaanko luvussa 3.3 kuvattuihin haasteisiin vastata käyttämällä käyttöjärjestelmäavusteista virtualisointia selaimen sormenjälkitunnistamisen torjunnassa. Selaimeen sormenjälkitunnistamiseen liittyvät haasteet on kuvattu tarkemmin luvussa 3.3.

Luvussa 3.3 esitetyt torjunnan haasteet perustuvat aiempaan tutkimukseen selaimen sormenjälkitunnistamisen torjumiseen. Selaimen sormenjälkitunnistamiseen on esitetty eri torjuntamenetelmiä, joilla sormenjälkitunnistaminen voidaan torjua käyttäjän selaimessa. Selaimen sormenjälkitunnistamisen torjuntaan esitetyistä torjuntamenetelmistä kerrotaan luvussa 3.2.

Esitetyt torjuntamenetelmien torjuntatavat perustuvat joko tiedon yksilöitävyyden poistamiseen tai yksilöivän tiedon keräyksen rajoittamiseen (Fifield ja Egelman 2015). Selaimen sormenjälkitunnistamisen torjuntamallit on lueteltu taulukossa 5 ja kuvattu tarkemmin seuraavassa luvussa 3.1.

3.1 Torjuntamallit

Selaimen sormenjälkitunnistamisen torjunta perustuu joko tiedon yksilöitävyyden poistamiseen tai yksilöivän tiedon keräyksen rajoittamiseen (Fifield ja Egelman 2015). Tiedon yksilöitävyyttä voidaan poistaa tiedon väärentämisellä, satunnaistamisella tai tiedon entropiaa pienentämällä (Luangmaneerote, Zaluska ja Carr 2016). Tiedon väärentämisellä pyritään poistamaan tiedon yksilöllisyys ilmoittamalla todellisuudesta poikkeavaa tietoa, joka ei ole yksilöitävissä tiettyyn käyttäjään (Luangmaneerote, Zaluska ja Carr 2016). Tiedon satunnaistamisella pyritään vaikuttamaan sormenjälkitunnisteen vakauteen ilmoittamalla eri tieto eri tiedon keräyskerroilla (Baumann ym. 2016). Tiedon entropiaa voidaan pienentää rajaamalla ilmoitetun tiedon tarkkuutta (Luangmaneerote, Zaluska ja Carr 2016).

Sormenjälkitunnisteeseen tarvittavan käyttäjää yksilöivän tiedon keräämistä voidaan rajoittaa joko kokonaan tai osittain (Fifield ja Egelman 2015). Tiedon keräämisen kokonaan ra-

Tiedon yksilöitävyyden poistaminen	
1. Tiedon väärentäminen	Ilmoitetaan todellisuudesta poikkeavaa tietoa.
2. Tiedon satunnaistaminen	Ilmoitetaan eri tieto eri kerroilla.
3. Tiedon entropian pienentäminen	Vähennetään ilmoitetun tiedon tarkkuutta.
Tiedon keräämisen estäminen	
1. Keräämisen täysi rajoittaminen	Estetään selaimen ominaisuuden tai rajapinnan käyttö.
2. Keräämisen osittainen rajoittaminen	Sallitaan vain rajattu määrä kerättävää tietoa.

Taulukko 5. Selaimen sormenjälkitunnistamisen torjuntamallit (Fifield ja Egelman 2015; Luangmaneerote, Zaluska ja Carr 2016).

joittaminen estää selaimen ominaisuuden käyttämisen. Www-palvelulta voidaan esimerkiksi estää käyttämästä selaimen ohjelmointirajapintaa tai lisäosaa, joiden käyttäminen mahdollistaa käyttäjää yksilöivän tiedon keräämiseen (Baumann ym. 2016).

Osittainen tiedon keräämisen rajoittaminen ei estä www-selaimen ominaisuuden tai ohjelmointirajapinnan käyttöä, mutta rajaa www-palvelun käytettävissä olevaa tiedon määrää (Torres, Jonker ja Mauw 2015). Tämä pienentää saatavilla olevan tiedon entropiaa, mikä edesauttaa vähentämään käyttäjien yksilöitävyyttä. Www-palvelulle voidaan asettaa rajoitus, joka määrittää kuinka paljon www-palvelulla on käytettävissä ohjelmistorajapinnan tai lisäosan kautta saatavaa tietoa. Esimerkiksi www-selain voi asettaa rajoituksen www-palvelun käytettävissä olevien fonttien lukumäärälle (Baumann ym. 2016).

Seuraavassa luvussa on tarkasteltu selaimen sormenjälkitunnistamisen torjuntaan esitettyjä torjuntamenetelmiä perustuen aiempaan tutkimukseen. Esitetyt torjuntamenetelmät pohjautuvat tässä luvussa kuvattuihin selaimensormenjälkitunnistamisen torjuntatapoihin. Torjuntamenetelmissä kohdattuihin haasteisiin on syvennetty tarkemmin luvussa 3.3.

3.2 Esitetyt torjuntamenetelmät

Selaimen sormenjälkitunnistamisen torjuntaan on esitetty eri torjuntamenetelmiä, jotka mahdollistavat selaimen sormenjälkitunnistamisen torjunnan käyttäjän selaimessa. Esitetyt torjuntamenetelmät on toteutettu joko selaimen laajennoksena tai suoraan selaimen lähdekoodia muuttamalla. Torjuntamenetelmiä torjuntamallit on kuvattu taulukossa 5.

Esitetyt torjuntamenetelmät poikkeavat toisistaan toimintaperiaatteidensa osalta siten, miten ne pyrkivät estämään selaimen sormenjälkitunnisteella toteutetun käyttäjien seurannan. Seuraavissa alaluvuissa on kuvattu kolmen eri selaimen sormenjälkitunnistamisen torjuntamenetelmän toimintaperiaatteet.

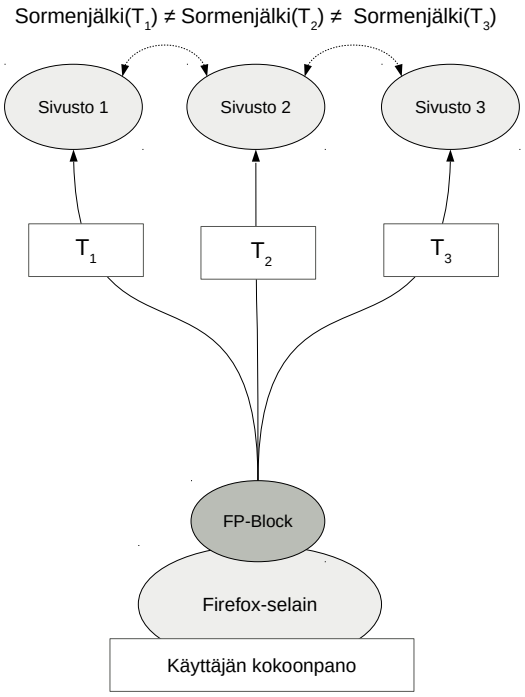
3.2.1 FP-Block

Torres, Jonker ja Mauw (2015) ovat esittäneet FP-Block nimisen Firefox-selaimen laajennoksena toteutetun torjuntamenetelmän. FP-Blockin toimintaperiaate on muuttaa selaimen sormenjälkitunniste yksilölliseksi jokaiselle käyttäjän vierailemalle sivustolle. Siten FP-Block sallii vain sivustojen sisäisen käyttäjien seurannan, mutta pyrkii estämään käyttäjien laajamittaisen seurannan sivustojen välillä. (Torres, Jonker ja Mauw 2015)

FP-Block luo jokaiselle sivustolle oman identiteetin. Identiteetti sisältää muutettuja tietoja, joita voidaan käyttää yksilöllisen sormenjälkitunnisteen luomiseen. FP-Block pitää yllä tietoa luoduista identiteeteistä, ja palauttaa identiteetin, kun käyttäjä vierailee sivustolla uudelleen. FP-Block käyttää tiedon yksilöitävyyden poistamiseen perustuvia torjuntatapoja identiteetin sisältämien tietojen manipuloimisessa. FP-Blockin toimintaperiaate on kuvattu kuviossa 2. (Torres, Jonker ja Mauw 2015)

3.2.2 Tor-selain

Käyttäjän anonyymiutta edistävä Tor-selainprojekti pyrkii yleistämään selaimien sormenjälkitunnisteen riippumattomaksi selaimen alustasta ja käyttäjän kokoonpanosta (Torres, Jonker ja Mauw 2015). Tor-selaimen toimintaperiaate on, ettei yksittäistä Tor-selaimen käyttäjää tulisi pystyä erottamaan muista Tor-selaimen käyttäjistä selaimen sormenjälkitunnisteen avulla

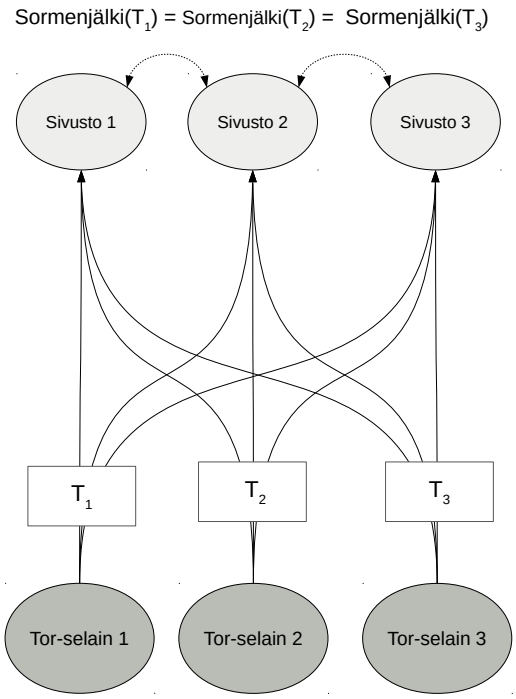


Kuvio 2. FP-Block -torjuntamenetelmän toimintaperiaate, missä T_i sisältää FP-Blockin sivustolle i tallettaman identiteetin sisältämät tunnistetiedot (Torres, Jonker ja Mauw 2015).

(Nikiforakis, Joosen ja Livshits 2015). Tor-selain käyttää sekä yksilöivän tiedon keräyksen rajoittamiseen että tiedon yksilöitävyyden poistamiseen perustuvia torjuntatapoja (Baumann ym. 2016). Tor-selaimen toimintaperiaate on kuvattu kuviossa 3.

Baumann ym. (2016) pitävät niin FP-Blockin kuin Tor-selaimen torjuntamenetelmiä puutteellisina. Tor-selain käyttää torjuntatapoinaan tiedon keräämisen rajoittamista. Baumann ym. (2016) pitävät tätä puutteena, koska se rajoittaa selaimen käytettävyyttä tai estää ominaisuuden käyttämisen. He lisäksi katsovat satunnaistamiseen perustuvien torjuntatapojen käyttäminen tekevän torjuntamenetelmän käytöstä havaittavan, mikäli satunnaistaminen toteutetaan epärealistisilla arvoilla (Baumann ym. 2016).

Baumann ym. (2016) pystyivät osittain myös kiertämään Tor-selaimen torjuntamenetelmän, johtuen sen puutteellisesta toteutuksesta. Tor-selain asettaa sivustokohtaisen rajoituksen asennettujen fonttien selvittämiseen (Baumann ym. 2016). He kiersivät sivustokohtaisen rajoituksen luomalla sivustolle dynaamisesti useita sivun sisäisiä alasivuja.

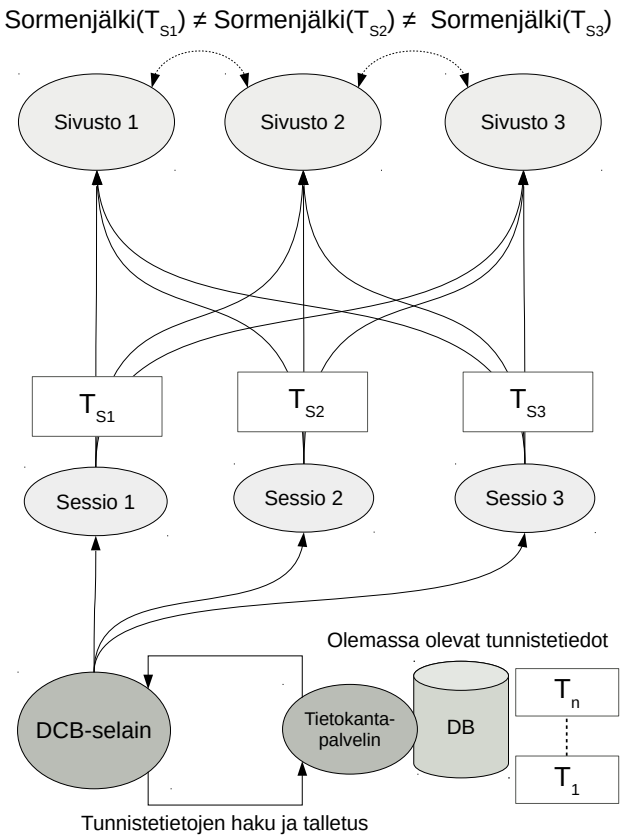


Kuvio 3. Tor-selaimen torjuntamenetelmän toimintaperiaate, missä T_i on Tor-selaimen i ilmoittamat tunnistetiedot (Nikiforakis, Joosen ja Livshits 2015).

3.2.3 DCB-selain

Baumann ym. (2016) pitävät esitettyjä torjuntamenetelmiä yleisesti puutteellisina. Olennaisina puutteina he pitävät torjuntamenetelmissä sitä, että niiden käyttäminen rajoittaa selaimen käytettävyyttä ja että niiden käyttö on havaittavissa. Kartoittamiensa puutteiden osalta, Baumann ym. (2016) esittivät oman selaimen sormenjälkitunnistamisen torjuntamenetelmän ”Disguised Chromium Browser” (DCB).

DCB on toteutettu suoraan selaimen lähdekoodia muuttamalla käyttäen torjuntamenetelmän toteutukseen avoimen lähdekoodin Chromium-selainta. DCB:n toimintaperiaate on muuttaa selaimen sormenjälkitunniste selaimen session aikaiseksi. DCB käyttää tiedon yksilöitävyyden poistamisessa tiedon satunnaistamisen sijaan tietoja olemassa olevista selainkokoonpanoista, joiden talletuksessa ja haussa käytetään erillistä tietokantapalvelinta. DCB:n toimintaperiaate on kuvattua kuviossa 4. (Baumann ym. 2016)



Kuvio 4. DCB-selaimen -torjuntamenetelmän toimintaperiaate, missä T_{Si} sisältää sessiolle i alustetut tunnistetiedot (Baumann ym. 2016).

3.3 Torjunnan haasteet

Luvussa kuvataan selaimen sormenjälkitunnistamisen torjuntaan liittyviä haasteita. Haasteet on johdettu aiemmasta tutkimuksesta selaimen sormenjälkitunnistamisen torjuntaan, sekä muissa tutkimuksissa esitetyistä torjuntamenetelmistä. Luvussa 7 selvitetään, voidaanko luvussa kuvattuihin haasteisiin vastata käyttäen torjuntamenetelmän toteutuksessa käyttöjärjestelmäavusteista virtualisointia.

3.3.1 Toteutusympäristön asettamat rajoitteet

Esitetyt torjuntamenetelmät on toteutettu joko selaimen laajennoksina tai erillisinä selainversioina. Molemmissa toteutustavoissa torjuntamenetelmän käyttö rajautuu valitun toteu-

tusympäristön mukaan. Esimerkiksi luvussa 3.2.1 kuvattu FP-Block -torjuntamenetelmä, on kehitetty Firefox-selaimen laajennoksena. Tämä tekee toteutuksen riippuvaiseksi Firefox-selaimen laajennosten toteutusympäristöstä. Jotta torjuntamenetelmä olisi käytettävissä muissa kuin Firefox-selaimessa, tulee laajennos ensiksi kääntää toisen selaimen tukemalle laajennosten toteutusympäristölle.

Toinen toteutusympäristön asettama rajoite selaimen laajennoksina toteutetuille torjuntamenetelmille on, etteivät ne voi suoraan muuttaa selaimen sisäistä toteutusta (Baumann ym. 2016). Sen sijaan niiden toiminta perustuu selaimen JavaScript-suoritusympäristön suorituksenaikaiseen manipulointiin (Baumann ym. 2016). Laajennos ylikirjoittaa selaimen JavaScript-ohjelmointirajapintojen metodeja ja muuttujia www-sivuston suorituksenaikana (Nikiforakis, Joosen ja Livshits 2015). Esimerkiksi taulukossa 3 kuvatun näytön resoluution tiedot voidaan muuttaa manipuloimalla Screen-rajapinnan prototyyppiä korvaamalla muuttujien aksessorit seuraavasti.

```
Object.defineProperty(screen.__proto__, "width", {
    get: function() {return 1920;}
});
Object.defineProperty(screen.__proto__, "height", {
    get: function() {return 1080;}
});
```

Suoraan selaimen lähdekoodia muokkaamalla vältytään suorituksen aikaiselta JavaScript-ohjelmointirajapintojen manipuloinnilta (Baumann ym. 2016). Varsinainen torjuntamenetelmän toteutus voidaan tehdä muuttamalla suoraan selaimen JavaScript-ohjelmointirajapintojen toteutusta (Nikiforakis, Joosen ja Livshits 2015). Kuten selaimen laajennoksiin perustuvia torjuntamenetelmiä, rajoittaa erilliseen selainversioon perustuvia torjuntamenetelmiä niiden toteuttamiseen valittu toteutusympäristö. Torjuntamenetelmän toteutusta ei voida suoraan siirtää toiseen selainympäristöön, vaan se edellyttää toteutuksen uudelleen toteuttamisen toisen selaimen lähdekoodille.

3.3.2 Torjunnan aiheuttamat ristiriidat

Torjuntamenetelmien käyttäminen selaimen sormenjälkitunnistamisen torjuntaan voi johtaa ristiriitaan käyttäjän tunnistamisen estämisessä (Eckersley 2010). Päinvastoin käyttäjän olettamusta, torjuntamenetelmän käyttäminen voi lisätä käyttäjän yksilöitävyyttä (Torres, Jonker ja Mauw 2015). Tämä paradoksi toteutuu, mikäli torjuntamenetelmää käyttäviä käyttäjiä on vähän suhteessa muihin käyttäjiin (Broenink 2012). Tietoa käytetystä torjuntamenetelmästä ja torjuntatavoista voidaan käyttää käyttäjän erottamiseen muista käyttäjistä (Broenink 2012). Esimerkiksi luvussa 3.2.2 Tor-selaimen käyttämä asennettujen fonttien osittainen rajoittaminen voi toimia yksilöllisenä tietona muiden selaimien käyttäjien suhteen.

Torjuntamenetelmän käyttäminen voi lisätä käyttäjän yksilöllisyyttä myös, mikäli sen käyttäminen aiheuttaa epäjohdonmukaisuutta käyttäjästä saatavissa olevissa tiedoissa (Luangmaneerote, Zaluska ja Carr 2016). Tiedon väärentämistä käyttävät torjuntamenetelmät ilmoittavat todellisuudesta poikkeavaa tietoa. Mikäli alkuperäinen tieto on saatavilla toista kautta tai muutettu tieto on ristiriidassa muiden saatavilla olevien tietojen kanssa, on käyttäjä yksilöllisempi kuin, mitä se olisi ilman torjuntamenetelmän käyttämistä (Luangmaneerote, Zaluska ja Carr 2016).

3.3.3 Torjunnan puutteellisuus

Selaimen sormenjälkitunnistamisen torjuntaan esitettyjä torjuntamenetelmiä voi pitää yleisesti puutteellisina (Baumann ym. 2016). Esitetyt torjuntamenetelmät vastaavat tiettyyn tunnettuun sormenjälkitunnistamiseen käytetyn menetelmän torjuntaan. Yksikään käytettävissä olevista torjuntamenetelmistä ei kykene torjumaan kaikkia tunnettuja selaimen sormenjälkitunnistukseen käytettyjä menetelmiä (Luangmaneerote, Zaluska ja Carr 2016). Sen lisäksi uusia menetelmiä havaitaan, joihin ei ole esitetty torjuntamenetelmiä (Englehardt ja Narayanan 2016).

Uutena selaimen sormenjälkitunnistamismenetelmänä, Englehardt ja Narayanan (2016) havaitsivat AudioContext-ohjelmointirajapinnan käytön selvittäessään selaimen sormenjälkitunnistamisen käytön levinneisyyttä. He arvioivat rajapinnan mahdollistavan uuden käyttäjää yksilöivän tiedon lähteen. AudioContext-rajapintaa voidaan käyttää audiosignaalin aallonpi-

tuuden mittaamiseen (Englehardt ja Narayanan 2016). Koska rajapinnan ilmoittama tieto pysyy samana, mutta tuottaa eroja muilla selaimilla ja eri laitekokoonpanoilla, voidaan rajapinnan ilmoittavaa tietoa käyttää käyttäjän yksilöimiseen (Englehardt ja Narayanan 2016). Tiettävästi yksikään esitetyistä selaimen sormenjälkitunnistamisen torjuntaan esitetyistä menetelmistä ei kykene torjumaan äänisignaalin käyttämistä sormenjälkitunnisteen muodostamisessa.

3.3.4 Torjunnasta aiheutuvat haittavaikutukset

Torjuntamenetelmien haasteena on olla vaikuttamatta selaimen ominaisuuksiin tai selaamisen käyttäjäkokemukseen. Väärien tietojen ilmoittaminen ja tiedon keräämisen estäminen voi aiheuttaa häiriöitä sivustojen toiminnallisuudessa tai aiheuttaa ongelmia sivustojen esittämisessä (Luangmaneerote, Zaluska ja Carr 2016). Torjuntamenetelmien aiheuttamia haittavaikutuksia voidaan arvioida neljällä eri kriteerillä, joita ovat

1. häiriöt sivuston esittämisessä,
2. häiriöt sivuston toiminnallisuudessa,
3. käyttäjäkokemukseen vaikuttavat tekijät ja
4. häiriöt sivustoon sisäänkirjautumisessa (Luangmaneerote, Zaluska ja Carr 2016).

Haittavaikutuksia aiheuttavat eniten torjuntatavat, jotka perustuvat tiedon keräämisen estämiseen (Luangmaneerote, Zaluska ja Carr 2016). Esimerkiksi luvussa 3.2.2 kuvattu Torselain käyttää tiedon keräämisen rajoittavia torjuntatapoja, jotka estävät rajapinnan tai lisäosan käyttämisen www-palvelulta. Tämä johtaa siihen, ettei kyseiset toiminnallisuudet ole käytettävissä www-sivustolla, mikä voi aiheuttaa edellä lueteltuja haittavaikutuksia.

4 Pohdinta

Yksinkertainen ratkaisu selaimien sormenjälkitunnistamisen torjumiseen olisi estää JavaScript-suoritusympäristön käyttäminen www-palveluilta. Suurin osa selaimien välittämästä tiedosta on kerättävissä aktiivisilla tiedonkeräysmenetelmillä, jotka perustuvat selaimen JavaScript-suoritusympäristön käyttämiseen. Ilman aktiivista tiedonkeräystä, käyttäjien tunnistaminen ei olisi mahdollista, koska saatavilla olevan yksilöllisen tiedon määrä ei riitä käyttäjien yksilöimiseen (Laperdrix, Rudametkin ja Baudry 2016). JavaScript-suoritusympäristön estäminen johtaisi kuitenkin lähestulkoon kaikkien nykyisten www-sivustojen rikkoutumiseen.

Selaimen sormenjälkitunnistamiseen perustuvat seurantamenetelmät ovat kuitenkin usein kolmansien osapuolien teettämiä (Acar ym. 2013). JavaScript-suoritusympäristön käyttäminen voitaisiin siten estää pelkästään niiltä osapuolilta, joiden on havaittu tarjoavan sormenjälkitunnistamiseen käytettyjä menetelmiä www-sivustoilla (Torres, Jonker ja Mauw 2015). Menettely jättää kuitenkin huomiotta ne osapuolet, joita ei ole etukäteen tunnistettu. Se ei myöskään ratkaise varsinaista ongelmaa selaimen sormenjälkitunnistettavuudesta.

Selaimen sormenjälkitunnistaminen on ylipäätään mahdollista, koska selain sallii yksilöitävän tiedon keräämisen. Siten ongelmaa voidaan lähestyä selainten kehityksen näkökulmasta. Jos selainten JavaScript-ohjelmointirajapintojen suunnittelussa olisi huomioitu käyttäjien yksilöitävyys, olisi selainten sormenjälkitunnistamisen mahdollisuutta voitu vähentää (Olejnik ym. 2016). Jälkikäteen korjauksia selaimien sormenjälkitunnistettavuudessa voitaisiin toteuttaa esimerkiksi vähentämällä ohjelmointirajapintojen välittämän tiedon tarkkuutta (Olejnik ym. 2016). Muutoksien toteuttaminen standardoituihin ohjelmointirajapintoihin on kuitenkin vaikeaa ilman niiden toiminnallisuuden muuttamista.

Selaimien kehityksestä riippumattomana lähestymistapana on tarkastella selaimien suoritusympäristöön vaikuttamista. Selaimen suoritusympäristö toimii tiedon lähteenä käyttäjää yksilöivien kokoonpanotietojen keräämisessä. Sormenjälkitunniste muuttuu, mikäli suoritusympäristön kokoonpanotiedoissa tapahtuu muutoksia, mikä vaikeuttaa käyttäjän laitteistoriippumatonta tunnistamista (Blakemore, Redol ja Correia 2016). Virtuaalikoneiden avulla

samalla laitteistolla on mahdollista suorittaa useampaa eri suoritusympäristöä. Virtuaalikoneessa suoritettavan selaimen suoritusympäristö voidaan eriyttää käyttäjän varsinaisesta laitteen suoritusympäristöstä (Aggarwal ym. 2010).

Virtuaalikonetta kevyempi ratkaisu on toteuttaa selaimen suoritusympäristön virtualisointi käyttöjärjestelmätasolla, jolloin säästytään virtuaalikoneiden resurssien kuormitukselta (Zeldovich 2014). Menetelmän käyttöä selaimen sormenjälkitunnistamisen torjunnassa selvitetään tämän tutkielman osalta.

Seuraavassa luvussa tarkastellaan käyttöjärjestelmäavusteista virtualisointia. Luvussa 6 kuvataan Linux-kernelin tuki käyttöjärjestelmäavusteiselle virtualisoinnin toteuttamiselle. Luvussa 7 selvitetään voidaanko käyttöjärjestelmäavusteisella virtualisoinnilla torjua selaimen sormenjälkitunnistamista ja vastata luvussa 3.3 kuvattuihin torjuntamenetelmien haasteisiin.

5 Käyttöjärjestelmäavusteinen virtualisointi

Käyttöjärjestelmäavusteinen virtualisointi on kevyt virtualisointitekniikka, joka mahdollistaa virtuaalisen suoritusympäristön luomisen käyttöjärjestelmätasolla ilman laitteiston virtualisointia (Scheepers 2014). Virtualisointi on tietojenkäsittelyssä yleisesti käytössä oleva menetelmä, jonka avulla luodaan käytössä olevasta resurssista tai laitteistosta sitä jäljittelevä looginen yksikkö (Portnoy 2012). Looginen yksikkö toimii rajapintana resurssin, kuten laitteiston, käyttöjärjestelmän tai muistin käyttöön (Laan 2013).

5.1 Virtualisoinnin historia

Virtualisoinnin tutkimustausta pohjaa tietokoneresurssien osittamiseen usean käyttäjän kesken (osituskäyttö, time-sharing) ja usean prosessia rinnakkain suorittamiseen (moniajo, multiprogramming) (Dittner ja Rule 2007). Ensimmäinen virtualisointia hyödyntävä tietokone oli 1960-luvulla IBM:n kehittämä tutkimuskäyttöön tarkoitettu IBM M44/44X (Laan 2013). IBM M44X/44X koostui yhdestä keskuskoneesta (M44) ja useasta keskuskonetta jäljittelevästä loogisesta yksiköstä (44X:t) (Dittner ja Rule 2007). Virtuaaliset keskuskoneen yksiköt mahdollistivat saman fyysisen laitteiston eri käyttäjän samanaikaisen käytön, mikä kasvatti laitteiston käyttöastetta ja teki laitteistosta kustannustehokkaamman (Cafaro ja Aloisio 2011).

IBM M44/44x osoitti, että virtualisointi ja virtuaalikoneet ovat toimivia menetelmiä, joilla mahdollistettiin keskuskoneen suorakäytön samanaikaistaminen (Chiueh ja Brook 2005). Virtualisointia hyödynnetään nykyisin useisiin eri käyttötarkoituksiin (Chiueh ja Brook 2005). Laan (2013) ja Dittner ja Rule (2007) ryhmittelevät yleisimmät käytössä olevat virtualisoinnin käyttöalueet neljään eri käyttöaluekerrokseen, joita ovat palvelinkerros, tallennuskerros, verkkokerros ja sovelluskerros (Laan 2013). Kerrokset on kuvattu taulukossa 6.

Käyttöaluekerros	Tehtävät
Palvelinkerros	Virtuaalinen palvelinten hallinta, palvelinalustan abstrahointi.
Tallennuskerros	Virtuaalinen levynhallinta, fyysisen laitekerroksen abstrahointi.
Verkkokerros	Virtuaalinen verkonhallinta, fyysisen laitekerroksen abstrahointi.
Sovelluskerros	Virtuaalisen suoritusympäristön luominen, suoritusympäristön abstrahointi.

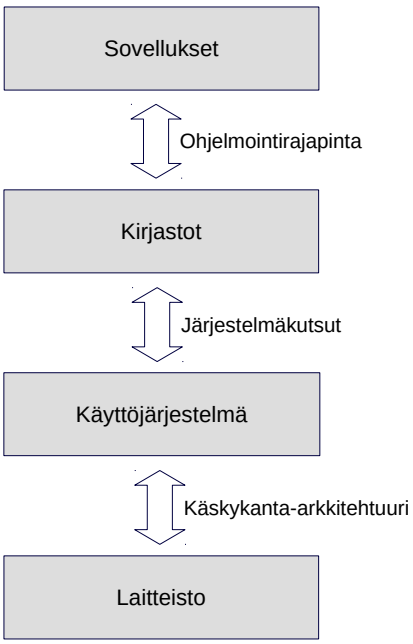
Taulukko 6. Virtualisoinnin käyttöaluekerrokset (Laan 2013; Dittner ja Rule 2007).

5.2 Virtualisoinnin abstraktiotasot

Virtualisointia voidaan pitää abstraktiotason lisäämisenä virtualisoitavan kohteen ja kohdetta käyttävän yksikön välille (Portnoy 2012). Abstraktiotasolla voidaan muuttaa virtualisoitavan kohteen olemusta, kuten piilottaa laitteiston fyysisiä ominaisuuksia tai poistaa rajoitteita laitteistoa käyttävältä ohjelmistolta (Cafaro ja Aloisio 2011). Esimerkiksi käyttöjärjestelmän muistinhallinnassa keskusmuisti virtualisoidaan loogiseksi muistiksi mahdollistaen dynaamisen muistiosoitteiden hallinnan (Hämäläinen 2012). Näin muistia varaava ohjelma vapautuu fyysisen muistin osoiteavaruuden rajoitteelta (Chiueh ja Brook 2005).

Abstraktiotaso erottaa varsinainen resurssin ohjelman näkemästä loogisesta resurssista, mikä mahdollistaa riippuvuuden poistamisen ohjelman ja resurssin väliltä (Laan 2013). Virtualisoinnin muodostamaa abstraktiotasoa käytetään riippuvuuksien poistamisen lisäksi resurssien osittamiseen tai niiden yhdistämiseen (Dittner ja Rule 2007). Osittavassa virtualisoinnissa abstraktiotaso jakaa yhden resurssin useaan eri loogiseen yksikköön (Dittner ja Rule 2007). Yhdistävässä virtualisoinnissa abstraktiotaso yhdistää usean resurssin yhdeksi loogiseksi yksiköksi (Dittner ja Rule 2007).

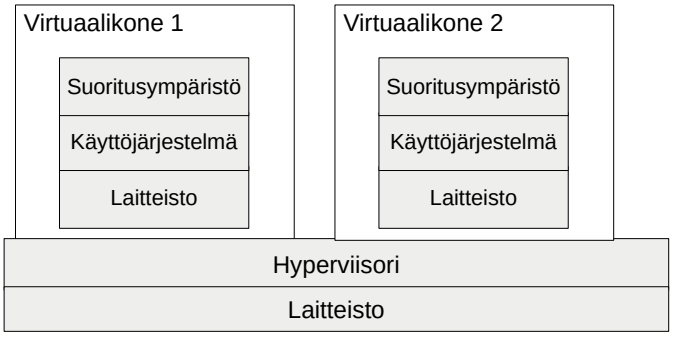
Virtualisointimenetelmiä voidaan luokitella perustuen siihen, mihin abstraktiotaso virtualisoinnissa lisätään (Chiueh ja Brook 2005). Kuviossa 5 on kuvattu viisi virtualisoinnissa ab-



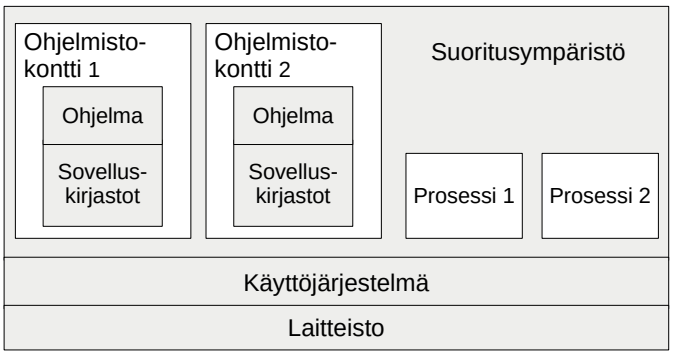
Kuvio 5. Virtualisoinnin abstraktiotasot (Chiueh ja Brook 2005).

strahoitavaa tasoa. Kun abstraktiotaso lisätään fyysisen laitteiston ja käyttöjärjestelmän välille, puhutaan virtuaalikoneista (Eder ja Kinkelin 2016). Hyperviisori-pohjaisen virtuaalikoneen arkkitehtuuri on kuvattu kuviossa 6. Hyperviisori on ohjelma, joka vastaa virtuaalikoneen luomisesta ja siten fyysisen laitteiston abstrahoinnista (Eder ja Kinkelin 2016). Käyttöjärjestelmäavusteisessa virtualisoinnissa abstraktiotaso lisätään käyttöjärjestelmän ja suoritusympäristön välille luoden virtuaalisen suoritusympäristön (Scheepers 2014). Virtuaalinen suoritusympäristö mahdollistaa prosessien ja resurssien eriyttämisen (Scheepers 2014).

Käyttöjärjestelmäavusteista virtualisointi voidaan käyttää ohjelmistokonttien toteuttamiseen (Eder ja Kinkelin 2016). Ohjelmistokontti on virtualisointitekniikkka, jossa ohjelman suoritusympäristö eristetään siirrettäväksi yksiköksi perustuen käyttöjärjestelmän tukeen (Scheepers 2014). Ohjelmistokontti sisältää ohjelman ja sen kaikki riippuvuudet (Eder ja Kinkelin 2016). Ohjelmistokonttien käyttö on yleistynyt, ja ne tarjoavat uudenlaisia lähestymistapoja ohjelmistokehitykseen, ohjelmiston suoritukseen ja ohjelmistojen jakeluun (Vase 2016). Ohjelmistokontit ovat erityisesti saaneet suosiota palveluiden jakelumallina (Eder ja Kinkelin 2016).



Kuvio 6. Hyperviisori-pohjaisen virtuaalikoneen arkkitehtuuri (Scheepers 2014).



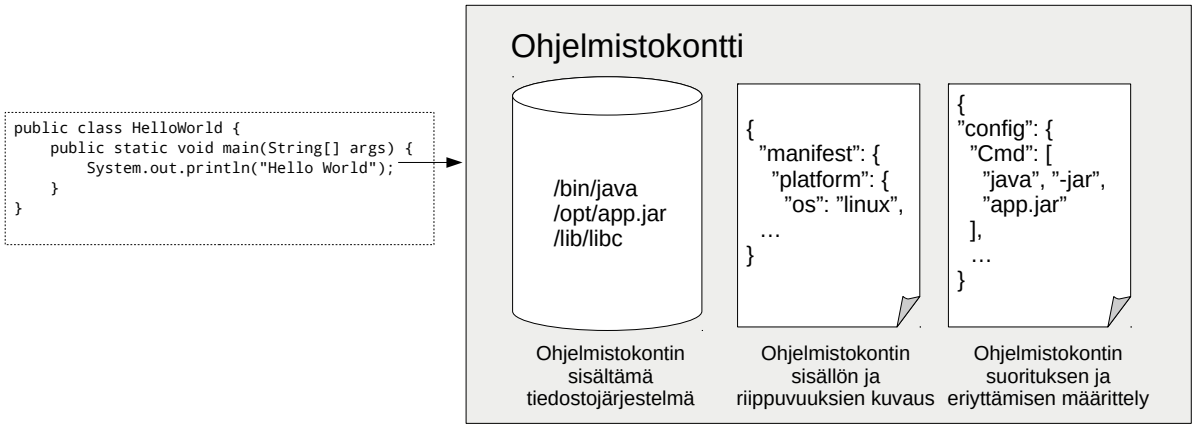
Kuvio 7. Ohjelmistokontti-virtualisointitekniikan toimintaperiaate (Scheepers 2014).

Virtuaalikoneet mahdollistavat saman prosessien ja resurssien eristämisen periaatteen kuin ohjelmistokontit, mutta toisin kuin virtuaalikoneet, ohjelmistokontit voidaan suorittaa käyttöjärjestelmän tavallisina prosesseina (Eder ja Kinkelin 2016). Laitteiston emuloinnin puuttuminen tarkoittaa samalla myös sitä, että ohjelmistokonttien sisällä suoritettavan ohjelman täytyy olla yhteensopivia laitteiston ja käyttöjärjestelmän kanssa (Eder ja Kinkelin 2016). Ohjelmistokontti-virtualisointitekniikan arkkitehtuuri on kuvattu kuviossa 7.

Seuraavassa luvussa on syvennytty tarkemmin niihin käyttöjärjestelmän mekanismeihin ja ominaisuuksiin, jotka mahdollistavat virtualisoinnin toteuttamisen käyttöjärjestelmätasolla. Tarkastelun kohteeksi on valittu avoimen lähdekoodin Linux-kernel, joka toimii Linux-pohjaisten käyttöjärjestelmien ytimenä. Selvitys Linux-kernelin tuesta käyttöjärjestelmäavusteiseen virtualisointiin perustuu Linux-kernelin dokumentaatioon ja aiempaan tutkimustietoon.

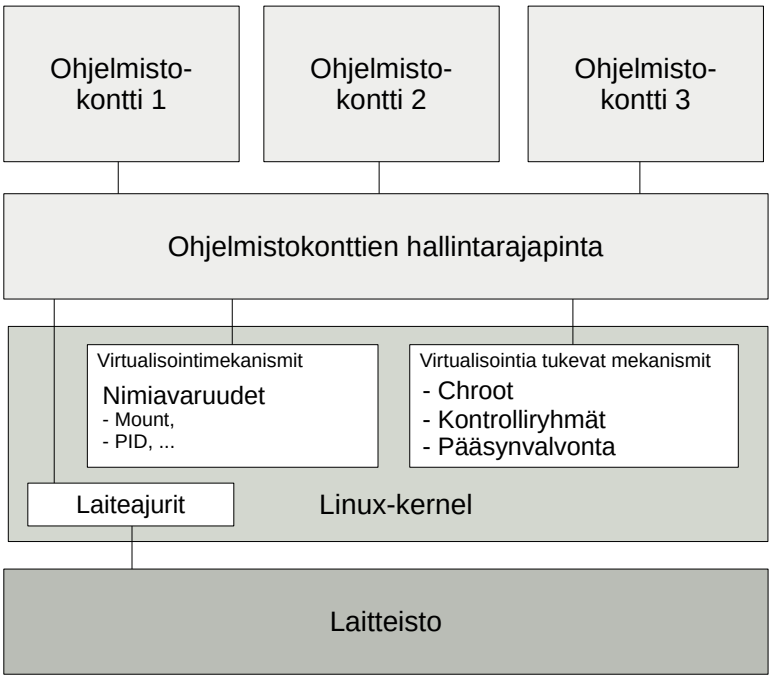
6 Linux-kernelin tuki käyttöjärjestelmäavusteiseen virtualisointiin

Tutkielmassa tarkastellaan Linux-kernelin tukemia mekanismeja, joilla on mahdollista toteuttaa käyttöjärjestelmätason virtualisointia. Linux-kernelin virtualisointimekanismit käyttöjärjestelmäavusteiseen virtualisointiin ovat kattavia ja ne toimivat useiden ohjelmistokonttiteknologian toteuttavien ohjelmistojen taustalla (Reshetova ym. 2014). Ohjelmistokonttien toteuttaminen pohjautuu käyttöjärjestelmän tukeen ja sen tarjoamiin suoritusympäristön virtualisointimekanismeihin (Scheepers 2014). Suoritusympäristön virtualisointi edellyttää käyttöjärjestelmältä tukea prosessien ja resurssien eriyttämiselle (Eder ja Kinkelin 2016).



Kuvio 8. Java-ohjelman ohjelmistokonttiin paketoiminen OCI-arkkitehtuurikuvauksen mukaisesti (Open Container Initiative 2017b).

Ohjelmistokonttien rakennetta voidaan tarkastella Open Container Initiative (OCI) -projektin ohjelmistokontteja koskevien määrittelyjen avulla. OCI on projekti, joka pyrkii tarjoamaan yhtenäisen standardin ohjelmistokontin tiedostomuodon rakenteelle ja ohjelmistokonttien laite- ja käyttöjärjestelmäriippumattomalle suoritusympäristölle (Open Container Initiative 2017a). OCI-projektin määrittelemä ohjelmistokontin tiedostorakenne koostuu ohjelmistokontin sisällön ja riippuvuuksien määrittelystä, ohjelmistokontin tiedostojärjestelmän määrittelystä ja ohjelmistokontin asetuksien määrittelystä (Open Container Initiative 2017b). OCI-projektin määrittelemä ohjelmistokontin rakenne on kuvattu kuviossa 8.



Kuvio 9. Linux-kernelin tukeman ojelmistokonttien suoritusympäristön arkkitehtuuri (Open Container Initiative 2017d; Red Hat 2017a).

OCI-projektiin perustuva ohjelmistokonttien suoritusympäristön määrittely kuvaa ohjelmistokontin standardimallin, ohjelmistokonttien standardioperaatiot, ohjelmistokontin suorituksen elinkaaren ja käyttöjärjestelmäkohtaiset toteutusmäärittelyt ohjelmistokonttien suoritusympäristön toteuttamiseen. Ohjelmistokonttien suoritusympäristön toteutus on siten käyttöjärjestelmästä ja käyttöjärjestelmän tukevista virtualisointimekanismeista riippuvainen. Käyttöjärjestelmäkohtainen toteutusmäärittely kuvaa ne käyttöjärjestelmän tukemat ominaisuudet ja mekanismit, jotka mahdollistava ohjelmistokonttien suoritusympäristön toteuttamisen. Linux-kernelin ohjelmistokonttien suoritusympäristön toteutusmäärittely määrittelee vaadituiksi mekanismeiksi Linux-kernelin nimiavaruudet, kontrolliryhmät, tiedostojärjestelmän eristämisen mekanismit sekä Linux-kernelin tukemat tietoturvamekanismit. Linux-kernelin tukema ohjelmistokonttien suoritusympäristön rakenne on kuvattu kuviossa 9. (Open Container Initiative 2017d; Red Hat 2017a)

Linux-kernelin käyttöjärjestelmäavusteista virtualisointia tukevat virtualisointimekanismit

voidaan jakaa kahteen kategoriaan, virtualisointia tukeviin mekanismeihin ja varsinaisiin virtualisointimekanismeihin. Virtualisointia tukevat mekanismit mahdollistavat prosessien eriyttämisen tai tukevat sitä, mutta näitä mekanismeja ei ole varsinaisesti suunniteltu käyttöjärjestelmäavusteiseen virtualisoinnin toteuttamiseen. Niiden avulla voidaan tehostaa prosessien eriyttämistä ja vahvistaa ohjelmistokonttien tietoturvaa (Eder ja Kinkelin 2016). Näitä Linux-kernelin sisältämiä mekanismeja ovat chroot, kontrolliryhmät ja pääsynvalvonta (Eder ja Kinkelin 2016).

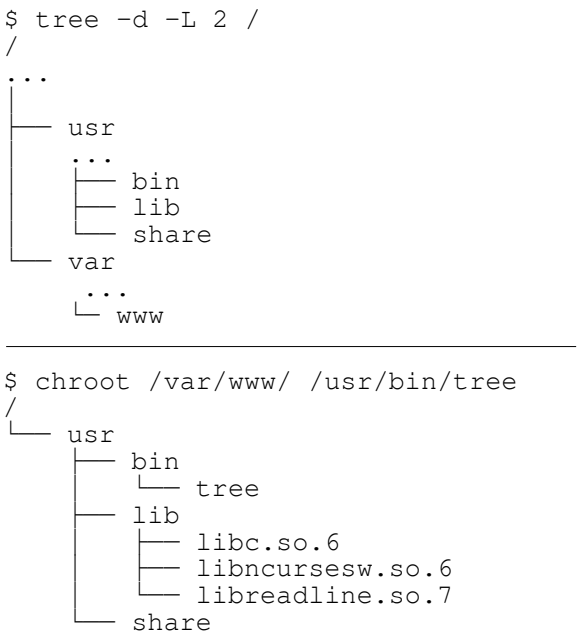
Varsinaiset käyttöjärjestelmätason virtualisontimekanismit ovat mekanismeja, jotka ovat suunniteltuja käyttöjärjestelmätason virtualisointiin. Niiden avulla voidaan eriyttää prosesseja toimimaan muista suoritusympäristöistä riippumattomissa suoritusympäristöissä (Eder ja Kinkelin 2016). Linux-kernelin varsinaiset virtualisointimekanismit ovat Linux-kernelin nimiavaruudet, jotka toteuttavat käyttöjärjestelmäavusteisen virtualisoinnin ja mahdollistavat ohjelmistokonttien toteuttamisen (Scheepers 2014). Linux-kernelin nimiavaruudet mahdollistavat abstraktiotason lisäämisen käyttöjärjestelmän globaalien resurssien ja prosessien välille (Open Container Initiative 2017c).

Seuraavissa alaluvuissa tarkastellaan tarkemmin Linux-kernelin tukemia käyttöjärjestelmäavusteisen virtualisoinnin mahdollistamia mekanismeja. Ensiksi tarkastellaan käyttöjärjestelmäavusteista virtualisointia tukevia mekanismeja, joita ovat chroot, kontrolliryhmät ja pääsynvalvonta. Lopuksi luvussa 6.4 perehdytään Linux-kernelin nimiavaruuksiin, jotka toteuttavat Linux-kernelin käyttöjärjestelmäavusteisen virtualisoinnin.

6.1 Chroot

Chroot on Linux-kernelin systeemikomento, joka muuttaa suoritettavan ohjelman tai prosessin juurihakemiston (ks. *Linux man-pages* 2016, chroot). Prosessin juurihakemiston vaihtamisen avulla voidaan vaikuttaa siihen, miten prosessi näkee oman suoritusympäristönsä tiedostojärjestelmän hakemistopolun. Tämä mahdollistaa prosessin tiedostojärjestelmään pääsyn rajaamisen (Eder ja Kinkelin 2016). Chroot-komennon prosessin juurihakemiston vaihtaminen on kuvattu kuviossa 10.

Ohjelmistokonttien toteuttamisen kannalta, ohjelmistokontin suoritusympäristöstä ei pitäi-



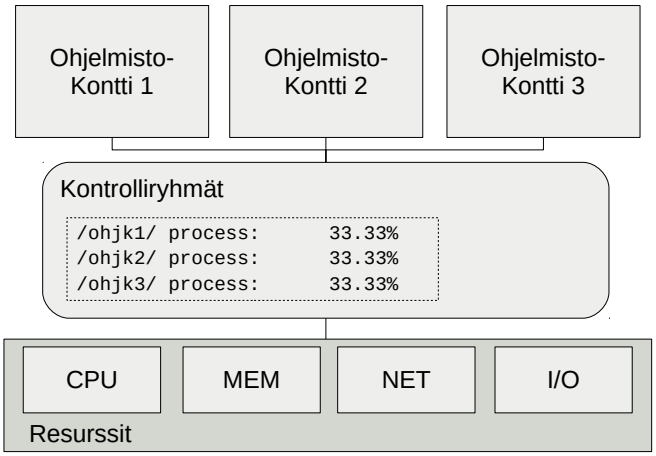
Kuvio 10. Tree-ohjelman suorittaminen juurihakemistossa (yllä) ja chroot-komennolla vaihdetussa juurihakemistossa (alla).

si olla pääsyä käyttöjärjestelmän suoritusympäristöön tai muiden ohjelmistokonttien suoritusympäristöihin. Koska Chroot-komento mahdollistaa vain prosessin juurihakemiston vaihtamisen, ei se mahdollista tiedostojärjestelmän täydellistä virtualisointia. Chroot-komentoa ei ole suunniteltu tietoturvalliseksi, eikä se takaa prosessin pääsyä vaihdetun juurihakemiston ulkopuolelle (ks. *Linux man-pages* 2016, chroot).

6.2 Kontrolliryhmät

Kontrolliryhmät (cgroups) on Linux-kernelin sisältämä resurssien hallintamekanismi prosessien resurssien hallintaan ja seurantaan. Kontrolliryhmä koostuu siihen sisällytetyistä prosesseista ja resurssikohtaisista käyttörajoista. Jokainen kontrolliryhmään kuuluva prosessi noudattaa kontrolliryhmälle asetettuja käyttörajoja. Kontrolliryhmiä on mahdollista määrittää hierarkkisesti siten, että hierarkiassa korkeammalla oleva kontrolliryhmä määrittää alempana olevien kontrolliryhmiin kuuluvien prosessien käyttörajojen ylärajat. (ks. *Linux man-pages*

2016, cgroups)



Kuvio 11. Ohjelmistokonttien resurssien hallinta kontrolliryhmien avulla (Red Hat 2017b).

Käyttörajojen asettamisen lisäksi kontrolliryhmien avulla on mahdollista seurata prosessien resurssien käyttöä ja hallita prosessien tiloja (Red Hat 2017b). Kontrolliryhmät toimivat myös tietoturvamekanismina, jolla voidaan estää resurssien liikakäyttöä (Reshetova ym. 2014). Kontrolliryhmien toimintaperiaate on kuvattu kuviossa 11.

Kontrolliryhmien toteuttama resurssien hallinta on tärkeä ohjelmistokonttien toteutuksen kannalta. Resurssien hallinnalla voidaan vaikuttaa siihen, kuinka prosessit näkevät käytettävissä olevat resurssit suoritusympäristössään ja jakaa isäntäkoneen resurssit tasaisesti useiden eri ohjelmistokonttien kesken (Eder ja Kinkelin 2016). Resurssien jakaminen mahdollistaa sen, ettei yksi ohjelmistokontti voi varata kaikkia järjestelmän resursseja siten, että se vaikuttaisi muiden ohjelmistokonttien prosessien toimintaan. Tämä mahdollistaa resurssien saatavuuden riippumattomuuden toteuttamisen ohjelmistokonttien välille.

6.3 Pääsynvalvonta

Pääsynvalvontamekanismeilla voidaan vaikuttaa siihen, mihin resursseihin prosessilla on pääsy ja mitä resursseja ne voivat käyttää (Eder ja Kinkelin 2016). Pääsynvalvontamekanismit voidaan luokitella kolmeen luokkaan, joita ovat yksilöpohjainen pääsynvalvonta (Discretionary Access Control, DAC), pakotettu pääsynvalvonta (Mandatory Access Control, MAC) ja roolipohjainen pääsynvalvonta (Role-Based Acess Control, RBAC) (Klaper 2015). Ohjel-

mistokonttien kannalta oleellisimmat ovat yksilöpohjainen pääsynvalvonta (DAC) ja pakotettu pääsynvalvonta (MAC).

Yksilöpohjainen pääsynvalvonta perustuu käyttäjä ja käyttäjäoikeusryhmiin (Klaper 2015). Se onko prosessilla oikeus käyttää jotakin resurssia tarkoittaa yksilöpohjaisessa pääsynvalvonnassa sitä, onko prosessin käyttäjälle ja käyttäjäryhmille annettu tähän oikeudet (Eder ja Kinkelin 2016). Linux-kernelin yksilöpohjainen pääsynvalvonta perustuu POSIX-standardiin ja noudattaa POSIX-standardin pääsynvalvontalistojen (Acess Control List, ACL) käyttämistä (ks. *Linux man-pages* 2016, acl). Pääsynvalvontalista sisältää käyttäjät, jotka on oikeutettuja käyttämään resurssia tai operaatioita, joita käyttäjille on myönnetty resurssin käyttöön (ks. *Linux man-pages* 2016, acl).

Pakotetussa pääsynvalvonnassa (MAC) käytetään pääsynvalvonnan kriteereinä ennalta määriteltyjä valtuutussääntöjä (Klaper 2015). Pakotettua pääsynvalvontaa käytetään vahvistamaan yksilöpohjaisia pääsynvalvontamekanismeja (Reshetova ym. 2014) ja pitämään prosessien resurssien käyttöoikeudet minimissään (Eder ja Kinkelin 2016). Pakotetulla pääsynvalvonnalla voidaan vahvistaa ohjelmistokonttien prosessien eriyttämistä ja valvoa ohjelmistokontin sisäisten prosessien pääsynvalvontaa valtuutusääntöjen avulla (Reshetova ym. 2014).

Ongelmaksi pakotetussa pääsynvalvonnassa muodostuu ohjelmistokonttien sisäiset valtuutussäännöt. Ohjelmistokonttien siirrettävyyden kannalta, valtuutussäännöt tulisi pystyä virtualisoimaan siten, että ne olisivat riippumattomia isäntäkoneesta (Reshetova ym. 2014). Linux-kernel ei tue pakotetun pääsynvalvonnan virtualisointia, mutta tuen lisääminen sille on suunnitteilla nimiavaruuden muodossa (Reshetova ym. 2014). Linux-kernelin tukemat nimiavaruudet ja niiden toteuttama käyttöjärjestelmätason virtualisointi on kuvattu seuraavassa luvussa.

6.4 Linux-kernelin nimiavaruudet

Nimiavaruudet on Lixux-kernelin käyttöjärjestelmätason virtualisointimekanismi, joka mahdollistaa globaalien resurssien eriyttämisen prosessikohtaisesti (Rosen 2013). Nimiavaruudet mahdollistavat abstraktiotason lisäämisen käyttöjärjestelmän globaalien resurssien ja prosessien välille (Open Container Initiative 2017c). Siten jokaisella prosessilla voi olla oma nä-

Nimiavaruus	Virtualisoitava resurssi
Mount	Tiedostojärjestelmän eriyttäminen.
PID	Prosessien tunnisteiden eriyttäminen.
IPC	Viestinvälitysmekanismien eriyttäminen.
UTS	Verkkotunnusten eriyttäminen.
Network	Verkkoresurssien eriyttäminen.
User	Käyttäjien ja käyttäjäryhmien eriyttäminen.
Cgroup	Kontrolliryhmien eriyttäminen.

Taulukko 7. Linux-kernelin version 4.10 sisältämät nimiavaruudet (*Linux man-pages* 2016).

kymä käyttöjärjestelmän resursseista. Nimiavaruudet ovat välttämätön mekanismi ohjelmistokonttien toteuttamiselle Linux-pohjaisilla käyttöjärjestelmillä (Open Container Initiative 2017c). Nimiavaruudet mahdollistavat prosessin eriyttämisen siten, ettei prosessilla ole pääsyä virtualisoidun suoritusympäristönsä ulkopuolelle (Reshetova ym. 2014).

Taulukossa 7 on lueteltu Linux-kernelin version 4.10 sisältämät seitsemän nimiavaruutta. Seuraavissa alaluvuissa nimiavaruuksien toiminta ja niiden suoritusympäristön virtualisointi- ja eriyttämiskohteet on kuvattuna tarkemmin. Seuraavassa luvussa 7 selvitetään voidaanko Linux-kernelin nimiavaruuksien mahdollistamalla käyttöjärjestelmäavuesteisella virtualisoinnilla torjua selaimen sormenjälkitunnistamista.

6.4.1 Mount-nimiavaruus

Mount-nimiavaruus on ensimmäinen Linux-kerneliin lisätty nimiavaruus. Se on ollut saatavilla Linux-kernelin versiosta 2.4.19 saakka. Mount-nimiavaruus mahdollistaa tiedostojärjestelmän liitoskohtien eriyttämisen ja siten prosessien tiedostojärjestelmän virtualisoinnin. (ks. *Linux man-pages* 2016, mount_namespaces)

Tiedostojärjestelmän virtualisointi on olennainen virtualisointimekanismi tietoturvallisten ohjelmistokonttien toteuttamisessa. Tiedostojärjestelmän eriyttämisellä voidaan rajata prosessien tiedostojärjestelmään pääsyä eristämällä tiedostojärjestelmä prosessikohtaiseksi (Reshetova ym. 2014). Tällöin jokaisella prosessilla voi olla yksilöllinen näkymä suoritusympä-

ristönsä tiedostojärjestelmästä.

Ohjelmistokontin tiedostojärjestelmän eriyttämisessä on kaksi vaihtoehtoa, täysi eriyttäminen ja osittainen eriyttäminen. Täydessä eriyttämisessä luodaan prosessille kokonaan uusi tiedostojärjestelmä, jossa on prosessin kaikki suoritukseen tarvitsemat tiedostojärjestelmän resurssit. Osittaisessa eriyttämisessä osa käyttöjärjestelmän tiedostojärjestelmästä jaetaan prosessin virtuaalisen tiedostojärjestelmän kanssa. (Reshetova ym. 2014)

Mount-nimiavaruus toteuttaa tiedostojärjestelmän virtualisoinnin Linux-kernelin Proc-tiedostojärjestelmän välityksellä. Proc-tiedostojärjestelmässä jokaisella prosessilla on oma mountinfo-tiedosto, joka kuvaa prosessin tiedostojärjestelmän liitoskohdat, jotka sisältyvät prosessin mount-nimiavaruuteen. Saman mount-nimiavaruuteen kuuluvat prosessit näkevät saman näkymän mountinfo-tiedostosta. Mount-nimiavaruus tarjoaa rajapinnan tiedostojärjestelmien liitoskohtien automaattiseen jakoon mount-nimiavaruuksien välillä. (ks. *Linux man-pages* 2016, mount_namespaces)

6.4.2 PID-nimiavaruus

PID-nimiavaruus mahdollistaa prosessien tunnisteiden eriyttämisen (ks. *Linux man-pages* 2016, pid_namespaces). Prosessien tunnisteiden eriyttäminen on olennainen mekanismi ohjelmistokonttien toteuttamiselle (Reshetova ym. 2014). Prosessien eriyttämisellä voidaan vaikuttaa siihen, mitä toisia prosesseja prosessi näkee suoritusympäristössään ja minkä muiden prosessien kanssa prosessi voi kommunikoida (Reshetova ym. 2014).

Prosessien tunnisteiden eriyttäminen mahdollistaa prosessien riippumattomuuden ja ohjelmistokonttien siirrettävyyden toteuttamisen. Kahdella eri prosessilla voi olla samat tunnisteet eri PID-nimiavaruuksissa (ks. *Linux man-pages* 2016, pid_namespaces). Koska prosessien tunnisteet ovat riippumattomia, prosessin tila voidaan pysäyttää ja jatkaa PID-nimiavaruuden sisällä säilyttämällä alkuperäinen prosessin tunniste (ks. *Linux man-pages* 2016, pid_namespaces). Ominaisuus tekee mahdolliseksi sen, että PID-nimiavaruus voidaan siirtää suoritusympäristöstä toiseen ilman riippuvuusongelmia, mikä mahdollistaa siirrettävien ohjelmistokonttien toteuttamisen (Reshetova ym. 2014).

Prosessien eriyttämismekanismia on edellytys tietoturvallisten ohjelmistokonttien toteutta-

misessa. PID-nimiavaruudet on toteutettu puurakenteena, jossa alempana olevat prosessit eivät näe eivätkä voi kommunikoida puurakenteessa ylempänä olevien prosessien kanssa (ks. *Linux man-pages* 2016, pid_namespaces). Prosessien siirtyminen puurakenteessa ylöspäin on estetty, mikä estää prosessia pääsemästä nimiavaruuden ulkopuolelle (ks. *Linux man-pages* 2016, pid_namespaces). Tämä lisää ohjelmistokonttien tietoturvallisuutta, koska ohjelmistokontin sisäisillä prosesseilla eivät voi kommunikoida ohjelmistokontin ulkopuolisten prosessien kanssa (Reshetova ym. 2014).

6.4.3 User-nimiavaruus

User-nimiavaruus mahdollistaa prosessien käyttäjäoikeuksien eriyttämisen suoritusympäristöönsä ja käyttäjäoikeuksien rajaamisen eri suoritusympäristöjen välillä (ks. *Linux man-pages* 2016, user_namespaces). Tämän avulla voidaan vaikuttaa siihen, miten prosessi näkee käyttäjäoikeudet suoritusympäristössään riippumatta isäntäkoneesta muista suoritusympäristöistä.

Tietoturvamuuttujien virtualisointi mahdollistaa ohjelmistokonttien käyttäjäoikeuksien riippumattomuuden. Prosesseilla voi olla eri käyttäjätunnukset, käyttäjäryhmätunnukset, juurikansiot, avaimet ja käyttöoikeuskyvyt (capabilities) eri user-nimiavaruuksissa (ks. *Linux man-pages* 2016, user_namespaces). Käyttäjäoikeuksien riippumattomuudella voidaan mahdollistaa, ettei prosessilla ole käyttäjäoikeuksia suoritusympäristönsä ulkopuolella. Tämä mahdollistaa esimerkiksi sen, että ohjelmistokontin pääkäyttäjä oikeudet eivät oikeuta toisten tai isäntäkoneen pääkäyttäjäoikeuksiin (Reshetova ym. 2014).

User-nimiavaruudella voidaan vahvistaa prosessien eriyttämistä käyttäjäoikeustasolla. User-nimiavaruuksilla voidaan vahvistaa muiden nimiavaruuksien tietoturvallisuutta ja pienentää tietoturvahyökkäysten riskiä, mikä mahdollistaa tietoturvallisten ohjelmistokonttien toteuttamisen (Reshetova ym. 2014).

6.4.4 Muut Linux-kernelin tukemat nimiavaruudet

Linux-kernelin muut nimiavaruudet ovat IPC-, UTS-, Network- ja Cgroup-nimiavaruudet. ICP-nimiavaruus mahdollistaa IPC-resurssien virtualisoinnin (ks. *Linux man-pages* 2016,

namespaces). IPC (Interprocess communication) kuvaa viestinvälitysmekanismia, joka mahdollistaa prosessien välisen viestinnän (Hämäläinen 2012). Yleinen käyttöjärjestelmän tarjoama viestinvälitysmekanismi on sanomanvälitys, joka mahdollistaa prosessien välisen viestinnän ja tilan synkronoinnin (ks. Hämäläinen 2012, luku 4.3). Muita yleisiä IPC-mekanismeja ovat jaetut muistialueet (shared memory), putket (pipes), signaalit (signals) ja RPC-mekanismit (remote procedure call) (ks. Hämäläinen 2012, luku 4.4). IPC-nimiavaruus rajaa prosessin IPC-mekanismeihin pääsyn vain nimiavaruuden sisällä oleviin IPC-resursseihin (ks. *Linux man-pages* 2016, namespaces). Tämä mahdollistaa ohjelmistokontin ulkopuolisten IPC-resurssien, kuten jaetun muistialueen ja viestijonojen eriyttämisen (Reshetova ym. 2014).

Verkkoresurssien virtualisoinnista vastaavat UTS- ja Network-nimiavaruudet (ks. *Linux man-pages* 2016, namespaces). UTS-nimiavaruus mahdollistaa verkkotunnusten virtualisoinnin, kun Network-nimiavaruus vastaa muiden verkkoresurssien, kuten reititystaulujen, palomuurin ja porttien virtualisoinnista (ks. *Linux man-pages* 2016, namespaces). Verkkoresurssien virtualisoinnilla pystytään vaikuttamaan siihen, miten prosessit näkevät verkkoresurssit suoritusympäristössään. Verkkoresurssien virtualisoinnilla mahdollistetaan ohjelmistokontin riippumattomuus isäntäkoneesta. Verkkoresurssien riippumattomuus tarkoittaa sitä, että usean prosessin saman portin kuuntelemisen eri ohjelmistokonttien välillä (Reshetova ym. 2014). Verkkoresurssien virtualisointi parantaa myös ohjelmistokonttien tietoturvaa, koska sen avulla voidaan estää verkkohyökkäysten eteneminen ohjelmistokontin ulkopuolelle isäntäkoneeseen tai toisiin ohjelmistokonttien prosesseihin (Reshetova ym. 2014).

Linux-kernelin uusin nimiavaruus on Cgroup-nimiavaruus. Tuki Cgroup-nimiavaruudelle lisättiin Linux-kernelin versiossa 4.10. Cgroup-nimiavaruus lisää mahdollisuuden kontrolliryhmien virtualisoinnille. Kontrolliryhmien virtualisoinnilla voidaan parantaa ohjelmistokonttien tietoturvaa ja siirrettävyyttä. (ks. *Linux man-pages* 2016, cgroup_namespaces)

7 Selaimen sormenjäljen muuttaminen käyttäen ohjelmistokonttia

Luvussa kuvataan selaimen suoritusympäristön virtualisointi ohjelmistokontin avulla. Selaimen suoritusympäristön virtualisoinnissa hyödynnetään ohjelmistokonttiteknologioita ja työkaluja, jotka helpottavat luvussa 6 kuvattujen Linux-kernelin nimiavaruuksien hallintaa. Ohjelmistokonttien luomisessa käytetään LXD-hallintatyökalua, joka on kehitetty ohjelmistokonttien hallintaan ja jakeluun (Canonical Ltd 2017). LXD käyttää taustalla LXC-hallintarajapintaa, joka on kehitetty erityisesti helpottamaan Linux-kernelin käyttöjärjestelmäavusteisten virtualisointiteknologioiden käyttämistä käyttäjätasolla (Reshetova ym. 2014).

Selaimen ohjelmistokonttiin paketoimisen jälkeen selvitetään, pystytäänkö selaimen sormenjälkitunnisteeseen vaikuttamaan ohjelmistokontin sisältä. Testin toteuttamisen jälkeen analysoidaan testin tuloksen, ja selvitetään, voidaanko käyttöjärjestelmäavusteisilla virtualisointiteknologioilla vastata selaimen sormenjälkitunnistamisen torjunnan haasteisiin. Viimeisessä luvussa tehdään yhteenveto tutkielman tuloksista ja pohditaan tarvetta jatkotutkimukselle.

7.1 Firefox-selaimen ohjelmistokonttiin paketoiminen

Firefox-selaimen suorittaminen ohjelmistokontin sisällä edellyttää selaimen riippuvuuksien sisällyttämistä ohjelmistokontin tiedostojärjestelmässä. Tämä voidaan toteuttaa helpoiten käyttämällä valmiita ohjelmistokontteja, jotka paketoivat selaimelle yhteensopivan Linux-jakelun. LXD tarjoaa työkalut valmiiden ohjelmistokonttien lataamiseen ja jakeluun verkon yli (Canonical Ltd 2017). Ohjelmistokontti, joka paketoi Ubuntun Linux-jakelun voidaan ladata ja asentaa LXD:llä seuraavalla komennolla.

```
$ lxc launch ubuntu:16.04 ubuntu-kontti
```

LXD:n launch-komento lataa haettavan ohjelmistokontin levykuvan ensimmäisenä parametrina annetusta verkkosijainnista. Toisena parametrina määritetään luodun ohjelmistokontin nimi, jota käytetään jatkossa ohjelmistokontin hallinnassa. LXD lataa oletuksena alusta-

koneen suoritinarkkitehtuurin kanssa yhteensopivan ohjelmistokontin levykuvan. Launch-komento asentaa ja suorittaa ohjelmistokontin latauksen valmistumisen jälkeen.

Komennon suorittaminen ohjelmistokontin sisällä tapahtuu LXD:n exec-komennolla. Firefox-selaimen asennus tehdään Ubuntu-jakelun pakettienhallinnan apt-komentotyökalulla suorittamalla seuraavat komennot. Ensimmäisellä komennolla päivitetään saatavilla olevien pakettien tiedot ja jälkimmäisellä komennolla asennetaan Firefox-selaimen uusin saatavilla oleva versio.

```
$ lxc exec ubuntu-kontti -- apt update
$ lxc exec ubuntu-kontti -- apt install firefox
```

Selaimen asennuksen jälkeen täytyy ohjelmistokontille antaa oikeudet käyttää alustakoneen näytönohjainta laitteistokiihdytykseen, jotta graafisen käyttöliittymän omaava ohjelma voidaan suorittaa (Xenitellis 2017). Oletuksena LXD ei jaa näytönohjainta, vaan sen jakaminen tulee määrittää erikseen. Alustakoneen näytönohjain jaetaan ohjelmistokontin kesken seuraavalla komennolla.

```
$ lxc config device add ubuntu-kontti gpu gpu
```

Lopuksi, jotta ohjelmistokontin sisällä suoritettava selain voi piirtää käyttöliittymänsä alustakoneen työpöydälle, tulee sillä olla pääsy alustakoneen X-ikkunointipalvelimen näytölle. Tämä voidaan tehdä jakamalla Unix-pistoke X-ikkunointipalvelimen alustakoneen ja ohjelmistokontin välillä, sekä jakamalla alustakoneen käyttäjän Xauthority-tiedosto ohjelmistokontin käyttäjän kanssa (Xenitellis 2017). Unix-pistoke toteuttaa prosessien välisen kommunikaation Linux-järjestelmissä, ja Xauthority-tiedosto toimii pääsynvalvonnassa käyttäjän X-ikkunointipalvelimen sessiolle. Oheiset tiedostojärjestelmän jaot toteutetaan LXD:llä seuraavasti.

```
$ lxc config device add ubuntu-kontti X0 disk readonly=true \
    path=/tmp/.X11-unix/X0 source=/tmp/.X11-unix/X0
$ lxc config device add ubuntu-kontti Xauthority disk readonly=true \
    path=${XAUTHORITY} source=/home/${USER}/.Xauthority
```

Komennot tekevät kaksi liitosta alustakoneen ja ohjelmistokontin tiedostojärjestelmien välille. Komennoissa käytetään parametreina ”readonly”-parametria, joka määrittää liitokset lu-

kutilaan. Näin ohjelmistokontin prosessit eivät voi kirjoittaa tehtyihin liitoksiin alustakoneen tiedostojärjestelmään. LXD käyttää taustalla luvussa 6.4.1 kuvattua Mount-nimiavaruutta ohjelmistokontin tiedostojärjestelmän virtualisoinnissa ja liitoskohtien luomisessa.

Liitoskohtien määrittelyjen jälkeen tulee ohjelmistokontin käyttäjän tunnisteet asettaa vastaamaan alustakoneen käyttäjän tunnisteita, jotta ohjelmistokontin sisällä suoritettavalla prosessilla on pääsy jaetuille tiedostojärjestelmän liitoksilla ja oikeudet käyttää jaettua alustakoneen näytönohjainta (Xenitellis 2017). Nämä vaaditut pääsynvalvonnan määrittelyt voidaan tehdä seuraavilla komennoilla.

```
$ lxc config set ubuntu-kontti raw.idmap "both $UID 1000"
$ lxc config device set ubuntu-kontti gpu uid 1000
$ lxc config device set ubuntu-kontti gpu gid 1000
```

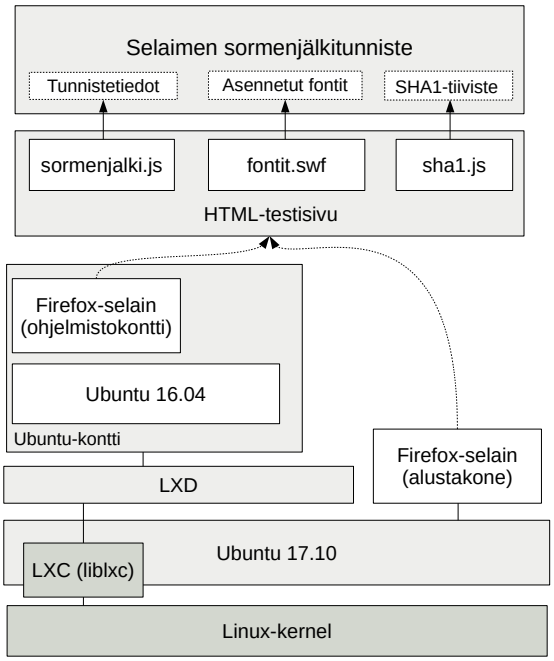
Ensimmäinen komento määrittää ohjelmistokontin oletuskäyttäjän tunnistenumeron vastaamaan alustakoneen nykyisen käyttäjän tunnistenumeroa. LXD käyttää taustalla luvussa 6.4.3 kuvattua User-nimiavaruutta käyttäjien tunnisteiden virtualisoimisessa. Käyttäjän tunnistenumeron vaihdon lisäksi jaetulle näytönohjaimelle määritetään ohjelmistokontin oletuskäyttäjän tunnistenumerot, jotta ohjelmistokontin käyttäjällä on oikeudet näytönohjaimen käyttämiseen (Xenitellis 2017). LXD:llä ladatun Ubuntu-jakelun paketoivan ohjelmistokontin oletuskäyttäjän tunnistenumero on 1000 ja käyttäjätunnus ”ubuntu”. Lopuksi ohjelmistokontti tulee käynnistää uudelleen, jotta muutetut ohjelmistokontin suoritusympäristön asetukset tulevat voimaan. Firefox-selaimen suorittaminen tapahtuu tämän jälkeen seuraavalla komennolla.

```
$ lxc exec ubuntu-kontti --env DISPLAY=$DISPLAY -- \
    sudo --login --user ubuntu firefox
```

Komennossa määritetään ympäristömuuttujana alustakoneen aktiivisen käyttäjän X-ikkunanäytön tunniste ja annetaan komento Firefox-selaimen suorittamiseen ubuntu-käyttäjänä. Tietoturvan takia selain halutaan suorittaa tavallisena käyttäjänä ilman pääkäyttäjänoikeuksia, joita LXD:n exec-komento muutoin käyttää oletuksena. Luvussa kuvatut vaiheet ovat tutkielman testiin luodun ohjelmistokontin taustalla. Testiin luodun ohjelmistokontin yksityiskohtainen asetusten määrittely on saatavilla liitteestä A.

7.2 Testimenetelmä

Testiin valittiin yhdeksän eri käyttäjää yksilöivää tunnistetta, jotka sisältävät tietoa käyttäjän suoritusympäristön kokoonpanosta. Testiin valitut tunnisteet on lueteltu taulukossa 8. Testissä pyrittiin vaikuttamaan ohjelmistokontin sisällä suoritettavan selaimen suoritusympäristöön siten, että selaimen sormenjälkitunnisteeseen kerätyt yhdeksän tietuetta pystyttiin muuttamaan. Tietojen muuttamisessa käytettiin luvussa 3 kuvattuihin torjuntamalleihin perustuvaa tiedon väärentämistä. Testiin luotu testiympäristö on kuvattu kuviossa 12.



Kuvio 12. Selaimen sormenjälkitunnisteen mittaamiseen luotu testiympäristö.

Selaimen sormenjäljen mittaamista varten, testiin luotiin selaimen tunnistetietoja keräävä JavaScript-skripti. Kaikki testissä kerätyt yhdeksän tunnistetta on kerättävissä aktiivisesti selaimen JavaScript-ohjelmointirajapintojen kautta. Testissä käytetty JavaScript-skripti on saatavilla liitteestä B. Kerättäviin tunnistetietoihin kuului tieto käyttäjän asentamista fonteista, jonka kerääminen toteutettiin Flash-liitännäistä käyttämällä. Asennetut fontit listaava ActionScript-skripti ja ohjeet sen kääntämiseen ovat saatavilla liitteestä C. Testiin toteutettiin lisäksi HTML-sivu, joka näyttää luodun sormenjälkitunnisteen ja erittelee kerätyt tunnistetiedot. HTML-sivu on saatavilla liitteestä D. Testi suoritettiin alustakoneella, johon asennettiin Ubuntu-jakelun viimeisin kehitysversio. Testissä käytettyjen ohjelmien versiotiedot on

nähtävissä liitteestä E.

Yksilöllinen sormenjälkitunniste muodostettiin laskemalla tiiviste kerättyjen tietojen JSON-tiedostomuodossa olevalle merkkijonolle. Tiivisteen laskennassa käytettiin SHA1-tiivistefunktiota ja sen toteuttavaa jsSHA -JavaScript-kirjastoa (Turek 2017). Apuna sormenjälkitunnisteen luonnissa hyödynnettiin AmIUnique-testipalvelun avointa lähdekoodia (Laperdrix 2017) ja Mozilla-organisaation ylläpitämää JavaScript-ohjelmointirajapintojen dokumentaatiota (Mozilla Contributors 2017). Seuraavassa luvussa raportoidaan testin tulokset ja analysoidaan, voidaanko testin tulosten perusteella käyttöjärjestelmäavusteisilla virtualisoinnilla vastata selaimen sormenjälkitunnistamisen haasteisiin.

7.3 Tulokset

Testiin kerätyistä yhdeksästä käyttäjää yksilöivästä tiedosta viidelle saatiin eri arvot alustakoneen ja ohjelmistokontin selaimille. Tulos johti eri sormenjälkitunnisteiden muodostumiseen alustakoneen ja ohjelmistokontin selaimille. Testissä kerätyt käyttäjää yksilöivät tiedot ja tietoihin vaikuttamisen mahdollisuudet on kuvattu taulukossa 8. Testissä mitatut arvot kerätyille tunnistetiedoille alustakoneessa ja ohjelmistokontissa ovat nähtävissä liitteestä F ja liitteestä G.

Testissä ei onnistuttu vaikuttamaan tietoihin, jotka sisälsivät käyttöjärjestelmän alustan, näytön, näytönohjaimen ja selaimen liitännäisiä kuvaavia tietoja. Ohjelmistokontin sisällä suoritetut prosessit käyttävät alustakoneen Linux-kerneliä, mikä johti samoihin selaimien käyttöjärjestelmää kuvaaviin tietoihin. Jotta tietoja olisi mahdollista muutta, tulisi Linux-kernelin tukea nimiavaruutta, joka mahdollistaisi näiden tietojen virtualisoinnin ohjelmistokontin sisällä. Nykyisistä Linux-kernelin nimiavaruuksista UTS-nimiavaruus mahdollistaa ainoastaan verkkotunnuksen virtualisoinnin. Linux-kernelin tunnistetietoja ei voida kuitenkaan virtualisoida.

Molemmat selaimet ilmoittivat testissä saman näytön resoluution ja värisyvyyden. Tietoon ei pystytty vaikuttamaan, koska molemmat selaimet käyttivät samaa alustakoneen näyttöä. Ohjelmistokontin luonnissa määritettiin ohjelmistokontin sisällä suoritetuille prosesseille mahdollisuus piirtää graafinen käyttöliittymä alustakoneen työpöydälle, jolloin ohjelmistokontin

Tunniste	Eri arvot	Vaikuttaminen
Asennetut fontit	Kyllä	Ohjelmistokontin asennettujen fonttien muuttaminen.
Kieli	Kyllä	Ohjelmistokontin kielen muuttaminen.
Aikavyöhyke	Kyllä	Ohjelmistokontin aikavyöhykkeen muuttaminen.
Näytönohjaimen tiedot	Kyllä	Vanhemman Ubuntu-jakelun käyttäminen.
Canvas-renderöintidata	Kyllä	Ohjelmistokontin fonttien renderöintiasetusten muuttaminen.
UserAgent	Ei	
Käyttöjärjestelmän alusta	Ei	
Liitännäiset	Ei	
Näytön tiedot	Ei	

Taulukko 8. Testissä kerätyt käyttäjää yksilöivät tiedot ja tietoihin vaikuttamisen mahdollisuudet.

sisällä suoritettu selain ilmoitti saman näytön tiedot.

Selaimen liitännäisiä kuvaavaan tietoon ei onnistuttu vaikuttamaan, koska Firefox-selaimen perinteisten NPAPI-liitännäisten tuki päättyi versiosta 52 alkaen. NPAPI-liitännäisten tuki säilytettiin vielä Flash-liitännäisellä, mikä näkyi testissä siinä, että molemmat testin selaimet ilmoittivat vain Flash-liitännäistä kuvaavat tiedot. Perinteisten liitännäisten poistuminen tarkoittaa sitä, että käyttäjien yksilöiminen selaimeen asennettujen liitännäisten osalta ei ole enää käytännössä mahdollista. Firefox-selaimen ilmoittamaa Flash-liitännäinen versionumeroa voidaan vielä käyttää yksilöllisenä tietona, mutta liitännäisten automaattisten päivitysten ansiosta suurimmalla osalla käyttäjistä on käytössä Flash-liitännäisen viimeisin versio, jolloin eroavaisuutta ei synny käyttäjien välillä.

Ohjelmistokontin selaimen näkemät kieli ja aikavyöhyke voitiin testissä muuttaa ohjelmistokontin Ubuntu-jakelun asetuksia muuttamalla. Tämä mahdollistaa suoritusympäristön kielen ja aikavyöhykkeen eriyttämisen alustakoneen suoritusympäristöstä. Eriyttäminen voitiin toteuttaa myös asennettujen fonttien osalta. Ohjelmistokontin Ubuntu-jakelu sisälsi oletuksena

vähemmän fontteja kuin alustakoneen Ubuntu-jakelu. Fonttien eriyttämisellä voidaan poistaa tiedon yksilöitävyyttä asentamalla ohjelmistokontin sisälle vain selaimen ja www-sivustojen näyttämiseen tarvittavat fontit.

Testissä ei pystytty suoraan vaikuttamaan näytönohjaimen valmistajaa ja mallia kuvaaviin tietoihin, mutta niille saatiin kuitenkin eri tunnisteet. Ohjelmistokontin selain ilmoitti näytönohjaimen malliksi ”Gallium 0.4 on NV124” kun alustakoneen selain ilmoitti pelkästään ”NV124”. Ohjelmistokontin selain ilmoitti siten yksityiskohtaisempaa tietoa. Tulokseen vaikutti Ubuntu-jakelujen eri versiot alustakoneen ja ohjelmistokontin välillä. Alustakoneessa käytettiin Ubuntu-jakelun kehitysversiota, kun taas ohjelmistokontin jakelu oli Ubuntu-jakelun viimeisin vakaa julkaisu. Jakelut käyttävät eri järjestelmäkirjastojen versioita, mikä aiheutti testissä sen, että vakaa Ubuntu-jakelu ilmoitti yksityiskohtaisemman näytönohjaimen mallin. Tieto näytönohjaimen mallista on kerättävissä käyttämällä selaimen WebGL-ohjelmointirajapintaa, joka edelleen käyttää järjestelmäkirjastoa tiedon haussa. Ohjelmistokontin luonnissa jaettiin alustakoneen näytönohjain ohjelmistokontin prosessien käyttöön selaimen laitteistokiihdytyksen mahdollistamiseksi. Siten molemmat selaimet näkevät saman näytönohjaimen. Käyttöjärjestelmäavusteinen virtualisointi ei toteuta laitteiston virtualisointia, joten näytönohjaimen tietojen muuttaminen ei ole mahdollista ohjelmistokontin sisällä nimiavaruuksien avulla.

Ohjelmistokontissa suoritetun selaimen canvas-elementin renderöintidata poikkesi alustakoneen vastaavasta tiedosta. Ero renderöintidatassa syntyi testissä fonttien piirtämisessä canvas-elementiin. Canvas-elementin renderöintidatat olivat samat, kun fonttien piirtäminen jätettiin tekemättä. Testissä käytetty fontti oli asennettuna molempiin suoritusympäristöihin. Ero syntyi myös käyttämällä ohjelmistokontin jakelussa samaa Ubuntu-jakelun versiota kuin alustakoneessa. Ero fonttien piirtämisessä voi selittyä fonttien eri ohjelmistokontin ja alustakoneen renderöinninasetuksista.

Testin tulosten perusteella käyttöjärjestelmäavusteisilla virtualisointiteknologioilla voidaan vastata osaan selaimen sormenjälkitunnistamisen torjunnan haasteista. Toisin kuin selaimen liitännäiset ja selaimen lähdekoodin suoritusympäristön suoramuokkaaminen, käyttöjärjestelmäavusteinen virtualisointi ei rajoita torjuntamenetelmän käyttämistä torjuntamenetelmän toteutusympäristön mukaan, vaan sitä voidaan käyttää selaimesta riippumatta. Käyttöjärjes-

telmäavusteinen virtualisointi mahdollistaa selaimen suoritusympäristön virtualisoinnin, jota voidaan käyttää selaimesta riippumattoman torjuntamenetelmän kehittämiseen. Käyttöjärjestelmäavuasteisilla virtualisoinnilla voidaan lisäksi estää torjunnasta muodostuvien ristiriitojen aiheuttamista. Muutokset suoritusympäristön kokoonpanotietoihin toteutetaan globaalisti koko suoritusympäristöön sen sijaan, että tiedon muuttaminen tehtäisiin selaimen ohjelmointirajapintoja manipuloimalla.

Käyttöjärjestelmäavusteisilla virtualisointimenetelmillä ei voida kuitenkaan vastata torjuntamenetelmien yleiseen puutteellisuuteen. Testissä ei pystytty vaikuttamaan kaikkiin testiin valittuihin yhdeksään käyttäjän suoritusympäristön kokoonpanoa kuvaaviin tietoihin. Lisäksi testin ulkopuolelle jäivät selaimen omien kokoonpanotietojen käyttäminen ja passiivisesti kerättävät tiedot selaimen HTTP-pyynnöistä. Käyttöjärjestelmäavusteisilla virtualisoinnilla ei voida myöskään välttää torjunnasta aiheutuvia haittavaikutuksia, jotka johtuvat selaimen sormenjälkitunnistamisen torjunnassa käytetyistä tiedon yksilöitävyyden poistamiseen ja tiedon keräämisen estämiseen perustuvisdta torjuntamenetelmistä.

8 Yhteenveto

Tutkielmassa selvitettiin käyttöjärjestelmäavusteisten virtualisointiteknologioiden käyttämisen mahdollisuutta selaimen sormenjälkitunnistamisen torjunnassa. Tutkielma keskittyi Linux-kernelin tukemiin käyttöjärjestelmäavusteisiin virtualisointiteknologioihin, jotka mahdollistavat ohjelmistokonttien toteuttamisen. Tutkielmassa osoitettiin, että käyttöjärjestelmäavusteisella virtulisoinnilla voidaan muuttaa selaimen sormenjälkitunnistetta, jonka luomisessa on käytetty selaimen suoritusympäristön kokoonpanotietoja.

Tutkielmassa testattiin yhdeksään suoritusympäristön kokoonpanoa kuvaavan tietoon vaikuttamista selaimen suoritusympäristön virtualisoimisen avulla. Selaimen suoritusympäristön virtualisoinnissa käytettiin LXD-hallintatyökalua selaimen ja sen riippuvuuksien paketoimisessa ohjelmistokonttiin. Viidelle testissä kerätylle kokoonpanotiedolle saatiin muuttuneet arvot, joista kolmeen voitiin suoraan vaikuttaa ohjelmistokontin suoritusympäristöön vaikuttamalla.

Testissä ei pystytty vaikuttamaan käyttöjärjestelmän alustaa, laitteistoa, sekä selaimen liitännäisiä koskeviin tietoihin. Käyttöjärjestelmän alustan kuvaavien tietojen muuttaminen katsottiin olevan mahdotonta nykyisillä Linux-kernelin tukemilla nimiavaruuksilla. Laitteistoa kuvaaviin tietoihin ei voitu vaikuttaa, koska laitteiston virtualisointia ei voida toteuttaa käyttöjärjestelmäavusteisella virtualisoinnilla. Selaimen liitännäisiä kuvaavaan tietoon ei pystytty muuttamaan, koska tuki NPAPI-liitännäisiltä oli päättynyt testissä käytetyssä selaimessa. NPAPI-liitännäisten tuen päättymisen todettiin auttavan vähentämään käyttäjien yksilöitävyyttä.

Tutkielmassa tunnistettiin neljä eri haastetta selaimen sormenjälkitunnistamisen torjuntaan, joista kahteen katsottiin käyttöjärjestelmäavusteisella virtualisoinnilla pystyttävän vastaamaan. Käyttöjärjestelmäavusteisella virtualisoinnilla todettiin voitavan välttää torjuntamenetelmän käytöstä aiheutuvia ristiriitoja ja poistamaan torjuntamenetelmän toteutusympäristön asettamia rajoitteita. Käyttöjärjestelmäavusteisten virtualisoinnin käyttämisellä ei katsottu olevan etua selaimen sormenjälkitunnistamisen torjunnan yleiseen puutteellisuuteen ja torjuntamenetelmien käytöstä aiheuttamien haittavaikutuksien välttämiseen.

Testissä kerättävien tiedot käsittivät vain käyttäjän suoritusympäristön kuvaavia tietoja. Siten testin ulkopuolelle jäivät selaimen kokoonpanoa kuvaavat tiedot, sekä passiivisesti kerättävät tiedot. Siksi tutkielmassa ei pystytty arvioimaan sitä, kuinka merkittävästi käyttöjärjestelmäavusteisella virtualisoinnilla voidaan torjua käyttäjien laajamittaista seurantaa.

Jatkotoimenpiteenä tulisi selvittää pystytäänkö ohjelmistokonttiteknologioita hyödyntämällä laajentaa sormenjälkitunnistamisen torjuntaa koskemaan tietoja, jotka jätettiin testin ulkopuolelle. Lisäksi tulisi testata voidaanko Linux-kernelin versiosta 4.10 lisättyä tukea virtuaalisten näytönohjaimien lisäämiselle käyttää ohjelmistokontin näytönohjaimen jakamisessa. Tällä voitaisiin vaikuttaa ohjelmistokontin selaimen ilmoittamiin näytönohjainta kuvaaviin tietoihin, jolla voitaisiin estää käyttäjien yksilöiminen näytönohjaimen kokoonpanon avulla.

Lähteet

Acar, Gunes, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens ja Bart Preneel. 2013. “FPDetective”. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS ’13:* 1129–1140. ISSN: 15437221. doi:10.1145/2508859.2516674. http://dl.acm.org/citation.cfm?doid=2508859.2516674.

Aggarwal, Gaurav, Elie Bursztein, Collin Jackson ja Dan Boneh. 2010. “An Analysis of Private Browsing Modes in Modern Browsers”. Teoksessa *Proceedings of the 19th USENIX Conference on Security,* 6. USENIX Security’10. Berkeley, CA, USA: USENIX Association. http://dl.acm.org/citation.cfm?id=1929820.1929828.

Baumann, Peter, Stefan Katzenbeisser, Martin Stopczynski ja Erik Tews. 2016. “Disguised Chromium Browser: Robust Browser, Flash and Canvas Fingerprinting Protection”: 37–46. doi:10.1145/2994620.2994621.

Blakemore, Christine, Joao Redol ja Miguel Correia. 2016. “Fingerprinting for Web Applications: from Devices to Related Groups”. *History* 7 (3): 8. doi:10.1109/TrustCom/BigDataSE/ISPA.2016.56.

Boda, Karoly, Adam Mate Foeldes, Gabor Gyoergy Gulyas ja Sandor Imre. 2012. “User Tracking on the Web via Cross-Browser Fingerprinting”. *Information Security Technology for Applications* 7161:31–46. ISSN: 03029743. doi:10.1007/978-3-642-29615-4.

Braden, R. 1989. *Requirements for Internet Hosts - Communication Layers.* RFC 1122. doi:http://dx.doi.org/10.17487/RFC1122. arXiv: arXiv:1011.1669v3. https://tools.ietf.org/html/rfc1122.

Broenink, Ralph. 2012. “Using browser properties for fingerprinting purposes”: 169–176.

Cafaro, Massimo, ja Giovanni Aloisio. 2011. *Grids, Clouds, and Virtualization,* toimittanut toimittaja Aloisio Giovanni, 1–21. Computer Communications and Networks. London: Springer London. ISBN: 978-0-85729-049-6. doi:10.1007/978-0-85729-049-6. arXiv: arXiv:1011.1669v3. http://dx.doi.org/10.1007/978-0-85729-049-6%7B%5C_%7D1%7B%5C%%7D5Cnhttp://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract.

Calzavara, Stefano, Alvise Rabitti ja Michele Bugliesi. 2016. "Content Security Problems?: Evaluating the Effectiveness of Content Security Policy in the Wild". Teoksessa *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,* 1365–1375. CCS '16. New York, NY, USA: ACM. ISBN: 978-1-4503-4139-4. doi:10.1145/2976749.2978338. http://doi.acm.org/10.1145/2976749.2978338.

Canonical Ltd. 2017. *What's LXD?* Viitattu 2. syyskuuta 2017. https://linuxcontainers.org/lxd/.

Chiueh, Susanta Nanda Tzi-cker, ja Stony Brook. 2005. "A Survey on Virtualization Technologies", numero Vm: 1–42.

Dittner, Rogier, ja David Rule. 2007. *The Best Damn Server Virtualization Book Period,* toimittanut David Rule, 795–822. Burlington, MA: Syngress. ISBN: 9781597492171. doi:10.1016/B978-1-59749-217-1.00022-8. http://www.sciencedirect.com/science/article/pii/B9781597492171000228.

Eckersley, Peter. 2010. "How unique is your web browser?" Teoksessa *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* nide 6205 LNCS, 1–18. PETS'10. Berlin, Heidelberg: Springer-Verlag. ISBN: 3642145264. doi:10.1007/978-3-642-14527-8_1. http://dl.acm.org/citation.cfm?id=1881151.1881152%20http://link.springer.com/10.1007/978-3-642-14527-8%7B%5C_%7D1.

Eder, Michael, ja Holger Kinkelin. 2016. "Hypervisor- vs. Container-based Virtualization". *Network Architectures and Services,* numero July: 1–7. doi:10.2313/NET-2016-07-1.

Englehardt, Steven, ja Arvind Narayanan. 2016. “Online Tracking: A 1-million-site Measurement and Analysis”. Teoksessa *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16,* 1388–1401. CCS '16 1. New York, NY, USA: ACM. ISBN: 9781450341394. doi:10.1145/2976749.2978313. http://dl.acm.org/citation.cfm?doid=2976749.2978313.

European Communities. 2002. “DIRECTIVE 2002/58/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 12 July 2002”. *Official Journal of the European Communities* 45 (L 201): 11. http://data.europa.eu/eli/dir/2002/58/oj.

Fielding, R, J Gettys, J Mogul, H Frystyk, L Masinter, P Leach ja T Berners-Lee. 1999. *Hypertext Transfer Protocol – HTTP/1.1.* United States.

Fifield, David, ja Serge Egelman. 2015. “Fingerprinting Web Users Through Font Metrics”. Teoksessa *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers,* toimittanut Rainer Böhme ja Tatsuaki Okamoto, 107–124. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-662-47854-7. doi:10.1007/978-3-662-47854-7_7. http://dx.doi.org/10.1007/978-3-662-47854-7%7B%5C_%7D7.

Hämäläinen, Pentti Sakari. 2012. *Käyttöjärjestelmät 2012 -luentomateriaali.*

Klaper, Janne. 2015. “Tietoturvan huomioiminen maksujärjestelmässä”. Diplomityö, Tampereen teknillinen yliopisto. http://dspace.cc.tut.fi/dpub/handle/123456789/23928.

Kohno, T, A Broido ja K C Claffy. 2005. “Remote physical device fingerprinting”. *IEEE Transactions on Dependable and Secure Computing* 2 (2): 93–108. ISSN: 1545-5971. doi:10.1109/TDSC.2005.26.

Kristol, David M., ja Lou Montulli. 1997. “HTTP State Management Mechanism”. https://www.ietf.org/rfc/rfc2109.txt.

Laan, Sjaak. 2013. *It Infrastructure Architecture - Infrastructure Building Blocks and Concepts Second Edition.* 2nd ed. 436. Kustannuspaikka tuntematon: Lulu Press. ISBN: 1291250794. https://books.google.com/books?id=YrcWRIdYFscC%7B%5C&%7Dpgis=1.

Laperdrix, Pierre. 2017. *Am I Unique ?* Viitattu 17. syyskuuta 2017. https://github.com/DIVERSIFY-project/amiunique.

Laperdrix, Pierre, Walter Rudametkin ja Benoit Baudry. 2016. "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints". *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016:* 878–894. doi:10.1109/SP.2016.57.

Linux man-pages. 2016. California. https://www.kernel.org/doc/man-pages/.

Liu, Xiaofeng, ja Xiaoxi Wang. 2016. "Fingerprinting Web Browser for Tracing Anonymous Web Attackers". doi:10.1109/DSC.2016.78.

Luangmaneerote, Sakchan, Ed Zaluska ja Leslie Carr. 2016. "Survey of existing fingerprint countermeasures". http://eprints.soton.ac.uk/401902/.

Lukas, J, J Fridrich ja M Goljan. 2006. "Digital camera identification from sensor pattern noise". *IEEE Transactions on Information Forensics and Security* 1 (2): 205–214. ISSN: 1556-6013. doi:10.1109/TIFS.2006.873602.

Mozilla Contributors. 2017. *Web APIs.* Viitattu 17. syyskuuta 2017. https://developer.mozilla.org/en-US/docs/Web/API.

Nikiforakis, Nick, Wouter Joosen ja Benjamin Livshits. 2015. "PriVaricator: Deceiving Fingerprinters with Little White Lies". Teoksessa *Research.Microsoft.Com,* 820–830. ISBN: 9781450334693. doi:10.1145/2736277.2741090. http://research.microsoft.com/en-us/um/people/livshits/papers%7B%5C%%7D5Ctr%7B%5C%%7D5Cprivaricator.pdf.

Nikiforakis, Nick, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens ja Giovanni Vigna. 2013. “Cookieless monster: Exploring the ecosystem of web-based device fingerprinting”. *Proceedings - IEEE Symposium on Security and Privacy:* 541–555. ISSN: 10816011. doi:10.1109/SP.2013.43.

Olejnik, Lukasz, Gunes Acar, Claude Castelluccia ja Claudia Diaz. 2016. “The leaking battery: A privacy analysis of the HTML5 battery status API”. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9481:254–263. ISSN: 16113349. doi:10.1007/978-3-319-29883-2_18.

Open Container Initiative. 2017a. *About The Open Container Initiative (OCI).* Viitattu 4. elokuuta 2017. https://www.opencontainers.org/about.

———. 2017b. *Image Format Specification.* Viitattu 4. elokuuta 2017. https://github.com/opencontainers/image-spec/blob/master/spec.md.

———. 2017c. *Linux Container Configuration.* Viitattu 4. elokuuta 2017. https://github.com/opencontainers/runtime-spec/blob/master/config-linux.md.

———. 2017d. *Open Container Initiative Runtime Specification.* Viitattu 4. elokuuta 2017. https://github.com/opencontainers/runtime-spec/blob/master/spec.md.

Portnoy, Matthew. 2012. *Virtualization Essentials.* 286. Indianapolis, IN: John Wiley & Sons, Inc. ISBN: 9781118240175. doi:10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3. http://www.regensburger-katalog.de/InfoGuideClient.ubrsis/start.do?Login=igubr%7B%5C&%7DQuery=540=%7B%5C%%7D22978-1-118-24017-5%7B%5C%%7D22%7B%5C%%7D5Cnhttp://proquest.tech.safaribooksonline.de/9781118240175.

Red Hat. 2017a. *Chapter 1. Introduction to Linux Containers.* Viitattu 8. elokuuta 2017. https://access.redhat.com/documentation/en-us/red%7B%5C_%7Dhat%7B%5C_%7Denterprise%7B%5C_%7Dlinux%7B%5C_%7Datomic%7B%5C_%7Dhost/7/html/overview%7B%5C_%7Dof%7B%5C_%7Dcontainers%7B%5C_%7Din%7B%5C_%7Dred%7B%5C_%7Dhat%7B%5C_%7Dsystems/introduction%7B%5C_%7Dto%7B%5C_%7Dlinux%7B%5C_%7Dcontainers.

———. 2017b. *Resource Management Guide.* Viitattu 10. elokuuta 2017. https://access.redhat.com/documentation/en-US/Red%7B%5C_%7DHat%7B%5C_%7DEnterprise%7B%5C_%7DLinux/7/html-single/Resource%7B%5C_%7DManagement%7B%5C_%7DGuide/index.html.

Reshetova, Elena, Janne Karhunen, Thomas Nyman ja N Asokan. 2014. "Security of OS-Level Virtualization Technologies". Teoksessa *Secure IT Systems: 19th Nordic Conference, NordSec 2014, Troms{ø}, Norway, October 15-17, 2014, Proceedings,* toimittanut Karin Bernsmed ja Simone Fischer-Hübner, 77–93. Cham: Springer International Publishing. ISBN: 978-3-319-11599-3. doi:10.1007/978-3-319-11599-3_5. http://dx.doi.org/10.1007/978-3-319-11599-3%7B%5C_%7D5.

Rosen, Rami. 2013. *Resource management: Linux kernel Namespaces and cgroups.* http://www.haifux.org/lectures/299/netLec7.pdf.

Saito, Takamichi, Kazushi Takahashi, Koki Yasuda, Takayuki Ishikawa, Ko Takasu, Tomotaka Yamada, Naoki Takei ja Rio Hosoi. 2016. "OS and Application Identification by Installed Fonts". *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA):* 684–689. doi:10.1109/AINA.2016.55. http://ieeexplore.ieee.org/document/7474155/.

Scheepers, Mathijs Jeroen. 2014. "Virtualization and Containerization of Application Infrastructure : A Comparison". *21st Twente Student Conference on IT:* 1–7.

Smedberg, Benjamin. 2015. *NPAPI Plugins in Firefox.* Mozilla. Viitattu 22. toukokuuta 2017. https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/.

Torres, Christof Ferreira, Hugo Jonker ja Sjouke Mauw. 2015. “FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting”. Teoksessa *Computer Security – ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II,* toimittanut Günther Pernul, Peter Y A Ryan ja Edgar Weippl, 3–19. Cham: Springer International Publishing. ISBN: 978-3-319-24177-7. doi:10.1007/978-3-319-24177-7_1. http://dx.doi.org/10.1007/978-3-319-24177-7%7B%5C_%7D1.

Turek, Brian. 2017. *jsSHA.* Viitattu 17. syyskuuta 2017. https://caligatio.github.io/jsSHA/.

Unger, T, M Mulazzani, D Frühwirt, M Huber, S Schrittwieser ja E Weippl. 2013. “SHPF: Enhancing HTTP(S) Session Security with Browser Fingerprinting”. Teoksessa *2013 International Conference on Availability, Reliability and Security,* 255–261. doi:10.1109/ARES.2013.33.

Vajapeyam, S. 2014. “Understanding Shannon’s Entropy metric for Information”. *ArXiv e-prints.* arXiv: 1405.2061.

Vase, Tuomas. 2016. *Integrating Docker To a Continuous Delivery Pipeline – a Pragmatic Approach.* http://urn.fi/URN:NBN:fi:jyu-201701181181.

Viestintävirasto. 2017. *Evästeet.* Viitattu 18. huhtikuuta. https://www.viestintavirasto.fi/kyberturvallisuus/palveluidenturvallinenkaytto/evasteet.html.

Xenitellis, Simos. 2017. *How to run graphics-accelerated GUI apps in LXD containers on your Ubuntu desktop.* Viitattu 2. syyskuuta 2017. https://blog.simos.info/how-to-run-graphics-accelerated-gui-apps-in-lxd-containers-on-your-ubuntu-desktop/.

Zeldovich, Nickolai. 2014. *6.858 Computer Systems Security - Lecture Notes 18, Private Browsing.* https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-858-computer-systems-security-fall-2014.

Zhao, B, ja P Liu. 2015. “Private Browsing Mode Not Really That Private: Dealing with Privacy Breach Caused by Browser Extensions”. Teoksessa *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks,* 184–195. doi:10.1109/DSN.2015.18.

Liitteet

A LXD-työkalulla luodun Firefox-selaimen paketoivan ohjelmistokontin asetukset

```
1  architecture: x86_64
2  config:
3    image.architecture: amd64
4    image.description: ubuntu 16.04 LTS amd64 (release) (20170815.1)
5    image.label: release
6    image.os: ubuntu
7    image.release: xenial
8    image.serial: "20170815.1"
9    image.version: "16.04"
10   raw.idmap: both 1000 1000
11   volatile.base_image: 58f90cbf68927c3fc43e6ee1386446a04f3d8068c1a75a291339cb2be01dec08
12   volatile.eth0.hwaddr: 00:16:3e:23:b3:85
13   volatile.idmap.base: "0"
14   volatile.idmap.next: '[{"Isuid":true,"Isgid":false,"Hostid":165536,"Nsid":0,"Maprange":1000},{"Isuid":
         true,"Isgid":true,"Hostid":1000,"Nsid":1000,"Maprange":1},{"Isuid":true,"Isgid":false,"Hostid
         ":166537,"Nsid":1001,"Maprange":64535},{"Isuid":false,"Isgid":true,"Hostid":165536,"Nsid":0,"
         Maprange":1000},{"Isuid":true,"Isgid":true,"Hostid":1000,"Nsid":1000,"Maprange":1},{"Isuid":false
         ,"Isgid":true,"Hostid":166537,"Nsid":1001,"Maprange":64535}]'
15   volatile.last_state.idmap: '[{"Isuid":true,"Isgid":false,"Hostid":165536,"Nsid":0,"Maprange":1000},{"Isuid
         ":true,"Isgid":true,"Hostid":1000,"Nsid":1000,"Maprange":1},{"Isuid":true,"Isgid":false,"Hostid
         ":166537,"Nsid":1001,"Maprange":64535},{"Isuid":false,"Isgid":true,"Hostid":165536,"Nsid":0,"
         Maprange":1000},{"Isuid":true,"Isgid":true,"Hostid":1000,"Nsid":1000,"Maprange":1},{"Isuid":false
         ,"Isgid":true,"Hostid":166537,"Nsid":1001,"Maprange":64535}]'
16   volatile.last_state.power: RUNNING
17 devices:
18   X0:
19     path: /tmp/.X11-unix/X0
20     readonly: "true"
21     source: /tmp/.X11-unix/X0
22     type: disk
23   Xauthority:
```

```
      path: /home/jupearmo/.Xauthority
      readonly: "true"
      source: /home/jupearmo/.Xauthority
      type: disk
    gpu:
      gid: "1000"
      type: gpu
      uid: "1000"
ephemeral: false
profiles:
- default
```

B JavaScript-ohjelmointikielellä toteutettu skripti selaimen suoritusympäristön kokoonpanotietojen mittaamiseen

```
function Fingerprint() {
    this.userAgent = navigator.userAgent;
    this.language = navigator.language;
    this.platform = navigator.platform;
    this.plugins = this.getPlugins();
    this.timezone = new Date().getTimezoneOffset();
    this.screen = this.getScreenInfo();
    this.WebGL = this.getWebGLInfo();
    this.fonts = this.getFonts();
    this.canvas = this.getCanvasData();
}

Fingerprint.prototype.getPlugins = function () {
    let plugins = [];
    for (let i = 0; i < navigator.plugins.length; i++) {
        let plugin = {};
        plugin.name = navigator.plugins[i].name;
        plugin.filename = navigator.plugins[i].filename;
        plugin.description = navigator.plugins[i].description;
        plugin.version = navigator.plugins[i].version ? navigator.plugins[i].version : "";
        plugins.push(plugin);
```

```
22        }
23        return plugins;
24    };
25
26    Fingerprint.prototype.getScreenInfo = function () {
27        let w = screen.width;
28        let h = screen.height;
29        let r = window.devicePixelRatio;
30        if (window.matchMedia("(orientation: portrait)").matches) {
31            let t = w;
32            w = h;
33            h = t;
34        }
35        return Math.ceil(w * r) + "x" + Math.ceil(h * r) + "," + screen.colorDepth;
36    };
37
38    Fingerprint.prototype.getWebGLInfo = function () {
39        try {
40            let canvas = document.getElementById('canvas');
41            let gl = canvas.getContext('webgl');
42            let debugInfo = gl.getExtension('WEBGL_debug_renderer_info');
43            let vendor = gl.getParameter(debugInfo.UNMASKED_VENDOR_WEBGL);
44            let renderer = gl.getParameter(debugInfo.UNMASKED_RENDERER_WEBGL);
45            return {vendor: vendor, renderer: renderer};
46        } catch(e) {
47            return "Not supported";
48        }
49    };
50
51    Fingerprint.prototype.getFonts = function () {
52        let swf = document.getElementById("swf");
53        if (typeof swf.getFonts !== "undefined") {
54            let fonts = [];
55            for (let font of swf.getFonts()) {
56                fonts.push(font.fontName);
57            }
58            return fonts;
```

```
59      } else {
60          return "Not supported";
61      }
62  };
63
64  Fingerprint.prototype.getCanvasData = function () {
65      try {
66          let canvas = document.createElement("canvas");
67          let size = 256;
68          let text = "Hello world! \ud83d\ude03";
69          canvas.height = size;
70          canvas.width = size;
71          let ctx = canvas.getContext("2d");
72          ctx.fillStyle = "green";
73          ctx.beginPath();
74          ctx.arc(size / 2, size / 2, size / 4, 0, 2 * Math.PI);
75          ctx.fill();
76          ctx.fillStyle = "red";
77          ctx.font = "24px DejaVu Sans";
78          ctx.fillText(text, 0, 32);
79          ctx.fillStyle = "blue";
80          ctx.font = "32px DejaVu Serif";
81          ctx.fillText(text, 0, 100);
82          return canvas.toDataURL();
83      } catch (e) {
84          return "Not supported";
85      }
86  };
87
88  Fingerprint.prototype.toHash = function () {
89      return sha1(this);
90  };
91
92  function sha1(obj) {
93      let shaObj = new jsSHA("SHA-1", "TEXT");
94      shaObj.update(JSON.stringify(obj));
95      return shaObj.getHash("HEX");
```

```
}

function displayFingerPrint() {
    let fingerprint = new Fingerprint();
    let data = document.getElementById("data");
    data.innerHTML = "";
    for (let [key, value] of Object.entries(fingerprint)) {
        let dl = document.createElement("dl");
        let dt = document.createElement("dt");
        let dd_hash = document.createElement("dd");
        let dd_value = document.createElement("dd");
        dt.textContent = key;
        dd_hash.textContent = "Hash: " + sha1(value);
        dd_value.textContent = "Value: " + JSON.stringify(value);
        dl.appendChild(dt);
        dl.appendChild(dd_value);
        dl.appendChild(dd_hash);
        data.appendChild(dl);
    }
    let img = document.createElement("img");
    img.src = fingerprint.canvas;
    data.appendChild(img);
    document.getElementById("fingerprint").textContent = fingerprint.toHash();
}

function onFlashAppletLoaded() {
    displayFingerPrint();
}

document.addEventListener("DOMContentLoaded", function () {
    if (typeof navigator.mimeTypes["application/x-shockwave-flash"] === "undefined") {
        displayFingerPrint();
    }
});
```

C ActionScript-ohjelmointikielellä toteutettu skripti asenennettujen fonttien listaamiseen

```
1  package {
2
3  import flash.display.Sprite;
4  import flash.text.Font;
5  import flash.external.ExternalInterface;
6
7  public class Fonts extends Sprite {
8
9      public function Fonts() {
10         ExternalInterface.addCallback('getFonts', getFonts);
11         ExternalInterface.call('onFlashAppletLoaded', true);
12     }
13
14     public function getFonts(): Array {
15         return Font.enumerateFonts(true);
16     }
17 }
18 }
```

Skripti tulee kääntää SWF-tiedostoksi sen suorittamista varten selaimen Flash-liitännäisellä. Tämä voidaan tehdä esimerkiksi Apachen Flex SDK:n mxmlc-kääntäjällä.

D HTML-sivu selaimen sormenjälkitunnisteen näyttämiseen

```
1  <!DOCTYPE html>
2  <html lang="fi">
3  <head>
4    <meta charset="UTF−8">
5    <title>Sormenjalki</title>
6    <script src="sha1.js" type="application/javascript"></script>
7  </head>
8  <body>
9  <h2>Sormenjalktitunnisteesi: <i id="fingerprint"></i></h2>
10 <hr>
```

```
11 <h4>Keratyt tiedot</h4>
12 <div id="data"></div>
13 <canvas id="canvas"></canvas>
14 <embed id="swf" src="fontit.swf"></embed>
15 <script src="sormenjalki.js" type="application/javascript"></script>
16 </body>
17 </html>
```

E Testissä käytettyjen ohjelmien versiotiedot

Firefox-selain	55.0.2
Linux-kernel	4.12.0
LXC (liblxc-kirjasto)	2.0.8
LXD-hallintatyökalu	2.17
mxmlc-kääntäjä	4.16.0
Ubuntu-jakelu (alustakone)	17.10
Ubuntu-jakelu (ohjelmistokontti)	16.04

F Testissä mitatut Firefox-selaimen jättämät tunnistetiedot alustakoneessa

```
1 Sormenjalkitunniste: 01048da348c89cc5dc1fefd094ff3d1888b27303
2 −−−−−−−−
3 UserAgent: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0) Gecko/20100101 Firefox/55.0"
4 Language: "fi−FI"
5 Platform: "Linux x86_64"
6 Plugins: [{"name":"Shockwave Flash","filename":"libflashplayer.so","description":"Shockwave Flash 26.0 r0
      ","version":"26.0.0.151"}]
7 Timezone: −180
8 Screen: "1920x1080,24"
9 WebGL: {"vendor":"nouveau","renderer":"NV124"}
10 Fonts: ["Abyssinica SIL","Bitstream Charter","Century Schoolbook L","Courier 10 Pitch","DejaVu Sans","
      DejaVu Sans Mono","DejaVu Serif","Dingbats","Droid Sans
```

```
11 Fallback","FreeMono","FreeSans","FreeSerif","Garuda","KacstArt","KacstBook","KacstDecorative","
       KacstDigital","KacstFarsi","KacstLetter","KacstNaskh","KacstOffice","KacstOne","KacstPen","
       KacstPoster","KacstQurn","KacstScreen","KacstTitle","KacstTitleL","Khmer OS","Khmer OS System
       ","Kinnari","Laksaman","Liberation Mono","Liberation Sans","Liberation Sans Narrow","Liberation
12 Serif","LKLUG","Lohit Gurmukhi","Loma","mry_KacstQurn","NanumBarunGothic","NanumGothic","
       NanumMyeongjo","NanumSquare","NanumSquare Bold","Nimbus Mono L","Nimbus Roman No9 L","
       Nimbus Sans L","Norasi","Noto Mono","Noto Sans CJK JP Bold","Noto Sans CJK JP Regular","Noto
       Sans CJK KR Bold","Noto Sans CJK KR Regular","Noto Sans CJK SC Bold","Noto Sans CJK SC
13 Regular","Noto Sans CJK TC Bold","Noto Sans CJK TC Regular","Noto Sans Mono CJK JP Bold","Noto
       Sans Mono CJK JP Regular","Noto Sans Mono CJK KR Bold","Noto Sans Mono CJK KR Regular","
       Noto Sans Mono CJK SC Bold","Noto Sans Mono CJK SC Regular","Noto Sans Mono CJK TC Bold","
       Noto Sans Mono CJK TC Regular","Noto Serif CJK JP","Noto Serif CJK KR","Noto Serif CJK SC","
       Noto
14 Serif CJK TC","OpenSymbol","Padauk","Padauk Book","Phetsarath OT","Purisa","Saab","Sawasdee","
       Standard Symbols L","STIX","STIX
15 Math","STIXGeneral","STIXIntegralsD","STIXIntegralsSm","STIXIntegralsUp","STIXIntegralsUpD","
       STIXIntegralsUpSm","STIXNonUnicode","STIXSizeFiveSym","STIXSizeFourSym","
       STIXSizeOneSym","STIXSizeThreeSym","STIXSizeTwoSym","STIXVariants","Symbola","
       TakaoPGothic","Tibetan Machine Uni","Tlwg Mono","Tlwg Typewriter","Tlwg Typist","Tlwg Typo","
       Ubuntu","Ubuntu
16 Condensed","Ubuntu Light","Ubuntu Mono","Umpush","URW Bookman L","URW Chancery L","URW
       Gothic L","URW Palladio L","Waree"]
17 Canvas: 4955c2a9cadfbfbd21832d299cc81c159d40b21e
```

G Testissä mitatut Firefox-selaimen jättämät tunnistetiedot ohjelmistokontissa

```
1 Sormenjalktitunniste: 98f6929724f68123efdc7530eeba0ef475285ca9
2 ---------
3 UserAgent: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:55.0) Gecko/20100101 Firefox/55.0"
4 Language: "en-US"
5 Platform: "Linux x86_64"
6 Plugins: [{"name":"Shockwave Flash","filename":"libflashplayer.so","description":"Shockwave Flash 26.0 r0
       ","version":"26.0.0.151"}]
7 Timezone: 300
8 Screen: "1920x1080,24"
```

```
WebGL: {"vendor":"nouveau","renderer":"Gallium 0.4 on NV124"}
Fonts: ["DejaVu Sans","DejaVu Sans Mono","DejaVu Serif","My Font"]
Canvas: caf19cb68564b6005566c887eb85c427ee1a6c58
```