Matias Hynnä

Verkkovierailijan tunnistaminen ilman evästeitä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

23.5.2022

Tiivistelmä

Tekijä:	Matias Hynnä
Otsikko:	Verkkovierailijan tunnistaminen ilman evästeitä
Sivumäärä:	29 sivua
Aika:	23.5.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaaja:	Tutkijaopettaja Hannu Markkanen

Opinnäytetyön tarkoituksena oli perehtyä verkkosivuilla käytettyjen evästeiden toimintaan ja niiden eri käyttötarkoituksiin. Tavoitteena oli tutkia verkkosivujen evästeiden luomiseen kehitettyjä menetelmiä ja sitä, mitä eri tietoja evästeet keräävät käyttäjästä, miksi se on tärkeää nykypäivän internetissä ja miten käyttäjän on mahdollista vaikuttaa kerättäviin tietoihin. Opinnäytetyössä selvitettiin lisäksi vaihtoehtoisia menetelmiä käyttäjän tunnistamiseksi, koska tulevaisuudessa kolmannen osapuolen evästeiden käyttö poistuu selaimista.

Nykypäivänä evästeitä käytetään lähes joka verkkosivulla, ja tässä insinöörityössä perehdyttiin siihen, mihin erilaisia evästeitä käytetään ja miksi niiden käyttö on herättänyt huolta käyttäjän yksityisyydensuojasta. Evästeillä kerätään tietoja käyttäjästä käyttäjäkokemuksen parantamiseksi, mutta monet verkkosivut keräävät tietoja myös kohdennetun mainonnan ja taloudellisen hyödyn vuoksi. Huolenaihe käyttäjän yksityisyydestä on johtanut siihen, että monet julkaisijat ja mainostajat ovat pyrkineet valmistautumaan evästeettömään maailmaan.

Työhön valikoitui verkkovierailijan tunnistamiseksi kaksi erilaista tapaa. Ensimmäinen menetelmä hyödynsi avoimen lähdekoodin kirjastoa, jonka avulla vierailijalle luodaan yksilöllinen tunnus selainohjelman kautta saatavien tietojen perusteella. Toinen vaihtoehto oli tallentaa vierailijalle henkilökohtainen Etag-tunniste palvelimelle tapahtuvien http-pyyntöjen yhteydessä. Opinnäytetyö toimii oppaana kyseisten menetelmien implementoimiseksi sekä menetelmien heikkouksista ja vahvuuksista. Työn tulokset osoittivat, että käyttäjän tunnistaminen ei lopulta ole kovin tehokasta ja täsmällistä ilman evästeiden käyttöä.

Avainsanat: evästeet, verkkovierailijan tunnistaminen, verkkomainonta, GDPR

Abstract

Author:	Matias Hynnä
Title:	Identifying a web visitor without the use of cookies
Number of Pages:	29 pages
Date:	23 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Media Technology
Supervisor:	Hannu Markkanen, Researching Lecturer

The purpose of the thesis was to get acquainted with the cookies used on websites and their different uses. The aim was to study the methods developed for creating cookies on websites and what different information cookies collect about the user, why it is important on today's internet and how the user has the opportunity to influence the information collected. In addition, alternative methods for identifying the user were investigated in this thesis, because in the future the use of third-party cookies will be removed from browsers.

Today, cookies are used on almost every website, and this thesis looked at what different cookies are used for and why their use has raised concerns about the privacy of the user. Cookies collect information about the user to improve the user experience, but many websites also collect information for targeted advertising and financial gain. Concerns about user privacy have led many publishers and advertisers to strive to prepare for a cookie-free world.

Two different ways of identifying a website visitor were selected for the work. The first method utilized an open-source library to create a unique ID for a visitor based on information obtained through a browser program. Another option chosen was to make use of Etag identifiers while sending http-requests to the server. The thesis serves as a guide for the implementation of these methods and their weaknesses and strengths. The results of the work showed that user authentication is ultimately not very effective and accurate without the use of cookies.

Keywords: cookies, identifying a web visitor, online advertising, GDPR

Sisällys

Lyhenteet

GDPR: General Data Protection Regulation. Henkilötietojen käsittelyä säätelevä laki.

NPM: Node Package Manager. JavaScript-ohjelmointikielen paketinhallinta.

VPN: Virtual Private Network. Virtuaalinen erillisverkko.

JSONP: JSON with Padding. JavaScript-tekniikka tietojen pyytämiseen lataamalla script-elementti.

1 Johdanto

Evästeet ovat oleellinen osa nykypäivän internetin käyttäjäkokemusta. Ne mahdollistavat sulavampia ja henkilökohtaisempia kokemuksia verkkosivuilla ja auttavat verkkosivun kehittäjiä tarjoamaan kätevämpiä ratkaisuja ja kohdennetumpaa sisältöä käyttäjästä riippuen. Modernit verkkosivut ja -sovellukset sellaisina kuin ne nykyään tunnetaan, olisi mahdotonta toteuttaa ilman evästeitä.

Lähes kaikki verkkosivut käyttävät evästeitä, mutta se ei kuitenkaan tarkoita, että kaikki mahdolliset evästeet olisivat välttämättömiä tai tarpeellisia verkkosivun toiminnalle kuten kolmannen osapuolen evästeet, jotka ovat peräisin muualta kuin kyseisellä hetkellä vieraillulta verkkosivulta. Kolmannen osapuolen evästeiden käyttö on herättänyt huolta yksityisyydestä, koska niiden avulla eri tahot seuraavat jatkuvasti käyttäjien verkkotoimintaa ja keräävät suuria määriä tietoa käyttäjistä, usein täysin heidän tietämättään ja jopa ilman suostumusta.

Viranomaisten ja kuluttajien painostus on johtanut siihen, että selainsovellukset ja monet teknologia-alan yritykset ovat luopumassa kolmannen osapuolen evästeistä ja niiden seurauksena tapahtuvasta mainonnasta lopullisesti.

Insinöörityössä perehdytään evästeiden perusteisiin ja käyttötarkoituksiin sekä menetelmiin, joilla verkkosivun käyttäjä voidaan tunnistaa käyttämättä evästeitä. Työn tavoitteena on selvittää vaihtoehtoisten tapojen tehokkuutta käyttäjän tunnistamiseksi ilman evästeitä.

Työ tehdään helsinkiläiselle ohjelmistoalan pienyritykselle, joka erikoistuu myynnin, markkinoinnin ja asiakaspalvelun tehostamiseen verkkosivuille asennetuilla chat-ikkunoilla ja interaktioilla.

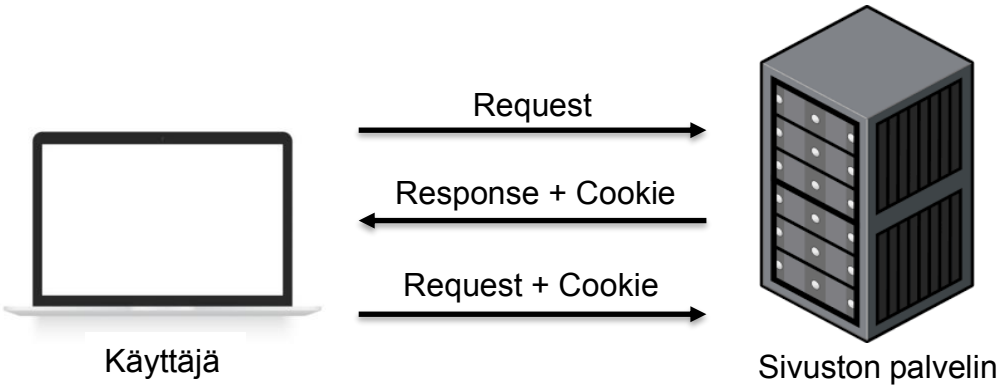
2 Evästeet

Eväste on pieni selainohjelman kautta tietokoneelle tallennettava tiedosto. Se koostuu yleensä lyhyestä tekstinpätkästä, joka tallennetaan käyttäjän laitteelle verkkosivuvierailun aikana (Setting Cookies in HTML 2019). Evästetiedostoon tallennetaan käyttäjästä tietoja, joiden avulla heidät voidaan jatkossa tunnistaa. Tieto, jota evästeeseen tallennetaan, ei kuitenkaan sisällä mitään käyttäjän henkilökohtaisia tietoja, vaan niiden tavoitteena on tunnistaa käyttäjän laite seuraavaa vierailukertaa varten paremman käyttökokemuksen tai esimerkiksi kohdennetun mainonnan tarjoamiseksi sekä verkkosivuston käyttäjiin perustuvan analytiikan keräämiseksi. Toisin sanoen, evästeiden avulla verkkosivu tunnistaa, onko käyttäjä käynyt sivulla aikaisemmin.

2.1 Evästeiden perusteet

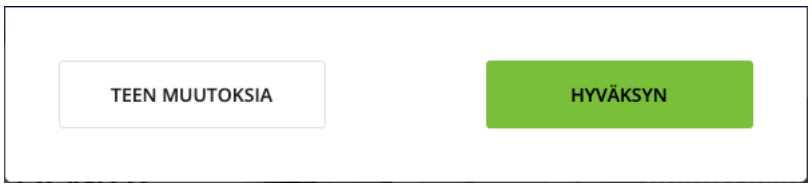
Evästetiedostot tallennetaan verkkosivujen palvelimille tehtyjen pyyntöjen yhteydessä. Verkkosivua tai jotain sen osaa ladatessa käynnistyy yhteyspyyntö (request), ja samalla selainohjelma tarkistaa, löytyykö kyseisen palvelimen evästeitä jo käyttäjän laitteelta. Tarkistuksen jälkeen selain lähettää olemassa olevan evästetiedoston palvelimelle pyynnön yhteydessä. Jos selaimesta ei löydy kyseisen palvelimen evästeitä, pyyntö lähtee ilman niitä, jolloin pyynnön vastaanottava palvelin tunnistaa, ettei evästeitä ole tallennettu ja täten pyytää lupaa evästeiden tallentamiseksi. (Brain 2007.)

Vastaanottaessaan pyynnön palvelin lähettää vastauksen (response) ja mahdollisesti tallentaa uusia evästeitä, jos niitä ei ole saapunut pyynnön mukana (kuva 1). Palvelimen vastauksissa hyödynnetään usein saapuneista evästeistä löytyvää dataa käyttäjän yksilöimiseksi esimerkiksi verkkosivun vierailijamäärän seurantaa. Muita evästeisiin tallennettuja tietoja voivat olla muun muassa käyttäjän suosima kieli tai verkkokaupassa ostoskorin sisältö. Eväste voi myös sisältää teknistä tietoa käyttäjän laitteesta.



Kuva 1: Evästeiden tallentaminen tai lähettäminen sekä request-response-ketju.

Evästeiden tallentamiseksi täytyy palvelimen pyytää siihen lupa, ja niiden hyväksymisessä voi olla useitakin vaihtoehtoja, jotka vaihtelevat sivustokohtaisesti. Suurin osa verkkosivuista tarjoaakin vähintään vaihtoehdot hyväksyä tai hylätä kaikki evästeet (kuva 2). On kuitenkin tärkeä muistaa, ettei verkkosivu välttämättä toimi oikein, jos kaikista evästeistä kieltäytyy. (Huttunen 2019.)



Kuva 2: Evästeiden hyväksyminen.

Osaa evästeistä kutsutaankin monesti pakollisiksi tai välttämättömiksi evästeiksi, ja ne varmistavat, että vierailtu verkkosivu toimii kunnolla. Näitä evästeitä voidaan kutsua myös toiminnallisiksi tai automaattisiksi evästeiksi. (Solla 2020.) Kaikki evästeet eivät ole kuitenkaan välttämättömiä, kuten esimerkiksi kuvassa 2 näkyvät tilastolliset ja markkinointiin hyödynnetyt evästeet. Suoraan kaikkien evästeiden hyväksymisen sijaan käyttäjällä on monesti mahdollisuus hallita näitä asetuksia ja valita yksitellen käyttötarkoituksen perusteella evästeet, joiden käytölle antaa suostumuksensa (kuva 3) (Gigantti verkkokauppa: 28).

Monet verkkosivut tekevät evästeiden valitsemisesta ja hallinnasta monimutkaista käyttäjälle. Kuten kuvassa 3 näkyy, evästeet on luokiteltu eri käyttötarkoitusten mukaisesti ja vaihtoehtoja on useita. Monelle käyttäjälle kyseiset termit ja käsitteet voivat myös olla tuntemattomia, joten kuvassa 2 näkyvä värikäs hyväksy-nappi houkuttelee painamaan sitä (Gigantti verkkokauppa).

Kuva 3: Evästeiden hyväksyminen Gigantti.fi-verkkokaupassa (Gigantti verkkokauppa).

Käyttötarkoituksen lisäksi selaimeen tallennettujen evästeiden kestoaika myös vaihtelee. Evästeet voivat olla niin sanottuja istuntokohtaisia evästeitä tai pitkäaikaisempia. Istuntokohtaiset evästeet lakkaavat toimimasta ja poistuvat käyttäjän sulkiessa selaimen, ja niiden avulla verkkosivut yleensä seuraavat käyttäjän selainistuntoa. Pitkäaikaiset tai pysyvät evästeet taas säilyvät tallessa evästeen lähettäjän määrittämän ajan tai kunnes käyttäjä poistaa ne itse (Morris 2020).

Tallennettuja evästeitä on mahdollista tarkastella ja poistaa manuaalisesti milloin tahansa selaimen asetuksista, ja se onkin suositeltavaa tehdä aika ajoin varsinkin, jos useammat henkilöt käyttävät samaa laitetta. Evästeasetuksista pääsee näkemään kaikki selaimeen tallennetut evästeet, niiden sisällön ja

alkuperän sekä päivän, jolloin ne poistuvat selaimesta (kuva 3). (What are Cookies? – How Cookies Work and What They Do.)

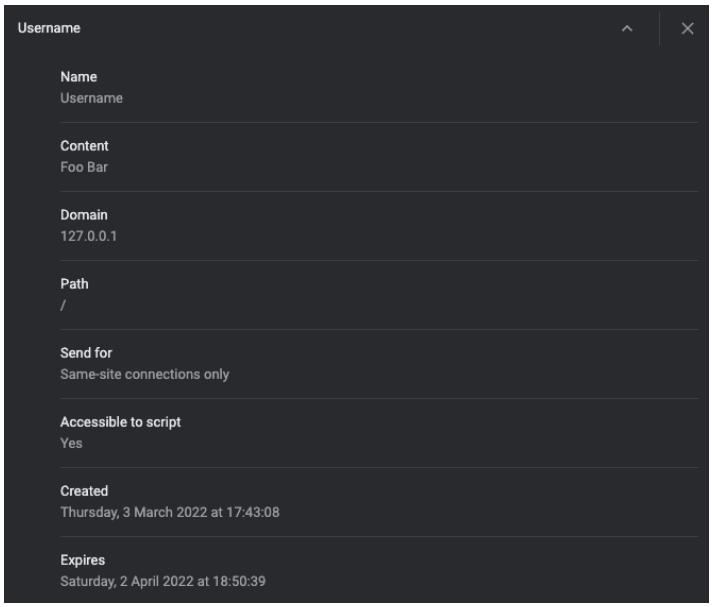
2.2 Evästeen luominen ja lukeminen

Evästeiden luomiseksi JavaScirptissä löytyy ohjelmointikieleen sisäänrakennettu ominaisuus, jota on kohtuullisen helppo hyödyntää (Mburu 2021: 29). JavaScriptin avulla evästeitä voidaan luoda ja hakea suhteellisen yksinkertaisilla esimerkkikoodien 1 ja 2 mukaisilla funktioilla.

```
function setCookie(name, value, daysToLive) {
    let cookie = name + "=" + value;
    cookie += "; max-age=" + (daysToLive*24*60*60);
    document.cookie = cookie;
}

setCookie("Username", "Foo Bar", 30);
```

Esimerkkikoodi 1. Evästeen luonti (Mburu 2021).

Esimerkkikoodissa 1 määritellyssä funktiossa eväste luodaan hyödyntäen ohjelmointikielen *document.cookie*-ominaisuutta. Funktiota kutsuttaessa sille annetaan tiedot evästeeseen tallennettavasta datasta, joka tässä esimerkissä on evästeen nimi, sisältö ja elinikä, mutta funktiota voidaan muokata tallentamaan erilaisia tietoja käyttötarkoituksen perusteella. Kuvasta 6 voidaan nähdä, miltä äsken luotu eväste näyttää selaimen evästeasetuksissa.



Kuva 4: Esimerkkievästeen sisältö.

Kun evästeitä on luotu, niiden arvoja voidaan hakea esimerkkikoodin 2 mukaisella funktiolla, joka hakee evästeitä nimen perusteella. Tässä esimerkissä funktion avulla haetaan esimerkkikoodissa 1 luotu ”Username”-eväste, joka sisältää arvon ”Foo Bar”. Todellisuudessa käyttäjänimi voisi olla esimerkiksi ohjelmallisesti luotu tai generoitu uniikki tunniste käyttäjästä.

```
function getCookie(name) {
    var cookieArr = document.cookie.split(";");
    for(let i = 0; i < cookieArr.length; i++) {
        var cookiePair = cookieArr[i].split("=");
        if(name === cookiePair[0].trim()) {
            return decodeURIComponent(cookiePair[1]);
        }
    }
    return null;
}

getCookie(“Username”);
```

Esimerkkikoodi 2. Evästeen lukeminen (Mburu 2021).

2.3 Ensimmäisen osapuolen evästeet

Evästeet voidaan jakaa ensimmäisen ja kolmannen osapuolen evästeisiin, mikä perustuu niiden alkuperään. Evästeiden soveltamisessa on kolme eri osapuolta.

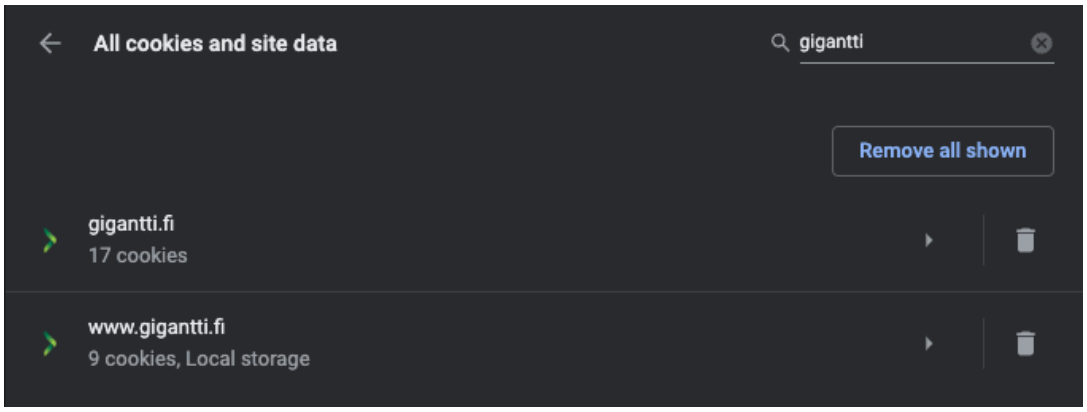
Ensimmäinen osapuoli on se verkkosivu, jolla kyseisellä hetkellä käyttäjä vierailee. Toinen osapuoli on käyttäjä itse, ja kolmannella osapuolella tarkoitetaan muita, ulkopuolisia verkkosivuja. Kolmannen osapuolen evästeet eivät siis ole peräisin siltä sivulta, jolla käyttäjä juuri nyt vierailee. Yleensä kolmas osapuoli on kohdennettuun mainontaan tai käyttäytymisen seurantaan erikoistunut yritys, joka saattaa myös myydä eteenpäin keräämäänsä dataa käyttäjistä. (Solla 2020.) Tyypillisimpiä kolmannen osapuolen kohdennetun mainonnan alustoja ovat esimerkiksi Facebook ja LinkedIn.

Ensimmäisen osapuolen evästeet ovat vierailtavan verkkosivun tarjoamia, jolloin ainoastaan kyseinen sivusto pääsee niihin käsiksi. Ne ovat oleellinen osa hyvää käyttäjäkokemusta, sillä esimerkiksi käyttäjän valitsema käyttökieli voidaan tallentaa, ja jatkossa verkkosivu tarjoaa sivuston käyttäjän valitsemalla kielellä, kysymättä sitä joka kerta uudelleen. Verkkokaupoissa käyttäjän ostoskorin sisältö tallentuu, jolloin koriin asetetut tuotteet pysyvät tallessa seuraavaa vierailukertaa varten. (Morris 2020.)

Esimerkkinä evästeistä tässä työssä käytetään Gigantin verkkokauppaa. Ennen Gigantin verkkosivulle siirtymistä verkkoselaimesta poistettiin kaikki kyseisen verkkotunnuksen evästeet. Verkkokaupan sivun avauduttua selaimeen ilmestyy kuvan 2 mukainen ikkuna, joka pyytää lupaa evästeiden hyväksymiseksi, koska verkkosivun palvelin ei löydä olemassa olevia Gigantti.fi:n evästeitä.

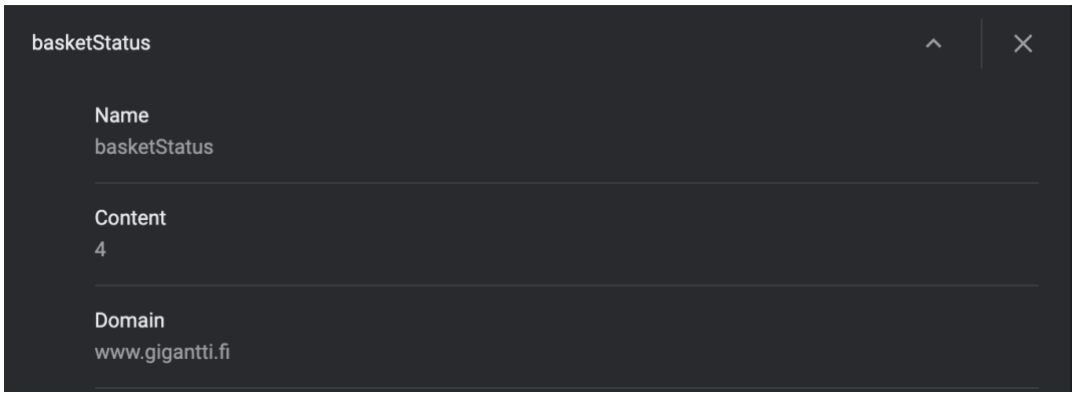
Verkkokaupasta valitaan tuotteita, ja niitä siirretään ostoskoriin yhteensä neljä kappaletta, minkä jälkeen verkkoselain suljetaan. Kun selain avataan uudelleen ja siirrytään takaisin verkkokaupan sivulle, tuotteet ovat yhä ostoskorissan ja ostoksia voi jatkaa siitä, mihin viime kerralla jäätiin. Jatkossa navigoidessa Gigantin verkkokauppaan evästeiden hyväksynnästä ei tule enää uutta lupapyyntöä, koska verkkosivun palvelin tunnistaa evästeiden olemassaolon ja on jo vastaanottanut ne sivua avatessa.

Tämän pienen esimerkkikokeilun jälkeen Google Chrome -selaimen evästeasetuksista löytyy Gigantti.fi-verkkotunnuksen tallentamat evästeet, joita löytyy nyt yhteensä 26 (kuva 5).



Kuva 5: Gigantti.fi-verkkotunnuksen tallentamat evästeet.

Evästeasetukset löytyvät Google Chromen asetuksien yksityisyys ja turvallisuus -osiosta. Kuvassa 6 näkyy ostoskoriin siirretyt tuotteet tallentava eväste.



Kuva 6: Esimerkki Gigantin evästeestä "basketStatus".

2.4 Kolmannen osapuolen evästeet

Kolmannen osapuolen evästeet ovat ensimmäisen osapuolen evästeiden vastaisesti peräisin muualta kuin kyseisellä hetkellä vieraillulta verkkosivulta, ja niillä seurataan yleensä käyttäjän liikkeitä eri verkkosivujen välillä. Monet

verkkosivut sisältävät omien evästeidensä lisäksi sivuillaan kolmannen osapuolen evästeitä. Verkkosivulla voi myös olla kolmannen osapuolen palveluita, jotka ladataan ulkopuolisilta palvelimilta, ja myös ne saattavat sisältää evästeitä. Tällöin evästeen sisältämä data ei ole sivustokohtaista, vaan verkkosivut, jotka käyttävät saman kolmannen osapuolen evästeitä, pystyvät myös jakamaan käyttäjiensä evästetietoja keskenänsä. (All you need to know about Third-Party-Cookies.) Käyttäjän vieraillessa useammilla verkkosivuilla, jotka kaikki sisältävät saman kolmannen osapuolen evästeitä, ulkopuolinen palvelin vastaanottaa evästeen, tunnistaa sen avulla käyttäjän ja alkuperän, josta eväste lähetetään, ja siten seuraa käyttäjän liikkeitä eri verkkosivustojen välillä. (Understanding and working with Third Party Cookies.)

Kolmannen osapuolen evästeet ovat usein mainostajien käyttämiä, ja niitä käytetään pääosin kohdennetun mainonnan luomiseen. Niiden keräämän tiedon perusteella käyttäjistä saadaan luotua hyvinkin tarkka profiili, joka perustuu esimerkiksi hakukoneella suoritettuihin hakuihin ja vierailtuihin verkkosivuihin. Profiilin perusteella käyttäjille voidaan suositella heidän kiinnostuksenkohteisiinsa perustuvia mainoksia, tavoitteena kasvattaa käyttäjän todennäköisyyttä kiinnostua ja täten myös ostaa kyseisiä tuotteita verkosta. Moni on varmasti törmännyt ilmiöön, jossa aikaisemmin selattuja verkkokaupan tuotteita ilmestyy mainoksissa eri sivustoilla. (All you need to know about Third-Party-Cookies.)

Yksinkertainen tapa ajatella asiaa on, että kolmannen osapuolen evästeitä tallennetaan silloin, kun verkkosivulle haetaan kolmannen osapuolen palvelua tai sisältöä (Understanding and working with Third Party Cookies). Yksi suosittu tapa on seurantapikseli eli tracking-pixel. Se on verkkosivulle ladattu olemattoman pieni ja näkymätön kuva, joka asetetaan verkkosivun lähdekoodiin. Kuva sisältää ulkoisen linkin palvelimelle, josta kuvan grafiikka ja sen sisältämät evästeet ladataan (Tracking Pixel). Seurantapikseli näyttää lähdekoodissa yleensä esimerkkikoodin 3 kaltaiselta.

```
<img height=0 width=0 src=" https://tracking_pixel_URL"> </img>
```

Esimerkkikoodi 3. Seurantapikseli HTML-koodissa (Tracking Pixel).

Kolmannen osapuolen evästeitä voidaan asettaa myös kolmannen osapuolen palvelua ladattaessa verkkosivulle. Otetaan esimerkiksi verkkosivu, joka käyttää sivullaan kolmannen osapuolen live-chat-palvelua. Chat-ikkunan lataaminen ulkopuoliselta palvelimelta voisi näyttää verkkosivun lähdekoodissa esimerkkikoodin 4 kaltaiselta.

```
<script src="https://chatti.com/js/livechat.js"> </script>
```

Esimerkkikoodi 4. Kolmannen osapuolen palvelu lähdekoodissa (All you need to know about Third-Party-Cookies).

Koodirivi ohjaa selaimen kyseiseen verkko-osoitteeseen ja hakee sen sisältämän palvelun verkkosivulle. Kun sivu ladataan, pyyntö käynnistyy. Selain saa vastauksen, joka sisältää pyydetyn live-chat-palvelun ja sen lisäksi mahdollisia evästeitä, joka voivat sisältää esimerkiksi tunnuksen käyttäjän tunnistamiseksi. Seuraavan kerran, kun käyttäjän selain hakee osoitteesta chatti.com sisältöä, saa palvelin vastauksena tunnisteen käyttäjästä.

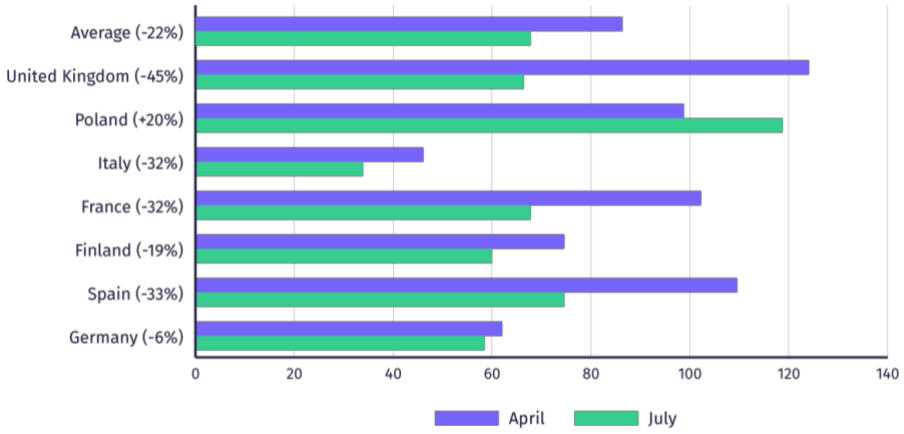
Todellisuudessa kolmannen osapuolen evästeen luomiseen käytetyt tekniikat voivat olla paljon monimutkaisempia. Mutta yleensä kaikki perustuu tässä luvussa esitettyjen esimerkkikoodien perusteisiin evästeen asettamisesta kolmannen osapuolen palvelua tai tiedostoa koskevien pyyntöjen aikana.

2.5 Evästeiden säätely

Mainostukseen käytetyt evästeet ovat lähes aina kolmannen osapuolen evästeitä, jotka eivät ole oleellisia verkkosivun toiminnalle. Evästeiden käytön kehittyessä myös huoli käyttäjien yksityisyydestä on kasvanut (Kupietzky 2021). Kun verkkokaupoissa selattuja tuotteita alkaa ilmestyä esimerkiksi Facebook-mainoksissa, se on selvä merkki siitä, että internetkäyttäytyminen ei ole enää niin anonyymiä kuin monet luulevat. Tämä huolenaihe on saanut aikaan evästeiden säätelyä ja lisännyt myös erilaisten tietosuojatyökalujen, kuten VPN-verkkojen ja mainostenestäjien kysyntää.

Evästeet luokitellaan Euroopan unionin lakien mukaan henkilötietoina ja henkilökohtaisena datana, mikä asettaa ne säätelyn alaisiksi. Euroopan unionin lisäksi muutamat Yhdysvaltain osavaltiot ja muut suuret lainkäyttöviranomaiset ovat sitä mieltä, että käyttäjien tulee olla tietoisia verkkosivun erilaisista evästeistä ja heillä on oltava mahdollisuus kieltäytyä niiden käytöstä. (Ramirez 2021.)

Uusien asetusten, lakien ja määräysten myötä hallitukset pyrkivät suojaamaan kansalaistensa yksityisyysoikeuksia. Vuonna 2018 tehdyn yleisen tietosuoja-asetuksen perusteella kaikki verkkosivut, joita EU:n asukkaat käyttävät, joutuvat pyytämään lupaa käyttäjiltä tallentaakseen mitä tahansa ei-välttämättömiä evästeitä käyttäjien laitteelle. Tarkoituksena on, että verkkosivun käyttäjällä on oikeus ja mahdollisuus käsitellä ja säädellä omien tietojensa käsittelyä. Tämän General Data Protection Regulation- eli GDPR-säätelyn käyttöönoton jälkeen toukokuussa 2018 kolmannen osapuolen evästeiden käyttö väheni välittömästi noin 22 prosenttia (kuva 7). (Ramirez: 2021.)



Kuva 7: Kolmannen osapuolen evästeiden määrän muutos vuonna 2018 (Ramirez 2021).

Nämä säännökset edellyttävät, että esimerkiksi verkkosivujen ylläpitäjä ilmoittaa käyttäjilleen, mitä tietoja heiltä kerätään ja kenelle nämä tiedot jaetaan, sekä mahdollisuuden kieltäytyä evästeiden tallentamisesta. Säätelystä huolimatta

monet sivustot eivät noudata näitä tietosuoja-asetuksia tietämättömyyden takia tai kieltäytyvät niistä tarkoituksellisesti. (Ramirez 2021.)

3 Vaihtoehtoisia tapoja tunnistaa käyttäjä ilman evästeitä

Evästeitä on käytetty viimeisen 20 vuoden ajan lähes kaikilla verkkosivuilla tunnistamaan verkkosivuvierailijoita, tarjoamaan yksilöllisempää sisältöä tai yksinkertaisesti seuraamaan käyttäjien toimintaa verkossa (5 Ways To Identify Your Users Without Using Cookies 2015). Tässä insinöörityössä perehdytään kahteen vaihtoehtoiseen menetelmään verkkovierailijan tunnistamiseksi. On kuitenkin olemassa muitakin melko kehittyneitä tekniikoita verkkosivua selaavan käyttäjän tunnistamiseen ilman evästeitä.

3.1 Verkkovierailijan tunnistamiseen olemassa olevia tekniikoita

Yksi ilmeisin ja kaikista yksinkertaisin ratkaisu olisi hyödyntää käyttäjän IP-osoitetta, mutta se on myös vähiten tehokas. IP-osoitteen käytön suurin ongelma on, että suuri osa käyttäjistä käyttää dynaamista IP-osoitetta, mikä tarkoittaa, että käyttäjän IP-osoite saattaa muuttua ajan myötä. Lisäksi, jos useat käyttäjät muodostavat yhteyden samaan verkkoon, kaikilla on sama IP-osoite, eivätkä ne ole palvelimen erotettavissa. (5 Ways To Identify Your Users Without Using Cookies 2015.)

Toinen menetelmä käyttäjän tunnistamiseksi on hyödyntää FingerprintJS-nimistä avoimen lähdekoodin kirjastoa. Selaimissa on paljon tietoa käyttäjän laitteesta ja ohjelmistosta, mikä saattaa aluksi vaikuttaa mitättömältä ja triviaalilta, mutta sitä voidaan teknisesti hyödyntää käyttäjien tunnistamiseen ja erotteluun. Tätä tapaa kutsutaan selaimen sormenjäljeksi, joka hakee selaimen ja päätelaitteen eri attribuutteja ja laskee niistä hajautetun vierailijatunnisteen. (Copland 2020.)

Kolmas vaihtoehto on käyttää hyödyksi palvelinta ja Etag-tunnistetta. Tunnisteen avulla palvelin osaa tunnistaa, onko verkkosivun käyttäjä vieraillut

sivulla aikaisemmin. Tekniikka hyödyntää selaimen välimuistiin tallennettua yksilöllistä Etag-tunnistetta, jonka palvelin tallentaa selaimeen silloin, kun käyttäjä vierailee sivustolla ensimmäistä kertaa. Myöhemmillä vierailukerroilla välimuistiin tallennettu tunniste voidaan lukea ja tunnistaa käyttäjä sen avulla. (Bhatia 2019.)

Neljäs tapa tunnistaa verkkosivun vierailija olisi hyödyntää tekoälyä ja koneoppimista käyttäjän käyttäytymisen seuraamiseen. Tämä menetelmä on kaikista monimutkaisin, ja se onkin vain joidenkin teknologiajättien, kuten Googlen, käytettävissä, joilla on resurssit sen toteuttamiseen. Tekniikka toimii tallentamalla käyttäjän käyttäytymistä, kuten hiiren liikkeitä, kiihdytystä ja rullauksen käyttöä, ja välittää nämä tiedot palvelimelle. Käyttäjän käyttäytyminen on hyvin henkilökohtaista, ja se voidaan havaita ryhmittelemällä koneoppimisalgoritmeja. (5 Ways To Identify Your Users Without Using Cookies 2015.)

Tähän insinöörityöhön valikoitui edellä esitetyistä menetelmistä FingerprintJS ja Etag-tunniste.

3.2 FingerprintJS-menetelmä

FingerprintJS-menetelmä kerää selaimesta ja laitteesta erilaisia tietoja ja kokoaa ne yhteen luodakseen yksilöllisen tunnisteen, jota kutsutaan sormenjäljeksi. Sormenjälki tunnistaa yksilölliset vierailijat, joita VPN-verkot, evästeiden estäjät ja muut tekniikat, joita käytetään internetselaamisen anonymisoimiseen, eivät estä. Tietoja, joita sormenjäljen luomiseen käytetään ovat

- käyttäjäagentin tiedot, kuten näytön koko, asennetut selaimet ja niiden versiot

- laitteiston tiedot, kuten näytön tarkkuus, laitteen muisti ja akun käyttö

- käytetyt selainlaajennukset sekä selaimen ja käyttöjärjestelmän asetukset

- käytettävissä olevat fontit ja laitteen maantieteellinen sijainti.

Jokaisen vierailijan saapuessa ensimmäisen kerran verkkosivulle, joka käyttää kyseistä sormenjälkikirjastoa, vierailija saa yksilöllisen tunnisteen, joka luodaan edellä mainituista tiedoista. (Copland 2020.)

FingerprintJS-kirjaston toteutus vierailijan tunnistamiseksi

Sormenjäljen testaamiseksi tässä työssä luodaan yksinkertainen React-verkkosovellus, johon FingerprintJS-kirjasto asennetaan. FingerprintJS ei laske sormenjälkiä itse selaimessa. Sen sijaan se käyttää kevyttä JavaScript-agenttia, joka kerää useita laitesignaaleja ja lähettää ne palvelimelle. Tässä toteutuksessa kirjasto asennetaan JavaScript-ohjelmointikielen paketinhallinnan, Node Package Managerin eli NPM:n avulla. (FingerprintJS Pro Introduction and quick start examples.)

```
npm install @fingerprintjs/fingerprintjs
```

Esimerkkikoodi 5. FingerprintJS-paketin asennus.

Kun sormenjälkikirjasto on asennettu, sen sisältämiä funktioita voidaan käyttää sormenjäljen luomiseksi. Esimerkkikoodissa 6 on yksinkertainen funktio, jonka sisällä FingerprintJS-agentti luodaan kutsumalla ”FingerprintJS.load(). Sormenjälki voidaan ladata agentin avulla kutsumalla sen sisältämää ”.get()” funktiota, jonka palauttaman arvon ”getFingerprint()”-funktio palauttaa.

```
import FingerprintJS from ”@fingerprintjs/fingerprintjs”

async function getFingerprint() {
     const fp = await FingerprintJS.load();
     const result = await fp.get();
     return result;
```

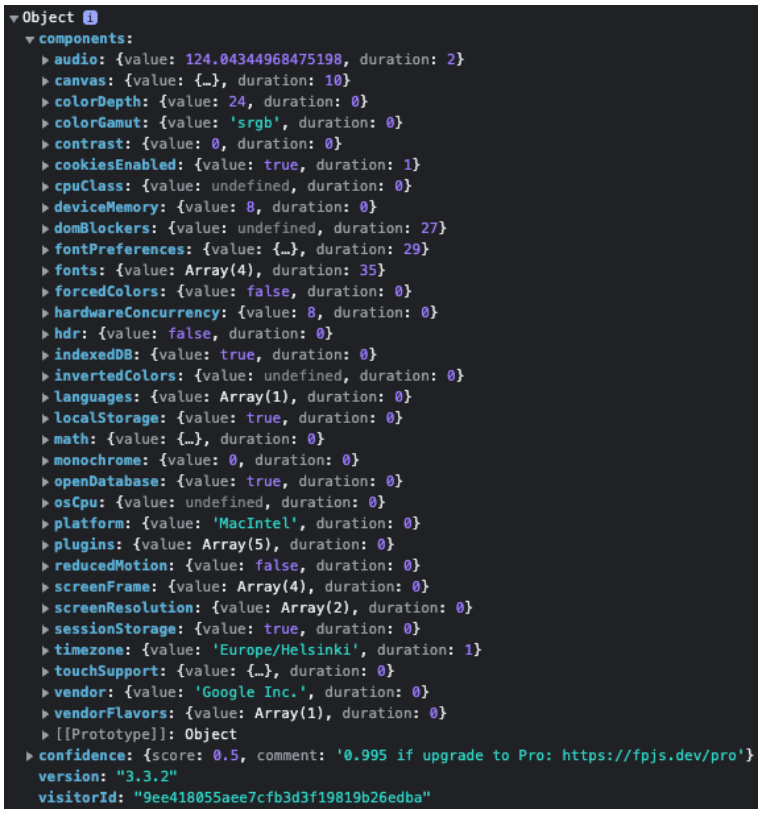
Esimerkkikoodi 6. FingerprintJS-kirjaston tuonti projektiin ja getFingerprint-funktio.

Sormenjälki voidaan nyt luoda ”getFingerprint”-funktiolla kutsumalla sitä verkkosivun koodissa. Tässä esimerkissä on hyödynnetty React-kirjaston sisäisiä funktioita “useState” ja “useEffect”, joilla sovelluksen tilaa voidaan hallita. Esimerkkikoodissa 7 sormenjälki ladataan ja tallennetaan tilamuuttujan ”fingerprint” arvoksi joka kerta, kun verkkosivu päivittyy.

```
const [fingerprint, setFingerprint] = useState({});

useEffect(() => {
     getFingerprint().then((res) => {
          setFingerprint(res);
     });
}, [])
```

Esimerkkikoodi 7. Sormenjäljen lataaminen verkkosivun koodissa.

Esimerkkikoodin 7 sisältämä fingerprint-tilamuuttuja sisältää nyt objektin, jossa on kaikki FingerprintJS-agentin lataamien komponenttien arvot sekä yksilöllinen tunnus käyttäjälle. Kuva 8 on kuvakaappaus tämän objektin sisällöstä.

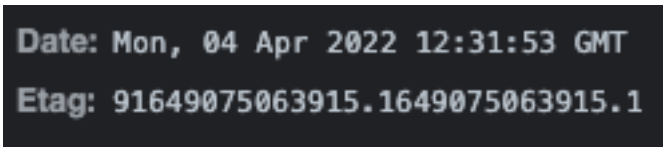


Kuva 8: FingerprintJS-agentin lataamat tiedot.

Sormenjäljen käytön huonoja puolia on, ettei mikään näistä tiedoista yksinään riitä yhden käyttäjän tunnistamiseen 100-prosenttisesti, sillä samanlaisia laitteita käyttäviä verkkovierailijoita tulee olemaan erittäin paljon. Kuitenkin mikä tahansa pieni muutos, kuten laajennuksen tai laiteohjaimen eri versio, riittää tekemään sormenjäljestä ainutlaatuisen muiden joukossa.

3.3 Etag-tunniste

Etag (entity tag) eli entiteettitunniste on palvelimelta haettavan resurssin tietyn version tunniste. Se on merkkijono, jonka palvelin generoi tiedostolle. Tyypillisesti Etag-arvo on resurssin versionumero tai viimeisen muokkauksen aikaleiman tiiviste. Kuva 9 on esimerkki Etag-tunnisteesta, joka on tallennettu verkkosivun palvelimen tiedostoon. Verkkosivustot, jotka käyttävät Etagia resursseissaan, mahdollistavat selaimen välimuistin tehokkaamman käytön, koska palvelimen ei joka kerta tarvitse lähettää haettua resurssia uudelleen, jos sen versio ei muutu. (Etag.)



Kuva 9: Palvelimelta saapuva Etag-tunniste.

Tekniikka hyödyntää menetelmää, jossa selain tekee palvelimelle ehdollisen pyynnön. Ehdolliset pyynnöt eroavat yleisistä pyynnöistä siten, että ne toimivat pyyntöjen otsikoissa määritetyn arvon mukaan. Toisin sanoen tämä arvo määrittää toiminnon, joka suoritetaan pyynnölle. (Etag.) Etagin sisältävien pyyntöjen yhteydessä käytetään monesti kuvan 10 mukaista If-None-Match-otsikkoa, joka tekee pyynnöstä ehdollisen. Pyynnön seurauksena palvelin palauttaa pyydetyn resurssin vain, jos palvelimella ei ole vastaavaa pyynnön otsikossa annettua Etag-arvoa. (If-None-Match.)

Kuva 10: If-None-Match-pyynnön otsikko.

Kun palvelimelta haetaan tiedostoa ensimmäistä kertaa, se tarjoaa kuvan Etagin kanssa, mikä on ikään kuin sisällön allekirjoitus, ainutlaatuinen tunniste kyseiselle tiedostolle ja sen versiolle. Myöhemmät pyynnöt sisältävät If-None-Match-otsikon, joka sisältää myös Etag-arvon, jolloin verkkopalvelin tarkistaa selaimen välimuistissa olevat ja palvelimen resurssista löytyvät Etag-arvot. Jos tunniste on sama kuin palvelimelta löytyvä, se tarkoittaa, että tiedosto ei ole muuttunut. Tällöin palvelin antaa vastauksen 304 Not Modified Status / Ei Muutettu, joka käskee selaimen lataamaan sisällön välimuistista sen sijaan, että se lähettäisi tiedoston uudelleen. Vastaavasti, jos Etag-tunnisteet eivät täsmää, se tarkoittaa, että tiedoston versio on muuttunut ja palvelin palauttaa koko vastauksen sekä uuden Etag-arvon kyseiselle resurssille. (HTTP Etag Explained 2011.)

Verkkovierailijan tunnistaminen Etagin avulla

Seuraavaksi tarkastellaan, miten Etageja voidaan hyödyntää käyttäjän tunnistamisessa. Tämä toteutus hyödyntää avoimen lähdekoodin Cookieless.js-implementaatiota, joka on kevyt toteutus vierailijoiden tunnistamiseen Etagien ja Node.js:n avulla.

Toteutus voidaan tehdä edellisen FingerprintJS-esimerkin kanssa samaan React-verkkosovellukseen. Aluksi projektiin luodaan tracker.js-niminen JavaScript-tiedosto, johon Cookieless.js-toteutus tehdään. Esimerkkikoodissa 8 luodaan ensin Tracker-niminen objekti, joka luo tunnukset uusille vierailijoille ja seuraa vierailijoiden istuntoja.

```
function Tracker(req, update) {
    if (typeof update === 'undefined') update = true;
    let reqUrl = url.parse(req.url, true);
    this.updated = false;
    this.callback = reqUrl.query.callback || 'cookielessCallback';
    this.etag = req.headers['if-none-match'] || this.generateEtag();
    this.parse();
    if (update) {
        this.update();
    }
}
```

Esimerkkikoodi 8. Tracker-niminen objekti, joka generoi vierailijoiden tunnukset.

Esimerkkikoodin 9 mukaan Etag generoidaan vain, jos pyynnön otsikosta saapuva If-None-Match-arvo ei täsmää olemassa olevan Etagin kanssa. Kun vierailija saapuu verkkosivulle ensimmäisen kerran, Etagia ei oletettavasti ole vielä olemassa, joten se luodaan ”generateEtag”-funktiolla yksilöllisestä tunnisteesta, joka luodaan esimerkkikoodissa 10 määritellyllä ”generateId”-funktiolla, sekä kellonajasta ja istuntojen määrästä. Tunnus, kellonaika ja istuntojen määrä yhdistetään merkkijonoksi ja erotellaan pisteellä.

```
Tracker.prototype.generateEtag = function() {
    let now = (new Date()).getTime();
    this.id = this.id || Tracker.generateId();
    this.session = this.session || 1;
    this.updated = true;
    return this.id + '.' + now + '.' + this.session;
}

Tracker.generateId = function() {
    let now = (new Date()).getTime();
    return Math.floor(Math.random()*100) + '' + now;
}
```

Esimerkkikoodi 9. generateEtag- ja generateId-funktiot.

Vierailijan yksilöllinen tunnus, kellonaika ja istuntojen määrä saadaan nyt luotua esimerkkikoodin 9 mukaisilla funktioilla. Koska kaikki edellä mainitut tiedot ovat nyt yhdistettynä yhdeksi Etag-merkkijonoksi, tarvitaan apufunktio, jolla kyseiset tiedot voidaan erotella toisistaan. Erotteluun käytetään yksinkertaista funktiota nimeltä ”parse”, joka erottelee pisteellä erotetut vierailijan tiedot Etagista ja tallentaa ne muuttujien arvoiksi (esimerkkikoodi 10).

```
Tracker.prototype.parse = function() {
    this.id = this.etag.match(/^([^.]+)/)[1];
    this.lastSeen = parseInt(this.etag.match(/^[^.]+\.([^.]+)/)[1]);
    this.session  = parseInt(this.etag.match(/\.([^.]+)$/)[1]);
    return this;
}
```

Esimerkkikoodi 10. Parse-funktio.

Funktio ”Respond” lähettää vastauksen, joka sisältää vierailijan tiedot ”callback”-funktiossaan. Kuten esimerkkikoodissa 11 ilmenee, se on yhdistelmä ”statusCode”-, ”buildHeader”- ja ”buildScript”-funktioita.

```
Tracker.prototype.respond = function(callback, res) {
    if (typeof res === 'undefined') {
        res = callback;
        callback = null;
    }
    callback = callback || this.callback;
    res.writeHead(this.statusCode(), this.buildHeader());
    if (this.updated) {
        res.end(this.buildScript(callback));
    } else {
        res.end();
    }
    return res;
}
```

Esimerkkikoodi 11. Respond-funktio.

Seuraavaksi määritellään myös ”statusCode”-, ”buildHeader”- sekä ”buildScript”-funktiot, jotta ymmärretään paremmin, mitä respond-funktio tekee. StatusCode palauttaa http-tilakoodin 200 tai 304 riippuen siitä, onko Etag muuttunut. BuildHeader luo http-vastauksen otsikon, joka sisältää Etagin. BuildScript puolestaan asettaa vastaukseen vierailijan tiedot JSONP-skriptinä, jotka luetaan ”callback”-funktiolla.

```
Tracker.prototype.statusCode = function() {
    return this.updated ? 200 : 304;
}

Tracker.prototype.buildHeader = function() {
    return {
        'Content-Type': 'text/javascript',
        'Etag': this.etag
    }
}

Tracker.prototype.buildScript = function(callback) {
    callback = callback || this.callback;
    return "; typeof " + callback + " === 'function' && " + callback
+"({"+
    "id: "+this.id+","+
    "session: "+this.session+","+
    "lastSeen: "+this.lastSeen+"}"+
    ");";
}
```

Esimerkkikoodi 12. statusCode-, buildHeader- ja buildScript-funktiot.

Jotta Tracker-objekti voidaan luoda ja sen ominaisuuksia hyödyntää, tarvitaan sovellukseen palvelin, joka mahdollistaa vastaamisen sovelluksesta lähteviin http-pyyntöihin. Projektiin luodaan server.js-niminen tiedosto ja asennetaan itse Express-palvelin. Seuraava komento asentaa ja tallentaa Expressin projektiin NPM:n avulla (esimerkkikoodi 13).

```
npm install express –save
```

Esimerkkikoodi 13. Express-palvelimen asennus NPM:n avulla.

Express-palvelimen asennuksen jälkeen voidaan muokata server.js-tiedostoa. Esimerkkikoodissa 14 server.js-tiedosto vaatii Expressin ja luo sen jälkeen palvelimen. Määritellään myös portin numero, jossa Express-palvelin toimii.

```
const express = require('express');
const app = express();
const Tracker = require('./tracker');
const port = process.env.PORT || 5000;

app.listen(port, () => console.log(`Listening on port ${port}`));
```

Esimerkkikoodi 14. Express-palvelimen luominen server.js-tiedostossa.

Server.js-tiedostoon lisätään myös GET-reitti, joka haetaan React-sovelluksesta vierailijan tunnistamiseksi. Kun kyseistä web-osoitetta (/i.js) haetaan, palvelin luo Tracker-objektin ja antaa vastauksen sille määritetyllä ”Respond”-funktiolla (esimerkkikoodi 15).

```
app.get('/i.js', (req, res) => {
  let tracker = new Tracker(req, true);
  tracker.respond(res);
});
```

Esimerkkikoodi 15. GET-reitti ja Tracker-luokan käyttö.

Palvelin on nyt valmiina vastaanottamaan kutsun sovelluksesta ja lähettämään vastauksessa vierailijan tiedot. React-sovelluksessa muokataan sitä lähettämään kutsu palvelimelle. Sovellukseen luodaan uusi tiedosto useTrackerScipt.js, joka lähettää http-pyynnön ja hakee vierailijan tiedot palvelimelta. Esimerkkikoodissa 16 on useTrackerScript.js-tiedoston sisältö.

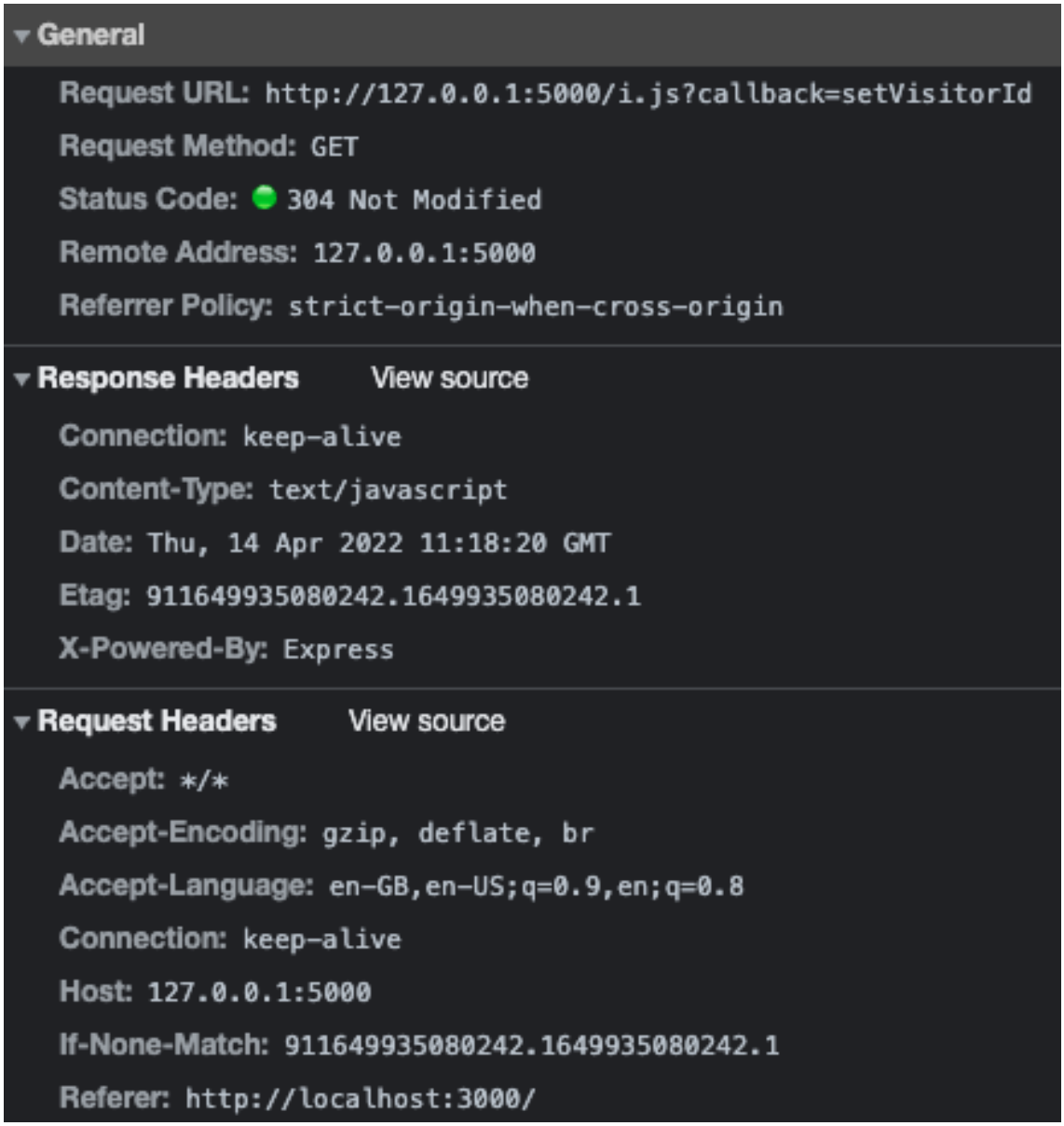
```
const useTrackerScript = (callback) => {
  useEffect(() => {
    const script = document.createElement('script');
    script.src = "http://127.0.0.1:5000/i.js?callback=setVisitorId";
    script.async = true;

    window.setVisitorId = (visitor) => {
      console.log(visitor);
      callback(visitor);
      window.setVisitorId = undefined;
    }

    document.body.appendChild(script);

    return () => {
      document.body.removeChild(script);
    }
  }, [callback]);
};
```

Esimerkkikoodi 16. useTrackerScript-tiedosto.

Käytetään ”useEffect”-funktiota, jossa haetaan palvelimelta resurssi, joka kerta, kun verkkosivu päivittyy. Funktion sisällä sovellukseen luodaan ensin uusi script-elementti ja sen lähteeksi asetetaan Express-palvelimessa määritelty GET-reitti. Kun reitti täsmää server.js-tiedostossa määriteltyyn polkuun, palvelin tunnistaa pyynnön ja vastaa siihen Tracker-luokan ”Respond”-funktiolla.

Esimerkkikoodissa 16 määritellään myös vierailijan tiedot palauttava, Tracker-luokasta tuttu ”callback”-funktio, jonka nimeksi annetaan setVisitorId.

Kuva 11 on kuvakaappaus palvelimelle lähtevistä ja sieltä saapuvista otsikoista, jotka sisältävät Etagin. Kuvassa on palvelimelle saapuvan pyynnön mukana tullut Etag ja palvelimen sisältämä Etag. Koska näiden Etagien arvot ovat samat, palvelin palauttaa koodin 304 Not Modified, ja vierailija on tunnistettu samaksi kuin edellisellä kerralla.



Kuva 11: Palvelimen pyynnön ja vastauksen otsikot.

Esimerkkikoodissa 16 määritelty useTrackerScript voidaan nyt tuoda React-sovellukseen. Kun skrpti ajetaan, se palauttaa vierailijan tiedot, josta tallennetaan tunnus (id) muuttujan ”visitorId”-arvoksi (esimerkkikoodi 17).

```
const [visitorId, setVisitorId] = useState(undefined;

useTrackerScript((visitor) => {
  setVisitorId(visitor.id)
});
```

Esimerkkikoodi 17. useTrackerScriptin käyttö.

4 Tulosten arviointi

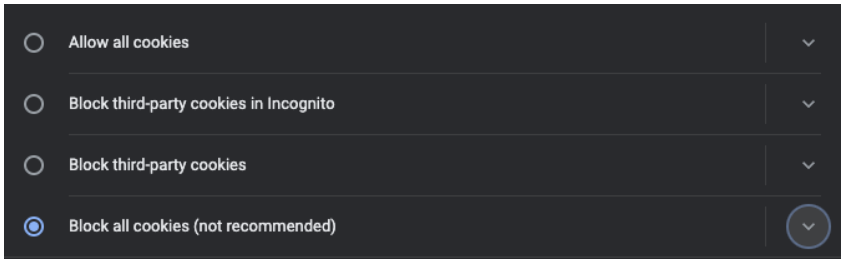
Tässä luvussa arvioidaan insinöörityössä esiteltyjä menetelmiä ja tarkastellaan niiden luotettavuutta verkkovierailijan tunnistamisessa. Kuvassa 12 on verkkosovelluksen ulkoasu, joka sisältää sekä FingerprintJS-sormenjäljen että Etag-tunnisteen.

Kuva 12: FingerprintJS- ja Etag-toteutuksien tunnukset.

4.1 FingerprintJS-sormenjälki

Sormenjäljen avulla luotu tunnus verkkovierailijalle hyödyntää siis käyttäjän laitteen ja ohjelmiston eri tietoja, joten sen luotettavuutta voidaan arvioida vaikuttamalla näihin tietoihin ja selvittää, muuttuuko vierailijatunnus, kun verkkosivu ladataan uudelleen.

Yksi sormenjäljen muodostamiseen käytetyistä komponenteista tarkistaa, onko evästeiden käyttö sallittu selaimessa. Voidaan tarkistaa, muuttuuko tämän ”cookiesEnabled”-komponentin arvo, kun selaimen asetuksista kielletään evästeiden käyttö kuvan 13 mukaan.

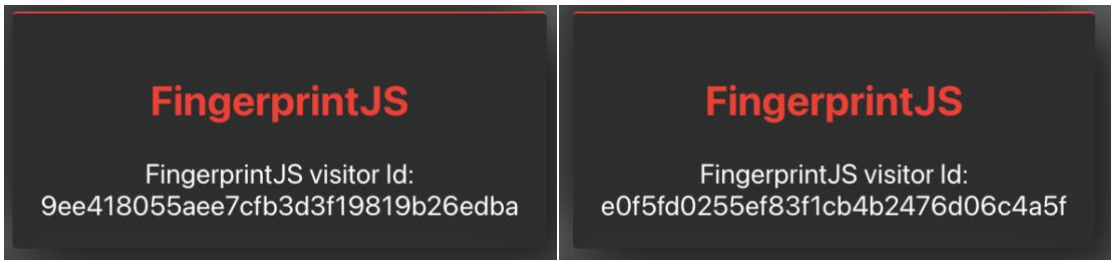


Kuva 13: Evästeistä kieltäytyminen Chrome-selaimen asetuksissa.

Muutoksen jälkeen, kun sormenjälki ladataan uudelleen, voidaan kuvasta 14 todeta, että ”cookiesEnabled”-komponentin arvo on muuttunut.

Kuva 14: cookiesEnabled-komponentin arvo ennen evästeasetuksien muuttamista ja sen jälkeen.

Evästeiden käytöstä kieltäydyttyä myös itse sormenjälki on muuttunut. Kuvassa 15 on sormenjälkeen saatu muutos selaimen evästeasetuksia muuttamalla.



Kuva 15: Sormenjäljen muutos evästeasetuksia muuttamalla.

Toinen muutos sormenjäljen sisältämiin komponentteihin voidaan myös testata helposti, jos käytössä on ulkoinen näyttö, jonka resoluutio on erilainen. Kun verkkosivu ja sormenjälki ladataan ulkoisella näytöllä, jossa on eri resoluutio, ”screenResolution”-komponentin arvoa saadaan muutettua, mikä saa myös sormenjäljen muuttumaan.

Sormenjäljen luotettavuus perustuu täysin FingerprintJS-agentin lataamien komponenttien arvoihin, joten pienikin muutos niihin saa sormenjäljen muuttumaan. Toisin sanoen, mitä tahansa muutoksia tekemällä sormenjälki olettaa käyttäjän olevan eri kuin aikaisemmalla kerralla. Sormenjäljen luoma vierailijatunnus ja sen tarkkuus ovat siis täysin riippuvaisia agentin lataamista tiedoista.

Sormenjälki ei siis ole varsin luotettava tapa yksittäisen verkkovierailijan tunnistamiseksi, koska järjestelmään tai selaimeen saattaa tulla muutoksia ilman, että käyttäjä tekee niitä tietoisesti vaikuttaakseen sormenjäljen luomaan vierailijatunnukseen. Esimerkiksi uuden näytön hankinta tai järjestelmäpäivityksen mukana tullut laajennus muuttavat vierailijatunnuksen toisenlaiseksi. On myös huomioitava, että lukuisten käyttäjien joukosta on mahdollista löytyä useampi identtinen sormenjälki, jos kaikki komponenttien arvot sattuvat olemaan samat.

4.2 Etag-tunniste

Etagin-tunnisteen avulla käyttäjä tunnistetaan selaimen välimuistiin tallennetun tiedoston avulla. Insinöörityössä esitelty menetelmä hakee palvelimelta tietyn verkko-osoitteen takana olevan resurssin aina, kun vierailija saapuu verkkosivulle ja asettaa sille tunnisteen. Vierailija tunnistetaan kyseisellä resurssille asetetulla tunnisteella, jos tiedosto löytyy selaimen välimuistista.

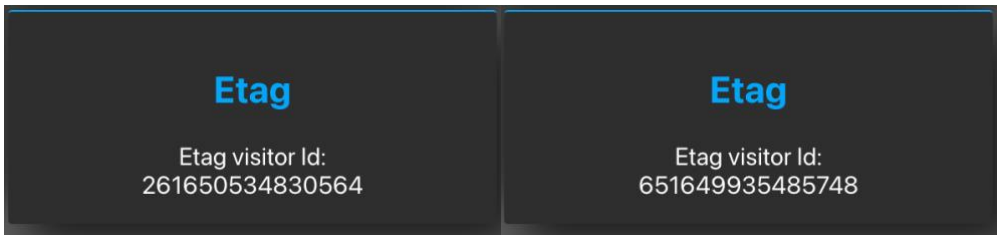
Yksi haittapuoli Etagin käytössä vierailijan tunnistamiseen on itse selaimen välimuistissa, koska sen tyhjentäminen poistaa myös sieltä löytyvän Etagilla varustetun tiedoston. Tämä tarkoittaa sitä, että myös vierailijatunnus katoaa, jos käyttäjä tyhjentää selaimen välimuistin. Sama pätee kuitenkin myös evästeiden käyttöön, koska jos käyttäjä poistaa evästeitä selaimen asetuksista, katoaa myös evästeiden avulla toteutettu tunnus.

Suurin ongelma vierailijan tunnistamiseksi Etagin avulla löytyi insinöörityötä tehdessä, kun tunniste luotiin kahdesta eri verkkotunnuksesta. Insinöörityössä toteutettu sovellus on mahdollista avata selaimessa eri verkkotunnuksista muokkaamalla päätelaitteen ”etc/hosts”-tiedostoa. Kun tiedostoon lisätään esimerkkikoodin 18 mukaiset rivit, saadaan työssä toteutettu verkkosivu toimimaan siinä asetettujen verkkotunnusten kautta.

```
127.0.0.1 page1.com
127.0.0.1 page2.com
```

Esimerkkikoodi 18. Etc/hosts-tiedostoon tehty muutos.

Nyt toteutus voidaan avata eri verkkotunnusten kautta. Kun selaimella navigoidaan osoitteisiin ”page1.com” ja ”page2.com”, Etag-tunnukset ovat erilaiset. Kuva 16 on kuvakaappaus verkkosivun Etag-tunnisteista kahdella eri verkkotunnuksella.



Kuva 16: Etag-tunnisteet eri verkkotunnuksissa.

Tämä ongelma ei ole ollut aina olemassa. Vanhemmilla selaimilla käyttäjän vieraillessa sivulla, joka pyytää palvelimelta resurssia, välimuistiavaimeksi tallentuu resurssin URL-osoite. Jos käyttäjä vierailee useammalla verkkosivustolla ja pyytää samaa resurssia, välimuistiavain on sama kaikilla sivuilla. Insinöörityön esimerkissä Etagit ovat erilaiset kahden verkkotunnuksen välillä, koska uudemmissa selainversioissa on otettu käyttöön niin sanottu välimuistin osiointi. Osiointi toimii vanhasta välimuistiavaimesta poiketen siten, että välimuistiavaimeen tallennetaan pyydetyn resurssin URL-osoitteen lisäksi myös kyseisellä hetkellä vieraillun verkkosivun osoite. Tässä tapauksessa selaimen välimuistissa olevaa Etagilla varustettua resurssia ei voida hakea välimuistista, koska kahden verkkotunnuksen väliset välimuistiavaimet ovat erilaiset. (Kitamura 2020).

5 Yhteenveto

Insinöörityössä pyrittiin selvittämään verkkovierailijan tunnistamiselle vaihtoehtoisia menetelmiä ilman evästeiden käyttöä. Työssä oli tarkoitus perehtyä evästeiden käytön perusteisiin ja käyttötarkoituksiin. Lisäksi tavoitteena oli tutkia vaihtoehtoisia tunnistamismenetelmiä ja luoda niistä käytännön esimerkit.

Työn aikana tehty tutkimustyö johti verkkosovelluksen toteuttamiseen, jossa kahden erilaisen menetelmän tehokkuutta voidaan testata. Vaihtoehtoiset menetelmät valikoituivat toteutuksien yksinkertaisuuden ja potentiaalisen tehokkuuden perusteella. Menetelmien toteutuksista saatiin luotua tarkat ohjeet

niiden implementoimiseksi, ja niiden täsmällisyyttä voidaan arvioida verkkoselaimen avulla.

Tulokset osoittivat, että käyttäjän tunnistaminen työhön valikoituneilla menetelmillä ei ole kovin täsmällistä. Koska evästeiden käytön mahdollistama käyttäjän tarkka tunnistaminen on monelle teknologiayritykselle tärkeä ominaisuus, sen korvaaminen mahdollisesti vaatii uusien menetelmien ja innovaatioiden kehittämistä.

Lähteet

All you need to know about Third-Party-Cookies. 2021. Verkkoaineisto. Cookie Script. <https://cookie-script.com/all-you-need-to-know-about-third-party-cookies.html#h5-1-what-happens-after-third-party-cookies-are-eliminated>. Päivitetty 27.9.2021. Luettu 2.10.2021.

Bhatia, Manmeet Singh. 2019. Etag Tracking 101. Verkkoaineisto. Secjuice. <https://www.secjuice.com/etag-entity-tag-tracking/>. Päivitetty 20.10.2019. Luettu 14.4.2022.

Brain, Marshall. 2007. How Internet Cookies Work. Verkkoaineisto. Mindpride Computer Services. <http://www.mindpride.net/root/Extras/how-stuff-works/how_internet_cookies_work.htm>. Päivitetty 21.11.2007. Luettu 12.4.2022.

Copland, Savannah. 2020. The Beginner's Guide to Browser Fingerprinting for Fraud Detection. Verkkoaineisto. FingerprintJS. <https://fingerprintjs.com/blog/what-is-browser-fingerprinting/>. Päivitetty 24.12.2020. Luettu 30.3.2022.

Etag. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag>. Luettu 20.3.2022.

FingerprintJS Pro Introduction and quick start examples. Verkkoaineisto. FingerprintJS. <https://dev.fingerprintjs.com/docs/introduction>. Luettu 30.3.2022.

Gigantti verkkokauppa. Verkkoaineisto. Gigantti Oy. <https://www.gigantti.fi/>. Luettu 5.3.2022.

HTTP Etag Explained. 2011. Verkkoaineisto. Avinash. Xpertdeveloper. <https://xpertdeveloper.com/http-etag-explained/>. Päivitetty 18.3.2011. Luettu 19.3.2022.

Huttunen, Kaisa. 2021. Mitä ovat evästeet? Verkkoaineisto. Zoner. <https://www.zoner.fi/nettisivujen-teko/evasteet/> Päivitetty 14.4.2021. Luettu 10.12.2021.

If-None-Match. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match>. Luettu 20.3.2022.

Kitamura, Eiji. 2020. Gaining security and privacy by partitioning the cache. Verkkoaineisto. Chrome Developers. <https://developer.chrome.com/blog/http-cache-partitioning/#is-this-standardized-do-other-browsers-behave-differently>. Päivitetty 24.10.2020. Luettu 23.4.2022.

Kupietzky, Jeff. 2021. What Is the Future of Advertising in A Post-Cookie World? Verkkoaineisto. Forbes. <https://www.forbes.com/sites/forbesbusinesscouncil/2021/06/15/what-is-the-future-of-advertising-in-a-post-cookie-world/?sh=70d133f05f7c>. Päivitetty 15.6.2021. Luettu 7.9.2021.

Mburu, Bernard. 2021. Understanding and Working with JavaScript Cookies. Verkkoaineisto. Section. < https://www.section.io/engineering-education/understanding-and-working-with-javascript-cookies/#creating-a-cookie>. Päivitetty 6.8.2021. Luettu 2.1.2022.

Morris, Will. 2020. What Are Cookies and How Do They Work? Verkkoaineisto. Elegant Themes. <https://www.elegantthemes.com/blog/wordpress/what-are-cookies-and-how-do-they-work>. Päivitetty 24.10.2020. Luettu 2.10.2021.

Ramirez, Noah. 2021. How cookies work, and how to conduct a cookie audit. Verkkoaineisto. Osano. <https://www.osano.com/articles/how-cookies-work>. Päivitetty 15.1.2021. Luettu 2.10.2021.

Setting Cookies in HTML: All you Need to Know About. 2019. Verkkoaineisto. Edureka. <https://www.edureka.co/blog/cookies-in-html/>. Päivitetty 21.10.2019. Luettu 2.10.2021.

Solla, Katja. 2021 Digitreenit: Mitä nettisivujen evästeet oikein tekevät? Onko ne pakko hyväksyä? Verkkoaineisto. Yle. <https://yle.fi/aihe/artikkeli/2020/02/22/digitreenit-mita-nettisivujen-evasteet-oikein-tekevat-onko-ne-pakko-hyvaksya> Päivitetty 3.12.2021. Luettu 10.1.2022.

Trackin Pixel. Verkkoaineisto. Ryte Wiki. <https://en.ryte.com/wiki/Tracking_Pixel>. Luettu 3.10.2021.

Understanding and working with Third Party Cookies. Verkkoaineisto. OpenGenius IQ. < https://iq.opengenus.org/third-party-cookies/>. Luettu 22.11.2021.

What are Cookies? – How Cookies Work and What They Do. 2020. Verkkoaineisto. Drsoft.<https://drsoft.com/2019/05/13/what-are-cookies-how-cookies-work-and-what-they-do/>. Päivitetty 13.11.2020. Luettu 27.3.2022.

5 Ways To Identify Your Users Without Using Cookies. 2015. Verkkoaineisto. Kompyte. <https://www.kompyte.com/blog/5-ways-to-identify-your-users-without-using-cookies/>. Päivitetty 29.10.2015. Luettu 19.3.2022.