Matti Leinonen

Kirjautuminen ja single sign-on -integroinnit tietojärjestelmään

Tietotekniikan pro gradu -tutkielma

17. heinäkuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Matti Leinonen

Yhteystiedot: matti.o.leinonen@student.jyu.fi

Ohjaaja: Vesa Lappalainen ja Ari Viinikainen

Työn nimi: Kirjautuminen ja single sign-on -integroinnit tietojärjestelmään

Title in English: Authentication and single sign-on integrations

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 120+0

Tiivistelmä: Tässä pro gradu -tutkielmassa perehdytään tietojärjestelmiin kirjautumiseen. Tutkielmassa esitellään kirjautumiseen liittyvää problematiikkaa ja keinoja vastata näihin ongelmiin. Lisäksi perehdytään erilaisiin kirjautumisen tapoihin ja erityisesti keskitytään single sign-oniin. Tutkielman empiirisessä osassa toteutetaan single sign-on -integrointeja tietojärjestelmään ja johdetaan toteutuksista prosessi single sign-on -integrointien suunnittelun ja toteuttamisen sujuvoittamiseksi.

Avainsanat: Single sign-on, kirjautuminen, autentikaatio

Abstract: This master's thesis familiarizes the reader with the processes of authentication into information systems such as websites. The thesis presents problems of authentication and some solutions to them. Different methods of authentication are expounded upon. Single sign on gets special attention and some single sign on integrations to an information system are presented. From these integrations a process for planning and developing single sign on integrations is derived.

Keywords: Single sign-on, authentication

Kuviot

Taulukot

Sisältö

1 Johdanto

Käyttäjien autentikoiminen on erittäin tärkeä osa verkkosivujen turvallisuutta (Woods 2016, s. 86). Sen avulla voidaan antaa oikeille käyttäjille pääsy oikeisiin resursseihin (Wiedenbeck ym. 2005, s. 103). Salasanat ovat kaikkein yleisin tapa hoitaa autentikaatio (Woods 2016, s. 86), mutta perinteiseen salasanoilla toteutettavaan autentikaatioon liittyy kuitenkin laajalti tunnettuja ongelmia.

Ongelmia syntyy salasanojen määrän kasvaessa kun käyttäjien muistamisvaikeudet lisääntyvät (O'Gorman 2003). Nykyään ihmisillä on enemmän salasanoja kuin mitä he kykenevät muistamaan tai ainakin uskovat kykenevänsä muistamaan (Woods 2016, s. 89-90). Muistivaikeuksien vuoksi käyttäjät omaksuvat turvallisuutta heikentäviä käytänteitä salasanojen hallintaan. Näitä ovat esimerkiksi salasanojen kirjoittaminen muistilapulle, salasanojen uudelleenkäyttäminen eri palveluissa ja helposti arvattavien salasanojen käyttäminen (Woods 2016, s. 86; Wiedenbeck ym. 2005, s. 105; O'Gorman 2003, s. 2037) .

Salasanoille on olemassa erilaisia vaihtoehtoja, kuten fyysisiin laitteisiin tai biometrisiin tunnisteisiin perustuvia autentikaation tapoja. Salasanoille kuitenkin näyttää olevan vaikea löytää korvaajaa. Vaikka vaihtoehtoiset autentikaatiomenetelmät ovat usein joiltain osin salasanoja parempia, niihin liittyy myös ongelmia, joita salasanoilla ei ole (Bonneau ym. 2012, s. 562). Näitä ovat esimerkiksi jonkin fyysisen tunnisteen mukana kuljettamisen tarve, autentikaation hitaus, hitaampi tunnisteiden häviämisestä palautuminen ja korkeammat käyttäjäkohtaiset kustannukset (Bonneau ym. 2012, s. 562-564). Tarvittavien salasanojen määrä näyttääkin olevan vain kasvussa (Woods 2016, s. 86).

Single sign-on (SSO) on autentikaation tapa, jossa käyttäjä pääsee yksillä tunnuksilla kirjautumaan useisiin eri palveluihin (Chan 2006; Ardagna ym. 2006, s. 209-210) . Näin vähennetään sitä aikaa ja vaivaa, joka koituu uudelleen ja uudelleen kirjautumisesta eri verkkopalveluihin sekä useiden eri tunnusten ja salasanojen muistamisesta. Lisäksi SSO:lla vähennetään palveluntarjoajien ylläpidon määrää, kun käyttäjien kirjautumisen hoitaa kolmas osapuoli. (Chan 2006; Ardagna ym. 2006, s. 209-210.) SSO:n tavoitteena ei ole pelkästään yhdet tunnukset useaan palveluun, vaan että kirjautumisprosessi tarvitsisi käydä läpi vain kerran, jon-

ka jälkeen käyttäjä pääsee käsiksi useisiin järjestelmiin (You ja Zhu 2012, s. 383; Chalandar, Darvish ja Rahmani 2007, s. 163; Chan 2006) .

Tässä pro gradu -tutkielmassa on tarkoituksena perehtyä siihen, millaisia eri tapoja kirjautumisen toteuttamiseen on, millaisia ongelmia kirjautumiseen liittyy ja miten ongelmiin voidaan vastata. Kuten todettua, single sign-onilla voidaan vastata joihinkin kirjautumiseen liittyviin haasteisiin. Tutkielman empiirisessä osassa toteutetaan single sign-on ratkaisuja TIM-tietojärjestelmään. Tutkimuskysymyksenä on, miten single sign-on -palveluja saadaan integroitua tietojärjestelmään. Tuloksena saadaan prosessimalli, jota noudattamalla tällaisten integrointien tekemistä voidaan sujuvoittaa.

Toisessa luvussa perehdytään autentikaation problematiikkaan sekä erilaisiin tapoihin toteuttaa käyttäjien autentikoiminen. Kolmannessa luvussa tutustutaan single sign-oniin tarkemmin. Neljännessä luvussa esitellään, miten kirjautuminen on toteutettu TIM-järjestelmässä. Viidennessä luvussa esitellään single sign-on -toteutukset. Kuudennessa luvussa esitetään tutkimuksen tulokset ja lopuksi seitsemäs luku on yhteenveto työn sisällöstä.

2 Kirjautuminen

Autentikaatiomenetelmät jaetaan usein kolmeen eri luokkaan sen mukaan, mihin ne perustuvat. Nämä luokat ovat

1. Jotain mitä käyttäjä tietää (kuten salasana),
2. jotain mitä käyttäjällä on hallussaan (kuten jokin fyysinen laite) ja
3. se, kuka käyttäjä on (kuten jokin biometrinen tunnise). (Brumen, Ivani ja Rozman 2016, s. 1923; Aloul, Zahidi ja El-Hajj 2009, s. 641; Dong, Clark ja Jacob 2008, s. 51; O'Gorman 2003, s. 2023-2024.)

Muitakin jakoja tietysti on olemassa. Dong ym. täydentävät jakoa kategorialla "jokin, johon käyttäjällä on pääsy". Tällä he tarkoittavat jotakin, joka ei ole suoraan käyttäjän hallussa, mutta johon hän pääsee esimerkiksi kirjautumaan, kuten vaikkapa sähköpostitili. He erottavat tämän perinteisestä "mitä käyttäjällä on hallussaan" -luokasta siksi, että luokkaan kuuluva autentikaatiomenetelmä luo riippuvuussuhteen autentikaation turvallisuuteen. Riippuvuussuhde tarkoittaa sitä, että jonkin järjestelmän turvallisuus riippuu jonkin toisen järjestelmän turvallisuudesta. Esimerkiksi jos järjestelmän autentikaatiotunnusten palauttaminen tapahtuu jonkin sähköpostin kautta, riippuu järjestelmän kirjautumisen turvallisuus tuon sähköpostitilin turvallisuudesta. (Dong, Clark ja Jacob 2008, s. 51-52.)

O'Gorman pysyy kolmiluokituksessa, mutta muuttaa sen jokseenkin erilaiseen muotoon (O'Gorman 2003, s. 2024-2025):

1. Tietoperusteiset autentikaattorit.
2. Esineperusteiset autentikaattorit.
3. ID-perusteiset autentikaattorit.

Tietoperusteiset autentikaattorit perustuvat niiden salaisuuteen tai obskuuriuteen. Salaisuudeksi lasketaan esimerkiksi perinteinen salasana, kun taas obskuuri autentikaattori on "salainen suurimmalta osalta ihmisiä", kuten esimerkiksi äidin tyttönimi. Esineperusteisen autentikaation perusta on autentikaattorin hallussapito. Esineeseen voi yhdistyä toinen autentikaatiomenetelmä kuten salasana sen käyttämisen mahdollistamiseksi. ID-perusteinen auten-

tikaatio nojaa käyttäjän uniikkiuteen. Esimerkkejä tästä ovat henkilötodistukset ja biometriset tunnisteet, joiden oleellisin ominaisuus ei ole salaisuus vaan vaikea väärennettävyys. Eri tyyppisiä autentikaatiomenetelmiä voidaan yhdistää, jolloin puhutaan monitekijäisestä autentikaatiosta (multifactor authentication). Monitekijäisellä autentikaatiolla voidaan parantaa autentikaation turvallisuutta. (O'Gorman 2003, s. 2024-2025.)

Dong ym. luokittelevat tunnisteita myös muilla perusteilla kuin autentikaattorin itsensä tyypillä. Tunnisteet voivat olla **pääasialliset** tai **hätätunnisteet**, joiden avulla voidaan palauttaa pääsy järjestelmään pääasiallisten tunnusten unohtuessa. Jos tunnisteet ovat **menetettävissä** (losable), pitää järjestelmässä olla mahdollisuus saada ne jotenkin palautettua tai asetettua uudelleen. Myös **taho, joka tunnukset asettaa** on luokittelun peruste. Järjestelmän itsensä antaessa tunnisteet voidaan niiden turvallisuus taata joiltain osin, mutta käyttäjän itsensä asettamat tunnisteet saattavat olla helpommin käytettäviä. Kolmannen osapuolen asettaessa tunnisteet joudutaan turvallisuudessa nojaamaan tämän kolmannen osapuolen käytänteisiin. (Dong, Clark ja Jacob 2008, s. 51-52.)

2.1 Kirjautumisen problematiikkaa

2.1.1 Turvallisuus ja käytettävyys

Voidaan miettiä, mitä autentikaatiojärjestelmän turvallisuus ylipäätään tarkoittaa. Esimerkiksi voidaan esittää listoja erilaisista hyökkäystyypeistä ja vertailla mitkä järjestelmät pystyvät niitä torjumaan. O'Gorman esittää suhteellisemman tavan määritellä järjestelmän turvallisuus: "vahvassa" järjestelmässä hyökkäyksestä koituvat kustannukset ovat mahdollisia hyötyjä korkeammat (O'Gorman 2003, s. 2023). Määritelmä on varmasti hyvin käytännöllinen, sillä autentikoitumisesta voitaisiin epäilemättä tehdä loputtomasti yhä turvallisempaa esimerkiksi lisäämällä siihen aina uusia vaiheita ja autentikaattoreita. Rajalliset resurssit kuitenkin estävät järjestelmän loputtoman parantamisen, minkä vuoksi on määriteltävä tapauskohtaisesti riittävä turvallisuuden taso.

Autentikaation turvallisuuden ja käytettävyyden on katsottu olevan ristiriidassa toistensa kanssa (Bonneau ym. 2012, s. 554; Lang 2011, s. 398) . Käyttäjät pyrkivät kiertämään autentikaatioon turvallisuutta lisääviä menetelmiä sitä enemmän mitä enemmän niitä autenti-

kaatioon lisätään. Käyttäjät kiertävät menetelmiä pitäessään niitä rasittavina ja yksityisyyttä loukkaavina. (Komanduri ym. 2011; Lang 2011, s. 398.)

Herley ja van Oorschot ovat sitä mieltä, että käyttäjiä neuvotaan kirjautumisen turvallisuuden suhteen väärällä painotuksella. Vahvan salasanan luomista korostetaan, vaikka väsytyshyökkäykset (brute force -attack) ovat heidän mielestään pienempi uhka kuin tietojenkalastelu- (phishing) tai näppäilyntallennushyökkäykset (keylogging attack). Kaikista vähiten neuvoja heistä annetaan näppäilyntallennushyökkäysten ehkäisemiseksi, vaikka heidän mukaansa niillä todennäköisesti saadaan varastettua enemmän salasanoja kuin kahdella muulla mainitulla. (Herley ja Van Oorschot 2012, s. 33.) On myös esitetty, että salasanojen kirjoittaminen ylös ei välttämättä ole huono tapa, kunhan salasanat pidetään turvallisessa paikassa (Brumen ja Černezel 2014, s.1367; Herley ja Van Oorschot 2012, s. 34) .

2.1.2 Hyökkäykset autentikaatiojärjestelmiä vastaan

Dong ym. esittävät passiivisen ja aktiivisen hyökkäystyypin (engl. passive attacks, active attacks) autentikaatiojärjestelmiä vastaan. Passiivisissa hyökkäyksissä ei vaadita käyttäjän omaa aktiivisuutta, vaan hänen tunnisteensa hankitaan esimerkiksi niiden korkean julkisuuden avulla. Esimerkkinä toimii vaikkapa äidin tyttönimen käyttäminen salasanana. Aktiivisissa hyökkäyksissä vaaditaan käyttäjän osallistumista. Niissä hyökkääjä kysyy tunnisteita teeskennellen olevansa jokin luotettava taho. (Dong, Clark ja Jacob 2008, s. 50.)

Eri tunnistetyypeillä on erilainen alttius joutua passiivisten hyökkäysten kohteiksi. Tämä johtuu muun muassa niiden erilaisista julkisuuden tasoista. Matala julkisuuden taso on tunnisteilla, jotka ovat vain käyttäjän ja järjestelmän tiedossa. Julkisesti saatavilla olevien tunnisteiden julkisuusaste on korkea. Muissa tapauksissa julkisuus on keskitasoinen. Jos tunnisteen julkisuusaste on keskitasoinen tai korkea, se on haavoittuvainen passiivisille hyökkäyksille. Käyttäjän ominaisuuksia kuvaavilla tunnuksilla on Dongin ym. mukaan aina keskitasoinen tai jopa korkea julkisuus. Salasanoilla voi olla eri tekijöistä riippuen matala tai korkea julkisuus. (Dong, Clark ja Jacob 2008, s. 51-52.)

2.1.3 Tunnisteiden kalastelu

Tunnisteiden kalastelulla (engl. phishing) tarkoitetaan sitä, että hyökkääjä tekeytyy oikeaksi palveluntarjoajaksi, jolle tunnisteet voi antaa (Grassi ym. 2017, s. 43). Dongin ym. määrittelemät aktiiviset hyökkäykset vastaavat tätä kalasteluhyökkäyksen määritelmää, joskin he itse näyttävät pitävän kalastelua vain yhtenä aktiivisten hyökkäysten muotona. He kuitenkin käsittelevät itsekin aktiivisista hyökkäyksistä kalasteluhyökkäyksiä. (Dong, Clark ja Jacob 2008, s. 50 ja 54.) Kun arvioidaan aktiivisen hyökkäyksen riskejä tulee arvioida mitä osapuolia hyökkääjä voi esittää olevansa. Hyökkääjä voi teeskennellä olevansa se taho, jonka kanssa tunnisteet on jaettu, jokin taho, jolla on oikeus tunnisteita kysellä tai jokin taho, jonka tunnisteiden avulla päästään käsiksi haluttuihin tunnisteisiin. Nämä eroavat erilaisissa autentikaatioratkaisuissa. Normaalisti tunnisteet on jaettu vain sen tahon kanssa, jonka palveluun käyttäjätili on luotu. Kuitenkin esimerkiksi OpenID:n kaltaisten single sign-on järjestelmien kanssa toimittaessa käyttäjätili luodaan sekä OpenID-tarjoajalle että haluttuun palveluun ja tunnisteet jaetaan vain OpenID-tarjoajan kanssa. Single sign-on palveluissa vastaan tulee myös se tilanne, että on olemassa lukuisia eri luotettavia palveluja, jotka tunnuksia voivat pyytää. (Dong, Clark ja Jacob 2008, s. 54-57)

Alttius aktiivisille hyökkäyksille riippuu myös tunnisteiden sensitiivisyydestä. Tunnisteiden sensitiivisyys vaikuttaa siihen, miten tarkkaavainen tai epäileväinen käyttäjä on, kun niitä häneltä pyydetään. Mitä sensitiivisemmät tunnukset, sitä tarkkaavaisempi käyttäjä on. Dong ym. toteavat, että tunnusten pitäisi olla niin sensitiiviset kuin on käytännöllistä, mutta että sensitiivisyyden määrittely on erittäin subjektiivista. (Dong, Clark ja Jacob 2008, s. 54.)

Dong ym. esittävät aktiivisten hyökkäysten tilaisuuksiksi niitä tilanteita, joissa tunnisteita välitetään osapuolten välillä. Jotta käyttäjiä voitaisiin suojella aktiivisilta hyökkäysiltä, tulisi tarjota heille keinoja autentikoida palveluja, jotka tunnisteita kyselevät. Nämä keinot taas vaihtelevat niiden viestikanavien mukaan, joiden välityksellä autentikaatio tapahtuu. Vaikuttavia tekijöitä ovat miten palveluja viestikanavalla tunnistetaan, miten tunnistetietoja voidaan viestikanavalla todentaa ja mitkä toimijat ovat viestikanavalla osallisia. (Dong, Clark ja Jacob 2008, s. 54.)

Esimerkiksi verkkosivut tunnistetaan URL-osoitteiden perusteilla ja näiden autenttisuus SSL/TLS-

sertifikaateilla. Toimijoina viestikanavalla ovat niin asiakas-, palveluntarjoaja- kuin infrastuktuuritoimijat. Asiakkaita edustavat esimerkiksi verkkoselaimet ja alustat, joilla niitä käytetään, palveluita verkkosivujen palvelimet ja infrastriktuuria muun muassa DNS-palvelimet ja sertifikaattien validointipalvelimet. (Dong, Clark ja Jacob 2008, s. 57.)

Aktiivisia hyökkäyksien mahdollisuutta arvioidessa pitää siis Dongin ym. mukaan kiinnittää huomiota siihen, tarjotaanko käyttäjille riittävät ja luotettavat tiedot palvelujen autentikoimiseen. Lisäksi tarkastellaan onko käyttäjillä teknistä ja kontekstuaalista tietämystä, jotta he voivat tehdä päätöksiä tunnisteidensa antamisesta palveluille. Pitää myös tarkastella niiden oletusten paikkansapitävyyttä, mitä käyttäjistä tehdään. Jos käyttäjät eivät toimi oletusten mukaan, syntyy autentikaatioon haavoittuvuuksia. Dongin ym. mukaan käyttäjiltä odotettu tietämys sekä käyttäjien toiminnasta tehdyt oletukset pitäisi uhkia mallinnettaessa dokumentoida. (Dong, Clark ja Jacob 2008, s. 58-59) Dong ym. toteavat esimerkiksi, että useat käyttäjät eivät tarkista verkkosivujen URL-osoitteita ja SSL sertifikaatteja sivujen autentikoimiseksi ja jos turvallisuuden suunnittelu perustuu oletukseen niiden tarkistamisesta, syntyy haavoittuvuus aktiiviselle hyökkäykselle (Dong, Clark ja Jacob 2008, s. 62).

Tekninen tietämys auttaa käyttäjiä autentikoimaan jonkin tietyn ulkopuolisen tahon kun taas kontekstuaalinen tietämys auttaa käyttäjiä päättelemään onko jollain ulkopuolisella taholla oikeus tunnisteita pyytää. Tekninen tietämys riippuu autentikoinnissa käytetyistä viestintäkanavista. Tunnisteita oikeutetusti pyytävän tahon tunnistaminen muuttuu monimutkaisemmaksi mikäli jokin palvelu on delegoinut tunnisteiden kyselemisen kolmannelle osapuolelle tai oikeuttanut jonkin kolmannen osapuolen niitä kyselemään. (Dong, Clark ja Jacob 2008, s. 58-59)

Aktiivisia hyökkäyksiä voidaan toteuttaa viestimällä käyttäjien kanssa tekeytyen luotettavaksi tahoksi esimerkiksi sähköpostin välityksellä. Käyttäjät tarvitsevat kontekstuaalista tietoa voidakseen tunnistaa legitiimit yhteydenotot. Heille tulisi tehdä selväksi, mitä viestiyhteyksiä (esim. sähköpostiosoitetta) palvelu käyttää ottaessaan käyttäjiin yhteyttä ja missä tilanteissa käyttäjiin otetaan yhteyttä. (Dong, Clark ja Jacob 2008, s. 59-60)

2.1.4 Tunnisteiden murtaminen

Yhteydettömissä hyökkäyksissä (engl. offline attack) hyökkääjällä on hallussaan hajautusalgoritmilla salattu salasanatiiviste. Hyökkääjä koittaa saada salasanan selville vertailemalla arvaamistaan salasanoista hajautusalgoritmilla muodostamiaan tiivisteitä hallussaan olevaan tiivisteeseen. Väsytyshyökkäyksessä hyökkääjä arvaa salasanaa yrittämällä muodostaa kaikki järjestelmässä mahdolliset salasanat. Väsytyshyökkäykset ovat usein kuitenkin toteuttamiskelvottomia ajan tai laitteiston rajoitteiden vuoksi. Väsytyshyökkäyksiä nopeammissa sanakirjahyökkäyksissä käytetään arvaamisen pohjana sanalistoja. Sanakirjahyökkäyksissä käytetään hyödyksi myös sanojen muokkaamista erilaisten sääntöjen perusteella, sillä käyttäjät eivät useinkaan käytä salasanoinaan suoraan käytettävien sanakirjojen sanoja. Esimerkki muokkaussäännöstä voi olla vaikkapa numerojen lisääminen sanakirjan sanojen loppuun. Mitä pitempiä sanakirjoja käytetään, sitä hitaampaa eri muokkaussääntöjen käyttämisestä tulee ja siksi hyvien muokkaussääntöjen valinnasta tulee yhä tärkeämpää. (M. Weir ym. 2009, s. 392)

Väsytyshyökkäyksiä ja sanakirjahyökkäyksiä voidaan tehdä myös verkon yli, joskin silloin niitä voidaan torjua esimerkiksi kirjautumisyritysten määrää rajoittamalla (Grassi ym. 2017, s. 46). Herley ja Florêncio tuovat esiin tämän mahdollistavan kuitenkin palvelunestohyökkäyksen, jossa ihmisten pääsy käyttäjätileilleen estetään. Vaihtoehdoksi he esittävät oikealta näyttävien hunajapurkkitilien (honeypot account) luomista, joille kirjautuminen kuitenkin paljastaa hyökkääjän. (Herley ja Florêncio 2008, s 682-685.) Yhdysvaltojen National Institute of Standards and Technology (NIST) suosittelee salasanojen murtamisen vaikeuttamiseksi salasanojen korkeaa entropiaa sekä salasanojen säilyttämistä salattuna hajautusalgoritmia ja suolausta käyttämällä (Grassi ym. 2017, s. 46). Informaatioentropiaa ja salasanojen vahvuutta käsitellään lisää luvussa 2.2.1 .

2.1.5 Tunnisteiden varastaminen

Mitkä tahansa fyysisesti olemassaolevat tunnisteet voidaan luonnollisesti varastaa. Ensisijaisesti varastamisen voi ajatella koskevan laitteistotokeneita, jotka luonteensa puolesta ovat välttämättä varastettavissa. Toisaalta myös vaikka muistilapulle kirjoitetut salasanat voidaan

varastaa. NIST suosittelee varkauksilta suojelemiseksi monitekijäisen autentikaation käyttämistä siten, että yhtenä tekijänä on joko biometrinen autentikaattori tai muistettava autentikaattori (esimerkiksi salasana) (Grassi ym. 2017, s. 45).

2.1.6 Tunnisteiden kopioiminen

Tunnisteita voidaan kopioida joko käyttäjän tietämättä tai tietäen. Kopioita voidaan tehdä vaikkapa käyttäjän muistilapuille kirjoittamista tai tiedostoon tallentamista salasanoista, biometrisistä tunnisteista tai kryptografisesta avaimesta. (Grassi ym. 2017, s. 42.) Erilaisten tunnisteiden kopioimisen mahdollisuudet ovat selvästi erilaiset. Muistilapulle kirjoitetun salasanan kopioiminen on triviaalia, jos siihen päästään käsiksi. Biometriset tunnisteet taas perustuvat juuri niiden vaikeaan väärennettävyyteen (O'Gorman 2003, s. 45).

2.1.7 Tunnisteiden salakuuntelu

Tunnisteiden "salakuuntelussa" on kyse siitä, että hyökkääjä onnistuu saamaan tietoonsa tunnisteen käyttäjän ja tunnistavan osapuolen viestinnästä. Hyökkääjä voi esimerkiksi tarkkailla käyttäjän näppäimistöä salasanaa kirjoitettaessa joko itse katsomalla tai näppäilyntallentajaohjelman avulla. Tunniste voidaan saada haltuun myös salakuuntelemalla suojaamatonta viestintäkanavaa, kuten salaamatonta Wi-Fi-verkkoa. Salakuuntelulta voi suojautua käyttämällä autentikoitumisprotokollassa autentikoituja ja suojattuja viestintäkanavia, kuten TLS-salattua kanavaa. Salakuuntelun vaikutuksilta voi myös suojautua käyttämällä replay-hyökkäyksiltä suojaavaa autentikaatioprotokollaa (ks. luku 2.1.12). Voidaan myös pyrkiä varmistamaan käytettävän kirjautumisalustan luotettavuus erityisesti haittaohjelmistojen osalta. NIST kehottaa myös välttämään epäluotettavien langattomien verkkojen käyttämistä salaamattoman out-of-band -lisäautentikaation kanavana. Tällä tarkoitetaan autentikaatiota, jossa muodostetaan erillinen kanava jonkin salaisuuden tai autentikaatiopyynnön toimittamista varten. (Grassi ym. 2017, s. 42-43, 28-29 ja 17.)

2.1.8 Sivukanavahyökkäys

Sivukanavahyökkäyksessä saadaan tietoa tunnisteista autentikaatiossa syntyvien havaittavissa olevien “sivutuotteiden” pohjalta. Voidaan esimerkiksi selvittää kryptografinen avain laitteistotokenin virrankulutusta analysoimalla. Tällaisilta hyökkäyksiltä voidaan suojautua käyttämällä autentikoinnissa algoritmeja, jotka pitävät ajankäytön ja virrankulutuksen vakiona riippumatta niiden sisältämistä salaisista arvoista kuten kryptografisista avaimista. (Grassi ym. 2017, s. 43 ja 46.)

2.1.9 Sosiaalinen manipulointi

Sosiaalisessa manipuloinnissa hyökkääjä saa tunnisteet haltuunsa hyväksikäyttämällä luottamusta, jonka hyökkääjä rakentaa käyttäjän tai muun järjestelmän toimijan ja itsensä välille. Hyökkääjä voi tekeytyä esimerkiksi järjestelmän ylläpitäjäksi. Hyökkääjä voi myös saada itselleen vaikka tekstiviestillä välitettäviä kertakäyttöisiä salasanoja näyttelemällä käyttäjää puhelinoperaattorille saaden operaattorin ohjaamaan käyttäjän viestit uuteen numeroon. Sosiaaliselta manipuloinnilta suojautumiseen NIST kehottaa välttämään sellaisia autentikaattoreita, jotka ovat alttiita kolmansien osapuolien sosiaaliselle manipuoloinnille. (Grassi ym. 2017, s. 44 ja 46)

2.1.10 Epäluotettava kirjautumisalusta

Epäluotettava kirjautumisalusta syntyy silloin, kun siihen ujutetaan jotain kautta pahantahtoista ohjelmakoodia. Pahantahtoisella ohjelmakoodilla pystytään esimerkiksi välittämään käyttäjän tunnisteita väärille tahoille tai autentikoitumaan johonkin palveluun hyökkääjän puolesta. NIST esittää suojautumiskeinoksi muun muassa sellaisten laitteistotokenien käyttöä, jotka vaativat autentikoitumiseen käyttäjän toimintaa, esimerkiksi OTP-salasanoja (one time password) antavia tokeneita. (Grassi ym. 2017, s. 44.)

2.1.11 Tunnisteen kaappaaminen

Tunnisteen kaappaamisessa (engl. unauthorized binding) hyökkääjä kaappaa oikealle käyttäjälle tarkoitetun tunnisteen ja voi sitten esiintyä tunnisteen avulla oikeana käyttäjänä. Tätä

voidaan ehkäistä käyttämällä autentikaattoreita välitettäessä menetelmiä, jotka estävät mies välissä -hyökkäykset. Tunniste voidaan välittää henkilökohtaisesti tai käyttää autentikoituja suojattuja viestikanavia, kuten TLS-suojattuja. (Grassi ym. 2017, s. 45-46 ja 28-29.)

2.1.12 Replay-hyökkäys

Replay-hyökkäyksissä hyökkääjä kaappaa autentikoitujan autentikoitumisviestin ja lähettää sen itse palveluntarjoajalle päästen kirjautumaan sisään. Replay-hyökkäyksiä voidaan vastustaa käyttämällä autentikaatioprotokollassa nonceja tai haasteita. Palveluntarjoaja voi tunnistaa tällöin viesteistä, ettei niissä ole mukana asianmukaisia nonceja tai oikea-aikaisuudesta kertovaa tietoa. Replay-hyökkäyksiä vastustavia autentikaattoreita ovat muun muassa OTP-salasanoja antavat tokenit (ks. luku 2.4), kertakäyttöisiä salasanoja sisältävät listat ja kryptografistet autentikaattorit. (Grassi ym. 2017, s. 30.)

2.1.13 Istuntohyökkäykset

Istuntokaappauksissa hyökkääjä esiintyy käyttäjänä käyttämällä käyttäjän istunnon tunnistetta. Istunnon tunniste voi olla eväste tai HTTP-parametri ja kirjautumisen jälkeen istunnon tunnisteesta tulee käytännössä käyttäjän tunniste. (Johns 2011, s. 1189-1190.)

Käyttäjän istunnon tunniste voidaan saada selville esimerkiksi salakuuntelemalla salaamattomia langattomia verkkoyhteyksiä. Evästettä voidaan tämän jälkeen käyttää käyttäjänä esiintymiseen ja täten hyökkääjä voi saada selville käyttäjän henkilökohtaisia tietoja ja päästä käyttämään käyttäjätilin toimintoja. Evästeet voidaan saada selville myös hyväksikäyttämällä XSS-hyökkäystä (cross-site scripting), mikäli evästeelle ei ole asetettu HttpOnly-ominaisuutta. (Sivakorn, Polakis ja Keromytis 2016, s. 726.)

XSS-hyökkäyksissä hyökkääjä lisää JavaScript-koodia verkkosivuille sivuilla vierailijan selaimessa ajettavaksi. Koska koodi saadaan osaksi verkkosivua, sitä voidaan myös ajaa verkkosivun oikeuksilla ja siten päästään käsiksi esimerkiksi sivujen asettamiin evästeisiin. Hyökkäykseen tarkoitettu koodi voidaan saada sisällytettyä sivulle pysyvästi, jolloin käyttäjien selaimet suorittavat sen aina heidän pyytäessään kyseistä sivua. Hyökkäyskoodi voitaisiin sijoittaa myös URL-osoitteeseen, jota käyttämällä koodi päätyy sivujen suoritettavaksi. (Chaud-

hary, Gupta ja Gupta 2016, s. 2132-2133)

Chaudhary ym. sekä Gupta ja Gupta jakavat XSS-hyökkäykset pysyviin (persistent), ei-pysyviin (non-persistent) ja DOM-pohjaisiin (DOM-based) (Gupta ja Gupta 2017, s. 518-521; Chaudhary, Gupta ja Gupta 2016, s. 2132-2133) . Pysyvissä hyökkäyksissä tallennetaan ohjelmakoodia pysyvästi verkkosivuille tyypillisesti käyttäjille tarjottujen syötemahdollisuuksien kuten lomakkeiden kautta. Näin saadaan aikaan tilanne, jossa käyttäjien kysellessä hyökkäyksen kohteeksi joutunutta verkkosivua, pahantahtoinen ohjelmakoodi päätyy heidän selaimeensa ajettavaksi. Ei-pysyvissä hyökkäyksissä pahantahtoinen koodi saadaan käyttäjän selaimen suoritettavaksi siten, että palvelin palauttaa sen jossain muodossa käyttäjän selaimelle. Käyttäjä voidaan esimerkiksi huijata kysymään URL-osoitetta, johon on sijoitettu pahantahtoista koodia, jonka palvelin sisällyttää HTTP-vastaukseen esimerkiksi osana virheviestiä. DOM-pohjaisissa hyökkäyksissä pahantahtoista koodia ei ajeta ennen kuin sivujen oma koodi ajetaan. Palvelin ei sisällytä pahantahtoista koodia HTTP-vastaukseen, vaan verkkosivun oma koodi lukee sen selaimessa esimerkiksi URL:in kyselyparametreista. (Gupta ja Gupta 2017, s. 518-521)

Sekä Chaudhary ym. että Gupta ja Gupta tekevät katsausta XSS-hyökkäysten torjumiseen kehitettyihin keinoihin (Gupta ja Gupta 2017, s. 523-528; Chaudhary, Gupta ja Gupta 2016, s. 2133-2135) . Gupta ja Gupta esittävät, että tärkeintä XSS-hyökkäysten estämisessä on syötteiden turvallinen käsittely. Käyttäjien syötteitä sisällytettäessä verkkosivuille tulisi käyttää enkoodaus- ja validointimekanismeja. (Gupta ja Gupta 2017, s. 528.) Ohjeita XSS-hyökkäyksiltä suojautumiseen löytyy myös Open Web Appliation Security Projectin (OWASP) ylläpitämältä verkosivulta (Williams, Manico ja Mattatal 2019).

Sivakorn ym. käsittelevät useita keinoja evästeiden kaappaamisen estämiseksi:

1. **Kaikkien käyttäjäprofiileja sisältävien ja käyttäjäkokemusta personoivien verkkosivujen tulisi käyttää salausta kaikissa verkkoyhteyksissä.**
2. **Evästeiden secure- ja HttpOnly -asetusten käyttö.** HttpOnly-asetus rajaa evästeet käytettäväksi vain HTTP-pyynnöille ja siten estää evästeen kaappaamisen esimerkiksi XSS-hyökkäyksellä (Barth 2011, luku 4.1.2.6). Secure-asetuksella estetään evästeen lähettäminen salaamattoman verkkoyhteyden yli. Secure-asetuksella varustettuja

evästeitä voidaan kuitenkin ylikirjoittaa salaamattomien yhteyksien kautta, joten pelkkä secure-asetuksen käyttäminen ei evästeitä suojaa.

3. **HTTP Strict Transport Securityn (HSTS) käyttäminen.** HSTS:n avulla Strict-Transport-Security HTTP-otsikkoa käyttämällä verkkosivut voivat ohjeistaa selaimia ottamaan yhteyttä palvelimeen vain HTTPS:ää käyttäen. Tällöinkin ensimmäinen kysely sivuille voi paljastaa evästeet, koska selain ei vielä ole vastaanottanut ohjetta käyttää vain HTTPS:ää. On kuitenkin mahdollista päästä mukaan selainten käyttämille listoille, joiden perusteella selaimet osaavat käyttää HTTPS:ää heti ensimmäiselläkin kyselyllä. Sivakorn ym. tuovat esiin myös sen, että aiemmassa tutkimuksessa on huomattu monilla sovelluskehittäjillä olevan vaikeuksia toteuttaa HSTS oikein.
4. **Käyttäjien omat valinnat.** Käyttäjät voivat vähentää kaappauksia selaimen HTTPS Everywhere -laajennoksella, jonka tarkoitus on pakottaa selain käyttämään HTTPS-yhteyttä joka paikassa kun se on mahdollista. Sivakorn ym. suosittelevat myös VPN:n käyttöä silloin kun liitytään epäluotettaviin julkisiin langattomiin verkkoihin, sillä VPN vähentää käyttäjän verkkoliikenteen salakuuntelun riskiä. (Sivakorn, Polakis ja Keromytis 2016, s. 737 ja 726.)

Kirjautuneen käyttäjän istuntoa on joskus mahdollista käyttää hyväksi vaikkei tunniste olisi selvilläkään. Cross-site request forgeryssä (CSRF) pyritään saamaan käyttäjä tekemään hyökkääjän haluamia asioita ilman, että hyökkääjä saa selville käyttäjän tietoja. Tavanomaisesti selaimet liittävät verkkosivujen käyttämät tunnistetiedot (kuten evästeet) automaattisesti HTTP-pyyntöihin, jolloin riittää että käyttäjä saadaan huijattua tekemään jokin haluttu HTTP-pyyntö. (Yadav ja Parekh 2017.)

Mikäli verkkosivu käyttää haluttuun toimintoon GET-metodia ja URL:n kyselyparametreja, voitaisiin hyökkäys toteuttaa pelkällä linkillä. Se voitaisiin toimittaa käyttäjälle vaikka sähköpostilla tai se voitaisiin piilottaa jollekin verkkosivulle HTML:n img-elementtiin, jolloin pyyntö tapahtuisi automaattisesti käyttäjän vieraillessa verkkosivulla. Esimerkiksi jos Matti haluaisi siirtää Olavin tililtä itselleen rahaa, hän toimittaisi Olaville oheisen linkin vaikkapa sähköpostilla: `https://matinpankki.fi/siirto?tili=Matti&summa=1000.00`. Mikäli Olavi olisi kirjautunut pankin sivuille ja klikkaisi linkkiä, siirtyisi Matin tilille rahaa. POST-metodia käyttäviä toimintoja varten voitaisiin luoda lomake-elementti jollekin verk-

kosivulle ja saada käyttäjä klikkaamaan lähetä-painiketta tai lomake voitaisiin lähettää JavaScriptillä automaattisesti. Muita HTTP-metodejakin voitaisiin hyökkäyksissä käyttää ja välittää hyökkäyksissä tarvittavaa tietoa vaikka JSON-muodossa. (Yadav ja Parekh 2017.)

CSRF-hyökkäyksiltä voidaan suojautua monella tavalla. Esimerkiksi voidaan tarkastella HTTP-otsikoista tuleeko pyyntö samasta alkuperästä kuin sen kohde on (Yadav ja Parekh 2017) tai lisätä HTTP-pyyntöihin CSRF-tokeneja, joiden perusteella voidaan päätellä oliko pyyntö legitiimi vai ei (Konakandla ym. 2019). CSRF-hyökkäyksiltä suojautumiseen löytyy ohjeita OWASP:n ylläpitämältä verkkosivulta (Konakandla ym. 2019).

2.2 Salasanat

Salasanat ovat kaikkein yleisin tapa hoitaa autentikaatio (Woods 2016, s. 86; Brumen, Ivani ja Rozman 2016, s. 1923) , mutta perinteiseen salasanoilla toteutettavaan autentikaatioon liittyy kuitenkin laajalti tunnettuja ongelmia. Salasanoja voidaan pitää erittäin hyvinä autentikaation välineinä, mutta niiden määrän kasvaessa syntyy ongelmia käyttäjien muistamisvaikeuksien vuoksi (O'Gorman 2003). Nykyään ihmisillä on enemmän salasanoja kuin mitä he kykenevät muistamaan tai ainakin uskovat kykenevänsä muistamaan (Woods 2016, s. 89-90). Muun muassa muistivaikeuksien vuoksi käyttäjät omaksuvat turvallisuutta heikentäviä käytänteitä salasanojen hallintaan. Näitä ovat esimerkiksi salasanojen kirjoittaminen muistilapulle, salasanojen uudelleenkäyttäminen eri palveluissa ja helposti arvattavien salasanojen käyttäminen (Woods 2016, s. 86; Brumen, Ivani ja Rozman 2016, s. 1923; Wiedenbeck ym. 2005, s. 105; O'Gorman 2003, s. 2037) . Tutkimuksissa on osoitettu merkittävän osan salasanoista olevan murrettavissa sanakirjahyökkäyksillä (Bonneau ym. 2012, s. 557; Matt Weir ym. 2010, s. 163) . Herleyn ja Oorschotin mukaan käyttäjille salasanojen suurin ongelma onkin käytettävyys, jota heikentävät niin säännölliset vaihtamiset ja muut monimutkaiset käytänteet kuin salasanojen määrän kasvukin (Herley ja Van Oorschot 2012, s. 28).

Herley ja van Oorschot tuovat esiin, että vaikka salasanoille on yritetty löytää korvaajaa vuosikymmeniä, niin yritykset eivät ole juuri onnistuneet. He esittelevät salasanojen selkeitä etuja muihin kirjautumismenetelmiin nähden. Single sign-on -menetelmät kärsivät heidän mukaansa siitä, että yhden järjestelmän osan lamautuminen johtaa koko järjestelmän lamau-

tumiseen. Salasananhallintaohjelmat eivät usein tue käyttäjän liikkuvuutta. Käyttäjien tarvitsema laitteisto kuten älykortit maksavat ja niitä olisi hankala kantaa mukana suurta määrää. Salasanat ovat itseasiassa käyttäjää kohti käytännössä ilmaisia ja niiden uudelleenasettaminen voidaan automatisoida yksinkertaisesti, toisin kuin jotain fyysisiä tokeneita vaativissa autentikaatiomenetelmissä. Salasana-autentikaatio voidaan myös toteuttaa yksinkertaisesti sekä siinä tunnusten luominen onnistuu ilman viiveitä ja tunnusten peruminen toimii helposti vain salasanaa vaihtamalla. Käyttäjät myös osaavat valmiiksi käyttää salasanoja. Herley ja van Oorschot esittävät, että salasanojen vaihtoehtojen turvallisuushyödyt eivät aina yksinkertaisesti oikeuta kustannuksiaan. Heidän mukaansa monelle palvelulle toimintansa alkuvaiheessa asiakaskohtaisia autentikaatiokustannuksia ei saa olla. (Herley ja Van Oorschot 2012, s. 29) Herley ja van Oorschot toteavatkin, että salasanat tulevat olemaan laajassa käytössä hamaan tulevaisuuteen saakka (Herley ja Van Oorschot 2012, s. 33).

2.2.1 Hyvän salasanan luominen

Brumenin ym. mukaan salasanaa voidaan pitää hyvänä ja vahvana, mikäli sitä ei saada selville väsytyshyökkäyksellä (brute-force -attack), sanakirjahyökkäyksellä (dictionary attack) tai arvaamalla (Brumen, Ivani ja Rozman 2016, s. 1923). Egelman ym. esittävät salasanan vahvuuden määrittäjäksi sitä, miten hyvin se kykenee vastustamaan nykyaikaisia salasanamurtamistyökaluja (Egelman ym. 2013, s. 2380).

Brumen ja ernezel lähestyvät salasanan vahvuuden arvioimista laskemalla mahdollisten kombinaatioiden määrää eri pituisille salasanoille kun merkit poimitaan 75 merkin joukosta, jotka edustavat isoja ja pieniä kirjamia, numeroita ja erikoismerkkejä. Sitten he laskevat kuinka pitkään konferenssijulkaisun tekemisen aikana käytössä olleella laskentateholla menee aikaa kaikkien suolaamattomalla hajautusalgoritmilla hajautettujen salasanakombinaatioiden läpikäymiseen. Brumen ja ernezel ottavat huomioon myös Mooren lain, jonka mukaan tietokoneiden laskentateho kaksinkertaistuu 1,5 vuoden välein ja laskevat sillä oletuksella kaikkien salasanakombinaatioiden läpikäymiseen kuluvan ajan kymmenen vuoden päästä. Tulokset näkyvät taulukossa 1. (Brumen ja Černezel 2014, s. 1366-1367.)

Taulukko 1: Kaikkien salasanojen läpikäyminen eri pituuksilla, kun merkit valitaan salasanaan sattumanvaraisesti 75 käytettävän merkin joukosta (Brumen ja Černezel 2014, s. 1367).

Salasanan pituus	Kombinaatioiden määrä	Murtamisaika nyt	Murtamisaika kymmenen vuoden päästä
7	1,00113E+15	11 päivää, 10 tuntia	2 tuntia, 46 minuuttia
8	7,50847E+16	2 vuotta, 139 päivää	8 päivää, 16 tuntia
9	5,63135E+18	178 vuotta, 207 päivää	1 vuosi, 286 päivää
10	4,22351E+20	13 392 vuotta	133 vuotta, 338 päivää
11	3,16764E+22	1 004 450 vuotta	10 044 vuotta

Salasanan vahvuuden mittaamisesta on erilaisia näkemyksiä. Salasanan pituuteen (N) ja käytettävissä olevien merkkien määrään (C) perustuvaa vahvuuden määritelmää (N*$\log_2$C) on kyseenalaistettu (Egelman ym. 2013, s. 2380) ja paremmaksi määrittäjäksi on esitetty esimerkiksi salasanan yleisyyttä (Herley ja Van Oorschot 2012, s. 34) Salasanan vahvuus on myös kontekstiriippuvaista eikä Herleyn ja van Oorschotin mukaan ole olemassa konsensusta tarvittavasta vahvuudesta eri konteksteissa. Voidaan kuitenkin esimerkiksi sanoa, että verkon yli tapahtuvia hyökkäyksiä vastaan salasanojen ei tarvitse olla yhtä vahvoja kuin offline-hyökkäyksiä vastaan, sillä salasanojen arvaamista voidaan rajoittaa vaikkapa lukitsemalla kirjautuminen riittävän monen epäonnistuneen yrityksen jälkeen (Herley ja Van Oorschot 2012, s. 33-34).

Salasanan julkisuusaste voi olla matala, mikäli järjestelmä asettaa sen. Mikäli käyttäjä asettaa salasanan, sen julkisuusastetta ei voida etukäteen tietää. Matala julkisuusaste vähentää mahdollisuutta passiivisten hyökkäysten toteuttamiseen, joissa ei vaadita käyttäjän omaa aktiivisuutta. (Dong, Clark ja Jacob 2008, s. 50-52.) Vahvat salasanat ovat Brumenin ym. mu-

kaan satunnaisesti valittuja sekä 11-merkkisiä tai pidempiä, mutta ne ovat käyttäjille vaikeita muistaa. Henkilökohtaisiin tietoihin tai ominaisuuksiin kuten syntymäaikaan tai osoitteeseen perustuvat salasanat taas ovat helppoja murtaa. (Brumen, Ivani ja Rozman 2016, s. 1923) Olisi yksinkertaista laittaa järjestelmä satunnaisgeneroimaan käyttäjille matalan julkisuusasteen salasanat, jotta vältyttäisiin käyttäjien asettamilta korkean julkisuusasteen salasanoilta. Tämä kuitenkin siis heikentäisi salasanan muistettavuutta.

Brumen ym. esittelevät muitakin tapoja luoda salasanoja. Kognitiivisiin salasanoihin perustuvassa auntentikaatiossa käyttäjälle esitetään sarja satunnaisesti valittuja kysymyksiä, joihin vain oikea käyttäjä osaisi vastata. Tällaiset kysymykset ovat kuitenkin usein käyttäjän perheen ja ystävien arvattavissa. (Brumen, Ivani ja Rozman 2016, s. 1923.) Ongelma on siis se, että kysymykset todennäköisesti perustuisivat melko korkean julkisuusasteen tietoihin. Salasanoja voidaan myös luoda satunnaisista helposti lausuttavista tavuista, mutta nämäkin ovat alttiita tietynlaiselle väsytyshyökkäykselle (Brumen, Ivani ja Rozman 2016, s. 1924). Käyttäjät muistavat helpommin mnemonisia salasanoja, jotka muodustuvat joidenkin kieliopillisesti oikeellisten lauseiden sanojen kirjaimista (esim. kunkin sanan ensimmäisestä kirjaimesta), mutta Brumen ym. toteavat niidenkin olevan omalla laillaan haavoittuvaisia (Brumen, Ivani ja Rozman 2016, s. 1924).

Brumen ym. esittelevät myös PsychoPass -metodin, jossa salasana luodaan ja muistetaan toimintasarjan perusteella (Brumen, Ivani ja Rozman 2016, s. 1924). Toimintasarja muodostetaan siirtymistä näppäimistön näppäimiltä toisille tiettyjen sääntöjen mukaan, joita kuvataan PsychoPassin alkuperäisessä esittelyssä (Cipresso ym. 2012) sekä Brumenin ym. parannuksissa (Brumen ym. 2013). Brumenin ja ernezelin mukaan PsychoPass-menetelmällä voidaan muodostaa turvallisia sekä muistettavia salasanoja (Brumen ja Černezel 2014, s. 1369).

On siis olemassa useita suosituksia siitä, millä käytännöillä salasanoja pitäisi muodostaa. Käyttäjä voidaan pakottaa luomaan salasanansa erilaisten käytänteiden mukaan, mutta on havaittu myös, että käyttäjät luovat salasanoja hyvin ennustettavasti niistä huolimatta. Esimerkkinä toimii numeroiden lisääminen vain yksinkertaisesti salasanan perään niitä vaadittaessa. (Grassi ym. 2017, s. 68.) Herleyn ja van Oorschotin mukaan salasanojen pakotettua säännöllistä vaihtamista on usein kuvailtu käytettävyyskatastrofiksi, eikä siitä kuitenkaan ole suuresti hyötyä turvallisuuden lisäämisessä (Herley ja Van Oorschot 2012, s. 34). Salasano-

jen turvallisuus riippuu siis pitkälti käyttäjien valinnoista, mutta kuten myös luvussa 2.1.1 tuodaan esiin, käyttäjiä on vaikea ohjata turvallisuutta lisääviin valintoihin.

Brumen ym. esittävät oman pienen tutkimuksensa perusteella, että tiukat salasanojen luomissäännöt heikentävät käyttäjien kykyä muistaa salasanojaan. Koehenkilöt eivät pystyneet juuri muistamaan tiukkojen salasanakäytänteiden mukaan luotuja salasanoja, mutta toisaalta Brumenin ym. tutkimuksessa ei kuitenkaan ollut mukana kontrolliryhmää, joka olisi käyttänyt salasanojen luonnissa löysiä sääntöjä. (Brumen, Ivani ja Rozman 2016, .) Woods taas esittää tutkimuksensa perusteella, että käyttäjien muistin hyvyys ei ennusta heidän kykyään muistaa salasanoja vaan esimerkiksi salasanojen uusiokäyttö perustuu enemmän käyttäjän uskoon kyvystään muistaa salasanoja (Woods 2016, s. 61). Joka tapauksessa on selvää, että salasanojen uusiokäyttö ja muut aiemmin mainitut salasanojen turvallisuutta heikentävät tavat ovat yleisiä keinoja salasanojen käytettävyyden parantamiseksi. Salasanoissa on siis käytettävyysongelma, joka on ratkaistava.

NIST on muuttanut salasanaohjeistustaan vuonna 2017. Salasanojen rakenteelle ei suositella muita rajoitteita kuin vähintään kahdeksan merkin pituus ja järjestelmän satunnaisgeneroiman salasanan pituudeksi riittää joissain olosuhteissa jopa vain kuusi merkkiä. NIST myös kehottaa haavoittuvien salasanojen listan ylläpitämiseen, minkä perusteella palvelu voi hylätä käyttäjän valitsemia haavoittuvia salasanoja. Salasanojen vaihtamista suositellaan nykyään vain silloin, kun salasanojen yksityisyyden tiedetään vaarantuneen. NIST suosittelee pitkien salalauseiden käyttämistä ja tukemista järjestelmissä. Esimerkiksi olisi mahdollistettava vähintään 64-merkkisten salasanojen käyttö sekä välilyönnit salasanan osana. (Grassi ym. 2017) Salalauseita on pidetty muistettavina ja vaikeina murtaa, mutta niiden käytettävyyttä on kyseenalaistettu kun salasanoja on käytettävä usein sekä mobiililaitteita käytettäessä (Brumen, Ivani ja Rozman 2016, s. 1923-1924). NIST:in salasanoja koskevassa ohjeistuksessa esitetään useita muitakin asioita, kuten kehotus salasanojen säilyttämisestä salattuna hajautusalgoritmilla ns. suolauksen kanssa sekä epäonnistuneiden kirjautumisyritysten määrän rajoittaminen (Grassi ym. 2017).

Egelmanin ym. tutkimuksen perusteella näyttää siltä, että salasanan luomisen yhteydessä näytettävillä salasanan vahvuusmittareilla on positiivinen vaikutus salasanojen vahvuuteen silloin, kun salasana annetaan tärkeille käyttäjätileille. Ei-tärkeiden käyttäjätilien salasanoi-

hin mittareilla ei ollut vaikutusta. Salasanojen muistettavuus ei kärsinyt mittareiden näyttämisen seurauksena. (Egelman ym. 2013) NIST kehottaa ohjeistuksessaan vahvuusmittarien käyttöön (Grassi ym. 2017).

Komanduri ym. perustavat omassa tutkimuksessaan salasanan vahvuuden informaatioentropiaan, joka kuvaa bitteinä salasanan sisältämän informaation oletettua määrää. Korkeampi entropia tarkoittaa salasanan suurempaa kompleksisuutta. Komandurin ym. käyttämä informaatioentropiaa ennustava menetelmä perustuu C. E. Shannonin kehittämään menetelmään. He myös testaavat salasanojen vahvuuksia sanakirjahyökkäyksellä. (Komanduri ym. 2011.)

Komaduri ym. testaavat erilaisten salasananluomiskäytänteiden vaikutusta käyttäjien luomien salasanojen vahvuuksiin. Koehenkilöille annettu skenaario oli, että he luovat uutta salasanaa pääasialliselle sähköpostitililleen sen jälkeen, kun salasanoja on tietomurrossa päässyt paljastumaan ja sähköpostipalvelu on muuttanut salasanakäytäntöjään. Koehenkilöitä pyydettiin toimimaan kuin he oikeasti toimisivat ja tekemään niitä asioita, joiden avulla he normaalisti muistavat ja suojaavat salasanojaan. Salasanojen muodostamiskäytännöt olivat seuraavat:

1. Yksinkertaiset 8-merkkiset: Salasanassa tuli olla vähintään 8 merkkiä.
2. 16-merkkiset: Salasanassa tuli olla vähintään 16 merkkiä.
3. Sanakirjatarkistetut 8-merkkiset: Salasanassa tuli olla vähintään 8 merkkiä eikä yhtään sanakirjasta löytyvää sanaa. Järjestelmä hylkäsi salasanan, mikäli siinä oli jokin sanakirjasta löytyvä sana sen jälkeen kun kaikki muut merkit kuin kirjaimet poistettiin salasanasta.
4. Monimutkaiset 8-merkkiset: Salasanassa tuli olla vähintään 8 merkkiä ja vähintään yksi pieni kirjain, iso kirjain, numero sekä erikoismerkki. Lisäksi salasanassa ei saanut olla sanakirjasta löytyviä sanoja.
5. Yksinkertaiset 8-merkkiset, eri skenaario: Salasanassa tuli olla vähintään 8 merkkiä. Koehenkilöille ei annettu sähköpostiskenaariota. Heille sen sijaan sanottiin salasanan muistamisen olevan tärkeää, sillä kyselyn vastaukset linkitettäisiin salasanan avulla. (Komanduri ym. 2011.)

Taulukko 2: Eri salasanluomiskäytänteiden vaikutuksia salasanojen ominaisuuksiin ja käyttäjien toimintaan (Komanduri ym. 2011).

	Yks. 8-merk.	16-merk.	Sanakirjatark. 8-merk.	Monim. 8-merk.
Entropia:	29,43 bittiä	44,67 bittiä	28,99 bittiä	34,30 bittiä
Murrettu sanakirja-hyökkäyk-sellä:	188/972	9/971	31/963	0/997
Murrettu sanakirja-hyökkäyk-sellä sekä läpäisi sanakirja-tarkistuk-sen:	99/972	6/971	26/963	0/997
Murrettu sanakirja-hyökkäyk-sellä eikä löytynyt laajasta sa-nakirjasta:	28/972	1/971	3/963	0/997
Keskim. salasanan muodostamis-yritysten määrä:	1,13	1,66	1,88	3,35

	Yks. 8-merk.	16-merk.	Sanakirjatark. 8-merk.	Monim. 8-merk.
Ennen tutkimuksen ensimmäisen vaiheen loppuun suorit- tamista kes- keyttäneet:	14,7%	15,0%	18,3%	25,0%
Salasanan täysi uudel- leenkäyttö:	25,6%	2,3%	19,7%	3,3%
Salasanan osit- tainen uudelleen- käyttö:	19,1%	25,0%	19,2%	31,3%
Loi kokonaan uuden salasanan:	48,2%	66,0%	53,3%	62,5%

Tutkimus sisälsi useita kiinnostavia havaintoja. Komandurin ym. mukaan NIST:in informaatioentropian laskutapa ennustaisi kahdeksanmerkkisten isoja ja pieniä kirjaimia sekä numeroita ja erikoismerkkejä sisältävien salasanojen sekä 16-merkkisten salasanojen olevan entropialtaan yhdenvertaiset. Kuitenkin heidän omassa analyysissään 16-merkkisten salasanojen entropia oli tosiasiassa huomattavasti suurempi. Toisaalta 16-merkkisistä salasanoista pieni osa oli murrettavissa sanakirjahyökkäyksellä, kun taas monimutkaisista 8-merkkisistä yksikään ei murtunut. Sanakirjatarkistetut 8-merkkiset salasanat eivät olleet entropialtaan suurempia kuin yksinkertaiset 8-merkkiset, mutta sanakirjahyökkäystä ne vastustivat selvästi

paremmin. Mikäli kaikki salasanat olisi luomishetkellään tarkastettu laajemmalla 24,7 miljoonaa sanaa sisältäneellä sanakirjalla olisi saatu kiinni suurin osa kaikista niistä salasanoista, jotka onnistuttiin sanakirjahyökkäyksellä murtamaan. Sanakirjatarkistus kuitenkin lisäsi käyttäjien ärtymystä ja vaikeuksia salasanaa luodessa. (Komanduri ym. 2011.)

Komanduri ym. päättelevät tuloksistaan myös, että monimutkaiset 8-merkkiset salasanat olivat koehenkilöille kaikkein vaikeimpia. Niiden luominen vaati koehenkilöiltä eniten yrityksiä ja sai muihin luomistapoihin verrattuna suuremman määrän koehenkilöitä keskeyttämään tutkimukseen osallistumisen. Vahvuudeltaan samaa luokkaa olevat 16-merkkiset salasanat olivat käyttäjille selvästi helpompia luoda. Monimutkaisia 8-merkkisiä ja 16-merkkisiä salasanoja luodessaan koehenkilöt uusiokäyttivät vähiten vanhoja salasanoja, mikä on turvallisuuden kannalta hyvä. Komanduri ym. esittävät tämän saattaneen kuitenkin johtua siitä, että käyttäjillä ei olisi ollut valmiiksi yhtä tiukoilla kriteereillä luotuja salasanoja. Monimutkaisia 8-merkkisiä ja 16-merkkisiä kirjoitettiin muistiin kaikkein eniten, monimutkaisia 8-merkkisiä selvästi enemmän kuin 16-merkkisiä. Käyttäjät myös kokivat näiden salasanojen luomisen ärsyttävimmäksi ja vaikeimmaksi, monimutkaisten 8-merkkisten ollen 16-merkkisiä monimutkaisempia ja vaikeampia. Toisaalta käyttäjät myös kokivat näiden vaikeampien salasanojen lisäävän turvallisuutta, minkä Komanduri ym. esittävät mahdollisesti lisäävän käyttäjien halukkuutta noudattaa tiukempia salasankäytäntöjä. (Komanduri ym. 2011.)

Monimutkaiset 8-merkkiset ja 16-merkkiset salasanat vastustivat samantasoisesti murtamisyritystä, mutta Komanduri ym. tuovat esiin, että heidän käyttämänsä murtotyökalu on optimoitu lyhyille salasanoille. Siten 16-merkkisten ja monimutkaisten 8-merkkisten salasanojen samantasoisesta kyvystä vastustaa murtamista ei voi olla täysin vakuuttunut. 16-merkkiset salasanat olivat kuitenkin selvästi käyttäjäystävällisempiä ollen kaikilla mittareilla vähintään yhtä käytettäviä kuin monimutkaiset 8-merkkiset. Komanduri ym. tuovat esiin myös, että 16-merkkiset salasanat olivat käytettävyydeltään vain vähän huonompia kuin 8-merkkiset sanakirjatarkastetut, vaikka kykenevät vastustamaan väsytyshyökkäyksiä huomattavasti paremmin. (Komanduri ym. 2011.) Komandurin ym. tutkimuksen perusteella voisi esittää, että salasanojen rajoittaminen vähintään 16-merkkisiin ilman muita vaatimuksia olisi hyvä kompromissi salasanojen vahvuuden ja käytettävyyden välillä.

2.3 Biometriikka

Biometriset autentikaattorit jaetaan yleensä kahteen luokkaan:

1. Pysyväisluonteiset kehon ominaisuudet, kuten sormenjäljet ja kasvot.
2. Käyttäytyminen, kuten näppäimistön käytön tavat ja puhe. (Bhattacharyya ym. 2009, s. 14; O'Gorman 2003, s. 2025-2027.)

Jako sisältää O'Gormanin mukaan monitulkintaisuutta, sillä kaikki biometriset autentikaattorit perustuvat jossain määrin henkilön pysyväisluonteisiin fyysisiin ominaisuuksiin. Puheen tapauksessa näitä ovat muun muassa äänihuulet ja keuhkot. O'Gorman esittääkin jakoa perustuen biometrisen signaalin tyyppiin:

1. Vakaa biometrinen signaali (stable biometric signal).
2. Muuttuvaisluonteinen biometrinen signaali (alterable biometric signal).

(O'Gorman 2003, s. 2025-2027.)

Biometriset signaalit tunnistetaan tavallisesti niiden ominaisuuksista muodostettuja malleja vastaan vertaamalla. Vakaiden biometristen signaalien mallit muodostetaan suoraan biometrisesta signaalista. Muuttuvaisluonteisen biometrisen signaalin malli sisältää vakaan biometrisen komponentin ja muuttujan. Puheen tapauksessa pysyvä komponentti on ääntöväylä ja muuttuja on jokin sana tai lause. (O'Gorman 2003, s. 2025-2027.)

Muuttuvaisluonteista biometrista signaalia voidaan verrata malliin kokonaisuutena tai se voidaan jakaa osiin. Esimerkiksi puheessa muuttujaosa voi toimia salasanana kun taas puhuja tunnistetaan signaalin vakaan biometrisen komponentin perusteella. Muuttuvaisluonteinen biometrinen signaali voi myös olla osa haaste-vastaus -autentikoitumista toisin kuin aina samana pysyvä vakaa biometrinen signaali. (O'Gorman 2003, s. 2025-2027.)

Biometriseen autentikoitumiseen liittyy järjestelmästä itsestään johtuvia virhetilanteita. Biometrisessä autentikaatiossa voidaan erilaisten epätarkkuustekijöiden vuoksi tunnistaa henkilö virheellisesti joksikin henkilöksi tai olla virheellisesti tunnistamatta jotakin henkilöä, joka olisi pitänyt tunnistaa. Biometriseen autentikaatioon liittyy siis virheiden osalta kaksi tekijää:

1. Väärien positiivisten tunnistamisten yleisyys (false match rate, FMR).
2. Väärien negatiivisten tunnistamisten yleisyys (false nonmatch rate, FNMR).

(O'Gorman 2003, s. 2025-2027.)

Bhattacharyya ym. esittelevät useita muitakin biometriseen autentikaatioon liittyviä käsitteitä. Yksi näistä on relative operational characteristic (ROC). Biometrisissa järjestelmissä näytteen tunnistamista määritteleviä parametreja säätelemällä voidaan lisätä tai vähentää FMR:ää ja FNMR:ää suhteessa toisiinsa, ja ROC-kuvaaja näyttää miten se tapahtuu. Equal error rate (EER) taas on se piste, jossa FMR ja FNMR ovat yhtä suuret. Alempi EER tarkoittaa tarkempaa järjestelmää ja sitä käytetäänkin usein järjestelmien nopeaan vertailuun. (Bhattacharyya ym. 2009, s. 22-23.)

Biometristen järjestelmien virheet voidaan jakaa verifikaatiovirheisiin ja identifikaatiovirheisiin. Verifikatiovirhe on väärä positiivinen tai väärä negatiivinen tunnistaminen verrattaessa yhtä biometrista signaalia yhteen käyttäjään. Identifikaatiovirhe taas on väärä positiivinen tai väärä negatiivinen tunnistaminen verrattaessa biometrista signaalia useaan käyttäjään. Väärän positiivisen tunnistamisen yleisyys identifikaatiojärjestelmässä riippuu järjestelmässä olevien vertailtavien biometristen näytteiden määrästä olettaen, että näytteet ovat itsenäisiä. Mitä enemmän näytteitä on, sitä todennäköisempiä väärät positiiviset tunnistamiset ovat. Tämän vuoksi väärät positiiviset tunnistamiset ovat yleisempiä identifikaatioissa kuin verifikaatiossa. (O'Gorman 2003, s. 2025-2027.)

Biometrisen tunnistautumisen etuina nähdään se, että tunnisteet ovat aina mukana eikä niitä voi unohtaa sekä se, että biometriset tunnisteet ovat muita tunnisteiden tyyppejä tiukemmin sidoksissa käyttäjän identiteettiin (Mayron, Hausawi ja Bahr 2013, s. 197). Biometrisen tunnisteen on siis esitetty takaavan paremmin, että autentikoituja on todella se henkilö jolle tunnisteet kuuluvat, sillä biometrisia tunnisteita ei voi lainata tai varastaa yhtä helposti kuin salasanoja ja tokeneita. Tämän vuoksi autentikoituja ei voisi kiistää osallisuuttaan johonkin tapahtumaan, joka autentikoituneena on tehty. Tosiasiassa biometrisia tunnisteita kuitenkin voidaan väärentää ja varastaa. (O'Gorman 2003, s. 2022.)

Biometristen tunnisteiden julkisuusaste on Dongin ym. mukaan keskitasoinen tai korkea. Esimerkiksi sormenjälkiään ihminen jättää jatkuvasti ympäristöönsä ja mahdollisesti myös

autentikaatiojärjestelmiin. Tämän vuoksi biometriset tunnisteet ovat alttiita passiivisille hyökkäyksille, joissa ei vaadita käyttäjän omaa aktiivisuutta. (Dong, Clark ja Jacob 2008, s. 50-52.) Liou ja Bhashyam tuovat esiin, että mikäli tunnisteiden lukijalaite ei ole täysin turvallinen ja vartioitu, on varastettua tunnistetta helppo käyttää replay-hyökkäyksessä (Liou ja Bhashyam 2010, s. 48-49). Mikäli varastettu biometrinen tunniste on vakaa biometrinen signaali, sitä ei pitäisi enää koskaan käyttää uudestaan autentikoitumiseen, sillä sitä ei voi mitenkään muuttaa (O'Gorman 2003, s. 2034). Mayron ym. pitävät tätä biometrisen tunnistautumisen suurimpana heikkoutena (Mayron, Hausawi ja Bahr 2013, s. 197). Biometristen tunnisteiden malleja voidaan kuitenkin suojata varkauksilta erilaisilla tavoilla, joita käyvät läpi esimerkiksi Murillo-Escobar ym. (Murillo-Escobar ym. 2015). Biometrinen tunnistautuminen vaatii myös laitetta, joka lukee biometrisen signaalin autentikoiduttaessa, mikä lisää autentikoitumisen kustannuksia (O'Gorman 2003, s. 231).

2.4 Tokenit

Tokenit ovat kirjautumisen väline, jotka kuuluvat luvussa 2 määriteltyyn "jotain mitä käyttäjillä on hallussaan" -luokkaan. Tokeneita käytetään usein kaksivaiheisessa tunnistuksessa salasanojen lisäksi (Liou ja Bhashyam 2010, s. 48). Tyypillisenä esimerkkinä toimii pankkikortin ja salasanan yhdistäminen, jolloin salasana suojaa varastettua pankkikorttia (O'Gorman 2003, s. 2024-2025). Luonnollisesti voidaan myös sanoa, että pankkikortti suojaa varastettua salasanaa. Fyysisen tokenin katoamisen myös huomaa, mikä tarjoaa käyttäjälle tiedon tunnisteensa vaarantumisesta (O'Gorman 2003, s. 2037). Tokeniin voi myös varastoida mielivaltaisen pitkän salasanan, mikä ehkäisee salasanan arvaamista (O'Gorman 2003, s. 2032).

Aloul ym. jakavat tokenit fyysisiin tokeneihin ja ohjelmatokeneihin. Fyysiset tokenit ovat pieniä mukana kannettavia esineitä, jotka saattavat sisältää kirjautumista varten esimerkiksi biometristä dataa, salausavaimia tai ajan myötä vaihtuvia PIN-koodeja. Ohjelmatokenit ovat ohjelmia, jotka esimerkiksi näyttävät PIN-koodeja. PIN-koodi täytyy antaa kirjautumisen yhteydessä tavallisen salasanan lisäksi, jotta kirjautuminen onnistuu. (Aloul, Zahidi ja El-Hajj 2009, s. 642) Fyysisiksi tokeneiksi voi selvästi laskea myös useita kertakäyttöisiä OTP-salasanoja (one time password) sisältävät tunnuslukukortit, joita esimerkiksi osa pankeista käyttää verkkopankkiin kirjautumisessa. Pankit ovat kuitenkin luopumassa näistä listoista

turvallisempina pitämiinsä vaihtoehtoihin, kuten tunnuslukuja antaviin sovelluksiin. (Lehto 2018; Nordea 2019, .)

Liou ja Bhashyam kirjoittavat turvatokeneista (security token), virtuaalitokeneista (virtual token) ja ohjelmatokeneista (software token). Turvatokenit ovat fyysisiä esineitä, jotka näyttävät OTP-salasanoja, virtuaalitoken muodostuu jollekin kuljetettavalla muistilaitteelle kuten USB-muistille tallennettavasta suojatusta tiedostosta ja ohjelmatokenit ovat tietokoneelle ladattavia ohjelmistoja, jotka näyttävät OTP-salasanoja. (Liou ja Bhashyam 2010, s. 49.)

PIN-koodeja näyttävistä ohjelmatokeneista Aloul ym. toteavat, että niiden on toteutettava OTP-algoritmi (One Time Password algorithm). OTP-algoritmin on syytä tuottaa mahdollisimman sattumanvarainen ja ennustamaton sekä peruuttamaton (irreversible) PIN-koodien sarja, jotta kukaan ei voi arvata tulevia PIN-koodeja. (Aloul, Zahidi ja El-Hajj 2009, s. 642) Lioun ja Bhashyamin mukaan PIN-koodeja tuotetaan pääasiassa aikasynkronoiduilla tai tapahtumapohjaisilla algoritmeilla. Aikasynkronoidut algoritmit luovat uuden salasanan ajan pohjalta tietyin aikavälein ja jotkut tähän perustuvat tokenit vaativat ajoittaisia uudelleensynkronointeja palvelimen kanssa. Tapahtumapohjainen algoritmi voi antaa OTP-salasanan esimerkiksi käyttäjän painaessa tokenin painiketta. (Liou ja Bhashyam 2010, s. 49.)

Tokeneista voi koitua niitä käyttäville organisaatioille isoja kustannuksia ostamisen, asentamisen, ylläpidon ja käyttökoulutuksen kautta. Käyttäjät taas joutuvat kuljettamaan ja ylläpitämään useita eri tokeneita useita eri sovelluksia varten. (Tanvi, Sonal ja Kumar 2011, s. 85; Aloul, Zahidi ja El-Hajj 2009, s. 642.) Raha- ja aikakustannuksia koituu myös kadonneiden tokeneiden korvaamisesta. Tokeneita saatetaan joutua vaihtamaan ajoittain myös pattereiden kuluessa loppuun. Virtuaaliset tokenit vähentävät kustannuksia, koska niiden kanssa voidaan hyödyntää käyttäjillä jo valmiiksi olevia muistilaitteita. Liou ja Bhashyam kuitenkin toteavat, ettei virtuaalisten tokeneiden levittämistä voida kontrolloida. (Liou ja Bhashyam 2010, s. 49.)

Yksi ratkaisu tokenien ongelmiin on puhelimen käyttäminen tokenina, sillä suurin osa ihmisistä kuljettaa puhelinta jatkuvasti mukanaan ja puhelimeen voidaan asentaa useiden tahojen sovelluksia kirjautumiskäyttöä varten (Aloul, Zahidi ja El-Hajj 2009, s. 641). Aloul ym. esittävätkin oman token-ratkaisunsa. Siinä käyttäjän puhelimeen ladataan sovellus, jonka kautta

käyttäjä saa itselleen OTP-salasanan kirjautumista varten joko yhteydettömällä tavalla tai SMS-viestien välityksellä. (Aloul, Zahidi ja El-Hajj 2009, s. 642.)

Yhteydettömässä tavassa sovellus luo OTP-salasanoja perustuen erilaisiin tekijöihin, joita ovat mm. puhelimen IMEI-koodi, aika ja päivämäärä. Samojen OTP-salasanojen luomiseen vaadittavien tekijöiden täytyy olla myös varastoituna palvelimelle ja puhelimen sekä palvelimen ajan täytyy olla synkronoitu, jotta palvelin pystyy päättelemään onko sen vastaanottama OTP-salasana oikea. SMS:ään pohjautuvassa tavassa käyttäjä pyytää palvelimelta SMS-viestillä OTP-salasanaa ja todistaa henkilöllisyytensä sisällyttämällä SMS-viestiin jotain itselleen uniikkia informaatiota. Nämä tiedot ovat käyttäjän sovelluksessa tarvitsema käyttäjänimi, PIN-koodi ja puhelimen tunnistetiedot. Tiedot salataan lähetettäessä 256-bittisellä symmetrisellä avaimella. (Aloul, Zahidi ja El-Hajj 2009, s. 642-643.)

2.5 Konteksti

Zhongin, Yanin, Yun, Zhaon ja Wangin mukaan suurin osa autentikaatioteknologioista tarkastaa käyttäjän identiteetin vain tietyllä hetkellä perustuen käyttäjätilin nimeen ja salasanaan. Salasana voidaan kuitenkin varastaa, eikä salasanan tietäminen vielä takaa henkilön aitoutta. Zhong ym. hakevatkin ratkaisua phishing-hyökkäyksiin jatkuvan internetselailun tarkkailun avulla. (Zhong ym. 2014, s. 279.)

Zhongin ym. mukaan ihmisten verkkoselauskäyttäytyminen on jälittelemätöntä eikä helposti muutettavissa lyhyellä aikavälillä, minkä vuoksi sen mallintamisen avulla voidaan tunnistaa käyttäjä. Zhongin ym. menetelmässä tarkkaillaan käyttäjän verkkosivujen selailua koko laajuudeltaan eikä pelkästään yhden sivuston osalta. (Zhong ym. 2014, s. 279-278.) Tarkailussa tehdään käyttäjän verkkoselauksesta uutta mallinnusta 31 päivän välein edellisen 30 päivän perusteella. Tämä sen vuoksi, että ihmisten selauskäyttäytyminen muuttuu ajan kuluessa. Järjestelmä kykeni havaitsemaan 91,3% vääristä käyttäjistä väärien positiivisten havaintojen jäädessä alle 10 prosentin. (Zhong ym. 2014, s. 283-284.)

Zhongin ym. menetelmällä siis voitaisiin heidän tutkimuksensa mukaan parantaa autentikaation osuvuutta. Mentelmä ei silti vielä selvästikään riitä tarkkuutensa vuoksi yksinään autentikoimaan käyttäjiä. Yhdessä jonkin toisen järjestelmän kanssa tällainen käyttäjien käyttäy-

tymisen tarkkailu voisi lisätä järjestelmän turvallisuutta. Toisaalta tutkimuksesta ei käy ilmi toteutuksen yksityiskohtia tai sitä, missä tietoja käyttäjän selauskäyttäytymisestä säilytetään. Käyttäjän kaiken selaushistorian tallentaminen jonnekin muualle kuin käyttäjän omalle tietokoneelle antaisi selvästi syytä huoleen tietoturvasta ja yksityisyydestä.

2.6 Menetelmien vertailu

Bonneau, Herley, Oorschot ja Stjano esittävät viitekehyksen eri kirjautumismenetelmien arvioimiseksi. Viitekehys sisältää 25 kirjautumismenetelmien hyödyllistä ominaisuutta, jotka jaetaan käytettävyyshyödyiksi, toteutettavuushyödyiksi ja turvallisuushyödyiksi. Bonneau ym. kutsuvatkin viitekehystään käytettävyys-toteutettavuus-turvallisuus -arviointiviitekehykseksi. Heidän mukaansa alan tutkijoiden keskuudessa on jo aiemmin oltu tietoisia kirjautumismenetelmien käytettävyyden ja turvallisuuden välisestä ristiriidasta. Ristiriita tarkoittaa sitä, että yleensä on helppoa lisätä toista vähentämällä samalla toista. Bonneaun ym. arviointiviitekehys yrittää nähdä tämän ristiriidan moniulotteisempana sekä tuo mukaan toteutettavuuden käsitteen. Toteutettavuushyödyt ovat siis sellaisia, jotka tekevät kirjautumismenetelmästä helpommin toteutettavan. (Bonneau ym. 2012, s. 553-554.)

Tutkimuksessaan Bonneau ym. arvioivat 35 eri kirjautumisen menetelmää, jotka ovat ehdokkaita perinteisten salasanojen korvaajiksi. Arvioinnissa he käyttivät omaa käytettävyys-toteutettavuus-turvallisuus -viitekehystään. Kirjautumismenetelmien arviointeja verrattiin perinteisiin salasanoihin. (Bonneau ym. 2012, s. 553 ja 563.)

Bonneau ym. tulevat siihen tulokseen, että mitään kirjautumismenetelmää ei voida pitää heidän arviointinsa perusteella suoraan salasanoja parempana. Suurin osa menetelmistä on salasanoja parempi jossain arvioinnin kategorioissa, mutta kaikki ovat myös huonompia joissakin. Tilanne on siis sellainen, että kirjautumismenetelmää valitessa on tasapainoiltava erilaisten käytettävyyden, toteutettavuuden ja turvallisuuden välillä tehtävien kompromissien kanssa. (Bonneau ym. 2012, s. 562.). Erityisesti toteutettavuus on salasanojen valtti, mikä toisaalta johtuu osin arvioinnin osa-alueiden perustumisesta salasanojen ominaisuuksiin sekä salasanojen vallitsevasta asemasta kirjautumismenetelmänä (Bonneau ym. 2012, s. 566).

Graafiset salasanat näyttävät olevan käytettävyydeltään lähellä perinteisten salasanojen tasoa

ja tuovat joitakin turvallisuushyötyjä. Kognitiiviset menetelmät parantavat myös hieman turvallisuutta, mutta heikentävät käytettävyyttä. Paperitokeneihin, laitetokeneihin ja puhelimiin pohjautuvat kirjautumismenetelmät pärjäävät todella hyvin turvallisuudessa, mutta käytettävyydessä huonosti. Biometriset kirjautumismenetelmät parantavat joitakin käytettävyyden osa-alueita, mutta huonontavat yhtä lailla toisia. Lisäksi ne pärjäävät huonosti sekä toteutettavuudessa että turvallisuudessa. (Bonneau ym. 2012, s. 564.)

Bonneau ym. arvioivat tutkimuksessaan myös federoituja SSO-menetelmiä. Niillä he tarkoittavat menetelmiä, joissa verkkosivu ohjaa käyttäjän luotetulle identiteettipalvelimelle, joka todentaa käyttäjän identiteetin (Bonneau ym. 2012, s. 559). Federoitut single sign-on -menetelmät saavat tutkimuksessa hyvät arviot. Ne parantavat sekä käytettävyys- että turvallisuusominaisuuksia verrattuna perinteisiin salasanoihin, joskin toteutettavuudessa salasanat vievät voiton. (Bonneau ym. 2012, s. 563.) Tutkijoiden mukaan näiden menetelmien arvioiminen on kuitenkin hankalaa, sillä SSO-järjestelmissä voitaisiin käyttää salasanojen sijasta muita kirjautumisen menetelmiä, jotka saavat parempia arvioita turvallisuudesta. Silloin tosin käytettävyyden heikkouksia periytyisi SSO-järjestelmään, sillä yleisestiottaen paremmat turvallisuusarviot saavat kirjautumismenetelmät saavat huonommat käytettävyysarviot. Salasanojen kanssa toimiva federoitu SSO taas ei paranna kirjautumisen turvallisuutta kovin paljoa. (Bonneau ym. 2012, s. 564.)

Bonneau ym. joka tapauksessa toteavat, että parhaiten sekä käytettävyyttä että turvallisuutta parantavat kirjautumisen menetelmät näyttävät olevan SSO:n ilmentymiä. SSO:n ilmentymiksi he laskevat myös esimerkiksi salasanojen hallintaohjelmat sekä kannettavaan tokeniin perustuvan Picon. (Bonneau ym. 2012, s. 565-566.) Bonneaun ym. tutkimuksen perusteella näyttää siis siltä, että SSO-menetelmät ovat lupaavimpia kirjautumismenetelmiä perinteisen salasanakirjautumisen korvaajiksi. Valmiiseen salasanoilla toteutettuun single sign-on järjestelmään liittyminen poistaisi SSO-järjestelmän toteuttamisen kustannukset ja mahdollistaa monia luvussa 2.2 mainituista salasanojen hyvistä puolista, joskin niiden toteutuminen riippuu SSO-palvelun tarjoajasta.

3 Single sign-on

Single sign-on on autentikaation tapa, jossa käyttäjä pääsee yksillä tunnuksilla kirjautumaan useisiin eri palveluihin (Chan 2006; Ardagna ym. 2006, s. 209-210) . Näin vähennetään sitä aikaa ja vaivaa, joka koituu uudelleen ja uudelleen kirjautumisesta eri verkkopalveluihin sekä useiden eri tunnusten ja salasanojen muistamisesta. Lisäksi SSO:lla vähennetään palveluntarjoajien ylläpidon määrää, kun käyttäjien kirjautumisen hoitaa kolmas osapuoli. (Chan 2006; Ardagna ym. 2006, s. 209-210.)

SSO:n tavoitteena ei ole pelkästään yhdet tunnukset useaan palveluun, vaan että kirjautumisprosessi tarvitsisi käydä läpi vain kerran, jonka jälkeen käyttäjä pääsee käsiksi useisiin järjestelmiin (You ja Zhu 2012, s. 383; Chan 2006; Chalandar, Darvish ja Rahmani 2007, s. 163) .

3.1 Pseudo-SSO ja aito SSO

Single sign-on järjestelmiä luokitellaan kirjallisuudessa eri tavoilla ja niiden rakennetta kuvataan eri tavoilla. Pashalidis ja Mitchell tekevät jaon pseudo-SSO -tyyppiin ja aitoon SSO-tyyppiin. Kumpikin näistä tyypeistä voi toimia käyttäjällä paikallisesti tai jonkin välittäjäpalvelimen kautta. Pseudo-SSO -tyypissä käyttäjän ja palveluiden välissä on välittäjä, joka hoitaa kirjautumisen jokaiselle palvelulle erikseen, mitä varten sillä on tiedossaan käyttäjän tunnisteet jokaiseen eri palveluun. Aidossa SSO:ssa käyttäjä taas autentikoituu jollekin autentikaatiopalveluntarjoajalle eikä ikinä suoraan millekään yksittäiselle palveluntarjoajalle. Autentikaatiopalveluntarjoaja sitten tarvittaessa välittää käyttäjän identiteetin ja kirjautumisen tilan palveluntarjoajille. Pashalidiksen ja Mitchellin mukaan aidossa SSO:ssa käyttäjällä voisi olla järjestelmän toteutuksesta riippuen mahdollista käyttää useita identiteettejä useissa palveluissa, mikä mahdollistaisi identiteettien käyttäminen eri rooleissa. Pseudo-SSO:ssa tämä ei ole mahdollista, koska käyttäjän identiteetit ovat palvelukohtaisia. (Pashalidis ja Mitchell 2003, s. 250-251)

Paikallisesta pseudo-SSO:sta Pashalidis ja Mitchell tuovat esiin sen, että käyttäjän on luotettava välittäjäkomponentille salaamattomat tunnisteensa. Välittäjäpalvelimeen perustuvas-

ta pseudo-SSO:sta he taas tuovat esiin, että käyttäjän tunnisteet eivät koskaan paljastu sille laitteelle, jolta kirjautuminen tehdään. (Pashalidis ja Mitchell 2003, s. 252-253)

Pashalidis ja Mitchell esittävät, että aito SSO olisi mahdollista toteuttaa myös siten, että autentikaatiopalveluntarjoaja toimisi käyttäjällä paikallisesti. Tällöin täytyisi olla olemassa mekanismi todentaa autentikaatiopalveluntarjoajakomponentin luotettavuus palveluntarjoajille. SSO toimisi käyttäjällä kuitenkin vain sillä laitteella, johon autentikaatiopalvelukomponentti olisi toteutettu. (Pashalidis ja Mitchell 2003, s. 253 ja 261) Välittäjäpalvelimeen perustuvassa aidossa SSO:ssa vaaditaan luottamussuhde palveluntarjoajien ja autentikaatiopalveluntarjoajan välille. Pashalidis ja Mitchell toteavat, että kaikissa heidän esittämissään SSO-malleissa autentikaation toteuttava järjestelmän osa voi esiintyä valheellisesti käyttäjänä, joten luottamusta kyseiseen osaan vaaditaan sekä käyttäjiltä että palveluntarjoajilta. (Pashalidis ja Mitchell 2003, s. 253) Toisaalta he esittävät, että pseudo-SSO -järjestelmiä voidaan toteuttaa myös kokonaan ilman palveluntarjoajien tietämystä (Pashalidis ja Mitchell 2003, s. 256).

Aidon SSO:n etuna pseudo-SSO:hon verrattuna on, että se periaatteessa mahdollistaa käyttäjän yksityisyyden sekä hänen eri identiteettiensä yhdistämättömyyden. Aidossa SSO-järjestelmässä palveluntarjoajille voidaan siis välittää pseudonyymejä, jotka eivät sisällä käyttäjästä mitään identifioivia henkilötietoja. Pseudo-SSO -järjestelmissä tätä ei voida taata, koska tunnisteet ovat palvelukohtaisia. Pashalidis ja Mitchell kuitenkin huomauttavat myös, että käyttäjän tunnukset voidaan yhdistää toisiinsa silloin kun käyttäjän verkko-osoite kulkee ei-anonymisoituna viestinnän mukana. Anonymisointi voitaisiin toteuttaa välittämällä kaikki käyttäjän ja palveluntarjoajien välinen viestintä jonkin ulkopuolisen välityspalvelimen kautta. Pashalidis ja Mitchell esittävät, että käyttäjän SSO:n toteuttava välityspalvelin voisi toimia myös käyttäjän liikenteen välityspalvelimena, jolloin käyttäjän ei tarvitsisi luottaa enää mihinkään muuhun osapuoleen. (Pashalidis ja Mitchell 2003, s. 253-254)

Välitäjäpalvelimeen perustuvat SSO-ratkaisut tukevat Pashalidiksen ja Mitchellin mukaan lähtökohtaisesti käyttäjän liikkuvuutta. Käyttäjä voi kirjautua mistä osasta verkkoa vain, ellei sitä ole muuten estetty. Paikalliset SSO-rakaisut voisivat toimia näin mikäli tunnisteet ladataan niihin käytön alussa joltain palvelimelta. Pashalidiksen ja Mitchellin esittämänsä paikallinen aito SSO taas ei tue käyttäjän liikkuvuutta. Välityspalvelinpohjaiset ratkaisut voivat

myös auttaa käyttöä epäluotettavissa ympäristöissä, kuten yleisessä käytössä olevilla alustoilla. Näin silloin kun käyttäjän pääasiallisia tunnisteita ei milloinkaan välitetä kirjautumisalustalle ainakaan siten, että niiden avulla voisi tulevaisuudessa esiintyä käyttäjänä. Välityspalvelinpohjaiset ratkaisut voisivat myös tarjota riitatilanteissa luotettavana kolmantena osapuolena helpommin luotettavia tapahtumatietoja kirjautumisista. Paikalliset SSO-ratkaisut ovat Pashalidiksen ja Mitchellin mukaan kuitenkin todennäköisesti halvempia toteuttaa ja ylläpitää. (Pashalidis ja Mitchell 2003, s. 255-257)

Pashalidiksen ja Mitchellin mukaan pseudo-SSO -toteutuksia on halvempi toteuttaa, koska ne eivät vaadi yhteistä turvallisuusinfrastruktuuria, eikä olemassaolevia palveluntarjoajia tarvitse muuttaa välttämättä ollenkaan. Muutokset palveluntarjoajien autentikaatiojärjestelmiin kuitenkin vaativat muutoksia SSO-toteutukselta. Aidoissa SSO-toteutuksissa toteutuskustannukset taas ovat korkeita, koska joudutaan toteuttamaan laaja järjestelmä ja mahdollisesti tekemään sopimuksia järjestelmän osapuolten välillä. Järjestelmän aidon SSO-toteutuksen valmistuttua muutoksia siihen ei kuitenkaan tarvitse tehdä niin usein kuin pseudototeutuksiin. Pashalidis ja Mitchell tuovat esiin myös sen, että internetin kaltaisissa avoimissa ympäristöissä aidon SSO:n tarjoamista yksityisyyseduista on suuri hyöty. (Pashalidis ja Mitchell 2003, s. 255-257)

3.2 SSO-järjestelmien vaatimukset

Ardagna, Damiani, Vimercati, Frati ja Samarati esittelevät luottamusmallin käsitteen. Luottamusmallit kuvaavat järjestelmiä niiden ympäristöjen, toiminnallisuuksien, komponenttien ja sääntöjen kautta sekä järjestelmien toimijoiden ja niiden keskinäisen vuorovaikutuksen sääntöjen kautta. Malleissa kuvataan myös järjestelmien erikoispiirteitä. SSO-järjestelmät Ardagna ym. jakavat kolmeen malliin, jotka ovat

1. Autentikaatio- ja autorisaatiomalli (Authentication and Authorization Model, AAM),
2. Federoitu malli (Federated Model, FM) ja
3. Täyden identiteetinhallinnan malli (Full Identity Management Model, FIMM). (Ardagna ym. 2006, s. 210-211.)

AAM edustaa tavanomaista kirjautumisen mekanismia, jonka osapuolet ovat käyttäjät ja pal-

velut. Käyttäjät pyytävät resursseja palveluilta tunnistetietojensa avulla ja palvelut tekevät päätökset siitä, pääsevätkö käyttäjät käsiksi resursseihin. Malli perustuu asiakas-palvelin -arkkitehtuuriin. Myös FM:n osapuolet ovat resursseja pyytävät käyttäjät ja niitä tarjoavat palvelut. Erotuksena AAM-malliin on se, että FM:ssä palvelut ovat jakautuneet eri domaineille, mutta ne on rakennettu samalle tasolle siten, että niiden keskinäinen luottamus ja esimerkiksi palvelujen välinen autentikoituminen mahdollistuu. FIMM:n osapuolet ovat resursseja pyytävien käyttäjien ja niitä tarjoavien palvelujen lisäksi käyttäjien identiteettien hallitsija. Ardagnan ym. mukaan suurin ero tämän ja aikaisemman kahden mallin välillä on se, että FIMM yrittää tarjota ratkaisuja yksityisyyteen liittyviin tarpeisiin. (Ardagna ym. 2006, s. 210-211.)

Ardagna ym. esittelevät myös listan SSO-järjestelmiltä vaadituista ominaisuuksista. Vaaditut ominaisuudet ovat erilaisia eri luottamusmalleille, joskin kaikki mallit jakavat suurimman osan vaatimuksista. Heidän mukaansa SSO-järjestelmien vaatimuksia ovat

1. **Autentikaatio**. Vaatimus liittyy kaikkiin malleihin.
2. **Vahva autentikaatio**. Turvallisuuskriittisissä järjestelmissä tarvitaan perinteisen käyttäjätunnuksen ja salasanan lisäksi muitakin autentikaation menetelmä. Vaatimus liittyy kaikkiin malleihin.
3. **Autorisaatio**. SSO-järjestelmä voi tarjota autentikoidun käyttäjän resursseihin pääsemisen hallintaa. Vaatimus liittyy AAM ja FIMM -malleihin.
4. **Esiehtojen välittäminen (provisioning)**. Esiehtojen täytyy täyttyä ennen kuin voidaan tehdä autentikaatiopäätös. Esiehtojen täyttymättömyys vaatii käyttäjän toimenpiteitä niiden täyttämiseksi. Vaatimus liittyy AAM ja FIMM -malleihin.
5. **Federointi**. Tämä tarkoittaa käyttäjän mahdollisuutta määritellä mihin palveluihin luottaa ja siten mille palveluille jakaa autorisaatiotodenteitaan (authorization assertions). Tämä vaatimus liittyy FM ja FIMM -malleihin.
6. **Keskitetty identiteettien hallinta**. Käyttäjien identiteettien hallinta keskitetysti vähentäen palveluntarjoajilta hallintaan liittyviä kustannuksia. Vaatimus liittyy AAM- ja FIMM -malleihin.
7. **Asiakkaan statustiedot**. SSO-järjestelmät vaativat statustietojen vaihtamista SSO-palvelimen ja palvelujen välillä autentikaatio- ja autorisaatiopäätösten tekemiseksi.

Vaatimus koskee kaikkia malleja.

8. **Yksi hallinnan piste**. Yksi SSO-järjestelmien päätavoitteita on saada keskitettyä autentikaation ja autorisaation palveluja. Ardagnan ym. mukaan tämä vaatimus koskee vain AAM-malleja, mutta tämän voi tulkita olevan ristiriidassa vaatimuksen 6 kanssa, koska myös FIMM-mallissa käyttäjien identiteettien hallinta on keskitettyä.
9. **Standardien mukaileminen**. SSO-järjestelmien on hyvä toimia niitä varten olemassa olevien standardien mukaisesti mahdollisimman laajojen integraatiomahdollisuuksien edistämiseksi. Vaatimus koskee kaikkia malleja.
10. **Useiden ohjelmointikielien tuki**. SSO-järjestelmien on hyvä olla integroitavissa eri kielillä ja teknologioilla kehitettyjen palvelujen kanssa, mitä edistää edellisessä kohdassa mainittu standardien noudattaminen. Vaatimus koskee kaikkia malleja.
11. **Salasanojen määrän kasvun estäminen**.
12. **Skaalautuvuus**. Tarkoittaa järjestelmän helppoa mukautumista kasvaviin käyttäjä- ja palvelumääriin. Vaatimus koskee kaikkia malleja. (Ardagna ym. 2006, s. 211-213.)

SSO-järjestelmän valintaa voisi siis lähestyä Ardagnan ym. esittämien luottamusmallien ja niihin liittyvien vaatimusten kautta. Federoitu malli edellyttää luottamusta toisiin federaatioon kuuluviin toimijoihin ja autentikaatio- ja autorisaatiomalli taas ei toimi eri domainien välillä. Ardagnan ym. esittämän luokittelun perusteella voisi sanoa, että yksittäisen palveluntarjoajan lienee useissa tapauksissa helpointa liittyä johonkin täyden identiteetinhallinnan mallin SSO-järjestelmään. Näin vältytään usean eri federaation tahon luotettavuuden arvioinnilta ja saadaan keskitetty identiteetinhallinta taholta, joka tarjoaa identiteetinhallintaa useille tahoille.

3.3 Arkkitehtuurit

Suorannan ym. mukaan välittäjäpalvelimiin perustuvia aitoja SSO-järjestelmiä kutsutaan usein federoiduiksi SSO-järjestelmiksi. Niissä siis autentikaatio hoidetaan erillisellä identiteetintarjoajapalvelimella. Esimerkkejä tällaisista järjestelmistä ovat OpenID ja Shibboleth. (Suoranta ym. 2013, s. 147-148.)

Chanin mukaan SSO-järjestelmissä on pääasiassa kolme osapuolta, jotka ovat käyttäjät, pal-

veluntarjoajat ja identiteetintarjoajat (identity provider). Käyttäjä haluaa päästä käsiksi joihinkin autentikaatiota vaativiin resursseihin, joita palveluntarjoajat tarjoavat. Identiteetintarjoajat tarjoavat autentikaatioon tarvittavia palveluja. (Chan 2006, s. 509)

Esimerkiksi Ying, Yao ja Hua luokittelevat SSO-järjestelmät kahteen eri luokkaan. Nämä ovat keskitetty (centralized) ja hajautettu (distributed) malli (Ying, Yao ja Hua 2009, s. 448). Feng-man ja Qin-yu luokittelevat SSO-mallit keskitettyihin ja avoimiin (Feng-man ja Qiu-yu 2010, s. 115). Tulkitsen, että heidän luokkansa ”avoin” vastaa Yingin ym. hajautettua, sillä molemmat viittaavat esimerkeissään Liberty Allianceen. Myös Chalandar ym. kirjoittavat keskitetystä ja hajautetusta SSO:sta, minkä lisäksi he määrittelevät myös federoidun SSO:n (Chalandar, Darvish ja Rahmani 2007, s. 163-164).

3.3.1 Keskitetty

Keskitetyissä järjestelmissä on olemassa yksi luotettu auktoriteetti, jonka kautta kaikki kirjautumiset tapahtuvat ja joka pitää kirjaa käyttäjien tiedoista (Ying, Yao ja Hua 2009, s. 448; Chalandar, Darvish ja Rahmani 2007, s. 163-164) . Chalandarin ym. mukaan keskitettyyn SSO-palveluun liittyminen vaatii yleensä muutoksia sovellukseen (Chalandar, Darvish ja Rahmani 2007, s. 164) Keskitettyjä järjestelmiä kritisoitiin siitä, että niiden toiminnan sujuvuus nojaa liikaa yhden identiteetin tarjoajan toimintakykyyn, joka voi olla suuren rasituksen alla riittämätön. Keskitettyyn järjestelmään murtauduttaessa myös vaarantuvat kerralla kaikkien siihen nojaavien järjestelmien tiedot. (Feng-man ja Qiu-yu 2010, s. 115.)

3.3.2 Hajautettu

Hajautettu järjestelmä on Yingin ym. mukaan sellainen, että yhdelle palveluntarjoajalle kirjautuva pääsee sisään muidenkin palveluntarjoajien järjestelmiin. Keskitettyä auktoriteettia ei siis ole ja palveluntarjoajat säilyttävät omat autentikaatiomekanisminsa (Ying, Yao ja Hua 2009, s. 448). Chalandar ym. tuovat esiin, että hajautetussa mallissa säilyy paikallinen identiteettien kerääminen ja esittäminen ja profiilit sekä salasanat synkronoidaan keskitetylle palvelimelle. Hajautettu malli mahdollistaa Chalandarin ym. mukaan sen, ettei SSO-järjestelmään liittyvään sovellukseen tarvitse tehdä muutoksia. (Chalandar, Darvish ja Rah-

mani 2007, s. 164)

3.3.3 Muut

Gu & Zhang (Gu ja Zhang 2013) sekä You & Zhu (You ja Zhu 2012) jakavat SSO-järjestelmät kolmeen eri luokkaan. Näitä ovat

1. välittäjäperusteinen (broker-based)
2. toimijaperusteinen (agent-based)
3. porttiperusteinen (gateway-based)

Välittäjäperusteisessa SSO:ssa autentikointi keskittyy yhdelle sitä varten tarkoitetulle palvelimelle. Tämä palvelin sitten palauttaa asiakkaalle varmenteen, jolla pääsee käsiksi palveluntarjoajien palveluihin. Toimijaperusteisessa SSO:ssa asiakas käyttää jotain toimijaohjelmaa kirjautumisten hallinnassa. Ohjelma toimii siis välittäjänä palvelimen autentikaatiojärjestelmän ja käyttäjän autentikaatiometodin välillä. Gun ja Zhangin mukaan porttiperusteisessa SSO:ssa jokin porttina toimiva laitteisto eristää sen toisella puolella olevat palvelut käyttäjistä. Kunhan portti autentikoi käyttäjän, hän pääsee käsiksi portin toisella puolella oleviin palveluihin. Huonoa tässä on se, että järjestelmän tehokkuutta rajaa tuon yhden eristävän portin toiminta. (Gu ja Zhang 2013.)

3.3.4 Palvelujen eriyttäminen SSO-toteutuksessa

Niu Ying, Zhao Yao ja Zhou Hua (Ying, Yao ja Hua 2009) kehittävät SSO-järjestelmään käyttäjälle mahdollisuuden itse valita eri palveluntarjoajien turvallisuusluokituksen. Heidän järjestelmässään on siis osa, joka tallentaa käyttäjän määritykset eri palvelujen turvatasosta. Näiden määritysten perusteella sitten määritetään esimerkiksi ne salaustavat, joilla palveluntarjoajien, identiteetintarjoajan ja käyttäjän välillä kulkeva viestintä salataan. Turvallisuustasot toimivat myös siten, että alemman tason palveluntarjoajan palveluun kirjautumisella ei pääse automaattisesti ylemmän turvallisuustason palveluun.

Lisäksi turvallisuustasojen avulla ratkaistaan SSO-järjestelmille tyypillinen single sign-outin ongelma, jota käsitellään enemmän luvussa 3.6. Single sign-out on päinvastainen operaatio

single sign-onille, eli yhdestä palvelusta uloskirjautumalla uloskirjautuu myös kaikista muista palveluista. Yingin ym. tapauksessa single sign-out toimii niin, että käyttäjän kirjautuessa ulos palvelusta, kirjautuu hän automaattisesti ulos myös saman turvatason sekä korkeamman turvatason palveluista. (Ying, Yao ja Hua 2009.) Yingin ym. menettelytapa määrittelymahdollisuuden antamisesta käyttäjälle itselleen on kiinnostava ratkaisu ongelmaan. Kuitenkin on melko helppoa nähdä, että vastuun antaminen käyttäjälle itselleen tietoturva-asioista on jonkinlainen riski itsessään varsinkin kokemattomien käyttäjien osalta.

3.4 Luottamuksen rakentuminen SSO:n tyypeissä

Pashadaliksen ja Mitchellin mukaan niin pseudo-SSO- kuin aidoissa SSO-malleissa sekä käyttäjien että palveluntarjoajien on voitava luottaa SSO:n toteuttavaan komponenttiin tai palvelimeen. Toisaalta he esittävät, että pseudo-SSO-malleissa palveluntarjoajat eivät välttämättä edes tiedä SSO-toteutuksen olemassaolosta. (Pashadalis ja Mitchell 2003, s. 256.) Tällaisessa tapauksessa ei selvästikään voida puhua minkäänlaisesta luottamussuhteesta palveluntarjoajan ja SSO:n toteuttavan komponentin välillä, vaan käytännössä luottamusta vaaditaan vain käyttäjältä. Paikallisessa SSO:ssa käyttäjän ei tarvitse luottaa ulkopuoliseen SSO-palveluun, mutta edelleen SSO:n toteuttavaan ohjelmistoon kuitenkin (Pashadalis ja Mitchell 2003, s. 256).

Pseudo-SSO:ssa luottamussuhteet ovat Pashadaliksen ja Mitchellin mukaan aitoa SSO:ta epäselvemmät ja riippuvat toteutusyksityiskohdista. Ne voivat olla erilaisia eri palveluntarjoajien ja SSO:n toteuttavan komponentin välillä. Lisäksi ne voivat muuttua mikäli jokin palveluntarjoaja muuttaa autentikaatiorajapintojaan. Aidoissa SSO-järjestelmissä luottamussuhteen palveluntarjoajien ja autentikaatiopalveluntarjoajan välillä voidaan määritellä tarkasti erilaisilla sopimuksilla varsinaisen SSO-järjestelmän teknisen toteutuksen ulkopuolella. Tarvittaessa käyttäjän tunnukset voidaan myös jäädyttää keskitetysti. (Pashadalis ja Mitchell 2003, s. 256.)

Luottamuksen rakentuminen on selvästi erilaista esitellyissä SSO-arkkitehtuurien malleissa. Keskitetyssä järjestelmässä tarvitsee miettiä ainoastaan yhden identiteetintarjoajan luotettavuutta, kun taas hajautetussa järjestelmässä on mietittävä lukuisten eri järjestelmän toimijoi-

den luotettavuutta. Toimijoiden lukuisuus myös heikentää käyttäjien mahdollisuuksia tunnistaa niitä palveluja, joilla on oikeus tunnisteita heiltä pyytää, jolloin pahantahtoisille tahoille aukeaa mahdollisuuksia teeskennellä luotettavia käyttäjien tunnusten varastamiseksi (Dong, Clark ja Jacob 2008, s. 54-57).

3.5 SSO-totetuksiin liittyvät teknologiat

3.5.1 Evästeillä toteutettu SSO

Evästeet ovat tekstimuotoista dataa, jonka palvelin lähettää verkkoselaimeen, joka tekee pyyntöjä palvelimelle. Selain tallettaa evästeet välimuistiinsa ja niiden avulla selain voidaan identifioida. Chalandarin ym. mukaan evästeiden avulla tapahtuvassa käyttäjänimeen ja salasanaan perustuvassa autentikaatiossa täytyy käyttää SSL:ää. (Chalandar, Darvish ja Rahmani 2007, s. 164) SSL on nykyään vanhentunut (Sheffer, Holz ja Saint-Andre 2015), joten pitäisi käyttää TLS:ää (Rescorla 2018) ja siitä uudempaa versiota kuin 1.1, sillä siinäkin on puutteita (Sheffer, Holz ja Saint-Andre 2015) ja sen muodollista vanhentamista suunnitellaan (Moriarty ja Farrell 2019). Evästeen tulisi myös sisältää ainoastaan session tunnus, joka voidaan palvelimella yhdistää session tietoihin. Näin vältetään session tietojen paljastumista. Huomioon tulee ottaa myös, että evästeet lähetetään vain ne asettaneen palvelimen domainille, eikä ole mahdollista asettaa niitä lähetettäviksi muille domaineille. (Chalandar, Darvish ja Rahmani 2007, s. 164)

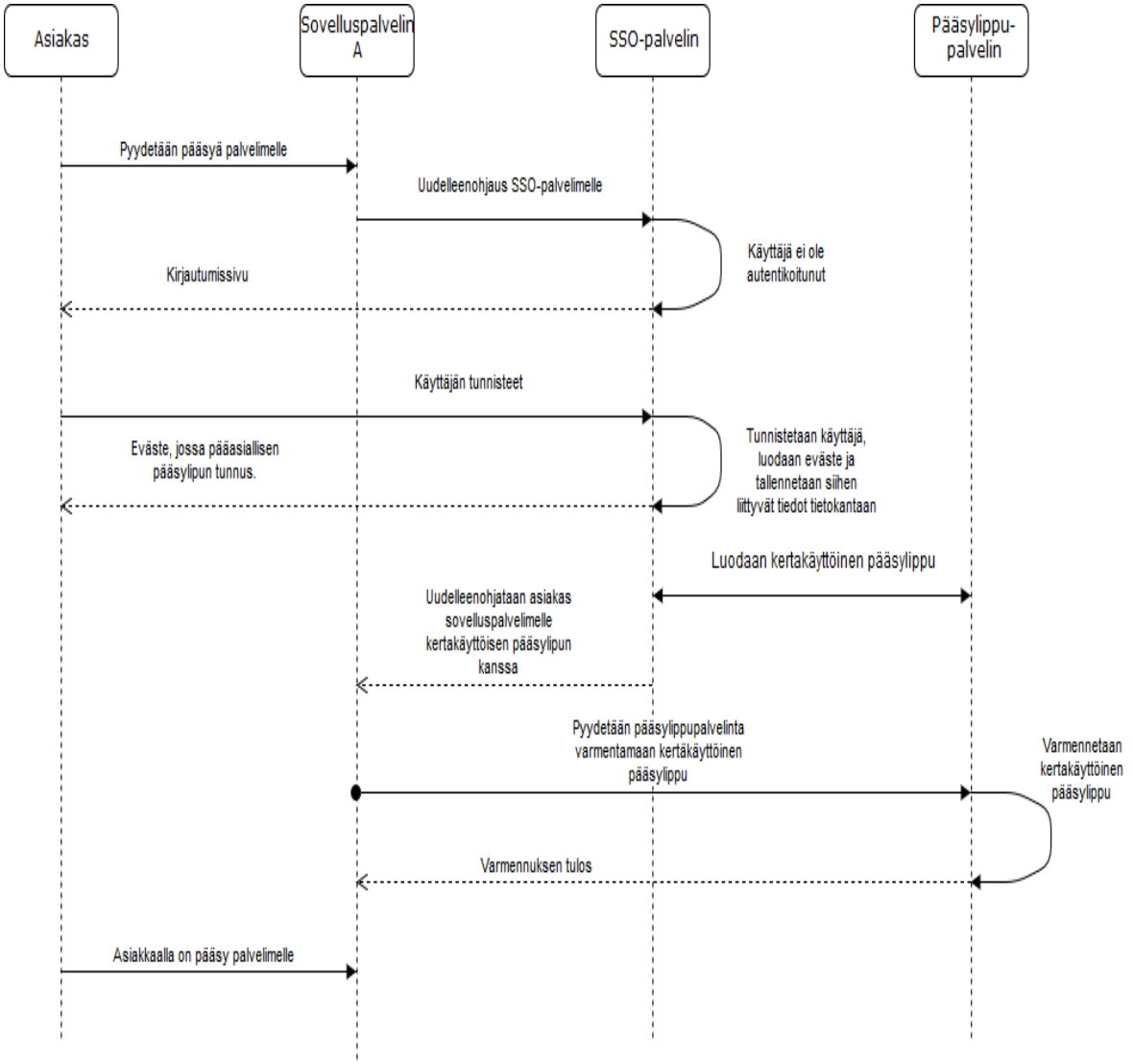
Chalandarin ym. mallissa käyttäjä ohjataan sovelluspalvelimelta A kirjautumaan SSO-palvelimen kirjautumissivulle. SSO-palvelin luo validia käyttäjää varten evästeen ja tallentaa tietokantaan tähän evästeeseen liittyviä tarvittavia tietoja, kuten evästeen nimen, sisällön ja vanhenemisajan sekä käyttäjän tai proksypalvelimen IP-osoitteen ja käyttäjän identifiointitiedot. SSO-palvelin välittää evästeen sovelluspalvelimelle, joka taas välittää sen käyttäjän selaimelle muun aiemmin pyydetyn datan ohessa. Jos käyttäjä tämän jälkeen tekee pyynnön toiselle samalla domainilla sijaitsevalle SSO-järjestelmään kuuluvalle sovelluspalvelimelle B, sujuu kirjautuminen käyttäjälle läpinäkyvästi. Tämä johtuu siitä, että käyttäjän selaimella ollut eväste lähetetään automaattisesti samalla domainilla sijaitseville palvelimille. Palvelin B lähettää tämän evästeen sitten SSO-palvelimelle autentikoitavaksi ja hyväksytyn autenti-

koinnin jälkeen lähettää käyttäjälle pyydetyn datan. (Chalandar, Darvish ja Rahmani 2007, s. 164.)

Chalandarin ym. järjestelmässä käyttäjän tunnistetietoja ei koskaan jaeta muiden palvelimien kuin SSO-palvelimen kanssa. Tätä voi pitää turvallisuushyötynä. Chalander ym. toteavat myös, että 20-merkkiä pitkät satunnaiset evästeen sisällöt tekevät evästeistä erittäin turvalliset. He tuovat myös esiin, että tilan säästämiseksi SSO-palvelimen tietokannasta poistetaan säännöllisesti vanhentuneet evästeet. (Chalandar, Darvish ja Rahmani 2007, s. 164-165.)

Chalanderin ym. mallissa huonoa on se, että läpinäkyvä kirjautuminen ei toimisi palvelimilla, jotka eivät sijaitsisi saman domainin alla kuin palvelin, jolle on jo kirjauduttu. Evästeet lähetetään vain niille palvelimille, joiden domain on evästeet lähettäneen palvelimen domain tai alidomain (Barth 2011, luku 4.1.2.3). Mallissa järjestelmään voisi siis liittää palvelimia muilta domaineilta, mutta käyttäjälle läpinäkyvä kirjautuminen toimisi ainoastaan tietyn domainin palvelimien välillä.

Youn ja Zhun mallissa SSO-järjestelmään kuuluvissa sovelluspalvelimissa on toimijaohjelma, joka uudelleenohjaa käyttäjän autentikoitumaan autentikaatiopalvelimelle. Evästeiden toiminnan rajoitus ratkaistaan siten, että autentikaatiopalvelin asettaa tunnistautumiseen käytettävän evästeen käyttäjän selaimeen. Käyttäjän tarvitsee siis syöttää tunnisteet kerran saadakseen evästeen, joka jatkossa toimitetaan autentikaatiopalvelimelle automaattisesti kun jokin sovelluspalvelin uudelleenohjaa käyttäjän autentikoitumaan autentikaatiopalvelimelle. (You ja Zhu 2012, s. 384-385.)



Kuvio 1. Asiakkaan kirjautuminen sovelluspalvelimelle (You ja Zhu 2012, s. 386).

Kun autentikoimaton käyttäjä pyytää sovelluspalvelimelta jotain resurssia, uudelleenohjaa sovelluspalvelimen toimijaohjelma siis käyttäjän SSO-palvelimelle, joka koostuu pääasiassa autentikaatiopalvelimesta ja pääsylippupalvelimesta. Uudelleenohjaus-URL:n parametrina on sovelluspalvelimen identifikaatiotieto. Koska käyttäjä ei ole vielä autentikoitunut, ei hänen selaimessaan ole vielä tallennettuna pääasiallista pääsylippua (primary ticket) evästeeseen. Tämän vuoksi autentikaatiopalvelin pyytää käyttäjältä tunnistetietoja. Käyttäjän annettua tunnistetiedot autentikaatiopalvelin selvittää onko käyttäjällä oikeutta saada pääsy sovelluspalvelimen tietoihin. Mikäli on, palvelin myös etsii käyttäjän id:n tuolla kyseisellä

sovelluspalvelimella ja generoi pääasiallisen pääsylipun. Pääasiallinen pääsylippu sisältää käyttäjän id:n SSO-järjestelmässä. Pääsylippu lähetetään evästeenä selaimelle ja tallennetaan autentikaatiopalvelimelle. (You ja Zhu 2012, s. 384-385.)

Pääsylippupalvelin luo pääasiallisen pääsylipun mukaan kertakäyttöisen palvelupääsylipun, joka sisältää sovelluspalvelimen ID:n, käyttäjän ID:n sovelluspalvelimella sekä käyttäjän pääasiallisen pääsylipun. Käyttäjä uudelleenohjataan sovelluspalvelimelle URL:n parametrina palvelulipun ID. Sovelluspalvelimen agenttiohjelma kysyy pääsylippupalvelimelta saamansa palvelupääsylipun ID:n avulla käyttäjän ID:n. Nyt käyttäjälle voidaan tarjota autentikaation vaativia resursseja. Tulevilla kerroilla käyttäjän ei tarvitse enää antaa tunnistetietojaan, sillä selaimeen tallennetun pääasiallisen pääsylipun sisältävän evästeen avulla prosessi sujuu automaattisesti. (You ja Zhu 2012, s. 384-385.)

3.5.2 PKI-perusteinen SSO

Bazazin ja Khaliquen mukaan Public Key infrastructure (PKI) -perusteisessa SSO:ssa järjestelmä nojaa sertifikaattiauktoriteettiin, joka jakaa ja hallinnoi käyttäjien digitaalisia sertifikaatteja ja siten heidän identiteettejään. Käyttäjä saa julkisen avaimen sertifikaatin sertifikaattiauktoriteetilta tunnistauduttuaan sille. Käyttäjä tunnistautuu palveluntarjoajalle sertifikaatin avulla siten, että luo palveluntarjoajalle lähetettäväksi tokenin sisällyttäen siihen julkisen avaimensa, ja sitten allekirjoittaa tämän tokenin salaisella avaimellaan. Saatuaan käyttäjältä autentikaatiopyynnön palveluntarjoaja varmistaa hänen identiteettinsä sertifikaattiauktoriteetilta. Bazaz ja Khalique mainitsevat esimerkeiksi PKI-pohjaisista SSO-ratkaisuista Globalsignin ja Verisignin (Bazaz ja Khalique 2016).

Butler, Welch, Engbert, Foster, Tuecke, Volmer ja Kesselman esittävät ratkaisun virtuaalisen organisaation käyttäjien autentikointiin, Grid Security Infrastructuren (GSI). Heidän mukaansa virtuaalisia organisaatioita muodostetaan erilaisten resurssien kuten data-arkistojen kokoamiseksi yhteen. Resurssien jakamista varten käyttäjien tulee kuitenkin pystyä autentikoitumaan eri resursseja tarjoaville tahoille. Koska virtuaaliset organisaatiot ovat luonteeltaan muuttuvaisia, tulisi autentikaatiomekanismien olla kevyitä uusien ressurssienjakojärjestelyjen nopean järjestämisen helpottamiseksi. Toisaalta jäsenorganisaatioiden tulisi saada

säilyttää omat paikalliset käytänteensä koskien omia resurssejaan. (Butler ym. 2000, s. 60)

Butler ym. toteavat, että virtuaalisen organisaation käyttäjien ensisijainen vaatimus järjestelmälle on yksinkertaisuus. Tätä tukee single sign-on. Resursseja tarjoavien sivujen vaatimukset taas ovat, että niiden ei tarvitse muuttaa omia turvallisuusjärjestelyjään ja että organisaation järjestelyt ovat vähintään yhtä vahvat kuin paikalliset järjestelyt. Sivujen hallinnoijilla tulisi pysyä tiukka kontrolli heidän resursseihinsa pääsystä ja siitä, miten käyttäjän identiteetti todennetaan. (Butler ym. 2000, s. 61)

Identiteettien todenteet Butlerin ym. järjestelmässä perustuvat julkisen avaimen infrastruktuuriin (Public Key Infrastricture, PKI). Kaikilla organisaatioon kuuluvilla käyttäjillä, ohjelmilla ja resursseilla on oma SSL/TLS -protokollaan perustuva salainen avain, jonka luotettu sertifikaattiauktoriteetti CA yhdistää julkiseen pariinsa. SSL/TLS -protokollaan perustuvalla autentikaatioalgoritmilla tarkistetaan todentautuvan osapuolen identiteetti. Järjestelmän osapuolet voivat jakaa oikeuksiaan luomalla väliaikaisia proksyiksi kutsuttuja identiteettejä ja nämä voivat luoda taas uusia proksyja. Tämä tapahtuu allekirjoittamalla proksysertifikaatteja, joiden muodostamia ketjuja seuraamalla voidaan prosessien aloittaja tunnistaa.

Resurssien hallinnoijat päättävät itse keiden pyyntöjä he hyväksyvät. Tämä onnistuu kartoittamalla sertifikaattien perusteella tunnistettavat globaalit identiteetit paikallisiin identiteetteihin, mikä tapahtuu yksinkertaisimmillaan tekstipohjaisella karttatiedostolla (map file). Luottamus järjestelmän jäsenen globaaliin identiteettiin perustuu siihen luottamukseen, joka sertifikaatteja antavaan sertifikaattiauktoriteettiin asetetaan. Siksi paikalliset resurssien hallinnoijat päättävät itse, mitä auktoriteetin sertifikaatteja he autentikoimiseen käyttävät. (Butler ym. 2000, s. 61-62.)

GSI:ssä resurssien tarjoajien luottamus järjestelmään perustuu siihen, että se on yksinkertainen ja sen käyttämät teknologiat ovat tunnettuja standardeja (Butler ym. 2000, s. 63). Lisäksi luottamus järjestelmän osapuolten identiteetteihin perustuu sertifikaattiauktoriteetteihin asetettuun luottamukseen. Käyttäjälle single sign-on syntyy hänen sertifikattiiauktoriteetilta saamansa sertifikaatin perusteella tunnistettavan globaalin identiteettinsä kautta. Resurssien tarjoajille jää paljon mahdollisuuksia itse päättää miten he paikallisesti näihin identiteettehin suhtautuvat. Toisaalta resurssien tarjoajat voivat rakentaa juuri niin luotettavat autentikaatio-

ja autorisaatiokäytännöt kuin haluavat. Toisaalta tämä jättää paljon töitä resurssien tarjoajille ja single sign-onilta usein odotetut järjestelmien ylläpitäjien vähentyneet ylläpitokustannukset eivät varmaankaan toteudu merkittävissä määrin. Lisäksi eri resurssientarjoajien luottaessa eri sertifikaattiauktoriteetteihin tarvitsevat virtuaalisen organisaation käyttäjät joskus useita sertifikaatteja (Butler ym. 2000, s. 63). Tällainen useiden identiteetin todenteiden omistaminen on SSO:n perusideaa vastaan ja lisää järjestelmään juuri sitä monimutkaisuutta, mitä SSO:lla halutaan poistaa.

Charlambous, Karapetris ja Athanasopoulos esittävät PKI:hin perustuvien autentikaatiojärjestelmien ongelmiksi tunnusten peruuttamisen hankaluutta ja avaintenhallintaa. Salasanan muuttamiseen verrattuna on hankalaa etsiä kaikki ne palvelut, joissa peruuttamisen tarpeessa olevaa kryptografista avainta käytetään ja sitten peruuttaa kyseinen avain. Avaimia tarvitaan kirjauduttaessa ja niitä pitääkin siirrellä laitteiden välillä, jos halutaan kirjautua palveluihin eri laitteilla. (Charalambous., Karapetris. ja Athanasopoulos. 2018.)

Zhaon ym. mukaan tunnusten peruuttamiseen yleinen keino on julkaista listoja peruutetuista avaimista. Ongelmia ovat ainakin, ettei listoja välttämättä päivitetä reaaliaikaisesti ja että niiden jakaminen on altis palvelunestohyökkäyksille. Zhao ym. esittelevät muitakin tapoja tunnusten peruuttamisen hoitamiseen. Avaimia on säilytetty esimerkiksi verkon yli saavutettavassa MyProxy-palvelussa, joista ne voidaan saada käytettäväksi salasanalla kirjautumalla. Tällaisen ratkaisun käyttäminen toisaalta altistaa järjestelmän salasanojen heikkouksille ja murtamisille. (Zhao, Aggarwal ja Kent 2007.)

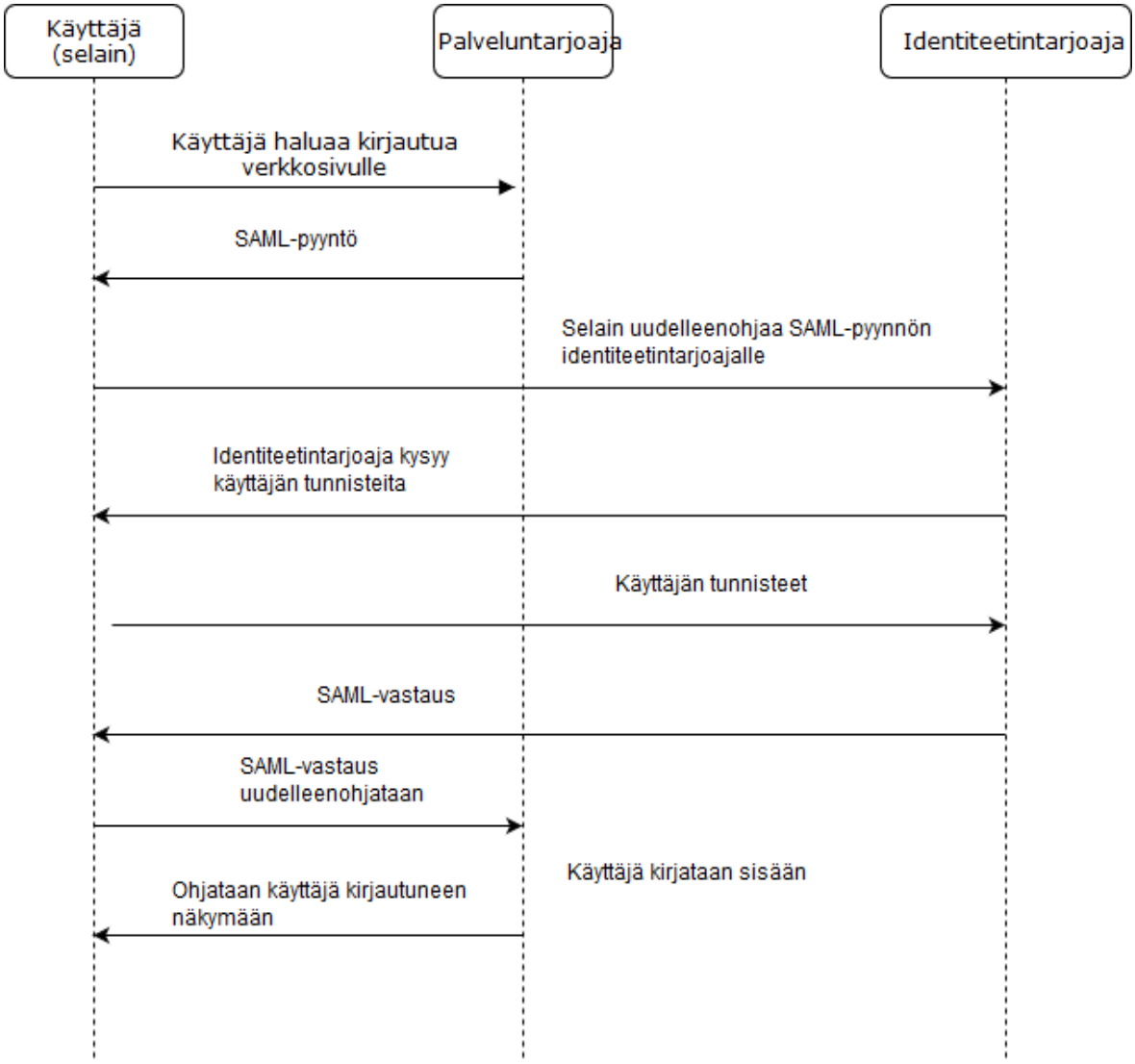
Charlambous ym. ovat kehittäneet KAuth-nimisen SSO-järjestelmän vastaamaan esittämiinsä PKI-kirjautumisen ongelmiin. Tässä järjestelmässä käyttäjällä on vain yksi salainen kryptografinen avain, jonka vain KAuth-järjestelmä tietää, joten sen peruuttaminen on helppoa. Järjestelmässä on kaksi erilaista tapaa kirjautua: KAuth ja KAuth+. KAuth ei vaadi PKI-avaimen hallintaa käyttäjältä, mutta KAuth+ vaatii tarjoten kuitenkin samalla vahvempaa tunnistautumista. (Charalambous., Karapetris. ja Athanasopoulos. 2018.)

Charlambous ym. esittävät KAuthin olevan käyttäjän näkökulmasta täysin samanlainen kuin Facebook Connect. Kun käyttäjä haluaa kirjautua KAuthia käyttävälle sivulle, hänet uudelleenohjataan KAuthin OAuth-palvelimelle, joka taas ohjaa hänet KAuthin Keybase-palvelimelle,

jolle käyttäjä kirjautuu. Onnistuneen kirjautumisen jälkeen asiakassivu saa autorisaatiokoodin, jonka se vaihtaa access tokeniin, jonka avulla käyttäjän tietoja voidaan pyytää. Ero normaaliin salasanakirjautumiseen tulee siitä, miten itse kirjautuminen KAuthin Keybase-palvelimelle hoidetaan. KAuthissa käyttäjä syöttää järjestelmään käyttäjänimensä tai sähköpostiosoitteensa ja salalauseensa normaalin salasanakirjautumisen tapaan. Näiden avulla selaimessa kuitenkin luodaan kryptografinen avain, jolla allekirjoitetaan tietty kirjautumisessa vaadittava viesti. Käyttäjän salalausetta ei lähetetä missään vaiheessa selaimesta mihinkään. KAuth+-kirjautumisessa on lisävaihe, jossa käyttäjä allekirjoittaa järjestelmältä saamansa viestin omalla kryptografisella avaimellaan. (Charalambous., Karapetris. ja Athanasopoulos. 2018.)

3.5.3 SAML

Biennierin mukaan SAML on OASIS:in XML-pohjainen merkintäkieli, jonka avulla voidaan välittää autentikaatioon ja autorisaatioon liittyviä tietoja SSO:n osapuolten (käyttäjä, identiteetintarjoaja ja palveluntarjoaja) välillä. SAML:in tarkoituksena on mahdollistaa näiden osapuolten välinen löyhä yhteys (loose coupling), mikä on internetin yli tapahtuvassa SSO:ssa tarpeellista. (Biennier 2011, s. 1377-1382.)



Kuvio 2. Tyypillinen SAML-käyttötapaus (Naik ja Jenkins 2017; Biennier 2011) .

SAML:issa on kolme osapuolta (Naik ja Jenkins 2017):

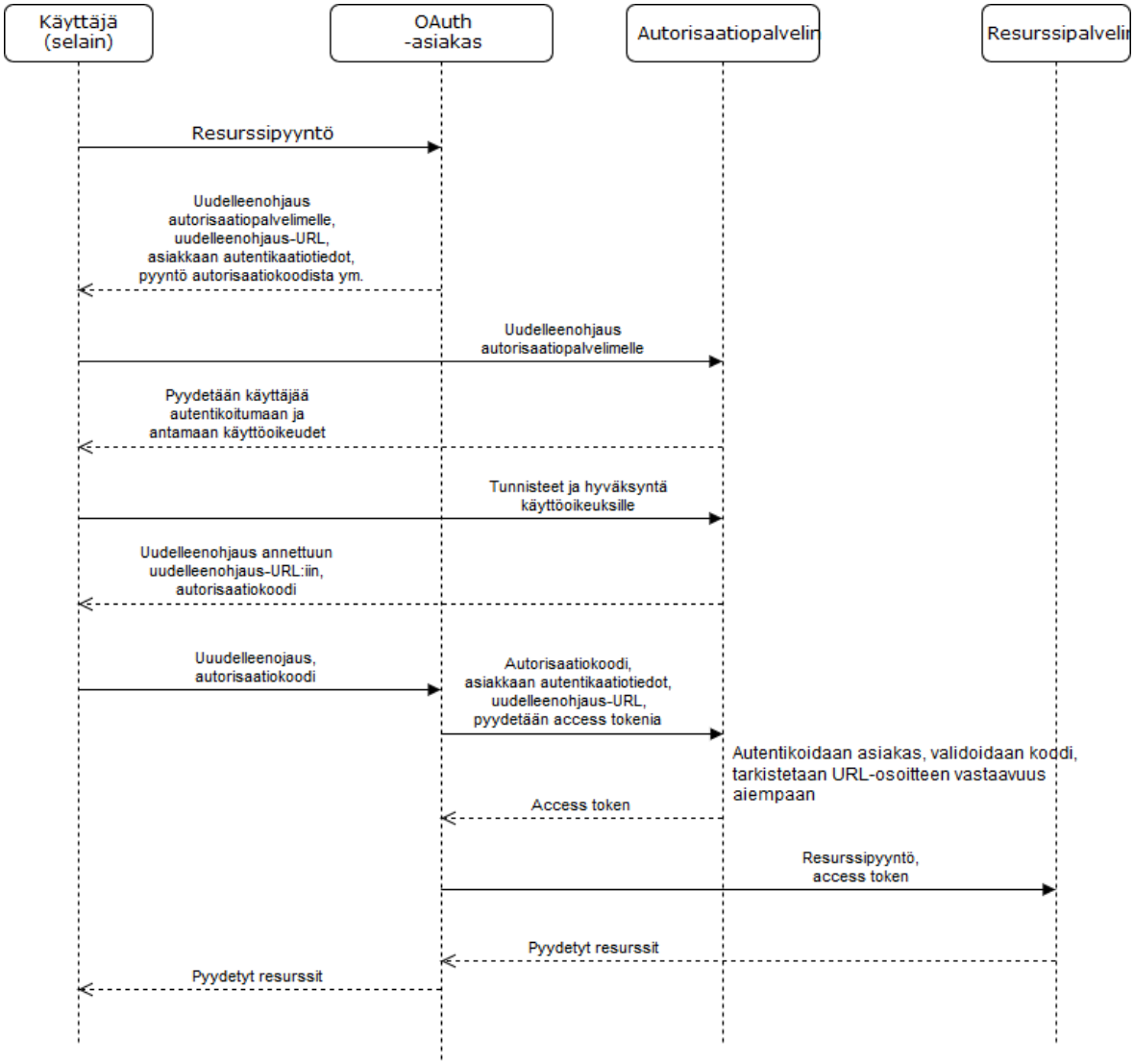
1. **Identiteetintarjoaja**, joka vastaa käyttäjien autentikoinnista ja autorisaatiosta sekä autentikointipäätösten välittämisestä palveluntarjoajille.
2. **Palveluntarjoaja**, joka vastaanottaa autentikaatio- ja autorisaatiopäätöksiä identiteetintarjoajalta.
3. **Käyttäjä**, joka haluaa päästä käsiksi palveluntarjoajan resursseihin ja joutuu tätä varten autentikoitumaan identiteetintarjoajalla.

SAML:iin kuuluvat seuraavat neljä elementtiä (Naik ja Jenkins 2017; Biennier 2011s. 1379-1380) :

1. **Assertioilla** välitetään tietoa käyttäjän identiteetistä, käyttöoikeuksista ja ominaisuuksista.
2. **Protokollat** määrittävät millä XML-viestien sarjoilla saavutetaan mitäkin tavoitteita, kuten pyydetään assertioita identiteetintarjoajalta.
3. **Sidokset (Bindings)** kuvaavat miten viestejä lähetetään alemman tason protokollien avulla.
4. **Profiilit** ovat sidosten kokoelmia, joilla määritellään rajoituksia ja laajennoksia SAML:in käyttöön tietyissä tarkoituksissa.

3.5.4 OAuth

OAuth on delegaatioprotokolla, jonka avulla käyttäjä voi antaa jollekin kolmannelle osapuolelle (jokin sovellus jollain palvelimella) käyttöoikeuksia johonkin toiseen sovellukseen, jossa käyttäjällä on omat käyttöoikeudet. Käyttöikeudet sovellukseen välitetään kolmannelle osapuolelle access tokenin avulla, jolla ei kuitenkaan voida antaa käyttöoikeuksia enempää kuin sovellus itse sallii. (Naik ja Jenkins 2017.)



Kuvio 3. Käyttöoikeuksien pyytäminen OAuth 2.0:n Autorisaatiokoodi-kulussa (RFC6749).

OAuthissa on seuraavat neljä osapuolta:

1. **Resurssipalvelin** sisältää dataa, joka on suojattu OAuthilla.
2. **Resurssin omistaja/Käyttäjä** on sovelluksen käyttäjä ja datan omistaja.
3. **OAuth-asiakas** on sovellus, joka haluaa käyttöoikeuksia käyttäjän dataan.
4. **Autorisaatiopalvelin** jakaa käyttöoikeuksia käyttäjän resurssipalvelimella olevaan dataan niitä pyytäville OAuth-asiakkaille käyttäjän hyväksynnän perusteella. (Naik ja Jenkins 2017.)

Access tokenin voi OAuthissa saada useammalla eri tavalla (Hardt 2012). Oheisessa kuvassa on esitetty, miten se saadaan autorisaatiokoodia käyttämällä. Tässä tavassa on muihin tapoihin verrattuna hyötyinä esimerkiksi mahdollisuus autentikoida OAuth-asiakas sekä se, ettei access tokenia tarvitse välittää käyttäjän käyttäjäagentin (user agent, esim. selain) kautta ja siten vaarantaa sen paljastumista muille tai käyttäjälle itselleen (Hardt 2012).

3.5.5 OpenID Connect

OpenID Connect -protokolla on rakennettu OAuth 2.0 -protokollan päälle (Nat Sakimura ym. 2014). Se tarjoaa asiakkaille mahdollisuuden identifioida käyttäjiä jonkin autorisaatiopalvelimen tekemän autentikaation perusteella ja kysellä näiden profiilitietoja. OpenID Connectissa käytetään access tokeneita ja ID tokeneita. ID token on JSON Web Token, joka on identiteetintarjoajan allekirjoittama ja jonka oikeellisuus voidaan varmistaa ilman yhteyttä identiteetintarjoajaan. (Naik ja Jenkins 2017.) OpenID Connect -protokollaa ei tule sekoittaa OpenID-protokollaan.

OpenID Connectissa on viisi roolia (Naik ja Jenkins 2017):

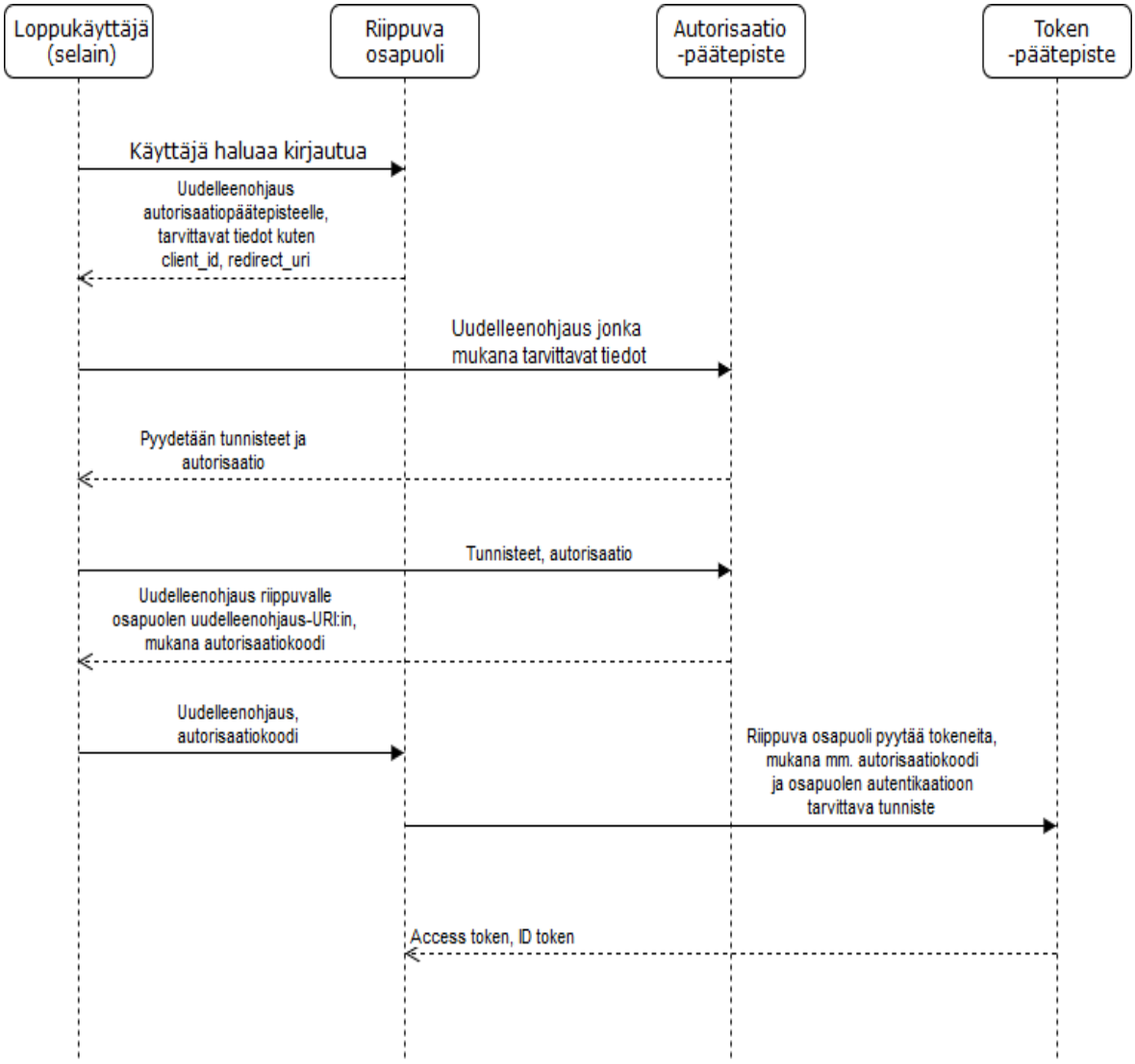
1. **Loppukäyttäjä** on sovelluksen käyttäjä ja tietojen omistaja.
2. **Riippuva osapuoli** on sovellus, joka kyselee suojattuja tietoja loppukäyttäjän puolesta.
3. **Autorisaatio-päätepiste** autentikoi loppukäyttäjän ja palauttaa autorisaatio-oikeuden tilanteesta riippuen asiakkaalle tai tai riippuvalle osapuolelle.
4. **Token-päätepiste** hoitaa access tokenien, ID tokenien, päivitystokenien yms. päivittämisen ja toimittamisen pyytäjille. Päätepisteeltä voi saada tokeneita validilla autorisaatiokoodilla.
5. **Käyttäjätieto-päätepiste** toimittaa käyttäjätietoja asiakassovellukselle access tokeneita vastaan.

OpenID Connectissa autentikaatiossa on kolme mahdollista kulkua (flow) (N. Sakimura ym. 2017, Luku 2):

1. Autorisaatiokoodi-kulku (Authorization code -flow)

2. Implisiittinen kulku (Implicit flow)
3. Hybridi-kulku (Hybrid flow)

Autorisaatiokoodi-kulku on tarkoitettu niille riippuville osapuolille, jotka voivat ylläpitää asiakassalaisuutta itsensä ja autorisaatiopalvelimen välillä turvallisesti ja implisiittinen kulku niille, jotka eivät. Hybridi-kulku on yhdistelmä kahta aiemmin mainittua, ja sen kautta access tokenin ja ID tokenin saa "yhdellä kierroksella" autorisaatiopalvelimen kautta samalla kun säilyy mahdollisuus hakea uusia tokeneita myöhemmin token-päätepisteestä. (N. Sakimura ym. 2017, Luku 2.)



Kuvio 4. Tyypillinen OpenID Connect -kirjautuminen Authorization code -kulun mukaisesti (Sakimura ym. 2017).

3.6 Single sign-out

SSO-järjestelmissä voi olla monenlaisia erilaisia uloskirjautumismahdollisuuksia, sillä käyttäjä voi olla kirjautunut useisiin palveluihin yhtä aikaa. Voidaan erotella käyttäjän istunnot eri palveluntarjoajilla ja niiden tarjoamissa eri sovelluksissa sekä SSO-istuntoa hoitavalla identiteetintarjoajalla. Käyttäjä voisi siis kirjautua ulos eli päättää näitä istuntoja yhden palveluntarjoajan yhdestä palvelusta, kokonaan yhdeltä palveluntarjoajalta, identiteetintarjoa-

jalta tai yhdellä kertaa kaikilta palveluntarjoajilta ja identiteetintarjoajalta ja niin edelleen. (Suoranta ym. 2013, s. 150-151.)

Single sign-out on ongelma siksi, että ilman sitä jostain palvelusta uloskirjauduttuaan käyttäjä jää vielä kirjautuneeksi kaikkiin muihin palveluihin, joihin on kirjautunut. Näin esimerkiksi julkisessa käytössä olevalle tietokoneelle saattaa jäädä käyttäjän tajuamatta mahdollisuus päästä käsiksi hänen tietoihinsa muissa palveluissa. (Rastogi ja Agrawal 2015.) Toisaalta voi olla, että käyttäjä ei halua kirjautua ulos kaikista palveluista kerralla, vaan vain tietyistä.

Suorannan ym. mukaan SSO-järjestelmässä uloskirjautumisen pitäisi olla aina joko paikallinen uloskirjautuminen tai single sign-out. Paikallisessa uloskirjautumisessa käyttäjän istunto yhdessä palvelussa ja palveluntarjoajalla päättyvät, mutta SSO-istuntoa tai istuntoja muiden palveluntarjoajien kanssa ei päätetä. Single sign-outissa taas päätetään kaikki istunnot muillakin palveluntarjoajilla sekä identiteetintarjoajalla. Suorannan ym. mukaan käytännössä single sign-out on useimmiten toteutettu kuitenkin vain yhdistämällä paikallinen uloskirjautuminen ja identiteetintarjoajalta uloskirjautuminen, jolloin istunnot toisten palveluntarjoajien kanssa jäävät voimaan. (Suoranta ym. 2013, s. 151)

Suorannan ym. mukaan keskitetyissä SSO-järjestelmissä single sign-out onnistuu, koska identiteetintarjoaja pitää kirjaa niistä palveluntarjoajista, joille asiakas on kirjautunut ja viestii niiden kanssa backend-viestikanavan kautta. Federoidut SSO-järjestelmät eivät näin yleensä tee, joten niissä täydellinen single sign-out vaatii muita ratkaisuja. (Suoranta ym. 2013, s. 155.)

Ratkaisut vaihtelevat sen mukaan, mikä osapuoli istunnoista huolehtii. Yksittäinen sovellus voi käyttää omaa istuntoaan tai palveluntarjoajan istuntoa. Mikäli käytetään palveluntarjoajan istuntoa, riittää että sovelluksessa on mahdollisuus pyytää palveluntarjoajaa toteuttamaan paikallinen uloskirjautuminen tai identiteetintarjoajalta uloskirjautuminen. Jos sovellus käyttää omaa istuntoaan, tulee tuo istunto poistaa ennen kuin kutsutaan palveluntarjoajan uloskirjautumista. Lisäksi sovelluksessa pitäisi olla käsittelijä palveluntarjoajan kautta tuleville identiteetintarjoajan aloitteesta tapahtuville uloskirjautumisille. Uloskirjautumispyyntöjen oikean toiminnan vuoksi sovellusten istuntojen eliniän tulisi olla lyhyempi kuin palveluntarjoajien. (Suoranta ym. 2013, s. 155-156.)

Suoranta ym. esittävät mahdolliseksi single sign-outin ratkaisuksi viestintää käyttäjän selaimen ja palveluntarjoajien sekä identiteetintarjoajien välillä tai viestintää palveluntarjoajien ja identiteetintarjoajan välillä. Selainpohjaisesti voitaisiin toimia niin, että käyttäjän ollessa jonkin palveluntarjoajan tai identiteetintarjoajan sivuilla selain lähettää niille sopivin aikavälein keep alive -viestejä. Kun käyttäjä poistuu sivuilta, keep alive -viestien lähettäminen loppuu ja istunnot voidaan lopettaa. Mikäli käyttäjä ei ole enää millään federaatioon kuuluvalla sivulla, loppuu keep alive -viestien lähettäminen myös identiteetintarjoajalle, joka voi poistaa SSO-istunnon ja vielä aloittaa uloskirjautumisen myös kaikilta palveluntarjoajilta. Toisaalta palveluntarjoajat voisivat säännöllisin väliajoin lähettää viestejä identiteetintarjoajille siitä, onko käyttäjän SSO-istunto vielä voimassa. Mikäli ei ole, ne voivat lopettaa käyttäjän istunnot omissa palveluissaan. Viestien lähettäminen kuormittaisi palveluntarjoajia ja identiteetintarjoajaa, joten niiden aikavälien pitäisi olla riittävän pitkä. (Suoranta ym. 2013, s. 156.)

4 Kirjautuminen TIM-järjestelmässä

Tässä luvussa käydään läpi TIMiin kirjautumista sellaisena kuin se on ennen sosiaalisen median SSO-integrointeja. Tähän esittelyyn kuuluvat kirjautumiseen oleellisesti liittyvät tietokannan osat, yksityiskohtaiset esitykset olemassaolevista kirjautumisprosesseista sekä esitys TIMin käyttöoikeuksista.

4.1 Käyttäjätilit TIMissä

Käyttäjätilit tallennetaan TIMin tietokannassa `useraccount`-tauluuun. Taulun sarakkeet ovat:

- `created`: Milloin tili luotiin; tyyppiä DateTime`DateTime`.
- `modified`: Milloin tiliä muokattiin; tyyppiä DateTime`DateTime`.
- `id`: Käyttäjätilin tunniste; tyyppiä `Integer`.
- `name`: Käyttäjänimi; tyyppiä `Text`.
- `given_name`: Käyttäjän etunimi; tyyppiä `Text`.
- `last_name`: Käyttäjän sukunimi; tyyppiä `Text`.
- `real_name`: Käyttäjän koko nimi; tyyppiä `Text`.
- `email`: Käyttäjän sähköpostiosoite; tyyppiä `Text`.
- `prefs`: Käyttäjän asetukset JSON-muodossa; tyyppiä `Text`.
- `pass_`: Salasana, joka on salattu bcryptillä; tyyppiä `Text`.
- `consent`: Hyväksyntä evästeille ja tiedonkeruulle; tyyppiä `Enum`.
- `origin`: Mitä kautta käyttäjätili rekisteröitiin TIMiin; tyyppiä `Enum`.

4.2 Tietokantaoperaatiot TIMissä

login.py:ssä tietokantaoperaatiot toimivat seuraavasti:

1. Luodaan reitissä tietuetta vastaava olio käyttäen apuna SQLAlchemyn malliluokkia
2. Kutsutaan oliota tekemään jotain muutoksia itseensä, kuten vaihtamaan salasana.
3. Reitissä lopuksi tehdään `db.session.commit()`, että muutokset tapahtuvat tie-

tokannassa.

4.3 Kulku autentikoidun istunnon luomiseksi TIMissä

4.3.1 Sähköpostikirjautuminen (login.py)

1. Tallennetaan polku, josta käyttäjä tuli `save_came_from()`.
2. Otetaan request-oliosta talteen kentät `email`, `password` ja `add_user`.
3. Etsitään käyttäjä tietokannasta ja luodaan sitä vastaava olio:

   - `user = User.get_by_email_or_username(email_or_username)`

4. Jos käyttäjä löytyy, tarkistetaan salasana.

   - Jos salasana on ok:
     a. Varmistetaan että käyttäjällä on henkilökohtainen ryhmä.
        – `user.get_personal_group()`

        Luodaan jos ei ole:

        –
        ```
        ug = UserGroup(name=user.name)
        user.groups.append(ug)
        db.session.commit()
        ```

     b. Asetetaan käyttäjä sessioon `set_user_to_session(user)`
     c. Viimeiseksi kutsutaan `finish_login()`, joka lähettää vastauksen kirjautumisesta käyttäjälle. Tarvittaessa ohjaa myös sivulle, jolta kirjautuminen tehtiin.
   - Jos ei löydy:
     a. Kutsutaan funktiota check_password_hash(...) luomaan viivettä, jotta vältytään kirjautumisen aika-analyysiin perustuvalta sivukanavahyökkäykseltä.
     b. Lähetetään virheviesti
        – Jos request-oli XMLHttp-pyyntö: `return abort(403, error_msg)`
        – Muuten:
          * `flash(error_msg, "loginmsg")` ja

`* return finish_login(ready=False)`

4.3.2 HAKA-kirjautuminen (saml.py ja login.py)

1. Valittavissa olevien identiteetintarjoajien tiedot ladataan palvelimelta kirjautumisdialogin avaamisen yhteydessä.
2. Käyttäjä valitsee haluamansa identiteetintarjoajan ja painaa kirjautumispainiketta.
3. Käyttäjä uudelleenohjataan palvelimelle reittiin `/saml/sso`, jonka toiminnallisuus on funktiossa `sso`.

    - Tehdään useita toimenpiteitä käyttäen apuna mm. python3-saml -kirjastoa, jotta saadaan aikaan URL, johon käyttäjä ohjataan kirjautumaan.
    - Ennen uudelleenohjaamista:
        - Lähetetään virheilmoitus, mikäli istuntoon yritetään lisätä uutta käyttäjää, vaikka istuntoon ei ole vielä kirjautunut kukaan.
        - Tallennetaan istuntoon tieto siitä, ollaanko lisäämässä istuntoon uutta käyttäjää.
        - Tallennetaan istuntoon äsken luodun Request SAML -viestin id.
        - Tallennetaan istuntoon käytetyn identiteetintarjoajan entity id.
4. Käyttäjä uudelleenohjataan kirjautumaan identiteetintarjoajalle.
5. Kirjautumisen jälkeen käyttäjä uudelleenohjataan takaisin TIM:in reittiin `/saml/acs`, jonka toiminnasta vastaa funktio `acs`.

    - Mikäli istunnosta ei löydy entity id:tä, palautetaan virheviesti.
    - Mikäli istunnosta ei löydy SAML -viestin identifioivaa request id:tä, palautetaan virheviesti.
    - Näitä tunnisteita käytetään identiteetintarjoajalta saadun vastausviestin käsittelyssä. Mikäli SAML-viestin käsittelyssä tapahtuu jotain virheitä tai kirjautumien ei muuten onnistu, lähetetään käyttäjälle virheviesti.
6. Jos kirjautuminen onnistuu:

    - Poistetaan istunnosta request id.
    - Luodaan olio, jolta saadaan kysyttyä SAML-viestillä välitettyjä käyttäjätietoja.

- Luodaan identiteetintarjoajan organisaatiota vastaava käyttäjäryhmä.
- Tarkistetaan, löytyykö käyttäjän tiedoista koodeja, joilla hänet voidaan yksilöidä (schacPersonalUniqueCode). Mikäli sellaisia ei löydy, logataan varoitusviesti.

7. Pyydetään käyttäjää vastaava `User`-olio kutsumalla funktiota `create_or_update_user`.

    - Parametreina välitetään identiteetintarjoajan organisaation käyttäjäryhmä sekä `UserInfo`-olio, joka sisältää tarvittavat käyttäjätiedot.
    - Funktiossa etsitään käyttäjätietoja vastaavaa käyttäjää, kunnes sellainen löydetään. Etsiminen tapahtuu seuraavassa järjestyksessä.
        1. Käyttäjänimen perusteella.
        2. Jos käyttäjätiedoissa on sähköposti, niin sähköpostin perusteella.
        3. Jos käyttäjätietoissa on käyttäjän yksilöiviä uniikkeja koodeja, niin etsitään niiden perusteella.
    - Jos käyttäjä (`user`) löytyy:
        1. Jos identiteetintarjoajalta saaduissa käyttäjätiedoissa on sähköposti ja tämä sähköposti on eri kuin tietokannasta löytyneen käyttäjätunnuksen sähköposti:
        - Etsitään tietokannasta käyttäjää (`ue`) identiteetintarjoajalta saadulla sähköpostilla.
        - Mikäli `ue` löytyy ja se on eri käyttäjä kuin aiemmin löydetty `user`, niin yhdistetään käyttäjätunnukset kutsumalla `do_merge_users` ja poistetaan `ue` tietokannasta kutsumalla `do_soft_delete`. Käyttäjätunnuksen poisto merkitsee tietyt käyttäjätilin tietojen arvot päätteellä `_deleted`.
        - Mikäli `ue` löytyy ja se on sama käyttäjä kuin `user`, lähetetään kirjautujalle virheviesti.
        2. Päivitetään käyttäjän tiedot identiteetintarjoajalta saaduilla.
    - Jos käyttäjää ei löydy, luodaan uusi käyttäjä ja sitä vastaava käyttäjäryhmä kutsumalla `User`-luokan metodia `create_with_group`.
    - Tarkistetaan, onko funktiolle toimitettu käyttäjäryhmä `group_to_add`-parametrissa, ja onko tämä käyttäjäryhmä jo asetettu käyttäjälle. Jos ryhmää ei

käyttäjällä vielä ole, se lisätään `User`-olion `groups`-listaan.
    - Lopuksi palautetaan käyttäjä.
8. Jos käyttäjän käyttäjäryhmissä ei ole ryhmää `Haka users`, lisätään tämä `User`-olion `groups`-listaan.
9. Tallennetaan tietokantaan tehdyt muutokset kutsumalla `db.session.commit()`.
10. Lisätään käyttäjä istuntoon kutsumalla `set_user_to_session`.
11. Päätetään kirjautuminen:

    - Jos istunnosta löytyy arvo `debugSSO`, palautetaan JSON-vastaus sisältönä käyttäjän SAML-attribuutit.
    - Jos pyynnön lomakedatasta löytyy arvo `RelayState`, uudelleenohjataan käyttäjä sen osoittamaan paikkaan.
    - Muussa tapauksessa uudelleenohjataan käyttäjä TIMin aloitussivulle.

4.3.3 Korppikirjautuminen (login.py)

Jyväskylän yliopiston opiskelijat ja henkilökunta pystyivät kirjautumaan TIMiin aiemmin Korppi-tietojärjestelmän kautta. Tällä lailla Jyväskylän yliopistoon kuuluvat käyttäjät saivat omia käyttöoikeuksiaan kuulumalla korppikäyttäjien käyttäjäryhmään. Sosiaalisen median SSO-integrointeja alettiin tekemään kun korppikirjautuminen oli vielä käytössä, mutta nyt tämän kirjautumisen on korvannut Haka-kirjautuminen.

Funktiossa `login_with_openid()`:

1. Tarkastetaan kyselyparametrista `add_user`, onko kyseessä yhden käyttäjän kirjautuminen vai uuden käyttäjän lisääminen istuntoon.
2. Tarkastetaan, onko jo kirjauduttu TIM:iin. Mikäli on, lähetetään käyttäjälle viesti, että näin on.
3. Mikäli istuntoon ei ole vielä tallennettu `adding_user` -muuttujaa, se tallennetaan: `session[adding_user] = add_user`.
4. Tarkistetaan, että kyselyparametri `provider` on arvoltaan “korppi”. Muussa tapauksessa lähetetään käyttäjälle virheviesti, sillä Korppi on ainoa TIMin käyttämä OpenID-tarjoaja (provider).

5. Tallennetaan URL josta kirjautuminen aloitettiin kutsumalla `save_came_from()`.
6. Kutsutaan `KorppiOpenID`-olion metodia `try_login()`, joka ohjaa käyttäjän kirjautumaan Korppiin.

Funktiossa `openid_success_handler(...)`:

7. Tarkistetaan, että
   - vastauksena saadun KorppiOpenIDResponse-olion `identity_url`-kenttä on oikeanlainen
   - vastaus sisältää myös muut tarvittavat tiedot
8. Kutsutaan `create_or_update_user(...)`-funktiota, jossa etsitään käyttäjää `useraccount`-tietokantataulusta ensisijaisesti käyttäjänimen ja toissijaisesti sähköpostin perusteella.
   - Jos käyttäjää ei löydy, se luodaan kutsumalla `User.create_with_group(...)`.
   - Jos käyttäjä löytyy, sen tiedot päivitetään funktiolle toimitetuilla parametreilla kutsumalla `User.update_info(...)`.
   - Lisätään funktiolle toimitettu `group_to_add`-käyttäjäryhmä käyttäjän käyttäjäryhmiin `user.groups`, ellei sitä ole jo lisätty.
   - lopuksi palautetaan käyttäjä
9. Kutsutaan `db.session.commit()`, että edellisessä kohdassa tehdyt muutokset tallentuvat tietokantaan.
10. Kutsutaan set_user_to_session(...) -funktiota, joka istunnon `adding_user`-muuttujan perusteella joko
    - lisää käyttäjän istunnon `other_users`-muuttujaan, tai
    - lisää käyttäjän id:n istunnon `user_id`-muuttujaan ja poistaa istunnosta `other_users`-muuttujan.
11. Kutsutaan `finish_login()` -funktiota.

4.4 TIMin kirjautumisen ja käyttäjätilien ominaisuuksia

Seuraavassa joitakin huomioitavia ominaisuuksia TIMin kirjautumisessa ja käyttäjätileissä:

- TIMin sähköpostitili ja korppitili yhdistyvät automaattisesti, jos niillä on sama sähköposti.
- Käyttäjätilejä voi yhdistää jo olemassaolevilla funktioilla tiedostossa timApp/routes/admin.py
- Jos luo Korpissa olevalla sähköpostitilillä TIM-tilin korppitilin luomisen jälkeen, lisätään korpitiliin vain salasana, minkä jälkeen samalle tilille voi kirjautua suoraan TIMissä sähköpostilla ja salasanalla.

4.5 Käyttöoikeustasot TIMissä

Käyttöoikeustasot ovat käyttäjäryhmiä, joilla määritellään käyttöoikeuksia yleisesti riippuen kirjautumistavasta. Käyttäjäryhmät ovat tietokannassa `usergroup`-taulussa. Koska käyttäjien ja käyttäjäryhmien välillä on monesta moneen -suhde, käyttäjien kuuluminen käyttäjäryhmiin toteutetaan `usergroupmember`-taulussa. Jos käyttäjällä on käyttäjätili, niin hän kuuluu omaan henkilökohtaiseen käyttäjäryhmäänsä. Jokainen käyttäjä kuuluu myös yhteen tai useampaan seuraavista ryhmistä:

- `Haka users`
  - Hakan kautta kirjautuneet käyttäjät lisätään tähän ryhmään kirjautumisen yhteydessä.
  - Haka-kirjautumisessa luodaan lisäksi käyttäjäryhmä kutakin identiteetintarjoajaa kohti. Haka-kirjautujat kuuluvat TIMissä siis myös omaa identiteetintarjoajaansa vastaavaan käyttäjäryhmään.
- `Logged-in users`
  - Tähän kuuluvat kaikki kirjautuneet käyttäjät riippumatta kirjautumistavasta.
  - Tietokannassa ketään ei lisätä tähän ryhmään, koska tähän pitäisi lisätä kaikki kirjautuneet käyttäjät.
- `Anonymous users`
  - Kirjautumattomat käyttäjät.

TIM-tiliin voi lisätä salasanan kirjautumista varten, vaikka tili olisi aiemmin luotu vaikkapa Hakan kautta kirjautumalla. Kun Hakan kautta luodulle tilille kirjaudutaan salasanan avulla,

käyttäjän käyttäjäryhmiin ei tehdä muutoksia. TIM-tilin oman salasanan avulla kirjautumalla saa siis kaikki ne käyttöoikeudet, joita käyttäjätilille on aiemmin annettu.

5 SSO-toteutukset

Tässä luvussa esitellään sosiaalisen median SSO-integrointeja TIM-järjestelmään. Ensimmäisenä esitellään erilaisia SSO-ratkaisuja. Sitten esitellään SSO-integrointien vaatimukset. Seuraavaksi esitellään mitä käyttäjätietoja Googlen, Facebookin ja Twitterin SSO-palveluilta on saatavissa. Tämän jälkeen esitellään tarvittavat muutokset tietokantaan ja yleisluontoinen esitys siitä, mitä osia SSO-palvelun kautta tapahtuvassa kirjautumisessa tulisi olla. SSO-integrointien toteutumat esitetään tämän jälkeen ja lopuksi niitä vielä vertaillaan keskenään. Toteutuksiin liittyvä ohjelmakoodi on nähtävillä GitLabissa[1]. Pääosa tämän tutkielman aikana kirjoitetusta ohjelmakoodista löytyy hakemistoista timApp/auth/[2] ja timApp/static/scripts/tim/user[3].

5.1 Google-kirjautuminen

Google tarjoaa verkkosivujen kirjautumisen toteuttamiseen kolmea eri ratkaisua:

1. Firebase authentication,
2. Google Sign-In,
3. Puhdas OAuth 2.0. (Google 2017.)

Firebase on sovelluskehitysalusta, joka mahdollistaa useiden eri kirjautumistapojen yhdistämisen (esim. sähköposti ja salasana, Google Sign-In) käyttämällä esimerkiksi Firebase SDK:ta. Firebase sisältää monenlaisia muitakin palveluja, kuten verkkoisännöintiä, analytiikkaa, tietokantoja ja niin edelleen. (Google, n.d.[a].) Firebase ei ole täysin ilmainen palvelu, mutta sitä voi käyttää tietyillä rajoituksilla ilmaiseksi ja autentikaatiopalvelut lukuunottamatta puhelinautentikaatiota ovat ilmaisia (Google, n.d.[b]). Google Sign-In on OpenID Connect -protokollan toteuttava kirjasto (Google 2019c), jonka avulla käyttäjille voi antaa mahdollisuuden kirjautua verkkosivuille, mobiilisovelluksiin ja niin edelleen käyttäen google-tunnuksia. On myös mahdollista toteuttaa kirjautuminen itse OAuth 2.0 -protokollan

1. https://gitlab.com/maollein/tim
2. https://gitlab.com/maollein/tim/-/tree/master/timApp/auth
3. https://gitlab.com/maollein/tim/-/tree/master/timApp/static/scripts/tim/user

avulla käyttäen Googlen OAuth endponttia. (Google 2017.) Google tarjoaa eri ohjelmointikielille OAuth 2.0 -protokollan toteutusta tukevia kirjastoja (Google 2019c).

Googlen kirjautumispalvelun toteutus perustuu OAuth 2.0 -protokollaan ja on tehty OpenID Connect -spesifikaation mukaan (Google 2019c). OpenID Connect tarjoaa asiakassovellukselle mahdollisuuden tunnistaa käyttäjä. Käyttäjä kirjautuu autorisaatiopalvelimelle, jolta asiakassovellus saa käyttäjän identiteetin ja profiilitietoja. (Foundation 2019.)

Googlen tarjoamista kirjautumisvaihtoehdoista Google Sign-In on arvioni mukaan paras, kun halutaan toteuttaa SSO-integrointi TIMiin. Google Sign-In:in voisi toteuttaa myös Firebasen kautta, mutta katson, että se ei ole tarkoituksenmukaista, koska sen voi toteuttaa suoraankin. Firebase toisi toteutukseen turhan “abstraktion tason”. Ei olisi myöskään järkevää käyttää sitä yhdistämään kaikkia kirjautumismenetelmiä, sillä niitä on jo TIMissä kaksi. Google Sign-In vaikuttaa yksinkertaisemmalta toteuttaa kuin oma ratkaisu käyttäen Googlen OAuth endpointtia. Google Sign-in:in toteuttaminen vaatii Google API Console projektin tekemistä ja sitä kautta client id:n saamista ja tarvittavien asetusten tekemistä (Google 2019b).

5.2 Facebook-kirjautuminen

Facebook tarjoaa verkkosivuille sekä autentikaatiopalvelua, että mahdollisuutta kysellä lupia käyttäjän tietojen saamiseen Facebookilta. Pelkkä autentikaatio ei vaadi sovelluksen arvioittamista Facebookilla. (Facebook, n.d.[e].) Facebook Loginissa sovelluksen käyttäjä ohjataan kirjautumaan Facebookiin ja Facebook välittää sovellukselle käyttöoikeustunnuksen (access token). Kun toteutus tehdään ilman Facebookin SDK:ta, on mahdollista pyytää kirjautuessa koodi-parametria, jonka avulla facebookilta saa käyttöoikeustunnuksen (Facebook, n.d.[i]). Käyttöoikeustunnuksen avulla voi Facebookilta kysyä käyttäjän autentikoimiseen tarvittavia tietoja.

Facebook Login perustuu OAuthiin (Facebook, n.d.[d]). Facebook tarjoaa ja listaa kirjautumisen toteuttamiseen verkkosivuille useita omia ja kolmansien osapuolien kirjastoja (Facebook, n.d.[a]) sekä Login-painikkeen (Facebook, n.d.[g]). Facebook ohjeistaa myös miten toteutus voidaan tehdä ilman mitään SDK:ta (Facebook, n.d.[i]). Facebook-kirjautumisen käyttöön saaminen vaatii sovelluksen rekisteröimistä ja tarvittavien asetusten tekemistä Faceboo-

kin App Dashboardissa, jota voi käyttää sovelluskehittäjätiliksi muutetulla facebooktilillä (Facebook, n.d.[b]).

Facebook tarjoaa käyttäjille mahdollisuuden pyytää tietojensa poistoja sovelluksilta vuorovaikuttamatta suoraan sovellusten kanssa. Sovellukset voivat toteuttaa tietojen poistamismahdollisuuden kuuntelemalla facebookin lähettämiä tietojenpoistamispyyntöjä. (Facebook, n.d.[c].)

5.3 Twitter-kirjautuminen

Twitterin tarjoama kirjautuminen perustuu OAuth 1.0a -protokollaan. Käyttäjälle voidaan näyttää kirjautumispainike, jonka ulkoasun Twitter tarjoaa. Seuraavassa Twitter-kirjautumisen kulku:

1. Käyttäjä painaa sovelluksessa näkyvää Sign in with Twitter -painiketta
2. Sovellus pyytää Twitteriltä request tokenia allekirjoitetulla viestillä toimittaen samalla osoitteen, johon käyttäjä uudelleenohjataan kirjautumisensa jälkeen.
3. Twitter lähettää sovellukselle vastauksen, joka sisältää parametrit `oauth_token`, `oauth_token_secret` ja `oauth_callback_confirmed`, jonka arvon tulee olla `true`. `oauth_token` ja `oauth_token_secret` tulee tallentaa myöhempää käyttöä varten.
4. Sovellus lähettää käyttäjälle uudelleenohjauksen Twitterin API:n `/oauth/authenticate`-osoitteeseen kyselyparametrina aiemmin saatu `oauth_token`.
5. Twitterin kirjautumisosoitteessa voi tapahtua kolme eri vaihtoehtoa:
    a. Jo kirjautunut ja sovelluksen hyväksynyt käyttäjä autentikoidaan ja uudelleenohajataan välittömästi takaisin sovellukseen OAuth request tokenin kanssa.
    b. Kirjautunutta, mutta sovellusta vielä hyväksymätöntä käyttäjää pyydetään hyväksymään sovellus, jonka jälkeen hänet uudelleenohjataan OAuth request tokenin kanssa takaisin sovellukseen.
    c. Kirjautumaton ja sovellusta hyväksymätön käyttäjä pyydetään antamaan tunnuksensa sekä hyväksymään sovellus, minkä jälkeen hänet uudelleenohjataan sovel-

lukseen OAuth request tokenin kanssa.

6. Kun käyttäjä uudelleenohjataan sovelluksen uudelleenohjausosoitteeseen, vastaanottaa palvelin parametrit `oauth_token` ja `oauth_verifier`. Sovelluksen tulee tarkistaa, että `oauth_token` vastaa kohdassa 2 saatua request tokenia.
7. Request token muutetaan access tokeniksi lähettämällä POST-viesti Twitterin API:n osoitteeseen `/oauth/access_token`. Viestiin sisällytetään `oauth_verifier`. `oauth_token` on viestissä mukana otsikkotiedoissa, mutta se sisällytetään otsikkotietoihin viestiä allekirjoitettaessa. Twitteriltä saadaan vastauksena autentikoitujen pyyntöjen tekemiseen tarvittavat `oauth_token` ja `oauth_token_secret`.
8. Käyttäjän identiteetin saa selville lähettämällä pyynnön Twitterin API:n osoitteeseen `account/verify_credentials`. (Twitter, n.d.[b].)

Twitterin tarjoamia palveluja käyttääkseen ohjelmistokehittäjällä tulee olla kehittäjätili. Kehittäjätilin avulla voi luoda Twitter appeja ja saada niille tunnisteet ja asettaa erilaisia asetuksia. (Twitter, n.d.[c].) Twitter listaa yhden oman ja lukuisia kolmansien osapuolien kirjastoja ohjelmointirajapintojensa käyttämiseen (Twitter, n.d.[d]). TIM:n kannalta relevantteja kirjastoja ovat Python-kirjastot, joita on useita.

5.4 Haka

Haka on käyttäjäntunnistusjärjestelmä, johon kuuluu mm. 54 jäsentä, 53 identiteetintarjoajapalvelinta ja 361 rekisteröityä palvelua. Sitä käyttää 326 000 loppukäyttäjää ja se onkin Suomen korkeakoulujen ja tutkimuslaitosten käytetyin käyttäjäntunnistusjärjestelmä. (Eduuni-wiki 2019b.) Hakaa kehittää CSC-Tieteen tietotekniikan keskus (CSC - Tieteen tietotekniikan keskus Oy, n.d.[a]), joka on valtion ja korkeakoulujen omistama voittoa tavoittelematon erityistehtäväyhtiö (CSC - Tieteen tietotekniikan keskus Oy, n.d.[b]).

Haka-luottamusverkoston jäseneksi voivat liittyä (Eduuni-wiki 2019a)

- "yliopistot ja ammattikorkeakoulut
- lailla perustetut, julkista tehtävää suorittavat tieteen ja taiteen toimielimet, viranomaiset ja tutkimuslaitokset
- em. organisaatioiden opetus- ja tutkimustoimintaa tukevat organisaatiot"

Muut organisaatiot voivat liittyä loppukäyttäjille palveluja tarjoaviksi kumppaneiksi. Hakaan liittyminen vaatii organisaation toiminnan yhteensopivuutta Haka-verkoston tarkoituksen kanssa, sekä niiden ehtojen täyttymistä, jotka määritellään Haka-palvelusopimuksessa sekä sen liitteissä. (Eduuni-wiki 2019a.)

Haka on toteutettu SAMLia hyväksikäyttäen, eli kotiorganisaatiot tarjoavat palveluille käyttäjätietoja SAML-määrittelyjen mukaisesti. Hakaan liittyvä palveluntarjoaja tai identiteetintarjoaja voi käyttää mitä tahansa määritykset täyttävää SAML-ohjelmistoa, mutta Hakassa käytetään pääosin avoimen lähdekoodin lisenssillä julkaistua Shibbolethia. (Eduuni-wiki 2016b; 2016a.)

5.5 MPASSid

Verkkosivuillaan MPASSid:tä kuvaillaan seuraavasti: “MPASSid on opetus- ja kulttuuriministeriön tarjoama tunnistusratkaisu, joka standardoi henkilötietojen välittämisen opetuksen tai koulutuksen järjestäjän henkilörekisteristä sähköisiin oppimispalveluihin. Se on suunnattu perus- ja toisen asteen oppilaitoksille.” MPASSid-palvelun omistaa Opetus- ja kulttuuriministeriö ja sen operaattorina toimii CSC - Tieteen tietotekniikan keskus Oy. (“MPASSid - Enemman aikaa opetukselle ja oppimiselle” n.d..)

MPASSid:n jäsenorganisaatiot muodostavat luottamusverkoston, jonka tarkoitus on “sopia yhteisestä loppukäyttäjien tunnistamista ja käyttäjäidentiteettien välittämistä koskevasta organisaatiorajat ylittävästä välityspalvelusta ja sen käyttöä koskevista säännöistä”. Luottamusverkoston jäseneksi voivat liittyä opetusta ja koulutusta järjestävät tahot sekä tahot, jotka tarjoavat opetuksen ja koulutuksen järjestäjille opetuksessa ja koulutuksessa käytettäviä sähköisiä palveluita. (“MPASSid liittymissopimus” n.d..)

Luottamusverkossa on kahta erilaista roolia. Kotiorganisaatiot, jotka käytännössä ovat opetuksen järjestäjiä, ylläpitävät rekisteriä loppukäyttäjien tiedoista ja toimivat rekisterinpitäjinä. Palveluntarjoajat taas käyttävät kotiorganisaatioiden ylläpitämien rekistereiden tietoja tarjoamissaan palveluissa. Luottamusverkoston operaattori tarjoaa jäsenille tunnistus- ja henkilötietojen välityspalvelun. (“MPASSid liittymissopimus” n.d..) Palveluntarjoajat voivat yhdistää palvelunsa MPASSid:hen käyttämällä joko SAML 2.0:aa tai OpenID Connectia

(Eduuni-wiki 2019c).

5.6 SSO-kirjautumisratkaisujen vaatimukset

- Ensimmäisen kerran SSO-tilillä kirjauduttaessa on voitava luoda paikallinen käyttäjätili TIMiin, johon voidaan jatkossa kirjautua.
- Käyttäjällä on oltava yksi paikallinen käyttäjätili, jolle kirjautuminen mahdollistetaan useilla eri tavoilla.
- Pakolliset käyttäjätiedot:
  - uniikki käyttäjänimi
  - sähköposti
  - koko nimi
- Ei-pakollisia SSO-palveluilta haluttavia käyttäjätietoja:
  - etunimi
  - sukunimi
- TIMissä samaan istuntoon voidaan lisätä useita käyttäjiä. Tämän ominaisuuden on toimittava myös SSO-kirjautumisissa.
- Eri kirjautumistavoilla saa erilaiset käyttöoikeudet.
  - Sosiaalisen median kirjautumisilla tulee saada vain `Logged-in users` -tason oikeudet.

5.7 Facebookilta, Googlelta ja Twitteriltä saatavat käyttäjätiedot

5.7.1 Facebook

Facebookilta tarvitaan default-käyttöoikeutta ja email-käyttöoikeutta, jotka saadaan käyttöön ilman mitään erillistä lupaa. Näillä oikeuksilla saadaan seuraavat käyttäjätiedot:

- default -käyttöoikeus:
  - id

  - first_name
  - last_name
  - middle_name
  - name
  - name_format
  - picture
  - short_name
- email -käyttöoikeus
  - email

Sähköpostia saa käyttää käyttäjin kirjautumiseen ja yhteydenpitoon käyttäjien kanssa. (Facebook, n.d.[j])

5.7.2 Google

Google Sign-in:in oletuskäyttöoikeudet eli scopet ovat “email profile openid”, joilla saadaan käyttöön mm. kaikki TIMissä tarvittavat tiedot:

- id
- name
- given_name
- last_name
- email (Google 2019b)

5.7.3 Twitter

Twitteriltä saadaan ainakin seuravaat TIM:issä tarvittavat tiedot:

- id
- name
- screen_name
- email, jos erikseen pyydetään ja sovelluksen yksityisyyskäytäntöjen ja käyttöehtojen URL:t on annettu Twitterille sovelluksen asetuksissa (Twitter, n.d.[a])

5.8 Kirjastoja

- Googlen oma OAuth-kirjasto Pythonille[4]
- Google Sign-in kirjasto, JavaScript[5]
- Facebok login JavaScript SDK[6]
- Useita OAuth-kirjastoja Pythonille[7]

Twitter käyttää OAuth 1.0a:ta, mitä varoittavat toteuttamasta selainpuolen JavaScriptillä. Twitterin osalta täytyy siten käyttää jotain lukuisista yhteensopivista python-kirjastoista[8].

5.9 Sosiaalisen median SSO-tilien integroiminen TIMiin

Sosiaalisen median SSO-kirjautumisia alettiin toteuttaa TIMiin Korppi-kirjautumisen ollessa vielä käytössä. Haka-kirjautuminen korvasi Korppi-kirjautumisen ennen SSO-kirjautumisten valmistumista. Haka-kirjautuminen toi kirjautumiseen ominaisuuksia, joihin ei SSO-integrointien suunnitteluvaiheessa osattu varautua. Tämän vuoksi useiden kirjautumistapojen yhdistäminen jää vajaaksi. Esimerkiksi kun tässä luvussa esitetään käyttäjätietojen hallintaa silloin kun niitä saadaan useasta lähteestä, ei oteta huomioon Hakaa yhtenä lähteenä.

SSO-integrointeja ollaan kuitenkin alusta asti suunniteltu silmällä pitäen sitä, että useita eri kirjautumistapoja halutaan jatkossa yhdistää samalle käyttäjätilille. Vaikka kirjautumistapojen yhdistäminen samalle käyttäjätilille jää toteutuksen tasolla osin vajaaksi, esitellään kuitenkin ne suunnitellut sekä toteutetut toiminnallisuudet, joita yhdistäminen vaatii. Näistä suunnitelmista ja toteutuksista johdetaan myös tuloksia kuudennessa luvussa.

5.9.1 Kirjautuminen SSO-palvelun avulla

Tässä esitellään karkeamman tason yleinen suunnitelma, mitä toimenpiteitä kuuluu TIMiin kirjautumiseen SSO-palvelun avulla. Toteutumat esitellään luvuissa 5.10, 5.11 ja 5.12.

4. https://github.com/googleapis/google-api-python-client/blob/master/docs/README.md
5. https://developers.google.com/identity/sign-in/web/sign-in
6. https://developers.facebook.com/docs/javascript
7. https://oauth.net/code/python/
8. https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries

Kirjautuminen SSO-palveluntarjoajalle:

1. Käyttäjä klikkaa kirjautumisdialogissa SSO-palveluntarjoajalle kirjautumaan ohjaavaa painiketta.
2. Käyttäjä ohjataan kirjautumaan palveluntarjoajalle.
3. Käyttäjä kirjautuu ja tiedot käyttäjästä välitetään TIMiin.

SSO-palveluun kirjautumisen jälkeen TIM-palvelimella:

4. Tarkistetaan `sso_useraccount`-taulusta, onko käyttäjätili jo olemassa. Jos ei ole, jatketaan kohdasta 5, muuten jatketaan kohdasta 8.

Ensimmäinen kirjautuminen:

5. Luodaan `User`-olio SSO-palvelusta saaduilla tiedoilla luvun 5.9.2 mukaisesti.
6. Luodaan tietue `sso_useraccount`-tauluun ja yhdistetään se `User`-olion kautta muodostuneeseen `useraccount`-tietueeseeen.
7. Lisätään käyttäjä istuntoon jne. sähköpostikirjautumisen ja korppikirjautumisen malliin.

Kirjautuminen jo olemassaolevalle tilille

8. Luodaan `User`-olio `useraccount`-taulun tietueella, jonka `id`-numero on sama kuin `sso_useraccount`-taulusta löytyneen tietueen `useraccount_id`.
9. Päivitetään `sso_useraccount`-taulun tietueen tiedot.
10. Mikäli `sso_useraccount`-taulun tietue on merkitty ensisijaiseksi tälle käyttäjätunnukselle, päivitetään myös käyttäjätunnuksen tiedot, eli `useraccount`-taulun rivi.
11. Lisätään käyttäjä istuntoon ja päätetään kirjautumisprosessi sähköpostikirjautumisen ja korppikirjautumisen malliin.

5.9.2 Käyttäjän luominen

1. Käyttäjälle tulee generoida jokin käyttäjänimi
    - Nimen tulee olla uniikki

- Nimi ei saa olla sama kuin mikään tulevaisuudessa luotava korppinimi
  - Onnistuu erikoismerkki lisäämällä
- Koostetaan nimi SSO-palvelun identifioivasta etuliitteestä sekä käyttäjän sähköpostiosoitteesta.

2. Käyttäjälle tulee saada SSO-palveluntarjoajalta sähköposti
   - Kaikilta palveluntarjoajilta on ainakin mahdollista saada sähköposti.
   - Mikäli sähköpostia ei kirjautumisen yhteydessä saada, ei luoda käyttäjätunnusta, vaan pyydetään kirjautujaa antamaan sähköpostin käyttölupa ja tarvittaessa lisäämään sähköposti käyttäjätililleen SSO-palveluntarjoajalla.
3. Käyttäjällä pitää olla koko nimi
   - Mikäli nimeä ei saada SSO-palveluntarjoajalta, ei luoda käyttäjätunnusta, vaan pyydetään kirjautujaa antamaan käyttölupa ja tarvittaessa lisäämään nimi käyttäjätililleen SSO-palveluntarjoajalla.

5.9.3 SSO-tilit tietokannassa

SSO-tilit tallennetaan tietokantaan omaan tauluunsa:

Taulukko 3: `sso_useraccount`-taulu. Sarakkeiden ja rivien paikka vaihdettu.

id	1	2
useraccount_id	14	783
email	titus.p.atticus@example.com	atticus_p@example.com
username	Google:titus.p.atticus@example.com	Twitter:atticus_p@example.com
real_name	Titus P. Atticus	Pomponius Atticus
given_name	Titus P.	Pomponius
last_name	Atticus	Atticus
origin_id	235432563	434532
origin	Google	Twitter
primary_to_user	14	783

Selitykset:

- `id`: Tietueiden tunniste tässä taulussa, tyyppiä `integer`.
- `useraccount_id`: Käyttäjän tunniste TIMissä, viittaa taulun `useraccount` id-numeroon, tyyppiä `integer`.
- `email`: Käyttäjän sähköposti, tyyppiä `text`.
- `username`: Käyttäjän käyttäjänimi, tyyppiä `text`.
- `real_name`: Käyttäjän oikea nimi kokonaisuudessaan, tyyppiä `text`.
- `given_name`: Käyttäjän etunimi, tyyppiä `text`.
- `last_name`: Käyttäjän sukunimi, tyyppiä `text`.
- `origin_id`: Käyttäjän tunniste palvelussa, jota käytetään käytetään kirjautumiseen. Tyyppiä `text`.
- `origin`: Palvelu, jota käytetään kirjautumiseen, tyyppiä `userorigin` (`enum`).
- `primary_to_user`: Sama kuin `useraccount_id`, jos kirjautumistapa on ensisijainen käyttäjätililleen, muulloin `NULL`. Tyyppiä `integer`.

Rajoitukset:

- Asetetaan kenttien `origin_id` ja `origin` yhdistelmä uniikiksi. Tällä estetään tilanne, jossa kolmannen osapuolen tili yhdistettäisiin useampaan käyttäjätiliin TIMissä.
- Asetetaan kaikki kentät pakollisiksi, eli NOT NULL -rajoitetuiksi.

Jos käyttäjätilin tiedot saadaan vain yhdestä lähteestä, SSO-tilin käyttäminen tuona lähteenä on yksinkertaista. TIMissä voitaisiin tallentaa käyttäjätiedot vain `useraccount`-tauluun kuten nykyisinkin. SSO-tilejä varten voitaisiin tehdä `sso_useraccount`-taulu, joka sisältäisi vain viittauksen oikeaan `useraccount`-taulun riviin (`useraccount_id`), käyttäjän tunnisteen SSO-palveluntarjoajan järjestelmässä (`origin_id`) sekä kyseisen SSO-palveluntarjoajan tunnisteen (`origin`).

Kuitenkin mikäli samaan käyttäjätiliin haluttaisiin yhdistää useita kirjautumistapoja, kuten Facebook- ja Google-kirjautumiset, on käyttäjätietojen lähteitäkin useita. Käyttäjätiedot voivat olla erilaisia eri palveluissa. Esimerkiksi sähköpostiosoite on merkittävä tieto, joka todennäköisesti onkin eri palveluissa eri. On kiusallista, jos tämä tieto muuttuu käyttäjätilille

aina eri kirjautumistapaa käyttäessä toiseksi. Sähköposti, kuten muutkin käyttäjätiedot, on kuitenkin syytä pitää ajantasaisina, että palvelun toiminnan oikeellisuus säilyy. Sähköposteja ei haluta lähettää osoitteeseen, joka ei ole enää käytössä. Tämän vuoksi jokaisen SSO-tilin käyttäjätiedoista pidetään kirjaa erikseen. SSO-tilin käyttäjätiedot siirretään paikalliselle käyttäjätilille siinä tapauksessa, että SSO-tili on valittu ensisijaiseksi kirjautumistavaksi kyseiselle käyttäjätilille. Käyttäjätietojen hallintaa useiden käyttäjätietolähteiden kanssa esitetään luvussa 6.1.8.

5.9.4 Haka-kirjautumisen ja sosiaalisen median SSO-kirjautumisten konflikti

Haka-kirjautumista ja sosiaalisen median SSO-kirjautumisia ei olla toteutettu siten, että niiden yhdistäminen samaan käyttäjätiliin olisi suoraviivaista. Haka-kirjauduttaessa käyttäjää etsitään tietokannasta myös sähköpostin perusteella. Tämän vuoksi on mahdollista, että sosiaalisen median SSO-tilillä tehdylle käyttäjätilille päästään kirjautumaan Hakan avulla.

Näitä kahdesta eri lähteestä tulevaa käyttäjätiliä ei haluta yhdistää automaattisesti. Siispä muutetaan Haka-kirjautumista siten, että mikäli kirjautumisen yhteydessä löydetään sosiaalisen median SSO-palvelulla luotu käyttäjätili, ei päästetä käyttäjää kirjautumaan sille. Sen sijaan käyttäjälle välitetään virheviesti, jossa kerrotaan sähköpostin olevan jo käytössä TIMissä.

5.9.5 Käyttäjätilien yhdistämistoiminto

Käyttäjätilien yhdistämistoiminto jää suunnitelman asteelle. Yhdistämistoiminto olisi ollut se osa SSO-integrointeja, joilla eri kirjautumistapoja olisi saatu yhdistettyä hallitusti yhdelle käyttäjätilille. Yhdistämistoiminnon toteutus SSO-integrointien nykytilassa ennen Haka-kirjautumista olisi kuitenkin ollut melko triviaali, sillä toiminnallisuus kahden tilin yhdistämiselle on TIMissä jo olemassa.

Ennen Korpin ja nyt Hakan sekä sosiaalisen median SSO-palveluiden kautta kirjauduttaessa luodaan ensimmäisellä kirjautumiskerralla käyttäjätili. Tämän jälkeen kyseiselle käyttäjätilille pääsee kirjautumaan sillä kirjautumistavalla, jolla tili luotiinkin. Tämän vuoksi päädyttiin siihen, että eri kirjautumistapojen yhdistäminen samaan tiliin tarkoittaa TIMissä kahden

käyttäjätilin yhdistämistä toisiinsa.

5.9.6 Käyttöoikeudet käyttäjätilien yhdistämisessä

TIMissä käyttöoikeudet määritetään käyttäjäryhmillä, joista voi lukea luvusta 4.5. Eri SSO-palveluita voidaan pitää eri lailla luotettavina. Kuka tahansa voi tehdä Google-tilejä mielensä mukaan, mutta jotkut SSO-palvelut voivat vaatia käyttäjän todistamaan henkilöllisyytensä tai olevan esimerkiksi korkeakouluopiskelija. Käyttäjätilejä yhdistettäessä tarvitsee siis tarkastella niitä käyttäjäryhmiä, jotka saadaan kirjautumistavan perusteella.

TIMissä kirjautumistavan perusteella saatavat oikeudet asetetaan käyttäjälle, kun hän kirjautuu ensimmäistä kertaa tietyllä kirjautumistavalla tai tiettyä identiteetintarjoajaa käyttäen. Oikeudet tallennetaan tietokantaan, joten niitä ei tulevilla kirjautumisilla lisätä uudestaan, ellei niitä välissä poisteta. Kaikki aiemmin annetut oikeudet olisivat myös voimassa kaikilla kirjautumistavoilla, mikäli niitä olisi yhdistetty useita samaan tiliin eikä kirjautumisprosesseissa poistettaisi mitään oikeuksia. Hakan kautta TIMiin luotuun tiliin on mahdollista lisätä salasana, jonka avulla samalle tilille pääsee kirjautumaan samoilla oikeuksilla ilman Haka-kirjautumista.

Kun halutaan toteuttaa mahdollisuus sekä SSO-kirjautumisten että Haka-kirjautumisen yhdistämisestä samalle käyttäjätilille, tulee SSO-kirjautumisten yhteydessä käyttäjältä poistaa käyttöoikeus `Haka users` sekä kaikki Hakan identiteetintarjoajakohtaiset oikeudet. Kaikki olemassaolevat identiteetintarjoajakohtaiset oikeudet saadaan, kun yhdistetään taulun `haka_organization` rivien `name`-sarakkeen arvojen perään välimerkki ja “users”. Koska käyttäjältä poistetaan Hakaan liittyviä käyttäjäryhmiä sosiaalisen median SSO-palveluilla kirjauduttaessa, täytyy ne myös lisätä uudestaan Haka-kirjautumisen yhteydessä. Koska näin tapahtuu TIMissä jo nyt, mitään muutoksia ei lisäyksen osalta tarvitse tehdä.

5.10 Google

5.10.1 Google-kirjautumisen kulku

Kirjautuminen Googleen tapahtuu **selainpuolella** käyttäen Googlen JavaScript SDK:ta:

1. Käyttäjä avaa kirjautumisdialogin.
2. Kirjautumisdialogia avatessa Googlen JavaScript SDK alustetaan tarvittaessa. Alustuksen jälkeen käytäjälle näytetään Sign in with Google -painike.
3. Käyttäjä painaa Sign in with Google -painiketta.
4. Käyttäjä syöttää tunnisteensa avautuvaan ponnahdusikkunaan.
5. TIM saa nyt käyttäjälle kuuluvan access tokenin käyttöönsä ja se lähetetään TIM:in palvelimelle käyttäjän sisäänkirjautumista varten.

Palvelinpuolella kirjautumisprosessi jatkuu seuraavasti:

1. Aivan ensimmäisenä tarkastellaan kirjautumispyynnöstä, onko käyttäjä kirjautunut jo aiemmin ja onko kyseessä uuden käyttäjän lisääminen istuntoon.

    1. Mikäli istuntoon yritetään lisätä käyttäjää, mutta kukaan ei ole vielä kirjautunut istuntoon (istunnossa ei ole `user_id`-tietoa) ei tehdä mitään muuta, kuin palautetaan virheilmoitus.
    2. Jos yritetään kirjautua uudestaan istunnon pääkäyttäjänä (eli käyttäjänä jonka id on istunnon `user_id`-kentän arvona), palautetaan käyttäjälle virheilmoitus.
    3. Mikäli istunnossa ei ole `adding_user`-tietoa, tallennetaan tiedoksi joko `True` tai `False` sen perusteella, pyydetäänkö kirjautumispyynnössä lisäämään käyttäjä jo kirjautuneiden käyttäjien joukkoon vai ei.
2. Validoidaan kirjautumispyynnön mukana saatu id-token lähettämällä se `validate_google_sign_in_token`-funktiolle.

    - Mikäli `validate_google_sign_in_token`-funktio palauttaa arvon `None`, lähetetään käyttäjälle virheviesti.
    - Muussa tapauksessa funktiolta saadaan `Mapping`-olio, joka sisältää käyttäjän tiedot.
3. Kutsutaan funktiota `get_user_info_from_google_id_token`, joka etsii parametrina saamastaan käyttäjätieto-oliosta käyttäjän sähköpostin, etunimen, sukunimen ja koko nimen. Jos sähköpostia ei löydetä, käyttäjänimeksi asetetaan tyhjä viittaus. Muussa tapauksessa käyttäjänimeksi tulee käyttäjän sähköposti, jonka eteen liitetään merkkijono `Google:`. Lopuksi funktio palauttaa käyttäjätiedot.

4. Käyttäjätiedot tarkistetaan kutsumalla funktiota `check_user_info`, joka keskeyttää kirjautumisen heittämällä poikkeuksen, mikäli tiedoista puuttuu sähköposti tai koko nimi. Virheviestinä lähetetään kehotus lisätä tiedot SSO-tilille ja antamaan TIM:lle näihin oikeudet.
5. Sisäänkirjattava käyttäjä saadaan pyydettyä funktiolta `find_or_create_user_from_social_login`. Parametrina välitetään `SSOAccountInfo`-olio, joka sisältää käyttäjän Google-id:n ja Googlelta aiemmin saadut käyttäjätiedot sekä tiedon siitä, että kirjautuminen tapahtuu Googlen avulla (`UserOrigin`-enumin arvo `Google`).

    - Funktio etsii tietokannan `sso_useraccount`-taulusta tietuetta Facebook-id:llä.
        - Jos tietue löytyy, sen tiedot päivitetään äskettäin saaduilla.
        - Mikäli tietue on merkitty ensisijaiseksi sen omistavalle käyttäjälle, eli `useraccount`-taulun tietueelle, päivitetään myös tämän tietueen tiedot.
    - Mikäli käyttäjää ei löydy, se luodaan kutsumalla funktiota `create_user_from_social_login`. Parametrina välitetään `UserInfo`-olio, joka sisältää käyttäjätiedot.
        - Jos käyttäjätietoihin kuuluvalla sähköpostilla on jo luotu käyttäjätili, keskeytetään käyttäjätilin luominen ja lähetetään käyttäjälle virheilmoitus. Muussa tapauksessa käyttäjätili luodaan kutsumalla `User`-luokan metodia `create_with_group`.
    - Uudelle käyttäjälle luodaan myös rivi `sso_useraccount`-tauluun kutsumalla funktiota `create_sso_useraccount`. Parametrina välitetään aiemmin parametrina saatu `SSOAccountInfo`-olio, johon täydennetään juuri luodun käyttäjätilin id-numero.
6. Poistetaan käyttäjältä kaikki Hakaan liittyvät käyttäjäryhmät kutsumalla funktiota `remove_haka_groups` parametrina käyttäjä-olio.
7. Tallennetaan tietokantaan tehdyt muutokset kutsumalla `db.session.commit()`.
8. Viimeinen kirjautumisprosessin vaihe on käyttäjän lisääminen istuntoon.

    1. Lisätään käyttäjän istuntoon tieto siitä, että tämä käyttäjätili on viimei-

sin istuntoon kirjautunut google-käyttäjä. Tämä tehdään kutsumalla funktiota `set_last_google_logged_user` käyttäen parametrina käyttäjä-oliota.

2. Lisätään käyttäjä istuntoon kutsumalla funktiota `set_user_to_session` parametrina käyttäjä-olio.
3. Viimeiseksi lähetetään käyttäjälle vastausviesti kirjautumisesta kutsumalla funktiota `finish_login`.

5.10.2 Uloskirjautuminen

Koska käyttäjän istunnot TIM:ssä ja Googlessa ovat erilliset, täytyy uloskirjautuminenkin näistä molemmista hoitaa erikseen. TIM:ssä toteutetaan asia niin, että uloskirjautumisen yhteydessä kirjataan käyttäjä ulos sovelluksesta myös Googlen osalta kutsumalla Googlen JavaScript SDK:n metodia `signOut()`. Tämä tosin ei kirjaa käyttäjää ulos Googlesta kokonaan.

Yhteen istuntoon on mahdollista lisätä useita google-käyttäjiä, kunhan kirjautumisten välillä kirjaudutaan Googlesta aina ulos. Jotta selaimen puolella osataan kirjata oikea käyttäjä ulos Googlesta, lisätään google-kirjautumisen yhteydessä istuntoon kenttä `last_google_logged_user`. Sen arvoksi asetetaan kirjautuvan käyttäjän id-numero. Tämä tieto lähetetään selaimelle kirjautumisvastauksen yhteydessä.

5.10.3 Asetukset TIMissä

Config-tiedostoon:

- `GOOGLE_APP_ID` = [App id; merkkijono]

App id ei saa päätyä versionhallintaan!

5.11 Facebook

5.11.1 Facebook-kirjautumisen kulku

Kirjautuminen Facebookiin tapahtuu **selainpuolella** käyttäen Facebookin JavaScript SDK:ta:

1. Käyttäjä avaa kirjautumisdialogin.
2. Käyttäjä painaa Login with Facebook -painiketta.
3. Käyttäjä syöttää tunnisteensa avautuvaan ponnahdusikkunaan.
4. TIM saa nyt käyttäjälle kuuluvan access tokenin käyttöönsä ja se lähetetään TIM:in palvelimelle käyttäjän sisäänkirjautumista varten.

Palvelinpuolella kirjautumisprosessin aikana voi syntyä tilanteita, jolloin koko prosessi keskeytetään tietoisesti heittämällä poikkeus. Nämä poikkeukset käsitellään erillisissä poikkeuskäsittelijöissä, jotka huolehtivat virheviestin lähettämisestä käyttäjälle. Seuraavassa esitellään Facebook-kirjautumisen kulku TIM:issä.

1. Aivan ensimmäisenä tarkastellaan kirjautumispyynnöstä, onko käyttäjä kirjautunut jo aiemmin ja onko kyseessä uuden käyttäjän lisääminen istuntoon.

    1. Mikäli istuntoon yritetään lisätä käyttäjää, mutta kukaan ei ole vielä kirjautunut istuntoon (istunnossa ei ole `user_id`-tietoa) ei tehdä mitään muuta, kuin palautetaan virheilmoitus.
    2. Jos yritetään kirjautua uudestaan istunnon pääkäyttäjänä (eli käyttäjänä jonka id on istunnon `user_id`-kentän arvona), palautetaan käyttäjälle virheilmoitus.
    3. Mikäli istunnossa ei ole `adding_user`-tietoa, tallennetaan tiedoksi joko `True` tai `False` sen perusteella, pyydetäänkö kirjautumispyynnössä lisäämään käyttäjä jo kirjautuneiden käyttäjien joukkoon vai ei.
2. Seuraavaksi tarkastellaan kirjautumispyynnön mukana toimitettua access tokenia `validate_facebook_access_token`-funktiossa:

    - Access token lähetetään Facebookin API:n `/debug_token`-päätepisteeseen validoitavaksi.

- Vastauksena saadaan tieto siitä, onko access token validi ja tarkoitettu TIMiin. Vastauksessa saadaan mukana myös käyttäjän id Facebookissa.
- Mikäli access token on validi ja tarkoitettu TIMille, palautetaan käyttäjän Facebook-id.
- Muussa tapauksessa heitetään poikkeus.

3. Käyttäjän tiedot Facebookilta `get_user_info_from_facebook`-funktiossa:

   - Käyttäjän tiedot saadaan pyytämällä niitä Facebookin API:sta käyttäjän id:llä, joka saatiin access tokenin validoinnin yhteydessä. Pyynnön mukana täytyy olla myös kyseinen access token.
   - Mikäli käyttäjätietopyyntöön saadun vastauksen statuskoodi ei ole 200, heitetään poikkeus.
   - Muussa tapauksessa otetaan vastauksesta talteen käyttäjän etunimi, sukunimi, koko nimi ja sähköpostiosoite.
   - Käyttäjän käyttäjänimeksi asetetaan käyttäjän sähköpostiosoite etuliitteen `Facebook:` kanssa. Mikäli sähköpostia ei saatu, käyttäjänimeksi tulee tyhjä merkkijono.
   - Lopuksi funktiosta palautetaan käyttäjätiedot.

4. `check_user_info`-funktiossa tarkastetaan, onko käyttäjästä saatu riittävästi tietoja.

   - Jos käyttäjältä puuttuu sähköposti tai koko nimi, heitetään tästä poikkeus virheviestillä, joka kehottaa käyttäjää lisäämään nämä SSO-tiliinsä ja antamaan TIMille käyttöoikeuden niihin.
   - Jos sähköposti ja koko nimi ovat tiedossa, funktio palaa takaisin tekemättä mitään.

5. Seuraavaksi hankitaan käyttäjälle session info access token, jonka avulla voidaan valvoa facebook-istunnon validiutta.

   - Ensin kutsutaan funktiota `get_long_term_fb_access_token`, joka pyytää Facebookin API:lta lyhytkestoisella access tokenilla pitkäkestoista access tokenia.

- Pitkäkestoista access tokenia vastaan voidaan Facebookin API:lta pyytää session info access token. Tämä tehdään kutsumalla funktiota `get_session_info_access_token`, joka palauttaa session info access tokenin.

6. Sisäänkirjattava käyttäjä saadaan pyydettyä funktiolta `find_or_create_user_from_social_login`. Parametrina välitetään `SSOAccountInfo`-olio, joka sisältää käyttäjän Facebook-id:n ja Facebookilta aiemmin saadut käyttäjätiedot sekä tiedon siitä, että kirjautuminen tapahtuu Facebookin avulla (`UserOrigin`-enumin arvo `Facebook`).

   - Funktio etsii tietokannan `sso_useraccount`-taulusta tietuetta Facebook-id:llä.
     - Jos tietue löytyy, sen tiedot päivitetään äskettäin saaduilla.
     - Mikäli tietue on merkitty ensisijaiseksi sen omistavalle käyttäjälle, eli `useraccount`-taulun tietueelle, päivitetään myös tämän tietueen tiedot.
   - Mikäli käyttäjää ei löydy, se luodaan kutsumalla funktiota `create_user_from_social_login`. Parametrina välitetään `UserInfo`-olio, joka sisältää käyttäjätiedot.
     - Jos käyttäjätietoihin kuuluvalla sähköpostilla on jo luotu käyttäjätili, keskeytetään käyttäjätilin luominen ja lähetetään käyttäjälle virheilmoitus. Muussa tapauksessa käyttäjätili luodaan kutsumalla `User`-luokan metodia `create_with_group`.
   - Uudelle käyttäjälle luodaan myös rivi `sso_useraccount`-tauluun kutsumalla funktiota `create_sso_useraccount`. Parametrina välitetään aiemmin parametrina saatu `SSOAccountInfo`-olio, johon täydennetään juuri luodun käyttäjätilin id-numero.

7. Poistetaan käyttäjältä kaikki Hakaan liittyvät käyttäjäryhmät kutsumalla funktiota `remove_haka_groups` parametrina käyttäjä-olio.
8. Tässä vaiheessa aiemmin tehdyt tietokantamuutokset tallennetaan lopullisesti kutsumalla `db.session.commit()`.
9. Viimeinen kirjautumisprosessin osuus on käyttäjän lisääminen istuntoon.

1. Kutsutaan `set_facebook_user_expiration`-funktiota parametreina käyttäjä sekä käyttäjän session info access token. Funktio lisää istuntoon käyttäjälle ajankohdan, jonka jälkeen hänen facebook-istuntonsa validius on tarkastettava. Istuntoon lisätään luonnollisesti myös käyttäjän session info access token, jonka avulla validius tarkastetaan.
2. Kutsutaan funktiota `set_last_fb_logged_user`, joka asettaa käyttäjän id-numeron istuntoon. Tämän perusteella voidaan uloskirjautuessa tarkastella selainpuolella, pitääkö kirjautua ulos myös Facebookista.
3. Kutsutaan `set_user_to_session`, joka asettaa käyttäjän istuntoon.
4. Palataan funktiosta ja kutsutaan samalla `finish_login`-funktiota, joka taas palauttaa vastauksen, joka kirjautujalle lähetetään.

5.11.2 TIM-istunnon ja Facebook-istunnon erillisyys

Käyttäjän istunto TIM:issä on erillinen käyttäjän istunnosta Facebookissa. TIM-istunto on toteutettu omalla evästeellään, kun taas Facebook-istuntoa edustaa access token. TIM:issä näiden istuntojen välille rakennetaan "keinotekoista" riippuvuutta yhdistämällä niiden uloskirjautumistoiminnot käyttöliittymässä sekä kirjaamalla käyttäjä automaattisesti ulos mikäli Facebook-istunnon havaitaan olevan epävalidi. Luvuissa 5.11.3 ja 5.11.4 käsitellään tätä riippuvuuden rakentamista istuntojen välille.

5.11.3 Uloskirjautuminen

Uloskirjautuminen toteutetaan niin, että mikäli uloskirjattava käyttäjä on kirjautunut TIM:iin Facebookin kautta, kirjataan hänet myös ulos Facebookista. Uloskirjaaminen tehdään Facebookin JavaScript SDK:n `logout`-metodilla.

`logout`-funktion kutsumisesta voi seurata erilaisia asioita, jotka riippuvat kirjautumistavasta. Mikäli käyttäjä on TIM:iin kirjautuessaan kirjautunut myös Facebookiin, `logout`-funktion kutsumisesta seuraa myös käyttäjän kirjaaminen ulos Facebookista. Jos käyttäjä on ollut TIM:iin kirjautuessaan jo valmiiksi kirjautunut Facebookiin, ei `logout`-funktion kutsuminen kirjaa häntä ulos Facebookista. Toisaalta jos käyttäjä on alun perin kirjautunut Face-

bookin jonkin toisen verkkosivun kautta, niin `logout`-funktion kutsuminen kirjaa käyttäjän ulos myös Facebookista, tapahtuipa se TIM:issä tai tuolla toisella verkkosivulla. (Facebook, n.d.[d].)

TIM:issä on mahdollista olla samassa istunnossa useita käyttäjiä ja luonnollisesti siis myös Facebook-käyttäjiä. Useiden facebook-käyttäjien lisääminen samaan istuntoon mahdollistuu, kun facebook-kirjautumisen jälkeen käydään kirjaamassa käyttäjä ulos Facebookista ennen kuin uutta facebook-käyttäjää yritetään lisätä istuntoon.

Viimeiseksi kirjautunutta facebook-käyttäjää ei haluta kirjata ulos Facebookista `logout`-funktiota kutsumalla, mikäli istunnosta ollaan uloskirjaamassa jotain toista facebook-käyttäjää tai ketä tahansa muuta käyttäjää. Näin ollen on pidettävä kirjaa siitä, mikä käyttäjätunnus on viimeisenä kirjautunut Facebookin kautta TIM:iin, jotta osataan kirjata tämä käyttäjä ulos Facebookista TIM-uloskirjautumisen yhteydessä. Tämä toteutetaan lisäämällä TIM:in istuntoon kenttä `last_fb_logged_user`, jonka arvo on aina istuntoon viimeksi Facebookin kautta kirjautuneen käyttäjän id-numero.

5.11.4 Facebook-istunnon validiuden tarkkailu

Tietyissä tilanteissa kuten salasanan vaihdossa aiemmin Facebookilta saadut access tokenit mitätöidään. Facebook ohjeistaa, että tällaisissa tilanteissa käyttäjä pitäisi kirjata ulos palvelusta. Jos palvelussa ei säilytetä access tokenia, voidaan access tokenin validiutta tarkastella access tokenin avulla luotua session info access tokenia käyttäen. Tämän tokenin avulla ei ole mahdollista päästä käsiksi käyttäjän tietoihin, joten sen käyttäminen on turvallisempaa kuin access tokenin käyttäminen. (Facebook, n.d.[l].) Käyttäjän facebook-istunnon validius on Facebookin mukaan syytä tarkistaa ainakin kerran päivässä (Facebook, n.d.[h]).

TIMissä pärjätään hyvin ilman access tokenin tallentamista, joten käyttäjän turvallisuuden vuoksi sitä ei myöskään Facebook login -integroinnissa tallenneta. Sen sijaan vaihdetaan kirjautumisen yhteydessä saatu lyhytaikainen access token ensin pitkäaikaiseen access tokeniin ja sitten pitkäaikainen access token session info access tokeniin. Session info access token tallennetaan käyttäjän istuntoon yhdessä seuraavan tarkistusajan kanssa.

Käyttäjän Facebook-istunnon validius tarkastetaan aikavälillä, joka määritellään TIMin ase-

tustiedostossa. Jokaisen TIMiin tulevan HTTP-pyynnön yhteydessä:

- Tarkastetaan, onko kyseiseen pyyntöön liittyvässä istunnossa Facebookin kautta kirjautuneita käyttäjiä:
- Mikäli on:
  - Tarkistetaan onko aika selvittää heidän Facebook-istuntojensa validius.
  - Mikäli on:
    - Lähetetään käyttäjän session info access token Facebookin tarkistettavaksi.
    - Mikäli käyttäjän istunto on validi, vaihdetaan käyttäjän istuntoon TIMin asetusten mukainen uusi istunnon tarkistusaika.
    - Mikäli facebook-istunto ei ole validi:
      - Jos käyttäjä ei ole istuntoon ensimmäisenä kirjautunut käyttäjä (TIMissä current user) poistetaan TIMin istunnosta kaikki tämän käyttäjän tiedot. Tämä vastaa käyttäjän uloskirjaamista.
      - Jos käyttäjä on current user, pyyhitään TIMin istunnosta kaikki tiedot, mikä vastaa kaikkien käyttäjien uloskirjaamista.
    - Mikäli Facebook-istunto on validi, vaihdetaan käyttäjälle uusi istunnon tarkastusaika.
  - Mikäli ei:
    - Palataan takaisin tekemättä mitään.
- Mikäli ei:
  - Palataan takaisin tekemättä mitään.

5.11.5 API-pyyntojen turvaaminen `appsecret_proof`-parametrilla

Jos Facebookin access tokenit joutuvat vääriin käsiin, voidaan niitä väärinkäyttää. On kuitenkin mahdollista ottaa Facebookissa käyttöön asetus Require App Secret, jolloin Facebookin API:lle suunnattuihin pyyntöihin tarvitaan myös sovelluksen salaisuus (app secret). `appsecret_proof` on parametri, jonka arvo on HMAC (keyed-hash message authentication code) -koodi. Se luodaan tekemällä SHA256-hajautusalgoritmilla tiiviste käyttäjän access tokenista käyttämällä avaimena sovelluksen salaisuutta. (Facebook, n.d.[k].) Funktio

`get_appsecret_proof` palauttaa tämän HMAC-koodin, kun sitä kutsutaan parametrina access token.

5.11.6 Asetukset Facebookissa

Facebook esittelee Facebook Loginia koskevassa dokumentaatiossa useita tietoturvallisuuteen liittyviä asetuksia, joita tätä palvelua käyttävä sovellus voi säätää (Facebook, n.d.[h]). Osa esitetyistä säädöistä ei ole relevantteja tai mahdollisia TIMiin tehtävän Facebook Login -integraation kannalta eri syistä, kuten esimerkiksi siksi, että TIMissä käytetään Facebookin javascript sdk:ta. Tässä luvussa esittelen miten ehdotettuja säätöjä voidaan TIMissä soveltaa. Asetukset löytyvät osoitteesta `https://developers.facebook.com/` valitsemalla valikosta MyApps | [Oma sovellus] ja sitten Products-valikosta Facebook Login | Settings

- Client Oauth Login: Kyllä (Yes)
- Enforce HTTPS: Kyllä (Yes)
- Web OAuth Login: Kyllä (Yes)
- Embedded Browser OAuth Login: Ei (No)
- Use Strict Mode for Redirect URIs: Kyllä (Yes)

Samaan tapaan valikosta MyApps | [Oma sovellus] ja sitten App Dashboardilta Settings | Advanced:

- Require App Secret: Kyllä (Yes)

5.11.7 Asetukset TIMissä

Config-tiedostoon:

- FACEBOOK_SESSION_TIME = [Haluttu aika sekunneissa; kokonaisluku.]
- FACEBOOK_API_VERSION = [Facebookin API:n versio, jota halutaan käyttää; merkkijono.]
- FACEBOOK_APP_ID = [App ID; merkkijono.]
- FACEBOOK_APP_SECRET = [App Secret; merkkijono.]

App ID ja App Secret löytyvät osoitteesta `https://developers.facebook.com/` valikosta Settings | Basic.

App ID ja App Secret eivät saa päätyä versionhallintaan!

5.11.8 Ongelmat

Kaksinkertainen kirjautuminen istuntoon lisättäessä:

1. Jollakin tunnuksella TIMiin.
2. Lisää istuntoon toinen käyttäjä Facebook-kirjautumisen kautta. Lisääminen voi tapahtua joko samalla tunnuksella tai eri tunnuksella. Eri tunnuksen saa käyttöön, jos käy välissä Facebookissa kirjautumassa ulos.
3. Tee askel 2. vielä kerran.

Facebookin SDK varoittaa:

```
sdk.js?hash=0b4f59cb3f3bd761d7ac92b659ea6512&ua=modern_es6:52 You
are overriding current access token, that means some other app is expecting
different access token and you will probably break things. Please consider
passing access_token directly to API parameters instead of overriding the global
settings
```

Mikäli istuntoon lisäämisen jälkeen sivu ladataan uudestaan, ei tätä varoitusta nähdä seuraavan Facebook-kirjautumisen yhteydessä. Ongelma liittyy mahdollisesti jotenkin Facebookin SDK:n lataamiseen ja alustamiseen, joka tehdään aina sivulatauksen yhteydessä. Tämä ei kuitenkaan näytä vaikuttavan mitenkään kirjautumisen oikeaan toteutumiseen TIMissä.

5.12 Twitter

5.12.1 Twitter-kirjasto

Twitterin dokumentaatiossa luetellaan useampia epävirallisia Python-kirjastoja, joita voi käyttää apuna Log in with Twitter -integraation toteuttamisessa (Twitter, n.d.[d]). Kun tarkastellaan kirjastojen dokumentointia, koodia ja suosittuutta tullaan tulokseen, että Twython-

kirjasto soveltuu hyvin SSO-integraation toteuttamiseen. Siispä valitaan Twython toteutuksen avuksi.

5.12.2 Twitter-kirjautumisen kulku

1. Käyttäjä painaa Log in with Twitter -painiketta, joka on linkki palvelimen reittiin `/twitterLogin`.
2. Reitistä vastaavassa funktiossa `login_with_twitter` tarkastellaan aivan ensimmäisenä kirjautumispyynnöstä, onko käyttäjä kirjautunut jo aiemmin ja onko kyseessä uuden käyttäjän lisääminen istuntoon.

    1. Mikäli istuntoon yritetään lisätä käyttäjää, mutta kukaan ei ole vielä kirjautunut istuntoon (istunnossa ei ole `user_id`-tietoa) ei tehdä mitään muuta, kuin palautetaan virheilmoitus.
    2. Jos yritetään kirjautua uudestaan istunnon pääkäyttäjänä (eli käyttäjänä jonka id on istunnon `user_id`-kentän arvona), palautetaan käyttäjälle virheilmoitus.
    3. Mikäli istunnossa ei ole `adding_user`-tietoa, tallennetaan tiedoksi joko `True` tai `False` sen perusteella, pyydetäänkö kirjautumispyynnössä lisäämään käyttäjä jo kirjautuneiden käyttäjien joukkoon vai ei.
3. Luodaan `twython`-kirjaston `Twython`-olio kirjautumista varten. Parametreiksi tarvitaan Twitterin API:n avain ja salainen avain, jotka saadaan selville TIMiä varten luodun Twitter-sovelluksen tiedoista. Lisäksi välitetään parametrina tieto siitä, että kirjautumisessa käytetään `authenticate`-endpointia. Tämä on `[juuri]/twitterLogin/callback`.

    - Kutsutaan luodun `Twython`-olion metodia `get_authentication_tokens`. Parametrina välitetään `callback_url`, johon Twitterin halutaan ohjaavan kirjautuneen käyttäjän.
        - Vastauksena saadaan Twitteriltä `oauth_token` ja `oauth_token_secret`, jotka tallennetaan istuntoon, sekä URL-osoite käyttäjän kirjautumista varten.
    - Lopuksi uudelleenohjataan käyttäjä kirjautumaan saatuun osoitteeseen.

4. Käyttäjä kirjautuu Twitteriin, minkä jälkeen Twitter uudelleenohjaa käyttäjän saamaansa uudelleenohjausosoitteeseen.
5. Kirjautumisprosessi jatkuu uudelleenohjausreitistä vastaavassa funktiossa `after_login_with_twitter`.

    - Luodaan taas twythonin `Twython`-olio. Parametreiksi tarvitaan Twitterin API:n avain ja salainen avain sekä edellisessä vaiheessa istuntoon tallennetut `oauth_token` sekä `oauth_token_secret`.
    - Seuraavaksi kutsutaan `Twython`-olion metodia `get_authorized_tokens`. Parametriksi tarvitaan `oauth_verifier`, jonka Twitter on välittänyt TIMille URL-parametrina.
    - Vastauksena saadaan `oauth_token` ja `oauth_token_secret`, joiden avulla voidaan tehdä autentikoituja pyyntöjä Twitterille.
6. Pyydetään käyttäjän tiedot Twitteriltä kutsumalla `get_twitter_user_info` -funktiota, jolle välitetään kohdassa 5 saadut `oauth_token` ja `oauth_token_secret`.

    - Funktio pyytää Twitteriltä käyttäjän tietoja ja palauttaa käyttäjän Twitter-id:n, koko nimen, sähköpostin ja käyttäjänimen. Käyttäjänimi on käyttäjän sähköpostiosoite, johon on lisätty etuliite `Twitter:`. Mikäli sähköpostia ei saada Twitteriltä, palautetaan käyttäjänimenä tyhjä viittaus.
7. `check_user_info`-funktiossa tarkastetaan, onko käyttäjästä saatu riittävästi tietoja.

    - Jos käyttäjältä puuttuu sähköposti tai koko nimi, heitetään tästä poikkeus virheviestillä, joka kehottaa käyttäjää lisäämään nämä SSO-tiliinsä ja antamaan TIMille käyttöoikeuden niihin.
    - Jos sähköposti ja koko nimi ovat tiedossa, funktio palaa takaisin tekemättä mitään.
8. Sisäänkirjattava käyttäjä saadaan pyydettyä funktiolta `find_or_create_user_from_social_login`. Parametrina välitetään `SSOAccountInfo`-olio, joka sisältää käyttäjän Twitter-id:n ja muut Twitteriltä

aiemmin saadut käyttäjätiedot sekä tiedon siitä, että kirjautuminen tapahtuu Twitterin avulla (`UserOrigin`-enumin arvo `Twitter`).

- Funktio etsii tietokannan `sso_useraccount`-taulusta tietuetta Twitter-id:llä.
  - Jos tietue löytyy, sen tiedot päivitetään äskettäin saaduilla.
  - Mikäli tietue on merkitty ensisijaiseksi sen omistavalle käyttäjälle, eli `useraccount`-taulun tietueelle, päivitetään myös tämän tietueen tiedot.
- Mikäli käyttäjää ei löydy, se luodaan kutsumalla funktiota `create_user_from_social_login`. Parametrina välitetään `UserInfo`-olio, joka sisältää käyttäjätiedot.
  - Jos käyttäjätietoihin kuuluvalla sähköpostilla on jo luotu käyttäjätili, keskeytetään käyttäjätilin luominen ja lähetetään käyttäjälle virheilmoitus. Muussa tapauksessa käyttäjätili luodaan kutsumalla `User`-luokan metodia `create_with_group`.
- Uudelle käyttäjälle luodaan myös rivi `sso_useraccount`-tauluun kutsumalla funktiota `create_sso_useraccount`. Parametrina välitetään aiemmin parametrina saatu `SSOAccountInfo`-olio, johon täydennetään juuri luodun käyttäjätilin id-numero.

9. Poistetaan käyttäjältä kaikki Hakaan liittyvät käyttäjäryhmät kutsumalla funktiota `remove_haka_groups` parametrina käyttäjä-olio.
10. Tässä vaiheessa aiemmin tehdyt tietokantamuutokset tallennetaan lopullisesti kutsumalla `db.session.commit()`.
11. Viimeinen kirjautumisprosessin osuus on käyttäjän lisääminen istuntoon.

    - Kutsutaan `set_user_to_session` parametrina käyttäjä-olio.
    - Kutsutaan `finish_login`.

5.12.3 Asetukset Twitterissä

Twitter-kirjautumisen käyttöönotto vaatii joidenkin asetusten tekemistä sovelluksen Twitter-sivulla. Sovelluksen asetuksia pääsee tekemään osoitteessa `https://developer.twitter.com`, kunhan kirjautuu sisään ja valitsee profiilin nimestä avautuvasta valikos-

ta kohdan Apps. Sovelluksen asetukset löytyvät kohdista App details ja Permissions. Seuraavaksi tarvittavat asetukset:

App details:

- App name: Sovelluksen nimi
- Application description: Sovelluksen kuvaus
- Website URL: Verkkosivun osoite
- Sign in with Twitter: Enabled
- Callback URL: Osoite, johon Twitter ohjaa kirjautuneen käyttäjän. Tämä on siis sama uudelleenohjausosoite, joka mainitaan Twitter-kirjautumisen kulun vaiheessa 3.
- Terms of service URL: Sovelluksen käyttöehtojen URL
- Privacy policy URL: Sovelluksen yksityisyyskäytäntöjen URL
- Tell us how this app will be used: Lyhyt kuvaus sovelluksen käytöstä.

Permissions:

- Access permission: Read-only
- Request email address from users: Tämä asetus pitää klikata valituksi.

5.12.4 Asetukset TIMissä

Config-tiedostoon:

- TWITTER_API_KEY = [API key; merkkijono.]
- TWITTER_API_SECRET_KEY = [API secret key; merkkijono.]

API key ja API secret key eivät saa päätyä versionhallintaan!

5.13 Toteutusten vertailu

Sosiaalisen median kirjautumisprosesseissa on paljon yhteistä. Kirjautumistapojen tietyt eroavaisuudet johtivat aluksi siihen, että niiden käytännön toteutukset lähtivät eroamaan toisistaan tarpeettoman paljon. Vasta kolmannen kirjautumistoteutuksen kohdalla kirjautumis-

ten perustavanlaatuinen yhteneväisyys kävi ilmeiseksi ja samalla aiemmin tehtyjen toteutusten refaktoroiminen tarpeelliseksi.

Prosessien yhteiset vaiheet ja eroavaisuudet on hyvä tiedostaa, jotta saadaan luotua kirjautumisprosesseista ohjelmakoodinkin tasolla mahdollisimman yhdenmukaiset ja helposti ylläpidettävät. Samalla helpotetaan myös uusien kirjautumisvaihtoehtojen lisäämistä TIMiin. Seuraavaksi eritellään kirjautumisprosessien osat fonteilla koodattuina:

- Normaali fontti: Vaihe on kaikille kirjautumistavoille yhteinen. Se voidaan toteuttaa samalla ohjelmakoodilla.
- **Lihavoitu fontti**: Vaiheessa on eroja kirjautumistapojen kesken. Se on toteutettava kullekin vaiheelle erikseen.
- *Kursivoitu fontti*: Vaihe on uniikki jollekin kirjautumistavalle, eikä sen toteuttaminen ole relevanttia muiden tapojen kannalta.

0. **Kirjautuminen SSO-palveluntarjoajalle. Googlen ja Facebookin osalta tämä tapahtuu kokonaan selainpuolella, jonka jälkeen kirjautumisesta saatu token lähetetään TIMin palvelimelle. Twitterin osalta kirjautuminen kulkee koko ajan TIMin palvelimen kautta ja sen vuoksi tämän listauksen vaihe 1 toteutetaan Twitterin osalta ennen vaihetta 0.**
1. Uuden käyttäjän istuntoon lisäämisen hallinta sekä jo kirjautuneen käyttäjän uudelleenkirjautumisen estäminen. Twitterkirjautumisen vaihe 2, Facebook-kirjautumisen palvelinpuolen vaihe 1 ja Google-kirjautumisen palvelinpuolen vaihe 1.
2. **Kirjautuvan käyttäjän validointi. Googlen ja Facebookin tapaukset ovat verrattain samankaltaiset. Niissä selainpuolella saatu id-token tai access token validoidaan. Googlen sekä Facebookin osalta tämä tapahtuu kirjautumisprosessin palvelinpuolen vaiheissa 2. Twitterin osalta kirjautumisprosessi kulkee koko ajan TIMin palvelimen kautta ja siten aivan vastaavaa kirjautuvan käyttäjän validointia ei ole. Validoinnin voisi silti katsoa tapahtuvan vaiheessa 5.**
3. **Käyttäjätietojen selvittäminen SSO-palveluntarjoajalta. Facebook- ja Twitter-toteutuksissa käyttäjätiedot on kysyttävä erikseen SSO-palveluntarjoajalta, kun taas Google-kirjautumisen yhteydessä saatava ID-token sisältää jo tarvittavat käyttäjätiedot. Facebookin osalta tähän vaiheeseen liittyy esimerkiksi omia vies-**

tinnän turvallisuutta lisääviä toimia. Google- ja Facebook-kirjautumisten palvelinpuolen vaiheet 3 ja Twitter-kirjautumisen vaihe 6.

4. SSO-palveluntarjoajalta saatujen käyttäjätietojen tarkistaminen. Tämä toteutetaan funktiossa `check_user_info`. Google-kirjautumisen palvelinpuolen vaihe 4, Facebook-kirjautumisen palvelinpuolen vaihe 4, Twitter-kirjautumisen vaihe 7.
5. *Istunnon validiuden valvontaan tarkoitetut lisätoimenpiteet. Facebook-kirjautumisessa käyttäjälle hankitaan session info access token, jonka avulla ajoittain tarkistetaan, että käyttäjän Facebook-istunto on validi.*
6. Käyttäjän pyytäminen tietokannasta ja sen tietojen päivittäminen, tai uuden käyttäjän luominen. Tämä tapahtuu kutsumalla funktiota `find_or_create_user_from_social_login` parametrina aiemmin SSO-palveluntarjoajalta saadut käyttäjätiedot. Google-kirjautumisen palvelinpuolen vaihe 5, Facebook-kirjautumisen palvelinpuolen vaihe 6, Twitter-kirjautumisen vaihe 8.
7. Poistetaan käyttäjältä kaikki Hakaan liittyvät käyttäjäryhmät kutsumalla funktiota `remove_haka_groups` parametrina käyttäjä-olio.
8. Heti käyttäjä-olion pyytämisen jälkeen tallennetaan käyttäjän tietoihin mahdollisesti tehdyt muutokset tietokantaan kutsumalla `db.session.commit()`. Google-kirjautumisen palvelinpuolen vaihe 7, Facebook-kirjautumisen palvelinpuolen vaihe 8, Twitter-kirjautumisen vaihe 10.
9. **Käyttäjän lisääminen istuntoon ja kirjautumisen päättäminen. Google-kirjautumisen palvelinpuolen vaihe 8, Facebook-kirjautumisen palvelinpuolen vaihe 9, Twitter-kirjautumisen vaihe 11.**

    - **Google- ja Facebook-kirjautumisessa istuntoon tallennetaan tieto, että nykyinen kirjautuja on viimeisin kyseisestä palvelusta tähän istuntoon kirjautunut käyttäjä. Tämä tehdään, että selainpuolella osataan tarvittaessa käyttää JavaScript-kirjastoa käyttäjän uloskirjaamiseen. Facebook-kirjautumisen osalta tämä saattaa kirjata käyttäjän ulos myös Facebookista.**
    - *Facebook-kirjautumisessa istuntoon lisätään myös tieto, milloin käyttäjän istunnon validiutta on tarkasteltava uudelleen. Samoin istuntoon lisätään session info access token, jonka avulla validius voidaan tarkistaa.*

- Kaikissa kirjautumistavoissa käyttäjä lisätään istuntoon kutsumalla `set_user_to_session`.
- Kaikissa kirjautumistavoissa kirjautuminen päätetään kutsumalla `finish_login`.

5.14 Ongelmat SSO-integrointien käyttöönotossa

SSO-integrointien toteuttamisen jälkeen käytiin vielä uudestaan läpi Twitterin, Googlen ja Facebookin käyttöehtoja ja muita relevantteja dokumentteja. Tällöin huomattiin joitakin ongelmia tai mahdollisia ongelmia, joita TIMissä tulisi huomioida ennen kuin SSO-integrointeja voitaisiin ottaa käyttöön tuotantoympäristössä. Lisäselvityksiä ja mahdollisia muutoksia TIMiin ei kuitenkaan lähdetty enää tekemään aikataulusyistä ja siksi, ettei niiden katsottu olevan tutkielman tulosten kannalta kovinkaan oleellisia. Tuloksena esiteltävän prosessimallin ei ole mielekästä sisältää yksityiskohtaisia ohjeita jonkin tietyn SSO-palvelun kaikkien käyttöehtojen toteuttamiseksi.

Vielä toteutuksen viime vaiheessa esiin nousseet ongelmat SSO-integrointien käyttöönoton tiellä antavat kuitenkin kokonaisuudessaan hyvän lisän prosessimalliin. On oletusarvoista, että toteutuksen kuluessa kasvaa tietoisuus ohjelmistosta, johon SSO-integrointeja ollaan tekemässä. Siten on hyvin mahdollista, että aiemmin huomaamatta jääneet toimenpiteitä vaativat asiat tulevat esiin, kun SSO-palveluntarjoajien käyttöehtoja ja muita vastaavia dokumentteja käydään läpi SSO-integrointien ollessa ainakin näennäisesti jo valmiita. Lisäksi on tietenkin hyvin mahdollista, että toimenpiteisiin johtavia vaatimuksia jää muuten vain huomaamatta. Näin ollen on syytä jättää SSO-integrointien loppuvaiheeseen aikaa mainittujen dokumenttien läpikäyntiin sekä mahdollisesti löytyvien ohjelmiston puutteiden korjaamiseen.

5.14.1 Esimerkkejä ongelmista

1. Google vaatii, että sen API:n kautta saatua ei-julkista tietoa ei saa näyttää muille käyttäjille ilman käyttäjän eksplisiittistä suostumista (Google 2019a). SSO-integroinnissa käyttäjänimenä käytetään käyttäjän sähköpostiosoitetta, joka tietyissä tilanteissa saattisi näkyä joillekin käyttäjille.

2. Google vaatii, että kaikkien sen API:n kautta saatujen tietojen olisi myös oltava käyttäjien vietävissä käytettäväksi muissa palveluissa. Viemisen olisi oltava yhtä nopeaa ja helppoa kuin vastaavan datan vieminen Googlen tuotteista ja palveluista. (Google 2019a) TIMissä käyttäjä ei pääse näkemään tai lataamaan näitä tietoja tällä hetkellä.
3. Sekä Facebook että Twitter vaativat niiden kautta saatujen tietojen poistamista tietyissä tilanteissa, esimerkiksi käyttäjän sitä pyytäessä (Twitter 2020; Facebook, n.d.[f]). TIMissä ei kuitenkaan ole toteutettu esimerkiksi käyttäjätilin poistamismahdollisuutta.
4. Facebook muistuttaa, että käytettäessä teknologioita, jotka mahdollistavat Facebookin tiedonkeruun käyttäjistä, tulee käyttäjiltä kysyä ensin lupa. Käyttäjiä tulee informoida evästeiden ja muiden vastaavien teknologioiden käytöstä. (Facebook, n.d.[f]) TIMissä käytetään Facebookin JavaScript SDK:ta, joten käyttäjien hyväksynnän kysyminen tulisi tehdä.

6 Tulokset

6.1 Prosessi SSO-ratkaisujen integroimiseksi käytössä olevaan järjestelmään

Tässä luvussa esitellään prosessi, jolla SSO-järjestelmän integroiminen käytössä olevaan tietojärjestelmään voidaan toteuttaa. Luvut 6.1.1-6.1.6 ovat suoritusjärjestyksessä, eli jokaisen luvun määrittämät toimenpiteet on tarkoitus suorittaa samassa järjestyksessä kuin alaluvut on esitetty. Alalukujen sisällä esitetyt toimenpiteet on tarkoitettu suoritettaviksi numerojärjestyksessä, sikäli kuin ne on numerojärjestyksessä esitetty. Luvun 6.1.12 kuvaama vaihe on tarkoitettu toteutettavaksi siinä vaiheessa, kun SSO-integroinnin katsotaan olevan valmis.

6.1.1 Kirjautumisjärjestelmän toiminnan selvittäminen

1. Otetaan selville missä tiedostoissa ja tietokantatauluissa käyttäjätilin luomisen ja kirjautumisen toiminnallisuus sijaitsee.
2. Otetaan selville missä funktioissa käyttäjätilin luominen ja kirjautuminen käytännössä toteutetaan.
3. Muunnetaan kirjautuminen helposti ymmärrettävään muotoon, esimerkiksi tehdään ihmisluettava listaus.
    - Esityksestä tulee käydä ilmi, mitkä ovat kirjautumistavasta riippumatta tilin luomiseen ja kirjautumiseen tarvittavat toimenpiteet, ja mikä niiden suoritusjärjestys on.
    - Esitetään samalla oikeissa yhteyksissään ne funktiot, joissa toteutetaan kirjautumistavasta riippumattomia välttämättömiä kirjautumistoimenpiteitä.

6.1.2 Käyttäjätunnuksen luomiseen tarvittavien tietojen selvittäminen

Käyttäjätunnuksen luomiseen tarvittavat tiedot voi pitkälti johtaa luvussa 6.1.1 tehdystä selvitystyöstä. Erikseen tulee kuitenkin selvittää, mitkä tiedoista ovat välttämättömiä. Esimerkiksi TIM-järjestelmässä tietokannasta löytyvän käyttäjätaulun rakenteesta ei voinut vielä

päätellä mitä kaikkia tietoja todellisuudessa tarvitaan. Taulussa ei oltu määritelty pakollisiksi (so. NOT NULL -rajoitetuiksi) kaikkia tosiasiassa vaadittavia tietoja. Selvitys välttämättömistä tiedoista tehdään järjestelmän dokumentaatiosta ja sen puuttuessa järjestelmän ylläpitäjiltä kysymällä. Lisäksi SSO-järjestelmän dokumentaatiosta selvitetään, minkä tiedon perusteella käyttäjä tunnistetaan. Kyseessä voi olla esimerkiksi numerosarja.

6.1.3 SSO-palvelujen tarjoamien tietojen selvittäminen

Tässä vaiheessa tehdään käyttäjätietoselvitys, jossa selvitetään pystyykö SSO-palvelu tarjomaan ne tiedot, joita käyttäjätunnuksen rekisteröintiin tarvitaan. Selvityksestä tehdään taulukointi, jossa luetellaan kaikki käyttäjätunnuksen rekisteröimiseen vaadittavat ominaisuudet niiden saatavuustietojen ja muiden ominaisuuksien kanssa. Näitä ovat:

- Saatavuus:
  - Saadaan aina: Tieto on aina saatavissa.
  - Mahdollinen: Tieto voidaan saada, mutta se ei ole välttämättä saatavissa.
  - Ei saada: SSO-palvelu ei tarjoa kyseistä tietoa ollenkaan.
  - Epävarma: Saatavuudesta ei ole löytynyt varmaa tietoa.
- Muuttuvuus:
  - Ei muutu: Tieto pysyy aina samana.
  - Voi muuttua: Tieto saattaa muuttua.
  - Epävarma: Muuttuvuudesta ei ole löytynyt varmaa tietoa.
- Ehdollisuus:
  - Suoraan saatavilla: Tiedon saamiseksi ei tarvita minkäänlaista lupamenettelyä.
  - Käyttölupa haettava: Tietoa ei saada SSO-palvelulta ennen kuin siihen on erikseen saatu lupa.
  - Epävarma: Ehdollisuudesta ei ole löytynyt varmaa tietoa.
- Lisätiedot: Lisätiedoissa voi esimerkiksi arvioida mahdollisesti saatavissa olevan tiedon saamisen todennäköisyyttä, kertoa miksi jokin tiedon ominaisuus on epävarma ja kertoa millainen lupamenettely tiedon saamiseksi tarvitaan.

Taulukko 4: Esimerkki käyttäjätietojen saatavuus- ja ominaisuusselvityksen raportoinnista.

Tieto	Saatavuus	Muuttuvuus	Ehdollisuus	Lisätiedot
Nimi	Mahdollinen	Voi muuttua	Suoraan saatavilla	Ei välttämättä oikea.
Käyttäjänimi	Saadaan aina	Ei muutu	Suoraan saatavilla	
Sähköposti	Saadaan aina	Epävarma	Suoraan saatavilla	
Osoite	Ei saada			
Puhelinnumero	Mahdollinen	Voi muuttua	Käyttölupa haettava	Voi olla myös useita.

6.1.4 SSO-palvelun käyttöönottomahdollisuuden arviointi käyttäjätietoselvityksen pohjalta

Kun käyttäjätietoselvitys on tehty, voidaan arvioida SSO-palvelun käyttöönoton mahdollisuuksia käyttäjätietojen suhteen.

1. Mikäli kaikkien käyttäjätietojen saatavuudelle on saatu arvoksi “saadaan aina”, voidaan SSO-palvelu ottaa käyttöön.
2. Mikäli jollekin käyttäjätiedolle on saatu saatavuuden arvoksi “ei saada”, voidaan SSO-palvelu ottaa käyttöön, jos
    a. tieto voidaan johtaa muista saaduista käyttäjätiedoista (esimerkiksi käyttäjänimen automaattigenerointi käyttäjän nimestä),
    b. tieto voidaan korvata käyttäjästä riippumattomalla tiedolla (esimerkiksi käyttäjänimen satunnaisgeneroiminen jollain algoritmilla) tai
    c. tieto voidaan kysyä erikseen käyttäjältä.
3. Mikäli jonkin käyttäjätiedon saatavuuden arvoksi on saatu “mahdollinen”, voidaan SSO-palvelu ottaa käyttöön, mutta puuttuviin käyttäjätietoihin tulee varautua joko
    a. pyytämällä käyttäjää kirjautumista yritettäessä täydentämään puuttuvat käyttäjätietonsa SSO-palveluun että kirjautuminen mahdollistuu,
    b. johtamalla tieto muista saaduista käyttäjätiedoista (esimerkiksi käyttäjänimen automaattigenerointi käyttäjän nimestä),

c. korvaamalla tieto käyttäjästä riippumattomalla tiedolla (esimerkiksi käyttäjänimen satunnaisgeneroiminen jollain algoritmilla) tai
d. kysymällä tietoa käyttäjältä erikseen kirjautumisen yhteydessä.

4. Mikäli jonkin käyttäjätiedon saatavuuden arvo on “epävarma”:
   a. Jos kyseessä on tilanne, jossa tiedetään että käyttäjätieto on ainakin mahdollinen, mutta ei ole varmaa onko se aina saatavilla, voidaan tiedon puuttumiseen varautua kohdan 3. toimenpiteillä.
   b. Mikäli ei ole varmuutta voiko tietoa saada ollenkaan, pitää asiasta tehdä lisäselvityksiä tai harkita kohdan 2. toimenpiteitä.

6.1.5 Ehdollisuuden aiheuttamien toimenpiteiden arviointi

Mikäli jonkin käyttäjätiedon ehdollisuuden arvoksi on saatu “käyttölupa haettava”:

1. Otetaan selville miten kyseiset ehdot täytetään.
2. Päätetään, ovatko ehdot täytettävissä.
   a. Mikäli ovat, voidaan SSO-palvelu ottaa käyttöön.
   b. Mikäli ehtoja ei voida tai haluta täyttää, on harkittava toimenpiteitä puuttuvan tiedon korvaamiseksi. Vaihtoehtoja tähän ovat ainakin luvun 6.1.4 toimenpiteet 2a.-2c.

6.1.6 Käyttäjätietojen muuttumiseen varautuminen

Käyttäjätiedot on oleellista pitää ajan tasalla hyvän käyttökokemuksen ja järjestelmän oikeellisen toiminnan varmistamiseksi. Erityisesti tulee miettiä millä käyttäjätiedoilla on merkittäviä vaikutuksia järjestelmän toiminnallisuuksien oikeellisuuteen. Jos käyttäjille esimerkiksi lähetetään sähköpostiviestejä, on oleellista, että sähköposti on oikea.

1. Jos käyttäjätiedon muuttuvuuden arvo on “ei muutu”, ei sen muuttumiseen tietenkään tarvitse varautua.
2. Jos käyttäjätiedon muuttuvuuden arvo on “voi muuttua”, tulee käyttäjätieto päivittää aina käyttäjän kirjautuessa, jotta SSO-palveluun mahdollisesti tulleet käyttäjätietomuutokset saadaan päivitettyä järjestelmään.

- Mikäli käyttäjätiedon saatavuuden arvoksi on saatu “mahdollinen” tai “epävarma”, saattaa käyttäjätiedon muutos tarkoittaa myös tiedon poistumista kokonaan. Tällöin on päätettävä miten tiedon poistumiseen reagoidaan. Vaihtoehtoja ovat luvun 6.1.4toimenpiteet 3a. - 3d.

3. Jos käyttäjätiedon muuttuvuuden arvo on “epävarma”, voidaan tehdä asiasta lisäselvityksiä tai varautua muutoksiin kohdassa 2. mainituin toimenpitein.

6.1.7 SSO-palvelun käyttöehtojen selvittäminen ja arviointi

Ennen kuin SSO-integrointia kannattaa alkaa suunnittelemaan, tulee ottaa selville ne vähimmäisehdot, joita SSO-palvelun käyttöönotto edellyttää. Palvelu saattaa esimerkiksi vaatia, että käyttäjän tulee pystyä poistamaan SSO-palvelun kautta välitetyt käyttäjätiedot palvelua käyttävästä tietojärjestelmästä. Ehdot on hyvä kirjoittaa ylös ja niihin mukautumiseen tulee integrointia suunniteltaessa varautua.

6.1.8 Usean SSO-tilin yhdistäminen paikalliseen käyttäjätiliin: käyttäjätiedot

SSO-tilin tunnistamiseen ja yhdistämiseen paikalliseen käyttäjätiliin vaaditaan periaatteessa vain SSO-tilin omistavan käyttäjätilin tunniste, SSO-palvelun tunniste, sekä käyttäjän tunniste kyseisen SSO-palvelun sisällä. Nämä voidaan tallentaa omaan SSO-tileistä vastaavaan tietokantatauluun (`sso_useraccount`-tauluun). Muut käyttäjätiedot kuten käyttäjän nimi ja sähköpostiosoite ovat nopealla tarkasteltuja redundantteja, sillä samat käyttäjätiedot tallennetaan myös käyttäjätileistä vastaavaan tietokantatauluun (`useraccount`-tauluun). Näitäkin tietoja kuitenkin tallennetaan tarkoituksena ratkaista käyttäjätietojen ristiriitoja eri SSO-palveluiden välillä. Aihetta tarkastellaan luvussa 5.9.3. Seuraavaksi on listattu esitys käyttäjätietojen hallinnasta silloin, kun samaan käyttäjätiliin yhdistetään useita SSO-tilejä.

1. Lisätään käyttäjätiedot soveltuvilta osin myös `sso_useraccount`-tauluun.
   - Epäsoveltuvia tietoja ovat luonnollisesti sellaiset paikalliseen tiliin liittyvät tiedot, joita ei saada SSO-palveluilta.
2. Lisätään `sso_useraccount`-tauluun sarake, jolla merkitään, onko SSO-tili ensisijainen kirjautumistapa sen omistavalle käyttäjätilille.

- Tämä voi olla yhden suhde yhteen -viittaus `useraccount`-taulun tietueeseen.
- Kun SSO-tilillä kirjaudutaan ensimmäisen kerran, asetetaan se luotavalle käyttäjätilille ensisijaiseksi kirjautumistavaksi.

3. SSO-tilillä kirjautumisen yhteydessä tulee käyttäjätiedot päivittää, kuten luku 6.1.6 esittää.
    - Käyttäjätiedot päivitetään aina `sso_useraccount`-tauluun.
    - Käyttäjätiedot päivitetään myös `useraccount`-tauluun silloin, kun kirjautumiseen käytetty SSO-tili on merkitty ensisijaiseksi kirjautumistavaksi käyttäjätilille.

6.1.9 Usean SSO-tilin yhdistäminen paikalliseen käyttäjätiliin: käyttöoikeudet

Käyttöoikeuksien problematiikkaa käyttäjätilien yhdistämisen yhteydessä pohditaan luvussa 5.9.6. Eri kirjautumistapojen perusteella saatuja käyttäjäoikeuksia voidaan selvittää luvussa 6.1.1 kuvaillun selvituksen perusteella. Näin ei kuitenkaan välttämättä saada tietoon kaikkia relevantteja käyttöoikeuksia.

Voisi esimerkiksi olla, että tähän mennessä kaikkia järjestelmän kirjautumistapoja ollaan kohdeltu tasa-arvoisina ja nyt lisättävät uudet kirjautumistavat katsotaan eriarvoisiksi. Tällaisessa tilanteessa koko käyttöoikeuksien hallinta täytyisi tietysti rakentaa alusta saakka. Joka tapauksessa tulee selvittää, missä vaiheissa käyttöoikeuksia annetaan, että useiden kirjautumistapojen yhdistämistä varten tarvittavat muutokset voidaan tehdä.

- Jos järjestelmässä ei ole tarkoitus erotella käyttäjien käyttöoikeuksia heidän kirjautumistapansa perusteella, mitään ei tarvitse käyttöoikeuksien osalta tehdä.
- Jos eri käyttöoikeuksia on tarkoitus antaa kirjautumistapojen perusteella, tulee käyttäjän oikeuksia muuttaa aina kirjautumisen yhteydessä. Tarvittavat muutokset tulee muistaa tehdä myös vanhoihin kirjautumistapohin.
    1. Käyttäjälle tulee antaa kirjautumisen yhteydessä ne käyttöoikeudet, joihin kyseinen kirjautumistapa oikeuttaa.
    2. Käyttäjältä tulee aina kirjautumisen yhteydessä poistaa kaikki ne käyttöoikeudet, joihin tämä kirjautumistapa ei häntä oikeuta.

6.1.10 Usean SSO-tilin yhdistäminen paikalliseen käyttäjätiliin: yhdistämistoiminto ja ensisijaisen kirjautumistavan valinta

Yhdistämistoimintoa ei toteutettu TIMiin, mutta sitä kuitenkin suunniteltiin. Seuraavaksi esitellään niitä perusperiaatteita joihin päädyttiin.

1. Käyttäjän tulee päästä kirjautumaan toiminnossa kahdelle käyttäjätilille samanaikaisesti, jotta hän todistaa omistavansa molemmat käyttäjätilit.
2. Käyttäjän tulee valita, kumpi käyttäjätileistä on hänelle ensisijainen käyttäjätili.
3. Käyttäjätili yhdistetään, kun käyttäjä hyväksyy yhdistämisen.
    - Yhdistämisessä ensisijaiselle tilille periytetään kaikki toissijaisen käyttäjätilin tiedot ja käyttöoikeudet.
    - Ensisijaisen tilin alkuperäinen kirjautumistapa merkitään ensisijaiseksi kirjautumistavaksi.
    - Ensisijaisen tilin käyttäjätiedot (etunimi, sukunimi, sähköposti jne.) pidetään voimassa.
    - Toissijaisen tilin kirjautumistavat muutetaan viittaamaan ensisijaiseen käyttäjätiliin.
    - Toissijainen käyttäjätili poistetaan.

Kun käyttäjätiliin on yhdistetty useampia kirjautumistapoja, käyttäjälle pitää antaa mahdollisuus myös muuttaa ensisijaista kirjautumistapaansa. Tulisi luoda toiminto, jossa käyttäjän kirjautumistavat listataan ja annetaan mahdollisuus valita ensisijainen käyttäjätili. Valittu kirjautumistapa merkitään tietokannassa ensisijaiseksi ja kyseiseen kirjautumistapaan liittyvät käyttäjätiedot päivitetään käyttäjätilille. Tämä onnistuu kun käyttäjätiedot tallennetaan kirjautumistapakohtaisesti luvun 5.9.3 mukaisesti.

6.1.11 Käytännön toteutuksen suunnittelu

Usean kirjautumistavan käyttöönotto sujuu helpoiten, kun tunnistetaan, mitä vaiheita kirjautumisprosessiin kuuluu. Lisäksi halutaan tunnistaa, mitkä kaikki asiat voidaan todennäköisesti toteuttaa eri kirjautumistavoille samalla ohjelmakoodilla ja mitkä asiat pitää toteuttaa kirjautumistavoille erikseen. Näin saadaan kirjoitettua mahdollisimman vähän koodia. TI-

Miin tehtyjen SSO-integraatioiden toteutuksia vertaillaan luvussa 5.13. Luvussa 5.9.3 taas esitellään SSO-tilien tietokantaan tallentamiseen tarkoitettu tietokantataulu.

1. Kun käytettävät SSO-palvelut ovat keskenään yhteensopimattomia, tulee niihin kirjautuminen toteuttaa erikseen. Yhteensopimattomat SSO-palvelut käyttävät esimerkiksi eri SSO-protokollia. Eri palvelut voivat tarjota omia kirjastojaan SSO-integrointeja helpottamaan.
2. Kun käyttäjätiedot saadaan eri SSO-palveluntarjoajilta, jotka eivät ole keskenään yhteensopivia, tulee käyttäjätietojen kyseleminen toteuttaa palveluille erikseen. SSO-palvelujen tarjoamista kirjastoista voi olla apua tässäkin vaiheessa.
3. Kirjautumistavoille yhteisiä täytyy kuitenkin olla ainakin niiden tietojen, jotka on määritelty järjestelmässä välttämättömiksi käyttäjätiedoiksi. Näin ollen käyttäjätietojen riittävyyden tarkastelu voidaan toteuttaa yhdellä kaikille kirjautumistavoille yhteisellä ohjelmakoodilla.
4. Kun käyttäjätietojen hankkiminen ja niiden riittävyyden tarkastelu erotetaan uuden käyttäjän luomisesta ja vanhan käyttäjän päivittämisestä, voidaan nämäkin toteuttaa kaikille kirjautumistavoille samalla ohjelmakoodilla.
5. SSO-tilien tallentaminen tietokantaan:

    - Vähimmillään SSO-käyttäjien tunnistamiseksi täytyy pitää kirjaa seuraavista käyttäjien tiedoista:
        - SSO-palvelu, josta käyttäjä on peräisin.
        - Käyttäjän tunniste SSO-palvelussa.
        - Viite paikalliseen käyttäjätiliin, jolle SSO-tili kuuluu.
    - Useiden SSO-tilien yhdistäminen paikalliseen käyttäjätiliin voi vaatia käytännössä suuremman käyttäjätietomäärän tallentamista tässä mainittujen vähimmäistietojen yhteyteen. Lisätietoja löytyy luvusta 5.9.3.
6. Käyttäjätilille tulee antaa oikeat käyttöoikeudet, katso luku 6.1.9 .
7. Käyttäjän lisääminen istuntoon on perustaltaan sama riippumatta kirjautumistavasta. Kuitenkin voi olla, että joihinkin kirjautumistapoihin liittyy lisätoimintoja, joita varten istuntoon on syytä tallentaa lisätietoja. Näin ollen tähän vaiheeseen voidaan joutua toteuttamaan osittain eroavia ohjelmakoodeja eri kirjautumistavoille.

- Esimerkiksi Facebook haluaa, että Facebook-istuntojen validiutta tarkasteltaisiin säännöllisin väliajoin ja kirjattaisiin epävalideihin istuntoihin nojaavat käyttäjät pois palvelusta. Tätä varten TIMissä tallennetaan istuntoon session info access token sekä aika, josta voidaan päätellä istunnon seuraava tarkistusajankohta. Lisätietoja luvussa 5.11.4.

6.1.12 SSO-järjestelmän käyttöehtojen uudelleenselvittäminen

On luonnollista, että SSO-integraation toteuttamisen aikana lisääntyy tietämys siitä ohjelmistosta, johon integraatiota ollaan tekemässä. Siispä SSO-järjestelmän käyttöehdoista ja muista vastaavista dokumenteista saattaa paljastua SSO-integraation toteuttamisen lopussa uusia asioita, jotka vaativat toimenpiteitä. On siis syytä varata integraation loppuun aikaa kyseisen dokumentaation läpikäyntiin sekä läpikäynnin aikana huomattujen puutteiden korjaamiseen.

6.2 Tulosten yhteenveto

Tutkielman empiirisen osan tuloksina esitettiin prosessi, jota noudattamalla single sign-on -integrointien toteuttamista voidaan sujuvoittaa. Prosessi tarjoaa:

- Ohjeita SSO-integraation vaatimusten selvittämiseen niin integraation kohteena olevan tietojärjestelmän näkökulmasta kuin integroitavan SSO-palvelun näkökulmasta.
- Ohjeita SSO-palvelujen tarjoamien käyttäjätietojen selvittämiseen.
- Ohjeita SSO-palvelujen käyttöönottomahdollisuuksien arviointiin käyttäjätietojen saatavuuden perusteella sekä toimenpiteitä erilaisiin saatavuuksiin sopeutumiseksi.
- Ohjeita SSO-palvelun tarjoamien käyttäjätietojen muutoksiin varautumiseksi.
- Ohjeita useiden SSO-kirjautumismahdollisuuksien lisäämiseksi tietojärjestelmään.

Tutkimuskysymyksenä oli, miten single sign-on -palveluja saadaan integroitua tietojärjestelmään. Tulokset vastaavat tähän kysymykseen käytännönläheisellä tavalla. Ne ovat syntyneet SSO-integraatioita toteutettaessa ja pyrkivät käsittelemään koko integraatioprosessia. Tuloksissa yritetään erityisesti huomioida niitä asioita, jotka toteutusten aikana johtivat pohdintoihin, ongelmiin ja tarpeisiin selventää toteutusten vaatimuksia. Arvioni on, että tutkielman

empiirisessä osassa esitettyjen SSO-integraatioiden toteutus olisi ollut helpompaa, mikäli tutkielman tuloksena syntynyt integraatioprosessi olisi ollut käytettävissä.

Tutkielman empiirisen osan tulosten onnistumista heikentää se tosiasia, että muun muassa aikataulusyistä SSO-integraatioiden toteutukset jäivät osin kesken. Tämä tuodaan kuitenkin myös esiin tuloksissa. Katson myös, että toteutusten osittainen kesken jääminen ei kuitenkaan vaikuta integraatiota helpottamaan tarkoitetun prosessin kaikkiin osiin. Tulosten laatua heikentävien vaikutusten voi katsoa rajoittuvan niihin lukuihin, jotka käsittelevät useiden SSO-tilien yhdistämistä paikalliseen käyttäjätiliin.

Toinen tulosten onnistumista heikentävä tekijä on, että SSO-integrointeja ei otettu aikataulusyistä käyttöön tuotantoympäristössä. On mahdollista, että tuotantoympäristössä havaittaisiin toteutuksista joitain toimenpiteitä vaativia puutteita. Arvioni kuitenkin on, että tämä tekijä ei heikennä tulosten laatua merkittävästi. Puutteet olisivat todennäköisimmin alhaisen abstraktiotason ongelmia koskien joitakin toteutuksen yksityiskohtia, jotka tuskin päätyisivät tuloksiin, joiden on tarkoitus keskittyä integraatioprosessin perusperiaatteisiin.

7 Yhteenveto

Tässä pro gradu -tutkielmassa ollaan päädytty siihen, että single sign-on -järjestelmillä voidaan parantaa tietojärjestelmiin kirjautumisen epäkohtia. Esimerkiksi ihmisten ongelmat muistaa salasanoja vähentyvät, kun he voivat kirjautua samoilla tunnuksilla useisiin eri järjestelmiin. Näin ollen salasanoista voidaan luoda pidempiä ja monimutkaisempia, mikä lisää niiden vahvuutta.

Tutkielmassa esitelläänkin single sign-on -integrointeja tietojärjestelmään. Näistä toteutuksista johdetaan tuloksena prosessi, jota voidaan hyödyntää single sign-on -järjestelmien käyttöönoton mahdollisuuksien arvioinnissa ja integrointien toteuttamisen sujuvoittamisessa. Prosessissa otetaan myös kantaa siihen, miten useita eri kirjautumistapoja voidaan yhdistää yhdelle käyttäjätilille kirjautumiseen.

Prosessin yleistettävyyttä single sign-on -järjestelmien integrointiin heikentää se, että prosessi on johdettu yhteen tietojärjestelmään tehdyistä single sign-on -integroinneista. Tietojärjestelmien arkkitehtuurit vaihtelevat luonnollisesti hyvin paljon, mikä vaikuttaa varmasti myös kirjautumiseen ja single sign-on -integrointien toteuttamiseen. Prosessin laatimisessa ollaan kuitenkin pyritty yleistettävyyteen ja ollaan tarjottu erilaisia ratkaisumahdollisuuksia eri tilanteisiin, vaikka TIMissä oltaisiinkin päädytty vain yhteen vaihtoehdoista.

Joka tapauksessa esitetty prosessi sekä käytännön single sign-on -integrointien esittely soveltuu orientaatioksi omia integrointeja suunnitteleville tahoille. Lisäksi tutkielman teoriaosuudesta löytyy tietoa kirjautumisen haasteista ja niihin vastaamisesta yleisellä tasolla, mikä myös on hyvin relevanttia kirjautumisjärjestelmiä suunniteltaessa ja toteutettaessa.

Lähteet

Aloul, F., S. Zahidi ja W. El-Hajj. 2009. “Two factor authentication using mobile phones”. Teoksessa *2009 IEEE/ACS International Conference on Computer Systems and Applications,* 641–644. Toukokuu. doi:10.1109/AICCSA.2009.5069395.

Ardagna, Claudio Agostino, Ernesto Damiani, Sabrina De Capitani di Vimercati, Fulvio Frati ja Pierangela Samarati. 2006. “CAS++: An Open Source Single Sign-On Solution for Secure e-Services”. Teoksessa *Security and Privacy in Dynamic Environments,* toimittanut Simone Fischer-Hübner, Kai Rannenberg, Louise Yngström ja Stefan Lindskog, 208–220. Boston, MA: Springer US. ISBN: 978-0-387-33406-6.

Barth, A. 2011. *HTTP State Management Mechanism.* RFC 6265. RFC Editor, huhtikuu. http://www.rfc-editor.org/rfc/rfc6265.txt.

Bazaz, Tayibia, ja Aqeel Khalique. 2016. “A Review on Single Sign on Enabling Technologies and Protocols”. *International Journal of Computer Applications* 151 (11): 18–25.

Bhattacharyya, Debnath, Rahul Ranjan, Farkhod Alisherov, Minkyu Choi ym. 2009. “Biometric authentication: A review”. *International Journal of u-and e-Service, Science and Technology* 2 (3): 13–28.

Biennier, Frdrique. 2011. “Web Single Sign On and SAML”. Teoksessa *Encyclopedia of Cryptography and Security,* toimittanut Henk C. A. van Tilborg ja Sushil Jajodia, 1377–1382. Boston, MA: Springer US. ISBN: 978-1-4419-5906-5. doi:10.1007/978-1-4419-5906-5_906. https://doi.org/10.1007/978-1-4419-5906-5_906.

Bonneau, J., C. Herley, P. C. v. Oorschot ja F. Stajano. 2012. “The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes”. Teoksessa *2012 IEEE Symposium on Security and Privacy,* 553–567. Toukokuu. doi:10.1109/SP.2012.44.

Brumen, B., ja A. Černezel. 2014. “Brute force analysis of PsychoPass-generated Passwords”. Teoksessa *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* 1366–1371. Toukokuu. doi:10.1109/MIPRO.2014.6859780.

Brumen, B., M. Heriko, I. Rozman ja M. Hölbl. 2013. “Security analysis and improvements to the PsychoPass method”. *Journal of medical Internet research* 15 (8). doi:10.2196/jmir.2366.

Brumen, B., R. Ivani ja I. Rozman. 2016. “A comparison of password management policies”. Teoksessa *2016 Portland International Conference on Management of Engineering and Technology (PICMET),* 1922–1927. Syyskuu. doi:10.1109/PICMET.2016.7806737.

Butler, R., V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer ja C. Kesselman. 2000. “A national-scale authentication infrastructure”. *Computer* 33, numero 12 (joulukuu): 60–66. ISSN: 0018-9162. doi:10.1109/2.889094.

Chalandar, M. E., P. Darvish ja A. M. Rahmani. 2007. “A centralized cookie-based single sign-on in distributed systems”. Teoksessa *2007 ITI 5th International Conference on Information and Communications Technology,* 163–165. Joulukuu. doi:10.1109/ITICT.2007.4475639.

Chan, Yuen-Yan. 2006. “Weakest Link Attack on Single Sign-On and Its Case in SAML V2.0 Web SSO”. Teoksessa *Computational Science and Its Applications - ICCSA 2006,* toimittanut Marina Gavrilova, Osvaldo Gervasi, Vipin Kumar, C. J. Kenneth Tan, David Taniar, Antonio Laganá, Youngsong Mun ja Hyunseung Choo, 507–516. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-34076-8.

Charalambous., Panayiotis, Marios Karapetris. ja Elias Athanasopoulos. 2018. “KAuth: A Strong Single Sign-On Service based on PKI”. Teoksessa *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications - Volume 1: SECRYPT,* 478–483. INSTICC, SciTePress. ISBN: 978-989-758-319-3. doi:10.5220/0006851906440649.

Chaudhary, P., B. B. Gupta ja S. Gupta. 2016. “Cross-site scripting (XSS) worms in Online Social Network (OSN): Taxonomy and defensive mechanisms”. Teoksessa *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom),* 2131–2136. Maaliskuu.

Cipresso, P., A. Gaggioli, S. Serino, S. Cipresso ja G. Riva. 2012. “How to create memorizable and strong passwords”. *Journal of medical Internet research* 14 (1). doi:10.2196/jmir.1906.

CSC - Tieteen tietotekniikan keskus Oy. n.d.(a). “Haka-käyttäjätunnistusjärjestelmä”. Viitattu 27. elokuuta 2019. https://www.csc.fi/fi/haka-kayttajatunnistusjarjestelma.

———. n.d.(b). “Tietoa meistä”. Viitattu 27. elokuuta 2019. https://www.csc.fi/fi/tietoa-meista.

Dong, Xun, John A. Clark ja Jeremy L. Jacob. 2008. “Threat Modelling in User Performed Authentication”. Teoksessa *Information and Communications Security,* toimittanut Liqun Chen, Mark D. Ryan ja Guilin Wang, 49–64. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-88625-9.

Eduuni-wiki. 2016a. “SAML-tuotteet”. Viitattu 27. elokuuta 2019. https://wiki.eduuni.fi/display/CSCHAKA/SAML-tuotteet.

———. 2016b. “Tekninen dokumentaatio”. Viitattu 27. elokuuta 2019. https://wiki.eduuni.fi/display/CSCHAKA/Tekninen+dokumentaatio.

———. 2019a. “Liittyminen ja rekisteröinti”. Viitattu 27. elokuuta 2019. https://wiki.eduuni.fi/pages/viewpage.action?pageId=27297702.

———. 2019b. “Luottamusverkosto”. Viitattu 27. elokuuta 2019. https://wiki.eduuni.fi/display/CSCHAKA/Luottamusverkosto.

———. 2019c. “Testipalveluun liittyminen”. Viitattu 2. syyskuuta 2019. https://wiki.eduuni.fi/display/CSCMPASSID/Testipalveluun+liittyminen.

Egelman, Serge, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov ja Cormac Herley. 2013. “Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection”. Teoksessa *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2379–2388. CHI ’13. Paris, France: ACM. ISBN: 978-1-4503-1899-0. doi:10.1145/2470654.2481329. http://doi.acm.org/10.1145/2470654.2481329.

Facebook. n.d.(a). “APIs and SDKs”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/apis-and-sdks.

———. n.d.(b). “App Development”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/apps.

———. n.d.(c). “Data Deletion Request Callback”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/apps/delete-data.

———. n.d.(d). “Facebook Login for the Web with the JavaScript SDK”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/facebook-login/web.

———. n.d.(e). “Facebook Login Overview”. Viitattu 4. heinäkuuta 2019. https://developers.facebook.com/docs/facebook-login/overview.

———. n.d.(f). “Facebook Platform Policy”. Viitattu 5. heinäkuuta 2020. https://developers.facebook.com/policy/.

———. n.d.(g). “Login Button”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/facebook-login/web/login-button.

———. n.d.(h). “Login Security”. Viitattu 13. tammikuuta 2020. https://developers.facebook.com/docs/facebook-login/security.

———. n.d.(i). “Manually Build a Login Flow”. Viitattu 5. heinäkuuta 2019. https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow/.

Facebook. n.d.(j). “Permissions Reference - Facebook Login”. Viitattu 5. huhtikuuta 2020. https://developers.facebook.com/docs/facebook-login/permissions.

———. n.d.(k). “Securing Graph API Requests”. Viitattu 13. helmikuuta 2020. https://developers.facebook.com/docs/graph-api/securing-requests/.

———. n.d.(l). “Session Info Access Tokens”. Viitattu 16. tammikuuta 2020. https://developers.facebook.com/docs/facebook-login/access-tokens/session-info-access-token.

Feng-man, M., ja Z. Qiu-yu. 2010. “Cross-Domain Authentication Model Based on Lattice”. Teoksessa *2010 WASE International Conference on Information Engineering,* 1:115–118. Elokuu. doi:10.1109/ICIE.2010.35.

Foundation, OpenID. 2019. “Welcome to OpenID Connect”. Viitattu 5. kesäkuuta. https://openid.net/connect/.

Google. 2017. “Google Identity Platform”. Viitattu 4. heinäkuuta 2019. https://developers.google.com/identity/.

———. 2019a. “Google APIs Terms of Service”. Viitattu 5. heinäkuuta 2020. https://developers.google.com/terms.

———. 2019b. “Integrating Google Sign-In into your web app”. Viitattu 4. heinäkuuta 2019. https://developers.google.com/identity/sign-in/web/sign-in.

———. 2019c. “OpenID Connect”. Viitattu 5. kesäkuuta 2019. https://developers.google.com/identity/protocols/OpenIDConnect.

———. n.d.(a). “Firebase”. Viitattu 4. heinäkuuta 2019. https://firebase.google.com/.

———. n.d.(b). “Pricing plans”. Viitattu 4. heinäkuuta 2019. https://firebase.google.com/pricing.

Grassi, Paul A, Ray A Perlner, Elaine M Newton, Andrew R Regenscheid, William E Burr, Justin P Richer, Naomi B Lefkovitz, Jamie M Danker ja Mary F Theofanos. 2017. *NIST Special Publication 800-63B Digital Identity Guidelines: Authentication and Lifecycle Management [including updates as of 12-01-2017].* Tekninen raportti. doi:10.6028/NIST.SP.800-63b.

Gu, G., ja Y. Zhang. 2013. "Design and Implementation of Cookie-Based CD-SSO". Teoksessa *2013 International Conference on Computational and Information Sciences,* 1785–1788. Kesäkuu. doi:10.1109/ICCIS.2013.467.

Gupta, Shashank, ja B. B. Gupta. 2017. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art". *International Journal of System Assurance Engineering and Management* 8, numero 1 (tammikuu): 512–530. ISSN: 0976-4348. doi:10.1007/s13198-015-0376-0. https://doi.org/10.1007/s13198-015-0376-0.

Hardt, D. 2012. *The OAuth 2.0 Authorization Framework.* RFC 6749. RFC Editor, lokakuu. http://www.rfc-editor.org/rfc/rfc6749.txt.

Herley, C., ja P. Van Oorschot. 2012. "A Research Agenda Acknowledging the Persistence of Passwords". *IEEE Security Privacy* 10, numero 1 (tammikuu): 28–36. ISSN: 1540-7993. doi:10.1109/MSP.2011.150.

Herley, Cormac, ja Dinei Florêncio. 2008. "Protecting Financial Institutions from Brute-Force Attacks". Teoksessa *Proceedings of The Ifip Tc 11 23rd International Information Security Conference,* toimittanut Sushil Jajodia, Pierangela Samarati ja Stelvio Cimato, 681–685. Boston, MA: Springer US. ISBN: 978-0-387-09699-5.

Johns, Martin. 2011. "Session Hijacking Attacks". Teoksessa *Encyclopedia of Cryptography and Security,* toimittanut Henk C. A. van Tilborg ja Sushil Jajodia, 1189–1190. Boston, MA: Springer US. ISBN: 978-1-4419-5906-5. doi:10.1007/978-1-4419-5906-5_661. https://doi.org/10.1007/978-1-4419-5906-5_661.

Komanduri, Saranga, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor ja Serge Egelman. 2011. “Of Passwords and People: Measuring the Effect of Password-composition Policies”. Teoksessa *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2595–2604. CHI ’11. Vancouver, BC, Canada: ACM. ISBN: 978-1-4503-0228-9. doi:10.1145/1978942.1979321. http://doi.acm.org/10.1145/1978942.1979321.

Konakandla, Manideep, Dave Wichers, Paul Petefish, Eric Sheridan ja Dominique Righetto. 2019. “Cross-Site Request Forgery Prevention Cheat Sheet”. Viitattu 31. toukokuuta 2019. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md.

Lang, Michael. 2011. “Reconciling Usability and Security: Interaction Design Guidance and Practices for On-Line User Authentication”. Teoksessa *Information Systems Development,* toimittanut Jaroslav Pokorny, Vaclav Repa, Karel Richta, Wita Wojtkowski, Henry Linger, Chris Barry ja Michael Lang, 397–416. New York, NY: Springer New York. ISBN: 978-1-4419-9790-6.

Lehto, T. 2018. “Pankkitunnusten vaatimukset menevät täysin uusiksi 2019 pankitkaan eivät tarkkaan tiedä mitä se tarkoittaa asiakkaille”. Tekniikka&Talous. 11. huhtikuuta. Viitattu 28. maaliskuuta 2019. https://www.tekniikkatalous.fi/tekniikka/ict/pankkitunnusten-vaatimukset-menevat-taysin-uusiksi-2019-pankitkaan-eivat-tarkkaan-tieda-mita-se-tarkoittaa-asiakkaille-6719519.

Liou, J., ja S. Bhashyam. 2010. “A feasible and cost effective two-factor authentication for online transactions”. Teoksessa *The 2nd International Conference on Software Engineering and Data Mining,* 47–51. Kesäkuu.

Mayron, Liam M., Yasser Hausawi ja Gisela Susanne Bahr. 2013. “Secure, Usable Biometric Authentication Systems”. Teoksessa *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion,* toimittanut Constantine Stephanidis ja Margherita Antona, 195–204. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-39188-0.

Moriarty, Kathleen, ja Stephen Farrell. 2019. *Deprecating TLSv1.0 and TLSv1.1.* Internet-Draft draft-ietf-tls-oldversions-deprecate-03. IETF Secretariat, maaliskuu. https://tools.ietf.org/html/draft-ietf-tls-oldversions-deprecate-03.

"MPASSid - Enemman aikaa opetukselle ja oppimiselle". n.d. Mpass.fi. Viitattu 30. elokuuta 2019. https://mpass.fi/.

"MPASSid liittymissopimus". n.d. Mpass.fi. Viitattu 30. elokuuta 2019. https://mpass.fi/kayttoonotto/mpassid-liittymissopimus/.

Murillo-Escobar, MA, César Cruz-Hernández, Fausto Abundiz-Pérez ja Rosa Martha López-Gutiérrez. 2015. "A robust embedded biometric authentication system based on fingerprint and chaotic encryption". *Expert Systems with Applications* 42 (21): 8198–8211.

Naik, N., ja P. Jenkins. 2017. "Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect". Teoksessa *2017 11th International Conference on Research Challenges in Information Science (RCIS),* 163–174. Toukokuu. doi:10.1109/RCIS.2017.7956534.

Nordea. 2019. "Pankkitunnukset henkilöasiakkaille | Nordea.fi". Viitattu 28. maaliskuuta. https://www.nordea.fi/henkiloasiakkaat/palvelumme/verkko-mobiilipalvelut/pankkitunnukset.html.

O'Gorman, L. 2003. "Comparing passwords, tokens, and biometrics for user authentication". *Proceedings of the IEEE* 91, numero 12 (joulukuu): 2021–2040. ISSN: 0018-9219. doi:10.1109/JPROC.2003.819611.

Pashalidis, Andreas, ja Chris J. Mitchell. 2003. "A Taxonomy of Single Sign-On Systems". Teoksessa *Information Security and Privacy,* toimittanut Rei Safavi-Naini ja Jennifer Seberry, 249–264. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-45067-2.

Rastogi, V., ja A. Agrawal. 2015. "All your Google and Facebook logins are belong to us: A case for single sign-off". Teoksessa *2015 Eighth International Conference on Contemporary Computing (IC3),* 416–421. Elokuu. doi:10.1109/IC3.2015.7346717.

Rescorla, E. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446. RFC Editor, elokuu. https://www.rfc-editor.org/rfc/rfc8446.txt.

Sakimura, N., J. Bradley, M. Jones, B. de Medeiros ja C. Mortimore. 2017. “OpenID Connect Basic Client Implementer’s Guide 1.0 - draft 38”. Viitattu 15. heinäkuuta 2019. https://openid.net/specs/openid-connect-basic-1_0.html.

Sakimura, Nat, John Bradley, Michael B. Jones, Breno de Medeiros ja Chuck Mortimore. 2014. “OpenID Connect Core 1.0 incorporating errata set 1”. Viitattu 3. heinäkuuta 2020. https://openid.net/specs/openid-connect-core-1_0.html.

Sheffer, Y., R. Holz ja P. Saint-Andre. 2015. *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS).* BCP 195. RFC Editor, toukokuu. http://www.rfc-editor.org/rfc/rfc7525.txt.

Sivakorn, S., I. Polakis ja A. D. Keromytis. 2016. “The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information”. Teoksessa *2016 IEEE Symposium on Security and Privacy (SP),* 724–742. Toukokuu. doi:10.1109/SP.2016.49.

Suoranta, Sanna, Asko Tontti, Joonas Ruuskanen ja Tuomas Aura. 2013. “Logout in Single Sign-on Systems”. Teoksessa *Policies and Research in Identity Management,* toimittanut Simone Fischer-Hübner, Elisabeth de Leeuw ja Chris Mitchell, 147–160. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-37282-7.

Tanvi, P., G. Sonal ja S. M. Kumar. 2011. “Token Based Authentication Using Mobile Phone”. Teoksessa *2011 International Conference on Communication Systems and Network Technologies,* 85–88. Kesäkuu. doi:10.1109/CSNT.2011.24.

Twitter. 2020. “Developer Agreement”. Viitattu 5. heinäkuuta 2020. https://developer.twitter.com/en/developer-terms/agreement.

———. n.d.(a). “GET account/verify_credentials”. Viitattu 5. huhtikuuta 2020. https://developer.twitter.com/en/docs/accounts-and-users/manage-account-settings/api-reference/get-account-verify_credentials.

———. n.d.(b). “Implementing Sign in with Twitter”. Viitattu 3. heinäkuuta 2019. https://developer.twitter.com/en/docs/twitter-for-websites/log-in-with-twitter/guides/implementing-sign-in-with-twitter.

Twitter. n.d.(c). “Twitter developer apps”. Viitattu 3. heinäkuuta 2019. https://developer.twitter.com/en/docs/basics/apps/overview.

———. n.d.(d). “Twitter libraries”. Viitattu 3. heinäkuuta 2019. https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries.

Weir, M., S. Aggarwal, B. d. Medeiros ja B. Glodek. 2009. “Password Cracking Using Probabilistic Context-Free Grammars”. Teoksessa *2009 30th IEEE Symposium on Security and Privacy,* 391–405. Toukokuu. doi:10.1109/SP.2009.8.

Weir, Matt, Sudhir Aggarwal, Michael Collins ja Henry Stern. 2010. “Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords”. Teoksessa *Proceedings of the 17th ACM Conference on Computer and Communications Security,* 162–175. CCS ’10. Chicago, Illinois, USA: ACM. ISBN: 978-1-4503-0245-6. doi:10.1145/1866307.1866327. http://doi.acm.org/10.1145/1866307.1866327.

Wiedenbeck, Susan, Jim Waters, Jean-Camille Birget, Alex Brodskiy ja Nasir Memon. 2005. “PassPoints: Design and longitudinal evaluation of a graphical password system”. HCI research in privacy and security, *International Journal of Human-Computer Studies* 63 (1): 102–127. ISSN: 1071-5819. doi:https://doi.org/10.1016/j.ijhcs.2005.04.010. http://www.sciencedirect.com/science/article/pii/S1071581905000625.

Williams, Jeff, Jim Manico ja Neil Mattatal. 2019. “Cross Site Scripting Prevention Cheat Sheet”. Viitattu 31. toukokuuta 2019. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md.

Woods, Naomi. 2016. “Improving the security of multiple passwords through a greater understanding of the human memory”. Julkaisussa painetun version ISSN 1456-5390. Tohtorinväitöskirja, Väitöskirja : http://urn.fi/URN:ISBN:978-951-39-6846-5.

Yadav, P., ja C. D. Parekh. 2017. “A report on CSRF security challenges amp; prevention techniques”. Teoksessa *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS),* 1–4. Maaliskuu. doi:10.1109/ICIIECS.2017.8275852.

Ying, N., Z. Yao ja Z. Hua. 2009. “The study of multi-level authentication-based single sign-on system”. Teoksessa *2009 2nd IEEE International Conference on Broadband Network Multimedia Technology,* 448–452. Lokakuu. doi:10.1109/ICBNMT.2009.5348533.

You, X., ja Y. Zhu. 2012. “Research and design of Web Single Sign-On scheme”. Teoksessa *2012 IEEE Symposium on Robotics and Applications (ISRA),* 383–386. Kesäkuu. doi:10.1109/ISRA.2012.6219204.

Zhao, S., A. Aggarwal ja R. D. Kent. 2007. “PKI-Based Authentication Mechanisms in Grid Systems”. Teoksessa *2007 International Conference on Networking, Architecture, and Storage (NAS 2007),* 83–90. Heinäkuu. doi:10.1109/NAS.2007.42.

Zhong, J., C. Yan, W. Yu, P. Zhao ja M. Wang. 2014. “A Kind of Identity Authentication Method Based on Browsing Behaviors”. Teoksessa *2014 Seventh International Symposium on Computational Intelligence and Design,* 2:279–284. Joulukuu. doi:10.1109/ISCID.2014.205.