Tampereen yliopisto

Mikael Korpimaa

IDENTITEETIN TARJOAJIEN ROOLI KÄYTTÄJÄHALLINNASSA

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Huhtikuu 2019

TIIVISTELMÄ

Mikael Korpimaa: IAM Käyttäjähallinta
Kandidaatintyö
Tampereen yliopisto
Tietotekniikka
Huhtikuu 2019

Käyttäjähallinta on kriittinen osa nykypäivän palveluissa. Palveluiden sisältö halutaan kohdentaa sen käyttäjälle ja palvelut itse tarvitsevat käyttäjän henkilökohtaisia tietoja, jotka eivät ole muille julkista tietoa. Käyttäjän tiedot voidaan tallentaa käyttäjätunnuksien taakse ja rajoittaa käyttäjän näkemää sisältöä.

Identiteetin- ja pääsynhallinta (Identity and Access Management, IAM) on palvelukokonaisuus, jonka tarkoituksena on tarjota ulkoisen rajapinnan kautta käyttäjän tunnistautumista, autentikointia ja auktorisointia. Tunnetuimpia esimerkkejä IAM:issa käytettävistä ulkoisista identiteetin tarjoajista ovat Facebook ja Google.

Tämän tutkimuksen tavoitteena on esitellä yleisimmät tunnistautumisprotokollat, joita IAM:issa voidaan käyttää, tutkia olemassa olevien identiteetin tarjoajien, kuten Facebookin ja Googlen, toteutuksia ja selvittää mitä kyseiset identiteetin tarjoajat tarjoavat kolmannen osapuolen palveluille.

Tutkimuksessa huomataan Facebookin käyttävän protokollana OAuth 2.0 ja Googlen OIDC:ta (OpenId Connect). OIDC:n auktorisoinnin mukana tuleva Id token helpottaa sovelluksen toimintaa, koska sen ei tarvitse tehdä erillisiä kutsuja noutaakseen käyttäjätietoja. Molempien toteutukset ovat hyvin samanlaisia, koska OIDC on rakennettu OAuth 2.0 protokollan päälle ja molemmat käyttävät standardin mukaista flowia kirjautumisessa.

Avainsanat: IAM, Identity provider, OAuth, Facebook, Google

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AM	Access Management
IAM	Identity- and Access Management
IDP	Identity Provider
IM	Identity Management
OIDC	OpenID Connect
RBAC	Role-Based Access Control
SSO	Single Sign-On
JSON	Javascript Object Notation

1. JOHDANTO

Käyttäjähallinnan merkitys on kasvanut digitalisaation myötä. Suurinta osaa palveluista pyritään digitalisoimaan ja ihmisiä pyritään palvelemaan ilman että heidän tarvitsee poistua kotoaan. Käyttäjien henkilökohtaisia tietoja tulee pitää turvassa, ja käyttäjän on tiedettävä, mitä tietoa hänestä jaetaan.

Identiteetin- ja pääsynhallinta (Identity and Access Management, IAM) on palvelukokonaisuus, jonka tarkoituksena on tarjota ulkoisen rajapinnan kautta käyttäjän tunnistautumista, autentikointia ja auktorisointia.

Tämän työn tavoitteena on tutustua käyttäjä- ja pääsyhallintaan, siihen liittyviin yleisimpiin käsitteisiin ja selvittää miten kolmannen osapuolen identiteetin tarjoajien (Identity provider) kuten Facebookin ja Googlen käyttäjänhallintoja voi hyödyntää omassa sovelluksessa. Työssä käydään läpi niiden kirjautumisprosesseja käyttäen apuna valmiita kehittäjän työkaluja.

Työn toisessa luvussa esitellään, mikä IAM on ja mihin sitä tarvitaan. Tämän osion tarkoituksena on antaa käsitys IAM:ista yleisellä tasolla lukijalle, joka ei entuudestaan tiedä, mikä IAM on. Kolmannessa luvussa käydään läpi yleisimpiä protokollia, joita tulee vastaan käyttäjähallinnan yhteydessä. Tässä luvussa esitellään protokollien yleispiirteet ja käyttötapaukset. Neljännessä luvussa käsitellään erilaisia OAuth floweja, joita voidaan käyttää käyttäjien auktorisointiin ja autentikointiin. Viidennessä luvussa tutustutaan tarkemmin kolmannen osapuolen identiteetin tarjoajiin (Identity Provider): Miten kirjautumisprosessi niiden kautta toimii, millaisia rajapintoja ne tarjoavat ja kuinka niitä voidaan hyödyntää oman sovelluksen käyttäjähallinnassa. Kuudennessa luvussa tehdään yhteenveto tutkimuksesta

2. IAM-PALVELUIDEN TAUSTA

Käyttäjähallinta on tärkeässä roolissa nykypäivän tietojärjestelmissä. Käyttäjien tulee useimmissa palveluissa rekisteröityä ja kirjautua sisään käyttääkseen heille räätälöityä kokemusta. Käyttäjätunnuksien avulla voidaan rajata käyttäjille näkyvää informaatiota ja jakaa heille oikeuksia. Palvelun laajentuessa käyttäjien, resurssien ja oikeuksien määrä kasvaa, joiden seurauksena tarvitaan tehokas tapa, kuten IAM (Identity and Access Management), hallitsemaan käyttäjiä.

IAM on tapa hallinnoida ihmisten käyttöoikeuksia ja rajata heille kuuluvaa sisältöä. Palvelu koostuu IM- (Identity Management) ja AM- (Access Management) komponenteista. Identiteetin hallinnan (IM) avulla voidaan autentikoida käyttäjä ja varmistaa, että hän on kuka hän väittää olevansa. Oikeuksien hallinnan (AM) avulla voidaan käyttäjät auktorisoida ja päästää heidät käsiksi heille kuuluviin resursseihin. Identiteetin tarjoaja IDP (Identity Provider) voi olla IAM-järjestelmän oma tai se voi käyttää muita olemassa olevia ulkoisia IDP:itä kuten Facebook tai Google. Mikroarkkitehtuuripohjaisessa palvelusssa IAM:ia voidaan hyödyntää käyttäen Single Sign-On (SSO) -periaatetta, jolloin käyttäjän tarvitsee kirjautua vain kerran käyttääkseen kaikkia mikropalveluja, jotka ovat kytkettynä IAM:iin ja joihin käyttäjillä on oikeudet

Roolipohjainen käyttäjähallinta RBAC (Role-Based Access Control) on monien organisaatioiden käyttöönottama käytäntö. Käyttäjähallintaa helpotetaan luomalla erilaisia rooleja, joille on määritelty sarja oikeuksia. Näitä rooleja voidaan luoda, poistaa tai muuttaa jälkeenpäin. RBAC tekee käyttäjähallinnasta helppoa ja joustavaa, sillä käyttäjille voidaan vapaasti jakaa rooleja tarpeen mukaan. Se ei ole kuitenkaan täydellinen ratkaisu, sillä rooleja joudutaan muuttamaan jatkuvasti dynaamisesti muuttuvien käyttäjien, oikeuksien ja tehtävien seurauksena. [1]

3. YLEISIMMÄT PROTOKOLLAT

Tässä osiossa käydään läpi yleisimmät protokollat liittyen IAM-ratkaisuihin. SAML, OAuth ja OpenID Connect käyttävät kaikki turvallisuus tokeneja (Security Token) käyttäjien digitaalisen identiteetin autentikointiin ja auktorisointiin. [3]

3.1 SAML

Security Assertion Markup Language (SAML) on XML-pohjainen protokolla, joka pitää huolen käyttäjän autentikoinnista, oikeuksista ja muiden attribuuttien informaatiosta. SAML transaktiossa on kolme tärkeää roolia: Identiteetin tarjoaja (Identity Provider), palveluntarjoaja (Service Provider) ja käyttäjä. Identiteetin tarjoaja on luotettu organisaatio, joka autentikoi ja auktorisoi käyttäjät, palveluntarjoaja on organisaatio, joka ylläpitää palvelua, ja käyttäjä on entiteetti, joka aloittaa transaktion (esimerkiksi nettiselain).

Tyypillisessä SAML-käyttötapauksessa käyttäjä yrittää päästä käsiksi palveluntarjoajan palveluun, minkä seurauksena palveluntarjoaja luo käyttäjälle SAML pyynnön. Selain välittää SAML-pyynnön identiteetin tarjoajalle, joka autentikoi käyttäjän ja generoi SAML-vastauksen selaimelle. Selain lähettää SAML-vastauksen palveluntarjoajalle todennettavaksi. [3]

3.2 OAuth

OAuth on autentikointiprotokolla, joka nousi suosioon Facebookin, Twitterin ja niihin liittyvien sovellusten yhteydessä. OAuth tukee SSO:ta (Single Sign-on), joka mahdollistaa käyttäjien autentikoinnin jakamatta heidän kirjautumistietojaan. [4] OAuth:in toimintaperiaate perustuu oikeuksien delegoimiseen käyttäjille.

OAuth auktorisointiprosessissa on neljä tärkeää roolia: resurssiserveri (Resource server), resurssin omistaja tai käyttäjä (Resource Owner/User), OAuth asiakasohjelma (OAuth Consumer/Client) ja auktorisointiserveri (Authorization server). Resurssiserveri on suojatun datan säilyttävä palvelin, resurssin omistaja on ohjelman käyttäjä ja datan omistaja, OAuth asiakasohjelma on ohjelma, joka tekee käyttäjän puolesta pyynnön päästä resurssiin käsiksi, ja auktorisointiserveri auktorisoi käyttäjän saadessaan luvan resurssin omistajalta.

Tyypillisessä OAuth käyttötapauksessa resurssin omistaja kirjautuu palveluun ja pyytää oikeutta resursseihin toiselta organisaatiolta. OAuth asiakasohjelma pyytää auktorisointiserveriltä Request Tokenia ja salaista avainta. Asiakasohjelma lähettää URL-linkin käyttäjälle ja pyytää auktorisointia, minkä jälkeen auktorisointiserveri pyytää käyttäjältä luvan antaa resurssit asiakasohjelman käyttöön. Auktorisointiserveri generoi asiakasohjelmalle Access Tokenin, jonka se välittää käyttäjälle. Resurssiserveri välittää resurssit asiakasohjelmalle, joka välittää ne käyttäjälle. [3]

3.3 OpenID Connect

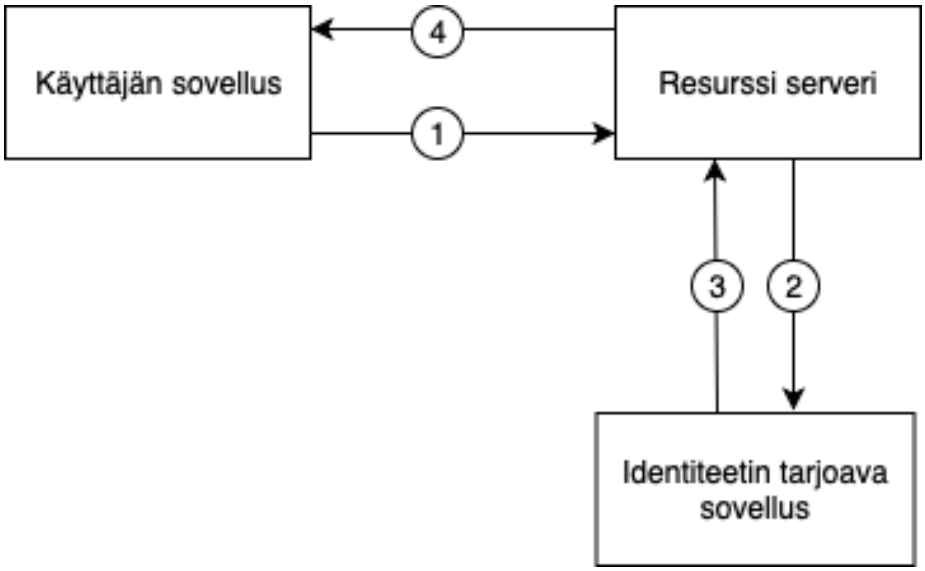
OpenID Connect (OIDC) on OAuth 2.0 pohjainen korvaaja aiemmalle OpenID 2.0 (OpenID) ja on nykyään yksi suosituimmista SSO-protokollista. Sitä hyödyntävät monet isot yritykset kuten Amazon, Google, Microsoft ja PayPal. [2] OIDC on OAuth:in päälle rakennettu identiteettikerros.

OIDC autentikointi ja auktorisointiprosessissa on viisi tärkeää roolia: Loppukäyttäjä (End User), luottava osapuoli (Relying Party), Auktorisoinnin loppupiste (Authorization Endpoint), Token loppupiste (Token Endpoint) ja käyttäjätiedon loppupiste (UserInfo Endpoint). Loppukäyttäjä on ohjelman käyttäjä ja informaation omistaja, ja luotettava osapuoli on sovellus, joka tekee rajapintapyynnön suojattuun resurssiin käyttäjän puolesta. Auktorisoinnin loppupiste auktorisoi käyttäjän, token loppupiste hoitaa tokenien nouto- ja päivityspyynnöt ja käyttäjätiedon loppupiste palauttaa suojattua tietoa käyttäjästä.

Tyypillisessä OIDC käyttötapauksessa loppukäyttäjä välittää kirjautumistietonsa luottavalle osapuolelle, joka välittää tiedot OpenID tarjoajalle. Auktorisointiloppupiste varmentaa loppukäyttäjän kirjautumistiedot ja autentikoi käyttäjän, jonka jälkeen loppukäyttäjä on kirjautuneena järjestelmään. Loppukäyttäjä lähettää auktorisointikoodin luottavalle osapuolelle, joka välittää tiedon OpenID tarjoajalle. Token loppupiste lähettää ID ja access tokenit luotettavalle osapuolelle, joka vahvistaa ID tokenin ja lähettää access tokenin käyttäjätiedon loppupisteelle. Käyttäjätiedon loppupiste lähettää informaatiota käyttäjän attribuuteista luotettavalle osapuolelle, joka välittää palvelut käyttäjälle. [3]

4. OAUTH FLOWIT

OAuth protokollassa on useita tapoja sovelluksen noutaa access token. Vaikka access tokenin nouto voi tapahtua useammalla eri tavalla, resurssien nouto on yksiselitteinen ja toimii kuvan 1 mukaisella flowilla.

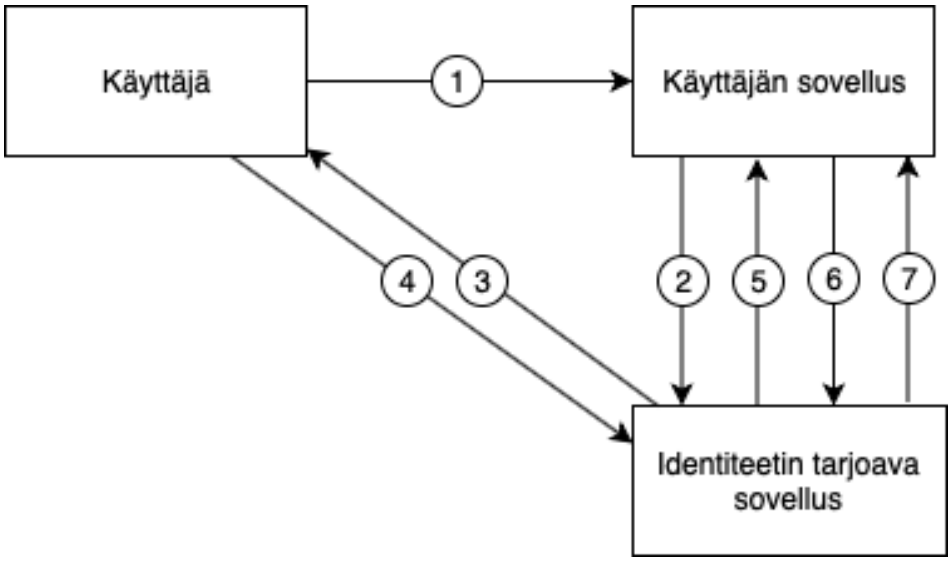


Kuva 1. *Perustuu lähteisiin [5][6]*

Resurssia noutaessa (1) sovellus tekee pyynnön resurssiserverille ja antaa sen mukana access tokeninsa. (2) Resurssiserveri välittää access tokenin identiteetin tarjoajalle. (3) Identiteetin tarjoaja lähettää informaatiota liittyen access tokeniin resurssiserverille, minkä perusteella (4) resurssiserveri verifioi access tokenin ja (5) lähettää pyydetyn resurssin sovellukselle.

4.1 Authorization Code

Authorization Code flow on loppukäyttäjälle kaikista floweista näkyvin. Tätä flowta käytetään esimerkiksi kirjautumisessa kolmannen osapuolen sivustolle käyttäen Facebookin tai Googlen tiliä. Authorization code flowin kulku näkyy kuvasta 2.

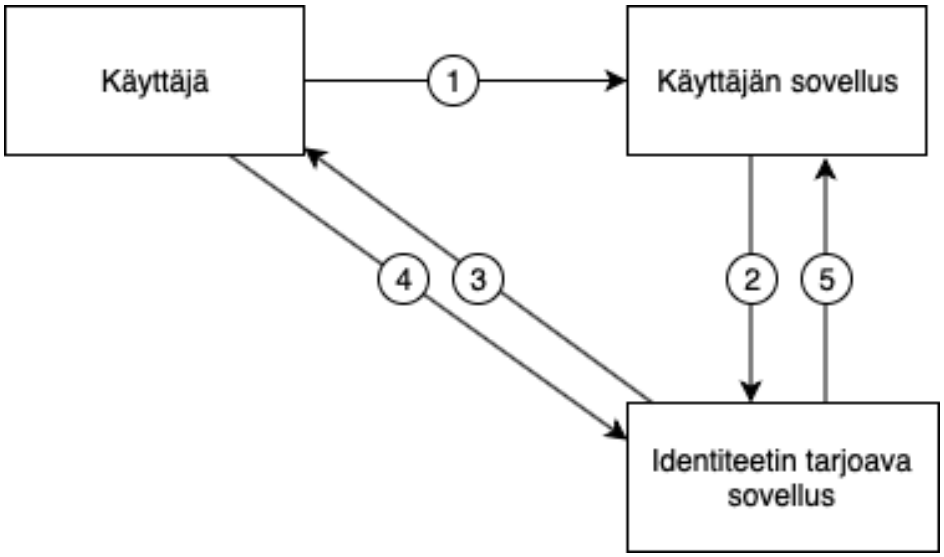


Kuva 2. *Perustuu lähteisiin [5][6]*

Flow alkaa, kun sovellus pyytää käyttäjän kirjautumista sisään identiteetintarjoajan kautta. Tämä pyyntö voi käytännössä olla vain kirjautumisen yhteydessä oleva vaihtoehtoinen nappi kirjautua sovellukseen sisään käyttäen sosiaalista mediaa. (1) Käyttäjä valitsee haluamansa identiteetin tarjoajan ja painaa nappia. (2) Sovellus lähettää identiteetintarjoajalle pyynnön käyttäjätietojen hakuun ja (3) identiteetin tarjoaja ohjaa omalle kirjautumissivulleen. (4) Käyttäjä kirjautuu sisään identiteetin tarjoajan kirjautumissivulle käyttäen omia tunnuksiaan. (5) Kirjautumisen onnistuessa identiteetin tarjoaja lähettää sovellukselle lyhytikäisen authorization coden, jonka (6) sovellus lähettää identiteetin tarjoajalle (7) saadakseen access tokenin.

4.2 Implicit

Implicit flow on yksinkertaisempi versio authorization code flowista. Implicit flowin käyttö ei ole suositeltavaa, koska authorization flow on tietoturvallisempi. [6] Implicit flowin kulku löytyy kuvasta 3.

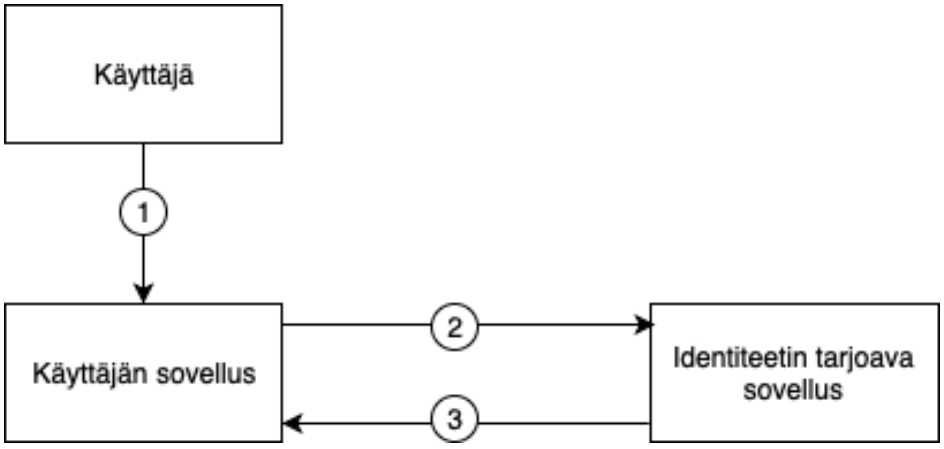


Kuva 3. *Perustuu lähteisiin [5][6]*

Kuvassa 3 vaiheet 1–5 ovat täysin samat kuin authorization flowissa. Implicit flowissa identiteetin tarjoaja lähettää sovellukselle suoraan access tokenin jättäen väliin authorization coden edestakaisin syöttelyn.

4.3 Password

Password flow on tietoturvallisuuden kannalta riskialttein, koska käyttäjätunnukset vaihdetaan suoraan access tokeniin. Tätä flowia tulisi käyttää vain silloin kun luotto molempien osapuolien välillä on suuri. [6] Tämän flowin kulku löytyy kuvasta 4.

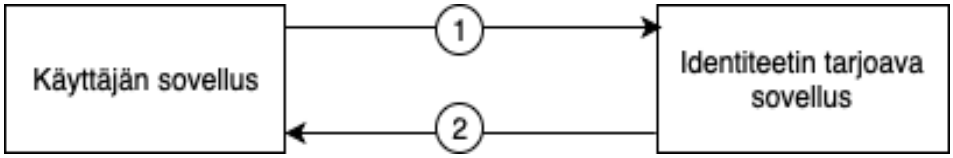


Kuva 4. *Perustuu lähteisiin [5][6]*

Password flowissa (1) Käyttäjä kirjautuu sivustolle sisään. (2) Sovellus välittää käyttäjätunnuksen ja salasanan identiteetin tarjoajalle, joka tarkistaa käyttäjän oikeudet ja (3) palauttaa käyttäjälle kuuluvat resurssit.

4.4 Client Credentials

Client Credentials flow on sovellusten käyttämä tapa hakea sovelluskohtaisia access tokeneja. Sovellus tarvitsee oman access tokeninsa käyttääkseen palveluntarjoajan palveluja. Access tokenin saadakseen sovellus tarvitsee oman client id:n ja client secretin, joita varten sovellus tulee rekisteröidä palveluntarjoajan sivustolla. Tämän flowin kulku löytyy kuvasta 5.



Kuva 5. *Perustuu lähteisiin [5][6]*

Flowissa (1) sovellus lähettää client id:n ja passwordin identiteetin tarjoajalle kutsuessa palvelun tarjoajan palveluita. Kun palveluntarjoaja on tarkistanut sovelluksen oikeudet, (2) se palauttaa sovellukselle access tokenin.

4.5 Refresh Token

Refresh token flowia käytetään uusimaan vanhentunutta access tokenia. Jotta refresh token flowia voi käyttää, sovellukselle pitää antaa alkuperäisen access tokenin noudon yhteydessä myös refresh token. Access tokenin voimassaolo aika on paljon lyhyempi kuin refresh tokenin. [6] Refresh tokenia voidaan käyttää toteuttaessa SSO kirjautumista, jossa access token vanhetessaan uusitaan refresh tokenilla ilman että käyttäjä joutuu kirjautumaan uudestaan. Refresh token flowin kulku voidaan visualisoida kuvan 5 avulla. Access tokenin vanhetessa (1) sovellus lähettää refresh tokenin identiteetin tarjoajalle, (5) joka luo uuden access tokenin ja palauttaa sen sovellukselle.

5. IDENTITEETIN TARJOAJIEN KIRJAUTUMIS-PROSESSIT

Tutkimuksen tavoitteena on selvittää mitä kehittäjän täytyy tehdä ottaakseen Googlen ja Facebookin OAuth 2.0 rajapintoja käyttöönsä, kuinka käyttäjälle näkyy mitä tietoa hänestä jaetaan ja miten aiemmin esitetyt OAuth 2.0 flowit näkyy kirjautumisprosessissa.

Tutkimusta varten täytyy selvittää minkälaisia tietoja kolmannen osapuolen sovellukset voivat tiedustella identiteetin tarjoajilta ja kuinka hyvin käyttäjälle ilmaistaan mitä tietoa on sovellukselle luovuttamassa.

5.1 Tutkimusmenetelmät

Facebookin ja Googlen kirjautumisprosessin ja datan keräyksen tutkimisessa voidaan hyödyntää valmiita kehittäjän työkaluja. Facebookilta löytyy oma Graph API Explorer [7] ja Googlelta OAuth 2.0 Playground [8]. Näiden työkalujen avulla voidaan käydä läpi koko kirjautumisprosessi ja hakea tietoja kirjautuneesta käyttäjästä. Näin tutkimuksessa ei tarvitse kehittää erillistä sovellusta testatakseen OAuth 2.0 toiminnallisuutta. Tutkimuksessa luotetaan tietoihin, jotka löytyvät Facebookin ja Googlen dokumentaatioista, ja kehittäjän työkaluilla tehtyihin havaintoihin

5.2 Facebook

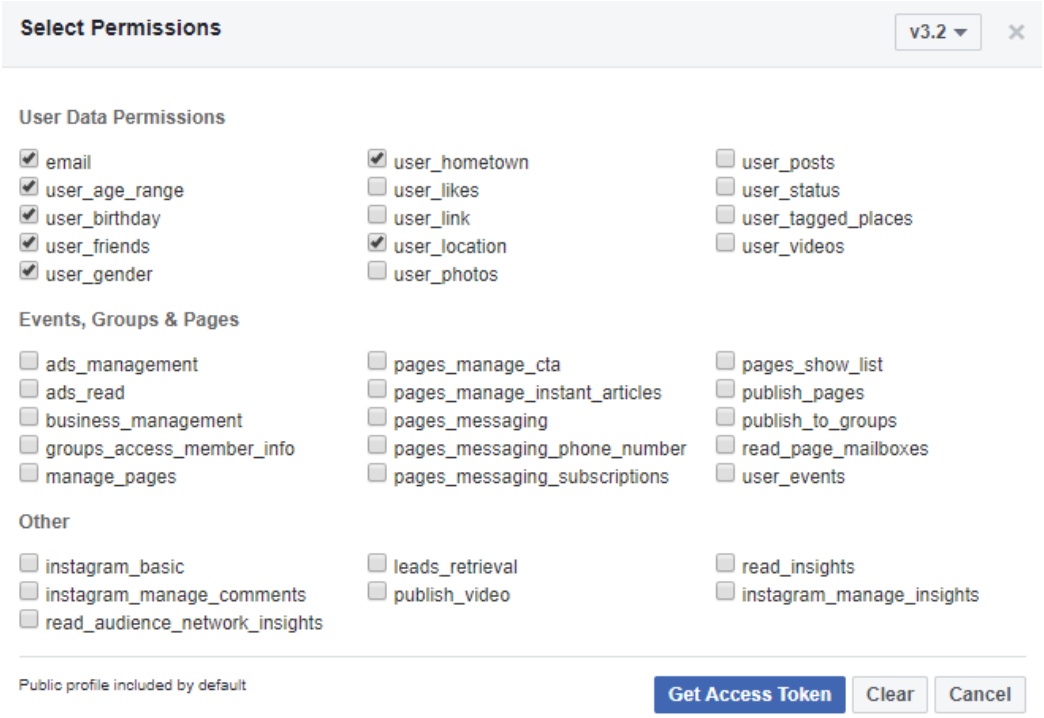
Facebook käyttää kirjautumisessa OAuth 2.0 protokollaa. Kolmannen osapuolen sovelluksen täytyy rekisteröidä Facebookin kehittäjä sivuston kautta ottaakseen Facebook kirjautumisen käyttöön. Rekisteröitymisen jälkeen sovellus saa oman client id:n ja client salaisuuden, joilla se voi tunnistautua Facebookille kirjautumista varten.

5.2.1 Kirjautumisprosessi

Sovelluksen OAuth 2.0 tunnusten haku tapahtuu Graph Api Explorerissa dropdown valikon kautta. Valitessa oma sovellus, Graph Api Explorer välittää automaattisesti tunnukset Facebookille kirjautumista varten.

Kirjautumisprosessi alkaa sovelluksen tekemällä kutsulla, jossa määritellään oikeudet, mitä käyttäjän tulee sovellukselle antaa. Oikeudet, mitä käyttäjältä voidaan pyytää,

näkyvät kuvassa 6. Oikeuksien väärinkäyttöä välttääkseen Facebook tarkastaa sovellusten oikeuspyynnöt. Sovelluksen pyytäessä vain käyttäjän julkista profiilia ja sähköpostiosoitetta tarkistusta ei tarvitse tehdä [9]. Jos Facebook ei ole tarkastanut sovelluksen pyytämiä oikeuksia, ilmoitetaan tästä käyttäjälle kirjautumisen yhteydessä ja käyttäjä voi lähettää oikeudet tarkistettaviksi. Käyttäjä voi kirjautumisen yhteydessä tarkastella oikeuksia, joita sovellus pyytää, ja manuaalisesti estää ei-toivottujen oikeuksien antamista. Käyttäjän kirjautumisen jälkeen Facebook ohjaa käyttäjän takaisin kutsun lähettäneelle sovellukselle ja antaa sille access tokenin, jolla on käyttäjän hyväksymät oikeudet.



Kuva 6. *Facebookin tarjoamat oikeudet [7]*

Kun sovellus on saanut access tokenin, voi tämä tehdä rajapintakutsuja Facebookille. Kutsu näyttää rajapintakutsu 1:n kaltaiselta ja se palauttaa vain tietoa, johon access tokenilla on oikeudet. Rajapintakutsujen kautta sovellus voi hakea esimerkiksi käyttäjän tietoja, julkaisuja, ryhmiä tai kiinnostuksen kohteita, jos käyttäjä on antanut niihin oikeudet. Rajapintakutsujen kautta voi myös tehdä käyttäjän puolesta julkaisuja.

Method: 'GET'
Url: 'https://graph.facebook.com/v3.2/me'
Data: 'fields=id,name,friends,birthday,gender,email,location' +
'&access_token=EAAFx2L…'

Rajapintakutsu 1. *Facebookin esimerkki rajapintakutsu*

Graph API Explorerin käyttämä flow näyttäisi olevan implicit flow, koska käyttäjän kirjautuminen palauttaa suoraan access tokenin eikä authorization codea. Kolmannen osapuolen sovelluksessa tulisi kuitenkin käyttää authorization flowia dokumentaation mukaisesti [9].

5.2.2 käyttäjädata

Rajapintakutsut palauttavat käyttäjädatan JSON-formaatissa. Aiemmin kuvassa 6 valituilla oikeuksilla ja rajapintakutsulla 1 tehdyllä pyynnöllä saatu käyttäjädata näyttää esimerkin 1 kaltaiselta.


```
{
  "id": "2493937697294773",
  "name": "Mikael Korpimaa",
  "friends": {
    "data": [
    ],
    "summary": {
      "total_count": 122
    }
  },
  "birthday": "05/26/1994",
  "gender": "male",
  "age_range": {
    "min": 21
  },
  "email": "email@gmail.com",
  "location": {
    "id": "106245332744025",
    "name": "Tampere, Finland"
  }
}
```


Esimerkki 1. *Esimerkkikäyttäjädata*

Ilman erikseen pyydettyjä oikeuksia Facebook antaa rajapintakutsun kautta vain käyttäjän id:n ja nimen. Ystävien haku kertoo käyttäjän ystävien määrän ja niiden ystävien tietoja, jotka ovat asentaneet haun tehneen sovelluksen [9]. Muuten kentät ovat hyvin yksiselitteiset. Sovellus voi tehdä lähes kaiken mitä käyttäjä itsekin, jos sillä on tarpeeksi laajat oikeudet.

5.3 Google

Google käyttää kirjautumisessaan OpenID Connect protokollaa. Suurin ero Facebookin protokollaan verrattuna on se, että sovellus saa id tokenin samalla kun se vaihtaa authorization coden acces tokeniin. OIDC toteutusta voi käyttää samalla tavalla kuin tavallista OAuth 2.0 protokollaa. Googlelta löytyy rajapintoja tietoihin, jotka löytyvät id tokenista. Googlen identiteetin tarjoajan käyttöönotto ei merkittävästi eroa Facebookin prosessista. Sovelluksen kehittäjän tulee rekisteröidä sovellus googlen developer sivustolla saadakseen sovelluksen client id:n ja client secretin.

5.3.1 Kirjautumisprosessi

Google OAuth 2.0 Playground esittää täyden OAuth 2.0 protokollan Facebookin Graph Api Exploreria paremmin. Prosessissa käydään läpi Authorization Code flowin kaikki vaiheet. Playgroundia voi käyttää valmiilla OAuth 2.0 tunnuksilla tai syöttää manuaalisesti oman sovelluksen tunnukset. OAuth flowin voi suorittaa playgroundissa Googlen serverillä tai omassa sovelluksessa. Tutkimuksessa OAuth flow suoritettiin Googlen serverillä.

Google tarjoaa erittäin kattavan valikoiman rajapinnoista, joista voi valita mihin sovellus haluaa käyttäjältä oikeudet. Sovelluksen lähettämän kirjautumispyynnön jälkeen käyttäjä ohjataan googlen kirjautumissivulle, jossa hän voi tarkastella tai muuttaa oikeuksia, joita on antamassa sovellukselle. Kun käyttäjä on kirjautunut ja hyväksynyt sovelluksen pyytämät oikeudet, Google palauttaa sovellukselle Authorization code:n. Tämän jälkeen sovelluksen tulee vaihtaa Authorization code tokeneihin. Tokenien vaihdossa sovelluksen tulee välittää googlelle client id, client secret ja Authorization code.

Tokenien vaihdossa sovellus saa takaisin access tokenin, id tokenin ja refresh tokenin. Access tokenin saatuaan sovellus voi tehdä kutsuja rajapintoihin, joihin käyttäjä on saanut oikeudet. Refresh tokenia sovellus voi käyttää päivittääkseen access tokenin sen vanhetessa. Sen avulla käyttäjän ei tarvitse kirjautua sovellukseen erikseen, ellei refresh token ole myös vanhentunut. Access tokenia päivittäessä sovellukselle annetaan myös uusi id token.

5.3.2 Id token

Id token sisältää yksinkertaista tietoa käyttäjästä ja kirjautumisesta. Tämän tokenin avulla sovellusten ei tarvitse tehdä erillisiä rajapintakutsuja yksinkertaisia tietoja käyttäjästä ja sovellus tietää koska sen tulee uusia access token. Id token annetaan JWT

(JSON Web Token) muodossa. JWT on standardin mukainen tapa kompressoida ja allekirjoittaa JSON objekteja [10]. Esimerkissä 2 on id tokenin rakenne dekoodattuna JSON:iksi.


```
{
  "iss": "https://accounts.google.com",
  "azp": "842153132876-onik9rhd4mfb6g24ua5febp3iciuu1tn.apps.googleusercontent.com",
  "aud": "842153132876-onik9rhd4mfb6g24ua5febp3iciuu1tn.apps.googleusercontent.com",
  "sub": "117856605291333804181",
  "email": "email@gmail.com",
  "email_verified": true,
  "at_hash": "oa63lZ2H1UCOMNixpm6aJw",
  "name": "Mikael Korpimaa",
  "picture": "https://lh3.googleusercontent.com/-
t3cANY1YTXc/AAAAAAAAAAI/AAAAAAAAABs/gfY2T8hq7Kc/s96-c/photo.jpg",
  "given_name": "Mikael",
  "family_name": "Korpimaa",
  "locale": "fi",
  "iat": 1556354150,
  "exp": 1556357750
}
```


Esimerkki 2. *Esimerkki id tokenista*

Id tokenissa osa kentistä ovat hyvin yksiselitteiset, mutta tokenin erikoisemmat kentät kertovat seuraavanlaista tietoa: *iss* kertoo mistä token on peräisin, *azp* millä client id:llä token on noudettu, *aud* mille client id:lle token on tarkoitettu käytettäväksi, *sub* käyttäjän uniikin tunnisteen, *at_hash* mihin access tokeniin id token on rinnastettu, *iat* koska token on luotu ja *exp* koska token vanhenee [11].

6. YHTEENVETO

Työssä esiteltiin kolme yleisintä autentikointi protokollaa, joista OAuth 2.0 pohjaiset protokollat valittiin syvempää tarkastelua varten. OAuth 2.0 protokolla tukee useita eri autentikointi floweja. Jokaisella flowilla on oma tarkoituksensa. Oli se sitten sovelluksen autentikoiminen käyttääkseen identiteetin tarjoajan rajapintoja, käyttäjän autentikoiminen päästäkseen käsiksi käyttäjän tietoihin tai päivittääkseen vanhenevia tokeneja.

Tutkimusvaiheessa tarkasteltiin Facebookin ja Googlen kirjautumisprosesseja ja selvitettiin, miten niiden prosesseissa näkyivät aiemmin esitellyt OAuth flowit. Googlen ja Facebookin toteutuksien suurimpana erona oli Googlen käyttämän OIDC:n mukana tuleva Id token, joka vähentää kolmannen osapuolen sovelluksen tarvetta tehdä kyselyjä käyttäjien perustiedoista.

Googlen ja Facebookin kehittäjäntyökaluissa oli merkittäviä eroja. Facebookin Graph API Explorerissa oli yksinkertaistettu kirjautumisprosessia tekemällä automaattisesti osia OAuth flowista. Googlen OAuth 2.0 Playgroundin prosessissa käytiin manuaalisesti läpi jokainen vaihe authorization code flowista. Facebookin kehittäjän työkalua oli helpompi käyttää kuin Googlen, mutta Googlen työkalu antaa paremman käsityksen koko autentikointiprosessista.

Tutkimuksen tavoitteena oli selvittää kuinka Facebookin ja Googlen rajapintoja käytetään ja miten niitä voidaan hyödyntää omassa sovelluksessa. Tutkimuksessa päästiin tavoitteeseen ja saatiin selville, minkälaista tietoa käyttäjästä jaetaan ja kuinka Facebookin ja Googlen rajapinnat toimivat. Tutkimuksen tuloksena ei saatu uutta tietoa, mutta tutkimusprosessi oli opettavainen ja sitä voi mahdollisesti joku muukin hyödyntää OAuthiin tutustuessa.

LÄHTEET

[1] Ning, P., Zhiqiang, Z., Liangsheng, H. & Lei, S. (2018). An efficiency approach for RBAC reconfiguration with minimal roles and perturbation, Concurrency and Computation: Practice and Experience, Vol. 30(11), pp. e4399. Saatavissa: https://onlinelibrary-wiley-com.libproxy.tut.fi/doi/full/10.1002/cpe.4399.

[2] Mainka, C., Mladenov, V., Schwenk, J. & Wich, T. (2017). SoK: Single Sign-On Security - An Evaluation of OpenID Connect, Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017, pp. 251-266.

[3] Naik, N. & Jenkins, P. (2017). Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect, 2017 11th International Conference on Research Challenges in Information Science (RCIS), pp. 163-174.

[4] Sujanani, T. & Vinod, S. (2018). Implementation of OpenId connect and OAuth 2.0 to create SSO for educational institutes, International Journal of Engineering and Technology (UAE), Vol. 7(2), pp. 153-157. Saatavissa: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85044094305&doi=10.14419%2fijet.v7i2.6.10142&partnerID=40&md5=ed15528da47170972ac57ab6a20cbfc9.

[5] Takahiko K. (2017). Diagrams And Movies Of All The OAuth 2.0 Flows. Saatavissa: https://medium.com/@darutk/diagrams-and-movies-of-all-the-oauth-2-0-flows-194f3c3ade85

[6] The OAuth 2.0 Authorization Framework dokumentaatio. Viitattu 17.3.2019, Saatavissa: https://tools.ietf.org/html/rfc6749#section-1.3.1

[7] Facebook Graph API Explorer, Viitattu 13.4.2019, Saatavissa: https://developers.facebook.com/tools/explorer

[8] Google OAuth 2.0 Playground. Viitattu 20.4.2019, Saatavissa: https://developers.google.com/oauthplayground

[9] Facebook Login dokumentaatio. Viitattu 26.4.2019, Saatavissa: https://developers.facebook.com/docs/facebook-login/

[10] JWT.io, Viitattu 26.4.2019, Saatavissa: https://jwt.io/introduction/

[11] Google OpenID Connect dokumentaatio. Viitattu 26.4.2019, Saatavissa: https://developers.google.com/identity/protocols/OpenIDConnect