Tampere University

Valtteri Lahti

VERKKOSOVELLUKSEN AUTENTIKOINTI MONIKÄYTTÄJÄHAKEMISTOYMPÄRISTÖSSÄ

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Olli Kivelä
Hannu-Matti Järvinen
Elokuu 2023

TIIVISTELMÄ

Valtteri Lahti: Verkkosovelluksen autentikointi monikäyttäjähakemistoympäristössä
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Elokuu 2023

Käyttäjähakemistojen käyttö on kerryttänyt organisaatioiden suuren suosion. Käyttäjähakemistojen ansiosta organisaation käyttäjillä on vain yksi tunnus, jolla voi kirjautua mihin tahansa organisaation sisäiseen käyttäjähakemiston tuntemaan sovellukseen. Tällöin organisaation käyttäjät voivat luoda ja muistaa yhden vahvan salasanan, eikä heidän tarvitse käyttää heikkoa helposti muistettavaa salasanaa tai käyttää uudelleen vanhaa salasanaa. Tunnuksien ja käyttöoikeuksien sekä niiden hallinnoimisen keskittäminen helpottaa myös luonnollisesti ylläpitäjien työtä. Tietyistä käyttäjähakemistoista on mahdollista autentikoida käyttäjiä käyttäjähakemiston ulkopuolelle käyttäen autentikointiprotokollia.

Tässä työssä käyttäjähakemistojen luoma hyöty haluttiin tuoda myös sovelluksen käyttäjille. Työn tarkoituksena oli toteuttaa autentikointi kahdesta erilaisesta käyttäjähakemistosta mahdollisimman tietoturvallisesti ja yksinkertaisesti. Työssä pyrittiin selvittämään käyttäjähakemistojen, protokollien ja identiteettipalveluiden toimintaperiaatteita sekä niiden yhteensopivuutta. Lisäksi selvitettiin sopivimmat identiteettipalvelut ja protokollat sekä tietoturvallisimmat menetelmät toteutettavalle autentikoinnille.

Autentikointiprotokollista selvitettiin toteutuksen kannalta oleellisimmat. Vertailemalla protokollien ominaisuuksia selvitettiin toteutukseen parhaiten soveltuva protokolla. Toteutuksessa käytettävä identiteettipalvelu valittiin samanlaista prosessia noudattaen. Työssä selvitettiin myös useita mahdollisia autentikointitapahtuman uhkia, jotka huomioitiin, kun sovellukseen kehitettiin tuki autentikointiprotokollalle.

Työn tuloksena ei löydetty yhtä tiettyä tapaa, jolla voitaisiin autentikoida kaikenlaisista käyttäjähakemistoista käyttäjiä. Sen sijaan huomattiin, että verkkosovelluksen autentikointi voidaan toteuttaa käyttäjähakemistosta ja verkkosovelluksen tarpeesta riippuen eri menetelmillä ja protokollilla. Kahden eri menetelmän toimivuus testattiin toteuttamalla autentikointi niiden avulla verkkosovellukseen.

Avainsanat: Käyttäjähakemisto, autentikointi, tietoturva, identiteettipalvelu, OpenID Connect, Lightweight Directory Access Protocol, Windows AD, Azure AD, JWT

ABSTRACT

Valtteri Lahti: Authentication for web application for multiple Active Directories
Master's thesis
Tampere University
Master's Programme in Computing Sciences
August 2023

The use of Active Directories has gained significant popularity within organizations. As a result of using Active Directories, users have only one credential that can be used to log in to any of the organizations applications as long as they are trusted by the Active Directory. This allows organization users to create and remember a single strong password, eliminating the need for a weak and easily memorable password or reusing an old one. Centralizing the credentials, permissions and their management also naturally eases the work of administrators. Certain Active Directories enable authenticating users outside the directory using authentication protocols.

The objective of this thesis was to extend the benefits created by Active Directories to the users of our web application. Before the implementation could be done, it was paramount to understand the principles of different directories, protocols, and identity services, as well as their compatibility. Additionally, the most suitable identity services and protocols were identified, along with the most secure methods of implementing the authentication.

Only the authentication protocols that were relevant to the implementation were examined. By comparing the characteristics of the protocols, the most suitable protocol for the implementation was determined. Following a similar process, the identity service for the implementation was selected. The potential threats to the authentication transaction were also explored and taken into account when developing the authentication.

As a result of this thesis, one specific method for authenticating users from all types of directories was not found. Instead, it was found that the authentication could be implemented using different methods and protocols depending on the directory and the needs of the web application. The functionality of two of the found methods was tested by using them to implement authentication to a web application.

Keywords: Active Directory, authentication, information security, Identity as a Service, OpenID Connect, Lightweight Directory Access Protocol, Windows AD, Azure AD, JWT

PREFACE

Tämä työ toteutettiin Atostek Oy:n asiakasprojektin tarpeisiin perustuen. Haluan kiittää Atostekia diplomityön tekemisen tukemisesta ja Atostekin Olli Kivelää neuvoista, ohjaamisesta ja tuesta työn aikana. Tampereen yliopistolta haluan kiittää professori Hannu-Matti Järvistä työn ohjaamisesta.

Tampereella, 8.10.2023

Valtteri Lahti

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AD	engl. Active Directory
AD DS	engl. Active Directory Domain Services
AD FS	engl. Active Directory Federation Services
AS	engl. Authorization server
AWS	engl. Amazon Web Services
Azure AD	engl. Azure Active Directory
CSRF	engl. Cross site request forgery
DC	engl. Domain Controller
DN	engl. Distinguished Name
IDaaS	engl. Identity as a Service
IdP	engl. Identity Provider
JWE	engl. JSON Web Encryption
JWS	engl. JSON Web Signature
JWT	engl. JSON Web Token
LDAP	engl. Lightweight Directory Access Protocol
LDAPS	engl. LDAP over SSL/TLS
MITM	engl. Man in the middle (attack)
OAuth	Open Authorization
OIDC	OpenID Connect
OP	engl. OpenID Provider
SAML	engl. Security Assertion Markup Language
SDK	engl. Software Development Kit
TLS	engl. Transport Layer Security
Windows AD	engl. Windows Active Directory
XML	engl. Extensible Markup Language

.

1. JOHDANTO

Käyttäjähakemistot ovat aikojen saatossa saaneet erilaisten organisaatioiden parissa suuren suosion. Käyttäjähakemiston avulla organisaation käyttäjiä ja heidän pääsyoikeuksiaan voidaan hallita keskitetysti. Käyttäjähakemistoa käytetään myös käyttäjän autentikoimiseen eri palveluihin. Sisäverkon käyttäjähakemistoihin käyttäjä voi kirjautua aina ollessaan sisäverkkoon yhdistettynä, jolloin hän saa yhteyden kaikkiin sisäverkon palveluihin ja laitteisiin. Jos käyttäjähakemisto on pilvipalvelupohjainen, ulkoverkossa oleva hakemisto, sillä voidaan autentikoida käyttäjä ulkoverkossa oleviin palveluihin.

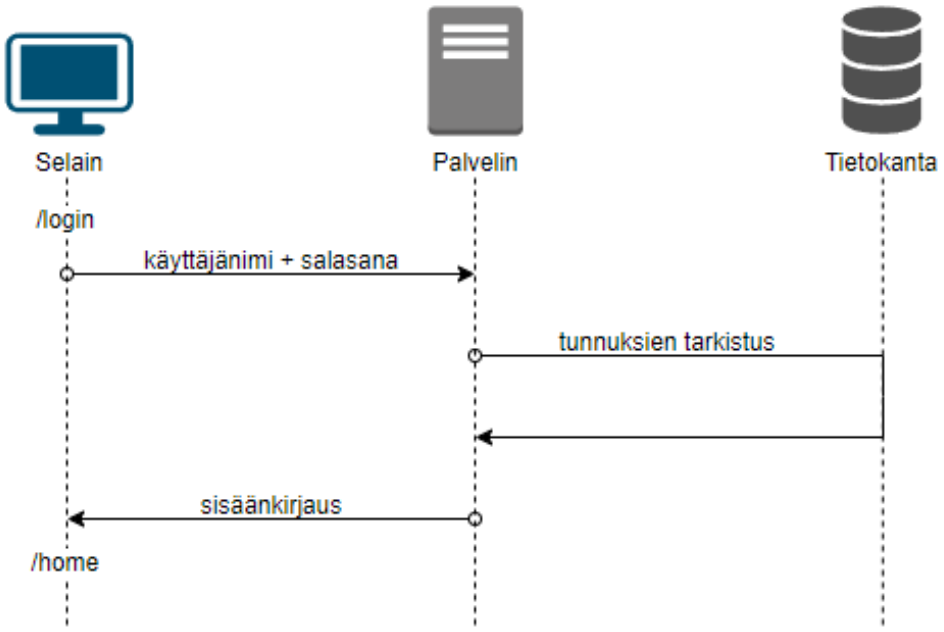
Jotkut sisäverkon käyttäjähakemistopalvelimet saadaan tukemaan autentikointia myös ulkoverkkoon asentamalla palvelimeen laajennuksia. Laajennukset mahdollistavat palvelimen rajapinnan avaamista ulkoverkkoon sekä tuen protokollille, joiden avulla autentikoiminen ulkoverkkoon voidaan tehdä. Pilvipalvelupohjaisissa käyttäjähakemistoissa tuki useimmille autentikointiprotokollille on yleensä oletuksena.

Käyttäjän autentikoiminen sovellukselle hakemiston avulla parantaa tietoturvaa monella tapaa. Käyttäjien ja heidän oikeuksien hallinnointi ei tällöin tapahdu sovelluksien tasolla vaan voidaan toteuttaa hakemistolla. Tästä seuraa se, että käyttäjien ei tarvitse keksiä uutta vaikeasti muistettavaa salasanaa tai pahimmassa tapauksessa käyttää uudelleen vanhaa tai heikkoa helposti muistettavaa salasanaa. Käyttäjien hallinnointi hakemiston tasolla mahdollistaa myös sen, että ylläpitäjät voivat asettaa monivaiheisen tunnistautumisen käyttäjälle, jolloin sovellukseenkaan ei pääse ilman tunnistautumisen suorittamista. Käyttäjän autentikoiminen hakemiston avulla myös parantaa sovelluksen käyttäjäkokemusta.

Uudet rajapinnat ja integraatiot tuovat mukanaan myös uusia riskejä, jotka on otettava huomioon. Jos integroinnin tietoturvassa on aukkoja, taitava hyökkääjä pystyy kaappaamaan käyttäjän istunnon ja päästä näin arkaluontoiseen dataan käsiksi. Tämän vuoksi on äärimmäisen tärkeää, että kaikki tarjolla olevat turvallisuusominaisuudet otetaan käyttöön.

Tämän työn tavoitteena on suunnitella ja toteuttaa käyttäjien autentikointi erilaisista käyttäjähakemistoista verkkosovellukseen käyttäen konstruktiivista menetelmää. Tätä varten työssä tutkitaan olemassa olevia autentikointiprotokollia ja käyttäjähakemistoja sekä niiden keskinäisiä suhteita. Protokollista ja menetelmistä arvioidaan sopivimmat toteutukseen. Sen jälkeen tutkimuksen tuloksia käytetään hyväksi ulkoisen autentikoinnin toteuttamiseksi kohdejärjestelmään.

Tässä työssä kehitettävä järjestelmä on terveydenhuollon ammattihenkilöiden käyttämä verkkosovellus, joka luokitellaan lääkinnälliseksi laitteeksi. Verkkosovellus on ajossa pilvipalvelussa julkisessa internetissä. Sovelluksessa on tällä hetkellä mahdollista kirjautua vain kuvan 1 kuvaamalla tavalla, eli antamalla sovelluksen tietokantaan tallennettu käyttäjänimi ja salasana, joiden oikeellisuus tarkastetaan sovelluksen omasta tietokannasta. Lähdekoodi on kirjoitettu C#-kielellä käyttäen .NET-frameworkia.



Kuva 1: Autentikoinnin nykytilanne

Sovellukseen lisätään loppukäyttäjien kirjautumisen helpottamiseksi mahdollisuus autentikoitua käyttämällä käyttäjän käyttäjähakemistoa hyväksi. Käyttäjähakemiston avulla autentikoitaessa loppukäyttäjän ei tarvitse muistaa uutta monimutkaista salasanaa päästäkseen kirjautumaan. Samalla kirjautumistilanteesta tulee loppukäyttäjän näkökulmasta suoraviivaisempi. Pääsyoikeuksien

hallinnointi jää kuitenkin sovelluksen vastuulle, koska käyttäjien pääsyoikeudet riippuvat sovellukseen asetetuista rooleista.

Loppukäyttäjät, jotka tulevat edellä mainittua autentikointia käyttämään, ovat joko Windows Active Directory -käyttäjähakemistojen tai Azure Active Directoryn ja sen kaltaisten käyttäjähakemistojen käyttäjiä. Tämän työn tavoitteena on suunnitella ja toteuttaa verkkosovellukseen toiminnallisuus, joka tukee molempien käyttäjähakemistojen avulla tehtävää autentikointia. Ideaalein ratkaisu olisi käyttää samaa menetelmää molempien käyttäjähakemistojen käyttäjien autentikointiin. Jos käyttäjähakemistojen erilaisuus ei salli saman menetelmän käyttöä, valitaan paras vaihtoehto kummallekin käyttäjähakemistolle erikseen. Autentikointia varten käytettävän menetelmän arvioinnissa keskitytään tietoturvallisuuteen, yhteensopivuuteen ja kustannustehokkuuteen. Tämän lisäksi on otettava huomioon, mitä protokollia integroitavat käyttäjähakemistot tukevat.

Käyttäjähakemistoja ja protokollia on useita erilaisia, eikä kaikkia ole järkevää käydä läpi. Tässä työssä käydään läpi vain hakemistot ja protokollat, jotka ovat olennaisia nykyaikaisissa sovelluksissa ja toteutuksen ja sen suunnittelun kannalta merkityksellisiä.

Luvussa 2 esitellään käyttäjähakemiston käsite, sekä kaksi olennaisinta käyttäjähakemiston tyyppiä. Lisäksi luvussa esitellään identiteettipalvelun käsite, sekä olennaisimpia markkinoiden identiteettipalveluita. Luvussa 3 esitellään tärkeimmät autentikointiprotokollat ja avataan niiden toimintaperiaatteita. Luvussa 4 vertaillaan ja päätetään parhaiten työn tarkoitukseen soveltuva identiteettipalvelu sekä autentikointiprotokolla. Luvussa 5 esitellään autentikointitapahtumien yleisimmät uhat sekä autentikointitapahtuman käsittely verkkosovelluksessa. Luvussa 6 esitellään työssä toteutettu toiminnallisuus. Luvussa 7 esitellään työn tulokset sekä mahdolliset jatkokehitystoimet. Luvussa 8 on työn yhteenveto.

2. KÄYTTÄJÄHAKEMISTO

Active Directory (AD), eli käyttäjähakemisto nimensä mukaisesti säilöö digitaalisia identiteettejä tietokantaan sekä mahdollistaa niiden hakemisen ja hallinnan [1]. Aluksi AD:t olivat vain sisäverkon käyttäjien autentikoimista ja hallinnoimista varten, mutta nykyään on olemassa myös pilvipalvelupohjaisia ratkaisuja, jotka mahdollistavat yhteyden muihin ulkoverkossa sijaitseviin palveluihin.

Tämän työn kannalta oleellisia käyttäjähakemistoja ovat Windows Active Directory ja Azure Active Directory. Näiden kahden käyttäjähakemiston ominaisuuksiin perehdytään seuraavissa kohdissa.

2.1 Windows AD

Windows AD on Microsoftin kehittämä hakemistopalvelu, joka mahdollistaa datan säilömisen ja sen jakamisen AD:n käyttäjille. Active Directoryn verkkoon kirjautuessaan käyttäjällä on pääsy kaikkiin AD:n palveluihin ja tietoihin, joihin hänelle on annettu pääsyoikeus fyysisestä sijainnista riippumatta [2]. Windows Server versioissa 2000 ja 2003 *Active Directory* viittaa hakemistopalveluun [3]. Myöhemmissä versioissa (Windows Server 2008 ja uudemmat) nimi muuttui muotoon *Active Directory Domain Services* (AD DS). Tässä työssä Windows AD kuitenkin viittaa yleisesti Windowsin käyttäjähakemistoihin versiosta riippumatta. LDAPv3 (Lightweight Directory Access Protocol v3 -protokolla) julkaistiin 1997, ja siitä lähtien Microsoft on kehittänyt hakemistopalveluitaan LDAPiin perustuen [1, luku 1]. Tästä seuraa, että ulkoverkossa autentikoimista varten, Windows AD tukee LDAP-protokollaa. LDAP-autentikointiprotokollaa käydään tarkemmin läpi luvussa 3.

Käyttäjähakemistojen käyttöönoton myötä käyttäjätietojen ja tunnuksien varmistamisen menetelmät siirtyivät itsenäisistä sovelluksista sisäverkossa sijaitsevalle palvelimelle. Tätä palvelinta kutsutaan Domain Controlleriksi (DC). DC on palvelin, joka tunnistaa jokaisen käyttäjän ja laitteen lähiverkossa ja voi sen vuoksi välittää tietoturvallisia yhteyksiä kahden osapuolen välillä. DC:n huono puoli oli se, että se toimii vain yrityksen lähiverkon sisällä, eikä voi sen vuoksi autentikoida

käyttäjää ulkoverkossa, esimerkiksi pilvessä ajossa olevaan sovellukseen. [4, luku 2]

Sisäverkon ulkopuolisten tapahtumien yleistyessä kehittäjät ja ylläpitäjät tahtoivat hallinnoida myös yhteistyökumppaneiden ja asiakkaiden sovelluksiin pääsyä Windows AD:n avulla. Microsoft vastasi asiakkaidensa toiveisiin uudella Windows Serverin roolilla Active Directory Federation Services (AD FS) [4, luku 3]. AD FS laajentaa sisäverkon autentikoimisen ulkoverkossa olevia sovelluksia varten [5]. AD FS mahdollistaa tuen mm. useille väitepohjaisille (engl. claims-based) autentikointiprotokollille, proxy-yhteyden sisäverkon ulkopuolelle autentikoimista varten ja AD ja LDAP-attribuuteista pohjautuvan tietueen luomisen. Vaikka palvelu on muuttunut vuosien aikana suuresti, edellä mainitut toiminnallisuudet ovat edelleen oleellisia. Nykyaikaiset AD FS -palvelut tukevat suosituimpia protokollia, kuten luvussa 3 esiteltäviä SAML2.0:aa, OAuth2:ta ja OpenID Connectia. [4, luku 3]

2.2 Azure AD

Azure Active Directory (Azure AD) on Microsoftin kehittämä ja ylläpitämä pilvipalvelupohjainen identiteetin- ja pääsynhallinnan palvelu. Azure AD mahdollistaa autentikoinnin organisaation sisäverkon lisäksi ulkoverkossa oleviin applikaatioihin [6]. Azure AD on suunniteltu olemaan pilvipalveluihin perustuva vastine ”tavalliselle” käyttäjähakemistolle [4, luku 3]. Siitä huolimatta ne ovat silti myös hyvin erilaisia. Tämän työn kontekstissa olennaisin ero käyttäjähakemistojen välillä on se, että ne tukevat eri autentikointiprotokollia [1, luku 1]. Azure AD tarjoaa tuen useimmille luvussa 3 esiteltäville autentikointiprotokollille [7]. Windowsin käyttäjähakemisto joko tukee tai ei tue ulkoverkkoon autentikoimisen mahdollistavia protokollia riippuen AD:n versiosta [1, luku 1].

Azure AD mahdollistaa paikallisen AD:n peilaamisen pilveen, jolloin organisaatiot voivat jatkaa oman AD:n hallintaa normaalisti, mutta samalla käyttäjät saavat Azure AD:n pilvipohjaisten palveluiden hyödyt [1, luku 1]. Ohjelmistokehityksessä on yleistä käyttää Azure AD:ta autentikointia varten käytettävien tietueiden lähteenä. Ennen autentikointia sovellus, johon halutaan autentikoida, pitää kuitenkin konfiguroida Azure AD:lle. [8, luku 1]

2.3 Identity Provider

Identity Provider (IdP) on organisaatioiden käytössä oleva palvelin, joka autentikoi käyttäjän ja välittää tietoa autentikointitapahtumasta ja autentikoinnin kohteesta [9, luku 6]. IdP:n avulla käyttäjähakemiston käyttäjä voidaan siis autentikoida hakemiston tunnuksilla ulkopuoliseen sovellukseen. Autentikointi tapahtuu käyttämällä yksityisellä salausavaimella allekirjoitettua, tietueeksi kutsuttua objektia. Tietueen allekirjoitus varmistetaan sovelluksen puolella konfiguraatiotiedosta saadulla julkisella salausavaimella. Jos tietueen sisältöä on muutettu matkan varrella, se huomattaisiin allekirjoituksen varmistuksen yhteydessä. Näin voidaan varmistaa tietueen tulevan luotettavalta IdP:ltä koskemattomana sovellukselle asti. [4, luku 2]

Autentikointitapahtuman jälkeen IdP luo istunnon käyttäjän ja IdP:n välille ja asettaa tiedon istunnosta käyttäjän selaimen evästeeseen. Jos samaa selainta käytetään toisessa sovelluksessa, joka käyttää samaa IdP:tä autentikoimiseen, käyttäjän ei tarvitse kirjautua sisään uudestaan. Sen sijaan, kun käyttäjän selain ohjataan taas IdP:lle, IdP päättelee evästeestä, että käyttäjällä on voimassa oleva istunto IdP:n kanssa. Tällöin käyttäjän selain ohjataan takaisin sovellukseen tietueiden kanssa ilman, että häntä pyydetään kirjautumaan sisään uudestaan. Tätä olemassa olevan istunnon tunnistamista ja automaattisen sisäänkirjautumisen tekemistä kutsutaan kertakirjautumiseksi (engl. single sign-on). [9, luku 11]

Useimmissa tapauksissa sovellus ja IdP vaativat hieman konfiguraatiota luodakseen keskinäisen luottamussuhteen. Luottamuksella tarkoitetaan tässä yhteydessä sitä, että sovellus luottaa siihen, mitä IdP väittää sovellusta käyttävistä käyttäjistä. Sovellukselle IdP konfiguroidaan asettamalla sovellukselle muun muassa IdP:n tunnus ja salausavain. IdP:lle taas konfiguroidaan esimerkiksi uudelleenohjauksen osoite, jonne selain ohjataan, kun käyttäjä on autentikoitunut IdP:lle. [4, luku 2]

OAuth-protokollan termi IdP:lle on Authorization Server (AS) [10]. OpenID Connect -protokollaan liittyvässä kirjallisuudessa voi nähdä IdP:tä kutsuttavan myös Authorization Serveriksi, mutta virallinen termi on OpenID Provider (OP) [11].

2.4 Identiteettipalvelu

Identiteetti palveluna (Identity as a service, IDaaS) on identiteetin- ja pääsynhallintaa mahdollistava pilvipalvelu. Identiteettipalvelut helpottavat ohjelmistokehittäjien työtä, koska ne mahdollistavat useita erilaisia autentikointitapoja ilman erillistä ohjelmistokehitystä. Koska identiteettipalvelu kykenee suorittamaan käyttäjän autentikoinnin useimpien palveluiden kanssa pienen konfiguraation jälkeen, sitä käytettäessä sovellukseen ei tarvitse tehdä kuin yksi autentikointitapa, joka on identiteettipalvelulta sovellukselle. [8, luku 1]

Lisäksi, koska identiteettipalvelu hoitaa IdP:n kanssa autentikoimisen, ohjelmistokehittäjien mahdollisten virheiden tekemisen riski pienenee. Koska uutta koodia tulee kehityksen yhteydessä vähemmän, mahdollisten ohjelmointivirheiden löytäminen yksinkertaistuu. Edellä mainituista syistä sekä mahdollisten tulevaisuuden laajennuksien vuoksi on järkevää ottaa identiteettipalvelu käyttöön. Tällöin uuden autentikointitavan tai IdP:n lisääminen ei vaadi ylimääräistä kehitystä eikä uusia kirjastoja sovellukseen, vaan tilanteesta selvitään identiteettipalvelun konfiguroinnilla.

Tässä työssä identiteettipalvelulla voitaisiin hallita sallittujen IdP:den lisäksi autentikoineiden profiilien kirjautumisoikeutta. Tällöin voitaisiin estää tietyn käyttäjän ulkoisen identiteetin käyttö sovelluksessa, jos käyttäjän identiteetillä huomataan väärinkäyttöä. Identiteettipalvelua voitaisiin siis hyödyntää pääsynhallinnassa. Käyttöoikeuksien hallintaan identiteettipalvelua ei voida käyttää, koska sovelluksen sisällä käyttäjillä voi olla useita eri rooleja, jotka määrittävät käyttäjän käyttöoikeudet.

2.4.1 Azure AD B2C

Azure AD B2C on julkisille verkkosovelluksille tarkoitettu identiteettipalvelu. Se mahdollistaa erilaisten käyttäjien autentikoinnin useilla protokollilla missä alustalla tahansa ajossa olevaan verkkosovellukseen [12]. Tuettuja protokollia ovat muun muassa SAML, OAuth 2.0 ja OpenID Connect. Oletuksena olevien autentikointimenetelmien lisäksi palveluun voi lisätä omia menetelmiä. Käyttäjät voivat käyttää sosiaalisen median, yrityksen tai natiivin, eli sovellukseen luodun, käyttäjän identiteettejä autentikoidakseen Azure AD B2C -palvelun kanssa. Verkkosovellus pitää ennen käyttöä konfiguroida Azure AD B2C -portaalissa, jossa voi

asettaa muun muassa tietueen väitteet ja kaksivaiheisen tunnistautumisen. [8, luku 1]

2.4.2 AWS Cognito

AWS Cognito on *Amazon Web Services* -pilvipalvelualustan tarjoama identiteettipalvelu, joka mahdollistaa autentikoinnin sosiaalisilla identiteeteillä, OIDC- ja SAML-protokollalla sekä sen hakemistoon luodulla natiivilla käyttäjällä. AWS Cognito luo kirjautumistavasta riippumatta jokaiselle käyttäjälle profiilin hakemistoonsa, jota voi käyttää palvelun ohjelmistokehityspaketin (Software Development Kit, SDK) avulla. SDK, joka mahdollistaa tietueiden käsittelyn, on saatavilla vain JavaScriptille. Identiteetin- ja pääsynhallinnan lisäksi AWS Cognito tarjoaa muokattavan verkkosovelluksen käyttöliittymän, josta käyttäjä voi valita haluamansa tavan autentikoitua. [13]

Kolmannen osapuolen IdP:n autentikointitapahtuman yhteydessä AWS Cognito luo profiilin autentikoidulle käyttäjälle omaan hakemistoonsa. Tämän jälkeen AWS Cognito kykenee itse toimimaan IdP:nä sitä käyttäville sovelluksille. AWS Cogniton IdP tominnallisuus on OIDC:n ja OAuth 2.0:n yhdistelmä, joka tuottaa OIDC:n määrittelemiä tietueita. Cogniton tarjoama käyttöliittymä voidaan ohittaa määrittelemällä halutun IdP:n tunniste Cognitolle silloin, kun selainta uudelleen ohjataan käyttäjän autentikoimiseen. Näin käyttäjä ohjataan suoraan organisaationsa IdP:lle ilman, että hänen tarvitsee itse valita tietty organisaatio. [14]

2.4.3 Okta

Okta on pilvipalveluille identiteettien hallintaan, autentikointiin ja auktorisointiin luotu palvelu, joka tukee muun muassa SAML- ja OIDC-protokollia [15]. Autentikointi on protokollien lisäksi mahdollista sosiaalisilla identiteeteillä ja natiiveilla käyttäjillä. Okta mahdollistaa myös integraation käyttäjähakemistoihin ja LDAP-protokollan käytön. Okta tarjoaa SDK:t useimmille ohjelmointikielille [15].

Okta mahdollistaa LDAPilla autentikoinnin delegoimalla sen käyttäjän organisaation LDAP-palvelimelle, jolloin Oktan ei tarvitse tallentaa käyttäjän kredentiaaleja. LDAP-integraatio vaatii Oktan *LDAP Agent* -palvelun asentamisen AD-palvelimelle. [16]

2.4.4 Auth0

Auth0 on autentikointi-, auktorisointi- ja käyttäjänhallintapalvelu. Auth0 tukee muun muassa protokollia OAuth 2.0, OIDC ja SAML. Autentikointi on protokollien lisäksi mahdollista sosiaalisilla identiteeteillä ja natiiveilla käyttäjillä. [17]

Auth0 mahdollistaa käyttäjähakemiston käyttäjän autentikoinnin LDAPilla. LDAP-integraatio vaatii Auth0:n *LDAP Connector* -palvelun asentamisen AD-palvelimelle. [18]

3. AUTENTIKOINTIPROTOKOLLAT

Autentikointiprotokollien avulla käyttäjä voidaan autentikoida sovellukselle organisaation käyttäjähakemiston avulla. Tällöin käyttäjän ei tarvitse kirjoittaa eikä muistaa sovellukseen asetettuja tunnuksia, vaan tutut organisaation tunnukset kelpaavat. Näin toteutettu autentikointi vapauttaa usein sovelluksen vastuun käyttäjätietojen ylläpitämisestä ja tarkistamisesta. Tämä ulkoinen autentikointi voi nostaa tietoturvaa muun muassa vähentämällä salasanojen uudelleenkäyttöä ja helposti muistettavien salasanojen luontia. Lisäksi ulkoinen autentikointi helpottaa ylläpitäjien työtä niin, että ilman käyttäjähakemistoa käyttäjiltä toisinaan unohtuisi palveluiden salasanoja ja niitä pitäisi ylläpitäjien nollata yhtenään. Protokollia on olemassa muitakin, mutta tässä työssä käydään läpi vain toteutuksen kannalta oleellisimmat.

Tietue (engl. token) sisältää käyttäjähakemiston hallinnoimia tietoja autentikoitavan kohteen identiteetistä ja autentikointitapahtumasta. Tietueet allekirjoitetaan yksityisellä salausavaimella ja allekirjoitukset validoidaan julkisella avaimella [19, luku 1]. Tietue koostuu useista väitteistä (engl. claim) [8, luku 3]. Väitteet ovat avain-arvo-pareja, jotka voivat sisältää tietoa esimerkiksi käyttäjän autentikoineesta IdP:stä tai ajankohdasta. Eri protokollat ja protokollien menetelmät käyttävät rakenteeltaan ja sisällöltään erilaisia tietueita. Protokollia ja niiden tietueiden rakenteita ja sisältöjä käsitellään seuraavissa kohdissa.

Väitepohjaiset (engl. claims-based) protokollat saavat nimensä siitä, että ne välittävät autentikoinnin jälkeen sovellukselle tietueen, joka sisältää useita väitteitä. Väite tarkoittaa käytännössä yhtä käyttäjään tai autentikointitapahtumaan liittyvää nimi-arvo-paria tietueen tietorakenteessa. Seuraavissa kohdissa esiteltävistä protokollista SAML, OAuth ja OIDC ovat väitepohjaisia protokollia. [4, luku 2]

Useat autentikointiprotokollat allekirjoittavat tietueet digitaalisesti ennen niiden toimitusta. Digitaalisessa allekirjoituksessa tietueen lähettäjä luo tietueen sisällön ja yksityisen salausavaimensa avulla tietueeseen allekirjoituksen. Tietueen vastaanottaja voi tarkistaa allekirjoittajan julkisella avaimella, että tietueessa oleva allekirjoitus voidaan purkaa vastaamaan tietueen sisältöä. Koska ainoastaan

allekirjoittajalla on tieto yksityisestä salausavaimesta, allekirjoituksen tarkistamisella voidaan varmistaa datan aitous ja eheys. [20, luku 2]

3.1 LDAP

Lightweight Directory Access Protocol -standardi (LDAP) määrittää viestintäprotokollan sekä käytettävät viestit ja niiden rakenteen, koodauksen ja lähettämisen [21]. LDAP suunniteltiin aluksi vaihtoehtoiseksi tavaksi hallita käyttäjähakemistoja, mutta teknologioiden kehittyessä se muuttui tarkoittamaan myös tietynlaista käyttäjähakemistoarkkitehtuuria [22, luku 1]. Tässä työssä LDAP-lyhenteellä kuitenkin viitataan protokollaan.

Sovellukset, jotka käyttävät LDAP-protokollaa autentikointiin, tavallisesti tekevät vain *bind*-operaation [22, luku 8]. *Bind*-operaation toiminnallisuus sallii autentikointi-informaation vaihtamisen asiakasohjelmiston ja palvelimen välillä [21]. Operaatiota voidaan ajatella autentikointioperaationa. Jos operaatio onnistuu, annetut käyttäjätiedot kuuluivat hakemistossa olevalle käyttäjälle [22, luku 8]. Yhteyden luomisen jälkeen asiakasohjelmisto pystyy oikeuksiensa mukaan lukemaan ja päivittämään palvelimella olevaa informaatiota. LDAP-palvelimen dataobjektien hakuun käytetään niiden yksikäsitteistä yksilöivää nimeä (engl. distinguished name, DN). Nimen lisäksi objekteilla on oma URLinsa, joka koostuu palvelimen osoitteesta, DN:stä ja valinnaisista tarkentavista parametreista. Lopuksi istunto palvelimen kanssa suljetaan *unbind*-operaatiolla. [23, luku 2]

LDAP v3 -standardissa sallitaan asiakasohjelmistojen pyytää vaihtamista suojaamattomasta kommunikaatiosta salattuun TLS-yhteyteen. Salattua yhteyttä käyttävästä LDAP-protokollasta käytetään lyhennettä LDAPS (LDAP over SSL/TLS). LDAPSia tukevat palvelimet ylläpitävät kahta porttia, joista toinen on suojaamattomalle liikenteelle ja toinen suojatulle liikenteelle [22, luku 4]. Jos LDAP-palvelimilla käytetään StartTLS-menetelmää, palvelin tarvitsee vain yhden portin, joka kykenee sekä suojaamattomaan yhteyteen että suojattuun yhteyteen. StartTLS on nykyään standardisoitu menetelmä suojatun yhteyden luomiseksi LDAP-palvelimen kanssa. Suojatun yhteyden luonnissa asiakasohjelmisto varmistaa palvelimen identiteetin palvelimen sertifikaatin ja julkisen salausavaimen avulla [23, luku 2]. Jos yhteyttä ei suojata, autentikoinnin yhteydessä lähetettävä salasana kulkee viestissä selvälukuisena [22, luku 4]. Kun yhteyden turvallisuus on

varmistettu, asiakasohjelmisto lähettää symmetrisen salausavaimen palvelimelle, jota käytetään jatkossa yhteyden viestien salaamiseen palvelimen yksityisen ja julkisen salausavainten sijasta [23, luku 2].

3.2 SAML

Security Assertion Markup Language (SAML) määrittelee syntaksin ja *assertionien*, eli SAMLin tietueiden, prosessoinnin menetelmät [24]. SAML 2.0 on XML-formaattiin perustuva standardi, jota käytetään kertakirjautumisen ja autentikoinnin mahdollistamiseen [9, luku 7]. SAML esittelee kaksi uutta teknologiaa: määrittelemänsä protokollan ja protokollan viestien välittämän *assertionin*, esitysmuodon [1, luku 14].

Tavallisen väitepohjaisen protokollan tavoin SAML kokoaa väitteet tietueeseen, allekirjoittaa sen ja lähettää sen autentikoinnin aloittaneelle sovellukselle [8, luku 4]. SAML mahdollistaa myös tietueen väitteiden tai koko tietueen salaamisen [24, s. 73]. SAML tietueet eivät kuitenkaan ole modernien protokollien suosiossa, koska niiden formaatti johtaa suuriin tietueisiin, joiden salaaminen ja allekirjoittaminen on työlästä ja välitys vaatii paljon kaistanleveyttä [1, luku 14].

Jotta sovelluksen ja IdP:n välinen keskustelu on mahdollista, ne on konfiguroitava luottamaan toisiinsa. Konfiguroitavaan dataan kuuluu muun muassa uudelleenohjauksien URL-osoitteet sekä IdP:n sertifikaatti, jotta viestien digitaaliset allekirjoitukset voidaan tarkistaa. SAML-autentikoinnin työnkulku alkaa käyttäjän toiveesta autentikoitua sovellukselle. Sovellus ohjaa käyttäjän selaimen IdP:lle, joka autentikoi käyttäjän käyttäjänimellä ja salasanalla. Kun käyttäjä on autentikoitunut IdP:lle, selain ohjataan takaisin sovellukselle tietueen tai virheilmoituksen kera. [9, luku 7]

3.3 OAuth

OAuth 2.0 suunniteltiin, jotta sovellus voi autentikoida kolmansien osapuolien rajapintojen kutsut [9, luku 5]. OAuthin tavallisessa käyttötapauksessa käytettävä sovellus tarvitsee pääsyn kolmannen osapuolen tietoihin, jotka sovellusta käyttävä käyttäjä omistaa. OAuth ei itsessään tarjoa autentikointiin menetelmiä, vaan ratkaisun delegoituun auktorisointiin. OAuth ei ole siis autentikointiin tarkoitettu protokolla, vaikka sillä pystyykin kehittämään autentikointiprotokollan [25, luku 1].

3.3.1 Tietueet

OAuth 2.0 määrittelee käyttöoikeustietueen (engl. Access token), päivitystietueen (engl. Refresh token) sekä valtuutuskoodin (engl. Authorization code). Käyttöoikeustietue kuvaa sovelluksen lupaa käsitellä kolmannen osapuolen rajapintaa. Käyttöoikeustietue on tarkoitettu vain kolmannen osapuolen verkkosovelluksen rajapinnan käyttöön, eikä sillä ole tarkoitus viestiä tietoa käyttäjästä tai autentikointitapahtumasta. Päivitystietueella voidaan hakea AS:ltä uusi käyttöoikeustietue, jos nykyinen on vanhentunut. Päivitystietuetta käytettäessä tietue voidaan hakea autentikoimatta käyttäjää uudestaan. Valtuutuskoodilla, joka on kertakäyttöinen, voidaan hakea pääsyavain ja päivitysavain AS:ltä. [9, luku 5]

Vaikka OAuth-standardi [10] ei erikseen määrittele tietueiden rakennetta, niiden formaatti on useimmiten JSON Web Token (JWT), koska se tarjoaa tehokkaan ja turvallisen tavan tiedonsiirtoon [8, luku 5]. JWT on nimensä mukaisesti JSON-muotoon rakennettu, kompakti, väitteiden esittämiseen tarkoitettu rakenne [26]. JWT:n väitteiden kokoelmaa käytetään JSON Web Signature (JWS) -rakenteen tai JSON Web Encryption (JWE) -rakenteen tietosisältönä. Tällöin JWT voidaan digitaalisesti allekirjoittaa ja/tai salata. JWS mahdollistaa tietosisältönsä turvaamisen digitaalisella allekirjoituksella [27]. Allekirjoituksen tarkistamisella sovellus varmistaa, että tietueen sisältö on tullut muuttumattomana sovellukselle. JWE puolestaan mahdollistaa tietosisällön salaamisen kryptografisilla algoritmeilla [28].

3.3.2 Päätepisteet

OAuth 2.0 määrittelee kaksi päätepistettä (engl. endpoint) AS:lle. Päätepisteellä tarkoitetaan tässä yhteydessä OAuth-palvelimelle tarkoin määriteltyä toiminnallisuutta, joka on saatavilla palvelimen tietystä URL-osoitteesta. Määritellyt päätepisteet ovat valtuutuspäätepiste (engl. authorization endpoint) ja tietuepäätepiste (engl. token endpoint). [10]

Valtuutuspäätepistettä käytetään resurssin omistajan identiteetin varmistamiseen eli toisin sanoen käyttäjän autentikoimiseen. Kun käyttäjä on autentikoitu, valtuutuspäätepisteellä AS voi varmistaa käyttäjältä, haluaako tämä valtuuttaa sovellukselle pääsyn kolmannen osapuolen tietoihin. Valtuutuspäätepistettä

käytetään *Authorization code grant* -menetelmässä, joka esitellään seuraavassa aliluvussa. [10]

Tietuepäätepistettä käytetään tietueiden hakemiseen päivitystietueen tai valtuutuskoodin avulla. Myös tietuepäätepistettä käytetään *Authorization code grant* -menetelmässä. Tietuepäätepistettä käytettäessä, luottamuksellisen asiakasohjelmiston, joka esitellään myöhemmässä aliluvussa, tulee autentikoitua AS:lle sen määrittelemällä tavalla. OAuth 2.0 -standardin mukaan AS:n on tuettava *HTTP Basic* -autentikaatiota, jossa HTTP-kutsun *Authorization*-otsikkokenttään asetetaan Base64-koodattu kaksoispisteellä erotettu asiakasohjelmiston asiakastunnisteen ja asiakassalaisuuden yhdistelmä. AS saattaa tukea myös asiakastunnisteen ja -salaisuuden vastaanottamista HTTP-kutsun kehyksessä. [10]

3.3.3 Tunnistautumisen prosessi

OAuthin työnkulku alkaa, kun AS:lle lähetetään auktorisointipyyntö, jossa määritellään haluttu data kolmannelta osapuolelta [9, luku 5]. Käyttäjä autentikoituu AS:lle, joka tavallisimmin esittää käyttäjälle valinnan hyväksyä tai hylätä sovelluksen pääsyn käyttäjän dataan [25, luku 1]. Auktorisoinnista AS:ltä saatua käyttöoikeustietuetta käytetään, jotta päästään toisen sovelluksen dataan käsiksi [9, luku 5]. OAuthin autentikointivaiheessa tarkistetaan, että käyttäjällä on oikeus pyytää haluamaansa dataa kolmannelta osapuolelta. Käyttöoikeustietueen oikeuksien rajaamiseksi on mahdollista asettaa kutsuihin *scope*-parametri, jolla voidaan määritellä tietuella saatavien oikeuksien laajuus [10]. Useimmissa tapauksissa AS on osa sovellusta, jonka resursseja tarvitaan [4, luku 2]. AS on siis kolmannen osapuolen luottama palvelu, joka auktorisoi kolmannen osapuolen rajapintaan tulevat kutsut [19, luku 1].

Yksi OAuthin menetelmistä on *Authorization code grant* [9, luku 5]. Tässä menetelmässä käyttäjä autentikoidaan AS:lle ja käyttäjä antaa suostumuksen siihen, että sovellus saa hakea toisen sovelluksen rajapinnan kautta käyttäjän omistamaa dataa. Varmistuksien jälkeen AS palauttaa käyttäjän takaisin alkuperäisen sovellukseen valtuutuskoodin kanssa. Sovellus vaihtaa valtuutuskoodin käyttöoikeustietueeseen AS:n kanssa. Käyttöoikeustietueen kanssa sovellus voi kutsua kolmannen osapuolen sovelluksen palvelurajapintaa. Kolmas osapuoli varmistaa, että kutsujalla on tarvittavat käyttöoikeudet käyttöoikeustietueesta.

Authorization code grant on ainoa tämän työn kannalta mielenkiintoinen OAuthin tarjoama mentelmä, koska se on ainoa menetelmä, jota on järkevä käyttää luottamuksellisten verkkosovelluksien kanssa, jotka kykenevät turvallisesti säilömään arkaluontoisia konfiguraatiotietoja [4, luku 2].

3.3.4 State-parametri

Lisäturvallisuutta OAuth kutsuihin saadaan, kun käytetään ensimmäisessä kutsussa arvaamatonta, eli pitkää ja satunnaisesti luotua, *state*-parametria. AS palauttaa tämän parametrin muuttumattomana, kun uudelleenohjaus takaisin sovellukseen toteutuu. Jos parametri on erilainen tai puuttuu, sovellus ei voi luottaa vastaukseen ja lopettaa autentikointitapahtuman. Näin vältytään siltä, että hyökkääjä pystyisi muun muassa asettamaan oman valtuutuskoodinsa käyttäjän selaimeen ja saisi näin pääsyn toisen omistamiin resursseihin. [25, luku 7] Autentikointitapahtuman mahdollisia uhkia käsitellään tarkemmin luvussa 5.

3.3.5 Julkinen ja luottamuksellinen asiakasohjelmisto

OAuth 2.0 määrittelee julkisen asiakasohjelmiston (engl. public client) olevan sovellus, joka on ajossa pääosin käyttäjän laitteella tai selaimessa, eikä voi turvallisesti säilöä dataa, joka auttaisi autentikoitaessa AS:lle. Luottamuksellinen asiakasohjelmisto (engl. confidential client), joka on ajossa palvelimella käyttäjän laitteen sijasta, on julkisen vastakohta, eli sille on mahdollista turvallisesti säilöä dataa, jota se voi käyttää autentikoitaessa AS:lle. Yksi määritellyistä luottamuksellisen asiakasohjelmiston profiileista on verkkosovellus, joka on ajossa suojatulla palvelimella. [9, luku 5]

Pääasiallinen ero on, että luottamuksellinen ohjelmisto pystyy säilyttämään asiakastunnisteen (engl. client id) ja asiakassalaisuuden (engl. client secret) turvallisesti. Tämä johtuu siitä, että ohjelmisto on ajossa palvelimella ja vain ohjelmisto pääsee käsiksi palvelimella säilöttyyn dataan. Verkkosovelluksissa asiakastunniste on saatavilla selaimen kautta, koska sitä käytetään OAuth ja OpenID Connect autentikoinneissa uudelleenohjauksen URL-osoitteessa, mutta asiakassalaisuus ei koskaan päädy selaimen tietoon, eikä täten käyttäjien tietoon. [8, luku 3]

3.4 OpenID Connect

OpenID Connect (OIDC) on OAuth 2.0:n laajennus, joka mahdollistaa sen, että sovellus voi varmistaa käyttäjän identiteetin AS:lle tehdyn autentikoinnin perusteella. Käyttäjän identiteettitietojen lisäksi OIDC:n avulla saadaan tietoja autentikointitapahtumasta. [11]

3.4.1 Laajennukset OAuthiin

AS kokoaa edellä mainitut tiedot väitteiksi uuteen tietueeseen, tunnistetietueeseen (engl. ID Token), joka määrittelyn mukaan tulee olla JWT-formaatissa [11]. Tällä tunnistetietueen laajennuksella OIDC mahdollistaa käyttäjien autentikoimisen [9, luku 6]. OIDC käyttää IdP:nä OIDC:n toteuttavaa OAuth 2.0 AS:ää, jota kutsutaan OpenID Provideriksi (OP). Koska OIDC on OAuth 2.0 laajennus, se käyttää samoja määrittelyitä asiakasohjelmistotyypeistä, päätepisteistä ja muista tietueista. Tunnistetietueen lisäksi OIDC lisää OAuthin AS:lle *discovery*-päätepisteen, josta voidaan hakea muun muassa muiden päätepisteiden osoitteet, tietueiden allekirjoituksen validointiin käytettävät avaimet, sekä esimerkiksi tuetut autentikointimenetelmät. [29, s. 39]

Jos OP:t tahtovat mahdollistaa *discovery*-toiminnallisuuden, tulee niiden sallia konfiguraatiotietojensa haku tarkoin määritellyllä tavalla. OIDC-määrittely määrittelee, että *discovery* -päätepisteen tulee löytyä polusta */.well-known/openid-configuration* ja sen tulee palauttaa JSON-muodossa oleva dokumentti. OIDC-määrittely kuvaa myös kaikki kentät, jotka konfiguraatiotietojen voi tai pitää sisältää. Kuvassa 2 on esimerkki siitä, mitä AWS Cogniton avulla luodun OP:n *discovery*-päätepiste voi palauttaa. Muiden päätepisteiden osoitteiden lisäksi konfiguraatiotiedoista saadaan osoite *UserInfo*-päätepisteelle, josta voidaan hakea tietoa autentikoineesta käyttäjästä käyttöoikeustietueen avulla. Kuvan rivin 8 *response_types_supported*-kenttä määrittelee mahdolliset autentikoinnin vastauksen menetelmät, joista *”code”* tarkoittaa valtuutuskoodin palauttamista. Seuraavassa aliluvussa opitaan, minkä takia tämä on tarpeellinen toteutettavan autentikoinnin kannalta. Rivin 12 *scopes_supported*-kenttä kertoo, mitä OAuthin *scope*-parametreja OP:n käyttöoikeustietue tukee. Rivin 18 *subject_types_supported*-kenttä määrittelee, missä muodossa käyttäjän tunniste annetaan tietueessa. Rivin 22 *token_endpoint_auth_methods_supported*-kenttä kertoo, millä eri tavoin

asiakasohjelmisto voi autentikoitua OP:lle. Tämä OP tukee siis molempia alilu-vussa 3.3.2 esiteltyjä autentikointitapoja. [30]

```
{
  "authorization_endpoint": "https://test-op.auth.eu-north-1.amazoncognito.com/oauth2/authorize",
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "issuer": "https://cognito-idp.eu-north-1.amazonaws.com/eu-north-1_<AWS_ID>",
  "jwks_uri": "https://cognito-idp.eu-north-1.amazonaws.com/eu-north-1_<AWS_ID>/.well-known/jwks.json",
  "response_types_supported": [
    "code",
    "token"
  ],
  "scopes_supported": [
    "openid",
    "email",
    "phone",
    "profile"
  ],
  "subject_types_supported": [
    "public"
  ],
  "token_endpoint": "https://test-op.auth.eu-north-1.amazoncognito.com/oauth2/token",
  "token_endpoint_auth_methods_supported": [
    "client_secret_basic",
    "client_secret_post"
  ],
  "userinfo_endpoint": "https://test-op.auth.eu-north-1.amazoncognito.com/oauth2/userInfo"
}
```

Kuva 2: AWS Cogniton konfiguraatiotiedot

3.4.2 Tunnistautumisen prosessi

OIDC:n työnkulku on hyvin samantapainen kuin OAuthissa. Ensin käyttäjän au-tentikoinnin tarve huomataan ja ohjataan selain OP:lle. OP autentikoi käyttäjän ja ohjaa selaimen takaisin sovellukselle. Autentikoinnin jälkeen OP toimittaa sovel-lukselle tietueet siten, kuin käytetty menetelmä määrittelee. [9, luku 6]

Taulukko 1: OIDC Autentikointimenetelmät [11]

Property	Authorization Code Flow	Implicit Flow	Hybrid Flow
All tokens returned from Authorization Endpoint	no	yes	no
All tokens returned from Token Endpoint	yes	no	no
Tokens not revealed to User Agent	yes	no	no
Client can be authenticated	yes	no	yes
Refresh Token possible	yes	no	yes
Communication in one round trip	no	yes	no
Most communication server-to-server	yes	no	varies

Taulukossa 1 on esitelty OIDC:n standardissa määritellyt autentikointimenetelmät. Taulukosta voidaan huomata, että *Implicit Flow* ja *Hybrid Flow* -menetelmissä tietueiden sisältö saatetaan selaimen tietoisuuteen ja kommunikointi tapahtuu ainakin osittain selaimen kautta. Tästä seuraa, että *Authorization code flow* on ainoa, jossa kommunikointi tapahtuu palvelimelta palvelimelle paljastamatta tietueita käyttäjän selaimelle. Tästä ominaisuudesta on se hyöty, että tietueet eivät voi päätyä ei-halutuille toimijoille eli mahdollisille hyökkääjille käyttäjän selaimen kautta. *Authorization code flow* on toimiva ratkaisu, jos sovellus voidaan luokitella luottamukselliseksi asiakasohjelmistoksi. [11]

Koska tunnistetietue ei tule selaimelle, se ei pääse sitä kautta vääriin käsiin. Selaimelle tulevaa tunnistetietuetta voidaan yrittää väärentää tai peukaloida, mutta palvelimelta tulevan tunnistetietueen eheydestä voidaan olla varmoja sen tullessa turvallisen ja luotetun TLS-yhteyden kautta. [4, luku 2]

Tästä syystä ainoa kiinnostava OIDC:n menetelmä on *Authorization code flow*. Se on samankaltainen, kuin OAuthin *Authorization code grant* -menetelmä mutta eroaa kuitenkin siten, että kutsut OP:lle ovat tarkemmin määriteltyjä ja käyttöoikeustietueen ja päivitystietueen lisäksi saadaan tunnistetietue. [9, luku 6]

3.4.3 Tunnistetietue

Normaalin TLS-salausprotokollan lisäksi tunnistetietuetta suojaa OP:n tekemä allekirjoitus [25, luku 13]. Suojatakseen itseään yleisimmiltä hyökkäyksiltä, sovelluksen tulee tarkistaa tunnistetietueen sisältö ja allekirjoitus JWT:n määrittelyn mukaisesti. Tietueen tarkistamista avataan tarkemmin aliluvussa 5.1. Allekirjoituksen lisäksi OIDC määrittää parametrin, jolla vähennetään hyökkäyksiä, jossa tietuetta käytettäisiin uudelleen [11]. Tämä *nonce*-parametri on sovelluksen asettama, turvallisuutta lisäävä parametri, joka lähetetään OP:lle sovelluksen lähettämän autentikointipyynnön mukana. Jos tietue ei sisällä *nonce*-parametria samalla arvolla, kun se saapuu sovellukselle vastauksena autentikointitapahtumasta, ei voida varmistua tietueen koskemattomuudesta. [25, luku 13]

4. VAIHTOEHTOJEN VERTAILU

Edellisissä luvuissa esiteltyjen identiteettipalveluiden ja protokollien ominaisuuksia listataan taulukoihin ja vertaillaan keskenään. Vertailujen lopuksi päätetään tämän työn käyttötarkoitukseen ominaisuuksiltaan sopivimmat vaihtoehdot.

4.1 Identiteettipalvelu

Identiteettipalveluiden arvioinnissa keskitytään helppokäyttöisyyteen, yhteensopivuuteen, turvallisuuteen, tuettuihin protokolliin ja kustannukseen. Lopullinen päätös valittavasta palvelusta perustellaan sen ominaisuuksien sopivuudesta tämän työn käyttötarkoitukseen. Taulukossa 2 on esitelty joitain tärkeitä identiteettipalveluiden ominaisuuksia.

Taulukko 2: Identiteettipalveluiden ominaisuudet

	AzureAD B2C	**AWS Cognito**	**Okta**	**Auth0**
LDAP-tuki	Ei	Ei	Asennettavan laajennuksen kanssa	Asennettavan laajennuksen kanssa
SAML-tuki	Kyllä	Kyllä	Kyllä	Kyllä
OAuth-tuki	Kyllä	Kyllä	Kyllä	Kyllä
OIDC-tuki	Kyllä	Kyllä	Kyllä	Kyllä
Nonce-parametrin generointi	Manuaalisesti	Automaattisesti	Manuaalisesti	Manuaalisesti
Autentikointitapahtumien lokitus	Kyllä	Kyllä	Kyllä	Kyllä

Palvelun käyttöliittymän muokkaus	Kyllä	Kyllä, myös täysin oma käyttöliittymä mahdollinen	Kyllä	Kyllä
Hinta	$0.00325–0.01625 per käyttäjä per kuukausi (50000 ilmaiskäyttäjää)	$0.015–0.050 per käyttäjä per kuukausi (50 ilmaiskäyttäjää)	$23–228 per kuukausi	$23–228 per kuukausi

4.1.1 LDAP-tuki

Kuten kohdan 3.5 esittelyissä tuli ilmi, Okta ja Auth0 tukevat LDAP-protokollaa vain, jos LDAP-palvelimelle saadaan asennettua palveluiden omat laajennukset. Laajennuksien asentaminen jokaiselle uuden asiakkaan palvelimelle ei kuitenkaan ole realistinen vaatimus. Lisäksi Auth0:n dokumentaatiossa [18] on erikseen maininta, että laajennus ei ole tarkoitettu käytettäväksi asiakkaiden AD-palvelimille. Oktan dokumentaatiossa ei ole erikseen samanlaista mainintaa, mutta oletettavasti sekin on tarkoitettu organisaation oman käyttäjähakemiston laajentamiseen, ei ulkopuolisten hakemistojen integraatioon.

4.1.2 Nonce-parametri

Autentikointitapahtuman turvallisuuden kannalta on tärkeää, että kutsu sisältää *nonce*-parametrin. Azure AD B2C:n dokumentaatio mainitsee, että *nonce*-parametrin luonti on sovelluksen vastuulla [31]. AWS Cognito luo *nonce*-parametrin automaattisesti, jos sitä ei sovellukselta autentikointia aloitettaessa tullut [32]. Cogniton luoma *nonce*-parametri suojaa käyttäjän IdP:tä toistohyökkäyksiltä. Oktan sivuilla ei ollut mainintaa *nonce*-parametrin luomisesta, joten on oletettava, ettei se tapahdu automaattisesti. Auth0:n dokumentaatioissa on mainintoja vain siitä, että tietyt Auth0:n tarjoamat ohjelmistokehityskirjastot luovat *nonce*-parametrin automaattisesti [33].

4.1.3 Käyttöliittymä

Kaikkien identiteettipalveluiden tarjoamia käyttöliittymiä on mahdollista muokata haluamansa näköisiksi, usein muokkaamalla sivujen .css ja HTML-templaatteja. AWS Cognito mahdollistaa myös sen tarjoaman käyttöliittymän sijasta täysin oman käyttöliittymän käytön [34]. Käyttöliittymän muokattavuus on tärkeätä muun muassa siksi, että käyttäjät tunnistavat olevansa saman järjestelmän sivuilla.

4.1.4 Kustannukset

Palveluiden ilmaisten käyttäjien määrä vaihtelee välillä 50–50000, jonka jälkeen tarvitsee maksaa kuukausittaisista käyttäjistä. Azure AD B2C:n ilmaisen 50000 käyttäjämäärän ylittäessä hinta käyttäjää kohden on valitusta palvelusta riippuen $0.00325–0.01625 [35]. Tämä on hiukan harhaanjohtavasti kirjattu Azure AD:n dokumentaatioon, sillä ulkoisten identiteettien käyttö vaatii vähintäänkin Azure AD Premium P1 -tilauksen, joka maksaa $6 jokaista käyttäjää kohden kuukaudessa [36]. AWS Cogniton avulla ensimmäiset 50 OIDC:llä autentikoituvaa käyttäjää kuukautta kohden ovat ilmaisia, jonka jälkeen hinta käyttäjää kohden on valitusta palvelusta riippuen $0.015–0.050 [37]. Okta ja Auth0 käyttävät samaa hinnoittelua palveluissaan [38]. Ennalta asetetut hinnat löytyvät palveluille 10000 käyttäjään asti, jonka jälkeen hinnasta täytyy erikseen sopia [39]. Palvelun hinta määräytyy valitun käyttäjämäärän mukaan väliltä $23–228 kuukautta kohden.

4.1.5 Standardit

Kaikki identiteettipalvelut noudattavat useita standardeja liittyen tietoturvallisuuteen. Esimerkiksi AWS Cognito ja Azure AD B2C molemmat noudattavat muun muassa HITRUST, ISO/IEC-27001, -27017, -27018, -27701, -22301 ja -9001 standardeja [40], [41]. Okta ja Auth0 noudattavat muun muassa ISO/IEC-27001, -27018 ja SOC 2 Tyyppi 2 -standardeja [42,43]. Kaikki palvelut noudattavat myös GDPR-asetusta.

4.1.6 Yhteenveto

Kaikki identiteettipalvelut mahdollistavat keskenään pitkälti samat toiminnallisuudet ja protokollat, ja kuten taulukosta voidaan huomata, identiteettipalveluiden väliset erot ovat kustannuksia lukuun ottamatta pieniä. Azure AD B2C on palveluista edullisin, jos palvelun käyttöönotto ei aiheuta lisäkuluja. AWS Cognito

erottuu joukosta automaattisella *noncen* generoinnilla ja sillä, että käyttöliittymän toteutukseen voi käyttää palvelun käyttöliittymän lisäksi omaa käyttöliittymää. Lisäksi AWS:n pilvipalvelualusta on nykyisellääänkin jo projektissa käytössä. Tästä syystä on helppoa turvautua AWS:n palveluun, eikä täysin uuden alustan opiskeluun, tutustumiseen ja konfiguroimiseen tarvitse käyttää aikaa ja rahaa.

4.2 Protokolla

Protokollien arvioinnissa keskitytään helppokäyttöisyyteen, yhteensopivuuteen ja turvallisuuteen. Tämän lisäksi on huomioitava, mitä protokollia integroitavat käyttäjähakemistot tukevat. Taulukossa 3 on esitelty joitain tärkeitä autentikointiprotokollien ominaisuuksia.

Taulukko 3: Protokollien ominaisuuksia

	LDAP	**SAML 2.0**	**OAuth 2.0**	**OIDC**
Autentikointi	Kyllä	Kyllä	Ei	Kyllä
Delegoitu auktorisointi	Ei	Ei	Kyllä	Kyllä
Datan salaus	Tuki TLS-yhteydelle	HTTPS + salaus IdP:n salausavaimilla	HTTPS + salaus toteutuksesta riippuen	HTTPS + salaus OP:n salausavaimilla
Datan digitaalinen allekirjoitus	Ei	Kyllä	Toteutuksesta riippuen	Kyllä
Integrointi identiteettipalveluun	Palvelimelle asennettavan laajennuksen kanssa	Kyllä	Kyllä	Kyllä
Määrittelee tietueen rakenteen	Ei	Kyllä	Ei	Kyllä

Kuten aiemmissa luvuissa kävi ilmi, OAuth on ainoa protokolla, joka ei tue käyttäjän autentikoimista, vaan ainoastaan auktorisoinnin delegoimista sovelluksille. Koska OIDC on OAuthin laajennus, se säilyttää kyvyn auktorisoinnin delegointiin, jonka lisäksi se mahdollistaa lisäämänsä tunnistetietueen avulla käyttäjän autentikoinnin.

LDAP on autentikoinnin mahdollistavista protokollista ainoa, joka ei määrittele salaamista mahdollisen TLS-yhteyden muodostamisen lisäksi. OAuth ei myöskään määrittele allekirjoitukseen tai salaukseen liittyvää menetelmää, vaan sen toteutus on kehittäjien vastuulla [10].

LDAP ei määrittele OIDCn tai SAMLin tavoin tietueita, mutta se kuitenkin määrittelee eri kutsujen vastauksien rakenteen [21]. Tällä ei toisaalta ole merkitystä, koska ainoa kiinnostava operaatio LDAP-protokollassa on ”autentikointi” eli *bind*-operaatio, jonka vastauksesta halutaan vain tietää, onnistuiko operaatio. OAuth ei erikseen määrittele tietueen rakennetta, vaan sen toteutus on kehittäjien vastuulla [10].

Koska järjestelmään halutaan autentikoida käyttäjä sisäänkirjautumista varten, OAuth 2.0 ei ole sopiva protokolla. Jäljellä olevista protokollista OIDC ja SAML mahdollistavat tietueiden allekirjoittamisen. SAMLin huono puoli on se, että sen tietueiden koko on suurempi ja niiden allekirjoittaminen vaatii sen vuoksi enemmän resursseja kuin OIDC:n tietue.

Edellä mainituista syistä käytettäväksi protokollaksi tulee priorisoida OIDC:tä tilanteen sen salliessa. Lisäksi, koska OIDC:n pohjana on OAuth 2.0, se mahdollistaa tulevaisuudessa myös sovelluksen delegoidun auktorisoinnin helposti, jos sille syntyy tarve. Vaikka LDAP-protokollan turvallisuusominaisuudet ovat heikompia kuin uudempien protokollien, sitä on kuitenkin käytettävä siinä tapauksessa, kun autentikoitava järjestelmä ei tue muita protokollia. LDAP-protokollaa pitää siis käyttää vanhojen Windows AD -käyttäjähakemistojen kanssa, jos ne eivät muita protokollia tue.

5. YLEISET AUTENTIKOINNIN UHAT

Koska autentikoinnin kommunikaatio tapahtuu julkisen verkon yli, riski hyökkäyksen kohteeksi joutumisesta on olemassa. Tässä luvussa käydään läpi tämän työn kannalta oleellisimmat hyökkäykset. Hyökkäyksien yhteydessä esitellään mahdolliset vastatoimet, joiden avulla voidaan välttää tai vähentää hyökkäyksien kohteeksi joutumista.

5.1 Uhat ja vastatoimet

Väliintulohyökkäyksessä (engl. man-in-the-middle attack, MITM) hyökkääjä kaappaa kahden osapuolen välisen yhteyden hallintaansa. Huonosti suojatuissa yhteyksissä osallistuvat osapuolet eivät huomaa, jos yhteys on kaapattu. Hyökkääjä voi kaapatulta yhteydeltä lukea ja kirjoittaa kutsujen tietoja. Tämä voi johtaa muun muassa salasanojen vääriin käsiin vuotamiseen. Hyökkäykseltä suojautuessa on tärkeätä käyttää yhteyden salausta ja ohjelmistojen välistä autentikointia. Esimerkiksi HTTPS-protokolla käyttää TLS-protokollaa sertifikaattien kanssa yhteyden suojaamiseksi. Jotta hyökkääjä voisi purkaa yhteyden salauksen, hänen tarvitsisi saada haltuunsa yhteyden salaamiseen käytetyt avaimet. [44]

Toistohyökkäyksessä (engl. replay attack) hyökkääjä toistaa sallitun osapuolen aikaisemman kutsun palvelimelle, jolloin vastaanottava palvelin luulee sen tulleen luotettavasta lähteestä [44]. Jos hyökkääjä saa käsiinsä aikaisemmin tehdyn autentikoinnin HTTP-kutsuja, se voi yrittää toistaa kutsut saadakseen pääsyn kohteen resursseihin [45]. Istunnon yksilöimiseen käytettävien tunnisteiden lisäksi hyökkäystä voi estää lyhyillä tietueiden voimassaoloajalla, säätämällä tietueet kertakäyttöisiksi tai käyttämällä kertakäyttöisiä salasanoja [44]. Lisäksi toistohyökkäyksien estämiseksi asiakasohjelmisto voi sisällyttää *nonce*-parametrin kutsuihin, jolloin toistettua kutsua ei hyväksytä, jos *nonce*n-parametrin arvoa on käytetty jo aiemmin. AS voi myös kutsujen aikaleimoja tarkastelemalla arvioida kutsun oikeellisuutta. Toistohyökkäyksien tekeminen on helpompaa ja niiden riski on suurempi, jos käytetään selainpohjaista menetelmää, koska selainpohjaiset menetelmät saattavat autentikoinnin aikana protokollan parametrin selaimen tietoon. Tämän vuoksi selaimen tietoon saatetaan ainoastaan lyhyen

voimassaoloajan omaava valtuutuskoodi, joka voidaan vaihtaa tietueisiin asiakasohjelmiston ja IdP:n palvelinten välisen yhteyden avulla. [45]

CSRF-hyökkäys (engl. Cross-site request forgery) tapahtuu, jos hyökkääjä tai käyttäjän laitteella oleva ohjelma saa käyttäjän selaimen suorittamaan tietyn ei-toivotun toiminnon sivustolla, jossa käyttäjä on autentikoituna [44]. Hyökkäys voidaan estää käyttämällä OAuthin *state*-parametria kutsuissa. CSRF-hyökkäyksien estämiseen käytetyn *state*-parametrin tulee olla sellainen, jota hyökkääjä ei kykene arvaamaan. Arvaamisen estäminen on määritelty niin, että todennäköisyys sille, että arvon saa arvaamalla oikein, on oltava korkeintaan 2^{-128} ja tulisi olla korkeintaan 2^{-160}. Tämä tarkoittaa siis sitä, että standardin mukainen arvo on vähintään 32 merkkiä pitkä kirjaimista ja numeroista koostuva merkkijono, mutta mieluiten vähintään 40 merkkiä pitkä. [10]

Avoimeksi jätetty uudelleenohjauksen osoite mahdollistaa hyökkääjän muokata osoitetta niin, että se läpäisee AS:n tarkastuksen, mutta lähettää valtuutuskoodin hyökkääjän määrittelemään osoitteeseen, jota hyökkääjä hallitsee. Tällöin hyökkääjä saisi siis haltuunsa asiakasohjelmistolle tarkoitetun koodin ja voisi käyttää sitä tietueiden hakemiseen. AS:t voivat estää tämän hyökkäyksen vaatimalla kokonaista uudelleenohjauksen osoitetta asiakasohjelmistoilta. Tämän ja muiden samankaltaisten koodien kaappaushyökkäyksien vuoksi tulee koodien voimassaoloaika asettaa riittävän lyhyeksi. Lisäksi AS:n tulisi sallia yhden valtuutuskoodin käyttö vain kerran. [45]

Hyökkääjä voi yrittää varastaa käyttäjän autentikointitunnukset clickjacking-hyökkäyksellä [45]. Tässä hyökkäyksessä käyttäjän selaimelle ladataan käyttäjälle tutun näköinen sivu, mutta sen päälle on luotu näkymätön iFrame-komponentti. Tämä komponentti mahdollistaa käyttäjän klikkauksen kaappaamisen, jolloin esimerkiksi käyttäjän autentikointipyyntö voidaan uudelleenohjata hyökkääjän hallitsemalle sivustolle. Jos käyttäjä ei huomaa olevansa vieraalla sivustolla, hyökkääjä voi kaapata käyttäjän tunnukset. Useimmilla nykyaikaisilla selaimilla iFrame-komponenttien käyttö voidaan estää käyttämällä verkkosivulla *X-FRAME-OPTIONS*-otsikkoa, jolla voidaan joko estää iFrame-komponettien käytön kokonaan tai toisten sivustojen lisäämän iFrame-komponentin käytön. Vanhemmille selaimille voidaan käyttää JavaScriptiä iFrame-komponenttien toimintojen estämiseksi, mutta tämä ei välttämättä toimi kaikilla selaimilla. [10] Tämä

haavoittuvuus on olennainen IdP:n toteutuksen kannalta ja täten olennainen vain siltä osin, että AWS Cogniton käyttöliittymästä täytyy varmistaa otsikon löytyminen.

Hyökkääjä voi päätellä salauksen purkamiseen ja allekirjoituksen validoimiseen kuluvan ajan tarkastelemalla vastaavia onnistuneita ja epäonnistuneita operaatioita. Tällä tiedolla hyökkääjä saattaa pystyä päättelemään salausavaimesta tiettyjä ominaisuuksia. Tämän hyökkäyksen estämiseksi asiakasohjelmiston ei validointivirheen takia tulisi välittömästi keskeyttää validointiprosessia. Jotta hyökkääjä ei pystyisi ajoittamaan salauksien ja allekirjoitusten validoimiseen kuluvaa aikaa, tulisi prosessointi suorittaa virheestä huolimatta loppuun. [11]

Sen varalta, että hyökkääjä pääsee käsiksi järjestelmän lähdekoodiin, arkaluontoiset tiedot eivät saa olla lähdekoodiin kirjoitettuja [45]. Arkaluontoisella tiedolla tarkoitetaan tässä yhteydessä autentikoinnin konfiguraatiotietoja tai esimerkiksi sovelluksen tietokannan tunnuksia. Tietojen vuotaminen hyökkääjälle johtaisi koko järjestelmän tietoturvan vaarantumiseen.

Hyökkääjä voi yrittää hakea tietueita AS:ltä arvaamalla voimassa olevan valtuutuskoodin arvon. Tämänlaisten hyökkäyksien estämiseksi AS:llä tulee olla sääntö, joka rajoittaa kutsuja, joissa samasta lähteestä lähetetään useita vääriä valtuutuskoodeja [45]. Tietueiden vuotamisen lisäksi tämänkaltaiset hyökkäykset voivat vaikuttaa negatiivisesti AS:n suorituskykyyn, jos kutsuja tulee suurissa määrissä. Jos AS huomaa valtuutuskoodin uudelleenkäytön, se voi yrittää mitätöidä sovelluksille annetut tietueet, jolloin hyökkääjän saama tietue ei toimisi enää [45]. Joissain tapauksissa AS ei välttämättä pysty estämään jo annettujen tietueiden käyttöä, minkä vuoksi tietueiden tulisi olla väärinkäytön estämiseksi kertakäyttöisiä ja erittäin lyhyen ajan voimassa [11].

5.2 Autentikointitapahtuman käsittely verkkosovelluksessa

Vaikka yksinkertaisimmat menetelmät vaativat vähiten konfiguroitavaa ja ovat turvallisuusominaisuuksiltaan hyvällä tasolla, on silti parempi ottaa kaikki mahdolliset turvallisuusominaisuudet käyttöön. Tämä tarkoittaa muun muassa sitä, ettei tietueiden tietoja saateta selaimen tietoisuuteen, joten käytetään *Authorization code flow* -menetelmää [29, s. 94]. Tämän lisäksi kutsuihin, jotka lähetetään

OP:lle, sisällytetään *nonce*-parametri, jonka avulla voidaan estää toistohyökkäykset [11].

Autentikoitava käyttäjä voidaan tunnistaa saatavasta tunnistetietueesta. Tunnistetietue validoidaan JWT-spesifikaation [26] määrittelemällä tavalla. Tietueesta tulee varmistaa, että se on JWT-formaatissa erottamalla se kolmeen osaan ”.”-merkkien avulla, dekoodaamalla jokainen osa Base64URL-formaatista, jäsentämällä ensimmäinen ja toinen osa JSON-muodossa ja tarkistamalla viimeisen osan eli allekirjoituksen oikeellisuus OP:lta haetulla avaimella [25, luku 13]. Useimmat autentikointikirjastot tekevät validoinnin automaattisesti, jolloin on pienempi riski sille, että kehittäjä toteuttaa virheellisen validoinnin. Julkiset salausavaimet allekirjoituksen tarkistamista varten voidaan hakea identiteettipalvelulta HTTPS-protokollaa käyttäen tai konfiguroida ne etukäteen sovelluksen tietoon.

JWT-formaatin varmistamisen lisäksi tietueen sisältämät tiedot on hyvä hyökkäyksien varalta varmistaa [25, luku 13]. Kentät, jotka tulee tarkistaa, ovat *iss*, joka on tietueen lähettävän palvelun tunniste, *aud*, joka on tietueen kohde, *alg*, joka kuvaa JWT:n turvaamiseksi käytettyä algoritmia, *exp*, joka kertoo tietueen voimassaolon päättymisajan, *iat*, joka kertoo tietueen luomisajan, ja *nonce*. Kentän *iss* on vastattava OpenID Providerin (OP:n) URL-osoitetta. Kentässä *aud* olevan asiakastunnisteen on vastattava autentikointipyynnön lähettävän verkkosovelluksen AS:lle konfiguroitua asiakastunnistetta. Jos *aud*-kenttä on usean tunnisteen sisältävä lista, tietueen pitää sisältää *azp*-kenttä, jonka arvon on vastattava verkkosovelluksen asiakastunnistetta. Kentän *alg* on oltava ”RS256”, joka on kentän oletusarvo, tai verkkosovelluksen rekisteröinnissä määrittelemä arvo. Kentän *exp* arvon on oltava ajankohta lähitulevaisuudessa. Kenttää *iat* käytetään liian vanhojen tietueiden hylkäämiseen. Saadun tietueen *nonce*-kentän on vastattava täysin sovelluksen lähettämää alkuperäistä arvoa. [11]

6. TOTEUTUS

Sovelluksen käyttöliittymään lisättiin näkymä, jonka avulla autentikoitava käyttäjä voi valita autentikoitua kolmannen osapuolen avulla. Kutsu autentikoinnin aloittamisesta ohjataan identiteettipalveluun, joka ohjaa selaimen IdP:lle. Kun konfiguroituja IdP:ta – toisin sanoen asiakkaita – on monia, asiakkaille voidaan toimittaa linkki, joka vie käyttäjän suoraan autentikoitumaan organisaation IdP:lle. Azure AD -käyttäjähakemistojen tapauksessa linkit saadaan luotua AWS Cogniton avulla ilman lähdekoodin muutoksia. Windows AD -käyttäjähakemistojen tapauksessa puolestaan muutokset lähdekoodiin ovat todennäköisiä, sillä niiden osoitteiden ja tunnisteiden konfiguroiminen pitää tehdä identiteettipalvelun sijasta itse järjestelmään.

Vaikka ulkoisesti autentikoivien käyttäjien sähköpostiosoitteet saataisiin järjestelmän tietoon käyttäjähakemistoilta, niiden paikkansapitävyydestä ei voida olla täysin varmoja. Tämän lisäksi on todennäköistä, että sovellukseen rekisteröity sähköpostiosoite ole välttämättä sama, kuin käyttäjähakemistoon rekisteröity sähköposti. Näistä syistä käyttäjän on ensin ”linkitettävä” sovelluksen profiili ulkoisen identiteettinsä kanssa. Linkittäminen tapahtuu ensin kirjautumalla sovelluksen tunnuksilla käyttäjällä sisään, minkä jälkeen käyttäjän asetuksissa on mahdollista autentikoitua kolmannen osapuolen avulla. Vain tällöin voidaan olla varmoja, että käyttäjällä on oikeus tiettyyn sovelluksen profiiliin. Yksi ulkoinen identiteetti voi olla linkitettynä vain yhteen sovelluksen käyttäjään.

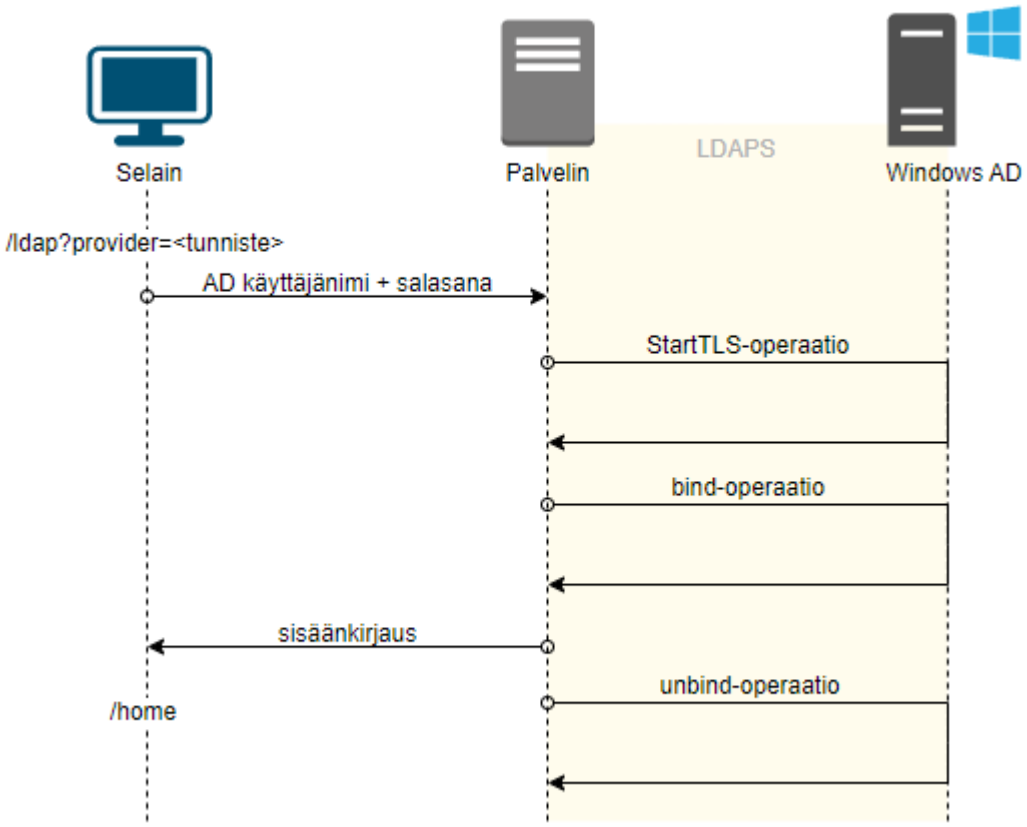
Linkitys sovelluksen sisällä Azure AD:n tapauksessa tapahtuu käyttämällä identiteettipalvelulta tulevan käyttäjän yksilöivää tunnistetta, joka löytyy tunnistetietueesta. LDAP-protokollaa käytettäessä käyttäjä ei tee autentikointia identiteettipalvelun kautta, vaan tieto autentikoinnista tulee suoraan sovellukselle. Tästä seuraa, että identiteettipalvelulta ei saada käyttäjää yksilöivää tunnusta linkitykseen vaan se pitää itse luoda.

Ylläpitäjä voi myös tehdä linkityksen sovelluksen puolella käsin, jolloin identiteettipalvelusta haettu tunniste yhdistetään manuaalisesti sovelluksessa olevaan käyttäjäprofiiliin. Ylläpitäjän pitää tätä varten hakea tunniste AWS Cognitolta käsin käyttäen AWS-pilvipalveluiden käyttöliittymää. Näin voidaan varmistaa sujuva

käyttö käyttäjälle, vaikka linkitys ei syystä tai toisesta onnistuisi. Samoin ylläpitäjä voi myös poistaa linkityksen käyttäjän pyynnöstä tai jos kolmannen osapuolen tunnuksia väärinkäytetään. Identiteettipalvelu kirjoittaa lokin jokaisesta autentikointitapahtumasta. Samaan tapaan sovellus ylläpitää lokia muun muassa ulkoisen identiteetin linkityksestä ja sisäänkirjautumisista. Ylläpidetyistä lokeista ylläpitäjä voi tarkistaa mihin profiiliin ulkoinen identiteetti yritettiin linkittää ja näin tehdä linkityksen manuaalisesti.

6.1 Windows AD

Windows AD:n käyttäjien autentikoimiseen on käytettävä LDAP-protokollaa, koska autentikoitavien käyttäjien hakemistot tukevat ainoastaan sitä. Tässä kohtaa on kuitenkin asetettava vaatimus, että palvelimien on tuettava LDAPSia. Salaamattomia LDAP-protokollan menetelmiä ei voida tukea, koska silloin salasana kulkee selkokielisenä julkisen verkon yli, mikä aiheuttaa huomattavan tietoturvallisuusriskin. Koska LDAP-protokollaa toteuttavaa identiteettipalvelua ei löytynyt, tuki ja autentikointi protokollan avulla on kehitettävä itse. Kuva 3 kuvaa Windows AD -käyttäjän autentikoinnin työnkulun.



Kuva 3: Windows AD -käyttäjän autentikointi

LDAP-palvelimeen yhdistettäessä täytyy ensin suorittaa StartTLS-operaatio, jonka avulla voidaan suojata yhteys. LDAP-protokollan dokumentaatio [21] määrittelee kutsujen ja vastauksien syntaksin ja käsittelyn. Kun yhteys on suojattu, käyttäjän autentikointi suoritetaan *bind*-operaatiolla, johon tarvitaan käyttäjän käyttäjänimi ja salasana käyttäjähakemistoon. Jos palvelin palauttaa virhekoodin missä tahansa vaiheessa, tulee yhteys katkaista *unbind*-operaatiolla. Palvelin palauttaa virhekoodin muun muassa silloin, kun se ei tue StartTLS-operaatiota tai kun *bind*-operaatioon annetut käyttäjätiedot ovat virheelliset. Koska LDAP-palvelin ei luo minkäänlaista istuntoa käyttäjän selaimen ja palvelimen välille, tietoa käyttäjän uloskirjautumisesta ei tarvitse viestittää sovelluksen ulkopuolelle. Lopuksi onnistuneen *bind*-operaation eli autentikoinnin jälkeen yhteys suljetaan suorittamalla *unbind*-operaatio.

LDAP-toteutukseen käytetään .NETin omaa laajennuskirjastoa *System.DirectoryServices.Protocols*, joka tarjoaa muun muassa LDAP V3 -standardin määrittelemät metodit [46]. Kuvassa 4 on esitelty paikallisesti ajossa olevan LDAP-palvelimen yhteyden alustaminen. Muuttujalla *host* määritellään palvelimen osoite ja

käytettävä portti. Jokaisella palvelimella on luonnollisesti eri osoite sekä palvelimen konfiguraatiosta riippuen eri käytettävä portti. Samoin muuttujan *userDN*, joka kuvaa käyttäjän DN-arvoa LDAP-käyttäjähakemistossa, kokoaminen riippuu käyttäjänimen lisäksi käyttäjähakemiston rakenteesta. Näistä syistä LDAP-käyttäjähakemistojen dynaaminen lisääminen on haastavaa. Lisäksi yhteyden versioksi asetetaan versionumero 3, koska StartTLS-operaatio vaatii tämän versionumeron, kuten aliluvussa 3.1 kerrottiin. Autentikoinnin tyypiksi asetetaan *Basic*, mikä tarkoittaa sitä, että autentikointi hakemistolle suoritetaan salasanalla ja käyttäjänimellä. Jotta voidaan varmistua, että vastaus tuli palvelimelta, jonne pyyntö lähetettiin, tarvitsee palvelimen vastauksen sertifikaatti validoida. Sertifikaatin validointi onnistuu .NETin sisäisellä kirjastolla yksinkertaisesti.

On mahdollista myös käyttää suojattua TLS-yhteyttä ilman StartTLS-operaatiota, jos LDAP-käyttäjähakemistolla on erikseen portti TLS-yhteyksille. Tällöin luotavan yhteyden asetuksiin *SessionOptions* asetetaan *SecureSocketLayer*-ominaisuus päälle, sekä suojatun portin numero *host*-muuttujaan.

```
string host = "127.0.0.1:1389";
LdapConnection connection = new LdapConnection(host);

string userDN = $"cn={username},ou=users,dc=example,dc=com";
NetworkCredential credential = new NetworkCredential(userDN, userPassword);

connection.SessionOptions.ProtocolVersion = 3;
connection.AuthType = AuthType.Basic;
connection.SessionOptions.VerifyServerCertificate = new
    VerifyServerCertificateCallback(ServerCertificateValidation);
```

Kuva 4: LDAP-yhteyden alustaminen

Kuvassa 5 on koodiesimerkki käyttäjän autentikoinnista käyttäjähakemistolle. Aluksi lähetetään käyttäjähakemistolle pyyntö StartTLS-operaatiosta, jotta yhteys saadaan suojatuksi. Tämän jälkeen yritetään autentikoida käyttäjä *bind*-operaatiolla, jonka onnistuessa linkitetty käyttäjä voidaan kirjata sisään. Molemmat operaatiot epäonnistuessaan aiheuttavat poikkeuksen, jolloin käyttäjän autentikointi epäonnistuu. Poikkeuksen tiedot kirjataan järjestelmän lokiin. Lopuksi autentikointitapahtuman lopputuloksesta huolimatta yhteys suljetaan.

```
try
{
    connection.SessionOptions.StartTransportLayerSecurity(null);
    connection.Bind(credential);
    // Sisäänkirjaus...

    connection.SessionOptions.StopTransportLayerSecurity();
}
catch (Exception ex)
{
    Console.WriteLine("Error: " + ex);
}
finally
{
    connection.Dispose();
}
```

Kuva 5: Käyttäjän autentikointi

Jotta saadaan yksikäsitteinen tunniste myös LDAP-protokollan avulla autentikoiville, rakennetaan heidän tunnisteensa LDAP-palvelimen osoitteen ja käyttäjänimen yhdistelmästä. Luotu tunniste lisätään sovelluksen profiiliin tietojen yhteyteen. Kun käyttäjä seuraavan kerran autentikoidaan Windows AD -käyttäjähakemiston avulla, autentikoidun käyttäjän tiedoista rakennetaan taas tunniste ja kirjataan löydetty linkitetty käyttäjä sisään sovellukseen.

On mahdollista, että tulevaisuudessa myös SAMLia tai OIDC:tä tukevia tai protokollia tukeviin identiteettipalveluihin integroituja Windows käyttäjähakemistoja tarvitsee autentikoida. Tällöin voidaan kyseessä olevien hakemistojen käyttäjien autentikointi toteuttaa käyttämällä protokollia tukevaa identiteettipalvelua hyväksi.

6.2 Azure AD

Azure AD tukee muun muassa SAML- sekä OIDC-protokollia, joten aliluvussa 4.2 mainittujen ominaisuuksien vuoksi, käytetään OIDC:ta. Käytettävät OP:t konfiguroidaan identiteettipalvelun tietoon etukäteen, ja autentikointi sallitaan vain niiltä palveluilta. Koska IdP:n lisääminen AWS Cognito -identiteettipalveluun vaatii vain hieman konfiguraatiota, on mahdollista lisätä autentikointi muillekin käyttäjähakemistoille, jotka tukevat OIDC-protokollaa.

AWS Cogniton ja asiakkaan käyttäjähakemiston OP:n luottamussuhteen luomiseksi AWS Cognitolle asetetaan jokaisen OP:n kanssa erikseen käytettävät

uudelleenohjauksen osoitteet, asiakastunnisteet ja asiakassalaisuudet. Sovelluksen ja AWS Cogniton välinen luottamus konfiguroidaan samalla tavalla. Sovelluksen ympäristömuuttujiin asetetaan asiakassalaisuus, jota käytetään, kun haetaan AWS Cognitolta tietueet valtuutuskoodilla. AWS Cognitossa konfiguroidaan autentikointitapahtuman enimmäiskestoksi kolme minuuttia ja tietueiden voimassaoloajaksi viisi minuuttia. Lyhyet aikavälit perustellaan sillä, että minimoimalla autentikoinnin ja tietueen voimassaoloajat voidaan pienentää hyökkäyksien, esimerkiksi toistohyökkäyksien, riskiä. Kohdejärjestelmässä on kirjautumissivulla satunnainen viive kirjautuessa sekä yläraja kutsujen määrälle, jolloin luvussa 5 mainituista uhista, joissa hyökkääjä yrittää päätellä tietueen validoinnin kestosta salausavaimen ominaisuuksia tai pommittaa sivua arvaamalla valtuutuskoodeja, ei ole suurta haittaa tai vaaraa.

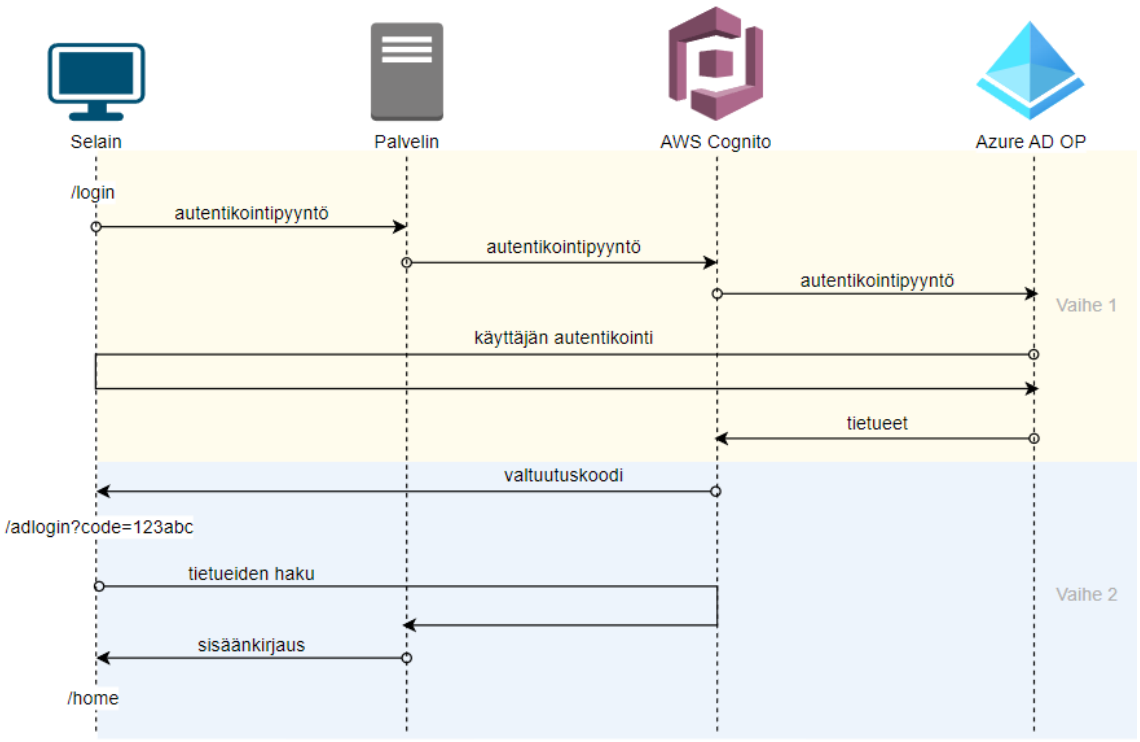
Kun sovellus käynnistyy, identiteettipalvelulta haetaan käytössä olevat salausavaimet ja tallennetaan ne tietokantaan. Salausavaimet haetaan osoitteesta, joka koostuu tiedoista, joilla AWS Cognito -palvelu konfiguroitiin:

> https://coginto-idp.<*AWS_alue*>.amazonaws.com/
> <*AWS_alue*>_<*generoitu_tunniste*>/.well-known/jwks.json.

Avaimet voitaisiin hakea myös *discovery*-päätepisteeltä mutta tässä kohtaa se ei ole tarpeellista, koska kaikki muu informaatio IdP:stä on jo sovelluksen tiedossa. AWS Cognito saattaa päivittää tietueiden allekirjoitukseen käytettäviä salausavaimia [47]. Tämän vuoksi avaimet on hyvä säilöä ja verrata nykyisien avaimien tunnistetta tietueen *kid*-kenttään asetettuun avaimen tunnisteeseen. Jos tunnistetta ei löydy nykyisistä avaimista, täytyy avaimet hakea AWS Cognitolta uudelleen. Tämä toiminnallisuus on myös OIDC:n virallisen dokumentaation suosittelema [11]. Virallisen OIDC:n dokumentaation noudattamisen lisäksi avainten säilöminen vähentää verkon yli identiteettipalvelulle tehtävien kutsujen määrää. Tämä pieni optimointi johtaa nopeampaan autentikointitapahtumaan, sekä alennettuihin AWS:n kuluihin.

Käyttäjille toimitettava linkki, jolla he pääsevät autentikoitumaan suoraan oman IdP:nsä kautta, on järjestelmän sivulle vievä linkki, joka uudelleenohjaa AWS Cogniton käyttöliittymään, jossa on parametrina käyttäjän IdP:n tunnus AWS Cognitossa. Kun autentikoinnin aloittaminen tapahtuu tämän linkin avulla,

voidaan kutsuun asettaa ennen IdP:lle uudelleenohjausta *state*-parametri, joka on satunnaisesti generoitu kirjaimista ja numeroista koostuva merkkijono. Merkkijono salataan ja asetetaan selaimen evästeisiin talteen. Kun autentikointitapahtuman jälkeen selain ohjataan takaisin järjestelmän sivuille, AWS Cognito palauttaa kutsun mukana tulleen *state*-parametrin. Jos lähetetyn ja saadun parametrin arvoissa on eroa, ei autentikointitapahtumaa voida hyväksyä. Näin *state*-parametria käyttäen pienennetään luvussa 5 mainitun CSRF-hyökkäyksen riskiä.



Kuva 6: Azure AD -käyttäjän autentikointi

Kuten kuvasta 6 voidaan huomata, autentikoinnin voi jakaa identiteettipalvelun näkökulmasta kahteen vaiheeseen. Ensimmäisessä vaiheessa AWS Cognito toimii asiakasohjelmistona ja toisessa vaiheessa OP:na [14]. Ensimmäisessä vaiheessa käyttäjä siis autentikoidaan oman Azure AD:n, joka toimii OP:n roolissa, avulla asiakasohjelmistona toimivalle identiteettipalvelulle. Identiteettipalvelu ylläpitää omaa sisäistä istuntoa käyttäjälle Azure AD:n kanssa. Toisessa vaiheessa, kun käyttäjä on autentikoituna identiteettipalvelulle, selain ohjataan takaisin alkuperäiselle asiakasohjelmistona toimivalle verkkosovellukselle, ennalta määriteltyyn URL-osoitteeseen mukanaan kyselyparametrina asetettu

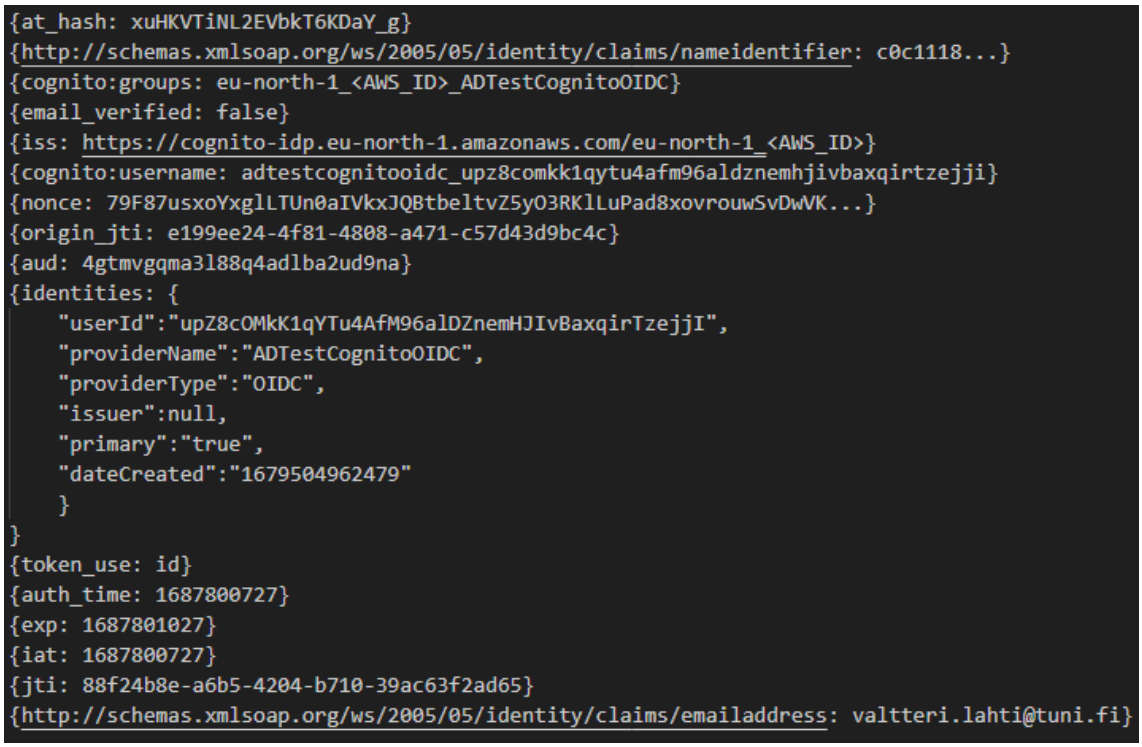
valtuutuskoodi. Uudelleenohjauksen jälkeen verkkosovellus hakee nyt OP:na toimivalta identiteettipalvelulta valtuutuskoodin avulla tietueet.

Kuvassa 7 on kuvattuna identiteettipalvelun tietuepäätepisteeseen lähetettävä kutsu. Kutsuun asetetaan menetelmän tyyppi, valtuutuskoodi, asiakastunniste ja -salaisuus sekä uudelleenohjauksen osoite, johon identiteettipalvelu aiemmin ohjasi selaimen valtuutuskoodin kanssa. Kuten kuvasta voi päätellä, OIDC-protokollalla autentikointia on kehitetty ja testattu paikallisesti ajossa olevaa verkkosovellusta käyttäen, jolloin tietueiden ja kutsujen tarkastelu on helpompaa kuin pilvipalvelussa ajossa olevan verkkosovelluksen kanssa. Kuten kuvasta voidaan nähdä, uudelleenohjauksen osoite on tarkka osoite, jolloin luvussa 5 mainittu avoimeen uudelleenohjauksen osoitteeseen kohdistuva hyökkäys ei ole mahdollinen.

```
{
    { "grant_type", "authorization_code" },
    { "code", code },
    { "client_id", "4gtmvgqma3l88q4adlba2ud9na" },
    { "client_secret", "zxcvbnm0987654321" },
    { "redirect_uri", "https://localhost:5025/adlogin" }
};
```

Kuva 7: Tietuepäätepisteen kutsun rakenne

Kuvassa 8 on kuvattuna AWS Cogniton palauttaman tunnistetietueen sisältö, jossa yksilöivä tunniste profiilin linkitystä varten on *cognito:username*-kentässä. Kuten kuvan *identities*-kentän arvoista voi huomata, tunniste on käyttäjähakemistolle AWS Cognitossa annetun nimen ja käyttäjälle generoidun tunnisteen yhdistelmä. Edellä mainittu *identities*-kenttä on AWS Cogniton luoma väite, jonka *userId*-kenttä on käyttäjän IdP:ltä saatu tunniste [14]. Tietueesta voidaan myös löytää kaikki aliluvussa 5.1 tärkeinä esitellyt kentät, jotka ovat *iss*, *aud*, *exp*, *iat* ja *nonce.* Puuttuva aliluvussa 5.1 mainittu kenttä *alg* kulkee tietueen otsikkokentässä. Tietueesta voidaan huomata tässä kohtaa, jos esimerkiksi *aud*-kentän arvo ei vastaa kuvan 7 *client_id*-kentän eli asiakastunnisteen arvoa.



Kuva 8: Tunnistetietueen sisältö

Tunniste, joka on konfiguroidun käyttäjähakemiston tunnisteen ja identiteettipalvelun käyttäjälle antaman tunnisteen yhdistelmä, lisätään sovelluksen profiiliin tietojen yhteyteen. Kun käyttäjä seuraavan kerran autentikoidaan ulkoisen palvelun avulla, tunnistetietueesta haetaan käyttäjän tunniste ja kirjataan linkitetty käyttäjä sisään sovellukseen.

Tietueen saatuaan sovelluksen tulee validoida tietueiden allekirjoitus ja sisältö aliluvussa 5.1 esitellyllä tavalla. Käytetty käsittelijä JWT-muotoisten tietueiden lukemiseen ja validoimiseen on .NETin *System.IdentityModel.Tokens.Jwt*-kirjaston *JwtSecurityTokenHandler*. *JwtSecurityTokenHandler*illa on tietueen validoimiseen metodi *ValidateToken*. Metodi ottaa parametreikseen tietueen merkkijonona ja *TokenValidationParameters*-objektin, jolla asetetaan kuvan 9 kuvaamat validointiparametrit. Useimmat parametrit ovat itsestäänselvyyksiä, ja validointi tapahtuu lähes täysin automaattisesti kirjaston validointimetodilla. Metodille tarvitsee vain asettaa muutamat asetukset sekä odotetut kenttien arvot, kuten *iss*-kentän odotettu arvo, *aud*-kentän odotettu arvo ja allekirjoitukseen käytetyt salausavaimet. Ylimääräisenä suojana toistohyökkäyksiä vastaan on otettu käyttöön *TokenReplayValidator*. *TokenReplayValidator* säilöö voimassa olevat tietueet ja varmistaa uuden tietueen validoinnissa, ettei tietuetta ole jo yritetty käyttää

sisäänkirjautumiseen. Tietueen validoinnin epäonnistuessa *ValidateToken*-metodi palauttaa poikkeuksen, joka sisältää epäonnistumisen syyn. Poikkeuksen tiedot kirjataan järjestelmän lokiin. Jos tietueen validointi epäonnistuu, käyttäjälle näytetään vain oletusarvoinen virheilmoitus. Tämä tehdään varotoimenpiteenä siitä syystä, etteivät mahdolliset hyökkääjät saa tietoonsa sitä, mikä tarkalleen tietueessa oli väärin.

```
ValidateIssuer = true,
ValidateAudience = true,
ValidateLifetime = true,
ValidateIssuerSigningKey = true,
ValidIssuer = issuer,
ValidAudience = "4gtmvgqma3l88q4adlba2ud9na",
TokenReplayValidator = TokenReplayValidator,
IssuerSigningKeys = signingKeys,
RequireSignedTokens = true,
RequireExpirationTime = true,
LogValidationExceptions = true,
ValidateSignatureLast = true,
```

Kuva 9: Tietueen validointiparametrit

Kun tietueiden sisältöjen oikeellisuudesta voidaan olla varmoja, voidaan linkitetty käyttäjä kirjata sisään. Tietueiden sisältö ei itsessään ole arkaluontoista, joten niiden lisäsalausta JWE:n avulla ei nähdä tarpeelliseksi. OIDC-spesifikaatio määrittelee, että OP:n tulee aina varmistaa tietueiden haun yhteydessä, ettei sovelluksen antamaa valtuutuskoodia ole käytetty aikaisemmin [11]. AWS Cognito noudattaa tätä määritelmää estäen valtuutuskoodin uudelleenkäytön, mikä laskee toistohyökkäyksien riskiä [48].

AWS Cognito tallettaa autentikoidun istunnon tiedot käyttäjän selaimen evästeisiin. [14] Jos käyttäjä yrittää autentikoitua uudelleen samaa IdP:tä käyttäen, se onnistuu voimassa oleville istunnoille ilman, että käyttäjän tarvitsee syöttää uudelleen tunnuksiaan. Tästä syystä tehdään sovelluksen uloskirjautumisen yhteydessä varotoimenpiteenä uloskirjautumiskutsu AWS Cognitolle. Uloskirjautuminen tehdään Cogniton */logout*-päätepisteelle, jolloin Cognito poistaa tiedot voimassa olevasta istunnosta [49].

6.3 Natiivit käyttäjät

Natiivien käyttäjien, eli sovellukseen luotujen käyttäjien, sisäänkirjautumista ei tässä vaiheessa muuteta. Tämä johtuu siitä, että kaikki käyttäjät eivät tule käyttämään ulkoista autentikointia. Tavallista sisäänkirjautumista saattavat myös tarvita ulkoisesti autentikoituvat käyttäjät esimerkiksi silloin, kun autentikointi ei onnistu organisaation käyttäjähakemiston avulla.

7. TULOKSET JA JATKOKEHITYS

Toteutuksen lopputulos vastasi tavoitetta mahdollistaa autentikointi erilaisista käyttäjähakemistoista. Toteutus, jolla lopputulokseen päästiin, ei kuitenkaan vastannut ideaalitilannetta, jossa sovellukseen ei tarvitse tehdä tukea useille eri protokollille. Käytetyt protokollat määräytyivät osittain asiakkaiden käyttäjähakemistojen mahdollisuuksien perusteella. Pilvipalvelupohjaiset käyttäjähakemistot tukevat yleensä edes joitain väitepohjaisia protokollia, joita pystyy käyttämään identiteettipalvelun kanssa, mutta tietyt Windows AD -käyttäjähakemistot tukevat korkeintaan LDAP-protokollaa ulkoiseen autentikointiin. Identiteettipalveluiden tuki LDAP-protokollalle on mahdollinen vain, jos hakemiston palvelimelle voidaan asentaa lisäohjelma, joka toimii hakemiston ja identiteettipalvelun välisenä siltana.

Käytetyt protokollat toteutettiin käyttäen kaikkia saatavilla olevia olennaisia turvallisuusominaisuuksia. Käyttäjien ulkoinen identiteetti yhdistettiin sovelluksen käyttäjään yksilöivällä tunnisteella, joka oli IdP:n ja ulkoisen identiteetin tunnisteiden yhdistelmä. Tunnisteen käyttö osoittautui toimivaksi, koska identiteettipalvelu luo käyttäjille automaattisesti yksilölliset tunnisteet. Windows AD -käyttäjähakemiston käyttäjille, jotka eivät siis autentikoidu identiteettipalvelun kanssa, yksilöllisinä tunnisteina käytettiin hakemiston URL-osoitteen ja käyttäjän DN:n yhdistelmää.

Tulevaisuudessa on tarpeellista, ettei käyttäjän tarvitse itse valita organisaatiota, jonka IdP:n avulla haluaa autentikoitua. Nykyinen toteutus sallii käyttäjän nähdä kaikki organisaatiot, jos hän ei käytä autentikoimiseen suoraa linkkiä, vaan navigoi käyttöliittymän kautta, jolloin käyttäjä saattaa valita väärän IdP:n. On siis tarpeen, että käyttäjän organisaatio voidaan päätellä käyttäjän tiedoista. Päättelyssä voidaan käyttää hyväksi OIDC:n käyttämää WebFinger-protokollaa, joka kykenee hakemaan käyttäjästä tietoja esimerkiksi sähköpostiosoitteen avulla [11]. Tämä luonnollisesti vaatisi käyttäjältä sähköpostin antamisen sovellukselle ennen autentikointiin pääsyä. Jos WebFinger-protokollan käyttö epäonnistuu, kuten Windows AD -hakemiston tapauksessa on todennäköistä, niin käytettävää

IdP:tä voisi yrittää päätellä sovelluksen käyttäjän yhdistetyn ulkoisen identiteetin tunnisteesta.

Toinen jatkokehitysidea on ulkoisen identiteetin yhdistäminen useampaan käyttäjään. Jos sattuu tilanne, että yhdellä henkilöllä on useampia käyttäjätunnuksia sovelluksessa, käyttäjäkokemuksen näkökulmasta olisi tarpeellista mahdollistaa autentikoiminen kaikille käyttäjille samanaikaisesti. Tällöin käyttöliittymästä voisi tarpeen tullen vaihtaa aktiivista käyttäjää ilman, että uutta autentikointia tai kirjautumista vaadittaisiin.

OIDC:n *discovery*-toiminnallisuuden myötä käyttäjät voisivat lisätä uusia OP:ta dynaamisesti. Tämä mahdollistaisi entistä sujuvamman autentikointitapahtuman uusille asiakkaille ja toisi näin lisäarvoa järjestelmälle.

Koska järjestelmä sisältää arkaluontoista ja henkilökohtaista tietoa, ei tässä työssä toteutettua autentikointia voida ottaa suoraan käyttöön tuotantoon. Jotta käyttöönotto voitaisiin tehdä, tulisi toteutukselle ensin tehdä kattavat penetraatiotestit ja tietoturvallisuusauditoinnit alan ammattilaisten suorittamana. Penetraatiotestit ja auditoinnit sekä niissä löytyvien ongelmien korjaaminen olisi iteratiivinen prosessi, jota toistettaisiin, kunnes ongelmia tai haavoittuvuuksia ei enää toteutuksesta löydetä. Kun toteutus on todettu turvalliseksi käyttää, vasta silloin se voidaan viedä tuotantoon asiakkaiden käytettäväksi.

8. YHTEENVETO

Käyttäjän autentikoiminen verkkosovellukseen käyttäjähakemiston avulla parantaa tietoturvallisuutta ja käyttäjäkokemusta sekä helpottaa käyttäjienhallintaa. Käyttäjien autentikoiminen voidaan toteuttaa monella eri protokollalla monesta eri käyttäjähakemistopalvelusta. Lisäämällä identiteettipalvelu sovelluksen ja käyttäjähakemiston väliin, sovellukseen tarvitsee tehdä autentikointi vain yhdellä protokollalla ja menetelmällä. Tällöin sovelluksen puolelle ei tarvitse muuttaa mitään, kun mahdollistetaan autentikointi uudelle käyttäjähakemistolle, vaan selvitään identiteettipalvelun konfiguroimisella. Koska sovellukseen tarvitsee tehdä vähemmän koodia autentikointia varten, tehtyjen virheiden riski pienenee ja virheiden huomaaminen yksinkertaistuu.

Tämän työn tavoite oli tutkia, suunnitella ja toteuttaa autentikointi verkkosovellukseen erilaisista käyttäjähakemistoista. Tutkitut protokollat ja menetelmät arvioitiin ja niiden käyttö suunniteltiin ja toteutettiin onnistuneesti. Priorisoiduksi autentikointiprotokollaksi valittiin OpenID Connect (OIDC) ja menetelmäksi *Authorization code flow*. Koska huomioon oli otettava käyttäjien käyttäjähakemistojen tukemat protokollat, myös LDAP-protokollaa oli käytettävä käyttäjän autentikoimiseksi.

Azure AD:n ja muiden samantapaisten käyttäjähakemistojen autentikointi mahdollistettiin identiteettipalvelun avulla, joka ensin suorittaa käyttäjän autentikoinnin käyttäjän IdP:n avulla, ja toimii sitten itse IdP:nä sovellukselle. Käytetty autentikointimenetelmä identiteettipalvelulta sovellukselle on OIDC-protokollan *Authorization code flow*-menetelmä, jonka avulla tieto käyttäjän autentikoinnista saatiin identiteettipalvelulta turvallisesti palvelimelta palvelimelle.

Integroitava Windows AD -käyttäjähakemisto mahdollisti tuen vain LDAP-protokollalle, joten sitä oli käytettävä. Identiteettipalveluista ei löytynyt LDAPille tukea, joten sen toiminnallisuus oli kehitettävä alusta asti itse. Tietoturvallisuussyistä LDAP-protokollan käyttöön asetettiin sellainen vaatimus, että hakemistojen tulee tukea StartTLS-toiminnallisuutta, jottei käyttäjien salasanat kulje verkon yli selkokielisinä.

Ideaaliin toteutukseen ei kuitenkaan siis päästy, sillä joidenkin käyttäjähakemistojen osalla LDAP-protokolla oli ainoa tuettu protokolla, eikä sen toteutusta saatu konfiguroitua identiteettipalvelulle OIDC-protokollan tavoin. Tehty toteutus kuitenkin mahdollistaa Azure AD:n ja Windows AD:n käyttäjähakemistojen käyttäjien autentikoinnin verkkosovellukselle.

LÄHDELUETTELO

[1] D. Francis, ”Mastering Active Directory,” 2021. Saatavissa: https://learning.oreilly.com/library/view/mastering-active-directory/9781801070393/.

[2] Microsoft, ”Active Directory Domain Services Overview,” 2022. Saatavissa: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

[3] Microsoft, ”Active Directory Structure and Storage Technologies,” 2014. Saatavissa: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759186(v=ws.10).

[4] V. Bertocci, ”Modern Authentication with Azure Active Directory for Web Applications,” 2015. Saatavissa: https://learning.oreilly.com/library/view/modern-authentication-with/9780735698475.

[5] Microsoft, ”AD FS Overview,” 2023. Saatavissa: https://learn.microsoft.com/en-us/windows-server/identity/ad-fs/ad-fs-overview.

[6] Microsoft, ”What is Azure Active Directory?,” 2023. Saatavissa: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis.

[7] Microsoft, ”How Azure AD Delivers Cloud Governed Management for On-Premises Workloads,” 2023. Saatavissa: https://learn.microsoft.com/en-us/azure/active-directory/hybrid/cloud-governed-management-for-on-premises.

[8] S. Zaal, ”Azure Active Directory for Secure Application Development,” 2022. Saatavissa: https://learning.oreilly.com/library/view/azure-active-directory/9781838646509.

[9] Y. Wilson ja A. Hingnikar, ”Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0,” 2019. Saatavissa: https://learning.oreilly.com/library/view/solving-identity-management/9781484250952.

[10] D. Hardt, ”The OAuth 2.0 Authorization Framework,” 2012. Saatavissa: https://www.rfc-editor.org/rfc/rfc6749.

[11] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros ja C. Mortimore, ”OpenID Connect Core 1.0 incorporating errata set 1,” The OpenID Foundation, 2014. Saatavissa: https://openid.net/specs/openid-connect-core-1_0.html

[12] Microsoft, ”What is Azure Active Directory B2C?,” 2023. Saatavissa: https://learn.microsoft.com/en-us/azure/active-directory-b2c/overview.

[13] A. W. Services, ”Amazon Cognito user pools,” 2022. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html.

[14] Amazon Web Services, ”Adding user pool sign-in through a third party,” 2022. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-identity-federation.html.

[15] Okta, ”SDKs and tools,” 2023. Saatavissa: https://developer.okta.com/code/.

[16] Okta, ”Get started with LDAP integration,” 2023. Saatavissa: https://help.okta.com/en-us/Content/Topics/Directory/ldap-agent-get-started.htm.

[17] Auth0, ”Auth0 Overview,” 2023. Saatavissa: https://auth0.com/docs/get-started/auth0-overview.

[18] Auth0, ”AD/LDAP Connector,” 2023. Saatavissa: https://auth0.com/docs/authenticate/identity-providers/enterprise-identity-providers/active-directory-ldap/ad-ldap-connector.

[19] M. Mayank ja M. Garg, ”Developing Applications with Azure Active Directory: Principles of Authentication and Authorization for Architects and Developers,” 2019. Saatavissa: https://learning.oreilly.com/library/view/developing-applications-with/9781484250402.

[20] C. Adams ja S. Lloyd, ”Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition,” 2002. Saatavissa: https://learning.oreilly.com/library/view/understanding-pki-concepts/0672323915/.

[21] J. Sermersheim, ”Lightweight Directory Access Protocol (LDAP): The Protocol,” 2006. Saatavissa: https://www.rfc-editor.org/rfc/rfc4511.

[22] M. Butcher, ”Mastering OpenLDAP,” 2007. Saatavissa: https://learning.oreilly.com/library/view/mastering-openldap/9781847191021.

[23] S. Tuttle, A. Ehlenberger, R. Gorthi, J. Leiserson, R. Macbeth, N. Owen, S. Ranahandola, M. Storrs ja C. Yang, ”Understanding LDAP - Design and Implementation,” 2004. Saatavissa: https://learning.oreilly.com/library/view/understanding-ldap/073849786X.

[24] S. Cantor, J. Kemp, R. Philpott ja E. Maler, ”Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,” 2005. Saatavissa: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[25] J. Richer ja A. Sanso, ”OAuth 2 in Action,” 2017. Saatavissa: https://learning.oreilly.com/library/view/oauth-2-in/9781617293276.

[26] M. Jones, J. Bradley ja N. Sakimura, ”JSON Web Token (JWT),” 2015. Saatavissa: https://www.rfc-editor.org/rfc/rfc7519.

[27] M. Jones, J. Bradley ja N. Sakimura, ”JSON Web Signature (JWS),” 2015. Saatavissa: https://www.rfc-editor.org/rfc/rfc7515.

[28] M. Jones ja J. Hildebrand, ”JSON Web Encryption (JWE),” 2015. Saatavissa: https://www.rfc-editor.org/rfc/rfc7516.

[29] V. Bertocci, ”OAuth2 and OpenID Connect: The Professional Guide,” 2023. Saatavissa: https://auth0.com/resources/ebooks/oauth-openid-connect-professional-guide.

[30] N. Sakimura, J. Michael, M. Jones ja E. Jay, ”OpenID Connect Discovery 1.0 incorporating errata set 1,” 2014. Saatavissa: https://openid.net/specs/openid-connect-discovery-1_0.html.

[31] Microsoft, ”Web sign in with OpenID Connect in Azure Active Directory B2C,” 2023. Saatavissa: https://learn.microsoft.com/en-us/azure/active-directory-b2c/openid-connect.

[32] A. W. Services, ”Using the ID token,” 2023. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-id-token.html.

[33] Auth0, ”Mitigate Replay Attacks When Using the Implicit Flow,” 2023. Saatavissa: https://auth0.com/docs/get-started/authentication-and-authorization-flow/mitigate-replay-attacks-when-using-the-implicit-flow.

[34] Amazon Web Services, ”Using your own domain for the hosted UI,” 2023. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-add-custom-domain.html.

[35] Microsoft, ”Azure Active Directory External Identities pricing,” 2023. Saatavissa: https://azure.microsoft.com/en-gb/pricing/details/active-directory/external-identities/.

[36] Microsoft, ”Azure Active Directory (Azure AD) pricing,” 2023. Saatavissa: https://azure.microsoft.com/en-us/pricing/details/active-directory/.

[37] Amazon Web Services, ”Amazon Cognito Pricing,” 2023. Saatavissa: https://aws.amazon.com/cognito/pricing/.

[38] Okta, ”Okta Pricing,” 2023. Saatavissa: https://www.okta.com/pricing/#customer-identity-pricing.

[39] Auth0, ”Auth0 Pricing,” 2023. Saatavissa: https://auth0.com/pricing.

[40] Amazon Web Services, ”AWS Services in Scope by Compliance Program,” 2023. Saatavissa: https://aws.amazon.com/compliance/services-in-scope/.

[41] C. Yuen ja S. Vidich, ”Microsoft Azure Compliance Offerings,” 2020. Saatavissa: https://azure.microsoft.com/en-gb/resources/microsoft-azure-compliance-offerings/

[42] Okta, ”Okta Service Certifications,” 2023. Saatavissa: https://trust.okta.com/compliance

[43] Auth0, ”Security, Privacy & Compliance,” 2023. Saatavissa: https://auth0.com/security

[44] Auth0, ”Prevent Common Cybersecurity Threats,” 2023. Saatavissa: https://auth0.com/docs/secure/security-guidance/prevent-threats.

[45] T. Lodderstedt, M. McGloin ja P. Hunt, ”OAuth 2.0 Threat Model and Security Considerations,” RFC Editor, 2013. Saatavissa: https://www.rfc-editor.org/rfc/rfc6819.

[46] Microsoft, ”System.DirectoryServices.Protocols Namespace,” Saatavissa: https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.protocols?view=dotnet-plat-ext-7.0

[47] Amazon Web Services, ”Verifying a JSON web token,” 2023. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-tokens-verifying-a-jwt.html

[48] Amazon Web Services, ”Amazon Cognito - Token endpoint,” Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/token-endpoint.html

[49] Amazon Web Services, ”Amazon Cognito - Logout endpoint,” 2023. Saatavissa: https://docs.aws.amazon.com/cognito/latest/developerguide/logout-endpoint.html

[50] Okta, ”What is Okta,” 2022. Saatavissa: https://developer.okta.com/docs/concepts/how-okta-works