KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / Ohjelmistotekniikka

Niko Säyriö

IDENTITEETIN JA KÄYTTÖOIKEUKSIEN HALLINTA MONITOIMITTAJAYMPÄRISTÖSSÄ

Opinnäytetyö 2016

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka / Ohjelmistotekniikka

SÄYRIÖ, NIKO Identiteetin ja käyttöoikeuksien hallinta monitoimittajaympäristössä
Opinnäytetyö 36 sivua, 1 liitesivu
Työn ohjaaja Yliopettaja Paula Posio
Toimeksiantaja Propentus Oy
Maaliskuu 2016
Avainsanat SAML, IDP, SSO, SLO

Opinnäytetyön tavoitteena oli tutustua identiteetin ja käyttöoikeuksien hallinnan teoriaan, sekä SAML-protokollaan niin laajasti, että opittujen tietojen avulla oli mahdollista toteuttaa Kuntalaistili-sovellukseen SAML-standardin mukainen Identity Provider -komponentti.

Kuntalaistilin uusin versio on ollut kehityksessä alkukeväästä 2016 asti, ja sen on suunniteltu menevän tuotantoon vuoden 2017 alkupuolella. Uusin versio pitää sisällään Identity Provider -komponentin lisäksi muun muassa käyttäjärekisterin ja integraation verkkomaksamis- ja tunnistuspalvelu Vetumaan.

Testi-integraatio Kuntalaistilin Identity Provider -komponenttiin on tehty kolmannen osapuolen toimesta ja se on vastannut palveluntarjoajien vaatimuksia.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology / Software Engineering

SÄYRIÖ, NIKO	Identity and Access Management in Multi-Provider Platform
Bachelor's Thesis	37 pages, 1 appendix page
Supervisor	Paula Posio, Principal Lecturer
Commissioned by	Propentus Ltd
March 2016	
Keywords	SAML, IDP, SSO, SLO

The objective of this thesis was to study about the theory of the identity and access management, and SAML-protocol so extensively that SAML-standard compliant Identity Provider component could be implemented into Kuntalaistili application.

The new version of Kuntalaistili has been in development since spring 2016 and it is planned to go to production in early 2017. Besides the Identity Provider component, the new version also includes user registry services and integration to an electronical payment and identification service called Vetuma.

The test integration to Kuntalaistili's Identity Provider component has been made by a third-party service provider and the implementation has met the requirements given to it.

SISÄLLYS

1 JOHDANTO

Viime vuosina on alettu puhua yhä enemmän turvallisuudesta ja sen merkityksistä eri aloilla. Asenteet, standardit ja odotukset turvallisuuden suhteen ovat muuttuneet merkittävästi sitten 80-luvun. Tästä on hyvä esimerkki käsite työturvallisuus. Nykyään monilla ruumiillisen työn aloilla kaikilla työntekijöillä pitää olla yhtenevä standardin mukainen varustus, toimintamallit ja työturvallisuuskurssi käytynä, muuten ei ole töihin mitään asiaa.

Turvallisuuden tärkeys on myös heijastunut vahvasti tietotekniikan alan kehittymiseen erityisesti 2000-luvulla; itseasiassa trendi on ollut niin kova, että on syntynyt uusi tietotekniikan osa-alue: tietoturvallisuus. Tietoturvallisuuden tehtävänä on suojata koneita, laitteita, verkkoja ja erityisesti niiden sisällään pitämää tietoa. Olemme pikkuhiljaa siirtymässä täydelliseen tietoyhteiskuntaan, jossa kaikki tieto ei olekaan enää paperilla ja melkein kaikki tärkeät palvelut ovat nykyään verkossa, mikä tarkoittaa sitä, että tietoturvan on oltava kunnossa ja siihen panostetaan.

Erityisesti yksityiset yritykset sijoittavat yhä enemmän rahaa turvallisuuteen, ja erityisesti tietoturvaan. Tämä trendi on myös synnyttänyt uusia yrityksiä, tuotteita ja teknologioita, jotka nimenomaan tähtäävät ratkaisemaan kaikki tietoturvaan liittyvät ongelmat.

Nykyään yritykset pystyvät harvemmin toimimaan niin sanotusti ”kuplassa” eli täysin itsenäisesti. Usein yrityksillä on sidosryhmiä, joiden kanssa yritys tekee yhteistyötä aktiivisesti. Esimerkiksi yritys valmistaa tuotetta ja sen sidosryhmään kuuluu toinen yritys, joka hoitaa näiden tuotteiden jakelun. Voi olla, että yritys, joka valmistaa tuotetta sallii kuljetusyritykselle pääsyn heidän tuotantojärjestelmään tietyillä rajoitetuilla oikeuksilla, josta kuljetusyritys näkee kuljetusta odottavien tuotteiden määrän ja näin osaa reagoida asianmukaisella tavalla lähettämällä juuri oikean määrän kuorma-autoja hakemaan tavaroita.

1.1 Toimeksiantaja

Tämän opinnäytetyön aiheen toimeksiantaja on Propentus Oy. Propentus Oy on noin kymmenen vuotta sitten perustettu suomalainen ohjelmistoalan yritys, jonka asiakkaita ovat niin kansainväliset pörssiyhtiöt, kuin julkisen sektorin toimijat. Tällä hetkellä Propentus työllistää noin 50 henkilöä. Yrityksen pääkonttori sijaitsee Kouvolassa, mutta yrityksellä on tällä hetkellä toimipisteet myös Lahdessa, sekä Kotkassa.

Olen työskennellyt Propentuksella noin 4 vuotta ohjelmistokehittäjänä kehittämässä Kuntalaistili-sovellusta. Tähän asti olen kerennyt olemaan monessa eri projektissa mukana, muun muassa kehittämässä Lahden kaupungille tehtyä kiinteän omaisuuden palvelua (Kompa), joka mahdollistaa kuntalaisille mahdollisuuden tarkastella omistukseen liittyviä kiinteistö- ja rakennustietoja sekä niihin liittyviä dokumentteja sähköisesti. Kompa-palvelu palkittiin vuoden 2013 Palvelujen partaveitsi –kilpailussa kolmen parhaan joukkoon.

Keväällä 2015 Kuntalaistiliin aloitettiin tekemään arkkitehtuuriuudistusta, joka käytännössä tarkoitti Kuntalaistilin osalta täydellistä uusiutumista teknologioiden ja samalla käyttöliittymien osalta, kuitenkin edelleen tukien vanhoja kehitettyjä palveluja. Olen ollut suunnittelemassa ja kehittämässä uusittua Kuntalaistiliä alusta lähtien, mikä on ollut hieno mahdollisuus.

1.2 Työn tavoite

Tämän opinnäytetyön tavoite on tutustua identiteetin ja käyttöoikeuksien hallinnan teoriaan, sekä SAML-standardiin niin laajasti, että tiedoilla on mahdollista toteuttaa standardin mukainen Identity Provider -komponentti Kuntalaistili-sovellukseen.

2 IDENTITEETIN HALLINTA

Identiteetin ja käyttöoikeuksien hallinta eli Identity and Access Management (IAM) on yläkäsite sovelluksille, jotka toteuttavat vaaditut identiteetin hallintaan liittyvät vaatimukset, kuten käyttäjien tunnistamisen ja käyttöoikeuksien hallinnan.

IAM-järjestelmä toimii yleensä jonkin toisen järjestelmän rinnalla ja sen tehtävä on hallita pääjärjestelmän käyttäjien identiteettejä. Identiteetin hallinta ei rajoitu pelkäs-

tään ihmisiin, vaan IAM-järjestelmän näkökulmasta myös esimerkiksi laitteet ja toiset palvelut ovat tasa-arvoisia käyttäjiä, entiteettejä.

Järjestelmästä riippuen, pelkkä tieto käyttäjän identiteetistä ei ole aina riittävä, vaan tarvitaan esimerkiksi käyttäjäkohtaista pääsynhallintaa. IAM-järjestelmään voidaan luoda sääntöjä esimerkiksi käyttäjäryhmien perusteella ja antaa näille ryhmille tiettyjä oikeuksia, tai rajoittaa tiettyjä toimintoja.

Johdannossa puhuttiin tietoturvasta ja sen merkityksestä. Tietoturvan ydinajatus on turvata ja suojata tärkeää tietoa joutumasta ulkopuolisen käsiin. Jotta ulkopuolinen uhka voidaan torjua, tarvitaan identiteetin hallintaa tunnistamaan järjestelmässä toimivat entiteetit. Voidaan todeta, että tietoturvan yksi perusvaatimuksista on, että identiteetin hallinta on luotettavaa ja se on toteutettu oikein.

3 PÄÄSYNHALLINTA

Pääsynhallinnalla tarkoitetaan prosessia, jolla kontrolloidaan entiteettien pääsyä eri järjestelmän resursseihin. Resurssi voi olla esimerkiksi dokumentti tietokoneella tai ohjelmiston tietty ominaisuus.

Pääsynhallinta eroaa identiteetin hallinnasta siten, että pääsynhallinta tapahtuu vasta kun toimijan identiteetti on tunnistettu.

Pääsynhallinnan logiikka rakentuu säännöistä, jotka noudattava jotain tiettyä mallia. Alla on esitelty neljä yleisemmin käytettyä pääsynhallinnan sääntömallia.

3.1 Mandatory Access Control

Mandatory Access Control (MAC) on sääntömalleista kaikista tiukin ja sen hallinta tapahtuu ohjelmistojen näkökulmasta kaikista matalimmalla tasolla eli käyttöjärjestelmätasolla. MAC:n malli perustuu turvallisuusetiketteihin (Security label), jotka lisätään jokaiseen järjestelmässä olevaan resurssiin. Jokainen etiketti koostuu kahdesta eri tiedosta: resurssin turvallisuusluokituksesta (Classification), esimerkiksi salainen,

luottamuksellinen tai ei luottamuksellinen ja kategoriasta johon resurssi liittyy (Category), esimerkiksi tietty valtionhallinnon osasto tai tietty projekti.

Kuten resurssiin, myös jokaiseen järjestelmässä olevaan käyttäjätunnukseen pitää lisätä turvallisuusetiketti. Kun käyttäjä yrittää päästä käsiksi resurssiin, käyttöjärjestelmä tarkistaa, että käyttäjän turvallisuusetiketin tiedot vastaavat resurssin vaatimuksia. Esimerkiksi vaikka käyttäjän turvallisuusluokitus olisi ”salainen”, ei hänellä olisi pääsyä muihin resursseihin kuin niihin, joiden kategoria on sama kuin hänen käyttäjätunnukselleen määritetty kategoria.

MAC malli on kaikista malleista turvallisin. Sen huonona puolena on vaikea toimeenpano, toimiakseen se vaatii huomattavan paljon suunnittelua. Toinen mallin huono puoli on sen vaatima jatkuva hallinta ja ylläpito. (OWASP 2016; InfoSec Institute 2014.)

3.2 Discretionary Access Control

Discretionary Access Control (DAC) eroaa MAC mallista siten, että se antaa käyttäjille kontrollin hallita heidän itsensä omistamien resurssien pääsynhallintaa. DAC on tyypillisesti esimerkiksi käyttöjärjestelmien oletuksena käyttämä sääntömalli.

Turvallisuusetikettien sijaan, DAC mallin perustana toimii jokaisella resurssilla oleva pääsynhallintalista (ACL, Access Control List). Pääsynhallintalista sisältää tiedon käyttäjistä ja ryhmistä, joilla on pääsy kyseiseen resurssiin, sekä hallintatason (Level of access), jolla he voivat kyseistä resurssia käyttää. (OWASP 2016; InfoSec Institute 2014.)

Esimerkiksi käyttäjä A voi antaa käyttäjälle B kirjoitusoikeuden tiedostoon, käyttäjälle C vain lukuoikeuden ja ryhmälle 1 täydet oikeudet kyseiseen resurssiin. Ryhmän hallintataso periytyy automaattisesti kaikille kyseisen ryhmän käyttäjille.

3.3 Role Based Access Control

Role Based Access Control (RBAC) mallin perustana toimii järjestelmään määritetyt roolit, joille on määritetty tietyt oikeudet. Jokaisen käyttäjän pitää kuulua vähintään yhteen rooliin, mutta hän voi kuulua samaan aikaan useampaan eri rooliin. Kun käyt-

täjä kuuluu johonkin rooliin, perii hän kaikki kyseiselle roolille määritetyt oikeudet itselleen.

RBAC malli on paljon käytetty erityisesti yrityksissä, koska mallin perusajatus on helppo johtaa yritysmaailmaan. On helppo ajatella, että yksi järjestelmän rooli vastaa yhtä yrityksessä käytettyä työtitteliä, esimerkiksi kaikki yrityksen palveluksessa olevat talonmiehet kuuluvat ”talonmiehet” ryhmään. (OWASP 2016; InfoSec Institute 2014.) RBAC mallin ero DAC malliin verrattuna on se, että RBAC mallin näkökulmasta yksittäinen käyttäjä ei voi omistaa mitään, vaan kaikki oikeudet periytyvät käyttäjälle roolien kautta, johon hänet on määritetty.

4 TEKNIIKKA

4.1 SAML

SAML (Security Assertion Markup Language) on XML-kielen pohjalta kehitetty kuvauskieli ja sovelluskehys, jota käytetään nimensä mukaisesti entiteettien, kuten laitteiden tai ihmisten identiteettien vakuuttamiseen (to assert) (OASIS 2005C, 1). Se kehitettiin alun perin tärkeän informaation vaihtamista varten. SAML on hyvinkin uniikki teknologia ja se eroaa muista kilpailevista teknologioista siten, että SAML:n näkökulmasta luottosuhteet kahden applikaation välillä eivät perustu pelkästään sertifikaatteihin, jotka jokin kolmas osapuoli on luovuttanut, vaan mikä tahansa järjestelmä verkossa voi vakuuttaa toiselle järjestelmästä tietävänsä jotain henkilöstä tai asiasta. Vastaanottava pää voi päättää, luottaako se tähän tietoon vai ei.

4.2 SAML-kielen historia

Ensimmäisen SAML:n versio 1.0 julkaistiin marraskuussa vuonna 2002 ja samalla siitä tehtiin standardi Organization for the Advancement of Structured Information Standards (OASIS) toimesta. Vuotta myöhemmin julkaistiin versio 1.1, joka sisälsi vain pieniä korjauksia. (OASIS 2005C, 1)

Ensimmäinen merkittävä julkaisu, versio 2.0 julkaistiin marraskuussa vuonna 2005. Versio 2.0 muutti SAML:ia niin paljon, että 2.0 versio ei ole taaksepäin yhteensopiva versioiden 1.1 tai 1.0 kanssa.

4.3 Käsitteet

SAML-teknologiaan liittyy vahvasti seuraavat käsitteet: *Binding*, *Profile*, *idP*, *SP*, *SSO* ja *SLO*. Lyhenne idP tulee sanoista Identity Provider, joka tarkoittaa identiteetin tarjoajaa. idP on se järjestelmä, joka pitää yllä käyttäjätietorekisteriä ja varmistaa käyttäjän antamien autentikointitietojen oikeellisuuden kirjautumisprosessissa. Lisäksi idP:n tehtävä on tarjota identiteettitietoja eteenpäin.

Lyhenne SP tulee sanoista Service Provider, joka tarkoittaa palveluntarjoajaa. SP on se järjestelmä, joka hyödyntää idP:n autentikointipalveluja ja tietovarantoja. Lyhenne SSO tulee sanoista Single-Sign On, joka tarkoittaa kertakirjautumista. SSO on ominaisuus, jonka avulla loppukäyttäjä voi käyttää useampaa palvelua yhdellä kirjautumisella. Lyhenne SLO tulee sanoista Single Logout, joka tarkoittaa kertauloskirjautumista. SLO on ominaisuus, jonka avulla loppukäyttäjä voidaan kirjata ulos kaikista hänen käyttämistään palveluista lähes samanaikaisesti.

Binding eli SAML Protocol binding määrittelee, miten idP ja SP välittävät tietoa keskenään. Esimerkiksi kulkevatko sanomat HTTP-protokollan POST- vai GET-metodia käyttäen. SAML-standardi määrittää useita eri binding-vaihtoehtoja, edellä mainittujen POST- ja GET-bindingien lisäksi SAML-sanomia voidaan välittää esimerksi SOAP-protokollaa käyttäen.

Profile on SAML-standardin määrittelemä protokollan käyttötapaus. SAML 2.0 -standardi määrittelee useita eri profiileja. Esimerkiksi ”Assertion Query/Request” -profiili määrittää, miten järjestelmä voi hakea toisesta järjestelmästä tietoa tietyin kriteerein ja hakuehdoin. ”Web Browser SSO” -profiili taasen määrittää, miten Internet-selaimessa toimiva kertakirjautumisratkaisu on toteutettavissa SAML-protokollaa käyttäen.

Hyvä käytännön esimerkki idP:n ja SP:n välisestä suhteesta on julkishallinnon palvelut, joihin kirjautuminen tapahtuu pankkitunnuksilla verkkopankin kautta, ennen kuin pääsee käyttämään itse palveluita. Pankki toimii tässä tapauksessa identiteetin tarjoajana ja julkishallinnon palvelu palveluntarjoajana.

Ei ole myöskään mitenkään poissuljettua, etteikö idP-järjestelmä voisi myös tarjota muitakin palveluita kuin identiteetin ja käyttäjärekisterin, tai etteikö SP voisi tarjota

kirjautumispalveluita. Käsitteet idP ja SP eivät siis ole absoluuttisia totuuksia vaan ne ovat aina suhteellisia tarkasteltavaan autentikointiprosessiin nähden.

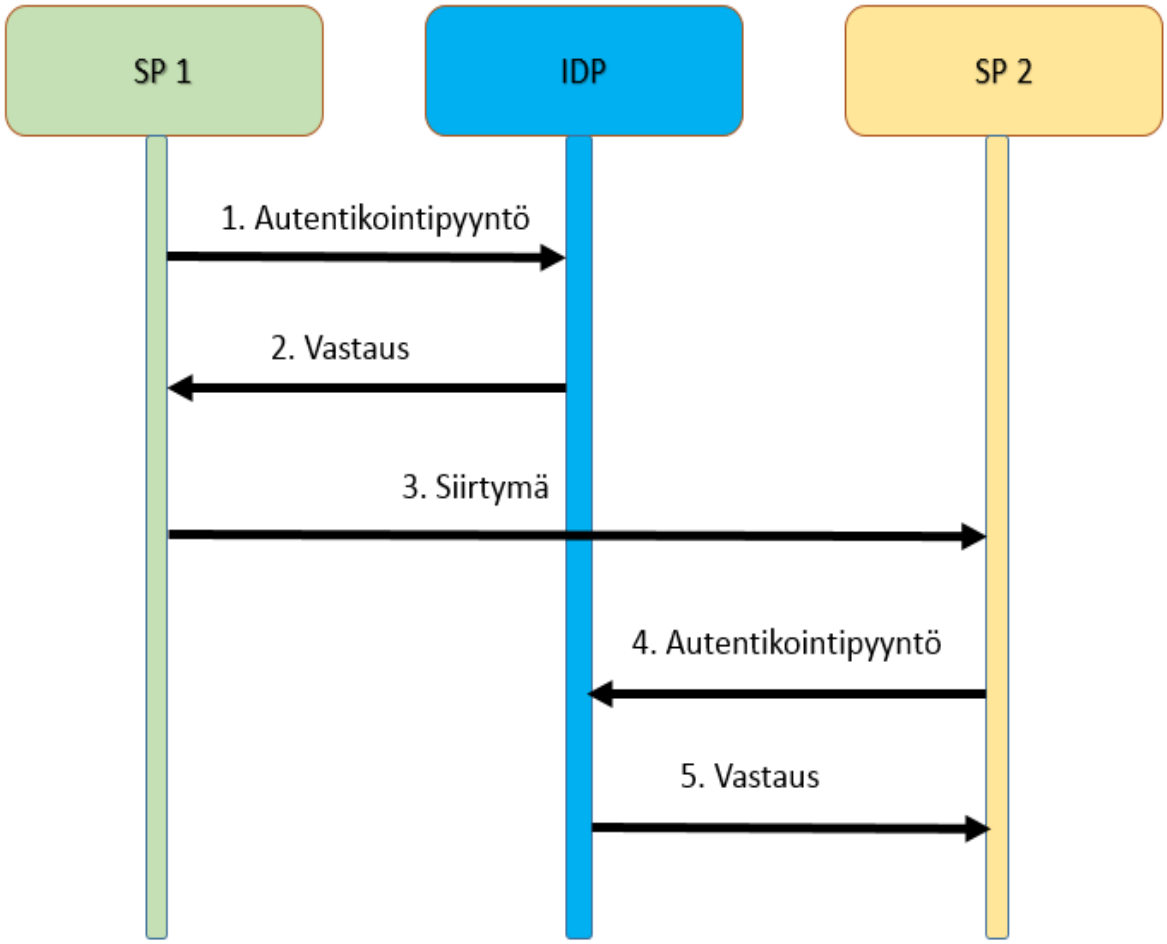
4.4 Käyttökohteet

SAML on hyvä teknologia, kun halutaan jakaa riskialtista tietoa kuten ihmisten henkilötietoja kahden tai useamman eri järjestelmän välillä. Tämä siitä syystä, että kaikki SAML-viestit voidaan sekä allekirjoittaa, joka takaa viestien eheyden osapuolten välillä, sekä salata, joka varmistaa, että vain haluttu vastaanottaja voi purkaa viestin sisällön. On kuitenkin tärkeää ymmärtää ennen kuin teknologia otetaan käyttöön, että SAML-protokolla ei itsessään tarjoa IAM-sovelluksiin yleisesti liitettyjä ominaisuuksia, kuten käyttäjän luontia, salasanan vaihtoa tai käyttöoikeuksien hallintaa, vaan nämä palvelut pitää jokaisen palveluntarjoajan toteuttaa itse parhaaksi näkemällään tavalla.

SAML-standardi määrittelee profiilit, joiden avulla internet-selaimessa toimiva kertakirjautuminen, sekä kertauloskirjautuminen ovat toteutettavissa. Nämä ovatkin yleisimmin käytetyt SAML-standardin määrittämät profiilit.

4.4.1 Kertakirjautuminen

Tarkastellaan SAML-standardin määrittämää Web Browser SSO -profiilin mukaista kertakirjautumisprosessia.



Kuva 1. Kertakirjautumisprosessin kulku

Kuvassa 1. on kuvattu seuraavat kertakirjautumisen vaiheet:

1. Käyttäjä on palveluntarjoajan SP1 sivulla ja haluaa autentikoitua palveluun. Palveluntarjoajan järjestelmä luo SAML-autentikointiviestin, *AuthnRequest*, joka käyttäjän selaimen välityksellä lähetetään idP:lle. IdP validoi lähetetyn autentikointiviestin ja esittää käyttäjälle kirjautumislomakkeen.

2. Käyttäjä autentikoituu idP-järjestelmään onnistuneesti ja hänelle luodaan istunto. idP luo SAML-vastauksen, *AuthnResponse*, joka pitää sisällään tiedon onnistuneesta kirjautumisesta sekä autentikoituneen käyttäjän tiedot. Tämä viesti välitetään takaisin autentikointipyynnön tekijälle eli SP1:lle, joka luo käyttäjälle istunnon omaan järjestelmäänsä idP:ltä saatujen tietojen pohjalta.

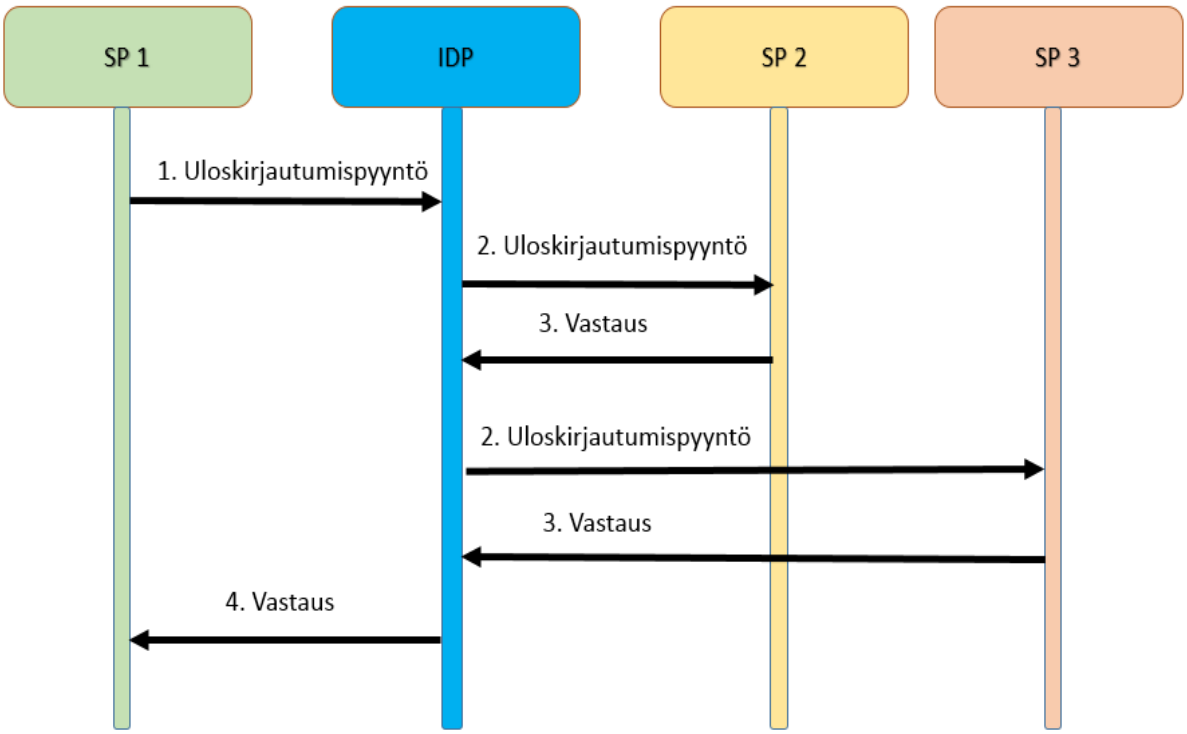
3. Käyttäjä siirtyy toiseen palveluun, SP2.

4. Kun käyttäjä yrittää autentikoitua palveluun SP2, luo järjestelmä autentikointipyynnön, joka välitetään käyttäjän selaimen avulla idP:lle.

5. idP vastaanottaa autentikointipyynnön, validoi sen ja huomaa, että käyttäjällä on jo istunto voimassa, joten kirjautumislomaketta ei tarvitse näyttää tällä kertaa. idP luo vastausviestin ja lähettää sen takaisin SP2:lle, joka luo käyttäjälle istunnon omaan järjestelmäänsä.

4.4.2 Kertauloskirjautuminen

Tarkastellaan SAML-standardin määrittämää Single Logout -profiilin mukaista kertauloskirjautumisprosessia.



Kuva 2. Kertauloskirjautumisprosessin kulku

Kuvassa 2. on kuvattu kertauloskirjautumisen seuraavat vaiheet:

1. Käyttäjä on SSO:n avulla kirjautunut useampaan eri järjestelmään saman idP:n kautta. Käyttäjä päättää lopettaa asioinnin palvelussa SP1 ja päättää kirjautua ulos palvelusta. SP1 lähettää idP:lle SAML-uloskirjautumispyynnön *LogoutRequest*.

2. idP validoi vastaanotetun uloskirjautumispyynnön, luo oman uloskirjautumispyynnön ja lähettää sen kaikkiin muihin palveluihin, joihin käyttäjä on kirjautunut idP:n kautta.

3. Jokainen palveluntarjoaja lähettää vastauksen idP:n tekemään uloskirjautumispyyntöön.

4. Lopuksi idP tuhoaa käyttäjän istunnon omasta järjestelmästään ja lähettää vastauksen alkuperäisen uloskirjautumispyynnön tekijälle eli SP1:lle. (OASIS 2009A, 59)

4.5 SAML ja tietoturva

SAML-protokolla käyttää hyväksi World Wide Web Consortium (W3C) yhteisön määrittämää XML Signature -teknologiaa, joka standardoitu XML-kieleen pohjautuva digitaalinen allekirjoitustekniikka. Kyseisen tekniikan avulla voidaan allekirjoittaa mikä tahansa XML-muotoinen dokumentti.

Toinen käytetty teknologia on saman yhteisön määrittämä XML Encryption, jonka avulla pystytään salaamaan XML-dokumentin sisältöä. Salaus voidaan purkaa vain vastaanottajan yksityisellä avaimella eli sertifikaatilla.

Digitaalisen allekirjoituksen ja salauksen perustana toimii X.509 standardin mukainen sertifikaatti. Sertifikaatti muodostuu kahdesta osasta: julkisesta- ja yksityisestä avaimesta, jotka muodostavat parin.

Sertifikaatti sisältää muun muassa seuraavat tiedot:

- liikkeellelaskija (Issuer), tarkoittaa sertifikaatin myöntäjää

- omistaja (Subject), taho jolle sertifikaatti on myönnetty liikkeellelaskijan toimesta

- voimassaoloaika eli päivämääräväli, jonka aikana sertifikaatti on voimassa. Jos sertifikaatti vanhenee, sitä ei voida enää pitää luotettavana

- sertifikaatissa käytetty algoritmi ja pituus, eli kuinka monta bittiä yhdessä sertifikaatin avaimessa on.

4.5.1 Digitaalisen allekirjoituksen muodostaminen

Digitaalisen allekirjoituksen muodostaminen tapahtuu seuraavasti:

1. SAML-viestistä muodostetaan tiiviste (Message digest) yksisuuntaisella tiiviste-funktiolla (esim. SHA-1).

2. Lyhennelmä salataan lähettäjän sertifikaatin yksityisellä avaimella ja lähetetään eteenpäin.

3. Vastaanottaja purkaa lyhennelmän sertifikaatin julkisella avaimella.

4. Samalla tiiviste-funktiolla, jolla muodostettiin alkuperäinen tiiviste, muodostetaan uusi tiiviste vastaanottajapäässä.

5. Jos vastaanotetun SAML-sanoman tiiviste ja vastaanottajan muodostama tiiviste vastaavat toisiaan, voidaan olla varmoja siitä, että sanoman sisältö ei ole muuttunut matkan varrella.

(Altova 2016.)

4.6 SAML-sanomat

SAML-standardi määrittelee yhteensä 4 eri tyyppistä sanomaa Web Browser SSO- ja Single Logout -profiilien toteuttamiseen, joita voidaan lähettää SP:n ja idP:n välillä. Eri sanomatyypit ovat nimeltään *AuthnRequest*, *LogoutRequest*, *Response* ja *Logout-Response*.

Toteutettaessa SAML 2.0 -standardin mukaista POST-tyylistä liittymää (Binding) on noudatettava seuraavaa sääntöä pyynnön tai vastauksen muodostamisessa:

Jos lähetettävä sanoma on pyyntö, pitää SAML-sanoman sijaita HTML-lomakkeella input-tyyppisessä HTML-elementissä, jonka name-attribuutti on ”*SAMLRequest*”. Jos lähetettävä viesti on vastaus, niin name-attribuutin arvo pitää olla silloin ”*SAMLRes-ponse*”.

SAML-sanomaa ei pidä lähettää XML-muotoisena HTML-lomakkeelta, vaan ennen lähetystä se pitää muuntaa 64-kantaiseksi merkkijonoksi. (OASIS 2009B, 22)

```
<form action="http://localhost:8040/SSO/" method="POST">
    <input type="hidden" name="SAMLResponse" value="PD94bWwgdmVyc21vj..."/>
</form>
```

Kuva 3. SAML-sanoma HTML-lomakkeella (lyhennetty)

Seuraavissa kappaleissa käsitellään eri tyyppisiä SAML-sanomia tarkemmin. Koska SAML-standardi on hyvin laaja, käsitellään sanomien sisältö Kuntalaistilin kannalta oleellisella laajuudella.

4.6.1 AuthnRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest>
    <saml2:Issuer>...</saml2:Issuer>
    <ds:Signature>... <ds:Signature>
    <saml2p:Extensions>...</saml2p:Extensions>
</saml2p:AuthnRequest>
```

Kuva 4. AuthnRequest-sanoman rakenne

AuthnRequest on pyyntö, jonka SP lähettää idP:lle. Tarkemmin sanoen *AuthnRequest* on autentikointipyyntö, joka tarkoittaa, että palveluntarjoaja haluaa autentikoida itsensä identiteetintarjoajalle.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest
    AssertionConsumerServiceURL="http://localhost:8040/spreturn"
    Destination="http://localhost/IDP/in"
    ID="_343933313434353736303832313932323634"
    IssueInstant="2015-07-16T08:06:47.113Z" Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
```

Kuva 5. *AuthnRequest*-sanoman juurielementti

Seuraavaksi tarkastellaan *AuthnRequest*-tyyppistä viestiä pala kerrallaan, joka on kuvattu kuvassa 5. Samalla käydään läpi viestin rakennetta ja tärkeimpiä tietoja.

Viesti alkaa elementillä <*saml2p:AuthnRequest*>, joka on viestin juurielementti. Kyseisen elementin attribuuttina määritellään autentikointiprosessin kannalta neljä tärkeintä tietoa, poislukien xmlns:saml2p, joka määrittää käytettävän XML-nimiavaruuden, joten kyseinen attribuutin arvo on SAML 2.0 -viesteissä aina vakio.

Attribuutti *AssertionConsumerServiceURL* määrittää, mihin osoitteeseen autentikointipyynnön vastaus, *AuthnResponse* lähetetään idP:n toimesta.

Attribuutti *Destination* kertoo, mihin osoitteeseen kyseinen *AuthnRequest*-viesti lähetettiin. *Destination* attribuutti ei ole pakollinen, mutta jos kyseinen attribuutti on viestissä, idP:n pitää tarkistaa, että sen arvo on oikein. (OASIS 2009A, 36)

Attribuutti *ID* on pakollinen viestin yksilöivä tunnistetieto. SAML-protokalla ei itsessään tarjoa keinoa generoida yksilöiviä tunnisteita, joten jokaisen järjestelmän on siis toteutettava tunnisteen generointi itse. (OASIS 2009A, 36)

Attribuutti *IssueInstant* on pakollinen tieto ja se kertoo, millä päivämäärällä ja kellonajalla kyseinen SAML-viesti on luotu. Aika annetaan UTC formaatissa ja ilman aikavyöhykettä. (OASIS 2009A, 9)

Attribuutti *Version* on pakollinen tieto ja se kertoo käytetyn SAML-standardin versionumeron. Tämä tieto on tärkeä, koska versio 2.0 ei ole taaksepäin yhteensopiva vanhempien versioiden 1.1 ja 1.0 kanssa. (OASIS 2009A, 36)

```
<saml2:Issuer
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">sp.kuntalaistili.fi
</saml2:Issuer>
```

Kuva 6. *AuthnRequest*-sanoman <*Issuer*>-elementti

Seuraava elementti viestihierarkiassa on <*Issuer*>, jonka sisälle tulee viestin lähettäneen järjestelmän nimi, joka on samalla järjestelmän yksilöivä tunniste. SAML-standardissa suositellaan, että käytettävä *Issuer* arvo sisältää lähettävän palvelun oman domainnimen. (OASIS 2009A, 84)

Seuraavana viestihierarkiassa oleva elementti <*Signature*> sisältää viestin digitaalisen allekirjoituksen, jota käsitellään tarkemmin kappaleessa ”Digitaalinen allekirjoitus SAML-viestissä”.

```
<saml2p:Extensions>
    <language
        xmlns="urn:locale:SAML:2.0:extensions">fi
    </language>
</saml2p:Extensions>
```

Kuva 7. *AuthnRequest*-sanoman <*Extensions*>-elementti

Kuva 7. kuvaa viimeistä elementtiä viestihierarkiassa, joka on <*Extensions*> ja se ei ole pakollinen elementti *AuthnRequest*-sanomissa. <*Extension*>-elementin sisään voi jokainen toimittaja vapaasti lisätä omia laajennoksiaan SAML-viestiin. Yllä olevassa esimerkkiviestissä on lisätty haluttu käyttöliittymäkieli laajennokseksi <*language*>-elementin sisään. (OASIS 2009A, 37)

4.6.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response>
    <saml2:Issuer>...</saml2:Issuer>
    <ds:Signature>...</ds:Signature>
    <saml2p:Status>...</saml2p:Status>
    <saml2:Assertion>...</saml2:Assertion>
</saml2p:Response>
```

Kuva 8. *Response*-sanoman rakenne

Response on vastaus *AuthnRequest*-sanomaan, minkä idP lähettää SP:lle. *Response*-tyyppinen sanoma lähetetään, kun käyttäjä on autentikoitunut onnistuneesti identiteettitarjoajan palveluun. *Response* voidaan myös lähettää, jos käyttäjän autentikointi epäonnistui, esimerkiksi jos käyttäjän antamat autentikointitiedot olivat virheelliset tai prosessi keskeytettiin käyttäjän toimesta. *Response* lähetetään aina autentikointipyynnön tekijälle, eli *AuthnRequest*-viestin lähettäjälle.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response Destination="http://localhost:8040/spreturn"
    ID="_34363431363635313536393939383330383138"
    InResponseTo="_32373737353838333034323334323537373938"
    IssueInstant="2015-07-16T11:52:20.579Z" Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

Kuva 9. *Response*-sanoman juurilementti

Response-sanoman juurielementti on <*Response*>. Kuten voidaan huomata, kyseinen elementti sisältää joitain samoja attribuutteja kuin aikaisemmin käsitelty *AuthnRequest*-viesti. Tässä kappaleessa ei käsitellä enää uudestaan jo läpikäytyjä attribuutteja.

Ainut uusi tieto <*Response*>-elementin sisällä on *InResponseTo* attribuutti. Kyseisen attribuutin arvo viittaa sen SAML-sanoman yksilöivään tunnisteeseen, johon tämä *Response*-viesti on vastaus.

```
<saml2p:Status>
    <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</saml2p:Status>
```

Kuva 10. *Response*-sanoman <*Status*>-elementti

Viestihierarkian elementti <*Status*> kertoo SP:lle, onnistuiko käyttäjän autentikointi idP:tä vasten vai ei. <*Status*>-elementti on pakollinen kaikissa vastaussanomissa ja se voi sisältää seuraavat alielementit:

- <*StatusCode*>, ainoa pakollinen alielementti. SAML-standardissa määritetty tilakoodi.

- <*StatusMessage*>, idP-järjestelmän luoma sanallinen vapaamuotoinen tilaviesti.

- <*StatusDetail*>, tarkempi tilatietojen kuvaus.

(OASIS 2009A, 39)

```
<saml2:Assertion
        ID="_32333838313135353039353435353339323230"
        IssueInstant="2015-07-16T11:52:20.579Z"
        Version="2.0" xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml2:Issuer>http://idp.kuntalaistili.fi/</saml2:Issuer>
    <saml2:Subject>
        <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
            E3TJYN5LK4YXXPUJRXAHDMARLQD4KIT2@ldap.login
        </saml2:NameID>
    </saml2:Subject>
    <saml2:AttributeStatement>
        <saml2:Attribute Name="firstNames" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">Mikko</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name="lastName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">Mallikas</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">mikko.mallikas@test.com</saml2:AttributeValue>
        </saml2:Attribute>
        <saml2:Attribute Name="phoneNumber" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
            <saml2:AttributeValue
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">0123456</saml2:AttributeValue>
        </saml2:Attribute>
    </saml2:AttributeStatement>
</saml2:Assertion>
```

Kuva 11. *Response*-sanoman <*Assertion*>-elementti

Viestihierarkian seuraava elementti, <*Assertion*> sisältää kaiken tiedon autentikoituneeseen käyttäjään liittyen.

```
<saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
        E3TJYN5LK4YXXPUJRXAHDMARLQD4KIT2@ldap.login
    </saml2:NameID>
</saml2:Subject>
```

Kuva 12. <*Assertion*>-elementin alielementti <*Subject*>

<*Assertion*>-elementin sisältämä <*Subject*>-elementti kuvaa entiteetin, jota myöhemmin viestihierarkiassa olevat toteamukset, <*AttributeStatement*> elementit, koskevat. Tärkein <*Subject*>-elementin tiedoista on <*NameID*>, joka määrittää yksilöivän idP-järjestelmän istuntotunnisteen autentikoituneelle käyttäjälle. Kyseinen arvo on tärkeä palveluntarjoajien kannalta, koska uloskirjautumispyyntöjen pitää sisältää kyseinen tunniste.

```
<saml2:AttributeStatement>
    <saml2:Attribute Name="firstNames" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            Mikko
        </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name="lastName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            Mallikas
        </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            mikko.mallikas@test.com
        </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute Name="phoneNumber" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            0123456
        </saml2:AttributeValue>
    </saml2:Attribute>
</saml2:AttributeStatement>
```

Kuva 13. *Response*-sanoman <*AttributeStatement*>-elementti

Viestihierarkian seuraava elementti, <*AttributeStatement*> sisältää listan alielementtejä <*Attribute*>, joiden tarkoitus on kuvata <*Subject*>-elementissä määritettyä entiteettiä. Yllä olevassa esimerkkisanomassa on palautettu autentikoituneen käyttäjän etunimi-, sukunimi-, sähköposti-, puhelinumerotiedot. SAML-standardi ei määritä vaatimuksia, mitä tietoja käyttäjästä pitää palauttaa. Tämän takia palautuvat tiedot voivat olla laidasta laitaan idP-järjestelmästä riippuen.

4.6.3 LogoutRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:LogoutRequest>
    <saml2:Issuer>...</saml2:Issuer>
    <ds:Signature>...</ds:Signature>
    <saml2:NameID>...</saml2:NameID>
</saml2p:LogoutRequest>
```

Kuva 14. *LogoutRequest*-sanoman rakenne

LogoutRequest on SP:n idP:lle lähettämä uloskirjautumispyyntö. Uloskirjautumispyyntö lähetetään SP:n toimesta idP:lle silloin, kun käyttäjän istunto halutaan lopettaa idP-järjestelmässä. *LogoutRequest*-viesti lähetetään myös idP:n toimesta SP:lle silloin kun kertauloskirjautuminen eli SLO on käytössä.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:LogoutRequest
    Consent="urn:oasis:names:tc:SAML:2.0:consent:unavailable"
    Destination="http://localhost/IDP/in"
    ID="_353038333030333230353538343535735393136"
    IssueInstant="2015-07-16T11:40:10.231Z"
    Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
```

Kuva 15. *LogoutRequest*-sanoman juurielementti

Kun palveluntarjoaja lähettää uloskirjautumispyynnön idP:lle, pitää loppukäyttäjälle näyttää uloskirjautumisen vahvistuslomake riippuen annetusta *Consent*-attribuutin arvosta. Jos *Consent*-attribuutin arvo on ***urn:oasis:names:tc:SAML:2.0:consent:unavailable***, se tarkoittaa, että käyttäjältä ei ole vielä saatu vahvistusta uloskirjautumiseen, joten idP:n pitää näyttää vahvistuslomake. Jos taas attribuutin arvo on ***urn:oasis:names:tc:SAML:2.0:consent:obtained***, se tarkoittaa, että vahvistus uloskirjautumiseen on jo saatu ja käyttäjä voidaan kirjata ulos automaattisesti ilman vahvistusta.

Uloskirjautumispyyntö eroaa autentikointipyynnöstä siten, että uloskirjautumispyyntö ei sisällä attribuuttia *AssertionConsumerUrl*, joka kertoo mihin osoitteeseen vastausanoma, *LogoutResponse*, pitää lähettää. Jokaisen idP-järjestelmän on siis ylläpidettävä eri palveluntarjoajien uloskirjautumisvastausten paluuosoitteita parhaaksi katsomallaan tavalla.

```
    <saml2:NameID xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
    TEC35IDF75IMCL5VX7YHBR255JXTAVIQ@ldap.login
    </saml2:NameID>
</saml2p:LogoutRequest>
```

Kuva 16. *LogoutRequest*-sanoman <*NameID*>-elementti

Uloskirjautumispyyntöjen pitää sisältää jokin SAML-standardissa määritetty elementti, jossa kulkee autentikoituneen käyttäjän yksilöivä istuntotunniste idP-järjestelmässä. Mahdolliset elementit ovat: <*BaseID*>, <*NameID*> tai <*EncryptedID*>. Kyseinen istuntotunniste palautetaan palveluntarjoajille *AuthnResponse*-sanomassa. Tunnisteen perusteella idP tietää, kenen käyttäjän istuntoa ollaan päättämässä. (OASIS 2009A, 62).

4.6.4 LogoutResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:LogoutResponse>
    <saml2:Issuer>...</saml2:Issuer>
    <ds:Signature>...</ds:Signature>
    <saml2p:Status>...</saml2p:Status>
</saml2p:LogoutResponse>
```

Kuva 17. *LogoutResponse*-sanoman rakenne

LogoutResponse on uloskirjautumispyynnön vastausanoma. *LogoutResponse*-sanoma lähetetään idP:n toimesta SP:lle, tai kertauloskirjautumisen yhteydessä SP:n toimesta idP:lle. Vastaussanomassa kerrotaan, onnistuiko käyttäjän istunnon päättäminen kohdejärjestelmästä vai ei.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:LogoutResponse
    Destination="http://localhost:8040/spreturn"
    ID="_32383331303834343338373438313139393633"
    InResponseTo="_34353333323431383431303937363034373736"
    IssueInstant="2015-07-16T11:54:22.838Z" Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
```

Kuva 18. *LogoutResponse*-sanoman juurilementti

LogoutResponse-sanoman rakenne ei eroa oleellisesti *AuthnResponse*-sanoman rakenteesta, vaan käytössä on kaikki samat attribuutit ja elementit.

4.7 Digitaalinen allekirjoitus SAML-sanomissa

Digitaalinen allekirjoitus löytyy SAML-sanomista elementistä <*Signature*>. Se voidaan sisällyttää kaikkiin eri SAML-sanomatyyppeihin, mutta se ei ole pakollinen elementti. Jos viesti sisältää <*Signature*>-elementin, on vastaanottajan **aina** validoitava allekirjoituksen oikeellisuus. (OASIS 2009A, 16)

<*Signature*>-elementin skeema ei kuulu SAML-standardin piiriin, mutta sen rooli SAML-sanomissa on merkittävä tietoturvan ja luotettavuuden kannalta.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest
    AssertionConsumerServiceURL="http://localhost:8040/spreturn"
    Destination="http://localhost/IDP/in"
    ID="_34393331343435373630383231393232323634"
    IssueInstant="2015-07-16T08:06:47.113Z" Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
 <ds:Signature
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="#_34393331343435373630383231393232323634">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>jsQI5f8xtP2ousCqT4PFAbT79wE=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>XkbbuxksNY74+0rLp...</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>
                    MIIDJzCCAg8CBFTvCnMwDQYJKoZIhvc
                    NAQEFBQAwWDELMAkGA1UEBhMCRkkx...
                </ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
</saml2p:AuthnRequest>
```

Kuva 18. Allekirjoitettu *AuthnRequest*-viesti

<*CanonicalizationMethod*>-elementti kuvaa algoritmin, jolla <*SignedInfo*>-elementti pitää normalisoida, ennen XML-dokumentin validointia. Tämä siitä syystä, että XML-dokumentti voidaan kuvata usealla eri tavalla, esimerkiksi elementin attribuuttien järjestys voi muuttua dokumentissa, ilman että tietosisältö muuttuu. Pienetkin muutokset dokumentissa vaikuttavat tiivisteen laskemiseen, joten allekirjoitettu dokumentti on

ensin normalisoitava, jotta lähettäjä sekä vastaanottaja käsittelevät dokumenttia samassa muodossa. (Oracle 2010.)

<*SignatureMethod*>-elementti kertoo, millä algoritmilla allekirjoitus on muodostettu. Tässä esimerkissä käytetty algoritmi on RSA-SHA1. (Oracle 2010.)

<*Reference*>-elementti kertoo, mikä osa dokumentista on allekirjoitettu. Tässä elementin URI-attribuutin arvo viittaa SAML-viestin ID-attribuutin arvoon, mikä tarkoittaa, että koko SAML-sanoma on allekirjoitettu. <*Transform*>-alielementit kertovat, mitkä toimenpiteet allekirjoitetulle osalle pitää tehdä ennen tiivisteen laskemista. (Oracle 2010.)

<*Transform*>-elementin algoritmi ***http://www.w3.org/2000/09/xmldsig#enveloped-signature*** tarkoittaa, että allekirjoitukseen liittyvät elementit pitää poistaa. Algoritmi ***http://www.w3.org/2001/10/xml-exc-c14n#*** tarkoittaa, että elementit on normalisoitava.

<*DigestMethod*>-elementti kertoo, millä algoritmilla tiiviste on laskettu ja <*DigestValue*>-elementti kertoo lasketun tiivisteen arvon. (Oracle 2010.)

<*SignatureValue*>-elementin arvo on muodostettu allekirjoitus. Allekirjoituksen arvo saadaan laskettua dokumentin tiivisteestä <*DigestValue*> käyttämällä X509-standardin mukaista sertifikaattia apuna. (Oracle 2010.)

<*X509Certificate*>-elementin arvo on allekirjoituksen muodostaneen sertifikaatin julkinen avain, jolla allekirjoituksen oikeellisuus voidaan verifioida. Sertifikaatti esitetään 64-kantaisena merkkijonona.

5 IDP-KOMPONENTIN TOTEUTUS KUNTALAISTILIIN

Tämän opinnäytetyön käytännön tavoite on toteuttaa SAML-standardin mukainen idP-komponentti Kuntalaistilin uudelle arkkitehtuurille.

5.1 Identiteetin hallinnan merkitys Kuntalaistilissä

Kuntalaistili on Lahden kaupungin omistama ja Propentus Oy:n tuottama sähköinen palvelualusta julkiselle sektorille. Kuntalaistilin tarkoitus on koota kunnan kaikki sähköiset palvelut yhteen paikkaan, joka helpottaa loppukäyttäjän eli tavallisen kuntalaisen elämää.

Kuntalaistili vaatii käyttäjiltä ensimmäisellä käyttökerralla vahvan tunnistautumisen pankkitunnuksilla Vetuman kautta. Vetuma on kansalaisen tunnistus- ja maksamispalvelu, johon voivat liittyä kaikki julkishallintoon kuuluvat organisaatiot. (Valtori 2016.)

Vahvan tunnistautumisen jälkeen käyttäjälle luodaan käyttäjätunnus-salasana-pari, jonka avulla hän pystyy jatkossa kirjautumaan sisään Kuntalaistiliin ilman vahvaa tunnistautumista.

Kuntalaistilin näkökulmasta identiteetinhallinta tarkoittaa sitä, että kuka tahansa kolmas osapuoli pystyy tuottamaan Kuntalaistiliin palveluita. Kuntalaistilin keskitetty identiteetinhallinta helpottaa palveluiden tuottajien elämää tarjoamalla valmiit komponentit ja rajapinnat, joihin he voivat helposti integroitua. Kuntalaistilin hoitaessa identiteetinhallinta ja siihen liittyvät toiminnot taustalla, voivat palveluntarjoajat keskittyä kehittämään omia palveluitaan.

Kuntalaistili tarjoaa muun muassa seuraavat identiteetin hallintaan liittyvät taustatoiminnot:

- Käyttäjän identiteetin varmistaminen. Koska Kuntalaistili vaatii vahvan tunnistautumisen pankkitunnuksilla, voidaan olla varmoja käyttäjän identiteetistä.

- SAML-standardin mukainen idP-komponentti tarjoaa kertakirjautumis- ja kertauloskirjautumisratkaisut palveluiden tuottajille.

- Automaattinen käyttäjän tietojen päivitys ulkoisesta lähteestä takaa sen, että loppukäyttäjän tiedot ovat aina ajan tasalla.

5.2 Työkalut ja teknologiat

5.2.1 Grails

Grails on avoimen lähdekoodin Web-sovelluskehys, joka toimii Java-virtuaalikoneen päällä. Grailsin visio ja tavoite on moninkertaistaa sovelluskehittäjien tuottavuus sen tarjoamilla uusilla ominaisuuksilla. Integroitu oliotietokantamalli GORM eli Grails Object Relational Model, joka helpottaa tietokantaentiteettien hallintaa ja vähentää natiivien tietokantakyselyiden määrää. GORM pohjautuu Red Hat yhtiön kehittämään Hibernate nimiseen oliotietokanta-sovelluskehykseen. (grails.org 2016.)

Convention-over-configuration ajatusmalli, eli ”Käytäntö konfiguraatioiden sijaan”. Convention-over-configuration on ohjelmistosuunnittelu paradigma, joka painottaa nimeämiskäytäntöjä konfiguraatioiden sijasta. Vain jos jokin asia poikkeaa normaalista käytännöstä, pitää se konfiguroida erikseen. (grails.org 2016.)

Grails on rakennettu Groovy-ohjelmointikielen päälle. Groovy on dynaaminen olio-ohjelmointikieli, joka toimii Java-virtuaalikoneen päällä. Groovy on ikään kuin Java-ohjelmointikielen laajennos. Tämä tarkoittaa sitä, että Groovyä pystytään ajamaan siellä, missä Javaakin. Groovy on dynaaminen kieli, mikä tarkoittaa, että sitä voidaan kääntää sovelluksen ajon aikana. Tämä vähentää huomattavasti aikaa, jonka sovelluskehittäjä käyttää odottaessaan sovelluspalvelinten uudelleenkäynnistystä tai koodin kääntymistä Javalla. (the Groovy project 2016.)

5.2.2 OpenSAML

OpenSAML on Shibboleth-projektin tuottama avoimen lähdekoodin ohjelmistokirjasto, jonka on tarkoitus helpottaa sovelluskehittäjiä toteuttamaan SAML-standardin mukaisia toteutuksia. OpenSAML ei tarjoa valmista idP- tai SP-toteutusta, vaan se on pelkästään kokoelma apumetodeita. OpenSAML-kirjasto on saatavilla Java- ja C++ -ohjelmointikielille. (wiki.shibboleth.net 2015.)

OpenSAML tukee kaikkia virallisia SAML-versioita eli 1.0, 1.1 ja 2.0. Kuitenkin näistä enää 2.0 version tukea kehitetään aktiivisesti. (wiki.shibboleth.net 2015.)

5.3 Grails-perusteet

Grails-web-sovelluskehykseen liittyy kolme peruskäsitettä: Controller, View ja Domain.

5.3.1 Controller (Käsittelijä)

Controller, on vastuussa HTTP-pyyntöjen vastaanottamisesta ja käsittelystä. Controller luo tai valmistelee HTTP-vastauksen pyynnön lähettäjälle. Controller pystyy lähettämään vastauksen suoraan, tai View'n avulla. (grails.org 2016.)

Controllerilla voi olla useita metodeja, joita kutsutaan Grailsissa termillä "action". Grailsin nimeämiskäytäntöjen mukaan kaikki Controller-luokat pitää nimetä niin, että ne päättyvät "Controller" sanaan, esimerkiksi "FrontpageController". (grails.org 2016.)

5.3.2 View (Näkymä)

Groovy Servers Pages (tai GSP lyhyesti) on Grailsin käyttämä näkymätekniikka. Se on hyvin samankaltainen kuin ASP (Active Server Pages) tai JSP (JavaServer Pages). (grails.org 2016.)

GSP:lle on mahdollista antaa model eli malli, joka on joukko muuttujia, joita sitten käytetään apuna näkymän mallintamiseen. (grails.org 2016.)

GSP on tyypillisesti sekoitus HTML:ää, sekä Grailsin omia GSP-tunnisteita (tageja), jotka auttavat näkymän mallintamisessa. Grails tukee myös uusien tunnisteiden luomista, mikä tukee ohjelmistokehityksen "Don't repeat yourself" -filosofiaa (DRY).

5.3.3 Domain (Tietomalli)

Domain-luokat kuvaavat Grailsissa sovelluksen tietomallia. Domain-luokkien sisältöä ylläpidetään tietokannassa oliotietokantamallin perusajatuksen mukaisesti. Määritetyistä domain-luokista voidaan GORM:in avulla luoda automaattisesti tietokannan taulut.

5.3.4 Toimintaesimerkki

Luodaan uusi Controller nimeltä ”HelloController”, jolla on action nimeltä ”sayHello”. Kyseinen action ottaa vastaan yhden parametrin nimeltä ”name”, jonka arvo palautetaan näkymälle.

```
1  package test
2
3  class HelloController {
4
5      def sayHello() {
6          String name = params.name
7          return [name: name]
8      }
9  }
10
```

Kuva 19. HelloController

Luodaan ”sayHello” actionille vastaava näkymä ”sayHello.gsp”, joka tulostaa tekstin ”Hello ${name}!”, jossa {$name} on controllerin palauttama arvo.

sayHello.gsp

```
1
2  <p> Hello ${name}! </p>
```

Kuva 20. sayHello-näkymä

Kutsutaan actionia navigoimalla selaimella osoitteeseen: ”http://localhost:8080/Test/hello/sayHello?name=World”

Saadaan vastaukseksi: ”Hello World!”

5.3.5 Autentikointiesimerkki

Luodaan esimerkkisovellus, jossa on HTML-lomake, johon käyttäjä syöttää käyttäjätunnuksen ja salasanan. Palvelin palauttaa eri vastauksen käyttäjälle riippuen siitä, löytyikö käyttäjätunnus-salasana-parille riviä tietokannan ”Account”-taulusta.

```
1  package test
2
3  class Account {
4
5      String username
6      String password
7
8      static constraints = {
9      }
10 }
```

Kuva 21. Account-tietomalli

Luodaan käyttäjän tiliä kuvaava ”Account”-tietomalli sovellukseen. Tietomalli koostuu kahdesta merkkijonotyyppisestä tiedosta: ”username” ja ”password”, kuten kuvasta 21. näkyy.

Lisätään tietokantaan pari testikäyttäjää:

id	version	password	username
1	0	sala123	mattim
2	0	qwerty12	test

Kuva 22. Tietokannan Account-taulu

Luodaan Controller nimeltä ”AccountController”, jolle lisätään kaksi actionia: ”login” ja ”loginForm”**.** ”login” action ottaa vastaan kaksi parametriä, käyttäjätunnuksen ja salasanan, ja vertaa löytyykö tietokannan Account-taulusta riviä, jolla on vastaavat tiedot. Tieto siitä, löytyikö riviä annetuilla tiedoilla, palautetaan näkymälle.

```
def login() {
    String username = params.username
    String password = params.password

    String query = "FROM Account AS acc WHERE acc.username = '${username}' "
    query += "AND acc.password = '${password}'"
    boolean found = !Account.executeQuery(query).isEmpty()

    return [found: found]
}
```

Kuva 23. AccountControllerin ”login”-action

```
<g:if test="${found == true}">
    <p> Käyttäjätunnus ja salasana oikein! </p>
</g:if>
<g:else>
    <p> Käyttäjätunnus tai salasana väärin! </p>
</g:else>
```

Kuva 24. ”login”-actionin näkymä

”login”-actionin näkymä tulostaa käyttäjälle eri tekstin riippuen siitä, löytyikö käyttäjätunnus-salasana-parille riviä tietokannasta, vai ei.

”loginForm”-action ei ota mitään parametrejä vastaan, vaan palauttaa aina saman staattisen HTML-näkymän käyttäjälle, mikä on tässä esimerkissä kirjautumislomake. Lomakkeella on kaksi <input>-kenttää, joihin käyttäjä syöttää käyttäjätunnuksen sekä salasanan sekä painonappi. Kun käyttäjä näpäyttää painonappia, lomake lähetetään palvelimelle. Palvelimella AccountControllerin ”login”-action käsittelee lomakkeen sisällön.

```
def loginForm() {

}
```

Kuva 25. AccountControllerin ”loginForm”-action

```
<form action="http://localhost:8080/Test/account/login" method="POST">

    <label>Käyttäjätunnus:</label>
    <input name="username" type="text"/>

    <br/>
    <br/>

    <label>Salasana:</label>
    <input name="password" type="text"/>

    <br/>
    <br/>

    <input value="Kirjaudu sisään" type="submit"/>

</form>
```

Kuva 26. ”loginForm”-actionin näkymä

5.4 Tulokset ja päätelmät

IdP-komponentin toteutus Kuntalaistiliin aloitettiin kesällä 2016 ja se valmistui arvioitujen työmäärien puitteissa kesän aikana, joten mielestäni projekti sujui onnistuneesti. Ennen projektin aloitusta SAML-teknologia oli minulle jonkin verran tuttu ennestään, mutta syvällistä ymmärrystä minulla ei ollut. Koska SAML-standardi on todella laaja, en voi vieläkään sanoa osaavani kaikkea, mutta Kuntalaistilin kannalta oleelliset standardin määrittämät profiilit, Web Browser SSO ja Single Logout, tunnen osaavani täysin.

Kuntalaistilin idP-komponenttiin tehtiin syksyn 2016 aikana testi-integraatio kolmannen osapuolen toimesta ja toteutus todettiin toimivaksi. Jatkokehityskohteena on lisätä laajennoksia, joiden avulla palveluntarjoaja pystyy rajaamaan tietyillä oikeuksilla olevia käyttäjiä autentikoitumasta idP:n kautta.

LÄHTEET

Altova. 2016. Technology Primer: XML Signature Technology Overview. Saatavissa: https://www.altova.com/XML_signature_technology_primer.html [viitattu: 4.10.2016].

grails.org. 2016. The Grails Framework – Reference Documentation. Saatavissa: http://docs.grails.org/ [viitattu 4.10.2016].

InfoSec Institute. 2014. Access Control: Models and Methods. Saatavissa: http://resources.infosecinstitute.com/access-control-models-and-methods/ [viitattu 4.5.2016].

OASIS. 2009A. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 – Errata Composite. Saatavissa: https://www.oasis-open.org/committees/download.php/35711/sstc-saml-core-errata-2.0-wd-06-diff.pdf [viitattu 6.6.2016]

OASIS. 2009B. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0 – Errata Composite. Saatavissa: https://www.oasis-open.org/committees/download.php/35387/sstc-saml-bindings-errata-2.0-wd-05-diff.pdf [viitattu 10.11.2016].

OASIS. 2005C. SAML Executive Overview. Saatavissa: https://www.oasis-open.org/committees/download.php/11785/sstc-saml-exec-overview-2.0-draft-06.pdf [viitattu 10.11.2016].

Oracle. 2010. Example of XML Signature. Saatavissa: https://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.6/tutorial/doc/XMLDigitalSignatureAPI7.html [viitattu 4.10.2016].

OWASP. 2016. Access Control Cheat Sheet. Saatavissa: https://www.owasp.org/index.php/Access_Control_Cheat_Sheet [viitattu 4.5.2016].

The Groovy project. 2016. Saatavissa: http://www.groovy-lang.org/ [viitattu 4.10.2016].

Valtori. 2016. Kansalaisen tunnistus- ja maksamispalvelu Vetuma. Saatavissa: http://www.valtori.fi/palvelut/vetuma [viitattu 10.11.2016].

wiki.shibboleth.net. 2015. Saatavissa: https://wiki.shibboleth.net/ [viitattu 4.10.2016].

KUVALUETTELO

LIITTEET

LAHDEN KUNTALAISTILI

https://www.palvelutarjotin.fi/

KOTKAN, HAMINAN JA PYHTÄÄN KUNTALAISTILI

https://ekymi.fi/

KOUVOLAN KUNTALAISTILI

https://ekouvola.fi/

PORIN KUNTALAISTILI

https://ekunta.fi/

KOKKOLA KUNTALAISTILI

https://ekokkola.fi/

JYVÄSKYLÄN KUNTALAISTILI

https://oma.jyvaskyla.fi