AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan tutkinto-ohjelma

ULOSKIRJAUTUMINEN SHIBBOLETHISSA

Kandidaatintyö

Asko Tontti

Tietotekniikan laitos
Espoo 2010

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan tutkinto-ohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä: Asko Tontti

Työn nimi:
Uloskirjautuminen Shibbolethissa

Päiväys: 20. joulukuuta 2010 **Sivumäärä:** 37
Pääaine: Tietoliikenneohjelmistot **Koodi:** T3005

Vastuuopettaja: Professori Ilkka Niemelä
Työn ohjaaja: TkL Sanna Suoranta, Aalto-yliopisto Tietotekniikan laitos

Työn aiheena on tunnistaminen Internetissä ja painopisteenä on erityisesti uloskirjautuminen. Työssä käydään läpi, miten sisään- ja uloskirjautuminen toteutetaan WWW-palvelussa, mitä ongelmia uloskirjautumiseen liittyy teorian näkökulmasta, miten se toimii käytännössä ja miten havaittuja ongelmia voi ratkaista.

Tutkinnan kohteena työssä on TKK:sta ja yliopistomaailmasta tuttu Shibboleth-ohjelmisto, joka pohjautuu laajasti käytettyyn SAML-standardiin. Tutustuttaessa TKK:lla paljon käytettyihin WWW-palveluihin, esimerkiksi Noppaan ja Oodiin, huomataan, että todellisuudessa uloskirjautuminen toimii eri tavalla eri palveluissa.

Työssä käydään läpi käyttäjän ja käyttöliittymän merkitystä toimivan uloskirjautumisen toteuttamisessa. Lisäksi työssä tutustutaan WWW-selaimien käyttämiin evästeisiin, istuntojen hallintaan AJAX-tekniikalla ja CoSign-ohjelmistoon, jonka kehittäjät ovat keksineet erilaisen ratkaisun uloskirjautumiseen. Työssä pohditaan myös sovellusten räätälöintiä, alaan liittyvää standardointityötä ja testausta.

Käyttäjien koulutuksella, käyttöliittymien kehittämisellä, sovellusten räätälöinnillä ja testauksella päästään jo pitkälle uloskirjautumisen ongelmien ratkaisussa. Tietoturvan mielessä tämä ei ole vielä täysin riittävä ratkaisu, vaan lähinnä tyydyttävä. Standardointityö ja Shibboleth IdP:n tulevat versiot ovat avainasemassa loppujen ongelmien ratkaisussa.

Avainsanat: Shibboleth, SAML, todentaminen, uloskirjautuminen, kertauloskirjautuminen, SLO
Kieli: Suomi

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Program of Computer Science and Engineering

ABSTRACT OF
BACHELOR'S THESIS

Author: Asko Tontti

Title of thesis:
Logging out with Shibboleth

Date: December 20, 2010 **Pages:** 37
Professorship: Data Communications Software **Code:** T3005
Supervisor: Professor Ilkka Niemelä
Instructor: Lic.Tech. Sanna Suoranta, Aalto University Department of Computer Science and Engineering

The subject of this thesis is identification on the Internet and the focus of the work is on logout in particular. This thesis examines how login and logout is implemented in web services, what problems there are in logout in theory, how it works in practice and how the identified problems can be solved.

The object of the study is the Shibboleth application familiar from TKK and academia. Shibboleth is based on the widely used SAML standard. Learning from the much used web services at TKK, like Noppa and Oodi, we see that in practice logout works differently in different services.

This thesis considers the significance of the user and the user interface for implementing the logout process correctly. In addition, this study looks at the cookies used by web browsers, session management with AJAX technology, and the CoSign software, whose developers have implemented a different kind of solution for single logout (SLO). This thesis examines also the adaptation of web applications, the standardization work for improving logout, and testing.

User training, user interfaces, the adaptation of the applications, and testing go a long way to solve the problems of logout. With security in mind, this is not yet a fully adequate solution, but only satisfactory. The standardization and the future versions of Shibboleth IdP will play a key role for solving the rest of the problems.

Keywords: Shibboleth, SAML, authentication, logout, single logout, SLO, global logout
Language: Finnish

Alkusanat

> *What's in your hands, I think and hope, is intelligence: the ability to see the machine as more than when you were first led up to it, that you can make it more.*

Kiitokset Sanna Suorannalle työn ohjaamisesta ja esimiehelleni Christopher Ariyolle sekä CSC – Tieteen tietotekniikan keskukselle siitä, että saatoin käyttää osittain työaikaa tämän työn tekemiseksi. Kiitokset Mikael Lindenille, Arto Tuomelle, Matti Liljavirralle, Peter Jenkinsille ja kielikeskuksen opettajille kommenteista. Lisäksi kiitokset Marjatalle ja Kirsille heidän tuestaan.

Espoossa 25. marraskuuta 2010

Asko Tontti

Käytetyt lyhenteet

SSO	Single Sign-On; yhdellä sisäänkirjautumisella pääsee moneen eri palveluun, kertakirjautuminen
SLO	Single Logout; yhdellä uloskirjautumisella ulos useasta eri palvelusta, kertauloskirjautuminen
SAML	Security Assertion Markup Language; XML-pohjainen kieli tietoturvatietojen välittämiseen
IdP	Identity Provider; tunnistustietojen tarjoaja, kotiorganisaatio
SP	Service Provider; palveluntarjoaja
SSL	Secure Socket Layer; alunperin Netscapen kehittämä tietoliikenneyhteyksien suojausmenetelmä
TLS	Transport Layer Security; IETF:n standardoima versio SSL:stä
XML	Extensible Markup Language; W3C:n standardoima kieli rakenteellisen tiedon tai dokumenttien kuvaamiseen tietokoneen ymmärtämässä muodossa
SGML	Standard Generalized Markup Language; XML-kielen edeltäjä
WWW	World Wide Web; Internetissä kokoelma hypertekstidokumentteja, jotka viittaavat toisiinsa linkeillä
HTML	HyperText Markup Language; SGML-pohjainen merkkauskieli linkitettyjen WWW-sivujen kuvaukseen
HTTP	HyperText Transfer Protocol; tiedonsiirtoprotokolla dokumenttien välittämiseen WWW-palvelimen ja -selaimen välillä
SOAP	Simple Object Access Protocol; Web-palveluille suunniteltu tietoliikenneprotokolla proseduurien etäkutsumiseen

URL	Uniform Resource Locator; kertoo, missä määritelty resurssi sijaitsee ja millä menetelmällä sen voi hakea
AJAX	Asynchronous JavaScript and XML; joukko menetelmiä, joiden avulla selaimella voi toteuttaa interaktiivisia WWW-sovelluksia
IETF	Internet Engineering Task Force; Internetin standardointiin keskittyvä avoin organisaatio
W3C	World Wide Web Consortium; WWW:n standardointiin keskittyvä avoin organisaatio
OASIS	Organization for the Advancement of Structured Information Standards; alan toimijoiden yhteenliittymä, joka kehittää avoimia standardeja informaatioyhteiskunnan tarpeisiin
RFC	Request for Comments; IETF:n julkaisema dokumentti, joka kuvaa teknisestä näkökulmasta Internetiä
TKK	Aalto-yliopiston Teknillinen korkeakoulu
IP	Internet Protocol; Internetin pääkommunikointiprotokolla, joka mahdollistaa pakettien välittämisen verkosta toiseen
JAAS	Java Authentication and Authorization Service; Javan turvallisuusjärjestelmä, joka mahdollistaa sovelluksen todennus- ja valtuutusmenetelmien vaihtamisen sovellusta muuttamatta
LDAP	Lightweight Directory Access Protocol; Internetprotokolla hakemistopalveluiden tietojen hakemiseen ja muuttamiseen
JDBC	Java Database Connectivity; Javan ohjelmointirajapinta tietokantoja varten
ID-FF	Identity Federation Framework; Liberty Alliancen standardi käyttäjän identiteetin välittämiseen organisaatioiden välillä

Sisältö

Luku 1

Johdanto

The Computer made me do it.

Palvelut ja sovellukset siirtyvät yhä kiihtyvällä nopeudella Internetiin. Tämän seurauksena tunnistautumisesta on tullut entistä tärkeämpi osa Internetin palveluita, ja tunnistukseen liittyvät tarpeet ja vaatimukset ovat lyhyessä ajassa kasvaneet huomattavasti. Viime vuosikymmenen aikana Internetissä tapahtuvaa tunnistusta varten on kehitty monia uusia ratkaisuita. Yksi tällainen ratkaisu on Shibboleth [13], joka on hyvä, alan standardeja tukeva ohjelmisto. Shibbolethissa, ja muutenkin tunnistautumisessa, on vähemmälle huomiolle jäänyt uloskirjautuminen, vaikka se on merkittävä osa tunnistusratkaisua. Tässä työssä tutustumme kertauloskirjautumiseen Shibbolethissa.

Yliopistoissa on ymmärretty jo aikaisessa vaiheessa organisaatiorajat ylittävän käyttäjähallinnon ja tunnistamisen tärkeys. Yhdysvalloissa yliopistot ovat yhdessä kehittäneet menetelmiä ja työkaluja käyttäjien hallintaan. Työ on pääasiassa tapahtunut Internet2-yhteenliittymän [12] sateenvarjon alla ja tuloksena on ollut esimerkiksi Shibboleth-ohjelmisto ja eduPerson-skeema [14]. Suomessa on myös tehty vastaavaa työtä Funetin puitteissa. Toiminta alkoi Haka-projekteilla, -tapaamisilla sekä -koululla, ja myöhemmin mukaan tuli Haka-luottamusverkosto [6], joka alkuperältään rakentui Shibboleth-ohjelmiston päälle [18]. Nyt Hakassa ollaan siirrytty SAML2-aikaan (Security Assertion Markup Language 2.0), joka on laajentanut mahdollisten ohjelmien valikoimaa.

Hakasta ja Shibbolethista on saatu yli viidessä vuodessa erinomaisia tuloksia ja kokemuksia. Ongelmilta ei kuitenkaan olla vältytty. Yksi keskeinen ongelma on uloskirjautuminen. Aluksi uloskirjautuminen ei ollut mahdollista, sillä Shibboleth ei tukenut sitä. Näin ollen Haka-luottamusverkosto joutui määrittelemään omat käytännöt uloskirjautumiseen. SAML2 toi tullessaan myös omat määritelmänsä,

mutta Shibboleth IdP (Identity Provider) ei vieläkään tue SAML2:n kertauloskirjautumista, mutta mahdollisesti tuleva IdP 3.0 ratkaisee tämän ongelman, ainakin osittain.

Periaatteessa uloskirjautuminen on yksinkertainen asia, mutta WWW:n (World Wide Web) tilattomassa maailmassa näin ei ole. Omat lisähaasteensa tuovat kertakirjautuminen ja organisaatiorajojen rikkominen. Perinteiset WWW-sovellukset vaativat usein räätälöintiä, jotta uloskirjautuminen saadaan toimimaan luotettavasti ja oikein. Uloskirjautumisen epäonnistuessa sekä käyttäjän että palvelun tietoturva ovat vaarassa. Seuraavalle yhteiskäyttöisen tietokoneen käyttäjälle tai hyökkääjälle avautuu mahdollisuus hyödyntää auki jäänyttä istuntoa.

Käytännön tutkimuskohteena työssä on joukko Aalto-yliopiston Teknillisen korkeakoulun (TKK) ja Hakan WWW-palveluita. Shibbolethia käytetään TKK:lla myös sisäisten palveluiden kertakirjautumisratkaisuna. TKK:n keskitetysti hankitut WWW-palvelut on pääsääntöisesti vuoden 2005 jälkeen räätälöity käyttämään Shibbolethia sisäänkirjautumisessa. TKK:n sisäisistä ja Hakaan kytketyistä palveluista tutustumme tarkemmin kirjastojen Nelliportaaliin, Noppaan, Oodiin, Wikiin ja TKK:n henkilökuntasivuihin. Näiden kaikkien uloskirjautuminen poikkeaa toisistaan tavalla tai toisella, esimerkiksi tapahtuuko paikallinen uloskirjautuminen vai kertauloskirjautuminen tai jääkö jokin istunto auki uloskirjautumisessa.

Ratkaisuina uloskirjautumisen ongelmiin käymme läpi käyttöliittymän vaikutuksia, sovelluksen vaatimia muutoksia, kirjautumistietojen tallentamisen merkitystä, CoSign-ohjelmistoa ja uusimpien WWW-tekniikoiden tuomia mahdollisuuksia. Käyttöliittymällä on merkittävä asema uloskirjautumisen toteuttamisessa. Sen pitää olla helppo ja yksinkertainen, jotta käyttäjä osaa käyttää sitä. Monesti käyttäjälle jää lopulliseksi käyttöliittymäksi WWW-selaimen sulkeminen, mikä sekään ei aina ole paras vaihtoehto.

Luku 2

Tunnistautuminen Internetissä Shibbolethilla

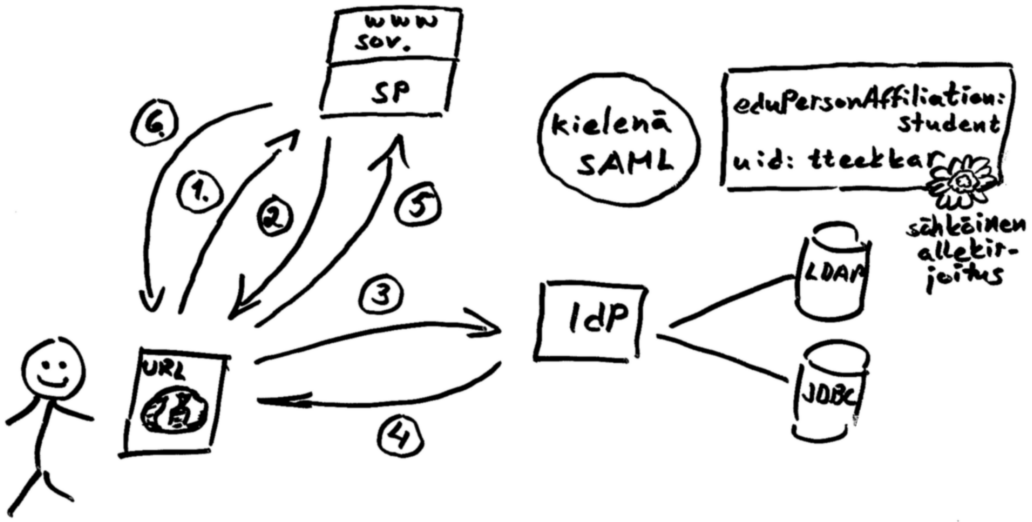
Polly want a cookie?

Shibboleth mahdollistaa käyttäjätietojen siirtämisen organisaatiorajojen ylitse silloin, kun käyttäjä haluaa käyttää jotain WWW-palvelua, joka vaatii käyttäjän tunnistamista. Organisaatioiden on tarvinnut etukäteen vain sopia yhteistyöstä ja vaihtaa teknisiä metatietoja.

Shibboleth jakautuu kahteen osaan, jotka ovat tunnistustietojen tarjoaja (kotiorganisaatio, Identity Provider, IdP) ja palveluntarjoaja (Service Provider, SP). Yksi SP voi tarvittaessa palvella useita WWW-palveluita tai -sovelluksia. IdP ja SP vaihtavat tunnistustietoja Security Assertion Markup Language -kielen (SAML) avulla. Pääasiassa SAML-viestit kulkevat käyttäjän WWW-selaimen välityksellä, mutta joissakin tilanteissa IdP ja SP voivat keskustella suoraan Simple Object Access Protocol -rajapinnan (SOAP) välityksellä.

Kuvassa 2.1 esitetyt sisäänkirjautumisen vaiheet ovat

1. Käyttäjä ottaa yhteyttä WWW-palveluun sp.example.org.
2. SP toteaa, ettei käyttäjää tunneta, joten se antaa vastauksena käyttäjän WWW-selaimelle SAML-tunnistuspyynnön (authentication request) ja pyytää selainta ottamaan yhteyttä IdP:hen.
3. Selain välittää SAML-tunnistuspyynnön IdP:lle, minkä jälkeen käyttäjä antaa käyttäjätunnuksensa ja salasanansa. IdP tarkastaa annetun tunnuksen ja salasanan oikeellisuuden.



Kuva 2.1: Yleiskuva Shibbolethista.

4. IdP antaa selaimelle allekirjoitetun SAML-viestin, jossa on käyttäjän tiedot (attribuutit). IdP pyytää selainta palaamaan SP:hen.

5. Selain välittää SAML-viestin SP:lle. SP tarkastaa IdP:n allekirjoituksen, purkaa viestistä attribuutit, tallentaa ne evästeeseen (cookie) sidottuun istuntoon ja antaa attribuutit lopuksi WWW-palvelulle. Palvelu on nyt käyttäjän käytettävissä istunnon vanhenemiseen asti.

6. Käyttäjä saa alkujaan pyytämänsä sivun WWW-palvelulta.

Seuraavaksi tutustumme tarkemmin Shibbolethiin ja siihen läheisesti liittyviin aiheisiin. Ensiksi käymme läpi, mitä luottamusverkostot ovat ja miksi ne ovat tärkeitä. Sitten syvennymme Shibbolethin käyttämään SAML-kieleen ja sen profiileihin. Tämän jälkeen tarkastelemme IdP:tä palvelinohjelmiston ja todennusmenetelmien kannalta. Lisäksi käymme lyhyesti läpi WWW-sovelluksen tarvitsemia muutoksia silloin, kun käyttäjän tunnistaminen on ulkoistettu IdP:lle. Lopuksi tutkimme SAML:n avulla välitettäviä attribuutteja.

2.1 Luottamusverkostot

Luottamusverkosto tarjoaa tavan hallita joukkoa IdP:itä ja SP:itä. Luottamusverkosto määrittelee yhteiset pelisäännöt jäsenilleen, jolloin eri toimijat voivat luottaa toisiinsa.

Luottamusverkoston operaattori ylläpitää verkoston yhteistä SAML-metatietoa, joka kuvaa verkoston jokaisen IdP:n sekä SP:n ja niiden ominaisuudet. IdP:t ja SP:t luottavat toisiinsa vain, mikäli niiden tiedot löytyvät metatiedosta. Operaattori myös ylläpitää tietoja attribuuttien luovutussäännöistä, jotka Shibbolethin tapauksessa ovat erillään muista metatiedoista.

Yleensä verkoston jäsenet tekevät sopimukset vain operaattorin kanssa, jolloin jokaisen toimijan ei tarvitse tehdä ristiin sopimuksia kaikkien muiden kanssa. Sopimuksessa jäsenet sitoutuvat noudattamaan verkoston sääntöjä.

Suomessa toimii korkeakoulujen yhteinen Haka-luottamusverkosto, jota operoi CSC – Tieteen tietotekniikan keskus Oy korkeakoulujen toimeksiantamana. Hakasta on kasvanut iso ja merkittävä luottamusverkosto viidessä vuodessa, myös kansainvälisesti mitattuna. Jäsenorganisaatioita Hakassa on 47, kumppaneita 14 ja SP:itä 99. Loppukäyttäjiä Hakassa on jo noin 280 000. Suomessa on tämän lisäksi pilottivaiheessa virkamiehille ja valtionhallinnon WWW-palveluille suunnattu Virtu-luottamusverkosto.

2.2 SAML

Security Assertion Markup Language (SAML) [22] on XML-pohjainen kieli, jolla voidaan esittää ja siirtää määrämuotoisesti tietoturvatietoja, kuten todennus-, valtuutus- ja henkilötietoja. SAML:n kehittämisestä vastaa Organization for the Advancement of Structured Information Standards (OASIS) [21], joka tunnettiin aikaisemmin nimellä SGML Open.

Esimerkiksi SAML-väite (SAML assertion) koostuu oleellisesti seuraavista osista:

- Myöntäjä R (issuer)
- Toimija S (subject)
- Edellytykset C, jolloin väite on tosi (conditions).
- Aika T, jolloin väite on annettu (time).

SAML:stä on olemassa kolme standardoitua versiota, jotka ovat 1.0, 1.1 ja 2.0. Kaksi ensimmäistä ovat lähellä toisiaan ja edelleen joissakin palveluissa sekä järjestelmissä käytössä. Tässä työssä keskitymme pääasiassa nykyiseen SAML 2.0:aan (SAML2), joka eroaa merkittävästi aikaisemmista versioista.

SAML2 määrittelee uutena useita XML-viestejä ja menetelmiä (protocols) viestien välittämiseen. Se yhdistää SAML 1.1:n, Liberty Alliancen [17] Identity Federation

Framework 1.2:n (ID-FF) [4] ja Shibboleth 1.3:n yhdeksi standardiksi. Tarkoituksena oli esimerkiksi korjata vanhassa SAML 1.1:ssä ollut heikkous, ettei se tarjonnut kunnollisia ja standardoituja menetelmiä SAML-viestien välittämiseen palvelusta toiseen.

SAML2:sta on kasvanut jo niin iso ja laaja standardi, että harva ohjelmisto tai palvelu toteuttaa, tai edes pystyisi toteuttamaan, sen kokonaan. Lisäksi SAML2:n määrittelyt ovat joiltakin osilta hyvin väljät ja yleisluontoiset. Taatakseen toteutusten yhteentoimivuuden SAML2 on jaettu profiileihin, jotka määrittelevät, miten standardin eri osat rakentuvat ja toimivat yhdessä. Monet SAML2:a käyttävät luottamusverkostot ovat myös määritelleet omia profiileita täsmentämään vaatimuksia, yksinkertaistamaan ratkaisuita ja parantamaan eri valmistajien tuotteiden sekä palveluiden yhteentoimivuutta.

Esimerkiksi Haka-luottamusverkosto on määritellyt oman Haka SAML2-profiilin [19], joka taas pohjautuu Interoperable SAML 2.0 Web Browser SSO Deployment Profile -määrittelyyn.

2.3 Shibbolethin rakenne

Kuten luvun alussa kävimme läpi Shibboleth-ohjelmisto jakautuu kahteen osaan, jotka ovat IdP ja SP. Nyt tarkastelemme lyhyesti, mitä IdP tarvitsee ympärilleen toimiakseen ja miten käyttäjän todentaminen toimii IdP:ssä. Sen jälkeen käymme läpi, miten SP ja WWW-sovellus liittyvät toisiinsa.

2.3.1 IdP ja Tomcat

Shibboleth IdP on Java-pohjainen palvelinohjelmisto, joka asennetaan Tomcatin [2] tai jonkun muun Java Servlet -alustan [24] päälle. Valitettavasti täysin avoimessa Javassa (IcedTea, OpenJDK) ei ole kaikkia IdP:n tarvitsemia toiminnallisuuksia, joten useimpiin Linux-jakeluihin pitää asentaa sen oman Javan rinnalle Oraclen Java Development Kit ja vaihtaa Tomcat käyttämään sitä.

Monet organisaatiot pystyttävät Tomcatin lisäksi Apache WWW-palvelinohjelmiston [1], joka asetetaan välittämään verkosta tulevat yhteydenotot Tomcatille. Syyt Apachen asennukselle vaihtelevat. Esimerkiksi käyttäjän varsinainen todennustapahtuma saattaa olla ulkoistettu toiselle ohjelmistolle, kuten PubCookielle [31], tai SOAP-rajapintojen asiakasvarmenteiden tarkastus on haluttu ulkoistaa Apachelle.

2.3.2 Todennusmenetelmät

Shibboleth IdP tukee valmiiksi kolmea todennusmenetelmää, jotka ovat salasanakysely (UsernamePassword), etäkäyttäjä (RemoteUser) ja aikaisempi yhteys (PreviousSession). Ensimmäinen kyselee käyttäjältä käyttäjätunnuksen ja salasanan ja tarkastaa ne Javan JAAS-rajapinnan (Java Authentication and Authorization Service) avulla. Toinen mahdollistaa verkon ja Tomcatin väliin asennetun WWW-palvelinohjelmiston, kuten Apachen, käyttämisen todennusmenetelmänä, jolloin Apache välittää onnistuneen tunnistuksen jälkeen käyttäjätunnuksen Tomcatille.

Aikaisempi yhteys on erikoistodennusmenetelmä, joka mahdollistaa kertakirjautumisen (Single Sign-On, SSO). Tällöin Shibboleth IdP luottaa aikaisemmin tapahtuneen, onnistuneen kirjautumisen yhteydessä tallennettuun WWW-selaimen evästeeseen. Toisin sanoen käyttäjä on jo aikaisemmin tunnistettu salasanan tai muun tunnistusmenetelmän avulla. WWW-palvelu voi halutessaan kieltää SSO-istunnon käyttämisen pyytämällä IdP:ltä pakotetun kirjautumisen (ForceAuthn).

2.3.3 WWW-sovellus ja SP

Shibboleth soveltuu parhaiten WWW-pohjaisten sovellusten käyttäjätunnistukseen, sillä WWW-selainta käytetään SAML-viestien välittämiseen. Lisäksi käyttäjätunnus sekä salasana annetaan normaalisti selaimen välityksellä IdP:lle. Tämä on sääli, sillä Shibbolethin kaltaiselle ratkaisulle olisi tilausta myös laajemmalti, esimerkiksi tietokoneen sisäänkirjautumisessa ja perinteisten sovellusten kanssa.

Tavallisesti WWW-sovellusta ei kytketä suoraan SP:hen, vaan SP kytketään WWW-palvelinohjelmistoon, kuten Apacheen tai Microsoft IIS:ään, lisämoduulin tai suodattimen avulla. IdP:n myöntämä SAML-väite ja siitä puretut tiedot näkyvät sovellukselle Shibbolethilla suojatun yhteyden otsikkotiedoissa. Tämä on vakiotapa välittää tietoja WWW-palvelinohjelmistolta sovellukselle.

Monesti sovellusta joudutaan hieman muokkaamaan, jotta se saadaan toimimaan yhteen SP:n kanssa. Tosin on olemassa myös sovelluksia, joihin ulkoisten tunnistuslähteiden sovittaminen on lähes mahdotonta. Pelkän todentamisen (authentication) sovittaminen on kaikista helpointa. Shibbolethilla voi kuitenkin tehdä paljon enemmän. Esimerkiksi valtuutus (authorization) voidaan osittain tai kokonaan ulkoistaa IdP:lle, tai sovelluksen käyttäjätunnukset voidaan luoda lennossa ensimmäisen kirjautumisen yhteydessä (varustaminen, provisioning).

2.4 Tietolähteet, attribuutit ja skeemat

SAML:ssä käyttäjään liittyvästä tietoturvatiedosta käytetään nimitystä attribuutti, joka siis kuvaa käyttäjän jotain ominaisuutta. Esimerkiksi Teemu Teekkarin attribuutit voisivat olla

```
eduPersonPrincipalName: tteekkar@hut.fi
uid: tteekkar
cn: Teemu Teekkari
displayName: Teemu
givenName: Teemu Toivo
sn: Teekkari
mail: teemu.teekkari@tkk.fi
preferredLanguage: fi
o: Helsinki University of Technology
eduPersonAffiliation: student
eduPersonAffiliation: member
funetEduPersonStudentID: 12345S
funetEduPersonStudentStatus: present
schacGender: 1
schacHomeOrganizationType:
  urn:mace:terena.org:schac:homeOrganizationType:fi:university
schacHomeOrganization: hut.fi
```

IdP tukee valmiina useita menetelmiä attribuuttien muodostamiseksi, ja IdP:n ylläpitäjä voi kirjoittaa niitä lisää sekä Javalla että JavaScriptillä. IdP voi hakea tietoja LDAP-hakemistopalvelusta [33] ja JDBC:tä [23] tukevasta tietokannasta. Lisäksi IdP voi yhdistellä tai muokata lähteistä tulevia tietoja mallinteen tai skriptauksen avulla. Tietolähteiden konfiguraatiossa voi myös määritellä staattisia arvoja attribuuteille, kuten schacHomeOrganization, jonka arvo on yleensä sama kaikille yhden organisaation käyttäjille.

SAML ei määrittele attribuuttien nimiä tai merkityksiä, vaan nämä tulevat skeemoista (schema). Monet skeemat ovat alkuperältään kirjoitettu LDAP-hakemistopalvelua varten, mutta ne käyvät hyvin Shibbolethin ja luottamusverkostojen tarpeisiin. Skeemat ovat tärkeitä yhteentoimivuuden ja luottamuksen takaamiseksi. IdP:iden ja SP:iden on puhuttava samaa kieltä: kaikki IdP:t laittavat displayName-attribuuttiin käyttäjän kutsumaetunimen, eikä esimerkiksi käyttäjän koko nimeä, ja SP:t voivat luottaa tähän yhdessä sovittuun merkitykseen.

Esimerkiksi Hakassa käytetään funetEduPerson 2.1 -skeemaa [7], joka rakentuu RFC 4519:n [27], amerikkalaisen eduPerson-skeeman [14] ja eurooppalaisen Schac-skeeman [29] päälle.

Luku 3

Uloskirjautumisen ongelmat

The end of childhood.

Jos sovellus tai palvelu vaatii sisäänkirjautumista, niin on luonnollista, että käyttäjän pitäisi voida myös kirjautua ulos siitä. Perinteisissä verkko- ja työpöytäsovelluksissa tämä ei ole kovinkaan iso ongelma, mutta World Wide Webissä (WWW) siitä muodostuu vaikea tekninen ongelma.

WWW rakentuu tilattoman HTTP-protokollan (HyperText Transfer Protocol) [8] päälle. Kuitenkin toimiakseen sovellukset tarvitsevat tilatietoja, jotka säilyvät WWW-selaimen eri aikoina ottamien HTTP-yhteyksien ylitse. Tähän ongelmaan on kaksi yleistä ratkaisua. Ensimmäinen on tallentaa tilatiedot selaimen käyttämien Uniform Resource Locator (URL) -osoitteiden loppuun. Toinen on taas käyttää pieniä evästeitä, joita WWW-palvelin voi tallentaa selaimen muistiin. Aina kun selain ottaa yhteyden palvelimeen, niin selain lähettää palvelimen aikaisemmin asettamat evästeet takaisin sille osana HTTP:n otsikkotietoja. Osa sovelluksista käyttää molempia menetelmiä tilanteen mukaan. Esimerkiksi jotkin tilatiedot kulkevat URL-osoitteissa ja loput evästeissä.

Evästeiden koko on käytännössä rajoitettu 4096 tavuun [16], joten usein sovellukset tallentavat evästeeseen vain istuntotunnisteen. Tunnistetta vasten sovellus etsii omasta muististaan tai muusta tallennusvälineestä istunnon, johon on tallennettu WWW-selaimen käyttäjään liittyvät tilatiedot. Joskus sovellus tallentaa tilatietoja suoraan evästeisiin, mutta tällöin ongelmaksi muodostuu se, että käyttäjä tai potentiaalinen hyökkääjä voi päästä muokkaamaan niitä vapaasti. Istunnon tunniste on myös syytä suojata hyvin, eli sen olisi hyvä olla kryptografisesti satunnainen ja se tulisi välittää vain SSL:llä (Secure Socket Layer) tai TLS:llä (Transport Layer Security) suojattujen HTTP-yhteyksien ylitse.

Uloskirjautuminen tarkoittaa käytännössä sisäänkirjautumisen yhteydessä luodun istunnon poistamista muistista tai kirjautumiseen liittyvien tilatietojen poistamista istunnosta. Jos istunto ei pääty käytön loppuessa, niin tällöin joku toinen henkilö voi mahdollisesti päästä käyttämään istuntoa. Toisin sanoen käyttäjä voi esiintyä toisena henkilönä kuin on.

Valitettavasti WWW-selaimet eivät auta luotettavan uloskirjautumisen toteuttamisessa, vaikka niillä olisi siihen hyvät mahdollisuudet. Oman haasteen uloskirjautumiseen tuo myös selkeän ja yksinkertaisen käyttöliittymän rakentaminen.

3.1 Uloskirjautumisen muodot

Shibbolethin ja SAML2:n maailmassa uloskirjautumisia on monta erilaista, mikä tekee uloskirjautumisen toteuttamisesta haasteellisen. Monesti sovellusta ja SP:tä käsitellään yhtenä kokonaisuutena, mutta esimerkiksi seuraavassa erottelussa niiden rajat tulevat selkeämmin esille.

Sovelluksen uloskirjautuminen. Mikäli sovellus pitää itse kirjaa istunnoista, niin silloin sen pitää myös toteuttaa niiden hallinta, kuten uloskirjautuminen. Ilman Shibbolethia tämä on ainoa tapa kirjautua sovelluksesta ulos. Sovelluksen tulisi rekisteröidä SP:lle käsittelijä, jolla voi käynnistää sovelluksen oman uloskirjautumisen. Käyttäjän tai Shibboleth SP:n pyytäessä uloskirjautumista, sovellus poistaa käyttäjään liittyvät istuntonsa.

SP:n uloskirjautuminen. Shibboleth SP pitää yllä istuntoa, jossa on tallessa esimerkiksi IdP:ltä saadut käyttäjän attribuutit. SP tarjoaa sovellukselle rajapinnan, jolla se voi käynnistää SP:n uloskirjautumisen. Tällöin SP poistaa istuntonsa ja mahdollisesti käynnistää lopuksi IdP:n uloskirjautumisen (SP-initiated logout).

Sovelluksen sijaan myös IdP voi aloittaa SP:n uloskirjautumisen. Samalla SP välittää uloskirjautumispyynnön sovellukselle, jos sovellus on rekisteröinyt sille käsittelijän. Näin myös sovelluksen istunto saadaan poistettua IdP:n käynnistäessä uloskirjautumisen.

Halutessaan osa sovelluksista voi ulkoistaa istuntonsa kokonaan SP:lle. Silloin niiden ei tarvitse itse huolehtia istuntojen hallinnasta.

IdP:n uloskirjautuminen. IdP pitää yllä SSO-istuntoa eli kertakirjautumisistuntoa. Sen ollessa voimassa käyttäjä pääsee ilman uudelleenkirjautumista IdP:hen kytkettyihin palveluihin. Kun käyttäjä ei enää halua pääsyä uusiin palveluihin, niin SSO-istunto voidaan poistaa. Pelkkä SSO-istunnon lopettaminen ei vaikuta

SP:iden voimassa oleviin istuntoihin mitenkään. Yleensä SSO-istunnolle on asetettu jokin aikaraja, jonka jälkeen se vanhenee. Esimerkiksi TKK:lla vanhenemisaika on kahdeksan tuntia.

Paikallinen uloskirjautuminen (Local Logout). Kun käyttäjä haluaa lopettaa vain yhden, edessään olevan, sovelluksen käyttämisen, niin silloin hän valitsee paikallisen uloskirjautumisen. Tämä on sama kuin edellä sovelluksen ja SP:n uloskirjautuminen yhdistettynä yhdeksi tapahtumaksi ilman IdP:n uloskirjautumisen käynnistämistä.

Kertauloskirjautuminen (Glogal Logout, Single Logout, SLO). Jos käyttäjä haluaa lopettaa yhdellä kertaa kaikkien avoinna olevien palveluidensa käyttämisen, esimerkiksi lähteäkseen tietokoneelta pois, niin silloin hän valitsee kertauloskirjautumisen. Tämä rakentuu sovelluksen, SP:n ja IdP:n uloskirjautumisten päälle. Näiden lisäksi IdP käy läpi jokaisen SP:n, jossa käyttäjä on vieraillut SSO-istunnon aikana, ja pyytää myös niitä tekemään SP:n ja sovelluksen uloskirjautumisen (IdP-initiated logout).

Osittainen uloskirjautuminen (Partial Logout). Kun sovellus ei toteuta uloskirjautumisprosessia oikein tai kertauloskirjautuminen epäonnistuu syystä tai toisesta, niin silloin tapahtuu osittainen uloskirjautuminen. Tällöin osa käyttäjän istunnoista jää aktiivisiksi ja käyttäjälle ei jää muuta vaihtoehtoa tilanteesta toipumiseen kuin sulkea WWW-selain.

3.2 Tietoturvaongelmat

Käyttäjä ei välttämättä aina hahmota tai muista, missä tilassa hänen istuntonsa ovat. Tietoturvaratkaisuissa tällainen käytettävyysongelma on aina myös tietoturvaongelma. Esimerkiksi käyttäjä saattaa vahingossa jättää istuntonsa auki. Tämä on erityisesti ongelmana yhteiskäyttöisten tietokoneiden kanssa, esimerkiksi kirjastoissa, nettikahviloissa ja kotona. Tällöin seuraava käyttäjä voi tahallaan tai vahingossa päästä esiintymään toisena henkilönä.

Istuntojen aukijääminen voi myös helpottaa sovellusta kohtaan tehtäviä hyökkäyksiä. Jos istuntoavaimet eivät ole tarpeeksi satunnaisia, niin hyökkääjä pystyy helpommin löytämään voimassa olevan istunnon, jota voi sitten väärinkäyttää. Hyökkääjän kannalta on vielä parempi, jos käyttäjä ei enää käytä istuntoa, mutta se on edelleen aktiivinen. Tällöin hyökkääjän paljastumisriski vähenee merkittävästi.

SSO-istunnon ollessa voimassa IdP avaa pyynnöstä uusia SP-istuntoja. IdP ei välttämättä aina pysty erottamaan, onko pyynnön tekijänä käyttäjä vai hyökkääjä. Toisin sanoen SSO-istunnon jäädessä auki hyökkääjä voi helposti käyttää kaikkia

IdP:hen kytkettyjä palveluita toisena henkilönä. SSO-istunnon aukijääminen on tietoturvan kannalta pahin tapahtuma.

Yhden SP:n istunnon jäädessä auki, ongelmat rajoittuvat vain yhteen tai muutamaan sovellukseen. Toisaalta jos käyttäjä on käyttänyt yhdellä kertaa suurta määrää SP:itä ja suurin osa niiden istunnoista jää roikkumaan, niin ongelmat voivat kertautua ja olla lähes yhtä pahoja kuin SSO-istunnon jääminen auki. Lisäksi SP:n istunnon ollessa auki SP tarvittaessa uusii sovelluksen oman istunnon, vaikka sovelluksen istunto ehtisikin välillä päättyä.

Sovelluksen istunnon roikkuessa tietoturvaongelmat ovat monesti aika rajoittuneita. Tosin esimerkiksi pankkisovelluksen jäädessä auki menetykset voivat olla merkittäviä. Vahinkojen rajaamiseksi pankkisovelluksissa voidaan laskuja maksettaessa käyttää lisävahvistusta, kuten uudelleen tunnistamista, toimenpiteen vahvistuskoodia tai kännykkävahvistusta. Lisävahvistus suojaa samalla myös muilta tietoturvaongelmilta, kuten välimieshyökkäykseltä.

Yleensä Shibboleth-ohjelmisto pyrkii suojaamaan omia istuntojansa hyökkääjää vastaan sitomalla istunnot ja SAML-viestit käyttäjän WWW-selaimen IP-osoitteeseen (Internet Protocol address). Käytännössä tämä voi aiheuttaa kuitenkin ongelmia esimerkiksi Internet-yhteyden toimittajan WWW-välitysryppään (WWW proxy cluster) kanssa, jolloin SP:n ylläpito saattaa kytkeä osan Shibbolethin tietoturvaominaisuuksista pois päältä.

Auki jääneet istunnot syövät lisäksi palvelun resursseja. Palvelussa, jota käytetään erityisen paljon, turhista istunnoista voi tulla jonkinlainen ongelma jo pelkällä normaalilla käytöllä. Hyökkääjä voi yrittää käyttää tätä hyväkseen hyökätäkseen sovellusta, SP:tä tai IdP:tä vastaan. Hyökkääjä pyrkii tällöin avaamaan paljon uusia istuntoja ilman, että niitä lopetetaan. Tämän tyyppistä hyökkäystä kutsutaan palvelunestohyökkäykseksi (denial-of-service attack).

Luku 4

Uloskirjautuminen käytännössä

All this has happened before, and all this will happen again.

Tässä luvussa tutustumme ensiksi siihen, miten uloskirjautuminen on määritelty Haka-luottamusverkostossa ja SAML2:ssa. Lopuksi käymme läpi tosielämän esimerkkejä uloskirjautumisesta.

4.1 Haka-luottamusverkoston uloskirjautuminen

Shibboleth 1.x ei tue uloskirjautumista, joten Hakaa varten piti suunnitella luottamusverkoston oma menetelmä uloskirjautumista varten [5]. Menetelmä ei ole läheskään täydellinen, mutta parempi kuin ei mitään. Seurauksena on yleensä osittainen uloskirjautuminen, eli sovellusten ja SP:iden istunnot jäävät roikkumaan muissa kuin uloskirjautumisen aloittavassa SP:ssä.

Hakassa uloskirjautumisen vaiheet ovat

1. Käyttäjä valitsee uloskirjautumisen WWW-sovelluksesta.
2. Sovellus ottaa talteen logout-url-attribuutin arvon.
3. Sovellus poistaa käyttäjän istunnon.
4. Sovellus ohjaa WWW-selaimen SP:n uloskirjautumisen käynnisteeseen (Initiator) /Logout ja antaa sille parametrin return, jossa on logout-url-attribuutista saatu arvo.

```
https://example.tkk.fi/Shibboleth.sso/Logout?
return=https://idp.aalto.fi/idp/aalto_logout.jsp
```

5. SP poistaa käyttäjän SP-istunnon.
6. SP ohjaa WWW-selaimen eteenpäin IdP:lle käyttäen return-parametrista saatua osoitetta.
7. IdP poistaa käyttäjän SSO-istunnon (PreviousSession).
8. Tarvittaessa IdP pyytää ulkopuolisen todennussovelluksen, kuten PubCookien, poistamaan istuntonsa.
9. WWW-selain palautetaan return-parametrin osoittamaan osoitteeseen, mikäli SP antoi IdP:lle parametrin.
10. Käyttäjälle kerrotaan, että hänet on kirjattu palvelusta ja SSO:sta ulos.
11. Käyttäjää varoitetaan siitä, että hänen olisi syytä sulkea selaimensa varmistaakseen uloskirjautumisensa kaikista palveluista. Tämän on tarkoitus estää istuntojen käyttäminen muista SP:istä kuin mistä uloskirjautuminen alkoi.

Vaikka käyttäjä sulkisi WWW-selaimensa, niin muiden SP:iden ja sovellusten istunnot jäävät auki aikakatkaisuun asti. Jos hyökkääjä saa tavalla tai toisella tietoonsa istunnon tunnisteen, niin hän voi hyödyntää näitä istuntoja jonkin aikaa.

4.2 SAML2:n uloskirjautuminen

SAML2:n uloskirjautuminen pohjautuu pitkälti Liberty Alliancen Identity Federation Frameworkiin. Valitettavasti uloskirjautuminen on kuvattu SAML2:ssa vain pintapuolisesti ja eri profiilit täydentävät sitä ristiriitaisesti. Esimerkiksi vanhempi versio dokumentista Interoperable SAML 2.0 Web Browser SSO Deployment Profile (SAML2simple) [28] määrittelee tarkoituksella, että SP:n on poistettava istuntonsa ennen siirtymistä IdP:lle, kun taas osa Identity Federation Framework ja SAML2-materiaaleista viittaavat siihen, että SP poistaa istuntonsa vasta saadessaan IdP:ltä positiivisen LogoutResponse-viestin [25]. Ensimmäinen tapa vaikuttaa tietoturvan näkökulmasta kestävämmältä. Näin aloittavan SP:n istunto päättyy aina, vaikka IdP:n tai muiden SP:iden kanssa tulisi ongelmia uloskirjautumisessa.

Tässä uloskirjautumisen vaiheet on kuvattu etukanavasidonnan (front-channel binding) mukaisesti. Tämä tarkoittaa sitä, että kommunikointi eri toimijoiden välillä tapahtuu WWW-selaimen välityksellä eikä taustalla SOAP:n avulla (back-channel binding).

1. Käyttäjä valitsee uloskirjautumisen WWW-sovelluksesta.

2. Sovellus poistaa käyttäjän istunnon.
3. Sovellus ohjaa WWW-selaimen SP:n uloskirjautumisen käynnisteeseen /Logout. Sovellus voi asettaa return-parametrin, jos se haluaa ohjata selaimen annettuun osoitteeseen onnistuneen uloskirjautumisen päätteeksi.
4. SP poistaa käyttäjän SP-istunnon.
5. SP etsii metadatasta IdP:n uloskirjautumispalvelun osoitteen (SingleLogoutService) [3] ja lähettää WWW-selaimen välityksellä siihen SAML2 LogoutRequest -viestin.
6. IdP poistaa käyttäjän SSO-istunnon (PreviousSession).
7. Tarvittaessa IdP pyytää ulkopuolista todennussovellusta poistamaan oman istuntonsa.
8. IdP ottaa yhteyttää jokaiseen SSO-istunnon aikana vierailtuun SP:hen, lukuunottamatta SP:tä, josta uloskirjautuminen alkoi.
   (a) IdP lähettää WWW-selaimen välityksellä SP:lle SAML2 LogoutRequest -viestin.
   (b) SP poistaa käyttäjän istunnon.
   (c) SP pyytää sovellusta poistamaan käyttäjän istunnon.
   (d) Lopuksi SP lähettää WWW-selaimen välityksellä IdP:lle SAML2 LogoutResponse -viestin uloskirjautumisen onnistumisesta tai epäonnistumisesta.
9. IdP lähettää WWW-selaimen välityksellä aloittaneelle SP:lle SAML2 LogoutResponse -viestin uloskirjautumisprosessin tuloksesta.
10. Vaihtoehtoisesti:
    (a) SP esittää käyttäjälle uloskirjautumisen tuloksen.
    (b) Jos uloskirjautuminen onnistui ja sovellus antoi SP:lle return-parametrin, niin SP ohjaa WWW-selaimen parametrin osoitteeseen.

Shibboleth SP 2.x tukee SAML2:n kertauloskirjautumista, mutta valitettavasti nykyinen Shibboleth IdP 2.x ei sitä tue. Tulevaan Shibboleth IdP 3.0:aan on suunnitteilla back-channel binding -pohjainen kertauloskirjautuminen.

Unkarilaiset ovat kehittäneet Shibboleth IdP 2.x:lle SLO-lisäkomponentin, joka toteuttaa toivotun kertauloskirjautumisen [20]. Shibbolethin pääkehittäjät Chad La

Joie ja Scott Cantor, joista jälkimmäinen on ollut keskeinen henkilö myös SAML:n määrittelyssä, suhtautuvat uloskirjautumisen ongelmakenttään huomattavan konservatiivisesti. Näin ollen he ovat suositelleet erityistä varovaisuutta SLO-lisäkomponentin käyttämisessä [15].

Koska yhtenäistä tapaa ei tällä hetkellä ole, niin käytännössä uloskirjautumisen toteuttamisessa jää ainoaksi vaihtoehdoksi luottamusverkostojen omat menetelmät ja määritykset. Joissakin luottamusverkostoissa tämä voi tarkoittaa Hakan tyylistä ratkaisua tai SLO-lisäkomponentin käyttämistä.

4.3 Käytännön esimerkkejä TKK:sta

TKK:lla Shibboleth IdP -tunnistuspalvelua kutsutaan nimellä WebLogin. Palvelulle on haluttu luoda helposti muistettava ja tunnistettava nimi, eräänlainen tuotemerkki. Seuraavaksi käymme esimerkinomaisesti läpi joitakin TKK:lla käytössä olevia WWW-palveluita, jotka hyödyntävät Shibbolethia. Tulemme huomaamaan, että näiden uloskirjautumisen toteutuksissa on merkittäviä eroja.

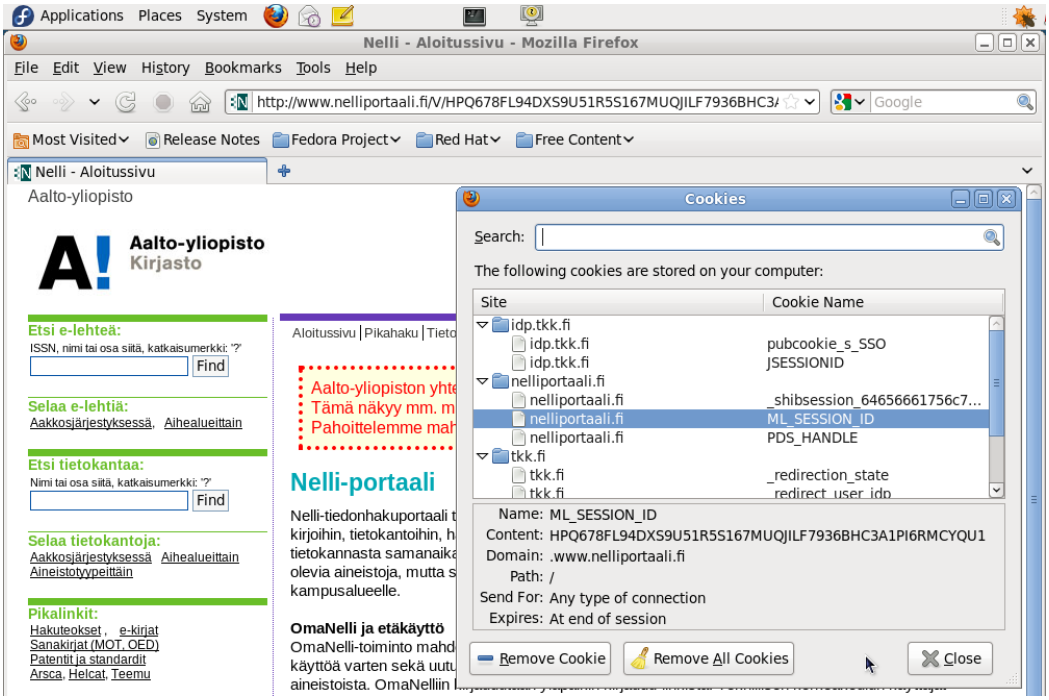
4.3.1 Nelliportaali

Nelliportaalissa uloskirjautuminen ohjaa IdP:lle, joka kertoo, että istunnot ovat päättyneet (kertauloskirjautuminen). Jos käyttäjä tämän jälkeen välittömästi palaa Nelliportaaliin ja valitsee sisäänkirjautumisen, niin käyttäjä on taas kirjautuneena palveluun kuin mitään ei olisi tapahtunut välissä. Seuraavaksi tarkastelemme, mitä oikeasti tapahtui.

SP:n istunto poistetaan oikein uloskirjautumisen yhteydessä, mutta Nelliportaalin sovellus ei siivoa pois PDS_HANDLE-nimistä evästettään. Testien perusteella vaikuttaisi siltä, että Nelliportaali tallentaa istuntotietoja kolmeen evästeeseen ja yhteen URL:iin (kuva 4.1):

1. Shibboleth SP:n _shibsession-eväste
2. Sovelluksen PDS_HANDLE-eväste
3. Sovelluksen ML_SESSION_ID-eväste
4. Sovelluksen URL:ssa polkuosa (path)

URL:n lopussa oleva tunniste näyttää olevan tärkein, sillä palvelu jatkaa toimintaansa täysin normaalisti, vaikka kaikki palvelun asettamat evästeet poistaa.



Kuva 4.1: Nelliportaalin evästeet ja URL

ML_SESSION_ID-eväste luodaan aina uudelleen palveluun mentäessä, jos sitä ei ole. Sen arvo on sama kuin URL:n polkuosa kauttaviivan ja viivaan välissä:

```
http://www.nelliportaali.fi/V/S4LM42QA4BUHUEJ4UTY
FHYCY1GLVYC8YALVPT1Q97UYXJQSF3U-14642?func=file
&file_name=home
```

Kun käyttäjä valitsee sisäänkirjautumisen, niin sovellus tarkastaa ensiksi PDS_HANDLE-evästeen. Jos eväste on olemassa, niin sen viittaaman istunnon käyttäjätiedot kopioidaan pääistuntoon. Muuten sovellus pyytää apua SP:ltä. SP:n saatua käyttäjätiedot sovellus kopioi SP:n istunnosta käyttäjän tiedot PDS_HANDLE-evästeen istuntoon ja sieltä sitten pääistuntoonsa.

Palvelun uloskirjautumiseen liittyvät isoimmat tietoturvaongelmat ovat

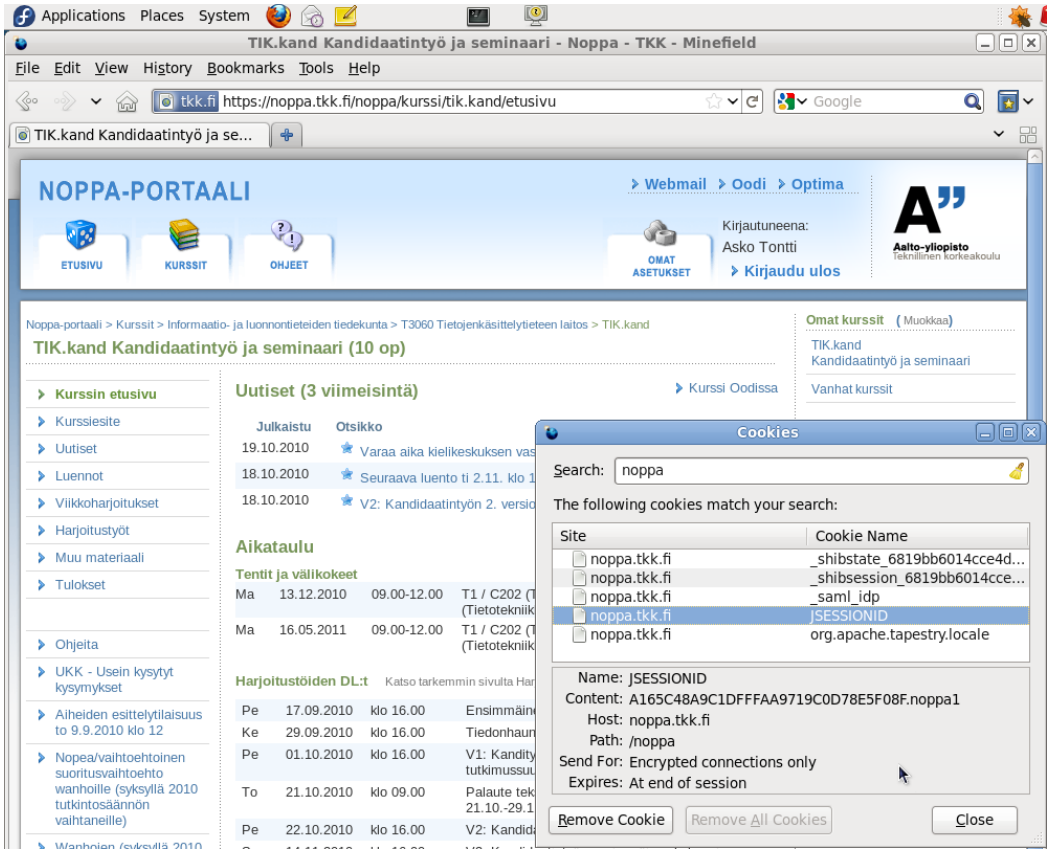
- Sovelluksen istunto jää auki, eli sovelluksen uloskirjautuminen ei toimi ollenkaan.
- Käyttäjä ei voi valita uloskirjautumisvaihtoehtoa, eikä käyttöliittymä kerro, mistä uloskirjautuminen tapahtuu. Toteutettu vaihtoehto on

kertauloskirjautuminen, mikä on tietoturvamielessä parempi vaihtoehto kuin paikallinen uloskirjautuminen.

- Muiden SP:iden ja sovellusten istunnot jäävät roikkumaan aikakatkaisuun asti, vaikka niitä ei pysty suoraan käyttämään WWW-selaimen sulkemisen jälkeen.

Käyttäjän näkökulmasta samalla tavalla käyttäytyviä WWW-palveluita on Halli.

4.3.2 Noppa



Kuva 4.2: Noppa-palvelu ja evästeet

Noppa tarjoaa Nelliportaalin tavoin vain kertauloskirjautumisen, ja käyttöliittymä ei tuo esille, että kyseessä on kertauloskirjautuminen.

Nopan käyttöliittymän kieli vuotaa uloskirjautumisen läpi. Tämä johtuu siitä, että kieli tallennetaan org.apache.tapestry.locale-nimiseen evästeeseen ja sitä ei poisteta

uloskirjautumisen yhteydessä. Ei ole itsestään selvää, onko tämä tarkoitus vai vahinko. Tietoturvan näkökulmasta tällä ei juurikaan ole merkitystä. Yhteiskäyttöisellä tietokoneella seuraava käyttäjä voi mahdollisesti saada tietoonsa, että Nopan edellinen käyttäjä oli suomen-, ruotsin- tai englanninkielinen.

Sisäänkirjautumisprosessin yhteydessä kaikki käyttäjätiedot kopioidaan joko JSESSIONID-evästeen istuntoon tai org.apache.tapestry.locale-evästeeseen (kuva 4.2). Muut palvelun evästeet voi poistaa sisäänkirjautumisen jälkeen, ja palvelu jatkaa toimintaansa normaalisti.

Palvelun uloskirjautumiseen liittyvät isoimmat tietoturvaongelmat ovat

- Käyttäjä ei voi valita uloskirjautumisvaihtoehtoa, eikä käyttöliittymä kerro, mistä uloskirjautuminen tapahtuu. Toteutettu vaihtoehto on kertauloskirjautuminen, mikä on tietoturvamielessä parempi vaihtoehto kuin paikallinen uloskirjautuminen.
- Kielivalinta vuotaa uloskirjautumisen läpi.
- Muiden SP:iden ja sovellusten istunnot jäävät roikkumaan aikakatkaisuun asti, vaikka niitä ei pysty suoraan käyttämään WWW-selaimen sulkemisen jälkeen.

Käyttäjän näkökulmasta samalla tavalla käyttäytyviä WWW-palveluita ovat eAge, Tilavaraus ja Rubyric.

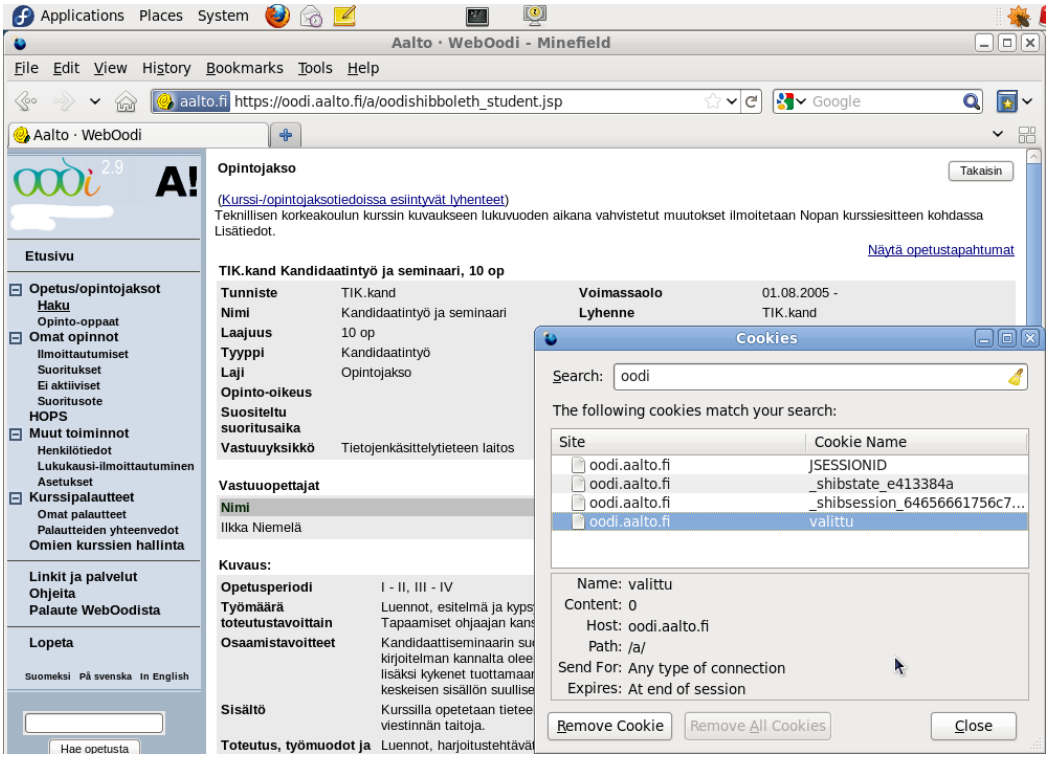
4.3.3 Oodi

Oodissa Lopeta-linkki poistaa vain sovelluksen paikallisen istunnon. Oodin SP-istunto, IdP:n SSO-istunto ja muiden SP:iden istunnot jäävät voimaan. Sovellus ei mitenkään kerro käyttäjälle, että istunnot ovat edelleen voimassa. Koska SP:n istunto jää aktiiviseksi, niin Oodiin pääsee heti takaisin valitsemalla sovelluksen käyttöliittymästä linkin *kirjaudu sisään*.

Oodi kopioi sisäänkirjautumisen jälkeen SP:n istunnosta kaikki käyttäjätiedot JSESSIONID-evästeen istuntoon. Käyttöliittymässä auki oleva näkymä on tallennettu valittu-nimiseen evästeeseen (kuva 4.3).

Palvelun uloskirjautumiseen liittyvät isoimmat tietoturvaongelmat ovat

- Käyttäjä ei voi valita uloskirjautumisvaihtoehtoa, eikä käyttöliittymä kerro, mistä uloskirjautuminen tapahtuu.
- Uloskirjautumisnappula tekee vain sovelluksen uloskirjautumisen. SP:n istunto jää aikakatkaisuun asti auki.



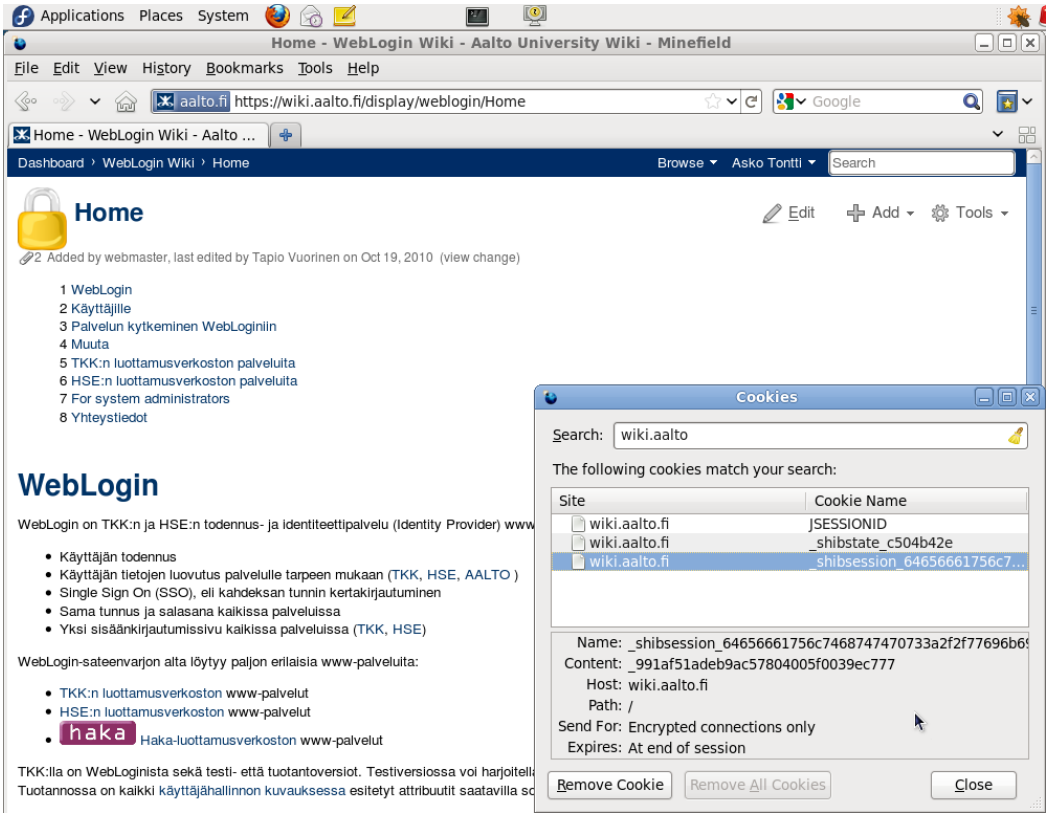
Kuva 4.3: Oodi ja valittu-eväste

- IdP:n SSO-istunto jää aikakatkaisuun asti auki.
- Muiden SP:iden ja sovellusten istunnot jäävät roikkumaan aikakatkaisuun asti.
- Jollei käyttäjä huomaa sulkea WWW-selainta, niin aukijääneet istunnot ovat esimerkiksi selaimen seuraavan käyttäjän käytettävissä.

Käyttäjän näkökulmasta samalla tavalla käyttäytyviä WWW-palveluita ovat Optima, Teemu, PersonecHR ja Goblin.

4.3.4 Wiki

Wikissä Log out -nappula poistaa oikein sovelluksen ja SP:n istunnon, mutta se ei tee kertauloskirjautumista IdP:lle. Näin IdP:n SSO-istunto ja muiden SP:iden istunnot jäävät voimaan. Käyttöliittymä ei tuo esille, että kyseessä on vain paikallinen uloskirjautuminen.



Kuva 4.4: Aalto-yliopiston Wiki-palvelu

Sen sijaan, että sovellus pyytäisi SP:tä siirtymään IdP:lle uloskirjautumista varten, sovellus pyytää palautusta omalle logout-sivulleen (return-parametri). Palvelu siis toteuttaa SAML2:n uloskirjautumisen eikä Hakan uloskirjautumista. Koska IdP ei osaa SAML2:n uloskirjautumista, niin SP päätyy poistamaan vain oman istuntonsa.

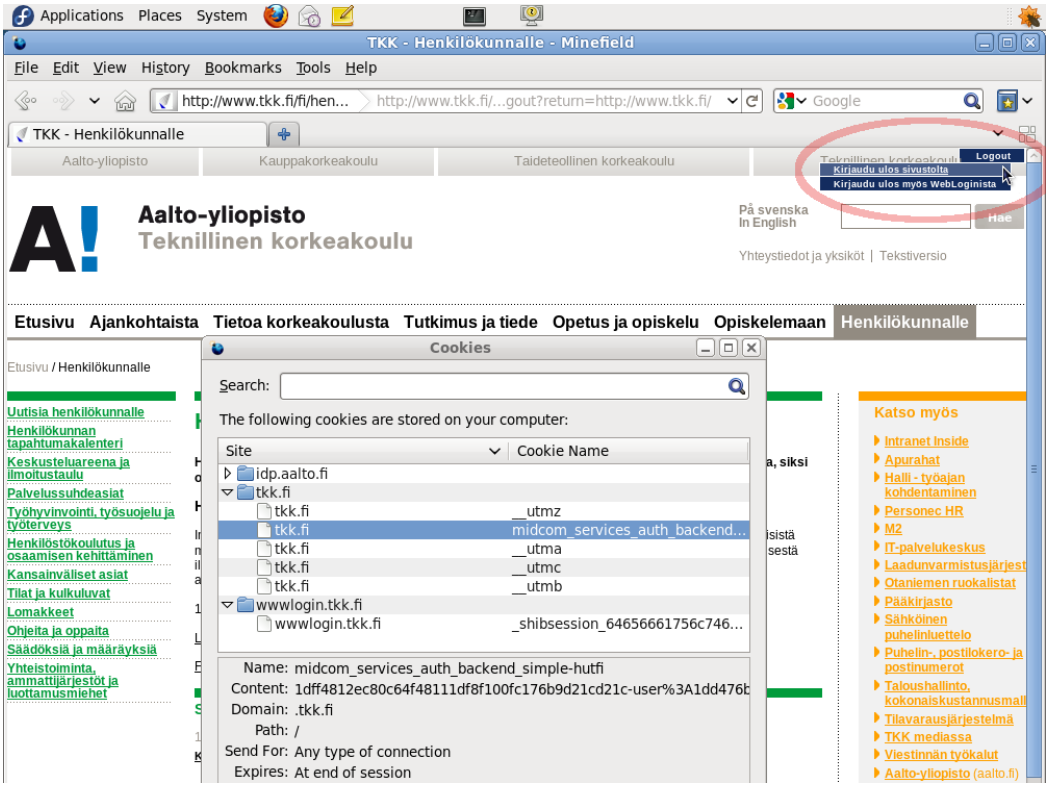
Palvelun uloskirjautumiseen liittyvät isoimmat tietoturvaongelmat ovat

- Käyttäjä ei voi valita uloskirjautumisvaihtoehtoa, eikä käyttöliittymä kerro, mistä uloskirjautuminen tapahtuu.
- Käyttäjälle ei kerrota, että kertauloskirjautuminen epäonnistui yhteensopivuusongelmien vuoksi.
- IdP:n SSO-istunto jää aikakatkaisuun asti auki.
- Muiden SP:iden ja sovellusten istunnot jäävät roikkumaan aikakatkaisuun asti.

- Jollei käyttäjä huomaa sulkea WWW-selainta, niin aukijääneet istunnot ovat esimerkiksi selaimen seuraavan käyttäjän käytettävissä.

Käyttäjän näkökulmasta samalla tavalla käyttäytyviä WWW-palveluita on Inside.

4.3.5 TKK:n henkilökuntasivut



Kuva 4.5: Uloskirjautumisnappulan päälle siirtyminen avaa valikon

TKK:n pääsivustolla on henkilökunnalle varattu osio, joka on suojattu Shibbolethilla. Palvelu poikkeaa edukseen siinä, että se tarjoaa käyttäjälle vaihtoehdon paikallisen uloskirjautumisen ja kertauloskirjautumisen välillä (kuva 4.5):

Kirjaudu ulos sivustolta. Käyttäjän valitessa tämän sovellus toteuttaa paikallisen uloskirjautumisen ja siirtää WWW-selaimen lopuksi sivuston aloitussivulle.

Kirjaudu ulos myös WebLoginista. Sovellus toteuttaa paikallisen uloskirjautumisen ja sen jälkeen käynnistää kertauloskirjautumisen SP:n avustuksella. Lopuksi IdP näyttää käyttäjälle viestin:

> *You have been logged out from the Single Sign On session.*
> *You might still have active sessions in the services that you have used during the Single Sing On session. To make sure you have logged out from all the services, please exit your browser.*

Koska TKK:n IdP toteuttaa Hakan uloskirjautumisen, niin WWW-selaimen sulkeminen on pakollista uloskirjautumisen varmistamiseksi. Näin estetään muiden SP:iden ja sovellusten avonaisten istuntojen käyttäminen.

Sivusto asettaa aina joukon evästeitä (__utma, __utmb, __utmc ja __utmz), joihin sisään- ja uloskirjautuminen eivät vaikuta. Kyseessä on Urchin Tracking Modulen eli nykyisen Google Analyticsin evästeet.

TKK:n pääsivustolla varsinainen palvelu ja SP on erotettu eri palvelimille. SP on asennettu koneelle wwwlogin.tkk.fi, kun taas muuten palvelu on nimen www.tkk.fi alla. Sisäänkirjauduttaessa wwwlogin.tkk.fi kopioi SP:n istunnosta palvelulle oleelliset henkilötiedot ja muodostaa niistä pitkän salatun merkkijonon. Tämän jälkeen wwwlogin.tkk.fi välittää merkkijonon www.tkk.fi:lle osana URL:a:

```
http://www.tkk.fi/fi/henkilokunnalle/?midcom_services_
auth_backend_simple-hutfi=1dfd50b44a31bf4d50b11dfb
475a564ea6022032203-user%3A1dd476b0494cac4476b
11dd8aab03b80a123ed13ed1
```

Palvelu www.tkk.fi erottaa URL:sta loppuosan (query string) ja muodostaa siitä samannimisen evästeen (midcom_services_auth_backend_simple-hutfi). Tätä evästettä ei uloskirjauduttaessa poisteta selaimesta, vaan istunnon tiedot poistetaan palvelimen muistista. Merkkijonon uudelleenkäyttö on estetty ainakin osittain (toistohyökkäys, replay attack). Muista palveluista poiketen uloskirjautuminen luo uuden PHPSESSID-evästeen. On mahdotonta sanoa varmasti, mihin PHP:n [30] istuntoa käytetään. Valistunut arvaus voisi olla, että PHP:n istuntoa käytetään Hakan logout-url-attribuutin välittämiseen uloskirjautumisprosessin seuraavalle vaiheelle.

Palvelu asettaa kaikki www.tkk.fi:lle tarkoitetut evästeet .tkk.fi-alueelle (domain), mikä on vaarallista käyttäjän ja palvelun tietoturvan kannalta. Käytännössä tämä tarkoittaa sitä, että kaikki suoraan tkk.fi:n alla olevat WWW-palvelut näkevät www.tkk.fi:lle tarkoitetut evästeet ja voivat muokata niitä. Erityisesti evästeen midcom_services_auth_backend_simple-hutfi vuotaminen muille palveluille on tietoturvan mielessä arveluttavaa.

TKK:n henkilökuntasivuilla uloskirjautuminen on toteutettu parhaiten. Se antaa käyttäjälle mahdollisuuden valita, mitä hän haluaa. Lisäksi sen toteutuksessa näyttää teknisesti olevan vähiten puutteita.

Palvelun uloskirjautumiseen liittyvät isoimmat tietoturvaongelmat ovat

- Ymmärtääkö käyttäjä uloskirjautumisvaihtoehtojen erot, ja osaako hän valita oikean vaihtoehdon.
- Kertauloskirjautumisessa muiden SP:iden ja sovellusten istunnot jäävät roikkumaan aikakatkaisuun asti, vaikka niitä ei pysty suoraan käyttämään WWW-selaimen sulkemisen jälkeen.

4.4 Yhteenveto palveluista

Näimme edellä, että uloskirjautumisen toteutuksissa on huomattavia eroja palveluiden kesken. Taulukkoon 4.1 on kerätty yhteenveto tutkituista palveluista ja lopputuloksista.

Taulukko 4.1: Yhteenveto uloskirjautumisen toteutuksista palveluittain

	Menetelmä	**Istunnot**				**Toimii**
		Sovellus	**SP**	**SSO**	**Muut SP:t**	
Nelliportaali	Haka SLO	○	✓	✓	† ○	ei
Noppa	Haka SLO	✓	✓	✓	† ○	ok[1]
Oodi	paikallinen	✓	○	○	○	ei
Wiki	SAML2 SLO	✓	✓	○	○	ei[2]
TKK:n henkilökuntasivut	paikallinen	✓	✓	○	○	ok[3]
	Haka SLO	✓	✓	✓	† ○	

○ istunto jää auki; ✓ istunto lopetettu; † ohje sulkea selain;
[1] kielivalinta vuotaa; [2] yhteensopivuusongelma;
[3] käyttäjä voi valita, mutta ymmärtääkö hän vaihtoehtojen eron?

On perusteltua kysyä, voiko puhua kertauloskirjautumisesta, jos muiden SP:iden istunnot jäävät auki. Tästä syystä IdP pyytää käyttäjää sulkemaan WWW-selaimen kertauloskirjautumisen päätteeksi. Selaimen sulkeminen poistaa samalla evästeet, jotka toimivat avaimina auki jääneisiin istuntoihin. Tällöin helpoin hyökkäysvektori estyy, kun selaimen seuraava käyttäjä ei pysty helposti hyödyntämään auki jääneitä istuntoja. Tietoturvamielessä ratkaisu ei siis ole täysin vedenpitävä, mutta monien palveluiden osalta jo riittävä suojaamaan niitä ja käyttäjää.

Toinen pohdintaa vaativa asia on, mikä on paikallisen uloskirjautumisen merkitys. Käyttäjälle luodaan palveluun uusi istunto, jos käyttäjä palaa paikallisen

uloskirjautumisen jälkeen palveluun ja käyttäjän SSO-istunto on edelleen voimassa. Käyttäjän näkökulmasta se näyttää samalta kuin hän ei olisi milloinkaan uloskirjautunut palvelusta. Paikallisen uloskirjautumisen merkitys nousee esille esimerkiksi tapauksissa:

- Käyttäjällä on palvelussa useita rooleja tai käyttäjätunnuksia, ja hänen pitää kirjautumisen yhteydessä valita niistä jokin, esimerkiksi Oodissa opiskelija- ja opettajaroolit.
- Palvelu vaatii lisätodennusta sisäänkirjautumisen yhteydessä. Tällöin paikallisella uloskirjautumisella on selkeä vaikutus.
- Käyttäjä haluaa käynnistää sovelluksen uudelleen, esimerkiksi palvelun toimintahäiriön vuoksi tai nollatakseen tilatiedot.
- Palvelussa on arkaluontoista tietoa, ja valveutunut käyttäjä haluaa minimoida tietoturvauhat päättämällä sovelluksen istunnon heti, kun palvelun käyttö päättyy.
- Käyttäjä haluaa lopettaa yhden palvelun käytön, mutta haluaa vielä jatkaa muiden palveluiden käyttämistä.

Voikin sanoa, että paikallisen uloskirjautumisen puutteen huomaa silloin kun sitä eniten tarvitsisi. Näitä tilanteita tulee aina välillä vastaan. Tavanomaisessa käytössä pelkkä kertauloskirjautuminen on täysin riittävä. Monet käyttäjät eivät edes vaivaudu tekemään sitä, vaan antavat istuntojen vanhentua aikakatkaisuun. Tätä tapahtuu erityisesti tilanteissa, joissa käyttäjällä on vain henkilökohtaisessa käytössä oleva tietokone. Tietoturvan ja palvelun resurssien kannalta tämä ei kuitenkaan ole paras vaihtoehto.

Luku 5

Ratkaisuja

Keep it simple, stupid!

Tässä luvussa käymme läpi erilaisia ratkaisuita uloskirjautumisen ongelmiin. Ensiksi tarkastelemme uloskirjautumista käyttäjän näkökulmasta. Sitten pohdimme lyhyesti, mitä muutoksia sovellus ja IdP tarvitsevat toimivan uloskirjautumisen toteuttamiseksi. Sen jälkeen tutustumme CoSign-ohjelmiston tapaan hallita istuntoja ja ratkaista kertauloskirjautuminen. Lopuksi katselemme uloskirjautumista WWW-selaimen näkökulmasta.

5.1 Käyttäjä ja käyttöliittymä

Turvaratkaisuiden tärkein sääntö on *yksinkertaisuus*. Erityisesti tämä korostuu käyttöliittymässä. Sen pitää olla helppo, selkeä ja yksinkertainen. Muuten vaarana on, että järjestelmän turvallisuus kompastuu jo heti ensimmäisellä askelmalla käyttäjään.

Käytännössä käyttäjät täytyy kouluttaa ymmärtämään paikallisen ja kertauloskirjautumisen ero, ja näille vaihtoehdoille pitää saada yhtenäinen ja helppo käyttöliittymä kaikkiin sovelluksiin. Olisi hyvä, että käyttöliittymä kertoisi yhdellä vilkaisulla, miten monessa palvelussa ollaan kirjautuneena, ja onko SSO-istunto voimassa, mutta tämän toteuttaminen ei ole helppoa, teknisesti tai käyttöliittymän näkökulmasta. Erityisesti kertauloskirjautumisessa on tärkeätä, että käyttäjä tietää, onnistuiko uloskirjautuminen loppuun asti oikein. Näin käyttäjä voi sulkea WWW-selaimen korjaavana toimenpiteenä, jos kaikki ei mennyt oikein.

Kaiken kaikkiaan WWW-selaimen sulkeminen on lopullinen käyttöliittymä uloskirjautumiseen, sillä se ratkaisee tyydyttävästi lähes kaikki

kertauloskirjautumisen ongelmat ja se on käyttäjälle tarpeeksi yksinkertainen sekä helppo. WWW-selaimen sulkemisessa on kuitenkin kaksi merkittävää huonoa puolta:

1. Istunnot jäävät roikkumaan aikakatkaisuun asti. Tosin niitä ei voi hyödyntää ilman, että saa jollakin tavalla selville istuntoavaimen.
2. Käyttäjä saattaa haluta tehdä uloskirjautumisen, vaikka hän ei halua samalla lopettaa WWW-selaimen käyttämistä. Esimerkiksi käyttäjä on kerännyt ison määrän selainikkunoita tai selaimen välilehtiä, joita hän ei halua sulkea.

5.2 Sovellusten muokkaus

Uloskirjautumisen integrointi WWW-sovellukseen vaatii työtä ja jää helposti puolitiehen. Ensimmäisenä sovelluskehittäjän täytyy päättää, käyttääkö sovellus pelkästään SP:n istuntoa vai ylläpitääkö sovellus istuntonsa itse.

Aiemmassa tapauksessa riittää, että käyttöliittymässä on nappulat, joilla voi käynnistää uloskirjautumisen ja kertauloskirjautumisen. Nappulat voi suoraan ohjata SP:n /Logout-käynnisteeseen sopivin parametrein. SP hoitaa kaiken muun. Jos sovellus on aikaisemmin käyttänyt HTTP:n perustodennusta (basic authentication), niin käyttöliittymässä ei todennäköisesti ole valmiina uloskirjautumisnappuloita.

Jälkimmäinen tapaus on pitkälti sama kuin aiempi, mutta uloskirjautumisnappuloiden pitää ensiksi poistaa sovelluksen oma istunto ennen /Logout-käynnisteen kutsumista. Lisäksi IdP:ltä tulevia uloskirjautumispyyntöjä (IdP-initiated logout) varten sovelluksen pitää rekisteröidä SP:lle Single Logout -käsittelijä. Kun SP saa IdP:ltä uloskirjautumispyynnön, niin SP kutsuu käsittelijää. Käsittelijän pitää poistaa sovelluksen kaikki istunnot, jotka liittyvät pyynnössä esitettyyn käyttäjään. Uloskirjautumispyynnöt voivat tulla SP:lle WWW-selaimen välityksellä tai SOAP-rajapinnan avulla. Luonnollisesti SOAP:n kanssa evästeiden poistaminen selaimesta ei ole mahdollista, joten sovelluksen pitää toteuttaa uloskirjautuminen poistamalla istunnon tiedot muististaan.

Lopuksi sovelluksen istunnon käyttöaikaa pitää muokata lyhyemmäksi kuin SP:n istunnon käyttöaika on, jotta ulkoapäin tulevat uloskirjautumispyynnöt toimivat oikein. Lisätietoja uloskirjautumiseen liittyvästä räätälöinnistä löytyy Shibboleth2:n wiki-sivuilta [10].

5.3 Kirjautumisen tilatiedot ja uloskirjautuminen

Kätevää Shibbolethissa on ollut se, että IdP:n ei ole tarvinnut pitää kirjaa käyttäjän auki olevista SP-istunnoista. On riittänyt, että IdP pitää kirjaa vain SSO-istunnon voimassaolosta ja SP:t pitävät kopioita käyttäjän attribuuteista välimuisteissaan. IdP:t ja SP:t voi vapaasti käynnistää uudelleen, sillä käytännössä kaikki oleellinen tilatieto on tallennettuna käyttäjän WWW-selaimen evästeisiin ja muut tiedot voi hakea aina uudelleen.

Tämä kuvio kuitenkin muuttuu kertauloskirjautumisen seurauksena. Jotta IdP voisi pyytää jokaista käyttäjän vierailemaa SP:tä tekemään paikallisen uloskirjautumisen, IdP:n pitää tallentaa käyttäjän SSO-istuntoon jokainen SP, jolle on myönnetty käyttäjää koskeva SAML-väite. Tämä muuttaa oleellisella tavalla IdP:n SSO-istunnon merkitystä ja tärkeyttä.

5.4 CoSign-ohjelmiston ratkaisu

CoSign-ohjelmisto [26] jakautuu kahteen osaan Shibbolethin tavoin: suodatin (filter) vastaa lähinnä SP:tä ja pääohjelma (daemon) vastaa IdP:tä. CoSignin ja Shibbolethin tavat toteuttaa uloskirjautumien eroavat merkittävästi toisistaan, ja CoSignin ratkaisua voi pitää mielenkiintoisena. Palveluun asennettu suodatin ottaa noin minuutin välein yhteyttä CoSignin pääohjelmaan ja kysyy, onko SSO-istunto edelleen voimassa.

Näin ollen kun pääohjelman SSO-istunto päättyy esimerkiksi kertauloskirjautumiseen tai aikakatkaisuun, niin istunnot päättyvät kaikissa palveluissa lyhyen ikkunan sisällä. Lisäksi tietoliikenneongelmat pääohjelman ja palvelun välillä eivät jätä istuntoja roikkumaan, kuten voi käydä SAML2:n uloskirjautumisessa. Tosin pääohjelman toimintaongelmat voivat samalla heijastua helpommin palveluihin ja käyttäjille. Kätevää tässä ratkaisussa on myös se, ettei pääohjelman tarvitse pitää tilatietoja palveluista, joihin käyttäjä on kirjautunut SSO-istunnon aikana. Kaikki palveluiden tilatiedot ovat vain palveluilla, eikä osittain pääohjelmalla.

CoSignin ratkaisua voi yrittää simuloida Shibbolethilla asettamalla SP:n ja sovelluksen istunnon pituudeksi vain pari minuuttia. Tosin tämä ratkaisu on erittäin raskas sovellukselle, SP:lle ja IdP:lle. Lisäksi HTTP:n POST-metodi, jota esimerkiksi WWW-lomakkeet käyttävät, todennäköisesti rikkoutuu jatkuviin WWW-selaimen uudelleenohjauksiin SP:n ja IdP:n välillä. Oikeampi tapa olisi toteuttaa IdP:hen SOAP-rajapinta, jolla voi kysyä käyttäjän SSO-istunnon tilaa. Tämän jälkeen SP:tä pitäisi muuttaa käyttämään rajapintaa istunnon voimassaolon lisävarmistukseen. Tosin tällöin Shibboleth ei enää olisi SAML2-yhteensopiva ohjelmisto.

5.5 WWW-selainten evästeiden hallinta

Evästeiden hallinnassa ei ole vuosien mittaan tapahtunut oleellisia muutoksia. WWW-selainten valmistajat ovat pääasiassa kiristäneet evästeiden asettamiseen ja lähettämiseen liittyviä sääntöjä erilaisten hyökkäysten torjumiseksi. Nämä säännöt estävät esimerkiksi sen, että SP ja IdP voisivat nähdä toistensa asettamia evästeitä.

Kertauloskirjautumisen kannalta olisi kuitenkin hyvä, jos sääntöjä voisi muuttaa. Olisi perusteltua, että IdP voisi luoda SSO-istuntoon liittyvän evästekokoelman, johon SP:t voisivat liittää omat istuntoevästeensä. Säännöissä olisi sitten sallittu, että IdP voisi evästekokoelman luojana käydä evästeitä läpi ja poistaa niitä. Lisäksi IdP:lle voisi sallia oikeudet asettaa evästeitä, jotka näkyvät kaikille evästekokoelman palvelimille.

Evästekokoelmat avaisi monia uusia mahdollisuuksia kehittää kertakirjautumista ja kertauloskirjautumista. Esimerkiksi IdP voisi kertauloskirjautumisen lopuksi poistaa SSO-istuntoon liittyvien SP:iden evästeet pakolla, jos ne ovat jostain syystä jääneet poistamatta.

5.6 AJAX-tekniikan uudet mahdollisuudet

Asynchronous JavaScript and XML (AJAX) [9] tarjoaa uloskirjautumiseen uusia, mielenkiintoisia mahdollisuuksia. Perinteisesti AJAX:a on käytetty interaktiivisten käyttöliittymien toteutukseen ja suorittamiseen WWW-selaimessa, mutta sillä voisi helposti toteuttaa esimerkiksi ratkaisun, jossa WWW-selain lähettää SP:lle ja IdP:lle *olen hengissä* -viestejä muutaman minuutin välein. Tästä olisi useampia hyötyjä.

Jos käyttäjä ei valitse WWW-sovelluksessa uloskirjautumista, vaan jatkaa suoraan toiselle sivustolle, niin selain lakkaa automaattisesti lähettämästä SP:lle viestejä. Tästä SP voi päätellä, että käyttäjä on lopettanut sovelluksen käyttämisen ja aloittaa käyttäjän puolesta paikallisen uloskirjautumisen. Näin SP:n ja sovelluksen istunnot eivät jää roikkumaan pitkäksi aikaa. Vastaavasti kun käyttäjä on sulkenut kaikki avoinna olevat WWW-sovellukset tai jatkanut niistä muille sivustoille, niin myös IdP lakkaa saamasta *olen hengissä* -viestejä. Tällöin IdP voi päättää käyttäjän SSO-istunnon ja vielä varmuuden vuoksi käynnistää SSO-istunnon SP:iden uloskirjautumisen. Näin IdP:n SSO-istunto ei jää roikkumaan ja avoimeksi hyökkäysyrityksille.

Haasteita tässä ratkaisussa aiheuttavat WWW-selaimet, jotka eivät tue JavaScriptiä, jolloin selain ei osaa lähettää viestejä SP:ille ja IdP:lle. Sama ongelma esiintyy, jos käyttäjä on kytkenyt JavaScriptin pois päältä. Lisäksi ratkaisu kuormittaa IdP:tä ja SP:itä ylimääräisillä viesteillä.

5.7 Testaus

Helposti unohtuva ratkaisu ongelmiin on testaus. Palvelut pitäisi testata kattavasti ennen niiden kytkemistä IdP:hen tai luottamusverkostoon. Testauksessa pitäisi sisäänkirjautumisen lisäksi painottaa uloskirjautumisen toimivuutta, käytettävyyttä ja varmuutta. Toimijoiden tulisi myös testata erilaisten uloskirjautumisratkaisuiden, kuten Hakan ja SAML2:n, yhteentoimivuutta ja määritellä tavat, joilla uloskirjautuminen saadaan selkeästi epäonnistumaan yhteensopivuusongelmien tullessa vastaan.

Ensiksi voisi lähteä liikkeelle siitä, että määrittelisi testattavat asiat, jotka palvelun omistajan tulisi itseauditoida ennen kytkentää. Teknisten yksityiskohtien lisäksi testattavissa asioissa voisi olla määritelty yhtenäiset käyttöliitymäkäytännöt uloskirjatumiselle, jolloin pahimmat käyttöliittymäongelmat saadaan poistettua ja käyttöliittymistä saadaan yhtenäisemmät. Testauksen tehostamiseksi jonkinlainen automaattinen yksikkötestausjärjestelmä (unit testing) voisi olla myös hyvä ratkaisu [32]. Sen kehittämisessä olisi paljon työtä, mutta se takaisi paremmat tulokset.

5.8 Johtopäätökset

WWW:ssä uloskirjautuminen on haastava ongelma, sekä teknisesti että käyttäjän näkökulmasta. Näitä ongelmia voi ratkaista monella eri tavalla. Luontevin vaihtoehto olisi, että OASIS jatkaisi SAML:n kehitystyötä ja korjaisi standardin seuraavassa versiossa uloskirjautumisen määritelmiä. Kuvaava on, että nykyisen SAML2:n uloskirjautuminen on hauras ja monimutkainen. Uusi, parempi näkökulma voisi olla esimerkiksi CoSign-ohjelmiston tapainen ratkaisu, jossa palvelut käyvät säännöllisin väliajoin kysymässä pääohjelmalta, onko SSO-istunto edelleen voimassa.

WWW-selaimien valmistajat voisivat myös auttaa tuomalla selaimiin uusia ominaisuuksia, jotka mahdollistaisivat luotettavan kertauloskirjautumisen toteuttamisen. Esimerkiksi evästeiden hallinnan ja käyttöliittymän puolella olisi hyviä mahdollisuuksia tähän. Selainvalmistajat tekevät aktiivisesti yhteistyötä HTML5:n [11] kehittämisessä ja määrittelyssä. Samassa yhteydessä olisi luonnollista yrittää ratkoa myös uloskirjautumisen ongelmia. Luottamusverkostoissa voitaisiin puolestaan panostaa uloskirjautumisen ja yhteentoimivuuden testaukseen.

Luku 6

Yhteenveto

All good things must come to an end...

Tunnistautumisesta on tullut yhä keskeisempi osa Internetin palveluita. Tästä on seurannut kasvava tarve kertakirjautumiselle ja uloskirjautumiselle. Ymmärrettävästi käyttäjä haluaa kirjautua useampaan palveluun yhdellä kertaa. Lisäksi tietoturvasyistä käyttäjän pitää pystyä lopettamaan palveluiden käyttö siististi.

Uloskirjautuminen Internetissä on haaste, sillä HTTP on tilaton ja evästeillä luotu illuusio tilallisuudesta ei mahdollista tilan purkamista luotettavasti. Haasteet eivät kuitenkaan lopu tähän. Käyttäjä ei välttämättä ymmärrä, mitä WWW-palveluiden eri uloskirjautumisvaihtoehdot tarkoittavat. Uloskirjautuminen voi myös epäonnistua, eikä ole itsestään selvää, että käyttäjä huomaa tai ymmärtää tämän. Syynä epäonnistumiseen voi olla se, että sovelluksen uloskirjautuminen on toteutettu virheellisesti tai tietoliikenteessä ja palvelimen toiminnassa on tilapäisiä ongelmia, jolloin uloskirjautumispyynnöt voivat viivästyä tai jopa kadota.

Internetissä tunnistusratkaisuiden yhteentoimivuus korostuu entisestään, sillä palveluita on paljon ja niiden toteutukset vaihtelevat merkittävästi. SAML2 on standardi tapa toteuttaa organisaatiorajat ylittävä käyttäjähallinto ja kertakirjautuminen Internetissä. Shibboleth-ohjelmisto on puolestaan merkittävä ja laajalle levinnyt SAML2-toteutus, erityisesti akateemisissa piireissä.

Ratkaisu uloskirjautumisen ongelmiin ei ole pelkästään tekninen, vaan käyttäjä on keskeinen osa yhtälöä. Voikin sanoa, että käyttäjän koulutusta ei voi ylenkatsoa tietoturvaratkaisuissa ja -asioissa. Käyttäjän täytyy esimerkiksi ymmärtää, mitä kertakirjautuminen, paikallinen uloskirjautuminen ja kertauloskirjautuminen tarkoittavat. Käyttäjän tukemiseksi käyttöliittymän on oltava tarpeeksi yksinkertainen ja helppo. Esimerkiksi WWW-selaimen sulkeminen on monesti varmin tapa

varmistaa uloskirjautuminen, vaikka se ei olekaan täydellinen ratkaisu tietoturvan näkökulmasta.

Tutkiessamme TKK:ssa käytössä olevia WWW-palveluita huomasimme, että yllättävän monessa palvelussa on teknisiä puutteita uloskirjautumisen toteutuksessa. Esimerkiksi sovelluksia ei ole muokattu oikein, joten uloskirjautuminen ei toimi kuten pitäisi. Yhdessä palvelussa tuli myös esille yhteensopivuusongelma Hakan ja SAML2:n uloskirjautumisen välillä. Hyvä ratkaisu näihin ongelmiin olisi testaus. Selvästi IdP:hen tai luottamusverkostoon kytkettäviä palveluita ei testata tarpeeksi ennen niiden käyttöönottoa. Lisäksi luottamusverkostojen ja palveluiden toimittajien pitäisi kiinnittää enemmän huomiota eri uloskirjautumismenetelmien yhteensovittamiseen.

CoSign-ohjelmisto toteuttaa istuntojen hallinnan ja uloskirjautumisen eri tavalla kuin SAML2 ja Shibboleth. CoSignissa suodatin tarkastaa muutaman minuutin välein, onko pääohjelman SSO-istunto edelleen voimassa. Näin SSO-istunnon päättyminen käyttäjän uloskirjautumiseen tai aikakatkaisuun leviää suhteellisen nopeasti palveluihin ja istunnot eivät jää roikkumaan. Kaunista tässä ratkaisussa on se, että pääohjelman ei tarvitse pitää kirjaa palveluista, joihin käyttäjä on kirjautuneena.

Halutessaan WWW-selainten valmistajat voisivat auttaa parempien sisään- ja uloskirjautumisratkaisuiden toteutuksessa. Esimerkiksi liikkeelle voisi lähteä käyttöliittymien kehittämisestä ja evästeiden käsittelystä. Evästekokoelmat voisivat lieventää evästeiden asettamiseen ja lähettämiseen liittyviä sääntöjä siten, että IdP ja SP:t voisivat helpommin hallita SSO-istunnon kaikkia evästeitä yhtenä kokonaisuutena.

Käyttöliittymäparannusten lisäksi uudehko AJAX-tekniikka mahdollistaa esimerkiksi *olen hengissä* -viestien lähettämisen selaimelta SP:ille ja IdP:lle. Viestivirran loppumisesta SP:t ja IdP tietävät, milloin käyttäjä on todennäköisesti lopettanut istunnon käytön, ja voivat käyttäjän puolesta suorittaa uloskirjautumiset.

Nykyinen SAML2:n uloskirjautuminen on valitettavan hauras ja monimutkainen. Paras ratkaisu SAML2:n ja Shibbolethin uloskirjautumisongelmiin olisi, että OASIS jatkaisi SAML:n kehitystyötä ja korjaisi tulevissa versioissa uloskirjautumisen määritelmiä.

Sisäänkirjautuminen on saatu Internetissä toimimaan jo suhteellisen hyvin, mutta uloskirjautuminen on vielä lapsenkengissä ja vaatii kovasti työtä. Tärkeätä olisi, että standardointiorganisaatiot, WWW-selainten valmistajat ja sovellusten kehittäjät ottaisivat itseänsä niskasta ja keskittyisivät näiden ongelmien ratkaisemiseen. Haasteista huolimatta pienelläkin työllä, kuten testauksella, voisi päästä pitkälle.

Kirjallisuutta

[1] Apache Software Foundation. Apache HTTP server project, lokakuu 2010. URL http://httpd.apache.org/. Apache Software Foundationin WWW-sivuilta. Viitattu 31.10.2010.

[2] Apache Software Foundation. Apache Tomcat, lokakuu 2010. URL http://tomcat.apache.org/. Apache Software Foundationin WWW-sivuilta. Viitattu 31.10.2010.

[3] Scott Cantor. NativeSPLogoutInitiator, elokuu 2010. URL https://spaces.internet2.edu/display/SHIB2/NativeSPLogoutInitiator. Internet2:n Shibboleth Wikistä. Viitattu 23.9.2010.

[4] Scott Cantor, Darryl Champagne, John Kemp, Eric Tiffany, Peter Thompson ja Thomas Wason. Liberty Alliance ID-FF 1.2 specifications, toukokuu 2005. URL http://projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications/. Liberty Alliancen vanhoilta WWW-sivuilta. Viitattu 8.10.2010.

[5] CSC - Tieteen tietotekniikan keskus. Haka uloskirjautuminen, 2010. URL http://www.csc.fi/hallinto/haka/ohjeet/ohjeet-yllapitajille/haka-logout. Haka-luottamusverkoston WWW-sivuilta. Viitattu 22.9.2010.

[6] CSC - Tieteen tietotekniikan keskus. Haka-käyttäjätunnistusjärjestelmä, 2010. URL http://www.csc.fi/hallinto/haka. Haka-luottamusverkoston WWW-sivut. Viitattu 8.10.2010.

[7] CSC et al. funetEduPerson schema, version 2.1, elokuu 2008. URL http://www.csc.fi/hallinto/haka/maaritykset/funeteduperson-skeema/fep_2.1.pdf. Haka-luottamusverkoston WWW-sivuilta. Viitattu 30.9.2010.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach ja T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft

Standard), kesäkuu 1999. URL http://www.ietf.org/rfc/rfc2616.txt. Updated by RFCs 2817, 5785.

[9] Jesse James Garrett. Ajax: A new approach to web applications, helmikuu 2005. URL http://www.adaptivepath.com/ideas/essays/archives/000385.php. Adaptive Pathin WWW-sivuilta. Viitattu 15.10.2010.

[10] Lukas Hämmerle ja André Cruz. Web application adaptation for SLO, huhtikuu 2010. URL https://spaces.internet2.edu/display/SHIB2/SLOWebappAdaptation. Shibboleth2:n wiki-sivuilta. Viitattu 12.10.2010.

[11] Ian Hickson et al. HTML5, lokakuu 2010. URL http://www.whatwg.org/specs/web-apps/current-work/multipage/. Web Hypertext Application Technology Working Groupin (WHATWG) WWW-sivuilta. Viitattu 14.10.2010.

[12] Internet2, 2010. URL http://internet2.edu/. Internet2:n WWW-sivut. Viitattu 8.10.2010.

[13] Internet2, 2010. URL http://shibboleth.internet2.edu/. Shibboleth-ohjelmiston WWW-sivut. Viitattu 8.10.2010.

[14] Internet2 et al. eduPerson object class specification, kesäkuu 2008. URL http://middleware.internet2.edu/eduperson/docs/internet2-mace-dir-eduperson-200806.html. Internet2:n WWW-sivuilta. Viitattu 8.10.2010.

[15] Chad La Joie ja Scott Cantor. The difficulties of single logout, heinäkuu 2010. URL https://spaces.internet2.edu/display/SHIB2/SLOIssues. Shibboleth2:n wiki-sivuilta. Viitattu 11.10.2010.

[16] D. Kristol ja L. Montulli. HTTP State Management Mechanism. RFC 2965 (Proposed Standard), lokakuu 2000. URL http://www.ietf.org/rfc/rfc2965.txt.

[17] Liberty Alliance Project, 2009. URL http://www.projectliberty.org/. Liberty Alliancen WWW-sivujen arkisto. Viitattu 8.10.2010.

[18] Mikael Linden. Organising federated identity in finnish higher education. *Computational Methods in Science and Technology*, 11(2):109–118, 2005.

[19] Mikael Linden ja Arto Tuomi. Haka SAML 2.0 profile, helmikuu 2010. URL http://www.csc.fi/hallinto/haka/maaritykset/haka-saml2. Haka-luottamusverkoston WWW-sivuilta. Viitattu 30.9.2010.

[20] NIIF Institute. Single logout in Shibboleth IdP, heinäkuu 2010. URL https://wiki.aai.niif.hu/index.php/ShibIdpSLO. Unkarin AAI-luottamusverkoston wikistä. Viitattu 9.10.2010.

[21] OASIS. Organization for the advancement of structured information standards, 2010. URL http://www.oasis-open.org/. OASIS:n WWW-sivut. Viitattu 8.10.2010.

[22] OASIS. SAML specifications, joulukuu 2009. URL http://saml.xml.org/saml-specifications. SAML:n viralliset WWW-sivut. Viitattu 8.10.2010.

[23] Oracle. Java SE technologies – database, 2010. URL http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html. Oraclen Technology Networkin WWW-sivuilta. Viitattu 12.10.2010.

[24] Oracle. Java servlet technology, 2010. URL http://www.oracle.com/technetwork/java/index-jsp-135475.html. Oraclen Technology Networkin WWW-sivustolta. Viitattu 31.10.2010.

[25] Pat Patterson ja Marina Sum. Single logout: A demo, kesäkuu 2007. URL http://developers.sun.com/identity/reference/techart/single-logout.html. Oraclen Sun Developer Networkin WWW-sivuilta. Viitattu 27.9.2010.

[26] Regents of the University of Michigan. CoSign: Secure, intra-institutional web authentication, kesäkuu 2010. URL http://cosign.sourceforge.net/. CoSign-ohjelmiston WWW-sivut. Viitattu 8.10.2010.

[27] A. Sciberras. Lightweight Directory Access Protocol (LDAP): Schema for User Applications. RFC 4519 (Proposed Standard), kesäkuu 2006. URL http://www.ietf.org/rfc/rfc4519.txt.

[28] A. Solberg, E. Maler, S. Cantor ja L. Johansson. Interoperable SAML 2.0 web browser SSO deployment profile, kesäkuu 2008. URL http://rnd.feide.no/documents/saml2simple.html. Feide-luottamusverkoston WWW-sivu. Viitattu 27.9.2010.

[29] TERENA. Schac: Schema for academia, attribute definitions for individual data, version 1.3.0, joulukuu 2006. URL http://www.terena.org/activities/tf-emc2/docs/schac/schac-schema-IAD-1.3.0.pdf. TERENAn WWW-sivuilta. Viitattu 8.10.2010.

[30] The PHP Group. PHP: Hypertext preprocessor, marraskuu 2010. URL http://www.php.net/. PHP-ohjelmiston WWW-sivut. Viitattu 25.11.2010.

[31] University of Washington, Information Technology. Pubcookie: open-source software for intra-institutional web authentication, elokuu 2010. URL http://pubcookie.org/. PubCookie-ohjelmiston WWW-sivut. Viitattu 31.10.2010.

[32] Don Wells. Unit tests, 1999. URL http://www.extremeprogramming.org/rules/unittests.html. Extreme Programmingin WWW-sivuilta. Viitattu 22.11.2010.

[33] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510 (Proposed Standard), kesäkuu 2006. URL http://www.ietf.org/rfc/rfc4510.txt.