SAML Basics

If you connect a service to SURFconext you will either use SAML or OpenID Connect. Since you are here, you will probably want to know more about SAML. This page will give you an introduction ans some tools to help you understand SAML. With SURFconext you can use SAML 2.0 (Security Assertion Markup Language). This is the protocol that makes single sign-on possible between an identity provider and a service provider. SAML is an XML-based markup language for security assertions, statements that service providers use to make access-control decisions. SAML is also:

- A set of XML-based protocol messages
- A set of protocol message bindings
- A set of profiles (utilizing all of the above)

You can also read out SAML for dummies blog before you continue.

SAML Web browser Single Sing-On step-by step

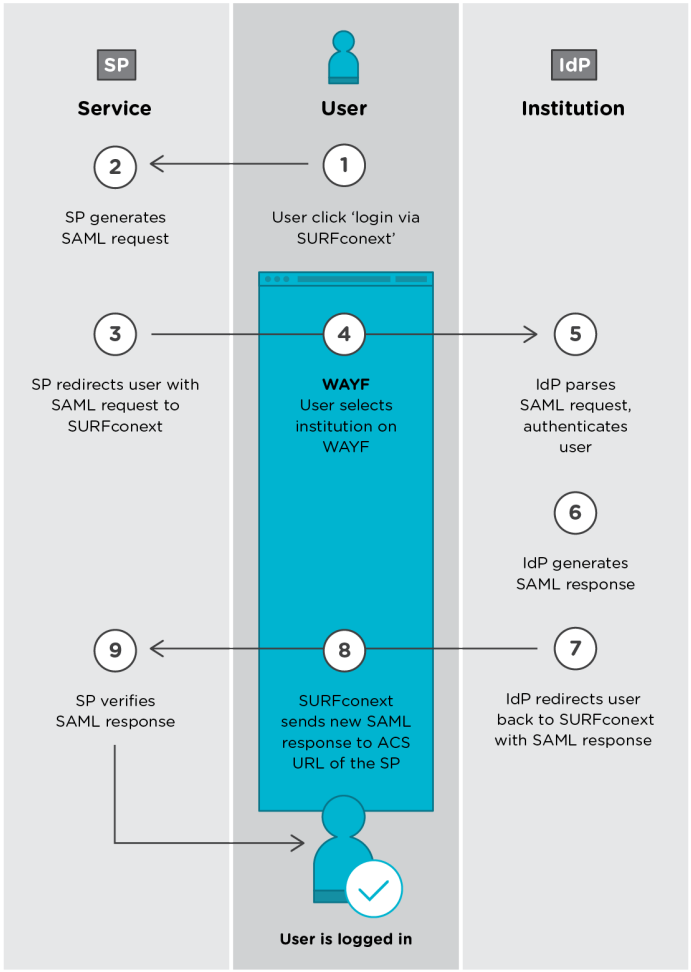
Let's go through the sign-on flow step-by-step. In de example below we have made use of a SAML-tracer. This is an add-on for Mozilla Firefox or Google Chrome that can decode and show you the messages that are send back and forth to SURFconext, the Identity Provider (IdP) and a Service Provider (SP). Let's go through the login process and you can use the SAML-tracer if you want to see this for yourself. The most rudimentary way to explain the single sign-on process, is by going through the following steps:

1. A user navigates to a service and is asked to log-in to the service.
2. The SP sends a message to the IdP, asking to authenticate the user.
3. The IdP asks the user for a username and a password.
4. If correct, the IdP sends an authentication response to the SP saying that the user has logged in successfully, together with a proof that the message was sent by the IdP.

This is the most rudimentary sign-on process possible and a lot of the functionality of SURFconext is not utilized. For instance, a service can be connected to multiple IdP's. The flow as depicted does not facilitate this. Now, let's have a look at how SURFconext handles a sign-on knowing that users from multiple IdP's are allowed to connect to the service.

SURFconext and SAML

The figure below shows you the sign-on process step-by-step. Let's imagine Alice want's to sign-on to a Google Mail instance that is registered for her institution on SURFconext. The authentication will be as shown in the figure below. All messages are send from the browser of the user. In blue you see what happens in the browser of the user at SURFconext. Let's dive into this.



1. Alice will access the remote application using a link on the intranet, a bookmark or similar and the application will load. Let's assume the **URL** is https://mail.google.com/a/my-university.nl
2. The application identifies the user's origin by the application subdomain, user IP address or similar and redirects the user back to the identity provider, asking for authentication. When connected to SURFconext, SURFconext is the Identity Provider to the service. This is the **authentication request** and the SP will generate a **SAML Request**.
3. The SP will instruct the browser to **redirect** the user to SURFconext with a request to authenticate the user..
4. When Alice's browser arrives at the SURFconext she must select her IdP for authentication. This is done using the so-called **WAYF (Were Are You From) page**. Alice chooses the institution she has an account with. The message to SURFconext may look like this (abbreviated):
    a. `https://idp.my-university.nl/sso?SAMLRequest=fVLLTuswEN0j8Q...c%3D`
5. Embedded in the redirection is a **SAML Authentication Request**. This message is compressed to save space in the URL and encoded because some characters are not allowed in URL's. The IdP authenticates the user, usually by asking to enter his credentials. Taken away the encoding and compression, the SAML message could look like this:

SAML Authentication request

```
<AuthnRequest ID="kfcn...lfki"
 Version="2.0"
 IssueInstant="2013-02-05T08:28:50Z"
 ProtocolBinding="urn:oasis:names:tc:SAML:2.0:
bindings:HTTP-POST"
 ProviderName="google.com"
 AssertionConsumerServiceURL="https://www.google.com
/a/idp.my-university.nl/acs"
>
 <Issuer>google.com</Issuer>
 <NameIDPolicy AllowCreate="true"
  Format="urn:oasis:names:tc:SAML:1.1:nameid-format:
unspecified"
 />
</AuthnRequest>
```

Simply said, this message is: "This is a request from the SP with EntityID 'https://mail.google.com/a/my-university.nl'. Please authenticate the user and send the result back to me." *For detailed information about the authentication request read our page about authentication requests.*

6. After successful authentication, the IdP builds the **authentication response** in the form of an XML-document, signs it using an X.509 certificate, and posts this information to the service provider, which is SURFconext for the IdP. This is the **SAML Response** and this message also contains the attributes from the user..
7. The IdP sends sends the **Encoded SAML Response** to Alice' browser. Basically this says “This is a message from 'idp.my-university.nl', I have successfully authenticated a user. Take note that this message will expire in a couple of minutes”. The message contains an XML digital signature, proving that the message was sent by 'idp.my-university.nl'. The signature was made using a public key algorithm, the public key being embedded in a certificate known to the SP. Decoded, the message looks like this:

SAML response to browser

```
<Response
 Version="2.0"
 IssueInstant="2013-02-05T08:29:00Z"
 Destination="https://www.google.com/a/idp.my-
university.nl/acs"
 InResponseTo="kfcn...lfki">
 <Issuer>https://idp.uni.nl/</Issuer>
 <Status>
 <StatusCode Value="urn:oasis:names:tc:SAML:2.0:
status:Success"/>
 </Status>
 <Assertion Version="2.0" IssueInstant="2013-02-
05T08:29:00Z">
 <Issuer>https://idp.my-university.nl/</Issuer>
 <Subject>
 <NameID>alice</NameID>
 <SubjectConfirmation ...>
 <SubjectConfirmationData
 NotOnOrAfter="2013-02-05T08:34:00Z"
 Recipient="https://www.google.com/a/idp.my-
university.nl/acs"
 InResponseTo="kfcn...lfki"/>
 </SubjectConfirmation>
 </Subject>
 <Conditions
 NotBefore="2013-02-05T08:28:30Z"
 NotOnOrAfter="2013-02-05T08:34:00Z">
 </Conditions>
 <AuthnStatement
 AuthnInstant="2013-02-05T08:29:00Z"
 SessionNotOnOrAfter="2013-02-05T16:29:00Z >
 </AuthnStatement>
 </Assertion>
</Response>
```

8. SURFconext validates the response message and if OK makes some alterations, e.g. rewriting the user's identifier and adding or modifying attributes. According to the attribute release policy applied, SURFconext determines the attributes that are allowed through to the Service Provider. The **SAML Response** instructs the browser of Alice to send the response to the **Assertion Consumer Service (ACS) URL** of the SP. An Assertion is a set of security statements about a subject created by an Asserting Party, being an IdP.
9. The SP **verifies the XML signature**, **checks** if the authentication was **successful** and if the message is **not expired**. Then it extracts the **user's identifier and attributes** as known to SP. If this step is OK, Alice is logged on, her mailbox is retrieved and she can read her mail.

Most services will use the flow as described. You will have noticed the service only needs to connect to SURFconext, which is the identity provider to the service, and all the user data such as passwords are managed by IdP's and attributes are managed by SURFconext which will save you a lot of hassle. Also, note that SURFconext acts as both an IdP and an SP. An IdP to the SP and an SP to an IdP.

We have also made a blog post you can read about this flow some time ago and it hasn't changed since.

Navigate