Second Factor Only (SFO) Authentication

Introduction

Second Factor Only authentication allows a SP to authenticate only the second factor of a user. SFO is suitable for situations where the application must perform the fist factor authentication of the user like that application gateway (e.g. Citrix Netscaler or F5 BIG-IP) or to the authentication or authorization gateway (e.g. Microsoft ADFS or Novell/NetIQ) of an institution. SFO has its own authentication endpoint at the SURFsecureID gateway.

Once a user is registered with a second factor in SURFsecureID both SFO authentication and the standard SURFsecureID authentication can be used.

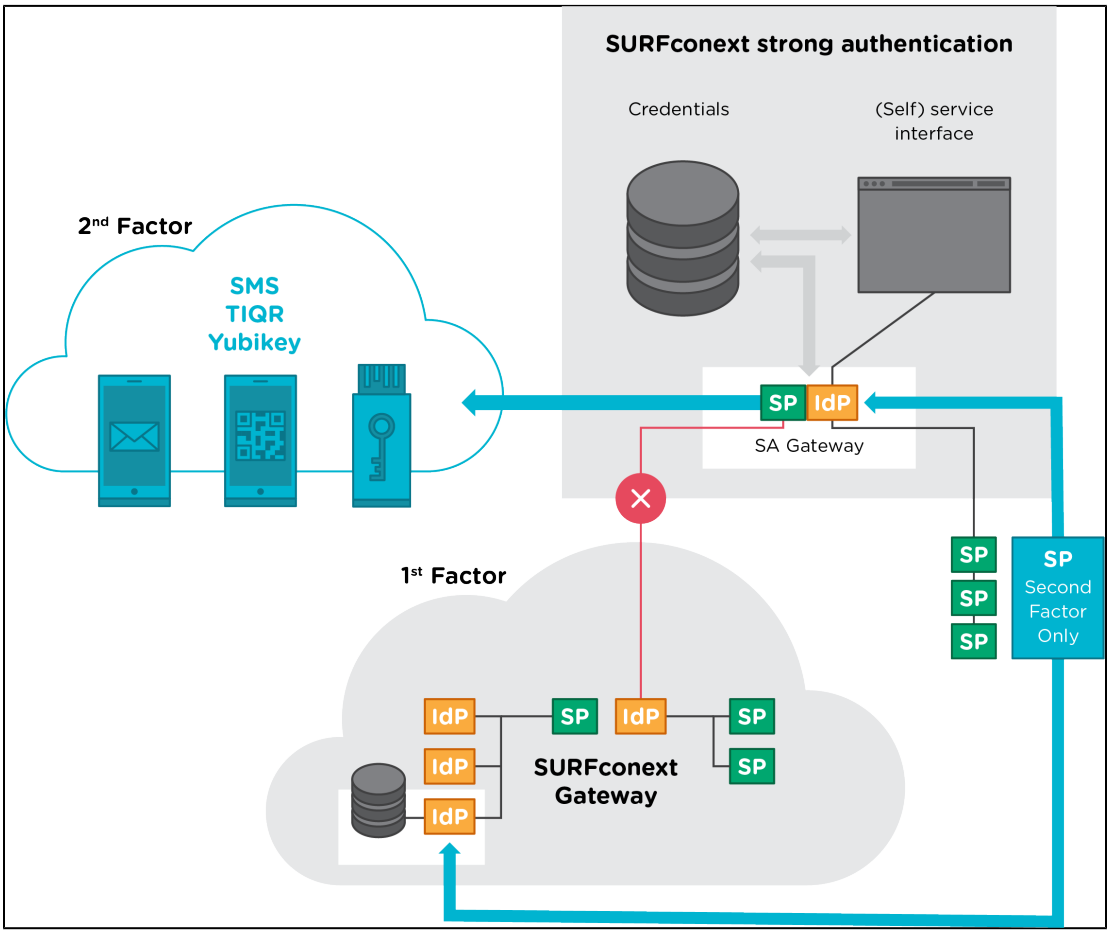
Differences between 'standard' SURFsecureID and SFO authentication

The table below lists the differences between standard authentication and second factor (SFO) authentication that is offered by SURFsecureID gateway to a service provider.

Description	**Standard authentication**	**SFO authenticaton**
Authentication of the user's first factor by the SURFsecureID gateway	Always	Never
Authentication of second factor by the SURFsecureID gateway	Based on policy between IdP and SP	Always
User registration	Using SURFsecureID selfservice registration and vetting by an RA	
Standard SURFconext features that can be used	Attributes, Authorization, persistent identifiers	None
The service provider specifies the identity of the user during authentication	Never	Always

With SFO the authentication via SURFconext is bypassed (see image below). This means that SURFconext functionality (e.g. attributes from the user's home IdP, the definition of authorization rules and persistent user identifiers) is not available. During a standard authentication the identity of the user is detemind by SURFconext during the 1st factor authentication. Because this step is omitted during SFO, the service provider must provide the identity of the user's to SURFsecureID during authentication instead.

Note that also with SFO the registration of users (i.e. linking second factors to user identities) will be done by the institutions (IdP's): there is no work to be done for the SP.



SAML AuthRequest

To start a SFO the SP must send a SAML 2.0 `AuthnRequest` to the SFO endpoint of the SURFsecureID Gateway. This request MUST:

- use the `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect` binding. The `HTTP-Redirect` binding is currently the only suppored binding for making standard or SFO request to SURFsecureID.
- be signed using the `http://www.w3.org/2001/04/xmldsig-more#rsa-sha256` algorithm. Unsigned authentications requests or request that are signed using a different algorithm are not accepted. Note that the `HTTP-Redirect` binding does not use XML signatures. The `HTTP-Redirect` binding specifies its own signature scheme. See Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, section 3.4.
- include a `RequestedAuthnContext` with an `AuthnContextClassRef` with the URI for one of the defined authentication levels for the SURFsecureID environment that your are authenticating to.
- include the SURFconext identifier of the user in the `Subject` element as a `NameID` (with `Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"`, see description of `AuthnRequest` in https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, line 2001).

SURFsecureID Implementation notes regarding `RequestedAuthnContext`

- The SAML standard allows multiple `AuthnContextClassRef` elements to be specified in the `RequestedAuthnContext`. Currenly SURFsecureID will only look at the first `AuthnContextClassRef` element.
- Specifying an `AuthnContextClassRef` other than one the of the defined authentication levels for SFO will result in an error.
- The SAML standard allows a `Comparision` attribute to be added to the the `RequestedAuthnContext` element. Currently SURFsecureID does not interpret the value of this attribute and behaves as if `"minimum"` was specified as value for the `Comparison` attribute, which is a deviation of the SAML standard which specifies `"exact"` as the default. `"minimum"` means that the authentication context in the authentication statement that is returned after a successfull authentication will either be the requested authentication context, or the the authentication context of a stronger (i.e. higher level) authentication. SURFsecureID currently always returns the authentication context corrsponding to the highst level at which the user could be authentictated.

Future SURFsecureID versions may support more complex processing of `RequestedAuthnContext` options and add new `AuthnContextClassRef` "families" to support different registration policies.

Example AuthnRequest

Below is an example SAML 2.0 SFO AuthnRequest request for the SURFsecureID production environment:

Example SFO AuthRequest to the SURFsecureID production environment

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                                        ID="_zQIibz9FKixdlgX8E7bHqE29wfatcgbsPdVn0NN"
                                    Version="2.0"
                                        IssueInstant="2016-03-10T15:09:21Z"
                                    Destination="https://sa-gw.surfconext.nl/second-factor-only/single-sign-
on"
                                    AssertionConsumerServiceURL="https://application-gateway.some-
organisation.example.org/consume-assertion"
                                    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST">
    <saml:Issuer>https://application-gateway.some-organisation.example.org/metadata</saml:Issuer>
    <saml:Subject>
        <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">urn:collab:person:some-
organisation.example.org:m1234567890</saml:NameID>
    </saml:Subject>
    <samlp:RequestedAuthnContext>
        <saml:AuthnContextClassRef>http://surfconext.nl/assurance/sfo-level2</saml:AuthnContextClassRef>
    </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

Note that the signature is not visible in the XML of the above `AuthnRequest`: it will be encoded in the HTTP GET parameters according to the specification of the `HTTP-Redirect` binding.

Note that SFO uses a different `SingleSignOn Location` and a different `AuthnConextClassRef` identifiers than a standard authentication to SURFsecureID. See SURFsecureID Metadata for Service Providers for the diffenrent `AuthnConextClassRef` identifiers that are being used by SURFsecureID.

Determining the SURFconext identifier of a user

The SURFconext identifier of a user is built from the values if two different attributes that the identity provider (IdP) of the user's institution sends to SURFconext during authentication. The two attributes that are used to create the SURFconext user identitfier are:

1. `urn:mace:terena.org:attribute-def:schacHomeOrganization`: the value of this attribute identifies the user's institution.
2. `urn:mace:dir:attribute-def:uid`: the value of this attribute identifies the user withing the institution

`urn:collab:person:{{urn:mace:terena.org:attribute-def:schacHomeOrganization}}:{{urn:mace:dir:attribute-def:uid}}`
where:

- `urn:collab:person:`
  = fixed prefix.
- `{{urn:mace:terena.org:attribute-def:schacHomeOrganization}}`
  = value of schacHomeOrganization attribute of the user; typically the same for all users of one institution and will be something like "`institution.nl`".
- `{{urn:mace:dir:attribute-def:uid}}`
  = value of `uid` attribute of the user. Replace each "@" (at) character in the uid with an "_" (underscore) character.

For the value of last two items: ask the administrator of the IdP .

Examples:
`urn:collab:person:some-organisation.example.org:m1234567890`
`urn:collab:person:uniharderwijk.nl:jan.jansen_uniharderwijk.nl`

SAML Response

The result of a successful authentication is a SAML `Response`. Note that it does not contain an `AttributeStatement` and that the `Assertion` element is signed and that the `Response` element is not signed. Response signing is not currently supported by SURFsecureID, it may be added in future versions.

Example Response

```
 <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                 ID="_ECAokbn0lm7lfVT7THQUl+dSbMrpeyAgiTv0+q16"
                 Version="2.0"
                 IssueInstant="2016-03-10T15:09:25Z"
                 Destination="https://application-gateway.some-organisation.example.org/consume-assertion"
                 InResponseTo="_zQIibz9FKixdlgX8E7bHqE29wfatcgbsPdVn0NN">
    <saml:Issuer>https://sa-gw.surfconext.nl/second-factor-only/metadata</saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </samlp:Status>
    <saml:Assertion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    ID="_zQIibz9FKixdlgX8E7bHqE29wfatcgbsPdVn0NN"
                    Version="2.0"
                    IssueInstant="2016-03-10T15:09:25Z"
                    >
        <saml:Issuer>https://gw.stepup.example.org/second-factory-only/metadata</saml:Issuer>
        <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/2001/04/xmldsig-more#rsa-sha256" />
                <ds:Reference URI="#_1ROuxGVzJi5bbre6W4woNza82aK41HKjp6aKtw9r">
                    <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/2001/04/xmlenc#sha256" />
                    <ds:DigestValue>YR5JFfJc1JZIKm7Ao3kXtDupEfeMrhKpD9T1lF1z0Lg=</ds:DigestValue>
                </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>...</ds:SignatureValue>
            <ds:KeyInfo>
                <ds:X509Data>
                    <ds:X509Certificate>...</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </ds:Signature>
        <saml:Subject>
            <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">urn:collab:person:
some-organisation.example.org:m1234567890</saml:NameID>
            <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml:SubjectConfirmationData NotOnOrAfter="2016-03-10T15:14:25Z"
                                              Recipient="https://application-gateway.some-organisation.
example.org/consume-assertion"
                                              InResponseTo="_zQIibz9FKixdlgX8E7bHqE29wfatcgbsPdVn0NN"/>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Conditions NotBefore="2016-03-10T15:09:25Z"
                         NotOnOrAfter="2016-03-10T15:14:25Z">
            <saml:AudienceRestriction>
                <saml:Audience>https://application-gateway.some-organisation.example.org/metadata</saml:
Audience>
            </saml:AudienceRestriction>
        </saml:Conditions>
        <saml:AuthnStatement AuthnInstant="2016-03-10T15:09:25Z">
            <saml:AuthnContext>
                <saml:AuthnContextClassRef>http://stepup.example.org/verified-second-factor/level2</saml:
AuthnContextClassRef>
            </saml:AuthnContext>
        </saml:AuthnStatement>
    </saml:Assertion>
</samlp:Response>
```

Error handling

For specific scenarios, when the authentication fails, the SURFsecureID gateway sends a `SAMLResponse` to the SP with a non success `status`:

- `urn:oasis:names:tc:SAML:2.0:status:Responder` with subcode `urn:oasis:names:tc:SAML:2.0:status:AuthnFailed` =
  Authentication was not successful. This specifically happens when the user cancels the authentication.
- `urn:oasis:names:tc:SAML:2.0:status:Responder` with subcode `urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext` =
  The user could not be authenticated at the requested level. The user does not have an activated (vetted) token at the requested level.
- Other situations can also lead to an error response. Specifically:
  - No `RequestedAuthnContext` or `AuthnContextClassRef` in the `AuthnRequest`
  - An unsupported URI in the `AuthnContextClassRef`
  - No `Subject NameID` in the `AuthnRequest`
  - The service provider is not authorised to authenticate the user specified in the `Subject NameID`

A service provider SHOULD be able to handle the first two errors scenarios above (`AuthnFailed` and `NoAuthnContext`) in a user friendly manner. These error responses will occur during normal use: users can and do cancel the authentication process and users that do no not yet, or no longer have, a registered seond factor will try to authenticate to your service, and fail.

SURFsecureID does not currenly add a `StatusMessage` to an "error" response. We plan to add a `StatusMessage` in a future version of SURFsecureID that provides more context about the error to the operator of the service than can be conveyed using the standard `statusCodes`.

Example Error Response

Below is an example SAML "error" Response:

Example Error Response

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                ID="YaszKubip05bTwe7hIWOc5AsNxwmEliPJ88nUQ"
                Version="2.0"
                IssueInstant="2015-05-12T12:17:38Z"
                Destination="https://your-sp.example.com/acs-location"
                InResponseTo="_6d93f735ccfb8d98454999b4016d515834211b0dde"
                >
    <saml:Issuer>https://sa-gw.surfconext.nl/authentication/metadata</saml:Issuer>
    <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Requester">
            <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext" />
        </samlp:StatusCode>
    </samlp:Status>
</samlp:Response>
```

Level: authentication strength

See explanation at "Levels of Assurance".

Implementation

SFO must be implemented at the SP. The authentication protocol is similar to the one used by the SURFsecureID gateway. The main difference is that the SP must send the identifier of the user in the Subject element of the SAML AuthnRequest (see description of AuthnRequest, line 2017).

- The SP will be registered at the SURFsecureID gateway as either SFO SP or a 'standard' SURFsecureID SP: this determines which endpoint he can access. If a SP implementation needs both, he can register an additional EnityID and use it to access the other endpoint.
- There is a white-list of SURFconext identities allowing SFO authentication. The SP must indicate in advance the institutions from which he wants to authenticate users with SFO.

You can find the metadata of the SFO endpoints on SURFsecureID Metadata for Service Providers.

Always do a first factor authentication before starting a SFO authentication

Starting an SFO authentication will immediately start an authentication at the SURFsecureID gateway: a push notification (tqr) or an SMS will be sent to the user being authenticated. If authentication is started for the wrong user, this will derange the targeted user and in case of SMS, incur a cost to the institution and possibly for the user.

By observing the behavior of the SFO authentication it is possible to determine whether a username exists.

For this reasons we advise to perform a first factor authentication before starting a SFO authentication.

Example

An example code for using SFO with SimpleSAMLphp can be found at: https://github.com/SURFnet/Stepup-SFO-demo