X509BindingSAML

An X.509 Binding for SAML

Currently, GridShib and CAS bind self-issued SAML assertions to proxy certificates. !GridShib, for example, has the ability to bind an authentication assertion (i.e., an assertion containing an `AuthenticationStatement=) to a proxy. Similarly, CAS binds an assertion containing an =AuthorizationDecisionStatement` to a proxy. These and other anticipated [X509BindingSAMLUseCases have been documented elsewhere.

Note: A poster entitled "An X.509 Binding for SAML", which was presented at the Midwest Security Workshop 2006, is attached to this wiki topic.

What to Bind

The question naturally arises as to what SAML blob should be bound to a certificate in general. There are two obvious choices:

1. A `<saml:Assertion>` element (like GridShib and CAS do now)
2. A `<samlp:Response>` element (obtained directly from a Shibboleth !IdP)

Does it make sense to *always* bind a `<saml:Assertion>` element to a certificate? If so, a Shibboleth response must be decomposed into its constituent `<saml:Assertion>` elements, all of which are subsequently bound to the certificate. This is problematic for the following reasons:

- Decomposition breaks the signature on the `<samlp:Response>` element (if it exists).
- Decomposition breaks the namespace declarations on the `<samlp:Response>` element. Moreover, these can not be patched since the assertions themselves may be signed.
- Decomposition may result in multiple assertions.

In the case of multiple assertions, binding each assertion to its own extension is difficult due to the following requirement from RFC 3280:

> A certificate MUST NOT include more than one instance of a particular extension.

Moreover, it is not feasible to have a separate OID for each "type" of assertion since the number of possibilities is theoretically infinite. Thus we must use a single extension (OID) to bind multiple assertions.

Solution

We bind an ASN.1 SEQUENCE of `<saml:Assertion>` elements at a well-known, non-critical X.509 v3 certificate extension.

SSO Assertions

The following lines are taken from the SAML V1.1 Bindings and Profiles specification SAMLBind:

> *In the discussion of the web browser SSO profiles, the term SSO assertion will be used to refer to an assertion that has a `<saml:Conditions>` element with `NotBefore` and `NotOnOrAfter` attributes present, and also contains at least one or more authentication statements about the subject. Note that an SSO assertion MAY also include additional information about the subject, such as attributes.*

Definition An *SSO assertion bundle* consists of one or more assertions obtained as a result of a web browser SSO profile. At least one of these assertions is an *SSO assertion* as described in SAMLBind.

In what follows, we use the term "SSO assertion" to refer to any assertion in an SSO assertion bundle.

Many use cases transform Shib-issued SSO assertions into attribute-bearing grid credentials. With this in mind, it is useful to enumerate the types of responses obtained in conjunction with a Shibboleth browser profile:

- Shibboleth Attribute Pull
    1. authentication response: one assertion containing an `AuthenticationStatement`
    2. attribute response: one assertion containing an `AttributeStatement`
- Shibboleth Attribute Push
    1. authentication response: two assertions, one containing an `AuthenticationStatement` and the other containing an `AttributeStatement`

Today, the Shib SP exposes the last response it receives from the !IdP. In the case of attribute pull (the default), this is an attribute response. In the case of attribute push, the authentication response is exposed. Thus the assertion payload depends on the protocol flow (push or pull). In both cases, however, an assertion containing an `AttributeStatement` is exposed by the Shib SP. It is this attribute assertion that is of primary interest.

Note: As required by the SAML request-response protocol, Shib-issued SSO assertions are contained by a `<samlp:Response>` element, which may be signed. Additionally, the assertions themselves may be signed.

Finally we observe that SAML assertions are not easily classified. For example, we often use the term "authentication assertion" to refer to "an assertion that contains an

```
AuthenticationStatement
```

" but this definition is vague since other statements can (and do) appear in such assertions, namely, `AttributeStatement` . So be aware that our working definitions of "authentication assertion" and "attribute assertion" overlap somewhat.

Binding Requirements

To solve this problem, let's classify embedded SAML assertions into three distinct groups:

1. Assertions that are issued by a CA
2. Assertions that are self-issued
3. Assertions that are neither CA-issued nor self-issued (i.e., third-party assertions)
    - An SSO assertion is a type of third-party assertion

For all these assertion types, we insert a top-level `<saml:Assertion>` element into the certificate (as done now by GridShib and CAS) at a well-known certificate extension. In the case of SSO assertions, we routinely insert a `<saml:Assertion>` element into the `<saml:Advice>` element of a self-issued assertion.

CA-issued Assertions

Definition If the certificate is an end-entity certificate (EEC), and the Issuer of the assertion is the Issuer of the EEC, the assertion is called a *CA-issued assertion*.

- A CA-issued assertion SHOULD NOT be signed. A relying party MAY ignore a signature on a CA-issued assertion.
- In a CA-issued assertion, the `NotBefore` and `NotOnOrAfter` attributes SHOULD be omitted. If these attributes are present, their values MUST be equal to the values of the `NotBefore` and `NotOnOrAfter` fields (resp.) of the EEC.
- The `<saml:NameIdentifier>` element of a CA-issued assertion MUST have a `Format` attribute whose value is equal to `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName` . The value of the `<saml:NameIdentifier>` element MUST be equal to the Subject DN of the EEC.
- A CA-issued assertion SHOULD NOT contain a `SubjectConfirmation` element. A relying party MAY ignore a `SubjectConfirmation` element in a CA-issued assertion.

Self-issued Assertions

Definition If the certificate is a proxy certificate, and the Issuer of the assertion is the Subject of the proxy, the assertion is called a *self-issued assertion*.

- A self-issued assertion SHOULD NOT be signed. A relying party MAY ignore a signature on a self-issued assertion.
- In a self-issued assertion, the `NotBefore` and `NotOnOrAfter` attributes SHOULD be omitted. If these attributes are present, their values MUST be equal to the values of the `NotBefore` and `NotOnOrAfter` fields (resp.) of the proxy.
- The `<saml:NameIdentifier>` element of a self-issued assertion MUST have a `Format` attribute whose value is equal to `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName` . The value of the `<saml:NameIdentifier>` element MUST be equal to the Subject DN of the proxy.
- A self-issued assertion SHOULD NOT contain a `SubjectConfirmation` element. A relying party MAY ignore a `SubjectConfirmation` element in a self-issued assertion.
- If the Subject of a self-issued assertion is the Subject of the certificate (i.e., the presenter is the subject), the assertion MUST contain only `<saml:AttributeStatement>` elements. The assertion is considered invalid if it contains any other type of statement.
- A self-issued assertion MAY contain a `<saml:Advice>` element with one or more nested SAML assertions. The nested assertions contain third-party assertions that the relying party MAY use in making its access control decision.

Third-party Assertions

Definition An assertion that is neither a CA-issued assertion nor a self-issued assertion is called a *third-party assertion*.

- A top-level, third-party assertion MUST be signed. A relying party MUST validate the signature on a top-level, third-party assertion.
- If the Subject of a top-level, third-party assertion is not the Subject of the certificate, the assertion SHOULD contain a `SubjectConfirmation` element with subject confirmation method `holder-of-key` .
- If the Subject of a top-level, third-party assertion is the Subject of the certificate, the assertion MUST contain a `SubjectConfirmation` element with subject confirmation method `holder-of-key` . In this case, the relying party MUST confirm the subject, that is, the relying party MUST verify that the subject (who is also the requester) possesses the corresponding private key.
- Regardless of its content, an SSO assertion MUST be bound to a certificate by inserting the `<saml:Assertion>` element into the `<saml:Advice>` element of a self-issued assertion. Thus an SSO assertion is a *nested assertion* (unlike a top-level assertion).
- A nested assertion SHOULD be signed. A relying party MAY reject an unsigned nested assertion, subject to policy. If the nested assertion is signed, a relying party MAY validate the signature or not, subject to policy.
- An entity that binds an SSO assertion to a certificate MUST authenticate the !IdP that issued the assertion. In particular, the entity MUST validate the signature on the `<samlp:Response>` element (if it exists) before stripping the `<samlp:Response>` element and binding the enclosed assertion to the certificate.
- An entity that binds an SSO assertion to a certificate SHOULD set the SIA extension of the certificate to the `entityID` of the SAML issuer. A relying party MAY query the !IdP based on the `entityID` in the SIA extension.
