OSTwoUsrManCPPsamlsign

`samlsign` is a test program developed to exercise a variety of options related to creating and verifying signatures using the OpenSAML and XMLTooling code and plugins. It isn't formally a utility designed for end users of applications that are built with the library, nor is it meant as a generic signature tool, but it's powerful enough to be used to sign and verify SAML-oriented XML content.

It's also useful in some cases to debug SAML interactions, because it has the ability to instantiate a MetadataProvider and apply it via a TrustEngine plugin to verify a signature, emulating the process an actual OpenSAML application should be using.

See the following sections for information on the various parameters supported.

> I realize the lack of a supported metadata signing tool has led people to rely on this utility for that purpose. Nevertheless, it's not supported for that, by which I mean things like command line options might change, which would break scripts that federations might be relying on, so use at your own risk.

To use the program to sign or verify something, you must supply XML that can be parsed into an XMLObject that is a subclass of the OpenSAML SignableObject class. This includes all SAML objects that contain signatures. You can also supply XML that contains such an object by also supplying an "-id" parameter to locate the SAML object in the document.

Signing

To use the program for signing, you must pass a "-s" parameter.

At the moment, only RSA signing with the default SHA-1 digest is supported.

Input and Output

-f	path to an XML file to process
-u	URL of an XML file to process (uses the XML parser's NetAccessor)
-id	XML ID to find in the document

If neither parameter is used, then stdin is used for input.

When signing succeeds, the signed output is written to stdout.

Supplying a Signing Key

-k	path to a file with an RSA private key
-R	path to a file that describes a CredentialResolver plugin

One of these parameters is required. In the latter case, you must create a small XML file containing a configuration snippet that instantiates a plugin. An example follows.

Example CredentialResolver snippet

```
<CredentialResolver type="File" key="/path/to/key.pem"/>
```

Optional Parameters

-c	path to a file with an X.509 certificate to include in the signature

Verifying

If the "-s" parameter is omitted, the program assumes a verify operation.

In theory, any signature that can be processed by the libraries, and that you supply the right verification material for should be usable, but only RSA signatures have been tested. Some algorithms may be unsupported by your version of OpenSSL as well.

Input and Output

-f	path to an XML file to process
-u	URL of an XML file to process (uses the XML parser's NetAccessor)

-id	XML ID to find in the document

If neither parameter is used, then stdin is used for input.

The result of the verification will be logged at the INFO level via whatever logging configuration is supplied via the XMLTOOLING_LOG_CONFIG environment variable. WARN is the default level, so success will produce no output, and a 0 return code. Failure results in log messages and a negative return code.

Supplying a Verification Key

-c	path to a file with an X.509 certificate
-R	path to a file that describes a CredentialResolver plugin
-T	path to a file that describes a TrustEngine plugin
-M	path to a file that describes a MetadataProvider plugin

One of the first three parameters is required. The last is usually required if "-T" is used, though not always.

In the latter cases, you must create a small XML file containing a configuration snippet that instantiates a plugin. Examples follow.

Example CredentialResolver snippet

```
<CredentialResolver type="File" certificate="/path/to/cert.pem"/>
```

Example TrustEngine snippet

```
<TrustEngine type="ExplicitKey"/>
```

Example MetadataProvider snippet

```
<MetadataProvider type="XML" file="/path/to/metadata.xml"/>
```

Supplying Metadata

Most uses of the "-T" option to utilize a TrustEngine plugin will require supplying SAML metadata via the "-M" option for the entity that signed the XML. The metadata will be used to establish whether or not the signing key is trusted.

When using this option, the following parameters are used to identify the issuer of the XML to find in the metadata, and the various qualifiers used to identify the role in which the entity is acting. Trust information in metadata is specific to a role.

-i	the signer's entityID
-p	a protocolSupportEnumeration value to use in finding the signer's role
-saml10	shortcut for "-p urn:oasis:names:tc:SAML:1.0:protocol"
-saml11	shortcut for "-p urn:oasis:names:tc:SAML:1.1:protocol"
-saml2	shortcut for "-p urn:oasis:names:tc:SAML:2.0:protocol"
-r	name of the role element/type to lookup
-ns	XML namespace of the role element/type to lookup (defaults to the SAML 2.0 metadata namespace)
-idp	shortcut for "-r IDPSSODescriptor"
-sp	shortcut for "-r SPSSODescriptor"
-aa	shortcut for "-r AttributeAuthorityDescriptor"
-pdp	shortcut for "-r PDPDescriptor"