SPPKIConfig

Server Credentials & PKI

The service provider uses PKI to authenticate signed assertions from and establish secure, mutually identified connections directly with IdPs. The credentials used by the SP for these flows are defined in `Credentials` element of `shibboleth.xml`. Individual sets of credentials for these purposes are specified on a per relying party and protocol handler (TLS or XML signing) basis. It is important that these credentials match those supplied to relying parties and federations as included in `metadata.xml` or trust failures will result.

Sessions between end-users and the webserver/webapps should be SSL-protected in most cases. It is permissible for Shibboleth to use the same keypair and certificate used by the web server itself, provided the certificate is signed by a CA accepted by both the IdPs that will be queried for attributes and commonly used browsers. Most well-known roots will satisfy both conditions but many federations will have specifications on accepted credentials.

On Unix, OpenSSL must be installed to use Shibboleth. On Windows, OpenSSL libraries and the command line tool are included in the package and can be used directly, if not otherwise available. Certain commands require the `openssl.cnf` configuration file, an example of which is included with the Windows installation in `C:\opt\shibboleth\etc\shibboleth\openssl.cnf`. To locate this file for a command that requires it, add the `-config C:\opt\shibboleth\etc\shibboleth\openssl.cnf` parameter to the command.

The certificate and key file location should be based on whether they will also be used for Apache. If they will be used as a server key pair as well, they should probably be in the Apache tree in the usual `mod_ssl` defined locations inside the Apache configuration folder, and the Shibboleth module can read them from there. If `shibd` is not running as `root`, permissions might need to be changed to allow this access. If the certificate and key will only be used for the SHAR, they can be put in the same folder with the `shibboleth.xml` file and protected appropriately.

Other web servers like IIS do not use the file formats that Apache and Shibboleth can share, and therefore the components must generally use separate copies of the key and certificate if they are to be shared. Most other servers can export and/or import keys to and from PEM or DER format. Refer to your server's documentation.

`shibd` is assigned a key and a certificate using the `CredentialUse` element of `shibboleth.xml`. Various formats are supported and OpenSSL can generate and convert among them.

If the key is to be shared with Apache, the web server's child processes, often running as `nobody` or a similar `uid` , must be able to read them while the server is running, which may require permission changes.

This particularly applies when sharing the key and certificate used by `mod_ssl` , which are only readable by root by default. The password, if any, must be placed in the `shibboleth.xml` file, since the Apache module cannot prompt for it during initial startup as `mod_ssl` can. Since the password will be stored in clear text in a frequently examined file, it is suggested to use a password not used elsewhere, or preferably not to use a password at all.

Example usage:

```
<Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
    <FileResolver Id="mycerts">
        <Key>
            <Path>file:/opt/shibboleth-sp/etc/shibboleth/supervillain.key</Path>
        </Key>
        <Certificate>
            <Path>file:/opt/shibboleth-sp/etc/shibboleth/supervillain.crt</Path>
        </Certificate>
    </FileResolver>
</Credentials>
```

```
<Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
```

This element is the container for credentials used by the credential mechanism specified by the `SPConfig` element. It must contain one `FileResolver` element for flat key and certificate files or one `KeyStoreResolver` element for compound keystores.

File-Based Key and Certificate Storage

Flat-file based certificate storage and protection is by far the most straightforward and common credential storage mechanism. Shibboleth supports `.pem` and `.der` encodings. These certificate files may be easily shared by both Apache and Shibboleth in the common case where the same key and cert are used for all flows.

It is **critical** that these files be stored with appropriate permissions in an appropriate location. Any compromise of these files represents a compromise of your entire Service Provider. Revoke the certificate and contact all your relying parties **immediately**.

```
<FileResolver Id="string">
```

This element defines a pair of files used to store a private key and certificate associated with a given identifier and is contained by the `Credentials` element. `RelyingParty` elements will refer to these identifiers allowing multiple resolver elements to be used to specify different credential storage for different federations or identity providers. It must contain one `Key` element and should contain one `Certificate` element.

`<Certificate>`

This specifies the certificate corresponding to this set of credentials. Certificates are supported in a variety of automatically detected formats: DER, PEM, PKCS8, PKCS12, and all associated encrypted forms. The certificate itself must be referred to using a `Path` element contained by this element. If this certificate isn't self-signed or signed by a root familiar to the identity provider, the files of certificates in the path to the root may be specified using one or more `CAPath` elements. It resides within the `FileResolver` element and must be paired with the corresponding private key using the `Key` element. An optional `type` attribute can explicitly force a format if needed.

`<Key password="`*`password`*`">`

This specifies the file containing a private key to be used by a set of credentials. Keys are supported in a variety of automatically detected formats: DER, PEM, PKCS8, PKCS12, and all associated encrypted forms. For encrypted keys, the password may be specified by adding an optional `password` attribute to this element. It resides within the `FileResolver` element, should be paired with a `Certificate` element, and contain a `Path` element. An optional `type` attribute can explicitly force a format if needed.

`<CAPath>`*`pathname`*`</CAPath>`

Paired with a `Path` element and contained by a `FileResolver` element, this element allows for the specification of additional certificates in the chain up to the trust anchor. As many `CAPath` elements as necessary to complete the chain may be specified. The expectations of identity providers and the federation may determine the necessity for the use of this field.

`<Path>`*`pathname`*`</Path>`

This mandatory element specifies the path to a file or directory utilized by other elements of the configuration. It may be contained by various elements to point to different types of files required by the SP.