IdPKeyRollover

Identity Provider Key Rollover

The term *key rollover* refers to a process whereby one key is systematically replaced by another key in SAML metadata. Since SAML entities (and therefore SAML metadata) are distributed, key rollover must be deliberate, so as not to break the key operations of a relying party.

The IdP key rollover process depends on the representation of keys in IdP metadata, specifically, the precise form of the `<md:KeyDescriptor>` elements appearing in metadata. Since the Shibboleth IdP does not support decryption operations, we can make the following important simplifying assumption:

> ⚠ All key descriptors in Shibboleth IdP metadata are of the form `<md:KeyDescriptor use="signing">`.

If your IdP is distributing metadata with other key descriptors, that's probably a bug. In particular, if your metadata has a key descriptor with no use XML attribute, then decryption support is implied, which is a bug.

Under normal circumstances, for each `<md:KeyDescriptor use="signing">` element in metadata, the IdP is configured to use the corresponding private key as both a signing key and an SSL/TLS key. However, while key rollover is in progress, there is at least one `<md:KeyDescriptor use="signing">` element in metadata for which the IdP is not configured. This implies that relying party implementations MUST be able to consume and utilize two signing keys bound to a single role descriptor in metadata. If that's not true, then you need to work directly with your federation partners to minimize service disruptions during key rollover.

By virtue of the above simplifying assumption, the key rollover process for the Shibboleth IdP is relatively straightforward:

1. Add the new `<md:KeyDescriptor use="signing">` element to metadata
2. Wait for the newly updated metadata to propagate
3. Configure the software to use the new key (instead of the old key) as the signing key and the SSL/TLS key
4. Remove the old `<md:KeyDescriptor use="signing">` element from metadata

In practice, there is actually a step 0: *create a new key pair for signing and SSL/TLS*. However, do **not** configure the IdP software with this new key initially. Instead the new key is configured at step 3 of the key rollover process.

In what follows, only step 3 is addressed since the other steps are beyond the scope of this article. If your IdP is a member of a SAML federation, consult your federation's documentation for further information about their key rollover process.

Although SAML metadata expresses a single key for both signing and SSL/TLS, the Shibboleth IdP has separate configurations for each. In Shibboleth 2.x, for example, a signing credential is configured in relying-party.xml and then that credential is associated with specific protocols as desired. The latter is beyond the scope of this article, so all you need to do is configure the new signing credential in relying-party.xml. Even that may not be necessary if the new credential physically replaces the location of the old credential.

The next step is to configure the key for SSL/TLS as well (assuming the IdP supports artifact resolution or attribute query, both of which likely require SSL/TLS server authentication). If Apache is used in front of the Java container (usually Tomcat), the configuration of the SSL/TLS credential occurs in the Apache configuration file. If, on the other hand, Tomcat is used as a stand alone server, the new SSL/TLS credential is added to both Tomcat's keystore and its configuration file (server.xml). In either case (Apache or Tomcat), the server is restarted to complete the configuration.