OSTwoUserManSigErrors

Troubleshooting XML Signature Problems

Background

Digital signatures of any kind (XML or otherwise) ensure that a piece of data is not modified. If signed data is modified even the slightest after signing, the signature fails to validate. Nearly all signature issues are due either to unintentional modifications or simple misconfiguration error (i.e. trying to verify a signature with a public key that is not other half of the private key used to sign the data).

In order to reduce some of the fragility, XML elements that are to be signed go through a canonicalization, or normalization, process that does the following (amongst other more specialized things):

- remove XML declaration (i.e. <?xml>) and DTD declarations
- encodes the document as UTF-8
- changes all line endings to one single type (#xA to be exact)
- convert <foo/> elements to <foo></foo>
- lexicographically order attributes (i.e. alphabetize them)
- convert any single-quoted attribute values to double-quoted values

However, some very common changes to XML documents are not covered by this. For example, the following changes will break an XML signature:

- adding a carriage return
- adding spaces to element content, e.g., <foo>this is a test</foo> to <foo>this is a test</foo> (and yes, part of the point is that the two look identical when displayed)
- changing the prefix of an XML namespace

Troubleshooting Steps

Here are some steps to take when troubleshooting a signature.

1. Ensure your keys are right

As mentioned above, a common misconfiguration that can lead to signature validation problems is using the private key of one key pair to sign and the public key of another key pair to validate. Make sure that you're using the right key to validate.

2. Ensure your software is update

The next step is to make sure that all the parties are running current software versions. Don't waste time trying to debug problems when they could just be bugs that have already been fixed. At the very least, make sure the current versions work before opening that can of worms.

3. Ensure the content isn't being changed by your parser

Some XML parsers contain options that will attempt to "help" you. For example, by performing some sort of string normalization on attribute/element content or namespaces. This will almost always break signatures. Enabling "in-line" schema validation (i.e., performing schema validation while parsing) can also lead to problems if the schema populates the resulting DOM with schema-defaulted values for attributes.

In the case of opensaml, which usually relies on Xerces for its XML parsing, be aware of the parser feature `http://apache.org/xml/features/validation/schema/normalized-value`.

4. Validate the signature with known-good tools

Software can have bugs, but it's unlikely that two pieces of software written by two entirely different groups will have the same bug in the same places. So, to ensure the signature problem isn't related to the software you're using, check the signature with some other software. For example, the oXygen XML Editor or the online signature verifier. If the signature verifies with some other software chances are good there is a bug in the software you're using.

5. Compare the content

The previous few checks were meant to be quick checks that could be easily performed. Now it's time to check the actual signed content. To do this you'll need to get the "pre-digest" value, that is, the XML after it is canonicalized but before it is signed/verified. You'll then need to compare the pre-digest value from before the signature to the one before the verification using something like the Unix Diff tool. **Any** differences will result in a failed signature.

For Java, you'll need to enable debug logging for `org.apache.xml.security.utils.DigesterOutputStream`. In some rare cases, where the the problem is not that the signed XML itself has been changed but rather the contents of the `<ds:Signature>` element itself, you may need to also examine the debug logging for `org.apache.xml.security.utils.SignerOutputStream`.

The C++ library starting with version 1.6 includes some logging support for obtaining the digested octets. You can set an environment variable called XSEC_DEBUG_FILE and point it to a file to receive the information. When the C++ OpenSAML stack is used, there is explicit logging support via the usual logging mechanisms in the "XMLTooling.Signature.Debugger" category.