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>)
- 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. 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 suing.

4. 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`

The C++ library is not simple to deal with because C/C++ have no intrinsic logging features and the code includes no logging support. It does include a bit of hacked code that is compiled out by default that causes the executing process to write to a file with the digested data. It overwrites the file every time a new operation is performed. Because it's compiled out, and because there are no current hooks to enable it, turning this on requires building from modified source, specifically enabling a bit of code inside **src/dsig/DSIGReference.cpp**:

Debugging support inside src/dsig/DSIGReference.cpp

```
#if 0
        TXFMOutputFile * of = new TXFMOutputFile(d);

        of->setFile("Output");
        chain->(of);
#endif
```

If you change that 0 to a 1 and set the filename to whatever you prefer, you can build a version that will create the file you specify any time it digests something. Very ugly, but that's all it can do right now.