From: Pratik Datta <pratik.datta@oracle.com>
Subject: **Re: Constrained implementation of exclusive C14N -- waht breaks?**
Date: 23 April 2009 00:09:02 GMT+02:00
To: Thomas Roessler <tlr@w3.org>
Cc: Konrad Lanz <konrad.lanz@iaik.tugraz.at>, XMLSec WG Public List <public-xmlsec@w3.org>
5 Attachments, 10.6 KB

Trying to illustrate graphically.

Case 1: Single complete subtree (can be done with simple algorithm)

In this picture red indicates the tree to be signed. green indicates the ancestors. All the namespace declarations and xml:base declarations from these ancestors need to be considered, that's why I have highlighted it in green. Everything in grey can be ignored by c14n algorithm.

This is the most common scenario, e.g. when signing by ID, and can be easily accomplished by the simplified c14n algorithm

Case 2: Single complete subtree with exclusions of complete subtrees (can also be done with simple algorithm)

This is also pretty common. E.g. an enveloped signature transform excludes the signature subtree. This kind of exclusion is also required in ebxml and some UK goverment signatures. (I had sent out those links before).

Case 3: Multiple complete subtrees (can also be done with simple algorithm)

These are common too, E.g. when you want to sign all instances of <Foo> in the document.

Case 4: Combination of 2 and 3. Multiple subtrees with exclusions of complete subtrees (can be done with simple algorithm too)

This is the maximum level of complexity that I have come across in any practical use cases. In my opinion we should stop here.

Case 5: Subtree with exclusions of subtree and then re inclusion of complete subtree (cannot be done with simple algorithm)

Supporting "re-inclusion" makes the algorithm more complicated, because there are missing ancestors in the middle too, and you need to take care of them.

Case 6: No restriction on element inclusions, but still limits on attributes
If we allow unlimited exlcusions and inclusions of elements, we end up with this.For this case we allow any elements nodes to be included / excluded regardless of position. But we put the following rules on attributes (note these rules are also applicable for cases 1-5)
  a) cannot include an attribute, if its owner element is not in the nodeset
  b) it is possible to exclude regular attributes, but not namespace attributes and not xml:base/lang/space attributes

Case 7: Further generalization with removing restriction on attributes, but still restrictions on namespaces (this is case that the exc C14N algorithm was talking about)
  In this scenario, attributes can exist without their owner elements, and xml: attributes are not treated specially.
However namspace attributes can still not be removed.
This is also the case that Konrad was talking about.

Case 8 : Still more generalization - namespace declarations can be removed, but not "namespace nodes"
Oracle's implementation and Apache implementation currently work with this restriction. With this namespace axis is not expanded out, esoteric XPath expressions that filter out namespace nodes are not allowed. The Y4 test vector in the very first interop had these test cases.

Case 9: Completely generic nodeset
Finally!

The code gets really complex and slow at this point.

Pratik

Thomas Roessler wrote:
> To pick up on the question that Scott asked today, what are the concrete examples of things that break if the constrained implementation of exclusive C14N is used (not just as an optimization, but as the *only* canonicalization available)?
>
> Since you seemed to have some ideas on the call, are there concrete examples that you can give?
>
> http://www.w3.org/TR/xml-exc-c14n/#sec-Implementation
>
> Thanks,
> --
> Thomas Roessler, W3C <tlr@w3.org>
>
> Come to the W3C track unconferences at WWW 2009!
> http://www.w3.org/2009/04/w3c-track.html