Network Working Group
Request for Comments: 3653
Category: Informational

J. Boyer
PureEdge Solutions
M. Hughes
Betrusted, Inc.
J. Reagle
W3C

December 2003

XML-Signature XPath Filter 2.0

Status of this Memo

Copyright Notice

Abstract

XML Signature recommends a standard means for specifying information content to be digitally signed and for representing the resulting digital signatures in XML. Some applications require the ability to specify a subset of a given XML document as the information content to be signed. The XML Signature specification meets this requirement with the XPath transform. However, this transform can be difficult to implement efficiently with existing technologies. This specification defines a new XML Signature transform to facilitate the development of efficient document subsetting implementations that interoperate under similar performance profiles.

This document is the W3C XML Signature XPath-Filter 2.0 Recommendation. This document has been reviewed by W3C Members and other interested parties and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Table of Contents

1. Introduction

The XML Recommendation [XML] specifies the syntax of a class of objects called XML documents. The Namespaces in XML Recommendation [XML-NS] specifies additional syntax and semantics for XML documents. The XML Signature Recommendation [XML-DSig] defines standard means for specifying information content to be digitally signed, including the ability to select a portion of an XML document to be signed using an XPath transform.

This specification describes a new signature filter transform that, like the XPath transform [XML-DSig, section 6.6.3], provides a method for computing a portion of a document to be signed. In the interest of simplifying the creation of efficient implementations, the architecture of this transform is not based on evaluating an [XPath] expression for every node of the XML parse tree (as defined by the [XPath] data model). Instead, a sequence of XPath expressions is used to select the roots of document subtrees -- location sets, in the language of [XPointer] -- which are combined using set intersection, subtraction and union, and then used to filter the input node-set. The principal differences from the XPath transform are:

* A sequence of XPath operations can be executed in a single transform, allowing complex filters to be more easily expressed and optimized.
* The XPath expressions are evaluated against the input document resulting in a set of nodes, instead of being used as a boolean test against each node of the input node-set.

* To increase efficiency, the expansion of a given node to include all nodes having the given node as an ancestor is now implicit so it can be performed by faster means than the evaluation of an XPath expression for each document node.
* The resulting node-sets can be combined using the three fundamental set operations (intersection, subtraction, and union), and then applied as a filter against the input node-set, allowing operations such as signing an entire document except for a specified subset, to be expressed more clearly and efficiently.

As with the original XPath transform, the primary purpose of this transform is to ensure that only specifically defined changes to the input XML document are permitted after the signature is affixed. This can be done by excluding precisely those nodes that are allowed to change once the signature is affixed, and including all other input nodes in the output. It is the responsibility of the signature filter transform author to ensure that nodes are not excluded which could affect the interpretation of the transform output in the application context.

Consider the motivating scenario where an application wishes to affix two enveloped signatures to the document; any other change to the document must cause the signatures to be invalid. When the application creates the first signature that signature is automatically omitted from its own digest calculations. However, it will also be necessary to exclude the subsequent (second) signature element from the digest calculations of the first signature. This specification can be used to efficiently satisfy this requirement using the set subtraction operation.

This transform also supports the ability to specify a set of nodes that will be included in a signature, with all non-specified nodes being excluded. This formulation is useful for isolating a portion of a document, such as a chapter of a document, or a payload in a protocol message, and can be expressed using the set intersection operation.

Complete familiarity with the first XML Signature XPath Transform [XML-DSig, section 6.6.3] is required.

NOTE: Since XPath Filter 2.0 depends on details of XPath, be sure to take into account the XPath Errata at <http://www.w3.org/1999/11/REC-xpath-19991116-errata>.

1.1. Acknowledgements (Informative)

The following people provided valuable feedback that improved the quality of this specification:

* Christian Geuer-Pollmann, Universitat Siegen
* Donald Eastlake, 3rd, Motorola
* Gregor Karlinger, IAK TU Graz
* Aleksey Sanin

1.2. W3C Status

The World Wide Web Consortium Recommendation corresponding to this RFC is at:

http://www.w3.org/TR/xmldsig-filter2/

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [Keywords].

The XPath 1.0 Recommendation [XPath] defines the term node-set as "(an unordered collection of nodes without duplicates)" and specifies a data model for representing an input XML document as a set of nodes of various types (element, attribute, namespace, text, comment, processing instruction, and root).

An input document is the document that contains all the nodes available to processing by this transform. A document subset is a portion of an XML document indicated by an XPath node-set, which may not include all of the nodes in the document. For example, the input node-set is a collection of XPath nodes from the input document that is passed as a parameter to this transform. A subtree rooted by a given node is a document subset containing the given node and every node having the given node as an ancestor. Subtree expansion is the process of expanding a node-set to include all subtrees rooted at any node in the node-set. For example, the subtree expansion of a node-set consisting of just a single element node would be a node-set containing that element, its attribute nodes, namespace nodes, and all its descendants including their attribute nodes and namespaces nodes.

The XML Signature Recommendation [XML-DSig] defines a reference as a sequence of steps performed to obtain an octet stream to be digitally signed. A transform is an identified algorithm to be used as a step

in the reference processing model. A transform takes an octet stream or XPath node-set as input, and it produces an octet stream or XPath node-set as output (the reference processing model automatically converts the final output to an octet stream if it is an XPath node-set).

3. Specification of Signature Filter Transform

The transform operates by computing a node-set that is used to filter the input node-set: The output node-set consists of only those nodes in both the input node-set and the filter node-set. In other words, the output node-set is the intersection of the input node-set and the computed filter node-set.

The filter node-set is computed by evaluating a sequence of XPath expressions and combining their results. A node-set is initially computed containing the entire input document. In sequence, each XPath expression is then evaluated, subtree-expanded, and then used to transform the filter node-set according to a specified set operation; intersection, subtraction, or union. After all XPaths have been applied, the resulting node-set is used as the filter node-set.

3.1. Algorithm Identifier

The XML Signature Recommendation [XML-DSig] uses a [URI] to identify each algorithm to be performed when creating or validating a signature. The signature filter transform is identified as follows:

Algorithm Identifier
http://www.w3.org/2002/06/xmldsig-filter2

3.2. Syntax of Signature Filter Transform

The signature filter transform shall be represented by a sequence of one or more elements named XPath. The content of XPath is character data containing an XPath expression. The XPath has an attribute named Filter whose possible values are intersect, subtract, and union. The Filter attribute indicates the set operation that is performed with the resulting node-set when computing the filter node-set. The following is an example of markup for a signature filter that signs the entire input node-set except for elements with identifier foo and bar (and all nodes with one of those elements as an ancestor):

```
<XPath Filter="subtract"
 xmlns="http://www.w3.org/2002/06/xmldsig-filter2">
   id("foo bar")
```

```
 </XPath>

 Schema Definition:

 <?xml version="1.0" encoding="utf-8"?>
 <!DOCTYPE schema

  PUBLIC "-//W3C//DTD XMLSchema 200102//EN"
         "http://www.w3.org/2001/XMLSchema.dtd"
 [
   <!ATTLIST schema
     xmlns:xf CDATA #FIXED 'http://www.w3.org/2002/06/xmldsig-filter2'>
   <!ENTITY xf 'http://www.w3.org/2002/06/xmldsig-filter2'>
   <!ENTITY % p ''>
   <!ENTITY % s ''>
  ]>

 <schema xmlns="http://www.w3.org/2001/XMLSchema"
         xmlns:xf="http://www.w3.org/2002/06/xmldsig-filter2"
         targetNamespace="http://www.w3.org/2002/06/xmldsig-filter2"
         version="0.1" elementFormDefault="qualified">

 <element name="XPath"
          type="xf:XPathType"/>

 <complexType name="XPathType">
  <simpleContent>
    <extension base="string">
      <attribute name="Filter">
         <simpleType>
           <restriction base="string">
             <enumeration value="intersect"/>
             <enumeration value="subtract"/>
             <enumeration value="union"/>
           </restriction>
         </simpleType>
      </attribute>
    </extension>
  </simpleContent>
 </complexType>

 </schema>

 DTD:

 <!ELEMENT XPath    (#PCDATA) >
 <!ATTLIST XPath
    Filter         (intersect|subtract|union) #REQUIRED >
```

3.3. Input and Evaluation Context of Signature Filter Transform

The input required by this transform is an XPath node-set over the input document. If the input document is an octet stream, then the application MUST convert the octet stream to an XPath node-set that contains all of the document nodes (including comment nodes). The evaluation context for the XPath expressions in the filter transform will be:

* A context node equal to the root node of the document whose node-set was provided as input to this transform. The root node is the parent of the document element and any comment and processing instruction nodes outside of the document element.
* A context position, initialized to 1.
* A context size, initialized to 1.
* A library of functions equal to the function set defined in [XPath] plus a function named here().
* A set of variable bindings. No means for initializing these is defined. Thus, the set of variable bindings used when evaluating the XPath expression is empty, and use of a variable reference in the XPath expression results in an error.
* The set of namespace declarations in scope for the XPath element.

The function here() is defined as follows:

Function: node-set here()

The here() function returns a node-set containing the attribute or processing instruction node or the parent element of the text node that directly bears the XPath expression. In this transform, this will be the XPath element. This expression results in an error if the containing XPath expression does not appear in the same XML document against which the XPath expression is being evaluated.

3.4. Processing Model of Signature Filter Transform

Using the aforementioned evaluation context, the signature filter transform evaluates the XPath expressions appearing in the character content of the XPath elements and uses these to compute a filter node-set F, which is then used to filter the input node-set I resulting in an output node-set O:

* Initialize the filter node-set F to consist of all nodes in the input document.
* Iterate through each XPath expression, X, in sequence, and update the filter node-set F as follows:

- Evaluate the XPath expression X. The result is a node-set S.
- Compute the set S' consisting of all nodes in the input document that are either present in S or that have an ancestor in S. This is equal to the union of all the document subtrees rooted by a node in S.
- If the Filter attribute value is intersect, then compute the intersection of the selected subtrees, S', with the filter node-set F. The result will include only those nodes that are in both the filter node-set and the selected subtrees: F' = F INTERSECT S'.
- If the Filter attribute value is subtract, then compute the subtraction of the selected subtrees, S', from the filter node-set F. The result will include only those nodes that are in the filter node-set, but not the selected subtrees: F' = F - S'.
- Otherwise, if the Filter attribute value is union, then compute the union the selected subtrees, S', with the filter node-set F. The result will include all those nodes that are in either the filter node-set, the selected subtrees, or both: F' = F UNION S'.
- Update the filter node-set F to be the new node-set F'.

* Finally, after applying all the XPath expressions, compute the output node-set O to be the intersection of the computed filter node-set, F, with the input node-set, I. The result will include all nodes from the input node-set that are also in the filter node-set: O = I INTERSECT F.
* An empty input node-set will always result in an empty output node-set.

In this processing model, the conversion from a subtree interpretation of the XPath expressions to a node-set containing all nodes that must be used during the set operation, along with actual performance of the set operation, is described explicitly. Implementors SHOULD observe that it is possible to compute the effective result of this operation in a single pass through the input document without performing subtree expansion or any set operations:

* For each XPath expression X, in sequence, evaluate the expression and store the resulting node-set, S, along with the associated set operation.
* Prepend a node-set consisting of just the document node, along with the operation union.
* Create a new, empty filter node-set.
* Process each node in the input node-set document, adding each node to the output node-set F if a flag Z is true. The flag is computed as follows:

- o Z is true if and only if the node is present in any subtree-expanded union node-set and all subsequent subtree-expanded intersect node-sets but no subsequent subtree-expanded subtract node-sets, or false otherwise. If there are no subsequent intersect or subtract node-sets, then that part of the test is automatically passed.
- o Presence in a subtree-expanded node-set can be efficiently determined without actually expanding the node-set, by simply maintaining a stack or count that identifies whether any nodes from that node-set are an ancestor of the node being processed.

Implementers MAY further observe that, if this transform is followed by a canonicalization operation (e.g., [XML-C14N]), the described filter computation can be efficiently commingled with the document-order canonicalization processing.

4. Examples of Signature Filter Transform

The example below illustrates one way to create an enveloped signature with the signature filter transform. The function here() identifies the XPath element, and the subsequent location path obtains the nearest ancestor Signature element. Due to the subtract value of the Filter attribute, the output of the signature filter transform is a node-set containing every node from the input node-set except the nodes in the subtree rooted by the Signature element containing the example signature filter transform below.

```
<XPath Filter="subtract"
 xmlns="http://www.w3.org/2002/06/xmldsig-filter2"
 xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
   here()/ancestor::dsig:Signature[1] </XPath>
```

A suitable signature reference URI to use with this subtract filter would be URI="" (the entire signature document, without comments), URI="#xpointer(/)" (the entire signature document, with comments) or any same-document reference that includes the signature itself.

An example of an intersect filter is a signature that co-signs another signature. In this example, a Signature element identified by PrimaryBorrowSig must be signed. The XPath expression obtains the element node, and the transform expands the output node-set to contain all nodes from the input node-set that are also in the subtree rooted by the element node.

```
<XPath Filter="intersect"
 xmlns="http://www.w3.org/2002/06/xmldsig-filter2">
   id("PrimaryBorrowerSig") </XPath>
```

This type of intersect filter is useful for efficiently signing subsets of a document, whether this is the same document as the signature or an external document. For example, if the signature reference URI is URI="document.xml", then this document will be automatically parsed and just the identified element and its descendants will be signed.

Union filters, by themselves are of no particular use: The initial filter node-set consists of the entire input document; any union with this will have no effect, so the output of the transform will be identical to the input. The union operation is intended to follow a subtract operation, to allow a subtree to be removed, with the exception of a lower subtree which is still included in the output.

Consider the following document which contains a same-document enveloped signature reference with an XPath filter containing three

XPath operations:

```
<Document>
  <ToBeSigned>
    <!-- comment -->
    <Data />
    <NotToBeSigned>
      <ReallyToBeSigned>
        <!-- comment -->
        <Data />
      </ReallyToBeSigned>
    </NotToBeSigned>
  </ToBeSigned>
  <ToBeSigned>
    <Data />
    <NotToBeSigned>
      <Data />
    </NotToBeSigned>
  </ToBeSigned>
  <dsig:Signature
   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
   xmlns:dsig-xpath="http://www.w3.org/2002/06/xmldsig-filter2">
    <dsig:SignedInfo>
      ...
      <dsig:Reference URI="">
        <dsig:Transforms>
          <dsig:Transform
           Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
            <dsig-xpath:XPath
             Filter="intersect"> //ToBeSigned </dsig-xpath:XPath>
            <dsig-xpath:XPath
```

```
                Filter="subtract"> //NotToBeSigned </dsig-xpath:XPath>
              <dsig-xpath:XPath
                Filter="union"> //ReallyToBeSigned </dsig-xpath:XPath>
            </dsig:Transform>
          </dsig:Transforms>
          ...
        </dsig:Reference>
      </dsig:SignedInfo>
      ...
    </dsig:Signature> </Document>
```

The intersect operation computes the intersection of the XPath-selected subtrees with the filter node-set. In this case, the filter node-set initially contains the entire input document, and the XPath expression evaluates to the two ToBeSigned elements; these are expanded to include all their descendents and intersected with the filter node-set, resulting in the following:

```
  <ToBeSigned>
    <!-- comment -->
    <Data />
    <NotToBeSigned>
      <ReallyToBeSigned>
        <!-- comment -->
        <Data />
      </ReallyToBeSigned>
    </NotToBeSigned>
  </ToBeSigned><ToBeSigned>
    <Data />
    <NotToBeSigned>
      <Data />
    </NotToBeSigned>
  </ToBeSigned>
```

The subtract filter computes the subtraction of the XPath-selected subtrees from the filter node-set. In this case, the XPath expression evaluates to the two NotToBeSigned elements; these are expanded to include all their descendents and subtracted from the filter node-set:

```
<ToBeSigned>
    <!-- comment -->
    <Data />

  </ToBeSigned><ToBeSigned>
    <Data />

</ToBeSigned>
```

Next, the union filter computes the union of the XPath-selected subtrees with the filter node-set. In this case, the XPath expression evaluates to the ReallyToBeSigned element; this is expanded to include all its descendents and added to the filter node-set:

```
<ToBeSigned>
    <!-- comment -->
    <Data />
    <ReallyToBeSigned>
        <!-- comment -->
        <Data />
      </ReallyToBeSigned>
  </ToBeSigned><ToBeSigned>
    <Data />

</ToBeSigned>
```

Finally, this resulting filter node-set is used to transform the input node-set. In this example, the input node-set is the entire document, with comments removed. The transformed node-set will thus be all those nodes from the input document, less comments, that are also in the filter node-set:

```
<ToBeSigned>

    <Data />
    <ReallyToBeSigned>

        <Data />
      </ReallyToBeSigned>
  </ToBeSigned><ToBeSigned>
    <Data />

</ToBeSigned>
```

Note that the result contains no nodes that were not in the input node-set. Although the filter node-set included comments, these were not present in the input node-set so they are not present in the output node-set.

This signature filter does not provide any increased capability over the original XPath transform. For example, this reference could be replicated using the XPath transform as follows.

```
<dsig:Reference URI="">
  <dsig:Transforms>
    <dsig:Transform
```

```
       Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
        <dsig:XPath>
          (ancestor-or-self::ToBeSigned and
           not (ancestor-or-self::NotToBeSigned))
          or ancestor-or-self::ReallyToBeSigned
        </dsig:XPath>
      </dsig:Transform>
    </dsig:Transforms>
    ...  </dsig:Reference>
```

The advantage of the signature filter transform over the XPath transform is that the latter requires evaluation of a potentially-complex expression against every node in the input set, which has proved costly in practice for many useful operations. This specification's filter requires evaluation of simple XPath expressions and then the execution of some basic set operations or their equivalent, which can be implemented significantly more efficiently.

5. Normative References

[Keywords] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[URI] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[XML] "Extensible Markup Language (XML) 1.0 (Second Edition)", T. Bray, E. Maler, J. Paoli, and C. M. Sperberg-McQueen. W3C Recommendation, October 2000. Available at <http://www.w3.org/TR/2000/REC-xml-20001006>.

[XML-C14N] Boyer, J., "Canonical XML", RFC 3076, March 2001. Also a W3C Recommendation available at <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.

[XML-DSig] Eastlake, J., Reagle, J. and D. Solo, "XML-Signature Syntax and Processing", RFC 3275, March 2002. Also a W3C Recommendation available at <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>.

[XML-NS] "Namespaces in XML", T. Bray, D. Hollander, and A. Layman. W3C Recommendation, January 1999. Available at <http://www.w3.org/TR/1999/REC-xml-names-19990114/>.

[XPath] "XML Path Language (XPath) Version 1.0", J. Clark and S. DeRose. W3C Recommendation, November 1999. Available at <http://www.w3.org/TR/1999/REC-xpath-19991116>. (Note also XPath Errata at <http://www.w3.org/1999/11/REC-xpath-19991116-errata>.)

[XPointer] "XML Pointer Language (XPointer)", S. DeRose, R. Daniel, and E. Maler. W3C Candidate Recommendation, January 2001. Available at <http://www.w3.org/TR/2001/CR-xptr-20010911/>.

6. Authors' Addresses

John Boyer
PureEdge Solutions Inc.
4396 West Saanich Rd.
Victoria, BC, Canada V8Z 3E9

Phone: +1-888-517-2675
EMail: jboyer@PureEdge.com

Merlin Hughes
Betrusted, Inc.
11000 Broken Land Parkway Suite 900
Columbia, MD 21044

Phone: +1-443-367-7000
EMail: Merlin.Hughes@betrusted.com

Joseph M. Reagle Jr., W3C
Massachusetts Institute of Technology
Laboratory for Computer Science
NE43-350, 545 Technology Square
Cambridge, MA 02139

Phone: +1.617.258.7621
EMail: reagle@mit.edu

7. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.