XML signatures: Behind the curtain

Who can be trusted with authentication?

Larry Loeb (larryloeb@prodigy.net)
Author, Secure Electronic Transactions
December 2001

The XML Digital Signature Standard establishes how XML can functionally sign itself over an insecure network like the Internet. While this effort does not require an established PKI to function, it may require the use of trusted XML servers for authentication. Consequently, each enterprise will have to evaluate the potential security risk of outsourcing this increasingly critical business function.

Contents:

Introduction

There's a W3C candidate out for XML signatures that looks fairly close to being the final one, though it is still a work in progress and has not as of this writing officially advanced to the Draft Standard stage (see Resources). It's also been known to the Internet Engineering Task Force (IETF) as RFC 3075. Judging by the author list for this candidate -- which includes folks from the W3, MIT, and Microsoft -- it's clear that the Internet industry is taking this subject seriously.

The overview

XML signatures have been designed (according to the RFC) with the multiple goals of providing "integrity, message authentication, and/or signer authentication services for data of any type, **whether located within the XML that includes the signature or elsewhere.**" (I've bolded that last phrase because it is central to what this candidate implies for how the 'Net would work if it is adopted and implemented. More on this later.) These are fairly ambitious goals to be sure, and fairly extensive if considered in context. These signatures and their associated processes have as an ultimate goal providing the default basic server-based security services for the Web through the use of XML.

However, the authors do have some sense of proportion about their work. The candidate contains this passage: "The XML Signature ... does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed. Consequently, while this specification is an important component of secure XML applications, it is, by itself, not sufficient to address all application security/trust concerns, particularly with respect to using signed XML (or other data formats) as a basis of human-to-human communication and agreement. Such an application must specify additional key, algorithm, processing and rendering requirements." In short, the authors are cautioning against considering this work as a technical panacea; that it must be used within other security measures. This is wise, but begs the question of what's behind the XML curtain.

What they don't tell you in the specification

There is also an implicit assumption here, and it's in-your-face obvious upon due examination. XML is Web- and server-based. It's a sort of thin-client approach, as opposed to the "damn the bandwidth, load the servlet" approach of Java. XML signatures will therefore use the Web (via URLs in the XML) to find out the method to encode or decode things. Exactly how it uses the Web can be syntactically specified (as demonstrated in the technical discussion that follows), but that kind of discussion ignores the meta-question of what URL to use. That is, if the signature needs 'Net resources to complete its action, then who will supply those resources? If this sort of signature becomes a widely-used "semaphore" for authentication of messages and transactions, whichever URLs are pointed to in the XML code can become a de facto standard for authentication services.

But wait, there's more. The real use of authentication services will most likely be found in Web shopping; rather than, for example, secured e-mail. A merchant will want to know that he is receiving a valid order from a customer, and so will insist on some form of acceptable authentication. This is the sort of problem that was first solved by Electronic Data Interchange (EDI)

systems that used secured networks to operate. Over the insecure Internet, the same previously-solved operational problems remain open. The XML signature process is meant to address this in a sweeping way, but that sweep is part of the problem. This little-known candidate carries in it the seeds of a method for some company to monopolize Internet authentication services.

Let me posit a scenario. Imagine that some Commercial Off The Shelf (COTS) software company decides it wants to be the authorization service that all must pass through to get things done. Also further posit that this COTS proprietary operating system is found on the majority of the deployed systems under consideration. The COTS vendor then brings out an Xtra Proprietary version of its OS, which contains a client that uses XML signatures (and *only* XML signatures) for security. All the XML code points to this vendor's servers to provide information, sort of like a non-public PKI infrastructure. This client becomes widely used for authentication simply because it's easy to set up as a default since the OS is widely used. Also, said manufacturer embeds it in its popular "free" browser as the default security mechanism.

To put the icing on the cake, say this vendor starts to charge someone (perhaps the merchant and, indirectly, the consumer) every time its servers are used for authentication by this client. *Voila!* Monopoly and revenue streams all in one.

Now, having used these security services in good faith, a merchant might find that these services don't actually address underlying problem -- that of "chargebacks" from the credit card companies. These chargeback issues -- where people acknowledge that they had an agreement or some contact with the merchant but claim that the merchant did not performed in some or all aspects as promised (like nothing delivered, or the order was cancelled, or some other reason) -- are part of the credit card issuer's business model and beyond the scope of an XML signature specification. Indeed, in the US the Fair Credit Billing Act (15 U.S.C. 1666-1666j) preserves the customer's right to make arguments with an issuing bank about the correctness of a bill (including amount, computation, timing, and delivery/receipt of goods/services). Those rights can't be waived by contract.

I'm coming to the windup here.

So, even if schemes like XML signatures are in place and being used, they may not address the real world problems they were supposed to solve. Given that, and the potential for abuse by greedy companies to establish a 'Net monopoly, use of this technology may end up unsatisfying. That doesn't mean that a developer may have to, in the long run, use this sort of scheme just to be compatible; but don't think that it's just a magic incantation, either. Companies other than the largest ones may develop unique authentication services, and using them may require small rewrites in the XML code. The true problem may be the unquestioning acceptance of manufacturer-supplied XML signature code. So let's take a look at the tech specification with an eye towards understanding where things may need to be changed as services evolve.

The geek part

Note that in the examples that follow, it's OK to substitute in your mind the URL of some monopolistic software vendor in place of the W3C addresses. It'll make it seem more real. The following discussion borrows heavily from the candidate specifications, since they are vendor-neutral.

The first useful property of an XML signature is that it can be applied to any sort of digital content (sometimes called a data object), including XML. An XML signature may be applied to the content of one or more resources. Enveloped signatures are performed over data within the same XML document as the signature, so a detached signature is over data that is external to the signature's element itself. More specifically, the current XML signature specification defines an XML signature element type and an XML signature application; conformance requirements for each are specified in the document by way of schema definitions and prose, respectively.

Signature elements

Take a look at the signature element. The data objects are first digested (a digest is fixed-length representation of a variable length data object and is created using an algorithm like SHA-1) and the resulting value is placed in an element (with other information). This element is then digested and cryptographically signed. The signature element has the following structure (where "?" denotes zero or one occurrence, "+" denotes one or more occurrences, and "*" denotes zero or more occurrences):

Listing 1

```
<Signature>
  <SignedInfo>
    (CanonicalizationMethod)
    (SignatureMethod)
    (<Reference (URI=)? >
      (Transforms)?
      (DigestMethod)
      (DigestValue)
    </Reference>)+
  </SignedInfo>
  (SignatureValue)
 (KeyInfo)?
 (Object)*
</Signature>
```

An example to mull over
Consider a simple example with some real data. The following is the detached signature of the content of the HTML4 in XML specification. The XML is given first, followed by the annotations for each individually listed line of code. It's also assumed in this discussion that you've had some experience with private/public key cryptologic methods and are comfortable with the concepts. If not, there are excellent introductory articles here on developerWorks that can be perused. (See Resources.)

Listing 2

```
[s01] <Signature Id="MyFirstSignature" xmlns=
                    "http://www.w3.org/2000/09/xmldsig#">
[s02]   <SignedInfo>
[s03]   <CanonicalizationMethod Algorithm=
                    "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s04]   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]   <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
[s06]     <Transforms>
[s07]       <Transform Algorithm=
                    "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[s08]     </Transforms>
[s09]     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s10]     <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
[s11]   </Reference>
[s12]   </SignedInfo>
[s13]   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
[s14]   <KeyInfo>
[s15a]    <KeyValue>
[s15b]      <DSAKeyValue>
[s15c]        <p>...</p><Q>...</Q><G>...</G><Y>...</Y>
[s15d]      </DSAKeyValue>
[s15e]    </KeyValue>
[s16]   </KeyInfo>
[s17] </Signature>
```

[s02-12] The required SignedInfo element is the information that is actually signed. Core validation of SignedInfo consists of two mandatory processes: validation of the signature over SignedInfo and validation of each Reference digest within SignedInfo. The algorithms used in calculating the SignatureValue are also included in the signed information while the SignatureValue element is outside SignedInfo.

[s03] The CanonicalizationMethod identifies the algorithm that is used to canonicalize the SignedInfo element before it is digested as part of the signature operation. Canonicalization is how the process deals with differing data streams that can be

contained inside the same data element For instance, two differing ways can be contained to represent text. Canonicalization is the method in which raw data is interpreted to have spaces displayed as spaces and not as ASCII code.

[s04] The SignatureMethod is the algorithm that is used to convert the canonicalized SignedInfo into the SignatureValue. It is a combination of a digest algorithm, a key dependent algorithm, and possibly other algorithms. The algorithm names are signed to resist attacks based on the substitution of a weaker algorithm. To promote application interoperability, the candidate specifies a set of signature algorithms that are required to be implemented, though their use is at the discretion of the signature creator.

[s05-11] Each Reference element includes the digest method and resulting digest value calculated over the identified data object. It also may include transformations that produced the input to the digest operation. A data object is signed by computing its digest value and a signature over that value. The signature is later checked via reference and signature validation that recreates the digest value and makes sure that it matches what is held in the data object.

[s05] The optional URI attribute of Reference identifies the data object to be signed. This attribute may be omitted on at most one Reference in a Signature. (This limitation is imposed in order to ensure that references and objects may be matched unambiguously.)

[s05-08] This identification, along with the transforms, is a description provided by the signer on how they obtained the signed data object in the form it was digested (that is, the digested content). The verifier may also obtain the digested content in another method so long as the digest verifies it.

[s06-08] Transforms is an optional ordered list of processing steps that were applied to the resource's content before it was digested. This is the trail that needs to be followed for decryption. Transforms can include operations such as canonicalization, encoding/decoding (including compression/inflation), XSLT, and XPath. XPath transforms are a little tricky because they permit the signer to derive an XML document that omits portions of the source document, limiting the XML tree, as it were. The excluded portions can therefore change without affecting the signature validity. If no Transforms element is present, the resource's content is digested directly. It should be remembered that user-specified transforms are permitted even though a basic default set is specified in the candidate.

[s09-10] DigestMethod is the algorithm applied to the data after Transforms is applied (if it has been specified) to yield the DigestValue. The signing of the DigestValue is the mechanism that binds a resource's content to the signer's key.

[s14-16] KeyInfo indicates the key to be used to validate the signature. Identification mechanisms can include certificates, key names, and key agreement algorithms. KeyInfo is optional for two reasons. First, the signer may not wish to reveal any key information to all of the document processing parties. Why tell the world all the time? Second, the information may be known within the application's context and need not be represented explicitly. Since KeyInfo is outside of SignedInfo, if the signer wishes to bind the keying information to the signature, a Reference can easily identify and include the KeyInfo as part of the signature.

A pithy summary

XML is codification of author Donald Knuth's aphorism that "all computer problems can be solved with another layer of redirection." The whole XML syntax is designed to utilize redirected Web-based services. While outsourcing critical business services to trusted partners may be acceptable, outsourcing by default what could be a significant component of e-business in the years to come doesn't seem such a good idea.

Also, it must be stressed that understanding the context of your XML use (what data is actually being signed) is just as important to security analysis as the actual signing of the code itself. Any default use by unintended or unknown redirection to someone else's business model of Web services can end up being an open and insecure -- not to mention potentially expensive -- portal into an organization. Knowing where you're really going on the Web (as well as who is sending you there!) seems to be a prudent course of action for these times.

Resources

- Read the W3C Recommendation, "XML Schema Part 1: Structures" (D. Beech, M. Maloney, N. Mendelsohn, H. Thompson. May 2001.), which specifies how the XML Schema definition language offers facilities for describing the structure and constraining the contents of XML 1.0 documents.
- "XML Schema Part 2: Datatypes" (P. Biron, A. Malhotra. May 2001.) is the second part of the specification of the XML Schema language. It defines facilities for defining datatypes to be used in XML schemas as well as other XML specifications.
- RFC 2807 lists the design principles, scope, and requirements for the XML Digital Signature specification. It includes requirements as they relate to the signature syntax, data model, format, cryptographic processing, and external

requirements and coordination.

- The W3C Working Draft "XML-Signature Requirements" (J. Reagle. April 2000.) lists the design principles, scope, and requirements for the XML Digital Signature specification.
- Public comments on the standard should be sent to w3c-ietf-xmldsig@w3.org, the appropriate mailing list.
- For an introduction to XML encryption and XML signature, take a look at Enabling XML security by Murdoch Mactaggart, here on developerWorks.
- For the basics on Public Key Infrastructure, read Joe Rudich's developerWorks article PKI: A primer
- Of course, you can find a wide range of security articles in the developerWorks Security topic, and our XML zone is always an excellent resource for developers working with the language.
- The XML Security Suite, available on alphaWorks, provides security features such as digital signature, encryption, and access control for XML documents.
- IBM security services can help you determine what your risks are, and then design a security program to address them.
- RFC 3075, XML-Signature Syntax and Processing is the IETF's specification for the candidate.

About the author

Larry Loeb has written for many of the last century's major "dead tree" computer magazines, having been -- among other things -- a consulting editor for BYTE magazine and senior editor for the launch of WebWeek. He's been online since uucp "bang" addressing (where the world existed relative to !decvax), serving as editor of the Macintosh Exchange on BIX, and the VARBusiness Exchange. He's also written a book on the Secure Electronic Transaction Internet protocol. His first Mac had 128K of memory. His first 1130 had 4K, as did his first 1401. You can e-mail him at larryloeb@prodigy.net.