lecture

Lecture on

XML Security

How to secure XML documents and communications

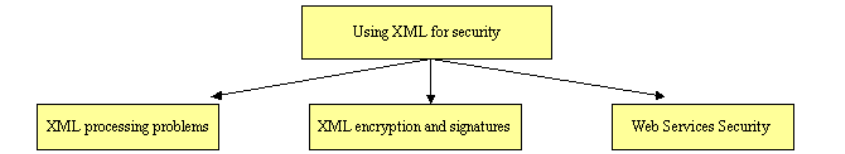
Walter Kriha

Goals

- Show XML related security problems and opportunities
- Show element-based encryption and authentication, partial encryption etc.
- Show how digital signatures work with XML.
- Get an understanding of canonicalization of formats. Canonical XML (like DER/BER in asn.1)
- Discuss security problems with XML processing of entities etc.

Finally: Prepare us for the new Web Services Security proposals which will at least partially rely on basic XML security mechanisms.

Overview



- Logical vs. physical validity
- Are XSL scripts code?
- Can entities be used to steal information?
- DOS attacks using entities

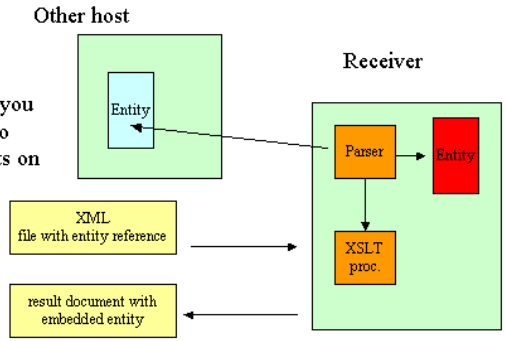
- Create canonical XML documents
- Sign XML documents or fragments
- Encrypt XML documents or fragments
- Multiple signatures

- Secure request through intermediates
- Implementation independent security
- Implicit (middleware) vs. explicit (document) base security

Web Services Security will be handled as a separate part. First we need to understand XML security issues.

Malicious documents?

If you offer a rendering service you might be abused to create artificial hits on some host.



Does your XML processing system check the URIs of entity references BEFORE accessing them?

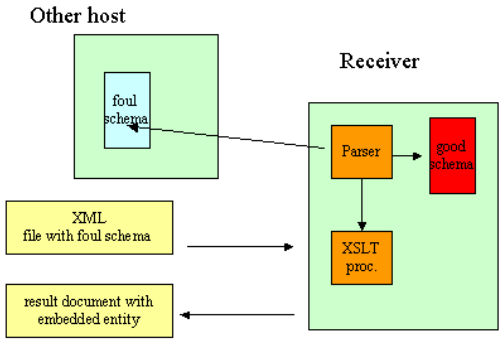
XML has some mechanisms that pose security problems by themselves – e.g. entities which are referenced automatically by a parser and which could be used to create denial-of-service attacks through the construction of a large number of those references. Or worse: those references could point anywhere on the target server and might pull secret information from such a server. Those problems are NOT the main focus of this lecture but they remind us on common vulnerabilities. Both examples have been taken from the XML-DEV mailing list (Miles Sabin, R.Tobin)

Extension Functions in XSLT

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform version='1.0'>
<xsl:output method="html„ encoding="ISO-8859-1„ indent="no"/>
<!-- ================================================================= -->
<xsl:script language=„java" implements-prefix=„sy" src=„java:java.util.system"/>
<xsl:template match="*">
  <xsl:message>
    <xsl:text>No template matches </xsl:text>
    <xsl:value-of select=„sy:exec()"/>
    <xsl:text>.</xsl:text>
  </xsl:message>
```

Calling extension functions from XSLT is easy. Several language bindings are supported (java, javascript etc.). What userid and rights is your XSLT processor using when you do server side processing of requests? (M.Kay, XSLT 2nd edition, page 568ff.)

Suppressing Validation



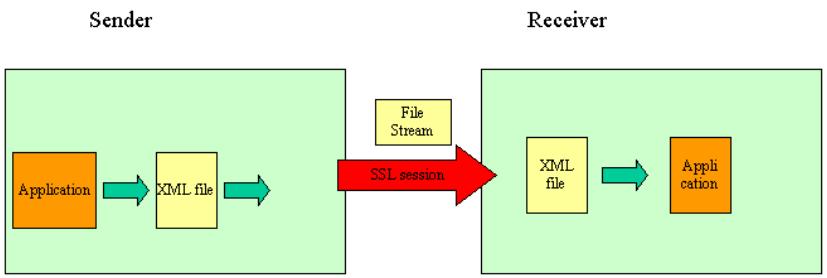
James Clark mentioned recently an especially evil way to work around validation: „Suppose an application is trying to use validation to protect itself from bad input. It carefully loads the schema cache with the namespaces it knows about, and calls validate(). Now the bad guy comes along and uses a root element from some other namespace and uses xsi:schemaLocation to point to his own schema that that has a declaration for that element and uses <xs:any namespace="##any„ processContents="skip"/>. Won't they just have almost completely undermined any protection that was supposed to come from validation?"

Mechanisms and Technologies

- XML Digital Signatures
- XML Encryption
- related XML basic standards (XML Infoset etc.)
- WS-Security
- Secure Association Markup Language
- SOAP and WSDL

We will see how all these technologies are needed to solve the security problems caused by the new internet based, distributed and collaborative business model of web services. But first a look at XML processing of documents is in order.

Sending XML Securely (Today)



Today the easiest solution to send an XML file securely (with authentication, integrity and confidentiality provided by the transport-level protocol) is to use SSL/TLS. There are a number of disadvantages associated with this solution:

Problems using a transport-level protocol

There are a number of disadvantages associated with the SSL/TLS solution:

-Security is provided by runtime code (SSL middleware etc.) NOT tied to the document itself. If the document is forwarded to another receiver its security is depending on the new security context.

-The receiver does not have non-repudiation: no signature attached. If it were, how would we communicate the keys etc. used for it to our receiver?

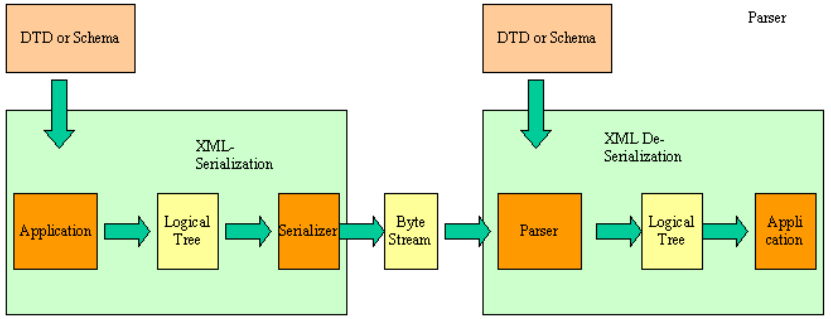
-Worse yet: how would the signer know what the receiver is able to understand and process? Same problem with encryption

-Encryption of parts of the document is possible but there is no mechanism to create several signatures and encrypted blocks for multi-party document exchange.

-If the document is encrypted itself, how do we tell the receiver which mechanism has been used? How do we transport the proper keys (if needed)?

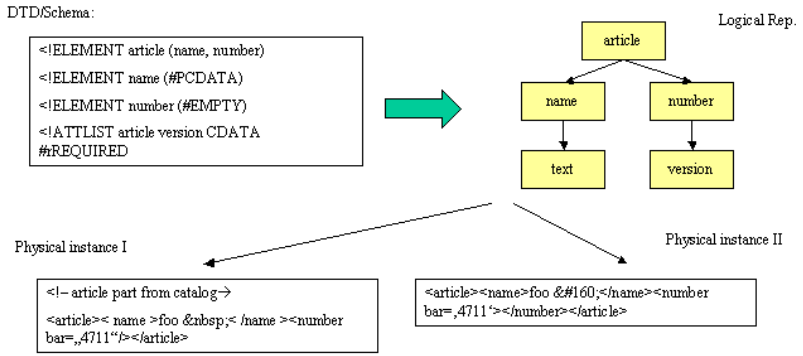
These problems are interestingly pretty much the same as for secure e-mail. They are caused by the same reason: using something that is SESSION oriented to transport single MESSAGES or DOCUMENTS. Eric Rescorla shows the problems with SSL when used for to secure e-mail. His „SMTP over TLS“ chapter sets the stage for most of the things in this lecture. Surprisingly Web Services seem to fall much more into the message/document model than the connection oriented model. Solutions for messages/documents are usually closer to the application (end-to-end argument in security). The latest security related proposals from the Web Services Industry seem to confirm this trend.

Sending and receiving XML documents



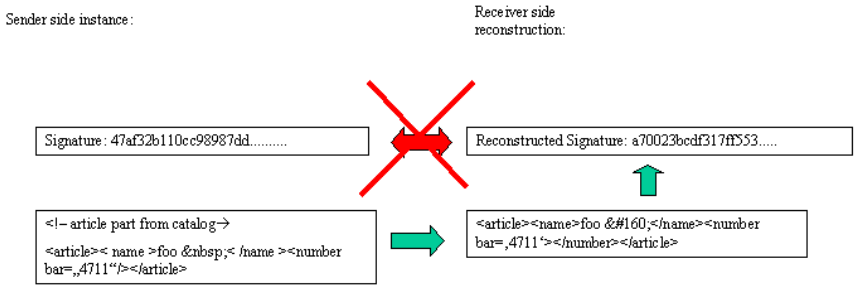
Both sender and receiver create or validate an xml instance using a schema or DTD which controls the LOGICAL content of the xml file. Different physical content can result in the SAME logical content. Unfortunately signatures e.g. work on the PHYSICAL content of an XML instance. Since serializers and parsers have considerable freedom with respect to physical content this means that a signature created over physical representation 1 (sender) may not fit to the physical representation 2 (receiver) re-created by the parser even though the logical content is the same: Signatures work on bit-level, not on XML element level (This is comparable to the C++ concept of „const“ methods which guarantee BITWISE constness of an object: you cannot even cash something in a const method)

Logical vs. Physical Representation



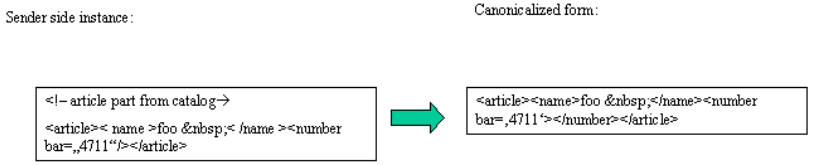
Watch the small differences in instances: whitespace in element names, character entities vs. character codes, special „empty“ syntax for number or not, whitespace in attributes, double quotes vs. single quotes etc. Please note: BOTH instances are a valid representation of the DTD or Schema because they both fit to the logical model above. For most applications the differences will not matter. But they will definitely matter if signatures over those representations are created. But XML itself has problems with it too as we will see.

Signatures over XML Instances



Once the signature is reconstructed on the receiver side it does not fit to the originally created signature – due to the differences in physical representation that serializer and parser used. It does not matter that the logical content is exactly the same.

Canonicalization of XML Instances



Canonical XML defines how a canonical instance needs to look like:

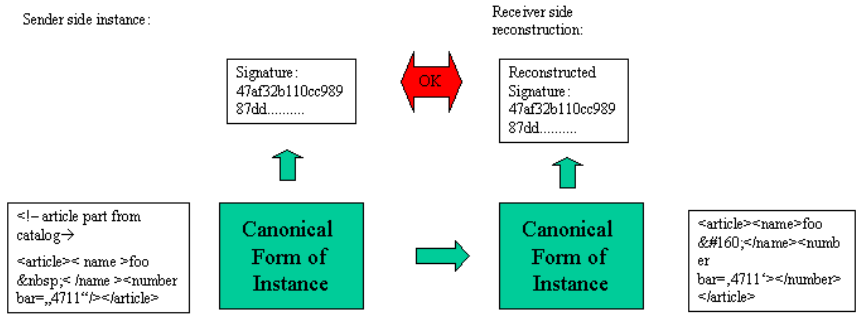
-UTF-8 encoding, line breaks normalized to #xA, attribute values normalized

-character references expanded, CDATA replaced with content, DTD and XML declaration removed, empty tags (<e/>) replaced with tag pairs (<e></e>)

-special characters replaced with character references, redundant namespaces removed, fixed attributes expanded, sorted according to defined order for attributes and namespaces

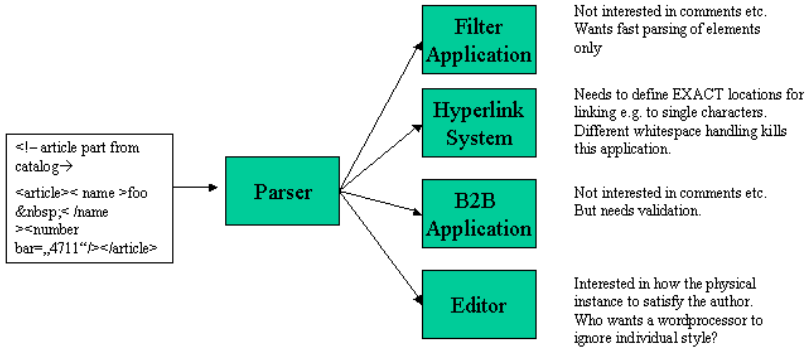
-(from Michael Kay, XSLT 2nd edition, pg. 71)

Signatures over canonical XML Instances



Signatures are constructed and compared based on the CANONICAL form of the instance.

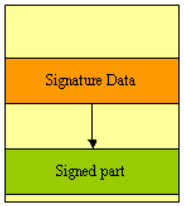
Off Topic: Property Sets, Groves, XML-Info Set



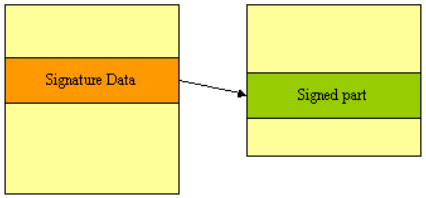
The linking (HyTime) problem made the old SGML folks realize that every application needed something different from a document via the parser and that one size would NOT fit all. They defined so called property sets where one could describe all things in a document which mattered and applications could say: Parser, give me the document X but respect property set Y in doing so. Cross document links now could be made reliable because a property set could be given which „canonicalized“ the document to make the link targets fit to the expectations of the locator. The XML – Info set will provide similiar features for XML. Notice that DOM made the mistake and tried to be everything for everybody.

Are we done with XML signatures now?

XML instance with ENVELOPING signature

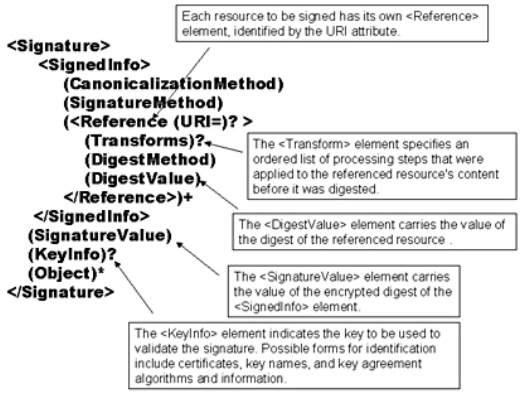


XML instance with DETACHED signature



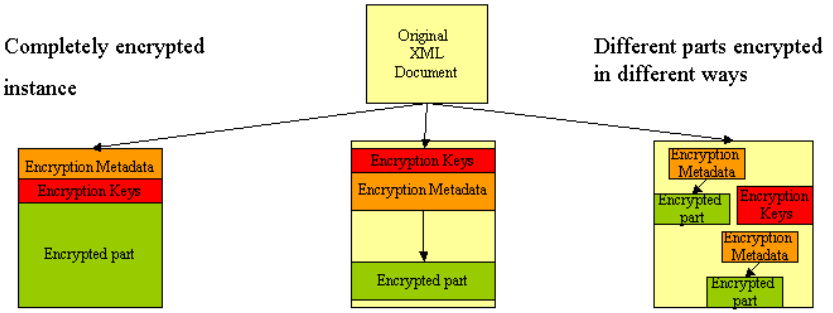
We still need to distinguish how we sign XML parts that are in different XML instances, how we apply several signatures to the same part (which might possibly be already encrypted and needs decryption before signing) etc. XML DSIG (www.w3.org/Signature)

The XML DSIG „Signature“ Element



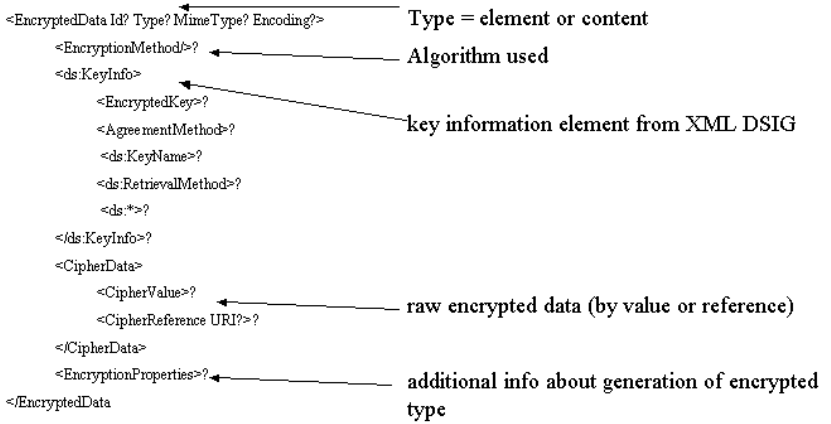
From Ed Simon et.al, (see Resources). Note that „object“ will only be there if the signature is „enveloping“ otherwise the reference element will point with the URI to an out-of-document object. Transforms defines e.g. that the object has been canonicalized. Information that the receiver needs for verification is contained in the DigestMethod, SignatureValue and possibly also in the KeyInfo element (e.g. which public key was used to sign the disgest)

Encrypting XML documents



Especially in a multi-party communication system encryption is difficult to realize. The core problem is how to authorize and control the viewing of different parts by different parties. There is also the problem of known plain-text attacks if the tags are well known because the DTD is known.

The EncryptedData Element



EncryptedData element which contains (via one of its children's content) or identifies (via a URI reference) the cipher data. When encrypting an XML element or element content the EncryptedData element replaces the element or conten (respectively) in the encrypted version of the XML document. (from XML Encryption spec. http://www.w3.org/Encryption/2001/Drafts/xmlenc-core/

Coding Example of XML Encryption

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
 <Name>John Smith</Name>
 <CreditCard Limit='5,000' Currency='USD'>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#
  Type='http://www.w3.org/2001/04/xmlenc#Content'>
   <CipherData>
    <CipherValue>A23B45C56</CipherValue>
   </CipherData>
  </EncryptedData>
 </CreditCard>
</PaymentInfo>
```

In this example form the XML encryption specification only the CONTENT of the credit card information has been encrypted and is enclosed in the CipherValue element. The specification also defines rule about the relation between encryption and signatures, e.g. in which order they should be applied. When data is encrypted, any digest or signature over that data should be encrypted as well to avoid guessing attacks.

Off-Topic: XML Namespaces

```
<schema xmlns='http://www.w3.org/2001/XMLSchema' version='1.0'
        xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
        xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
        targetNamespace='http://www.w3.org/2001/04/xmlenc#
        elementFormDefault='qualified'>
    <import namespace='http://www.w3.org/2000/09/xmldsig#'
      schemaLocation='http://www.w3.org/TR/2002/REC-xmldsig-
core-20020212/xmldsig-core-schema.xsd'/>
```

namespace used to denote a schema and how instance and schemas are related

```
http://www.w3.org/2001/04/xmlenc#tripledes-cbc
```

namespace used to define different encryption algorithms

```
<ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
 <pay:PaymentInfo xmlns:pay='http://example.org/paymentv2'>
<dummy xmlns="http://example.org/"
        xmlns:foo="http://example.org/foo"><One><foo:Two/></One></dummy>
```

namespace used within instances to avoid name clashes between elements of different schemas

Despite an ongoing discussion about their value, namespaces are increasingly used to denote all kinds of things. If you want to work with XML you will need to understand namespaces. Important: There is absolutely NO requirement that a namespace URI really points to a web resource. In most cases the URI is just used to make definitions unique (basically by using the DNS name system which already has unique names)

Are we done with signatures and encryption?

Please note that we still have other unsolved problems. Our view right now was very static and document centric. In a more message oriented environment one has e.g. to solve the problem of security context negotiation

-what kind of security and encryption is required by the provider of a service?

-How do potential requester know about those requirements?

-How do we establish initial trust?

For answers on those questions see the lecture on „Web Services Security“

Resources (1)

- Murdoch Mactaggart, Enabling XML security – an introduction to XML encryption and XML signature. If you are too lazy to read the original specs from w3c, at least read these 6 pages. Excellent introduction and easy to read too. Shows you with pieces of xml how to sign or encrypt parts of xml documents or messages. Not SOAP related. http://www-106.ibm.com/developerworks/xml/library/s-xmlsec.html/index.html
- An Introduction to XML Digital Signatures, By Ed Simon, Paul Madsen, Carlisle Adams http://www.xml.com/lpt/a/2001/08/08/xmldsig.html . Good and short. Shows the <signature> element and children of it clearly.
- www.w3.org/Signature, www.w3.org/Encryption . Find the latest specifications here.
- Michael Kay, XSLT 2nd edition for a real good introduction to XSLT and extensions.

Resources (2)

- Steve DeRose, David Durand, Making Hypermedia work. A good introduction to HyTime, the SGML based hypermedia architecture. If you want to understand what computer science really is about: Naming, addressing, linking, get this book.
- Eliot Kimber, Practical HyTime. Eliot sent this out as a draft but never finished it. VERY good. Explains the concept of an „enabling architecture“ by giving us the logical structures necessary for naming, addressing and linking. If you want to get into Topic maps etc., get these books first. I learnt more from these HyTime books than I did from reading most other computer science literature.
- Paul Prescott on Groves, Property Sets etc. Paul wrote a number of very good articles about the concept of Property Sets. I always wondered how e.g. LDAP models are somehow related to property sets and nodes???

Resources (3)

- Uche Ogbuji, Use XML namespaces with care. Some excellent info on how to use namespaces. Starts with basic principles and explains how namespaces work. Short and good. from developerworks.