Complexity as the Enemy of Security

Position Paper for W3C Workshop on Next Steps for XML Signature and XML Encryption
25/26 September 2007 – Mountain View, California
Brad Hill

Introduction

The XML Signature and XML Encryption specifications present very complex interfaces suitable for general purpose use in almost any situation requiring privacy or integrity. These technologies are quickly becoming the foundation for security in the service-oriented software world. They must be robust, predictable and trustworthy. As specified, they are not. It is possible to create and operate these technologies with a secure subset of the defined functionality, but many implementing vendors are not. Evidence of continued vulnerabilities in multiple implementations is evidence that the root cause must be addressed: the base specifications must be improved by the W3C, including a "self-contained and secure" profile, and additions are needed to the security considerations section to give a more frank and detailed discussion of the security issues in implementing and using these technologies.

Some of this advice and information has found its way into the specifications for SAML, for WS-Security, or into the WS-I Basic Security Profile, but all this advice is relevant generally, and should be consolidated in a place where it is available to all users and implementers of the base technologies.

Denial of service concerns, in particular, have received very scant attention from the specification committee or implementing vendors. The most common platform for utilization of these technologies are application server systems operating with limited thread pools and on "managed" runtimes, such as the .Net CLR or the Java Virtual Machine. In these programming and operating environments, the demands of, e.g. the XML Encryption standard, that "implementations should be able to restrict arbitrary recursion and the total amount of processing and networking resources a request can consume" are simply not reasonable to follow – even less so in languages like JavaScript that are even further removed from the underlying platform.

As a basic security technology, messages with XML Signatures and Encryption MUST be assumed to be hostile. It is unacceptable that a single or a very few poison messages can be used to disable critical business systems in a Service Oriented Architecture, or incapacitate the identity provider or single sign on gateway for an entire set of web applications and services. This is not reliable security, and such outages can cost millions of dollars an hour in a large enterprise.

For this reason, I believe the W3C must improve the security considerations and guidance for implementers and produce a "self-contained and secure" profile in which messages utilizing XML Signatures and Encryption will self-contain all necessary context for validation and that validation can be done with deterministic (for a given size message) and finite resource consumption. Invalid or tampered messages must fail quickly.

Proposed additions to the "Security Considerations" section

With regard to supported transform algorithms generally, and the XSLT algorithm specifically:

Implementers SHOULD disable the XSLT transform by default. Any implementation willing to process arbitrary XSLT must admit the possibility of denial of service. This transform is OPTIONAL and cannot be relied upon for interoperability purposes. In the case that the XSLT transform is required:

- The caller SHOULD be required to explicitly enable the algorithm.
- The XSLT processor SHOULD have security-sensitive extensions disabled.

Implementers must be wary of using shared, platform-default services, as the properties of these processors may change asynchronously. An instantiation and configuration private to the signature validation system is recommended.

The XSLT transform SHOULD NOT be a supported algorithm for KeyInfo RetrievalMethod without explicit user consent. Enabling the XSLT transform for SignedInfo References SHOULD NOT enable it for KeyInfo RetrievalMethods as a side effect.

Implementers SHOULD carefully consider the security implications of all transform algorithms, and whether it is appropriate to execute these processing instructions from anonymous and/or authenticated originators as part of signature validation. Callers SHOULD have the ability to explicitly enumerate all supported transform algorithms, and enable or disable them selectively and independently for both SignedInfo Reference and KeyInfo RetrievalMethod processing.

When validating a signature, callers SHOULD have the ability to set hard timeout values and limit the total amount of system resources consumed when validating signatures or dereferencing key. Setting a limit on the total input size is the simplest case. Callers SHOULD have the ability to use URI resolvers with different properties for processing the anonymous KeyInfo and the authenticated SignedInfo. For example, a caller may be willing to dereference remote URIs in SignedInfo after authenticating the originator, but only allow same-document references in KeyInfo as an attack surface reduction measure. In general, implementers should very carefully consider whether all exposed dependencies have been properly hardened against malicious input.

With regard to the order of operations when validating a signature:

From the cryptographic perspective, signature validation is a pure function, but following a proper order of operations when validating a signature can substantially reduce the attack surface of a concrete implementation. The signature should be able to be divided into two classes of attack surface to which differing levels of restriction may apply: anonymous and authenticated. KeyInfo is always anonymous, but the processing instructions in SignedInfo can be authenticated.

The following order of operations SHOULD be supported by an XML Signature API:

1. Selection of a trusted key.
    a. If KeyInfo is to be used, the user must have the option to extract the key first and make a trust decision, before continuing with core validation.
    b. APIs of the form: "KeyInfo validate()", which only return a key after performing all of core validation, unacceptably expose the instructions in SignedInfo on the anonymous attack surface because the returned key may not be trusted by the caller and all operations are completed before a trust decision can be made.
2. Cryptographic signature validation of the signature calculated over SignedInfo. This assures that the SignedInfo has not been tampered with.
3. At this point, the processing instructions have been authenticated, and the caller may choose to proceed to reference validation, the verification of the digest contained in each Reference in SignedInfo.

With regard to remote and complex references exploiting multiple-parser ambiguity:

The implementation should provide a way for relying applications to retrieve the actual verified Reference material, EXACTLY as it was processed by the signature validator, e.g. by caching a copy of the normalized node set or octet stream immediately prior to hashing.

With regard to canonicalization of the SignedInfo:

Comments are not semantically relevant to the SignedInfo block, and are unlikely to be processed by or visible to the relying application. To reduce an attacker's freedom in crafting messages that exploit hash collisions, canonicalization of the SignedInfo SHOULD utilize an algorithm which discards comments.

Proposal for a "Self-contained and Secure" signature profile

The goal of the self-contained and secure profile is to ensure that a conforming signature can be verified in a deterministic time, relative to its total size, and without relying on network resources or attacker-supplied processing instructions beyond a constrained set.

With regard to Reference URIs:

Reference URIs MUST be either whole-document references (URI="") or same-document bare XPointers identifying content by xml:Id (URI="#ref1").

With regard to Transforms:

The enveloped, enveloping, base64 and canonicalization transforms are the only allowed algorithms. Each algorithm may appear in its relevant context EXACTLY ONCE.

With regard to KeyInfo:

KeyInfo must meet the same constraints on URIs and Transforms specified above.

With regard to canonicalization:

Exclusive canonicalization should be used. Documents MUST be entity-normalized prior to signing. Entities other than the standard XML single-character escape sequences are not allowed and should cause an immediate failure of validation. Canonicalization of the SignedInfo MUST exclude comments.

About the submitter

Brad Hill is a principal security consultant with iSEC Partners, where he assists companies in the health care, financial services and software development industries in developing and deploying secure software. He has discovered vulnerabilities, written whitepapers, created tools and spoken on attacking the XML security standards at Syscan, Black Hat and to private audiences at OWASP chapters and major corporations. He can be reached at brad@isecpartners.com.