SAML V2.0 Metadata Interoperability Profile Version 2.0

Working Draft 01
2 March 2010

Specification URIs:

This Version:

TBD

Previous Version:

TBD

Latest Version:

http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop-2.0.html

http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop-2.0.odt (Authoritative)

http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop-2.0.pdf

Latest Approved Version:

TBD

Technical Committee:

OASIS Security Services TC

Chair(s):

Thomas Hardjono, M.I.T.

Editors:

Scott Cantor, Internet2

Related Work:

This specification supercedes the SAML V2.0 Metadata Interoperability Profile [MetaIOP].

Abstract:

This profile describes a set of rules for SAML metadata producers and consumers to follow such that federated relationships can be interoperably provisioned, and controlled at runtime in a secure, understandable, and self-contained fashion. Runtime use of assymmetric keys and/or Kerberos credentials is included.

Status

This document was last revised or approved by the SSTC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC by using the "Send A Comment" button on the TC's web page at http://www.oasis-open.org/committees/security.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the IPR section of the TC web page (http://www.oasis-open.org/committees/security/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/security.

Notices

Table of Contents

1 Introduction

The SAML V2.0 metadata specification [SAML2Meta] defines an XML schema and a set of basic processing rules intended to facilitate the implementation and deployment of SAML profiles, and generally any profile or specification involving SAML. Practical experience has shown that the most complex aspects of implementing most SAML profiles, and obtaining interoperability between such implementations, are in the areas of provisioning federated relationships between deployments, and establishing the validity of cryptographic signatures and handshakes. Because the metadata specification was largely intended to solve those exact problems, additional profiling is needed to improve and clarify the use of metadata in addressing those aspects of deployment.

This profile is the product of the implementation experience of several SAML solution providers and has been widely deployed and successfully used in furtherance of the goal of scaling deployment beyond small numbers into the hundreds and thousands of sites, without sacrificing security.

Experience has shown that the most frustrating part of using SAML (and many similar technologies) is that products approach the use of cryptography and trust in wildly inconsistent ways, and often the libraries that such products depend on do the same in their own domains. Key management is hard, and often relies on complicated tools with cryptic output. Standards only help insofar as they can be understood and widely implemented; this has generally not occurred above a basic level of cryptographic correctness. A formal public key infrastructure (PKI) is a tremendously complex, and some would say intractable, goal; it could be argued that SAML itself is a reaction to this. Often, the security of deployments is based on a presumption that required practices such as certificate revocation checking are being performed, when in fact they are not.

Of course, it is the case that some deployments, at least to date, have overcome some or all of these problems. They may have a mature PKI, possibly one that existed long before their use of SAML, or they may require such a PKI for other purposes. In such cases, it is obviously less beneficial to deploy a second trust infrastructure based on SAML metadata. Another factor in this profile's usefulness is the relationship between SAML and the other security technologies involved in a deployment; if the use of SAML is subordinated to a secondary role, this profile may be less applicable.

The purpose of this profile is to guarantee that in a correct implementation, all security considerations not deriving from the particular cryptography used (i.e., algorithm strength, key sizes) can be isolated to metadata exchange and acceptance, and not affect the runtime processing of messages. In other words, given a metadata instance and presuming that it is successfully processed and has not been updated or superseded, it should be possible with no other information supplied to determine whether a given credential (e.g., a key or certificate) will be accepted by an implementation when used to secure a SAML protocol or assertion.

If an implementation can be shown to rely solely on the acceptance of metadata to derive trust, it can be reasoned about in a much simpler way, and the security exposures can be well understood. Furthermore, this profile accomplishes a number of additional practical goals:

- simplifying ordinary implementations and deployments
- reducing the technical foundation required to understand and use implementations
- scaling the provisioning of federated relationships (via processing of metadata batches)
- facilitating the use of XML encryption without dependency on weaker methods for establishing knowledge of public keys (e.g., guessing based on TLS server certificates)
- radically simplifying interactions between existing federated deployments (i.e. interfederation)

Most importantly, these goals can be accomplished without sacrificing security. Too often, the reaction to security complexity is to produce competing approaches that start by rejecting the notion that a substantial degree of security is achievable in the general case.

Another benefit of this profile is to produce a greater awareness of the importance of securing the exchange of metadata. Deployers have sometimes tended to ignore this issue by falling back on the assumption that the underlying PKI would provide the real security of the system, resulting in other exposures due to insecure provisioning of other important information.

Finally, note that, in addition to SAML V2.0 itself, this profile is applicable to any set of use cases supported by SAML metadata, including SAML V1.x profiles (as in [SAML1Meta]) and any other specifications that may profile SAML metadata.

1.1 Notation

This specification uses normative text.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119]:

> …they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)…

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

```
Listings of XML schemas appear like this.
```

```
Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAML2Core].
`md:`	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace [SAML2Meta].
`krb:`	urn:oasis:names:tc:SAML:2.0:attribute:kerberos	This is the SAML V2.0 Kerberos Attribute Profile Version 1.0 namespace [KrbAttr].
`ds:`	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].

This specification uses the following typographical conventions in text: `<SAMLElement>`, `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

1.2 Normative References

[KrbAttr] OASIS Committee Draft, *SAML V2.0 Kerberos Attribute Profile Version 1.0*, December 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-attribute-kerberos.pdf

[RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt

[RFC2818] E. Rescorla. *HTTP Over TLS*. IETF RFC 2818, May 2000. http://www.ietf.org/rfc/rfc2818.txt

[RFC4120] C. Neuman et al. *The Kerberos Network Authentication Service (V5)*. IETF RFC 4120, July 2005. http://www.ietf.org/rfc/rfc4120.txt

[SAML2Bind] OASIS Standard, *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[SAML2Core] OASIS Standard, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[SAML2Errata] OASIS Approved Errata, *SAML V2.0 Errata*, October 2009. http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf

[SAML2Meta] OASIS Standard, *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[SAML2Prof] OASIS Standard, *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing, Second Edition*. World Wide Web Consortium Recommendation, June 2008. http://www.w3.org/TR/xmldsig-core/

1.3 Non-Normative References

[RFC4346] T. Dierks, E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*. IETF RFC 4346, April 2006. http://www.ietf.org/rfc/rfc4346.txt

[RFC5280] D. Cooper, et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 5280, May 2008. http://www.ietf.org/rfc/rfc5280.txt

[SAML1Meta] OASIS Standard, *Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x*, November 2007. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml1x-metadata-os.pdf

[MetaIOP] OASIS Committee Specification, *SAML V2.0 Metadata Interoperability Profile Version 1.0*, August 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop.pdf

2 SAML V2.0 Metadata Interoperability Profile

2.1 Required Information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:metadata-iop:2.0`

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: SAML V2.0 Metadata Interoperability Profile [MetaIOP].

2.2 Profile Overview

The SAML V2.0 profiles [SAML2Prof] and metadata [SAML2Meta] specifications, and subsequent profiles within OASIS and in other communities (e.g., [SAML1Meta]), describe the use of SAML metadata as a means of describing deployment capabilities and providing partners with information about endpoints, keys, profile support, processing requirements, etc.

This profile extends these practices by guaranteeing that a given metadata document will be consistently interpreted by any conforming implementation of higher level profiles. To this end, it requires that metadata be usable as a self-contained vehicle for communicating trust such that a user of a conforming implementation can be guaranteed that any and all rules for processing digital signatures, encrypted XML, and transport layer cryptography (e.g., TLS/SSL [RFC4346]) can be derived from the metadata alone, with no additional trust requirements imposed.

This profile requires that all runtime decisions are made on the basis of key or Kerberos principal name comparisons, and not on any traditionally certificate-influenced basis. This permits a signed metadata file that conforms to this specification to be semantically equivalent to an X.509-based public key infrastructure (PKI); hence there is little value in the additional layer of complexity provided by certificate validation as specified in [RFC5280]. Operational experience also shows that managing signed metadata is easier than managing a PKI of the corresponding size and scale.

This revision of the specification introduces the capability to use a conventional Kerberos [RFC4120] infrastructure.

2.3 Metadata Exchange and Acceptance

This profile does not constrain the method(s) by which metadata is published or acquired, but only its content and interpretation. It is assumed that, subject to the security and deployment requirements of the participants, some means of exchanging metadata exists that results in the "acceptance" of metadata by a consumer. Acceptance in this profile is defined as an explicit treatment of everything in the metadata as "true", for the purposes of the metadata consumer's operational behavior. The truth of a given set of metadata is of course contingent upon the metadata not being superseded by newer metadata, which may conflict with, and therefore render obsolete, the earlier information.

In other words, this profile does not define how (or how often) metadata is exchanged or how and why it is trusted, but rather assumes that it is exchanged and trusted, and proceeds from that starting point. Dynamic exchange (as described in [SAML2Meta]), manual exchange, the aggregation and signing of metadata by third parties, or any other mechanism, can be used in conjunction with this profile. Note that verification of metadata signatures, if applicable, is considered to be part of this prerequisite step.

The rest of this profile deals with the requirements for producing metadata, and a conformant consumer's obligations having accepted it.

Finally, note that accepting metadata does not imply that a relying party will interoperate with a specific asserting party; it implies only that if it does so, it does so in a predictable fashion based on the metadata it accepts about that party.

2.4 Implementation Constraints

2.4.1 Peer Authentication

An additional constraint is necessitated by the inability of SAML metadata to express the authentication requirements of back-channel communications between SAML-using entities, such as via the SAML SOAP binding [SAML2Bind]. In lieu of extending metadata to capture such requirements, this profile assumes that such communications are secured by means of some combination of TLS/SSL and digital signing. If this assumption cannot be made, this profile might need to be supplemented in such scenarios.

2.5 Metadata Producer Requirements

A producer of metadata that adheres to this profile may be an actual participant in a SAML (or other) profile, or an aggregator of metadata describing many such participants. In either case, the content of the metadata itself is independent of its source and MUST stand alone as a description of the requirements for securely communicating with the entity (or entities) described therein, to the extent that the constructs of the SAML V2.0 metadata specification [SAML2Meta] can express these requirements.

Subject to any constraints of the exchange mechanisms in use, a conforming metadata instance may be rooted by either an `<md:EntityDescriptor>` or `<md:EntitiesDescriptor>` element. Any `<md:RoleDescriptor>` element (or any derived element or type) appearing in the metadata instance MUST conform to this profile's requirements.

Within the context of a particular role (and the protocols it supports, as expressed in its `protocolSupportEnumeration` attribute), any and all cryptographic keys and Kerberos principal names that are known by the producer to be valid at the time of metadata production MUST appear within that role's element, in the manner described below in section 2.5.1. This includes not only signing and encryption keys, but also any keys used to establish mutual authentication with technologies such as TLS/SSL.

Signing or transport authentication keys or Kerberos principal names intended for future use MAY be included as a means of preparing for migration from an older to a newer key or Kerberos principal name (i.e., key rollover or change of Kerberos principal name). Once an allowable period of time has elapsed (with this period dependent on deployment-specific policies), the older key or Kerberos principal name can be removed, completing the change. Expired keys or Kerberos principal names (those not in use anymore by an entity, for reasons other than compromise) SHOULD be removed once the rollover process to a new key (or keys) or new Kerberos principal name (or names) has been completed.

Compromised keys MUST be removed from an entity's metadata. A Kerberos principal whose secret key has been compromised MUST also have its name removed from an entity's metadata until a new secret key has been securely established. The metadata producer MUST NOT rely on the metadata consumer utilizing online or offline mechanisms for verifying the validity of a key (e.g., X.509 revocation lists, OCSP, etc.). The exact time by which a compromise is reflected in metadata is left to the requirements of the parties involved, the metadata's validity period (as defined by a `validUntil` or `cacheDuration` attribute), and the exchange mechanism in use.

2.5.1 Key and Kerberos Principal Name Representation

Each key or Kerberos principal name included in a metadata role MUST be placed within its own `<md:KeyDescriptor>` element, with the appropriate `use` attribute (see section 2.4.1.1 of [SAML2Meta], as revised by E62 in [SAML2Errata]), and expressed using the `<ds:KeyInfo>` element.

One or more of the following representations within a <ds:KeyInfo> element MUST be present:

- <ds:KeyValue>
- <ds:X509Certificate> (child element of <ds:X509Data>)
- <krb:KerberosSname> (child element of <krb:KerberosData>)
- <krb:KerberosCname> (child element of <krb:KerberosData>)

In the case of <ds:X509Certificate>, only a single certificate is permitted. If both <ds:KeyValue> and <ds:X509Certificate> are used, then they MUST represent the same key.

Any other representation in the form of a <ds:KeyInfo> child element (such as <ds:KeyName>, <ds:X509SubjectName>, <ds:X509IssuerSerial>, etc.) MAY appear, but MUST NOT be required in order to identify the key (they are hints only).

In the case of an X.509 certificate, there are no requirements as to the content of the certificate apart from the requirement that it contain the appropriate public key. Specifically, the certificate may be expired, not yet valid, carry critical or non-critical extensions or usage flags, and contain any subject or issuer. The use of the certificate structure is merely a matter of notational convenience to communicate a key and has no semantics in this profile apart from that. However, it is RECOMMENDED that certificates be unexpired.

In the case of a Kerberos principal name, there are no special requirements except that the Kerberos principal MUST exist within the claimed realm. The <krb:KerberosCname> representation MUST only be used for metadata roles that sign or encrypt plaintext.The <krb:KerberosSname> representation MUST only be used for metadata roles that validate signatures or decrypt ciphertext.

2.6 Metadata Consumer Requirements

A metadata consumer MUST have the ability to fully provision and configure itself based on the content of a metadata instance that it has accepted (see section 2.3), within the constraints of the information represented by the SAML V2.0 metadata specification [SAML2Meta] and any profiles that make use of it. A consumer need not provision policy that is outside the scope of metadata, but MUST have the ability to interoperate with the entities described by a metadata instance that it accepts, constrained by whatever default policies it applies.

Subject to the constraints of the exchange mechanism(s) in use, a metadata consumer MUST be able to process instances rooted with either an <md:EntityDescriptor> or <md:EntitiesDescriptor> element. When processing an <md:EntitiesDescriptor> element, each <md:EntityDescriptor> element contained within it MUST be processed in accordance with this profile.

2.6.1 Key and Kerberos Principal Name Processing

Each key or Kerberos principal name expressed by a <md:KeyDescriptor> element within a particular role MUST be treated as valid when processing messages or assertions in the context of that role. Specifically, any signatures or transport communications (e.g., TLS/SSL sessions) verifiable with a signing key or Kerberos principal name MUST be treated as valid, and any encryption keys or Kerberos principal names found MAY be used to encrypt messages or assertions (or encryption keys) intended for the containing entity.

Subsequent to accepting a metadata instance, a consumer SHOULD NOT apply additional criteria of any kind on the acceptance, or validity, of the keys or Kerberos principal names found within it or their use at runtime, with the exception of Kerberos protocol exchanges between a metadata consumer and Kerberos Key Distribution Center, as these may be necessary for the protocol itself to operate (such as requesting a service ticket for an entity described in the metadata that is subsequently used to sign a SAML message).

Specifically, consumers SHOULD NOT apply any online or offline techniques including, but not limited to, X.509 path validation or revocation lists, OCSP responders, etc. Use of such mechanisms, if unavoidable, will cause interoperability issues. In any case, consumers MUST support the acceptance of certificates without CRL distribution points or Authority Information Access extensions.

The semantics of the following key and Kerberos principal name representations within a <ds:KeyInfo> element are defined:

- <ds:KeyValue>
- <ds:X509Certificate> (child element of <ds:X509Data>)
- <krb:KerberosSname> (child element of <krb:KerberosData>)
- <krb:KerberosCname> (child element of <krb:KerberosData>)

In the case of <ds:KeyValue>, the key itself is explicitly identified. In the case of <ds:X509Certificate>, a metadata consumer MUST extract the public key found in the certificate and SHOULD NOT honor, interpret, or make use of any of the information found in the certificate other than as an aid in identifying the key used (based, for example, on information found at runtime in an XML digital signature's <ds:KeyInfo> element or the certificate presented by a transport peer).

In the case that a candidate key is identified, a signature can be directly evaluated based on whether it is verifiable with the key. Authentication of a transport peer can be evaluated by extracting the key presented by the peer (often in the form of an X.509 certificate) and comparing it by value to the candidate key. This process has the effect of decoupling the certificates that may be present in metadata from those presented at runtime, provided that the public keys are in fact the same.

A metadata consumer, when implementing authentication of a transport peer via TLS/SSL, MAY retain the checking of server certificate names (e.g., subjectAltName or Common Name) in accordance with [RFC2818]. Note that this constrains the certificates that may be used at runtime for TLS/SSL server authentication, but does not affect certificates that might appear in metadata, because the eventual comparison is based solely on the key.

In the case that a candidate Kerberos principal name is identified, a signature can be evaluated by authenticating the Kerberos AP-REQ message included in the <ds:KeyInfo> element of the signature and comparing the service ticket's cname field by value with the Kerberos principal name given in metadata. Authentication of a transport peer can be evaluated by comparing the Kerberos principal name claimed by the peer's authenticated service ticket with the Kerberos principal name given in metadata.

2.7 Security Considerations

A number of important exposures arise from the reliance on metadata alone to control runtime trust decisions.

Metadata becomes a critical tool for the revocation of compromised sites and keys, and all of the standard practices in the use of tools like CRLs become relevant to the consumption of metadata. The specification has the mechanisms to address these issues, but they have to be used. Specifically, metadata obtained via an insecure transport should be both signed, and should expire, so that consumers are forced to refresh it often enough to limit the damage from compromised information. Either the `validUntil` or `cacheDuration` attribute may be appropriate to mitigate this threat, depending on the exchange mechanism.

In addition, distributing signed metadata without an expiration over an untrusted channel (e.g., posting it on a public web site) creates an exposure. An attacker can corrupt the channel and substitute an old metadata file containing a compromised key and proceed to use that key together with other attacks to impersonate a site. Repeatedly expiring (using a `validUntil` attribute) and reissuing the metadata limits the window of exposure, just as a CRL does. Note that the `cacheDuration` attribute does not prevent this attack.

A broad set of concerns arises in the dynamic exchange of metadata self-published by a site. In such cases, it may seem untenable to trust someone to properly identify their own key, and of course it may be. Rather than constraining the acceptance of that key, this profile relies on securing the exchange and acceptance of the metadata. Traditional PKI protections can be applied to that document and/or its exchange, subsequently leveraging that protection to establish trust in the key within the metadata.

For example, when using the Well Known Location resolution profile [SAML2Meta], a producer may use an X.509 certificate to sign the metadata. This certificate can be bound to the metadata through its subject or subjectAltName (which might contain a SAML entityID). This ensures the appropriate key/name binding for the signature.

3 Conformance

3.1 SAML V2.0 Metadata Interoperability Profile Version 2.0

3.1.1 Public Key Conformance Mode

A metadata producer supports the "Public Key Conformance Mode" of this profile if it can produce metadata consistent with the normative text in section 2.5 and if it supports the production of `<ds:KeyValue>` or `<ds:X509Certificate>` elements in accordance with section 2.5.1.

A metadata consumer supports the "Public Key Conformance Mode" of this profile if it can process metadata consistent with the normative text in section 2.6 and if it supports the consumption of `<ds:KeyValue>` or `<ds:X509Certificate>` elements in accordance with section 2.6.1.

3.1.1.1 Strict Public Key Conformance Mode

A metadata consumer supports the "Strict Public Key Conformance Mode" of this profile if it supports the "Public Key Conformance Mode", per section 3.1.1, and does so without the imposition of certificate path validation, revocation lists, OCSP, or other related technologies when evaluating certificates.

3.1.2 Kerberos Conformance Mode

A metadata producer supports the "Kerberos Conformance Mode" of this profile if it can produce metadata consistent with the normative text in section 2.5 and if it supports the production of `<krb:KerberosCname>` or `<krb:KerberosSname>` elements in accordance with section 2.5.1.

A metadata consumer supports the "Kerberos Conformance Mode" of this profile if it can process metadata consistent with the normative text in section 2.6 and if it supports the consumption of `<krb:KerberosCname>` or `<krb:KerberosSname>` elements in accordance with section 2.6.1.

Appendix A. Acknowledgements

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- TBD

The editor would also like to acknowledge the following contributors:

- Walter Hoehn, University of Memphis
- Chad LaJoie, SWITCH
- Ian Young, EDINA, University of Edinburgh
- Josh Howlett, JANET

Appendix B. Revision History

- Draft 01, revised from CS-01 of original profile with new Kerberos material and softened language on certificate processing.