Phasing Out SHA-1

Phasing Out the Use of SHA-1

As of January 1, 2014, NIST disallows the use of the SHA-1 digest algorithm in conjunction with digital signatures (see: NIST SP 800-57 Part 1, Revision 3, July 2012, Tables 3 and 4). This was a major driver behind the Phase 1 Recommendations of the Metadata Distribution Working Group, which gave rise to the Phase 1 Implementation Plan, an effort that is nearing completion.

On December 18, 2013, InCommon began a migration process for phasing out the use of the SHA-1 digest algorithm within the Federation. The next milestone event in that migration process is scheduled to take place on June 30, 2014, as described in the next section.

Contents:

SHA-1 and SAML Metadata

All SAML metadata distributed by InCommon is digitally signed for authenticity and integrity. The XML signature on the metadata uses a hashing algorithm, first to bulk-digest the XML metadata and then to sign the XML document itself. Although these are independent operations, a single hashing algorithm is often used for both purposes, and until recently, the SHA-1 hashing algorithm was used exclusively.

At this point in time (May 2014), all but one of the metadata aggregates published by InCommon are signed using the SHA-256 digest algorithm. The fallback aggregate, which still uses the SHA-1 digest algorithm, will be upgraded by the middle of this calendar year.

On June 30, 2014, the fallback metadata aggregate will be synced with the production metadata aggregate. That is, after June 30, *all metadata aggregates published by the InCommon Federation will be signed using the SHA-256 digest algorithm*. To avoid complications, all SAML deployments are strongly encouraged to *migrate to the production aggregate or the preview aggregate* ***now*** but no later than June 30, 2014.

A frequently asked question is: What will happen if I do nothing? One answer is: In 90% of the cases, if you do nothing, your deployment will continue to function as normal after June 30th. We say this because we are confident that the majority of SAML deployments in the InCommon Federation are compatible with SHA-256.

Instead of drilling down on the remaining 10%, consider this: We know with 100% certainty that the fallback metadata aggregate will be signed using a SHA-256 digest algorithm beginning on June 30th, so knowing nothing else but that simple fact, we conclude that all deployers are better off explicitly migrating to the production aggregate than they are doing nothing because all other things being equal (which they are) *a controlled migration is always safer than a forced migration*.

Indeed, migrating to the production aggregate is as simple as changing the URL in your metadata configuration. Go ahead, schedule that simple configuration change subject to whatever change management policy you have in place at your site. You'll know within minutes if it's going to work, and honestly, there is a very high probability it will Just Work. If it does, you're home free because you can do the rest of the migration on your own time. If it doesn't work, you can quickly back out the change and invoke Plan B.

All SAML deployments in the InCommon Federation should migrate to the production metadata aggregate (or the preview metadata aggregate) **now**. Recommended configuration options for Shibboleth and simpleSAMLphp are documented in the wiki.

So why does the simple migration outlined above work without bootstrapping an authentic copy of the new metadata signing certificate? Because a conforming metadata consumption process ignores all the certificate details except the public key bound to the certificate, and the public key has not changed.

For more information, see: Metadata Migration Process

Questions? Join this mailing list: https://lists.incommon.org/sympa/info/metadata-support

SHA-1 and X.509 Certificates

SAML entities (IdPs and SPs) manage at least two types of private keys: TLS keys and SAML keys. The corresponding public keys are bound to X.509 certificates, which are signed of course. Like the signature on XML metadata, the signature over an X.509 certificate relies on a digest algorithm, typically the SHA-1 digest algorithm.

SHA-1 is not an encryption algorithm

Encryption implies the ability to decrypt (i.e., encryption by definition is reversible) but SHA-1 and SHA-256 are *one-way hash functions*. A hash function by definition is **not** reversible. So neither SHA-1 nor SHA-256 are encryption algorithms.

SHA-1 and TLS Certificates

As you probably know, SAML endpoints in metadata are protected with TLS. In particular, browser-facing endpoints in metadata are associated with TLS certificates signed by a trusted CA (i.e., a CA trusted by your browser). Browser-facing TLS certificates do not appear in metadata and are therefore out of scope with respect to the Implementation Plan.

Endpoints in metadata that support the so-called back-channel protocols are associated with SAML certificates in metadata. These and other certificates in metadata are discussed in the next section.

Not surprisingly, the signature over a trusted (browser-facing) TLS certificate typically relies on the SHA-1 digest algorithm, and so the CA and browser vendors are also migrating to the SHA-2 family of digest algorithms. This will be the topic of a future blog post.

SHA-1 and SAML Certificates

The rest of this section concentrates on the public keys bound to certificates in SAML metadata.

A SAML entity is a secure web server that produces and/or consumes SAML messages. These SAML messages may be signed or encrypted using asymmetric crypto operations. A relying party uses a trusted public key in metadata to verify an XML signature or perform an XML encryption operation. These public keys are bound to long-lived, self-signed certificates in metadata. Like all X.509 certificates, the signature on a certificate in metadata uses a hashing algorithm, which is almost always the SHA-1 hashing algorithm.

Fortunately, *there are absolutely no security considerations concerning the signature on certificates in metadata*. In fact, the only concern is the actual public key bound to the certificate in metadata. Every other aspect of the certificate is ignored by the SAML software (or at least software that's doing the Right Thing). That's why InCommon recommends the use of long-lived, self-signed certificates in metadata: the focus is on the key, not the certificate.

Don't fiddle with the signature on self-signed certificates in metadata. Until the signature on a certificate in metadata actually becomes an interoperability issue, it is best to leave it alone.

SHA-1 and SAML Assertions

Like metadata, SAML messages may be digitally signed for authenticity and integrity. Any SAML entity (IdP or SP) may sign a message but in practice it is the IdP that wields a signing key since SAML assertions issued by the IdP MUST be signed. The SP uses the IdP's public signing key in metadata to verify the signature on the assertion just-in-time, that is, when the assertion is presented to the SP by the browser user.

It is believed that most IdPs in the InCommon Federation are signing assertions using the SHA-1 digest algorithm. This is because most IdP deployments use the Shibboleth IdP software, which is known to support SHA-1 only, at least out of the box.

SAML deployers should take note of the following facts:

1. If your SAML deployment can consume metadata signed using the SHA-256 digest algorithm, it very likely can consume SAML assertions signed using the SHA-256 digest algorithm.
2. The ability to sign SAML assertions using the SHA-256 digest algorithm is not fully supported by the Shibboleth IdP software (but please read on).

Clearly the former is an enabler while the latter is an inhibitor. Without ample software support, InCommon can not realistically recommend a firm SHA-256 migration path for IdPs. The best we can do is to ensure ubiquitous SHA-256 support at the SP and then encourage individual IdPs to devise a migration plan that makes sense for them.

After June 30, 2014, almost all SPs in the InCommon Federation will support SHA-256. Given this, each IdP should decide, subject to its own unique set of circumstances, when to begin signing SAML assertions using the SHA-256 digest algorithm.

SimpleSAMLphp has good support for signing assertions using the SHA-256 digest algorithm. It can be configured to sign assertions on a per-SP basis, that is, a simpleSAMLphp IdP can sign assertions using the SHA-1 digest algorithm for some SPs and the SHA-256 digest algorithm for others. Therefore sites running the simpleSAMLphp IdP software can and should migrate to SHA-256 as soon as possible. A perfectly reasonable strategy would be to configure the use of SHA-256 by default but to fall back to SHA-1 for those few remaining SPs (perhaps external to InCommon) unable to handle SHA-256.

To enable SHA-2 support in the Shibboleth IdP, you can use a 3rd-party extension to enable SHA-2 capability for Shibboleth IdP versions 2.3 and later. Note that the Shibboleth configuration is all or none, either SHA-1 or SHA-256, so Shibboleth IdPs are advised to wait until after June 30, 2014, when all SPs will be consuming metadata signed using a SHA-256 digest algorithm.

Actually, it's a bit more complicated than that for Shibboleth deployers. Shibboleth IdP v3 is due out later this calendar year and of course v3 will fully support the SHA-2 family of digest algorithms, including SHA-256. So the question for each Shibboleth deployer is whether to install the extension in their current deployment of Shibboleth IdP v2 or wait for Shibboleth IdP v3?

For those IdPs certified or looking to be certified in the Assurance Program, the Bronze and Silver Profiles require that "Cryptographic operations shall use Approved Algorithms" and since SHA-1 is no longer an "Approved Algorithm," it can not be used to meet the requirements of Bronze or Silver. However, the InCommon Assurance Program is working on releasing a variance to the requirements (called an *Alternative Means*) to permit the use of SHA-1 until January 15, 2015.

Be advised that if you intend to interoperate with a federal agency SP that requires Bronze or Silver, assertions MUST be signed using a SHA-2 digest algorithm. Such an IdP should begin the migration to SHA-256 immediately.