Phase 1 Recommendations

Phase 1 Report of the MD-Distro Subcommittee

Terminology

- *signing key*: a private key used for signing metadata; an RSA 2048-bit private key
- *signing certificate*: an X.509v3 certificate containing a public key used to verify the signature on a metadata file; a container for an RSA 2048-bit public key

Introduction

The InCommon metadata signing certificate is signed by a CA whose certificate expires in March 2014. The phase 1 goal of this subcommittee is to determine if a traditional X.509 PKI is required for the foreseeable future. With this goal in mind, this subcommittee offers the following observations, recommendations and guidance to InCommon's Technical Advisory Committee and InCommon Operations.

The discussions documented here occurred over a series of weekly phone calls with email discussion on the md-distro@incommon.org email list. Members of the InCommon participant community were invited to join these open discussions. Notes and archives are available on the subcommittee's home page.

Observations

In general, a PKI provides flexibility when an end-entity certificate (which in our case is the metadata signing certificate) is re-keyed. Another advantage is that an end-entity certificate can be revoked without much reconfiguration (provided end users are doing revocation correctly). However, reconfiguration headaches will occur when the PKI's root certificate expires or otherwise needs to be replaced.

Furthermore, even in a full-blown PKI, the root of a trust path does not have to be a CA. Path validation and trust in the path is what is important. The goal is that the certificate being evaluated is trusted and currently valid. In practice, the only case for real-time certificate validation is software intentionally pulling live metadata.

The discussion continued, dividing the analysis into two use cases: is there any value in rooting a metadata signing certificate in (1) an online or (2) an offline CA?

It was also noted that relying on Certificate Revocation Lists (CRLs) extends the vulnerability window in relation to a given signed metadata aggregate's expiry date. The Online Certificate Status Protocol (OCSP) is also redundant, relying on a third party lookup, which is equivalent to real-time validation of entity lookups with minimal time caching. The OCSP response is very brittle due to expiration date, which is not the case with cache duration in SAML.

Conclusion

Although there may be reason to further explore online signing and therefore related security measures of online signing keys (e.g., hardware security modules), the group found no compelling reason for a traditional CA, so long as adequate practice for and documentation of the protection of the signing key is provided.

Impact

We can't be 100% certain that no one is relying on the current CA. However, there is no practical reason or known deployment to make us think this is true. What might break if we adjust current infrastructure?

1. SimpleSAMLphp deployments rely on the fingerprint of the signing certificate (rather than the key itself). So if we decide to resign the signing certificate, migration and reconfiguration issues will exist for all simpleSAMLphp users. This was the case every time we renewed the signing certificate (every two years), so this represents no significant increase in the rate of reconfiguration (since the current signing certificate expires 2 May 2014 and needs to be resigned anyway).

2. There are few non-standard deployments that we know of. This is probably low risk with the exception of, for instance, government agencies relying on CA SiteMinder. We also need to explore the limitations of ADFS.

This raises an opportunity to be more explicit in our documentation regarding expectations with respect to metadata validation. We can remind participants if they are relying on the CA today, they're relying on security patterns that are no longer relevant.

If, in the future, a CA is needed, we can resign the signing certificate at that time, or wrap a PKI around a self-signed certificate. There are known examples of this. For example, eduGAIN uses a self-signed certificate for online metadata signing.

Lifetime of the Signing Key and Signing Certificate

We discussed whether the current 2048-bit metadata signing key needs to be replaced. The question is: What would restricting the lifetime of the signing key prevent? Two vectors of attack were discussed. First, there might be an attack that relies on having large amounts of cypher text available for analysis. The more cypher text available, the greater the vulnerability. However, there are no known attacks of this type for RSA 2048-bit keys, and in our case the volume of cypher text is relatively small (compared to, say, an IdP that is constantly exercising its signing key). Second, an attack might exist that relies on sheer computational power, that is, an attacker may have theoretically been applying resources over the key's lifetime to compute the value of the private key. We found no evidence to suggest that this is practical even with today's technology. We conclude that there is currently no reasoned terminus for the lifetime of a 2048-bit key.

With respect to the signing certificate, if we no longer need a CA, we should wrap the public key in a self-signed certificate to avoid confusion. This begs the question of how to choose an appropriate expiration date for the signing certificate. In actuality, a certificate's expiration date has nothing to do with the security of the private key, which is an ongoing concern. If a key vulnerability were discovered some time in the future, we would of course take immediate action. Thus the logical conclusion is that the expiration date on the signing certificate should be far into the future, so as to avoid the situation where the signing certificate needs to be resigned unnecessarily.

Given that, it is recommended that all self-signed certificates be set to expire prior to January 2038 to avoid the so-called Year 2038 Problem. At some point, we may wish to generate a signing certificate with an expiration beyond this date for testing purposes.

Note: A self-signed signing certificate requires an effective and well understood revocation strategy.

Recommendations

1. For the foreseeable future, continue using the current signing key to sign metadata.
2. Retire the old InCommon CA (that signed the current signing certificate) and all associated public-key infrastructure. (This CA is not to be confused with the CAs associated with the InCommon Certificate Service.)
3. Replace the current signing certificate with a long-lived, self-signed certificate using the current key pair. The recommended certificate expiration date is 18 December 2037 (to avoid the Year 2038 Problem in January).
4. Create a new policy document (to replace the current CP/CPS) that describes security practices surrounding the metadata signing key, especially the revocation policy and practices associated with the new self-signed signing certificate.
5. It is anticipated that users of simpleSAMLphp will experience issues since the new signing certificate will have a different fingerprint. We will contact the SSP developers as well as communicate the migration plan clearly to participants.
6. Communicate with known users of non-standard deployments and implementations. Of particular note are those who use unsupported software, such as NSF, NIH, and possibly those participants using ADFS. This is communication, not support. See InCommon Software Guidelines, which is under the advisement of the TAC.
7. Expired certificates signed by the old InCommon CA still exist in participant metadata. After the initial communication regarding the transition to a self-signed signing certificate has occurred, send a separate communication to participants about expired certificates in metadata. Since the two issues are mostly unrelated, this will avoid confusion.
8. Review the X.509 Certificates in Metadata documentation page (and its child pages) in light of the observations and recommendations in this report.