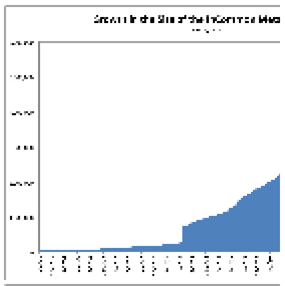
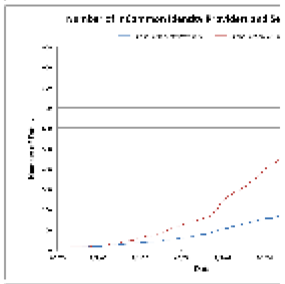
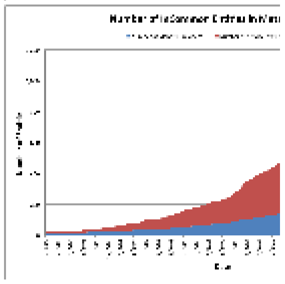
Copy of Metadata Distribution

Metadata Distribution ... a discussion, not a set of conclusions







The Problem

- the current method of metadata distribution is brittle and does not scale
- batch delivery of metadata inhibits scaling and interfederation

Problem Characterization

- *old model*: exhaustive lists of entities reminiscent of /etc/hosts
- *new model*: DNS-like model that queries for an entity's metadata on demand

The Proposed Solution

- *per-entity metadata*
- Why per-entity metadata?
    - **efficiency**: per-entity metadata is an efficient alternative to loading thousands of entity descriptors into every system when only a small subset are actually used in practice
    - **safety**: per-entity metadata avoids the brittleness of a single metadata aggregate, where one mistake somewhere breaks every system downstream

Solution Requirements

- latency of each request has to be low
- metadata hosting environment must be highly available and highly performant
- avoid moving the signature process online so the signing key can remain offline
- **barrier to adoption**: IdP implementations must support per-entity metadata

Other Neat New Features

- per-organization metadata (in conjunction with delegated administration)
- user-defined metadata aggregates based on self-asserted entity attributes
- support for JSON metadata format in addition to traditional XML

Resources

- Chart 1 tab = growth in InCommon Metadata file size (in bytes)
- Chart 2 tab = Growth in # of IdPs and SPs in metadata (line chart)
- Chart 3 tab = Growth in # of IdPs and SPs in metadata (area chart)

The charts are also shown as thumbnails to the right. Just click to expand a chart to full size.

Priorities

- a discussion -

- per entity lookup
- metadata in DNS
- Support for per-entity metadata lookup
- MDX
- `md.incommon.org`
- Support for per-entity metadata distribution

All of the above have something to do with metadata distribution, hence the name of this article. Michael’s point about “metadata in DNS” is an important long-term meta-question. Leif believes that “the mental model we want is akin to BGP not DNS,” but the point remains: the current method of metadata distribution does not scale, and in fact, is not supported outside of higher ed.

I wouldn't use "not supported outside of higher ed" as a benchmark unless the decision is to just abandon multi-lateral federation and metadata entirely.

(There are only a few packages that process metadata in the way we have come to expect: Shibboleth, simpleSAMLphp and the Eduserv products are really the main ones. We should not expect anything outside that set to be able to work with any other system we come up with (e.g., per-entity query) either.)

That said, here is a list of potential short-term tasks (most within the next 6 months) that fall under this heading:

1. new vhost for metadata distribution (`md.incommon.org`)
2. new endpoint for signed XML metadata
3. new signing key

4. MDX support
5. per-entity metadata
6. per-organization metadata
7. metadata aggregates based on self-asserted entity attributes
8. support for both XML and JSON formats (both signed)

The need for a new endpoint for metadata distribution is being driven by the fact that the CA certificate that signed the certificate containing the current (offline) metadata signing key will expire in April 2014. A new endpoint will provide a migration path away from this soon-to-become legacy CA. Moreover, this is a golden opportunity to segregate Discovery (DS) services from metadata services (which is desirable from a sysadmin point of view) and also provides an opportunity to introduce new metadata services (MDX, JSON, etc.) that will drive users towards the new vhost (`md.incommon.org`).

Why we need per-entity metadata:

1. **efficiency**: per-entity metadata is an efficient alternative to loading thousands of entity descriptors into every system when only a small subset are actually used in practice
2. **safety**: per-entity metadata avoids the brittleness of one single metadata aggregate, where one mistake somewhere breaks every system downstream

The latter is perhaps the number one problem we have to fix, and it needs to be fixed before we enable any form of XML submission (e.g., PEER/REEP, interfederation, etc.).

[inctac:probably worth observing that many other federations already work on the basis of some kind of XML submission, without providing per-entity metadata query far less relying on its universal use. Some of those are probably just ignoring the risk of a problem, but others are mitigating the risk by man-in-the-loop processes or by automated checking for known issues, or a combination of both. Establishing a distinction between what is submitted and what is published can also help.]

[inctac:yes, I've never been too sure how they manage it without the kind of effort your federation has put into it, and more to the point, I'm not sure InCommon is in a position to do the work we'd need to do]

Critical issue:

- Should the certificate that contains the new metadata signing key be self-signed?

New Metadata vhost

Can you explain why this matters? I'm not clear on why, unless you're proposing use of a particular PKI for the TLS layer as a possible trust mechanism (which is worth discussing). Otherwise, it's not a vhost issue, but just that the URL would change, which we only get limited chances to do. Seems to me we need to think about the use cases of having multiple aggregates along with this so that we plan out what locations to give people.

Multiple services run on host `wayf.incommonfederation.org`, with different server characteristics. There is the discovery service and the federated error handling service on the one hand, and the metadata service on the other hand. Manual failover suits the latter just fine. When we were asked a year ago (or was it two?) to improve on manual failover, the fact that `wayf.incommonfederation.org` runs a jumble of services presented a problem. Moreover, the metadata service (as it stands right now) needs fine-tuning with respect to HTTP conditional GET and HTTP compression, things that the DS and FEH don't need (since they are dynamic services). We concluded from this that the two groups of services need to be split (`md.incommon.org` and `ds.incommon.org`), which is where we were headed.

The expiration of the CA certificate in 2014 is causing the server split to escalate in importance, not because we want or need to reissue the CA certificate (although that is on the table), but because the more things that change at one time, the more compelling the story is, which we feel we need to motivate people.

Signed Aggregate Trust Model

See discussion of the CA certificate above. Currently, InCommon does **not** direct deployers to make use of that CA in any way. All trust flows from the signing key itself. Even renewing the signing certificate is optional.

If the signing key itelf is to change, then all deployers have a potential flag day.

[inctac:in the same way we support certificate migration for entity metadata, I assume we'd also want to say that we would support a migration for the metadata aggregate, no? A detail that might not be necessary at this point but that might quell anxieties for the punctilious]

[inctac:Sure, if it's a controlled change, then presumably moving to a new key involves moving to a new URL. Unfortunately the signature model doesn't support multiple signers well.]

If the trust model itself changes to include validation of a signing certificate, then we have to look at a number of pieces, such as revocation, and there will be some configuration differences based on software version. Not all software will support revocation.

Per-Entity Metadata Access

The main thrust of this item for most of the TAC members is that we feel batch delivery of metadata is not a long-term solution if we expect to scale and interfederate. The batches will get so large that memory requirements within the end systems will get large. Shibboleth's caching and ability to load metadata in the background is perhaps able to scale far higher than some think, but other implementations will not. Signature verification over large XML files (truly large ones) is also a performance issue. The additional benefit of per-entity metadata is that introducing an error into a given entity's metadata doesn't break any interaction but those with that entity. A batch is vulnerable to complete disruption of all service to all its entities due to one small error.

The assumed solution has been to move from the /etc/hosts model of lists of entities to the DNS model of querying for an entity's metadata if it's needed, on demand. Caching is managed via the SAML cacheDuration attribute. Actual lookup is just HTTP, which is really all MDX is. Software that supports this style of metadata access tends to not care what the URL will look like, so it can take many forms.

Obviously the metadata hosting moves from being highly available to both highly available and highly performant since the latency of a given request has to be low. DNS achieves this with UDP, but we have to pull it off with HTTP. It's worth noting that HTTPS is a nice to have, but not a hard requirement because of signatures. One could imagine generating relatively short-lived metadata instances and hosting them over http, and the risk there is much like any system with a revocation interval of that validity window.

[inctac:I don't think that HTTPS is even a nice to have. Obviously it has a negative performance impact, and if one follows the metadata interoperability profile a client will not rely on TLS for authenticity at all. The only circumstances I can think of in which it would make sense to use HTTPS would be if the client had to authenticate to the query endpoint, or if queries needed to be confidential. I don't think either of those applies in our use case.]

A key point is whether one can achieve this model without moving the signature process online, and thus moving the key online. There may be substantial concern about having an online signing key if we don't implement a full PKI to mitigate the cost of a key compromise, but it seems as though various creative strategies could be used to avoid this. Clearly, though, if one did want a PKI for other reasons, doing the signing on-demand would scale fine. It's much like running an IdP, but even lower volume.

File	Modified
PNG File entity-growth-chart2.png	Jun 26, 2013 by John Krienke (internet2.edu)
PNG File entity-growth-chart1.png	Jun 26, 2013 by John Krienke (internet2.edu)
PNG File metadata-growth-curve.png	Jun 26, 2013 by John Krienke (internet2.edu)

Download All