Fig. 1

Fig. 2

Fig. 3

1. HTTPS in the VO

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

- HTTPS is nice...
- ...but an operational liability.
- Mitigation

(cf. Fig. 3)

2. HTTPS is nice

- Basic protection of credentials in certain (arguably broken) authentication schemes
- Message integrity protection
- Half-working stand-in for code signing of javascript (though 99.9% of the malicious javascript isn't delivered by intercepting http and never was)
- Perhaps a bit more privacy

Anyway, people are migrating to HTTPS, and we'll have to work out a scheme to deal with it.

3. HTTPS is a liability

- Services need to manage their certificates, update them, make sure they cover all their host names,...
- Clients need to keep their CA certificates up to date, typically separately for OS, Java, and browser.
- No caching.
- Proxying is a lot messier.
- Bytestream-level debugging means a MITM attack against yourself.
- No WebSAMP. (Ok, we should wean our users from web pages anyway, but let's keep this realistic)

In my operation of GloTS, https services appear about three times more likely to fail than http services; things have improved a lot since I've switched of certificate checking, but then of course https becomes pointless, and we certainly can't recommend that as a mitigation of the operational liability.

4. Proposal for Mitigation

1. Recommend having *both* http and https interfaces. (Required for letsencrypt anyway)
2. Leave preference to operator.
3. In VOSI/Registry declare the alternative interface as mirrorURL.

That is:

```
<interface>
  <accessURL>http://example.org/tap</accessURL>
  <mirrorURL>https://example.org/tap</mirrorURL>
</interface>
```

or

```
<interface>
  <accessURL>https://example.org/tap</accessURL>
  <mirrorURL>http://example.org/tap</mirrorURL>
</interface>
```

5. Comments?

Could this be made more useful by agreeing on a bespoke `mirrorURL/@title` ("secured")?

Client writers feel this would be a lot easier to handle if the access URLs pointed to a whole bundle of DALI/VOSI resources.

Piecing together a full service profile from URLs from half a dozen `interface` elements isn't fun (but that's a general one, really).