Avoiding Mistakes and Minefields

Jeff Hodges

W3C Strong Authentication and Identity Workshop

10,11-Dec-2018

"common mistakes and minefields while building Strong Auth and Identity technologies" -- Which context?

i.e. at..

- protocol/system design time,
- component implementation time (eg, in libraries/packagers, clients, servers),
- deployment time (i.e., by Relying Parties and Service Providers)

..?

"common mistakes and minefields while building Strong Auth and Identity technologies" -- Which context?

i.e. at..

- protocol/system design time,
- component implementation time (eg, in libraries/packages, clients, servers),
- deployment time (i.e., by Relying Parties, Service Providers, Identity Providers)

..?

NOTE: the above are "motherhood & apple pie" aspects of building *anything*

Protocol / System Design Time

- Carefully define terminology and use it consistently
  - e.g., do you use the terms *names* and *identifiers* ?
  - Are they explicitly the same?
  - If not, what are the precise differences?

Protocol / System Design Time

- Carefully define terminology and use it consistently
    - e.g., do you use the terms *names* and *identifiers* ?
    - Are they explicitly the same?
    - If not, what are the precise differences?

- E.g.: names are fungible and non-unique, while identifiers are unique and persistent

Protocol / System Design Time

- Carefully define terminology and use it consistently
    - e.g., do you use the terms *names* and *identifiers* ?
    - Are they explicitly the same?
    - If not, what are the precise differences?

- E.g.: names are fungible and non-unique, while identifiers are unique and persistent

  e.g.: https://www.w3.org/TR/webauthn/#terminology
  https://w3c-ccg.github.io/did-spec/#terminology
  http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf

Component Implementation Time
(eg, in libraries/packages, clients, servers)

c.f. The Most Dangerous Code in the World:
Validating SSL Certificates in Non-Browser Software

- We demonstrate that SSL certificate validation is completely broken in many security-critical applications and libraries. [...]

  [...] The root causes of these vulnerabilities are badly designed APIs of SSL implementations (such as JSSE, OpenSSL, and GnuTLS) and data-transport libraries (such as cURL) which present developers with a confusing array of settings and options.

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

- are the employed underlying technologies secure? (see prior slide)

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

- are the employed underlying technologies secure *and* securely employed? (see prior slide)

- Having a carefully designed deployment architecture...
  - even though given high-level authentication and identity technologies, integrating them into delivered systems is typically non-trivial
  - e.g.: Stanford Registry & Directory Infrastructure: A Case History

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

- are the employed underlying technologies secure *and* securely employed? (see prior slide)

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

- are the employed underlying technologies secure? (see prior slide)

- Having a carefully designed deployment architecture...
  - even though given high-level authentication and identity technologies, integrating them into delivered systems is typically non-trivial
  - c.f.: Stanford Registry & Directory Infrastructure: A Case History

- Specific actual examples on following slides from deployer of federated identity

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

1) SP's requesting ForceAuthN but then not checking the concurrency of the values sent back by the IdP (ie its not a fresh re-auth) but assuming it is (oops!) [see prior slide 7]

2) SPs assuming "principal name" is an email address - or using such fields as the prime account ID where recycling of those values can occur at the IdP end over time...

3) the particular values of eduPersonAffiliation (student,staff,faculty,employee etc) are arbitrarily decided by each site in a federated world and dont mean the same thing across regions.

4) assuming all federations operate in the same way across the world (particularly when interfederated with eduGAIN - other rules of play are involved

Deployment Time
(i.e., by Relying Parties, Service Providers, Identity Providers)

5) wanting to use the best and up to date security algorithms

> loads of partners in federation space are not following BCP and there's a lowering of average level (for those with REALLY out of date stuff, you just have to remove them from your relationship)

6) users are easily confused with all the stuff

> the interface MUST be simple for selection of Identity Provider and logging in page. Decent education is needed to ensure users dont just think they can type institutional id/password into any random user/pass field on the internet!

Overall: Trust Does Not Scale

i.e.:

> "trust does not automagically scale across arbitrary policy domains"

Overall: Trust Does Not Scale

i.e.:

> "trust does not automagically scale across arbitrary policy domains"

corollary:

> "scaling trust across distinct policy domains requires specific agreements between the policy domains regarding (at least) the definition of trust, requirements for attaining degrees of trust, and what it is that is trusted."

Overall: Trust Does Not Scale

i.e.:

> "trust does not automagically scale across arbitrary policy domains"

corollary:

> "scaling trust across distinct policy domains requires specific agreements between the policy domains regarding (at least) the definition of trust, requirements for attaining degrees of trust, and what it is that is trusted."

e.g.:

> NIST.SP.800-63-3 is an attempt at defining such policy agreements across USGov agencies.

Further thoughts...

Consider implementing a simple design satisfying simple use case(s), and trying it out in practice, before attempting to "finalize" the specification. Iterate WRT satisfying more complex use cases and updating the specification.

- flexitility -- build something that is nominally useful yet malleable such that can evolve to satisfy further use cases

TODO: Diagram: (hourglass shape)

1995: 10s of home-grown web SSO and identity approaches

2001: SAML v1

Now: fewer than some number, say ~ 10 or 15?

- SAMLv2
- OIDC
- WS-Federation et al
- OpenID v2 and v1
- ?