Standards are the Nuts and Bolts of the Identity Engine*

* Warning: Some assembly required

Michael B. Jones

Principal, Self-Issued Consulting

Standards are about making choices

Phillip Hallam-Baker observed to me at an IETF meeting:

Standards make the choices that don’t matter.

What an odd thing to say,
but there’s a deep truth there.

Choices that don’t matter

It doesn’t matter that...

0x0800 is the EtherType value for IPv4 packets

6 is the IP protocol number for TCP packets

443 is the TCP port number for HTTPS octet streams

“`GET`” is the identifier for an HTTP request method

“`HTTP/1.1 200 OK`” indicates that an HTTP request succeeded

“`application/json`” is the Content-Type for JSON-encoded messages

65 is the number for the letter “A” in ASCII and Unicode

“`{`” and “`}`” delimit JSON objects

“`iss`” and “`sub`” are identifiers in JSON objects for JWT claims

Making choices deeply matters!

Interoperability requires implementations making the same choices

- Text can be input and displayed because everyone uses 65 for “A”
- HTTPS works because everyone uses TCP port 443

Standards are where those choices are written down

It’s our job as standards professionals to make those choices

Standards are the nuts and bolts of the Identity Engine

When building machines, we take for granted having standard parts

- Nuts, bolts, wires, light bulbs, and countless other parts
- All conforming to applicable standards
- Enables a marketplace of interoperable parts from multiple suppliers
- Without these standards, every part would be custom machined

The same is true of the identity and security standards we use to build the Identity Engine

Some Assembly Required

Our identity and security standards vary wildly in the degree to which they make choices

- Some define *one* way to do things, resulting in interoperability
- Others leave critical choices unmade, passing them on to implementers
- *Your mileage may vary!*

Naming Names & Taking Prisoners

Next, I'll critique existing and emerging identity and security standards through this lens

I'll give each my personal grade on choices made

A B C D F

BEST WORST

X.509

X.509 is a decades-old widely-deployed digital certificate format

There are interoperable profiles of X.509

- Especially for TLS certificates

But choices have evolved over time

- Domain names used to be in the commonName field
- Now in Subject Alternate Name (SAN) field

Multiple revocation mechanisms

- Certificate Revocation Lists (CRLs)
- Online Certificate Status Protocol (OCSP)

SAML 2.0

SAML is the original single-sign-on protocol standard

There are interoperable SAML 2.0 ecosystems

- Each made many profiling choices to achieve this

SAML NameID contents vary

- Can be transient, persistent, unspecified, emailAddress, X509SubjectName, WindowsQualifiedDomainName, Kerberos, Entity

Multiple protocol flows

- Browser profile, Artifact Binding, Enhanced Client Proxy (ECP)

Multiple logout mechanisms

Dependent upon brittle XML Canonicalization

OAuth 2.0: RFC 6749 & RFC 6750

OAuth 2.0 enables limited access to resources in controlled fashion

OAuth 2.0 is not interoperable without a profile

Different `response_type` values with different security properties

- `code`, `token`, and others defined by extensions

`scope` values completely unspecified

Multiple `token_type` possibilities

- Bearer and others

RFC 6750 defines three ways to pass access token

- Header, Body Parameter, Query Parameters

OpenID

OpenID Connect

OpenID Connect is a widely-used sign-in standard

Interoperable ecosystem enabled because of choices made

- For instance, chose exact `redirect_uri` matching
- Interoperability evidence: 754 OpenID Connect certifications to date!

Building on OAuth 2.0 introduced more choices than ideal

- Six response types, each with different security properties

Three IdP-initiated logout mechanisms

- Two using browser features, one using server-to-server communication

I E T F®

JSON Web Signature (JWS)

JWS is a widely-used JSON-based digital signature format

Compact serialization is the most used

JSON serialization was added late to satisfy vocal constituency

- JSON serialization also includes unprotected headers

Other than serialization choice, most choices made by spec

“`alg`” choice is needed to support cryptographic agility

JSON Web Token (JWT)

JWT is a widely-used JSON-based digital token format in which secured claims are made about a subject

Requires JWS compact serialization be used

Yes, all claims are optional at the JWT level

- Leaves room for profiles such as ID Token to specify claims used

JWT BCP [RFC 8725] further tightens choices made

Many interoperable implementations in different languages

I E T F®

CBOR Object Signing and Encryption (COSE)

COSE is a widely-used binary signing and encryption format

COSE makes similar degree of choices as JOSE (JWS, etc.)

Includes both protected and unprotected headers

Has some bells and whistles that JOSE doesn’t

- Such as countersignatures

Enough choices made to enable interoperability

IETF

CBOR Web Token (CWT)

CWT is a widely-used binary digital token format in which secured claims are made about a subject

CWT makes largely parallel choices to JWT

But does not narrow COSE features used (unlike JWT)

- Does not mandate using COSE_Sign1 over COSE_Sign
- Does not mandate that only protected headers be used

Same claims extensibility model as JWT

WebCrypto

WebCrypto defines Web API for in-browser cryptographic operations

Only one way to perform any operation

Limited number of key formats using existing standards

-
  ```
  enum KeyFormat { "raw", "spki",
  "pkcs8", "jwk" };
  ```

Intentionally excludes functionality some wanted

- Use of platform keys
- Has led to non-standard extensions

WebAuthn/FIDO 2 is deployed unphishable login infrastructure supported by all modern browsers

W3C® fido® alliance

WebAuthn/ FIDO 2

Evolved from and replaces U2F/CTAP 1

- Resulted in multiple signature formats, some X.509-based, some bare

Multiple and evolving attestation formats

Numerous extensions with varying degree of implementation

- Which extensions will be ubiquitously supported is still TBD

W3C Verifiable Credentials

VCs represent cryptographically secured claims by an issuer about a subject

VC 1.0, 1.1, and 2.0 made different choices

- VC 2.0 not backwards compatible with previous versions

- **Two ways of signing VCs, each with sub-variants**
  - VC-JOSE-COSE supports JWS, COSE, and SD-JWT signatures over JSON-LD payload
  - VC-DATA-INTEGRITY canonicalizes JSON-LD payload, converts it to RDF N-Quads, and signs over the RDF (or can use JCS [RFC 8785])

Decentralized Identifiers (DIDs)

DIDs are a framework for identifiers about subjects not dependent upon central authorities

Each kind of DID has its own DID method and algorithms

DID spec defines operations that DID Methods must implement

As of this writing, there are 193 registered DID Methods!

- None are mandatory to implement, giving no interop guarantee
- DID Methods are out of scope for the newly rechartered DID WG!

Multiformats

Defines multiplicity of encodings for binary data

The Multibase spec defines 23 equivalent and non-interoperable representations for the same data!

- base64*, base58*, base36*, base32* , hex*, decimal, base8, base2, binary
- Interop requires either implementing them all or profiles choosing some

Multiformats institutionalize the failure to make a choice!

***Warning:* Multiformats are used by VC-DATA-INTEGRITY and DIDs**

So bad, I wrote “Multiformats Considered Harmful” post!

Closing Remarks on Choices

Enabling layered protocols is a choice

- Ethernet packet types are identified by EtherType
- IPv4 protocols are identified by protocol number
- TCP protocols are identified by port number
- JWT types are identified by the “`typ`” header parameter
- These all enable higher-level protocols to be layered over them

Planning for evolution is a choice

- Sometimes it’s necessary for choices to change over time
- Particularly as the security threat landscape evolves
- For instance, enabling cryptographic agility is a must
- Which algorithms are secure changes over time
- Only supporting a fixed algorithm would be a bad choice!

Extensibility is a choice

All the specifications I've discussed have extensibility points

- Extensibility enables new features to be added
  - Such as adding DPoP to OAuth 2.0
- Extensibility enables new layered applications and protocols
  - Such as adding an ID Token to OAuth 2.0 for OpenID Connect

Use extension methods that don't break existing deployments

- For instance, the "If you don't understand it, you MUST ignore it" logic for OAuth 2.0 request parameters and JWT claims has served us well

Standards are about making choices,

so make good ones!

Thank you

This presentation and more are available at:
https://self-issued.info

MATTR