Identity First.

The Emerging JSON/REST-Based Identity Protocol Suite

Michael B. Jones

Identity Standards Architect

Microsoft

March 5, 2013

Background

- Identity interop requires agreement on data representations and protocols
  - Numerous existing standards
    - Kerberos, X.509, SAML, WS-*, OpenID 2.0, etc.
  - Using a variety of data representations
    - ASN.1, XML, custom binary formats
  - *None are ubiquitously adopted*

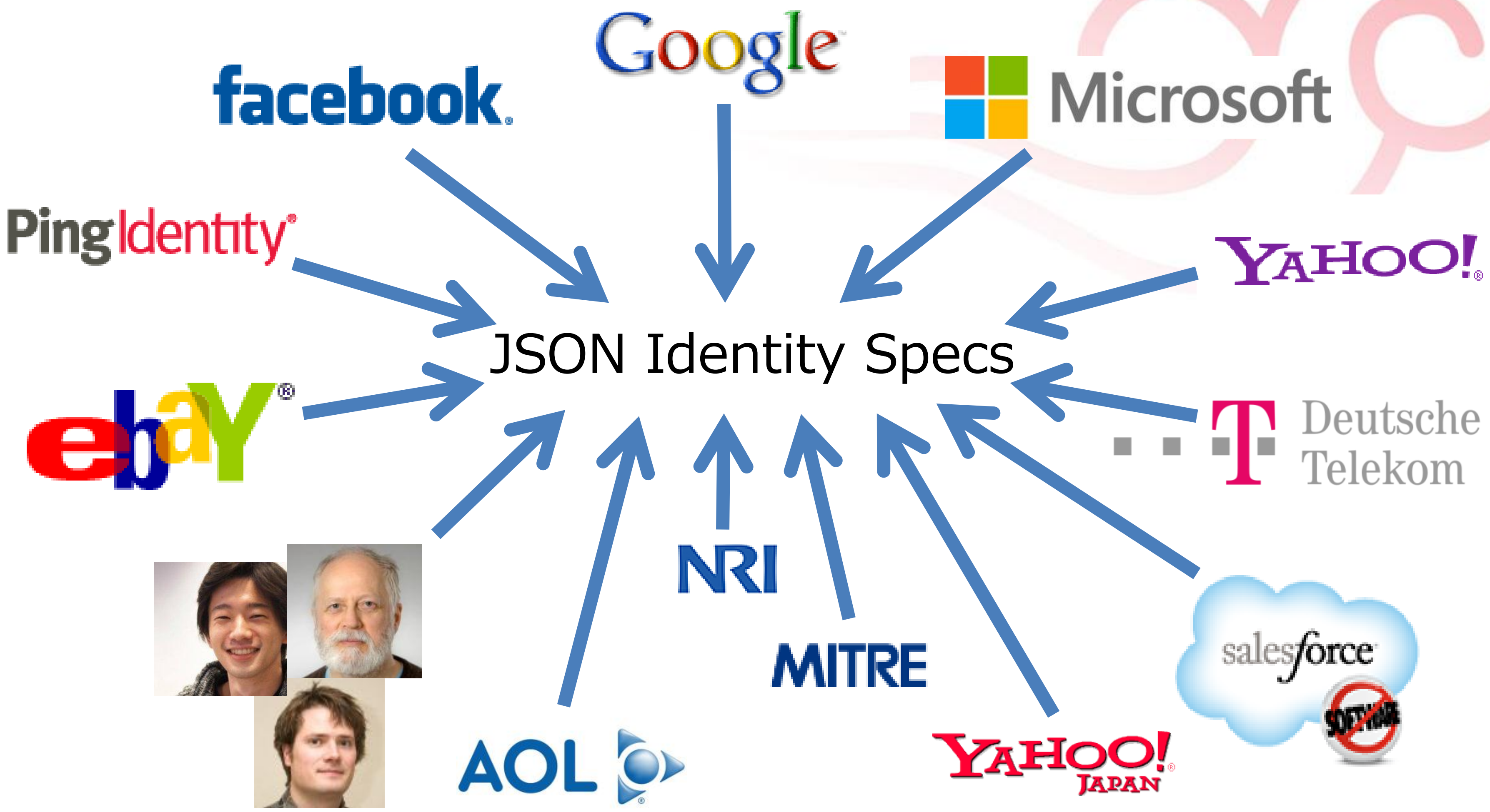
The Emerging JSON/REST-Based Protocol Suite

- A new suite of identity specifications is emerging
  - Using JSON data representations
  - Using REST design pattern
  - Reusing lessons learned from previous efforts
- Advantages
  - JSON ubiquitously supported in browsers and modern web development tools
    - Lets developers use tools they already have
  - Chance for much greater reach than past efforts
    - Increasing the scope of identity interop

Design Philosophy

- Make simple things simple
- Make complex things possible

Broad Set of Contributors



Presentation Overview

- Introduction
- Overview of the Specifications
- More Details on the Specifications with Examples
- Status of the Specifications and Interop Testing
- Resources and Discussion

Overview of the Specifications

JSON-Based Security Token Specification

- JSON Web Token (JWT)
  - JSON-based representation of signed and optionally encrypted claims
- JSON equivalent of the XML-based SAML Assertion
- JWT is pronounced like the English word "jot"

- *In IETF OAuth Working Group*

JSON-Based Cryptography Specifications

- JSON Web Signature (JWS)
  - JSON-based signature representation
- JSON Web Encryption (JWE)
  - JSON-based encryption representation
- JSON Web Key (JWK)
  - JSON representation for set of public keys
- JSON Web Algorithms (JWA)
  - Algorithms used with JWS, JWE, and JWK

- *In IETF JSON Object Signing and Encryption (JOSE) WG*

OAuth 2.0 Specifications

- OAuth 2.0 Authorization Framework – RFC 6749
  - Third party authorization protocol
- OAuth 2.0 Bearer Token Usage – RFC 6750
  - Using bearer tokens to access protected resources
- JWT Assertion Profiles for OAuth 2.0
  - Using a JWT to authenticate a client or request an access token
- OAuth 2.0 Dynamic Client Registration
  - Protocol for dynamically registering OAuth clients with authorization servers

- *In IETF OAuth Working Group*

WebFinger Discovery Specification

- Discover information associated with an identifier
  - For identifiers such as e-mail addresses and URLs
  - Discovering information such as URLs of service endpoints
- With an HTTPS GET
- Using simple JSON response

- *In IETF Applications Area Working Group*

OpenID Connect

- Simple identity protocol built on top of OAuth 2.0
- Enables sign-in and release of information about the end-user
  - Like Facebook Connect, but with an open set of providers
- Works well on mobile phones
- Works for both native and Web-based applications
- Works across a range of security profiles
- Modular design
  - Lets you build only the parts you need
- Underpinnings
  - OAuth 2.0, JWT, JWS, JWE, JWK, JWA, and WebFinger

- *In OpenID Foundation Artifact Binding/Connect Working Group*
- *OpenID Connect is a cornerstone of the emerging JSON/REST-based identity protocol suite, but is not a subject of this presentation*

More Details on the Specifications with Examples

JSON Web Token (JWT)

- http://tools.ietf.org/html/draft-ietf-oauth-json-web-token
- JSON-based security token format
  - Claims passed as signed & optionally encrypted JSON object
- Compact, URL-safe representation
  - Enabling use in URI query parameters, fragment values

JWT ID Token Example from OpenID Connect

- JWT representing information about an authentication event
- Claims:
  - "`iss`" – Issuer
  - "`sub`" – Subject – Identifier for the end-user at the issuer
  - "`aud`" – Audience for the ID Token
  - "`exp`" – Expiration time
  - "`nonce`" – Replay attack mitigation

ID Token Claims Example

```
{
 "iss": "https://server.example.com",
 "sub": "248289761001",
 "aud": "https://client.example.org",
 "exp": 1311281970,
 "nonce": "n-0S6_WzA2Mj"
}
```

JSON Web Signature (JWS)

- http://tools.ietf.org/html/draft-ietf-jose-json-web-signature
- Sign arbitrary content using compact JSON-based representation
  - Includes both true digital signatures and HMACs
- Representation contains three parts:
  - Header
  - Payload
  - Signature
- Parts are base64url encoded and concatenated, separated by period ('.') characters
  - URL-safe representation

- *JWS is used to sign JWTs*

JWS Header Example

- JWS Header:

```
{"typ":"JWT",
 "alg":"HS256"}
```

  - Specifies use of HMAC SHA-256 algorithm
  - Also contains optional type parameter
- Base64url encoded JWS Header:
  - `eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9`

JWS Payload Example

- JWS Payload (before base64url encoding):

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

- JWS Payload (after base64url encoding):
  - eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQog
    Imh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ

JWS Signing Input

- Signature covers both Header and Payload
- Signing input is the concatenation of encoded Header and Payload, separated by a period ('.') character
  - Enables direct signing of output representation
- Example signing input:

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiO
iJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9
leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

JWS Signature

Identity First.

- Example base64url encoded HMAC SHA-256 value:

  `dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk`

Complete JWS Example

- Header.Payload.Signature:

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiO
iJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9
leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ.dBjftJeZ4CVP-m
B92K27uhbUJU1p1r_wW1gFWFOEjXk
```

- Representation is compact, URL-safe
- Enables, JWS/JWT values to be passed as URI query parameters or fragment values

JWS Algorithm Identifiers

- Compact algorithm (`"alg"`) identifiers:
  - `"HS256"` – HMAC SHA-256
  - `"RS256"` – RSA SHA-256
  - `"ES256"` – ECDSA with P-256 curve and SHA-256
- Other hash sizes also defined:
  - 384, 512
- Other algorithms, identifiers MAY be used

- Defined in JSON Web Algorithms (JWA) Specification
  - http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms

JSON Web Encryption (JWE)

- http://tools.ietf.org/html/draft-ietf-jose-json-web-encryption
- Encrypt arbitrary content using compact JSON-based representation
  - With either asymmetric or symmetric keys
- Representation contains five parts:
  - Header
  - Encrypted Key
  - Initialization Vector
  - Ciphertext
  - Integrity Value
- Parts are base64url encoded and concatenated, separated by period ('.') characters
  - Compact, URL-safe representation

- *JWE is used to encrypt JWTs*

JWE Header Example

- JWE Header:

```
{"alg":"RSA1_5",
 "enc":"A128CBC+HS256"}
```

  - Block encryption key encrypted with RSAES-PKCS1-V1_5
  - Plaintext encrypted with AES-CBC using a 128 bit key
  - Integrity value for result calculated with HMAC SHA-256
- Header base64url encoded, just like JWS

JWE Key Encryption & Agreement Algorithms

- Algorithm (`"alg"`) identifiers:
  - `"RSA1_5"` – RSAES-PKCS1-V1_5
  - `"RSA-OAEP"` – RSAES using Optimal Asymmetric Encryption Padding (OAEP)
  - `"A128KW"`,`"A256KW"` – AES Key Wrap with 128, 256 bit keys
  - `"ECDH-ES"` – Elliptic Curve Diffie-Hellman Ephemeral Static
  - `"dir"` – Direct symmetric encryption without a wrapped key
- Other algorithms, identifiers MAY be used

- Defined in JSON Web Algorithms (JWA) Specification
  - http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms

JWE Plaintext Encryption Algorithms

- Encryption Algorithm (`"enc"`) identifiers:
  - `"A128CBC+HS256"`,`"A256CBC+HS512"` – AES Cipher Block Chaining (CBC) mode with 128/256 bit keys and integrity provided by HMAC SHA-256/HMAC SHA-512
  - `"A128GCM"`,`"A256GCM"` – AES Galois/Counter Mode (GCM) with 128/256 bit keys
- Other algorithms, identifiers MAY be used

- Defined in JSON Web Algorithms (JWA) Specification
  - http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms

JSON Web Key (JWK)

- http://tools.ietf.org/html/draft-ietf-jose-json-web-key
- JSON representation of a public key or set of keys

- Also recently extended to represent private and symmetric keys
  - http://tools.ietf.org/html/draft-jones-jose-json-private-and-symmetric-key

JWK Example

```
{"keys":
  [
    {"kty":"EC",
     "crv":"P-256",
     "x":"MKBCTNIcKUSDii11ySs3526iDZ8AiTo7Tu6KPAqv7D4",
     "y":"4Etl6SRW2YiLUrN5vfvVHuhp7x8PxltmWWlbbM4IFyM",
     "use":"enc",
     "kid":"1"},

    {"kty":"RSA",
     "n":"0vx7ag...(omitted)...Cur-kEgU8awapJzKnqDKgw",
     "e":"AQAB",
     "kid":"2011-04-29"}
  ]
}
```

JWK Key Type Identifiers

- Key type (`"kty"`) identifiers for asymmetric keys:
  - `"EC"` – Elliptic Curve key
  - `"RSA"` – RSA key
- Key type identifier for symmetric keys:
  - `"oct"` – Octets representing a bare key

- Types for asymmetric keys defined in JSON Web Algorithms (JWA):
  - http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms
- Type for symmetric keys defined in:
  - http://tools.ietf.org/html/draft-jones-jose-json-private-and-symmetric-key

What is OAuth?

- REST/JSON-based authorization protocol
- Enables a resource owner to authorize limited access to resources by specific applications
  - Without sharing passwords with the applications

Example OAuth Scenario

- Mary has Microsoft and Facebook accounts
- Mary wants to display her Facebook photos on her Microsoft pages
- Mary authorizes Microsoft to have read-only access to her Facebook photos using OAuth 2.0
  - She uses her Facebook password at Facebook to enable this authorization
  - Facebook grants a scoped access token to Microsoft
  - Microsoft uses the access token to retrieve Mary’s photos for her

Primary OAuth Specifications

- OAuth 2.0 Authorization Framework
  - Defines OAuth 2.0 flows
  - http://tools.ietf.org/html/rfc6749
- OAuth 2.0 Bearer Token Usage
  - Defines how to use Bearer Tokens to access protected resources
    - Bearer tokens can be used by any party in possession of them to access the corresponding protected resources
  - http://tools.ietf.org/html/rfc6750

OAuth Assertions Specifications

- OAuth 2.0 Assertions Profile
  - Defines how to use assertions as client credentials and authorization grants
    - In manner independent of assertion/token type
  - http://tools.ietf.org/html/draft-ietf-oauth-assertions
- OAuth 2.0 SAML Assertion Profiles
  - Defines how to use SAML assertions with OAuth 2.0
  - http://tools.ietf.org/html/draft-ietf-oauth-saml2-bearer
- OAuth 2.0 JWT Assertion Profiles
  - Defines how to use JWTs with OAuth 2.0
  - http://tools.ietf.org/html/draft-jones-oauth-jwt-bearer

- *The JWT Assertion Profiles spec is used by OpenID Connect*

Additional OAuth Specifications

- OAuth 2.0 Dynamic Client Registration
  - Protocol for dynamically registering an OAuth client
  - http://tools.ietf.org/html/draft-ietf-oauth-dyn-reg
- OAuth 2.0 Token Revocation
  - Defines how to voluntarily relinquish access
  - http://tools.ietf.org/html/draft-ietf-oauth-revocation
- OAuth 2.0 JSON Requests
  - Specifies JSON format encoding for OAuth requests
  - http://tools.ietf.org/html/draft-sakimura-oauth-requrl

- *OAuth* 2.0 *Dynamic Client Registration is used by OpenID Connect*

WebFinger Discovery Specification

- http://tools.ietf.org/html/draft-ietf-appsawg-webfinger
- Discover information associated with an identifier
- Uses these HTTP request parameters:
  - "`resource`"– Identifier for which discovery is being performed
    - Such as "`acct:mary@example.com`"
  - "`rel`" – Link relation type being queried for
    - Such as "`http://openid.net/specs/connect/1.0/issuer`"
- Response contains list of JSON links with discovered information
  - Such as:

```
{"rel": "http://openid.net/specs/connect/1.0/issuer",
 "href": "https://server.example.com"}
```

- *WebFinger is used by OpenID Connect*

Example WebFinger Request and Response

- Example WebFinger Request

```
GET /.well-known/webfinger?
  resource=acct%3Amary%40example.com&
  rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: example.com
```

- Example WebFinger Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/jrd+json

{
  "subject" : "acct:mary@example.com",
  "links" :
  [
    {
      "rel" : "http://openid.net/specs/connect/1.0/issuer",
      "href" : "https://server.example.com"
    }
  ]
}
```

Status of the Specifications and Interop Testing

Completed Specifications

- OAuth Authorization Framework – RFC 6749
- OAuth Bearer Token Usage – RFC 6750

Specifications Close to Completion

- JSON Web Token (JWT)
- JSON Web Signature (JWS)
- JSON Web Encryption (JWA)
- JSON Web Key (JWK)
- JSON Web Algorithms (JWA)
- OAuth Assertion Framework
- OAuth SAML Assertion Profiles
- OAuth JWT Assertion Profiles
- OAuth Dynamic Client Registration
- WebFinger
- OpenID Connect

Specifications Farther from Completion

- OAuth Token Revocation
- OAuth JSON Requests
- JSON Private and Symmetric Key Representation
- JWE Encryption of JWKs
- JWK PKIX (X.509) Key Representation

- *None are used by OpenID Connect*

Identity First.

Interoperability Testing

- Interop testing is ongoing for all the specs used by OpenID Connect:
  - JWT, JWS, JWE, JWK, JWA, OAuth (RFC 6749 and RFC 6750), OAuth JWT Assertions, OAuth Dynamic Registration, WebFinger
  - And the OpenID Connect specifications themselves
- Currently in 4th round of interop testing
  - See http://osis.idcommons.net/
- Interop mailing list:
  - http://groups.google.com/group/openid-connect-interop
- By the numbers:
  - 14 implementations participating
    - 66 cross-solution test results recorded
  - 110 feature tests defined
    - 1,260 feature test results recorded
  - 101 members of interop mailing list
    - 860 messages to interop mailing list

Resources and Discussion

Resources and Discussion

- IETF OAuth Working Group & Mailing List
  - http://datatracker.ietf.org/wg/oauth
  - https://www.ietf.org/mailman/listinfo/oauth
- IETF JSON Object Signing and Encryption (JOSE) WG & Mailing List
  - http://datatracker.ietf.org/wg/jose
  - https://www.ietf.org/mailman/listinfo/jose
- IETF WebFinger WG & Mailing List
  - http://datatracker.ietf.org/wg/appsawg
  - https://www.ietf.org/mailman/listinfo/webfinger
- OpenID Connect WG & Mailing List
  - http://openid.net/connect
  - http://lists.openid.net/mailman/listinfo/openid-specs-ab
- My Blog and Twitter Handle
  - http://self-issued.info/
  - @selfissued