Sender Constrained JWT for OAuth 2.0

-- https://www.ietf.org/id/draft-sakimura-oauth-rjwtprof-05.txt

Nat Sakimura

Kepeng Li

Background

- OAuth PoP Security Architecture talks about
  - Security threats
    - Token manufacture/modification
    - Token disclosure
    - Token redirect
    - Token reuse
  - Possible ways to alleviate security threats
    - Confidentiality protection
    - Sender Constraint ← Not written in POP Key Semantics.
    - Key confirmation
- Client Authentication @ Resource Server out of scope of POP Key Semantics.
  - But, we need it, do not we?

This draft was first written as the response to the WGLC for POP Key Semantics.

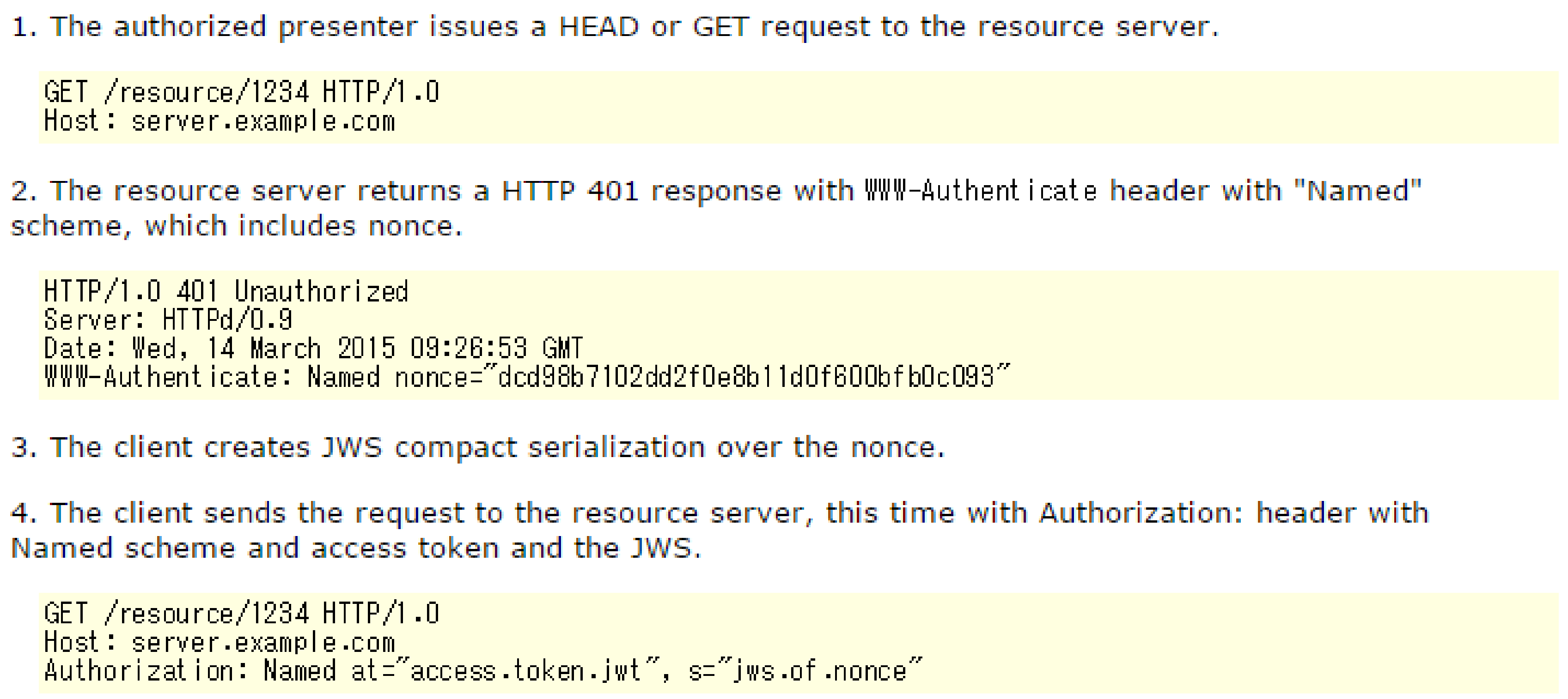
4. Sender Constraint Representation

- Include Client ID in the JWT payload
- Example:

```
{
        "iss": "https://server.example.com",
        "sub": "joe@example.com",
        "azp": "https://client.example.org",
        "aud": "https://resource.example.org",
        "exp": "1361398824",
        "nbf": "1360189224",
}
```

- Note that RS MUST authenticate the Client.

5. Client Authentication


1. The authorized presenter issues a HEAD or GET request to the resource server.

```
GET /resource/1234 HTTP/1.0
Host: server.example.com
```

2. The resource server returns a HTTP 401 response with `WWW-Authenticate` header with "Named" scheme, which includes nonce.

```
HTTP/1.0 401 Unauthorized
Server: HTTPd/0.9
Date: Wed, 14 March 2015 09:26:53 GMT
WWW-Authenticate: Named nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093"
```

3. The client creates JWS compact serialization over the nonce.

4. The client sends the request to the resource server, this time with Authorization: header with Named scheme and access token and the JWS.

```
GET /resource/1234 HTTP/1.0
Host: server.example.com
Authorization: Named at="access.token.jwt", s="jws.of.nonce"
```

6. Finding Client Key

- **6.1. URI client ID**
  - When the Client ID is a URI, then the key can be found from the . well-know/jwk URI.
- **6.2. pre-shared key tables**
  - Alternatively, the collection of the keys can be pre-shared among the participants in advance as a key table that lists the client ID - public key pair.
- **6.3. Via client metadata API of the authorization server**
  - Client Metadata can be exposed through a client metadata API at the Authorization Server, which can be defined by the authorizati on server in a way similar to OAuth 2.0 Token Introspection.

Questions

Should we merge into

- PoP Security Architecture draft?
  - https://www.ietf.org/id/draft-ietf-oauth-pop-architecture-02.txt
- Or to Proof-of-Possession Key Semantics for JSON Web Token s (JWTs)?
  - https://tools.ietf.org/html/draft-ietf-oauth-proof-of-possession-03

Or proceed as a separate document?

Or is it a bad idea that we should throw it away?