A SASL and GSS-API Mechanism for OAuth

draft-ietf-kitten-sasl-oauth-00

William Mills
Tim Showalter
Hannes Tschofenig
Status

draft-mills-kitten-sasl-oauth

draft-ietf-kitten-sasl-oauth

Indiv-00
Indiv-01
Indiv-02
Indiv-03
Indiv-04

 WG item -00

IETF#78 IETF#79 IETF#80 IETF#81 IETF#82

Status, cont.

- draft-ietf-kitten-sasl-oauth-00.txt submitted as WG item this Monday.
  - Content of draft-mills-kitten-sasl-oauth-04 and draft-ietf-kitten-sasl-oauth-00.txt identical!

- -04 version saw a number of changes:
  - Abstract and Introduction modified
  - Sections restructured.
  - References updated
  - Editorial bugfix
Reminder: Architecture

(A) Authorization Request ---> Resource
    Owner

<- (B) Access Grant

Client Credentials &

(C) Access Grant ---> Authorization
    Server

<- (D) Access Token
    (w/ Optional Refresh Token)

(Optional discovery)

(1) User Name

<- (2) Authentication
    endpoint information

(E) Access Token

<- (F) Protected Resource

OAuth

Plain

SASL/

GSS-

API

OAuth 2.0
Design Decisions

• (1) OAuth Integration into SASL
  a) HTTP-Style
  b) Native

• (2) Security Functionality Scope
  a) Bearer Token Support
  b) MAC security
  c) Public Key security

• (3) Discovery
  a) Part of OAuth SASL specification
  b) Independent mechanism
OAuth Integration into SASL
(a) HTTP-based Style

GET / HTTP/1.1
Host: server.example.com
User: user@example.com
Authorization: MAC token="h480djs93hd8", timestamp="137131200", nonce="dj83hs9s", signature="YTVjyNSujYs1WsDurFnvFi4JK6o="

GET / HTTP/1.1
Host: imap.example.com
Authorization: Bearer "vF9dft4qmTc2Nvb3RIckBhbHRhdmIzdGExY29tCg=="
OAuth Integration into SASL
(b) Native

TOKEN= "vF9dft4qmTc2Nvb3RlckBhbHRhdmIzdGEuY29tCg==",SCOPE="foo"
OAuth Integration into SASL

• **HTTP-Style**
  – Re-uses HTTP parsing library, re-uses OAuth specifications as much as possible, similar to tunneling request/responses defined in draft-ietf-kitten-sasl-saml-05 and in draft-ietf-kitten-sasl-saml-ec-00.txt

• **Native**
  – Requires more specification work to decide about defining request and response (including error parameters).
  – May require less code.
Security Functionality Scope

- OAuth WG develops two HTTP-based security variants:
- Other options: RFC 5849 (RSA signature method) or draft-balfanz-tls-obc-00 - TLS Origin-Bound Certificates
- What security mechanisms should be specified?
Discovery

- Certain OAuth profiles do not require OAuth support by the end host.
- Impacts design of discovery mechanism.
  - Proposals for discovery being discussed in the OAuth WG. E.g., draft-jones-simple-web-discovery.
- OAuth SASL is in a different position since end host changes are needed anyway.
  - A discovery mechanism could be incorporated into OAuth SASL.
Discovery, cont.

Example from http://openid.net/specs/openid-connect-discovery-1_0.html

```json
{
  "authorization_endpoint": "https://example.com/connect/authorize",
  "issuer": "https://example.com",
  "token_endpoint": "https://example.com/connect/token",
  "user_info_endpoint": "https://example.com/connect/user",
  "check_id_endpoint": "https://example.com/connect/check_id",
  "refresh_session_endpoint": "https://example.com/connect/refresh_session",
  "end_session_endpoint": "https://example.com/connect/end_session",
  "jwk_document": "https://example.com/jwk.json",
  "registration_endpoint": "https://example.com/connect/register",
  "scopes_supported": ["openid"],
  "flows_supported": ["code", "token"],
  "iso29115_supported": ["http://www.idmanagement.gov/schema/2009/05/icam/openid-trust-level1.pdf"],
  "identifiers_supported": ["public", "ppid"]
}
```