Chain Extension Proposal

Phil Hunt
January 26, 2011
phil.hunt@yahoo.com
Introduction

• Some web service customers has raised the issue of passing both user credential (UC) and client app context (AC) in HTTP request/responses
  – User --1--> Client --2--> REST ---3--> Service --4--> IAM/AAA infrastructure
Issues

- How to propagate user auth context though multiple service pairs and security domains?
- How to support HTTP level exchange of creds (as opposed to SOAP based)
- Performance – must keep lightweight
  - Support very high rate of app transactions
- Portability
  - Service providers may be in separate admin zones
  - May be multi-vendor
Chain Proposal

• Extend token endpoint to allow foreign access tokens to be exchanged for new ‘local’ tokens
• Depends on:
  – Ability of one domain token server to understand another’s access token.
    • Standard token format (e.g. some profile of JWT)
  – Pair-wise trust between domains
Terminology

• Glossary
  – Security Context – an abstract concept that refers to an established authentication state
  – Security Context Token – a representation of a security context
  – Signed Security Token – a signed security token (e.g. JWT)
  – CT – Type of signed security token representing client applications (may also be client credential)
  – UT – Type of signed security token representing users
  – AT – A type of extensible signed security token usually including at least one client security context and one user security context (aka access token)
Observations

• Originating user
  – There is desire to track original user context
  – Originating client node has delegation from user
  – Subsequent nodes proceeding under own authority plus original/pair-wise authorization

• Pair-wise Trust
  – Each SP must trust previous SP node as Client
  – Signing authority
    • Client’s authenticator/token service (fed model)
    • Client directly (via SP’s authenticator service)
Proposed Chain Flow

Note: this flow uses a client credential based on SAML IDPs for clients and users. Normal client_id/client_secret could also be used.
Client Obtains CT

- Client App Authenticates with IDP
  - SAML Authentication Assertion returned
Client obtains its token

• SAML Assertion Exchanged for Token ($C_1T$)
  – One-time

Note: client could use client_id/client_secret
• End-User Authenticated and AuthZ obtained
  – OAuth ‘grant code’ or SAML Bearer assertion (UA) returned
• End-User Authenticated and AuthZ obtained
  – OAuth ‘grant code’ or SAML Bearer assertion (UA) returned
Access token combines contexts

- End-User Authenticated and AuthZ obtained
  - OAuth ‘grant code’ or SAML Bearer assertion (UA) returned

Diagram:
- IDP Server
- Token Server
- Client App (User Profile)
- REST Svc (OVD)
- Web Svc (OIM)

Token Request Response
Access Token(AT₁)=«C₁T +UT»
Normal OAuth Access Request

- End-User Authenticated and AuthZ obtained
  - OAuth ‘grant code’ or SAML Bearer assertion (UA) returned

```
IDP Server

Token Server

Client App
(User Profile)

REST Svc
(OVD)

Web Svc
(OIM)

REST Svc Request
Authorization: AT₁ «C₁T+UT»
```
Chained AT Request

- End-User Authenticated and AuthZ obtained
  - OAuth ‘grant code’ or SAML Bearer assertion (UA) returned

Client 2 obtains/calcs combined token $AT_2^{C_2T+C_1T+UT}$
Chained Request

- End-User Authenticated and AuthZ obtained
  - OAuth ‘grant code’ or SAML Bearer assertion (UA) returned

HTTP Request
Authorization: AT₂ «C₂T+C₁T+UT»
Comments

• Chaining may not be required if resources in common domain
• Does allow bridging between federated resources
• Expensive for single-operations
• Inexpensive when more then one request per client
• Does not replace functionality of WS-SecureConversation (e.g. message protection)
• Suitable for REST based, lightweight scenarios where performance is an issue.