From OAuth1 to OAuth2 with Apache CXF and Hawk

Sergey Beryozkin, Talend

APACHECON EUROPE

CORINTHIA HOTEL
BUDAPEST, HUNGARY
NOVEMBER 17-21, 2014

What is Apache CXF ?

- Production quality Java framework for developing REST and SOAP web services
- CXF 3.0.2: JAX-RS 2.0, JAX-WS 2.2
- Major focus on the web services security: WS-Security, OAuth1/2, JOSE, immediate and public reaction to security issues
- Active community, healthy project environment

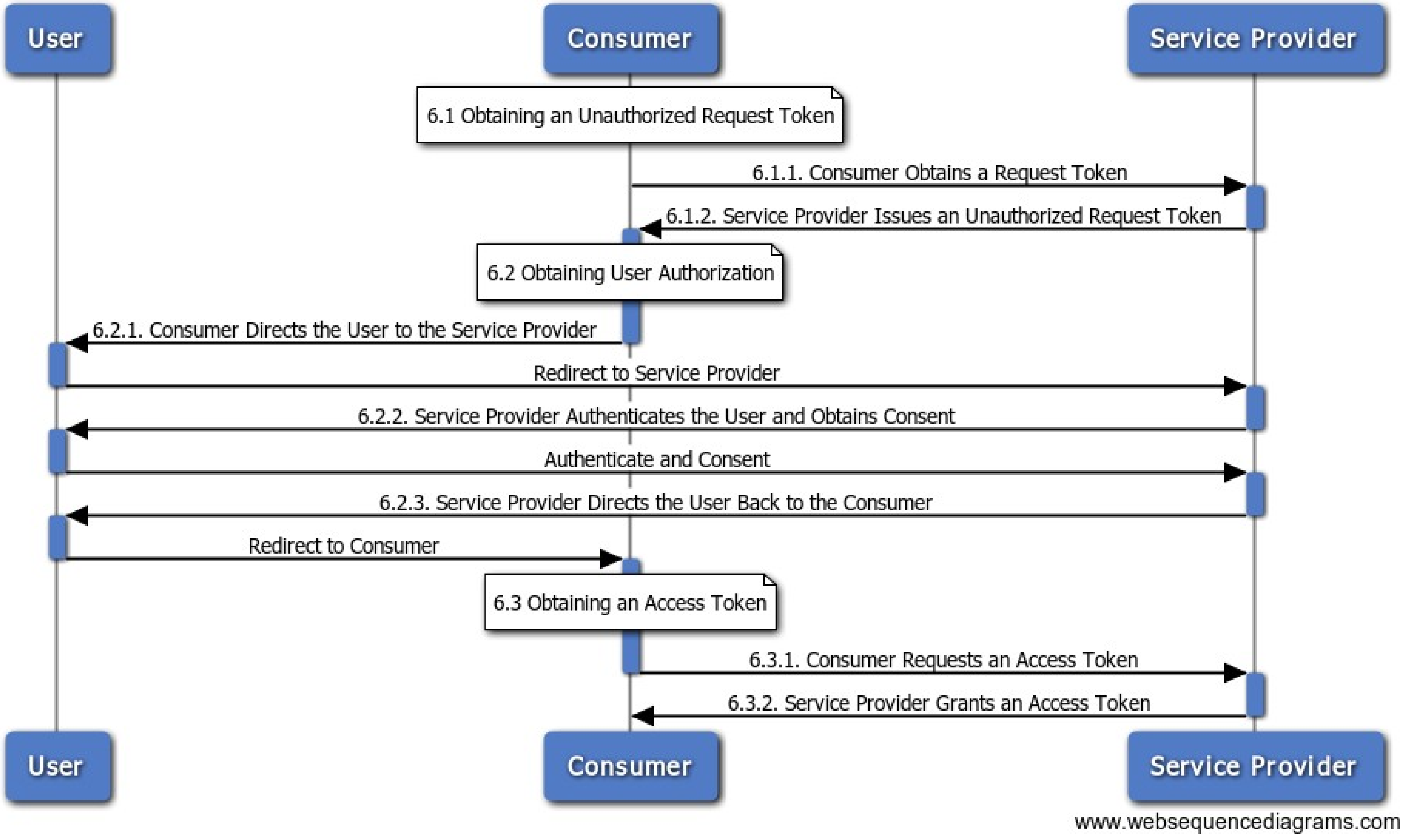
What is OAuth ?

- Allows third party clients such as web servers or mobile applications to access server resources on behalf of their owner
- Owners authorize the access via the redirection without sharing their secrets
- Major theme in the HTTP services world: drives relevant innovations, popularises the subject of web security, helps enrich the applications

History of OAuth

- 1.0: Eran Hammer-Lahav, RFC 5849, Apr 2010, implemented by many providers
- 2.0: The working group starts its work, Eran joins and eventually leaves
- 2.0: RFC 6749 is released in Oct 2012
- 2.0: Actively supported, many related enhancements are being developed
- The 1.0 vs 2.0 controversy is lingering

OAuth1 Diagramm



User
Consumer
Service Provider
6.1 Obtaining an Unauthorized Request Token
6.1.1. Consumer Obtains a Request Token
6.1.2. Service Provider Issues an Unauthorized Request Token
6.2 Obtaining User Authorization
6.2.1. Consumer Directs the User to the Service Provider
Redirect to Service Provider
6.2.2. Service Provider Authenticates the User and Obtains Consent
Authenticate and Consent
6.2.3. Service Provider Directs the User Back to the Consumer
Redirect to Consumer
6.3 Obtaining an Access Token
6.3.1. Consumer Requests an Access Token
6.3.2. Service Provider Grants an Access Token
User
Consumer
Service Provider
www.websequencediagrams.com

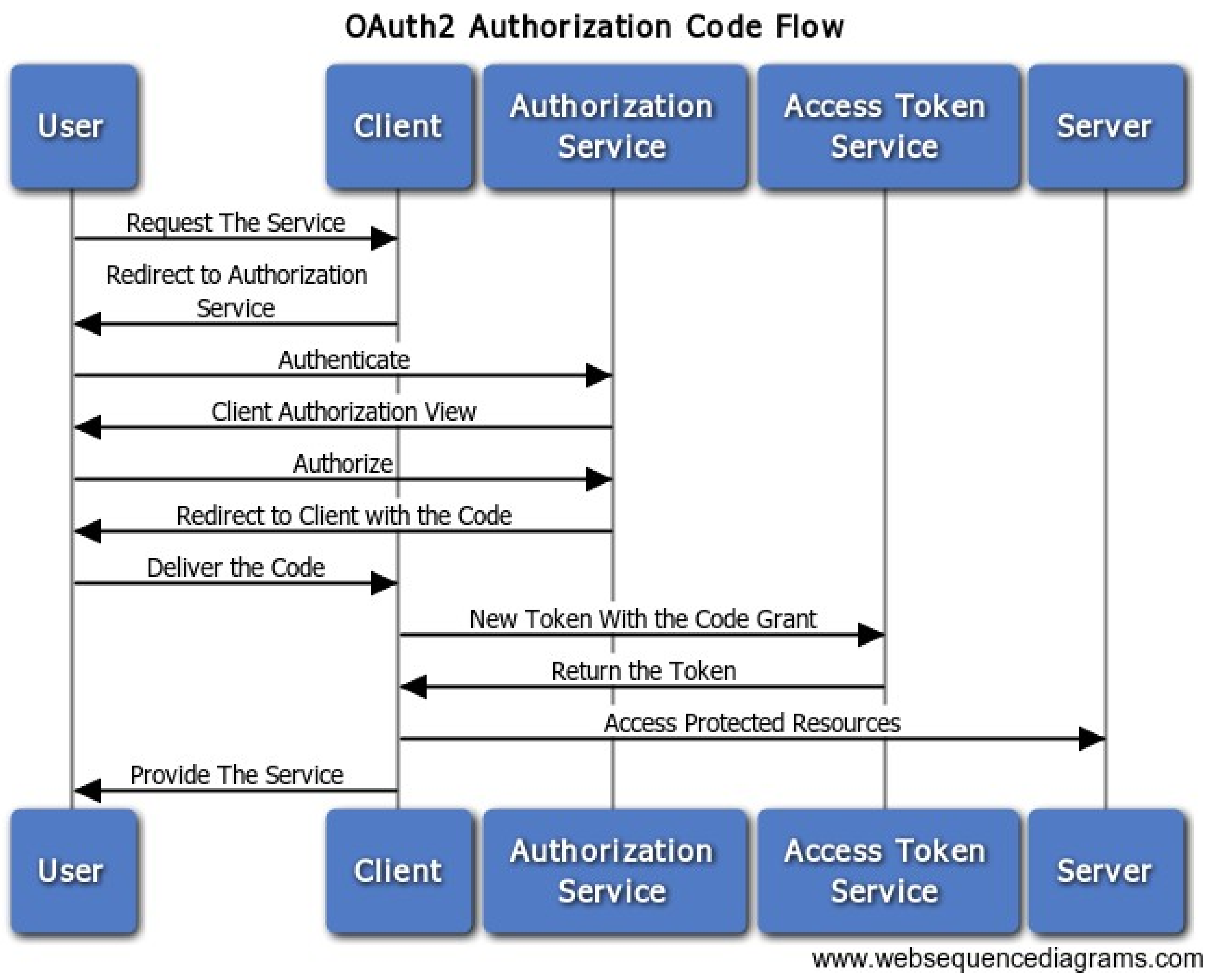
Key OAuth1 Features

- Classic flow requires a 3-step 'dance': getting a temp request token, getting an authorization verifier, exchanging the temp token and the verifier for the access token
- Support for Proof Of Possession and the 'best effort' data and replay protection with the clients using its secret and token keys to create a signature

OAuth1 Pros, Cons and Praise

- Proved to be functional and popular, opened a new chapter in the world of secure HTTP services: Great Effort !
- PoP, data integrity and replay protection
- 3-step dance is complex, simpler flows are not standardized
- Only SHA1 signature algorithms; keys are sent over TLS but only in plain text

OAuth2 Code Diagramm



OAuth2 Authorization Code Flow
User
Client
Authorization Service
Access Token Service
Server
Request The Service
Redirect to Authorization Service
Authenticate
Client Authorization View
Authorize
Redirect to Client with the Code
Deliver the Code
New Token With the Code Grant
Return the Token
Access Protected Resources
Provide The Service
www.websequencediagrams.com

Brief OAuth2 Overview

- Authorization code flow is simpler than OAuth1: a step involving a temp token request is dropped
- Many flows, grant and token types
- Some flows require the extra care (implicit flow), no PoP from the get go
- OAuth2 drives a lot of the innovation (OIDC, can utilize JOSE, etc), it will stay

From OAuth1 to OAuth2

- Developers who like OAuth1 value the PoP feature but OAuth2 does not have a standardized PoP scheme yet… (wait for a later slide though :-))
- Actually, Eran did author a MAC token draft before he left the OAuth2 group…
- OAuth2 is very extensible – non standard authentication schemes are OK, so…

What is Hawk

- Eran and others did work on the MAC scheme and how it can be used with OAuth2 (draft-hammer-oauth-v2-mac-token-05, see Links)
- Hawk has its roots in that spec; it is a new scheme, better version of OAuth1 scheme; documented not to be related to OAuth2, no reason not to use it when migrating to OAuth2 though :-)

What does Hawk Client do ?

- The Hawk client gets a secret (MAC) key out-of-band
- The Hawk client creates a Hawk scheme: “Authorization: Hawk id="...", ts="...", nonce="...", mac="..."”
- The sequence capturing various request properties, a body hash, is signed
- hueniverse/hawk at GitHub for more info

OAuth2 Access Token and Hawk

- “{ “access_token”:”123”, “token_type”:”hawk”, “secret”:”678” }”
- Authorization: Hawk id=”123” mac=”...”
- OAuth2 'access_token' -> Hawk 'id'
- OAuth2 'secret' -> is distributed to the client as part of the token response and used to calculate a Hawk 'mac' hash
- OAuth2 PoP will work, Hawk is here now.

Apache CXF and OAuth2

- OAuth2 runtime encapsulates most of the work a typical OAuth2 server will do.
- AuthorizationCode, ImplicitGrant and AccessToken JAX-RS services; pluggable grant and session handlers, validators, token and code response post-processors
- Developers are mainly focused on getting the data stored only

Apache CXF, OAuth2 and Hawk

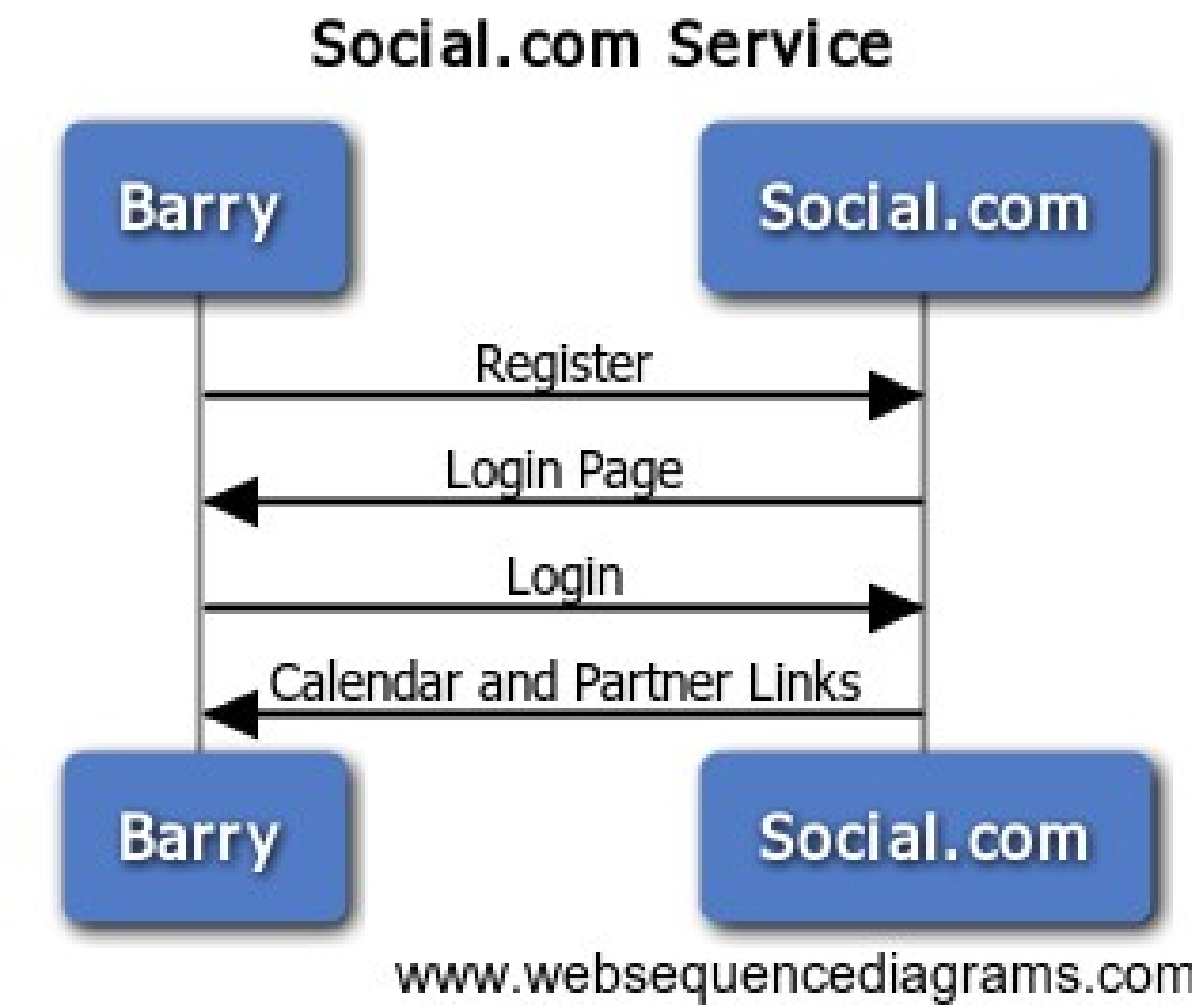
- Server:

  ServerAccessToken token = new HawkAccessToken(...HmacSHA256);

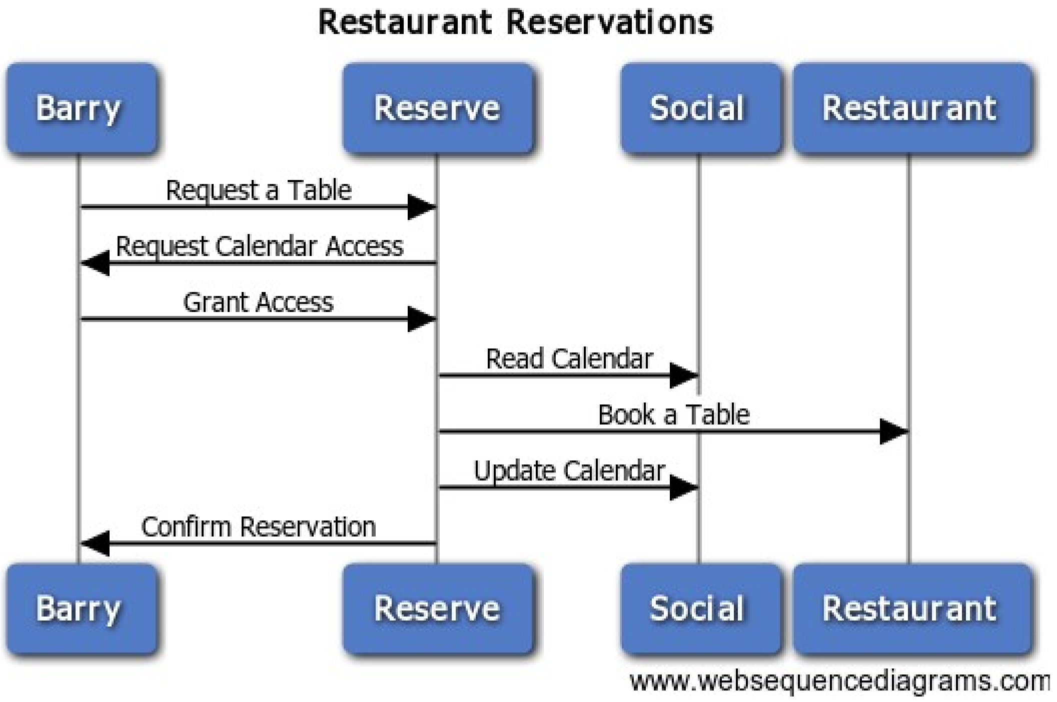
- Client: calculates the hash with the help of the Client utilitity code

  (Code example...)

The Demo


Social.com Service
Barry
Social.com
Register
Login Page
Login
Calendar and Partner Links
Barry
Social.com
www.websequencediagrams.com

The Demo Continued



OAuth2 and PoP: the latest

- Draft-bradley-oauth-pop-key-distribution-01: symmetric and asymmetric PoP keys, keys are JWK formatted, Hmac, RSA-SHA, Elliptic key signatures
- PoP keys can be JWE-encrypted
- Draft-richer-oauth-signed-http-request-01 – how the signatures can be done
- More sophisticated and capable PoP

OAuth2 and PoP: alternatives

- Use 2-way TLS (client certificates) to authenticate
- Use JWS to protect the integrity of the actual payload
- Use JWE to protect the sensistive content
- Combine TLS, JWE and JWS if really needed

Additional Resources

- More about CXF Security at Apache Con, 17 Nov:

  Dennis Sosnoski, “CXF Security and Reliability”, 13.40

  Andrei Shakirin, “Secure Services with Apache CXF”, 16.50
- CXF: http://cxf.apache.org/docs/jax-rs-oauth2.html
- Hawk: https://github.com/hueniverse/hawk
- OAuth2 PoP: http://tools.ietf.org/html/draft-bradley-oauth-pop-key-distribution-01

Questions ?

Thank You