enservio

Retrofitting OAuth 2.0 Security into Existing REST Service [CON1765]

Irena Shaigorodsky

Java One, 2014

ishaigorodsky@enservio.com

@ishaigorodsky

https://github.com/ishaigor/rest-retro-sample

Quick Survey

- How many
  - ☐Use or plan to use rich REST based UI for sensitive information?
  - ☐Know what OAuth is?
  - ☐Use or plan to use rich REST based UI with OAuth?
  - ☐Designed rich REST based UI with OAuth in mind before the audit?
  - ☐Use spring/spring-security/spring-security-Oauth?

Agenda

- Security Cost
- OAuth 2.0
- Sample deep-dive

Why My Company Needs Security?

- Cost of security breach in US[1]
  - $188 per record
  - average size: 28,765 records
  - customer loss
- Customer driven

[1] 2013 Cost of Data Breach Study: Global Analysis by Ponemon Institute© sponsored by Symantec

OAuth 2.0

"An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications."[1]

"The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service."[1]

[1] http://oauth.net/

OAuth 2.0 Lingo

- Resource
  - Resource Owner
  - Resource Server
- OAuth 2.0 scope
- OAuth 2.0 client
- Endpoints
  - Authorization Endpoint
  - Token Endpoint
- Tokens
  - Access Token
  - Refresh Token
- Authorization Grant

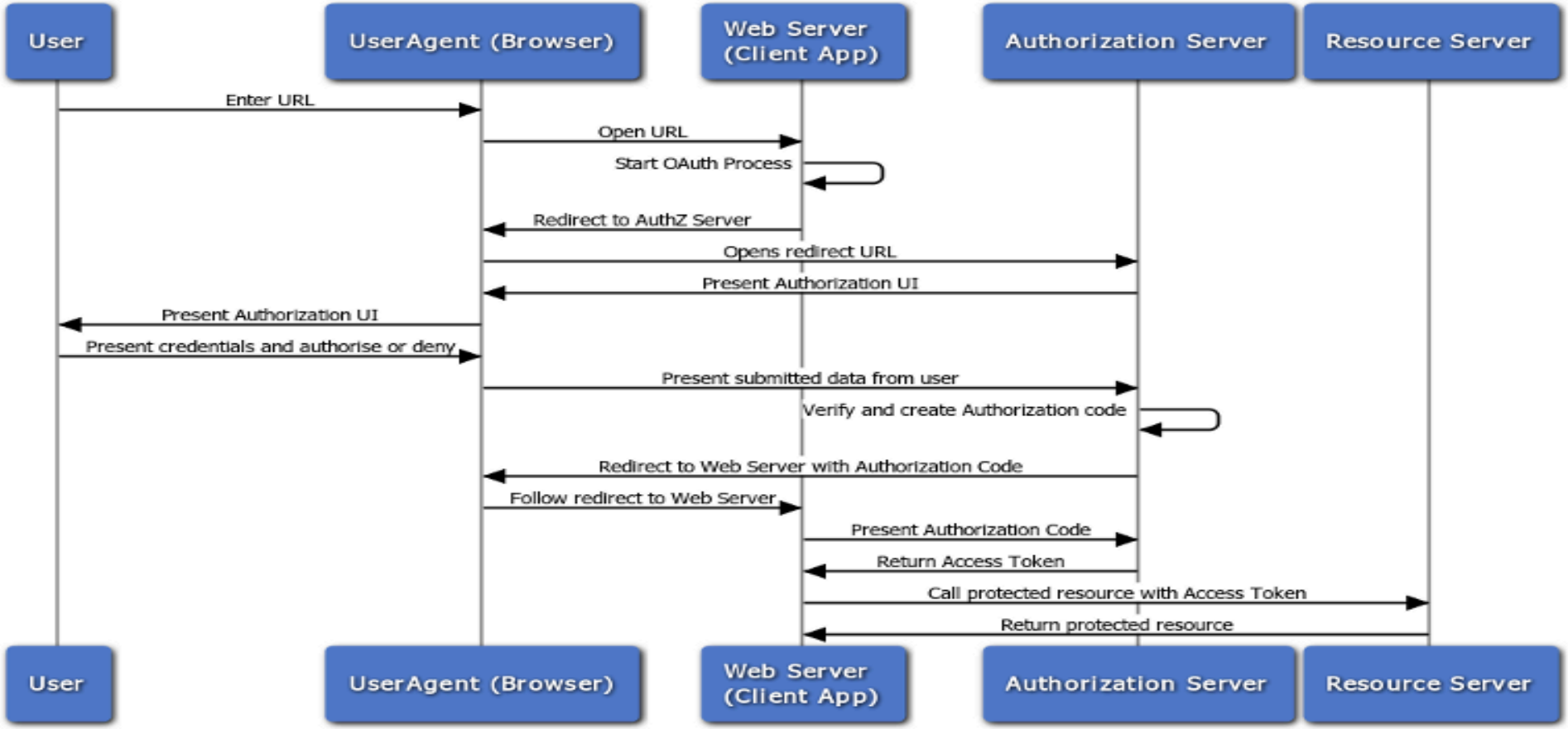
http://wiki.scn.sap.com/wiki/display/Security/OAuth+2.0+Terminology

OAuth 2.0 Flows

- **Authorization Code Grant Flow**
  - Google
  - Facebook
- Resource Owner Password Credential Flow
- Client Credential Flow
- Implicit Grant Flow
  - JavaScript client

Securing REST calls: OAuth 2.0

- Authorization Code Grant Flow



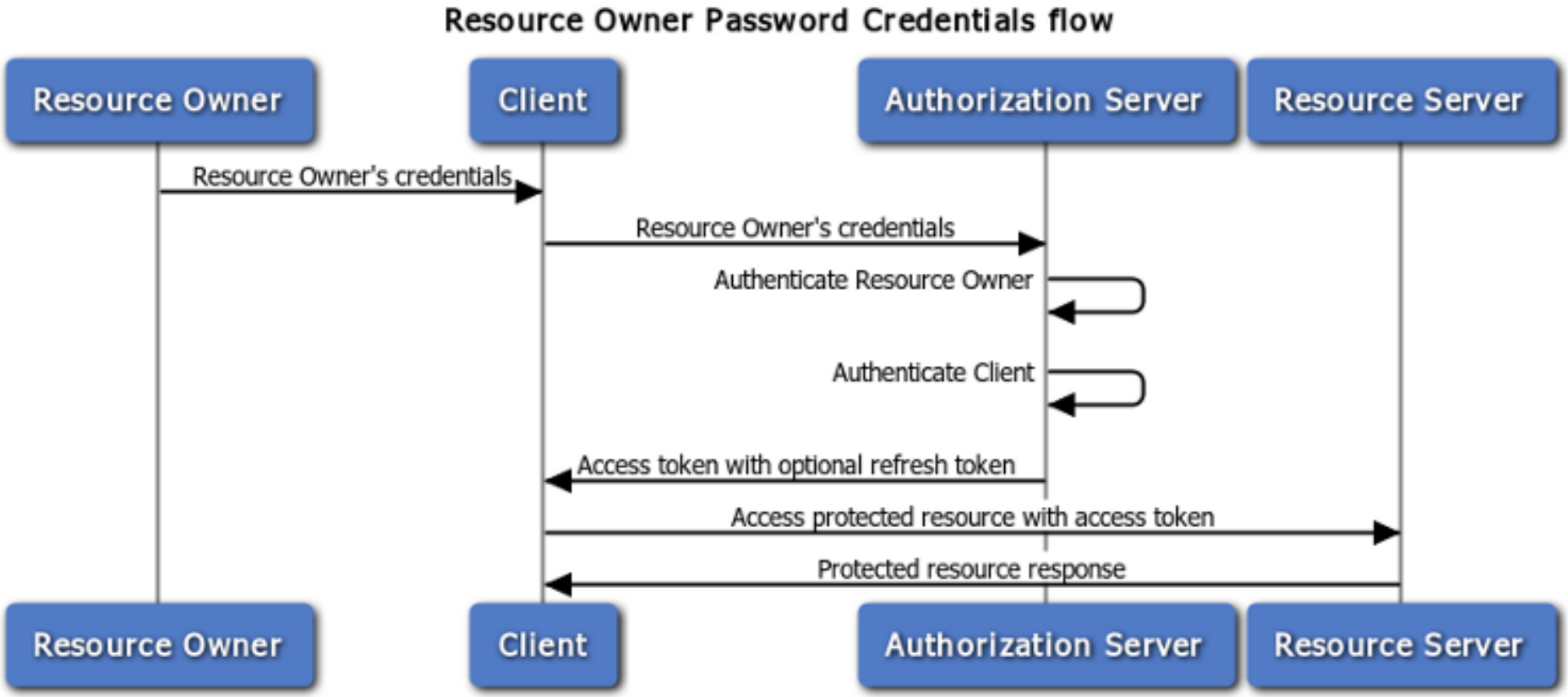
http://docs.oracle.com/cd/E39820_01/doc.11121/gateway_docs/content/images/oauth/oauth_web_server_flow.png

OAuth 2.0 Flows

- Authorization Code Grant Flow
  - Google
  - Facebook
- **Resource Owner Password Credential Flow**
- Client Credential Flow
- Implicit Grant Flow
  - JavaScript client

Securing REST calls: OAuth 2.0

- Resource Owner Password Credential Flow



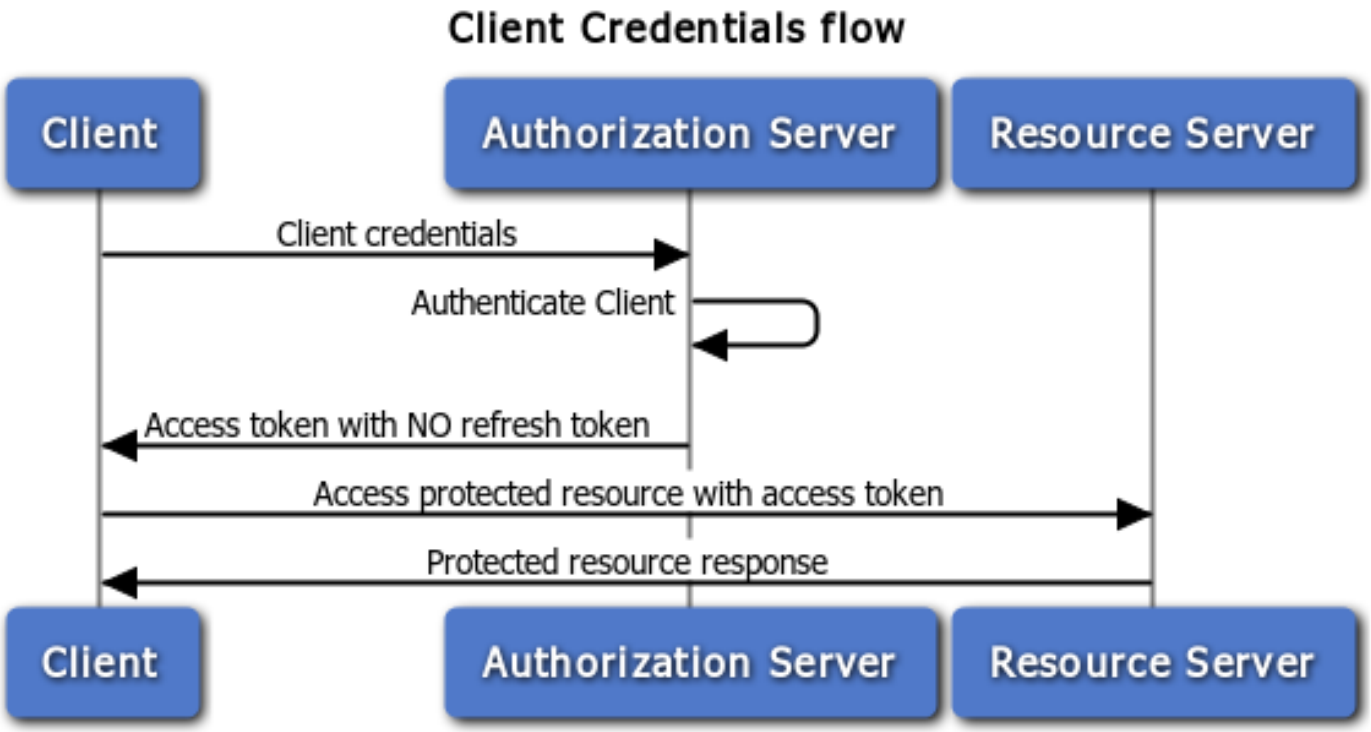
http://docs.oracle.com/cd/E39820_01/doc.11121/gateway_docs/content/images/oauth/oauth_username_password_flow.png

OAuth 2.0 Flows

- Authorization Code Grant Flow
  - Google
  - Facebook
- Resource Owner Password Credential Flow
- **Client Credential Flow**
- Implicit Grant Flow
  - JavaScript client

Securing REST calls: OAuth 2.0

- Client Credential Flow



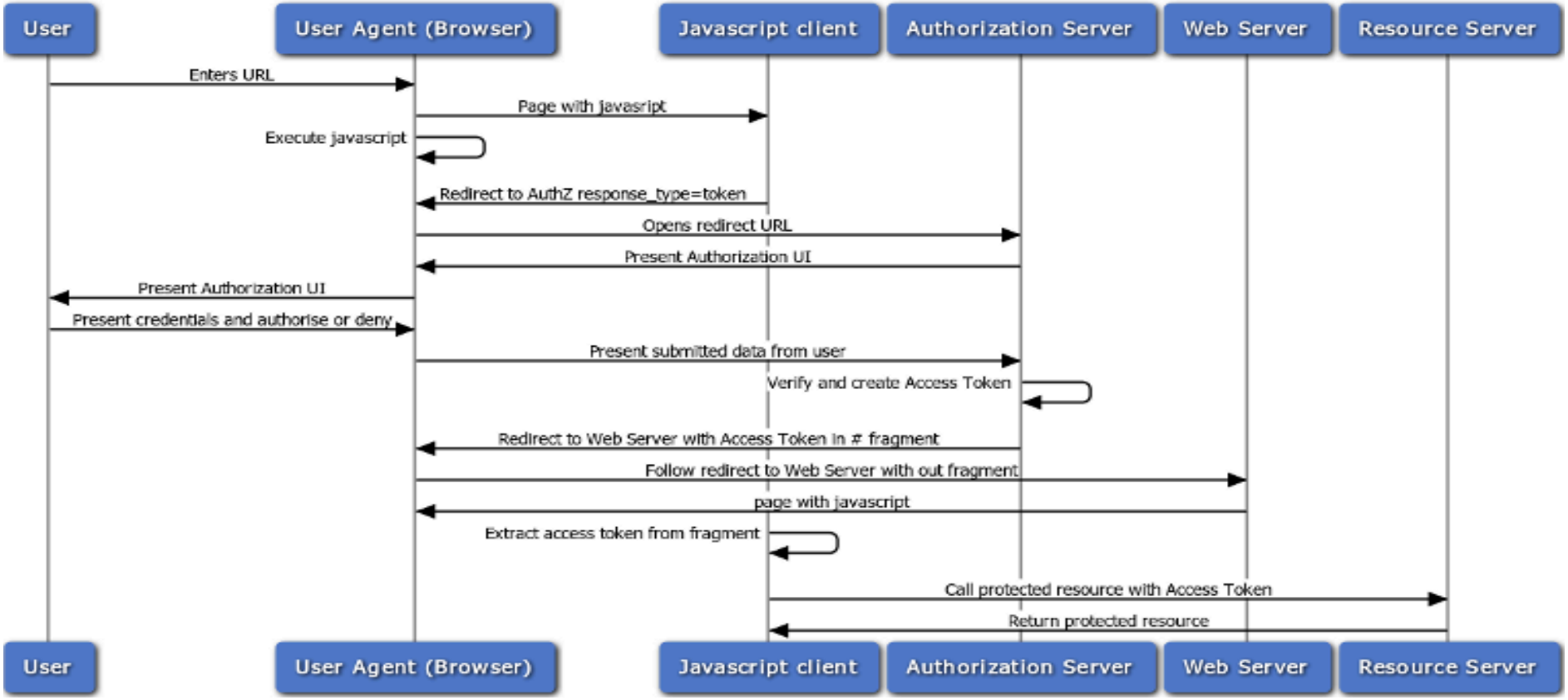
http://docs.oracle.com/cd/E39820_01/doc.11121/gateway_docs/content/images/oauth/oauth_client_credentials_flow.png

OAuth 2.0 Flows

- Authorization Code Grant Flow
  - Google
  - Facebook
- Resource Owner Password Credential Flow
- Client Credential Flow
- **Implicit Grant Flow**
  - JavaScript client

Securing REST calls: OAuth 2.0

- Implicit Grant Flow



http://docs.oracle.com/cd/E39820_01/doc.11121/gateway_docs/content/images/oauth/oauth_user_agent_flow.png

Sample deep-dive

https://github.com/ishaigor/rest-retro-sample

- **Unprotected JavaScript Widget**
  - Unprotected REST Words Service
    - Spring MVC
  - Legacy protected JSP / JavaScript Widget
    - Spring Security
    - AngularJS
- Protected Widget
  - Protected service
    - Spring Security OAuth
  - Protected client
    - Spring Security Oauth
  - HTTP Authorization Header
- Protected gateway
  - Spring Integration
  - Customization

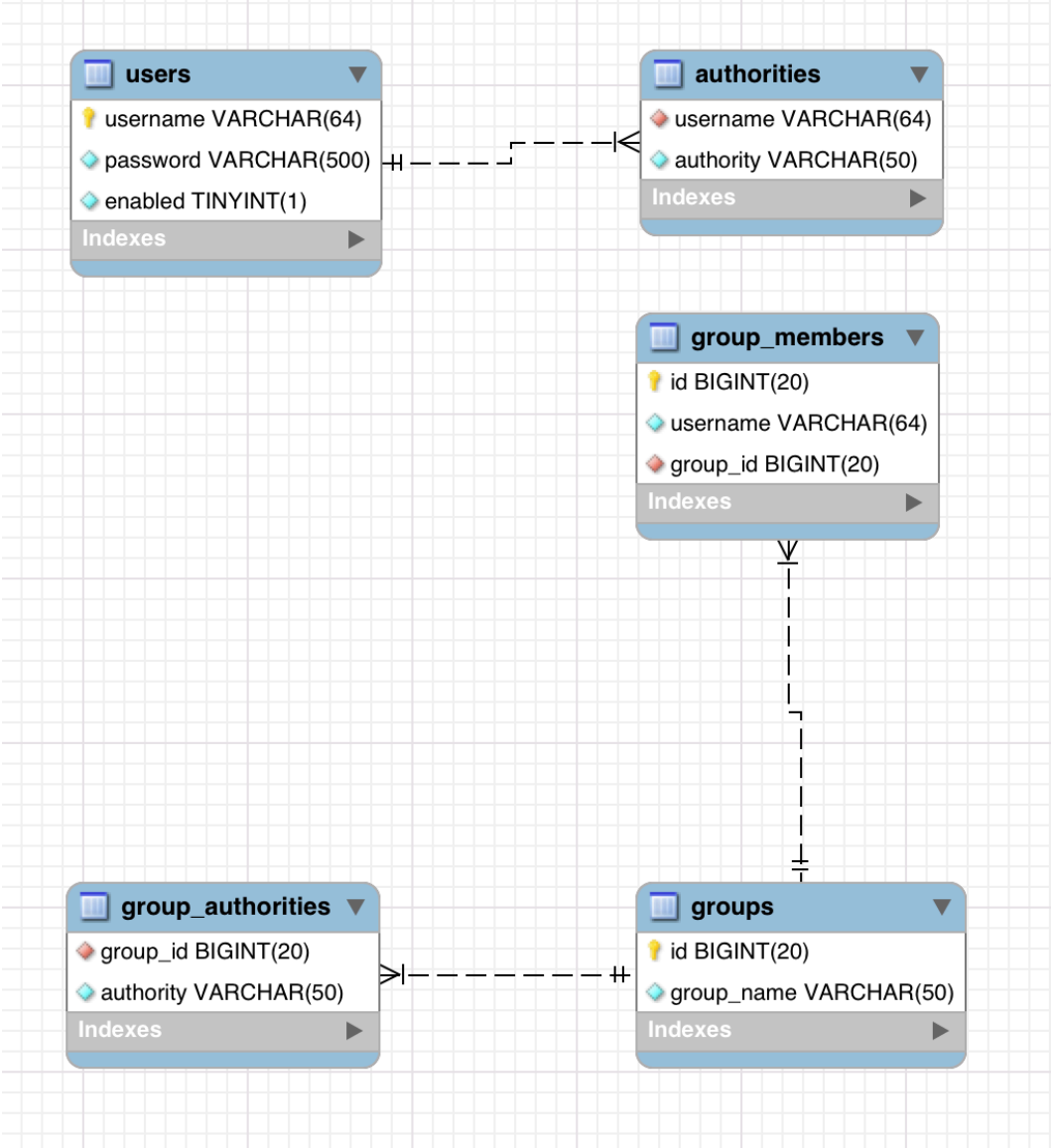
Meet the unprotected REST Service (Spring MVC)

- @RestController

Meet secure legacy client with unprotected Rich UI (Spring Security, Spring MVC, AngularJS)

- ng-infinite-scroll
- AbstractDispatcherServletInitializer
  - springSecurityFilterChain
- WebSecurityConfigurerAdapter
  - @EnableWebSecurity
  - AuthenticationManagerBuilder
  - WebSecurity
  - HttpSecurity
- Persistence
  - Data source
  - Group authorities by user name

Spring Security: User Details



Meet secure legacy client with unprotected Rich UI (Spring Security, Spring MVC, AngularJS) –cont'd

- <%@ taglib prefix="authz" uri="http://www.springframework.org/security/tags"%>
- <authz:authorize ifAllGranted=*"ROLE_USER"*>…</authz:authorize>

Sample deep-dive

https://github.com/ishaigor/rest-retro-sample

- Unprotected JavaScript Widget
  - Unprotected REST Words Service
    - Spring MVC
  - Legacy protected JSP / JavaScript Widget
    - Spring Security
    - AngularJS
- **Protected Widget**
  - Protected service
    - Spring Security OAuth
  - Protected client
    - Spring Security Oauth
  - HTTP Authorization Header
- Protected gateway
  - Spring Integration
  - Customization

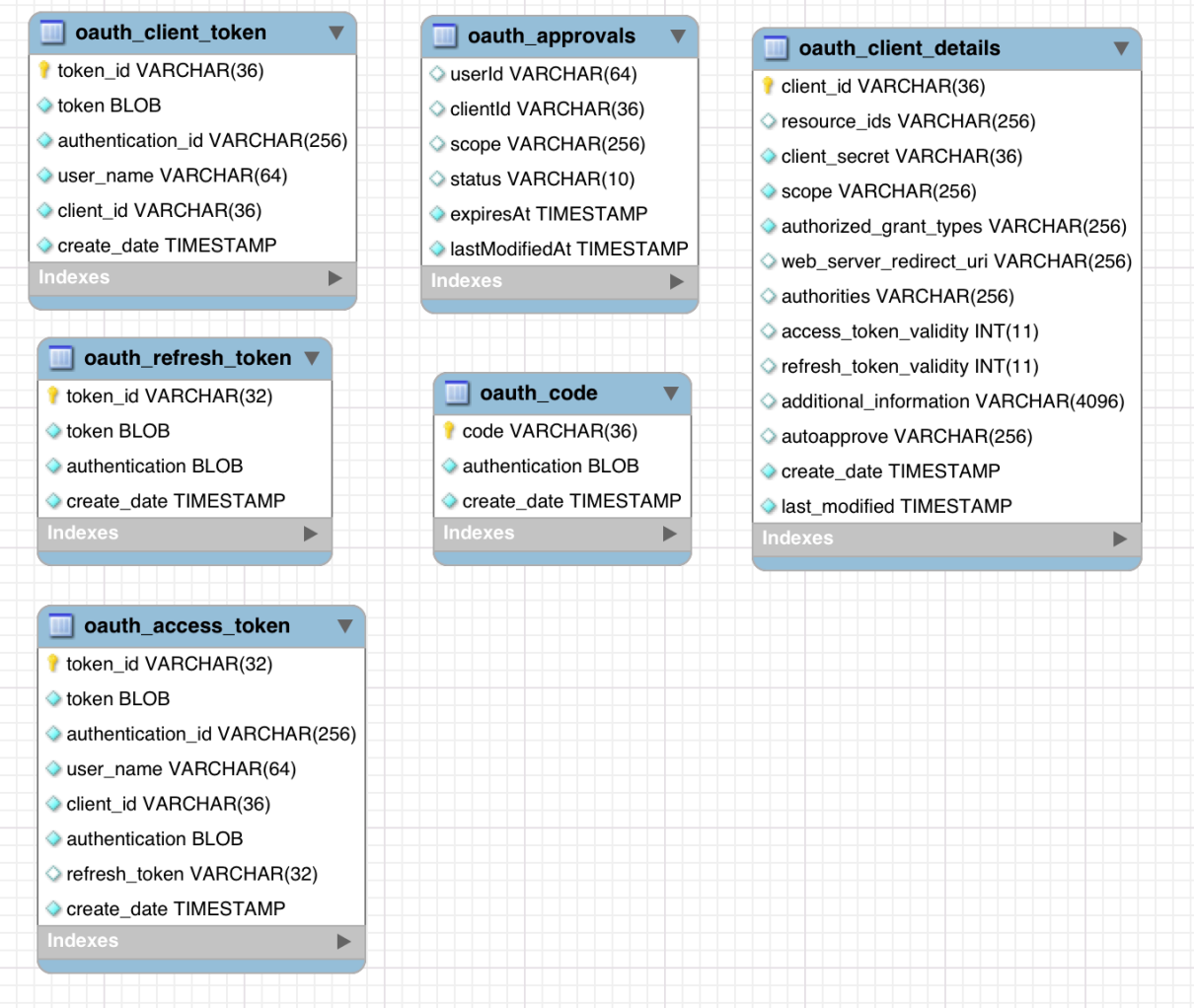
Protected Service (Spring Security, Spring MVC)

- AuthorizationServerConfigurerAdapter
  - ClientDetailsServiceConfigurer
  - @EnableAuthorizationServer
  - AuthorizationServerEndpointsConfigurer
  - AuthorizationServerSecurityConfigurer
- GlobalMethodSecurityConfiguration
  - @EnableGlobalMethodSecurity
  - OAuth2MethodSecurityExpressionHandler

Protected Service (Spring Security, Spring MVC) – cont'd

- ResourceServerConfigurerAdapter
  - ResourceServerSecurityConfigurer
  - HttpSecurity
    - .csrf().requireCsrfProtectionMatcher(new AntPathRequestMatcher("/oauth/authorize")).disable()
- Persistence
  - TokenStore
  - ClientTokenServices
  - AuthorizationCodeServices
  - ApprovalStore
- ApprovalStoreUserApprovalHandler

Protected Service (Spring Security, Spring MVC) – cont'd



Protected Service (Spring Security, Spring MVC): testing

- @OAuth2ContextConfiguration
- BaseOAuth2ProtectedResourceDetails
- IntegrationTest
- IntegrationTestHelper

Protected client, protected Rich UI (Spring Security, Spring MVC, Spring Security OAuth 2.0)

- AuthenticationManager
  - eraseCredentials
- ApplicationListener<AbstractAuthenticationEvent>
  - ResourceOwnerPasswordAccessTokenProvider
- CustomAuthenticationDetailsSource
  - CustomAuthenticationDetails
  - WebAuthenticationDetailsSource

Protected service with Spring

- Limitations:
  - Added security overhead
  - No unprotected internal access

Sample deep-dive

https://github.com/ishaigor/rest-retro-sample

- Unprotected JavaScript Widget
  - Unprotected REST Words Service
    - Spring MVC
  - Legacy protected JSP / JavaScript Widget
    - Spring Security
    - AngularJS
- Protected Widget
  - Protected service
    - Spring Security OAuth
  - Protected client
    - Spring Security Oauth
  - HTTP Authorization Header
- **Protected gateway**
  - Spring Integration
  - Customization

Security Gateway Pass Through with Spring Integration

- int-http:inbound-gateway
- int-http:outbound-gateway
- int:channel
- int:annotation-config
- int-jmx:mbean-export

Security Gateway Pass Through with Spring Integration: customization

- OutboundHeaderMapper
- RangeEnforcer
- CustomOAuth2WebSecurityExpressionHandler
- CustomSecurityExpressionMethods
- ClientHttpRequestFactory

Resources

- http://oauth.net/2/
- http://projects.spring.io/spring-security/
- http://projects.spring.io/spring-security-oauth/
- https://github.com/ishaigor/rest-retro-sample
- http://binarymuse.github.io/ngInfiniteScroll/

Security Roadmap

Address REST Services Exposure

Merge user identities in a single directory

Centralize identity management

Build secure APIs with our customers

Other enhancements

Next steps on the road map: IdP with SSO

- IdP – Identity Provider
  - centralized user directory
  - Identity management services
  - Self-services
  - Light linked identities from application side
- SSO – Single Sign On
  - Central Authentication Service
    - A library of clients for **Java, .NET**, PHP, Perl, Apache, uPortal, and others
    - Integrates with uPortal, Sakai, BlueSocket, TikiWiki, **Mule**, Liferay, Moodle and others