Securing OAuth2-Enabled, Multi-Tenant Applications with Spring Security

Rob Winch
SpringSource, VMware

About Me

- Spring Security Lead at SpringSource, VMware
- Past
  - Cerner: Secure Health Care Applications
  - Argonne Labs: Grid Computing
  - Loyola University Chicago: Proteomics Research
  - Self-Employed: Contractor
- From Kansas City and enjoy playing Softball

http://en.wikipedia.org/wiki/File:Kansas_City_MO_Skyline_14July2008v.jpg

Agenda

- Spring Security
- Multi Tenancy
- OAuth 2
- Links
- Q&A

Tell me about Spring Security

- Formerly known as Acegi Security
- Authentication
  - Database, LDAP, CAS, OpenID, Pre-Authentication, custom, etc
- Authorization
  - Interface-based proxies, Class-based proxies, AspectJ
- Extensions
  - SAML, Kerberos, OAuth
- Simple yet powerful

Basic Spring Security Setup

- Add Spring Security Maven Dependencies
- Update web.xml
- Create Spring Security Configuration

Abstractions

- If you are not implementing a Spring Security interface, it is good practice to abstract usage
- SpringSecurityUserContext accesses our Employee from the SecurityContextHolder
- Customizing the AuthenticationProvider will allow placing your own domain representation in SecurityContext

Defense in Depth

- Securing URLs is not enough. Always secure your service tier too
- Spring Security uses annotations like @PreAuthorize and the <global-method-security> element to protect your services
- Choice of interface-based proxies, class-based proxies, or AspectJ integration

Multi Tenancy – Data

- Multiple Strategies
  - Tenant discriminator columns
    - Simple to setup, but not as secure and not as flexible (scaling per client not possible)
  - Schema or Database per tenant
    - Isolation of data and flexibility but more complex to setup
- Multiple implementations
  - ORM's (i.e. Hibernate, EclipseLink, etc)
  - Spring (AbstractRoutingDataSource)

Multi Tenancy – Resource Mapping

- Domain / Subdomain
  - i.e. https://tenantname.example.com/resource/ https://tenantname.com/resource/
  - More complex setup
  - More Secure due to same origin policy
- URL
  - i.e. https://example.com/tenantname/resource/
  - Simple to setup
  - Less secure due to no help from same origin policy

Mutli Tenancy Abstractions

- TenantRoutingDataSource
- TenantFilter
  - Obtains and allows access to the current tenant
  - Overrides the HttpServletRequest so that the new context root appears to be /context/tenantname/ which means generating links is transparent to us
- TenantContext
  - Application uses to obtain the current tenant
  - TenantFilter implements this interface
- TenantAware
  - For resources/domain objects that are aware of which tenant owns them

Making Resources TenantAware

- We would like to do this without modifying our application code (separation of concerns)
- Create a TenantAwareAspect with AspectJ
  - Integrates nicely in Eclipse using AJDT
  - m2e provides support for integrating with aspectj-maven-plugin integrates with m2e

Tenant Security

- Do not want tenants to access data from another tenant
- Spring supports custom expressions
- TenantWebExpressionHandler
  - Nice abstraction to as how to determine if current user has access

OAuth

- “Valet Key”
- OAuth 2.0 vs OAuth 1.0
  - OAuth 2.0 more simple
  - OAuth 2.0 designed for scalability
  - Not compatible
  - Requires HTTPS
  - OAuth 2.0 is not finished

OAuth 2.0 – Basic Flow

- Client asks user for authorization
- Client obtains authorization grant
- Client requests authorization token by authenticating and presenting the authorization grant
- Client requests protected resource using access token

Links

- https://github.com/rwinch/finance/
- http://springsource.org/spring-security

Questions?