Security Tutorial

Shawn McKinney

November 11, 2015

LDAPCon, Edinburgh

Session Objectives

Learn:

- ✓ Apache Fortress infrastructure setup
- ✓ Web security techniques
- ✓ TLS over all endpoints

Introductions

Shawn McKinney

- 

Systems Architect
-  PMC Apache Directory Project
- OpenLDAP™ PROJECT Engineering Team

Agenda

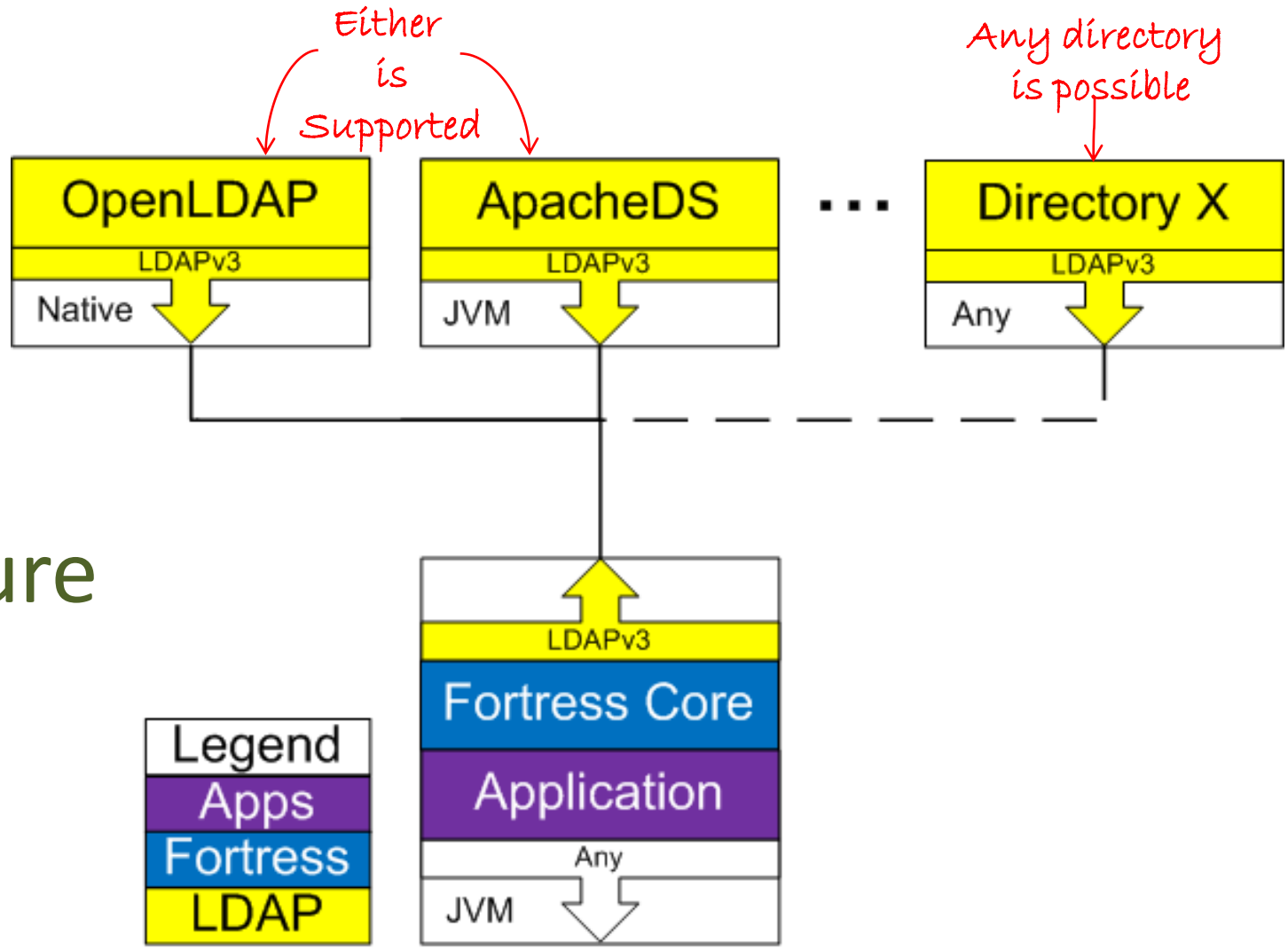
- Ten Minute Guide
- End-To-End Security
- Role Engineering
- Wicket Sample
- SAML 2.0 & RBAC

IMAGE FROM: HTTP://EVENTS.LINUXFOUNDATION.ORG/EVENTS/APACHECON-NORTH-AMERICA

Tutorial Machines

- Cloud environment machines for this security tutorial from CenturyLink Cloud.
- Find out more at www.ctl.io

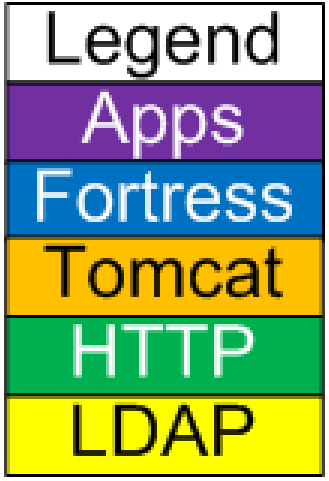
Core System Architecture

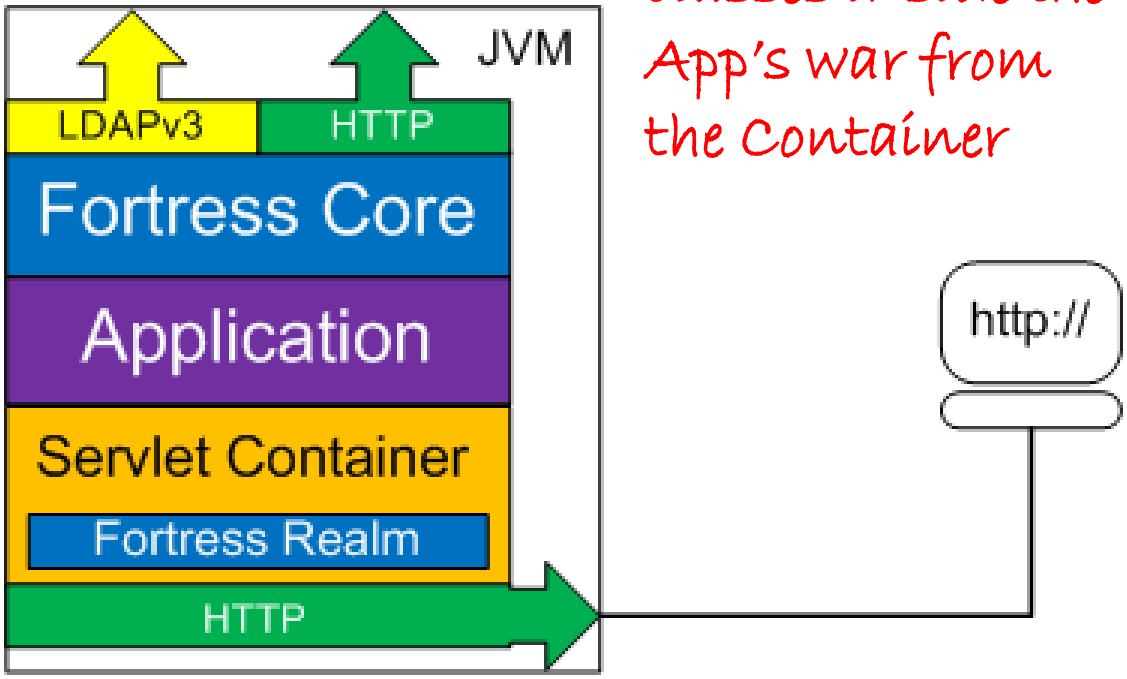


symas

Realm System Architecture

1. Shares the RBAC session (activated roles) created inside the Realm with the App.

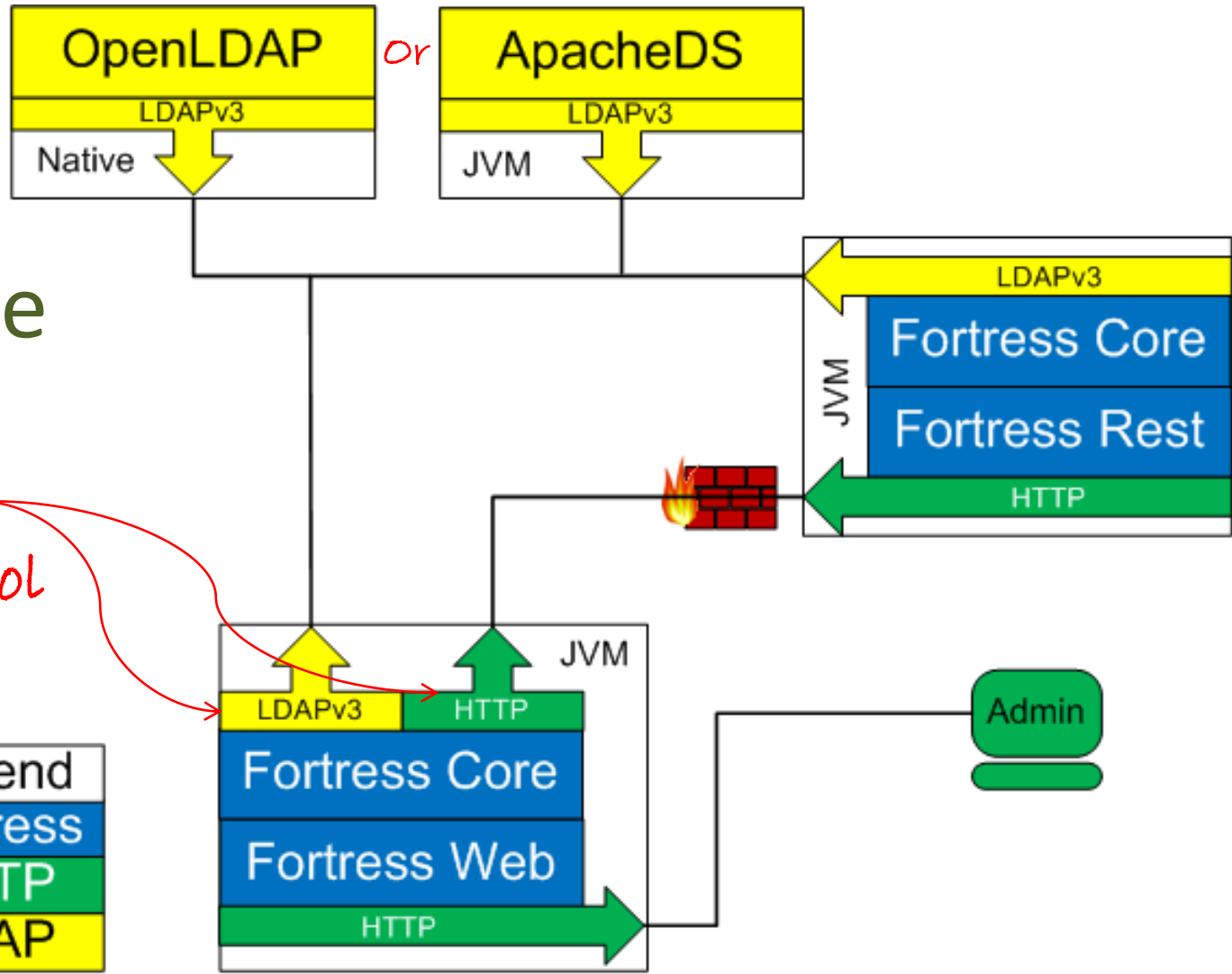




2. Isolates Fortress classes inside the App's war from the Container

Web System Architecture

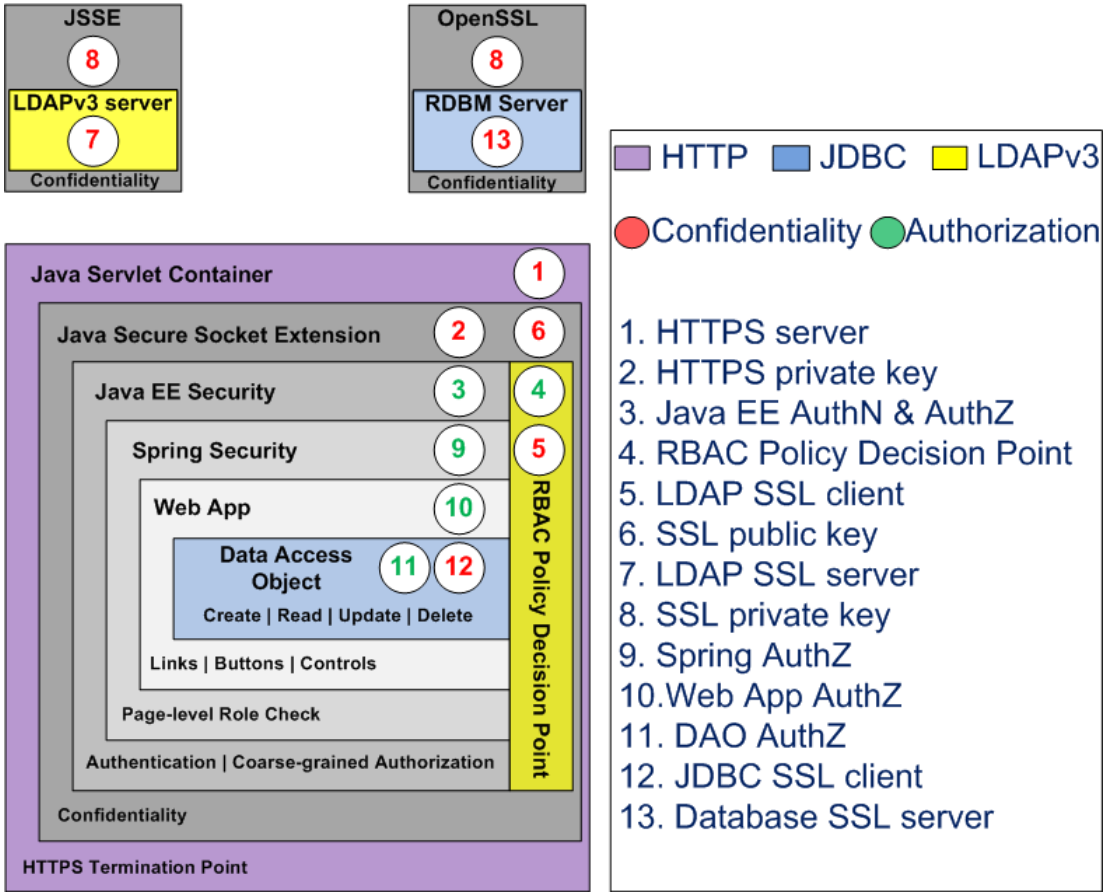
Option to use either HTTP or LDAPV3 protocol



symas

Tutorial #1

Apache Fortress End-to-End Security Tutorial

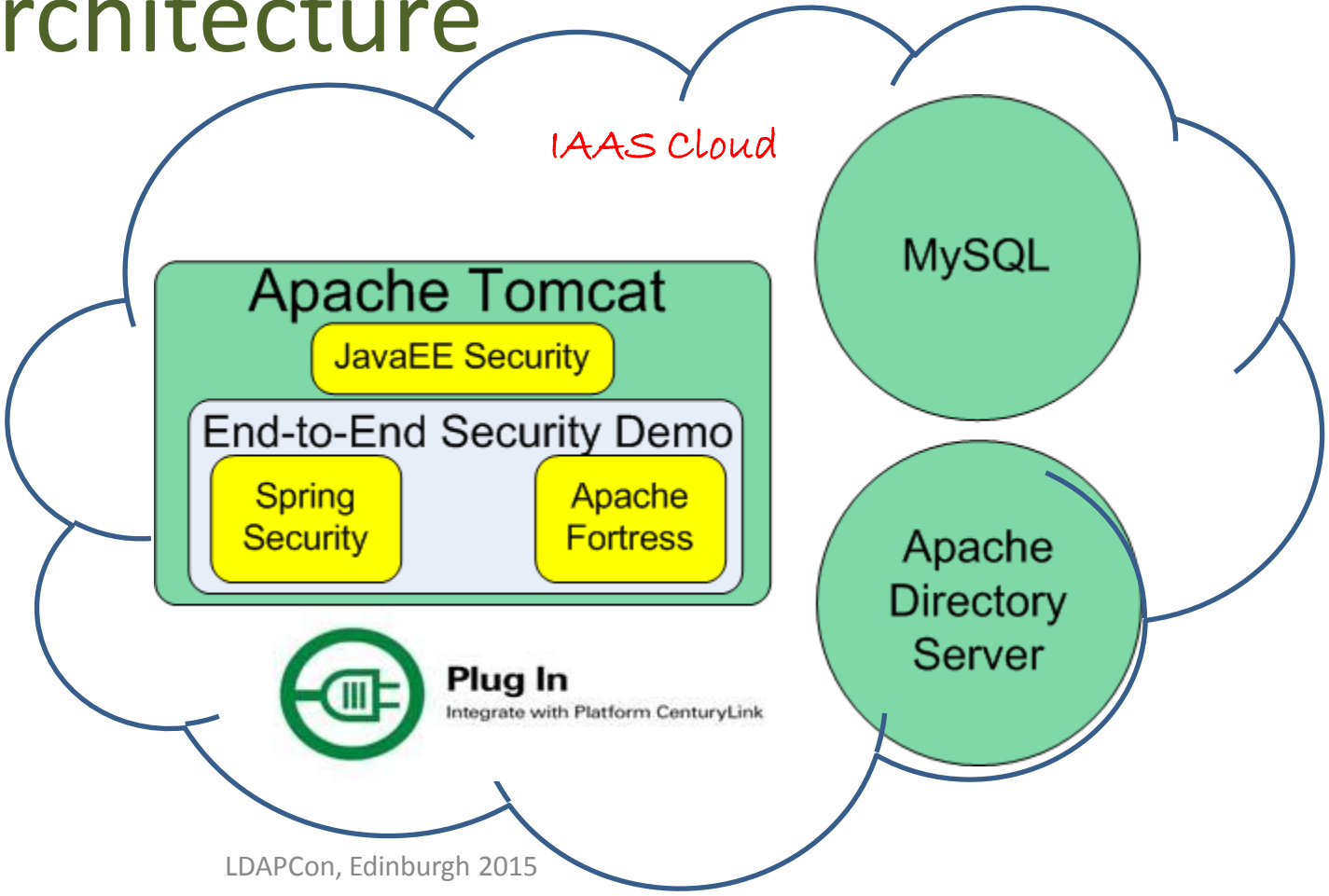


http://iamfortress.net/2015/02/16/apache-fortress-end-to-end-security-tutorial/

symas

System Architecture

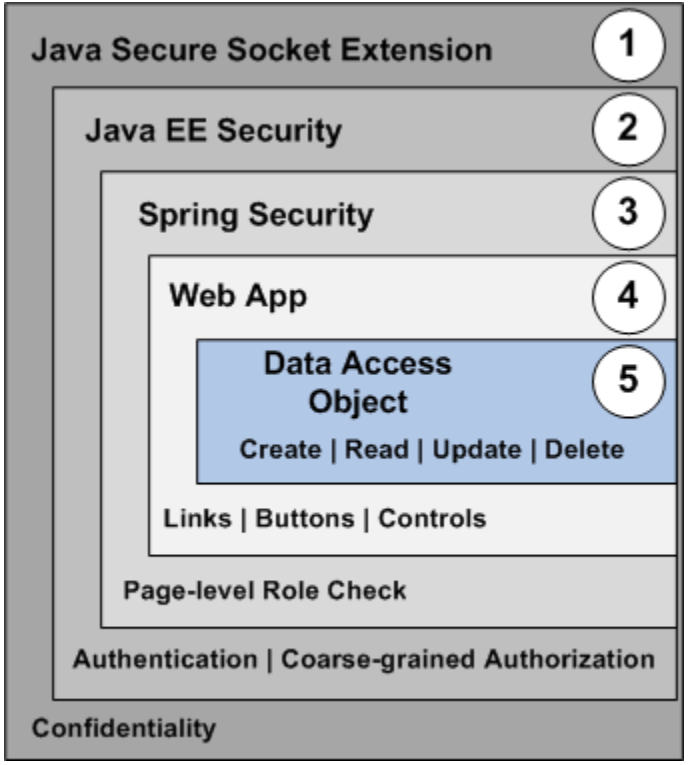
Tutorial # 1



symas

The Five Security Layers of Java Web Apps

1. Java Secure Socket Extension (JSSE)
2. Java EE Security
3. Spring Security
4. Web App Framework
5. Database Functions



Security Layers with Fortress Demo

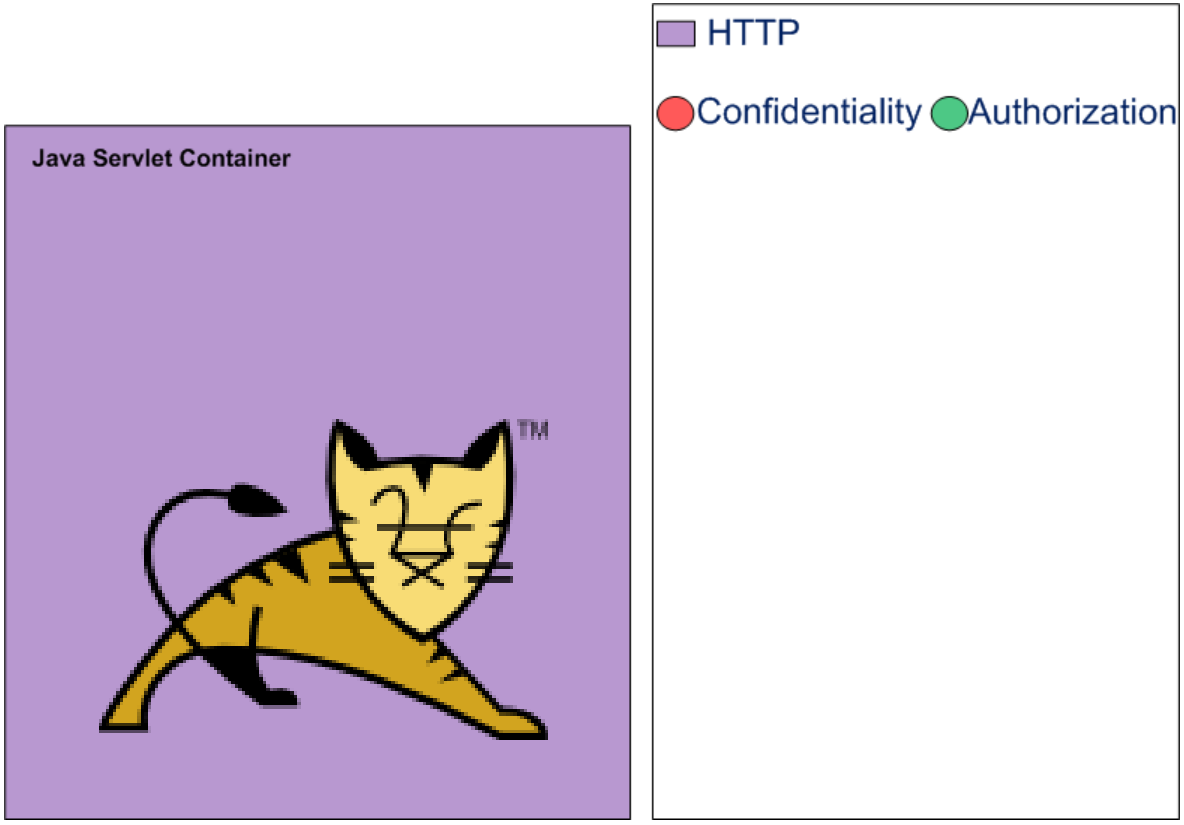
1. JSSE ← Confidentiality and Integrity
2. Java EE Security ← authN and coarse-grained authZ
3. Spring Security ← medium-grained authZ
4. Web App Framework ← fine-grained authZ
5. Database Functions ← very fine-grained authZ

Two Areas of Access Control

1. Java EE and Spring Role Declarative checks

2. RBAC Permission Programmatic checks

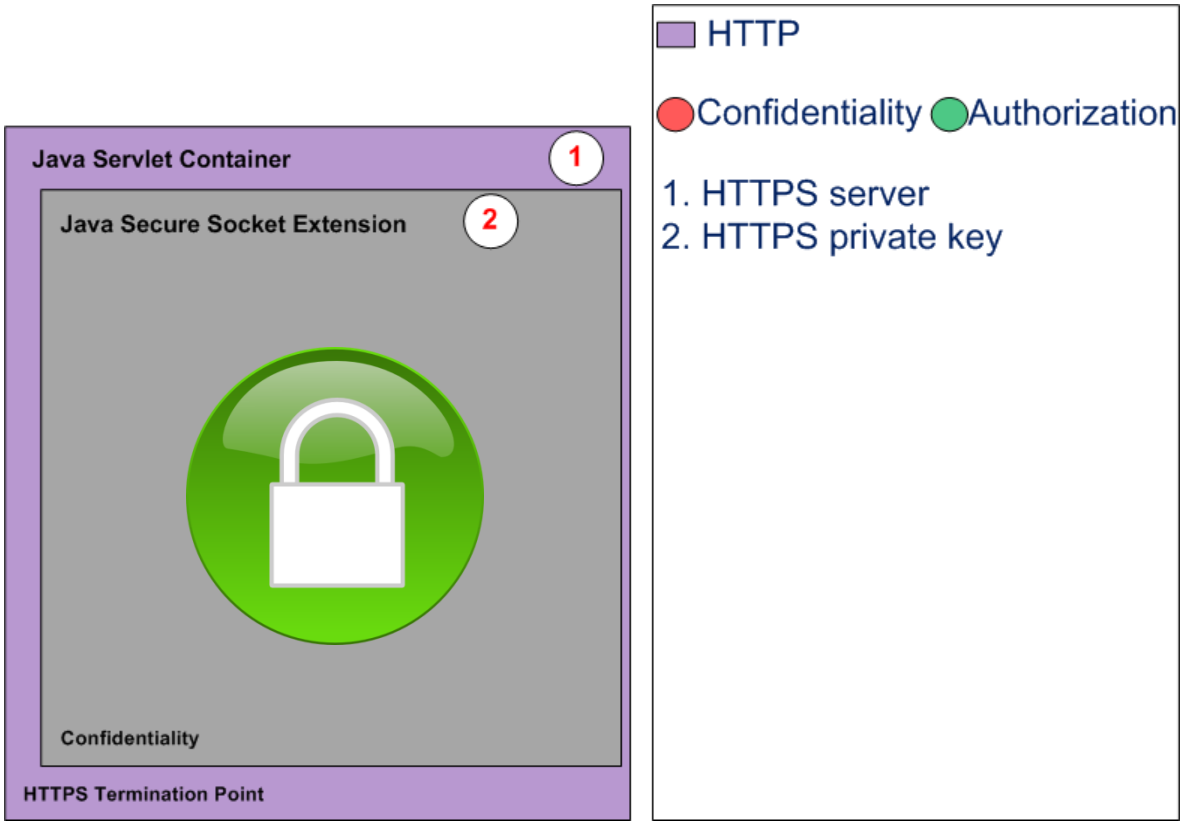
Start with Tomcat Servlet Container



1 & 2. Enable HTTPS

1. Update the Server.xml

2. Add private key

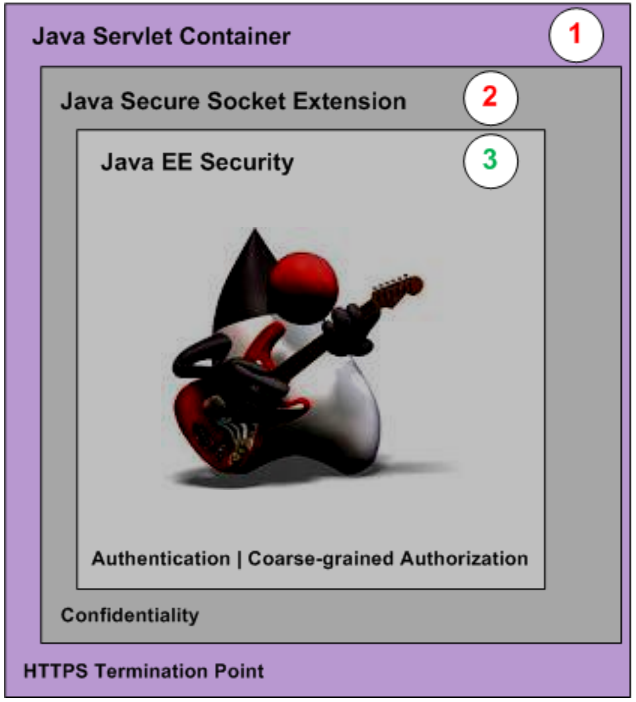


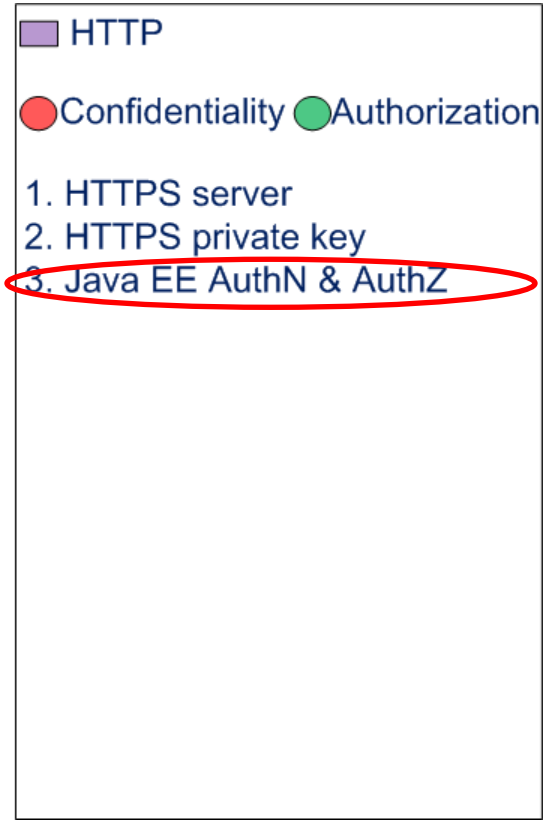
3. Enable Java EE Security

a. Drop the proxy jar

b. Update web.xml

c. Add context.xml





Enable Java EE Security Realm

Drop the Fortress Realm Proxy Jar in Tomcat's lib folder

```
[root@IL1SCOLSP102 lib]# pwd
/usr/local/tomcat7/lib
[root@IL1SCOLSP102 lib]# ls -l fortress*
-rw-r--r-- 1 root root 15119 Aug 29 12:37 fortress-realm-proxy-1.0-RC41-SNAPSHOT.jar
[root@IL1SCOLSP102 lib]#
```

Fortress Realm Proxy loads implementation jars from the app via a URLClassloader 'trick'.

```
[root@IL1SCOLSP102 lib]# pwd
/usr/local/tomcat7/webapps/apache-fortress-demo/WEB-INF/lib
[root@IL1SCOLSP102 lib]# ls -l fortress*
-rw-r--r-- 1 root root 502112 Aug 30 06:55 fortress-core-1.0-RC41-SNAPSHOT.jar
-rw-r--r-- 1 root root  22005 Aug 29 12:20 fortress-realm-impl-1.0-RC41-SNAPSHOT.jar
-rw-r--r-- 1 root root 789927 Aug 29 12:40 fortress-web-1.0-RC41-SNAPSHOT-classes.jar
[root@IL1SCOLSP102 lib]#
```

Enable Java EE Security Realm

Add to App's Web.xml

```
<security-constraint>
  <display-name>My Project Security Constraint</display-name>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <url-pattern>/wicket/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>DEMO2_USER</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>MySecurityRealm</realm-name>
  <form-login-config>
    <form-login-page>/login/login.html</form-login-page>
```

1. Java EE container protects this URL Automatically.

2. All users must have this role to gain entry.

3. Route un-authN requests to my form.

https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/webapp/WEB-INF/web.xml

Enable Java EE Security Realm

Add context.xml to META-INF folder:

```
<Context reloadable="true">
 <Realm className=
  "org.apache.directory.fortress.realm.tomcat.Tc7AccessMgrProxy"

   defaultRoles="ROLE_DEMO2_SUPER_USER,DEMO2_ALL_PAGES,
                 ROLE_PAGE1, ROLE_PAGE2, ROLE_PAGE3"
   containerType="TomcatContext"
   realmClasspath=""
 />
</Context>
```

Fortress Tomcat Realm engaged

Activate these roles into RBAC session.

impl jars will be found in app's war

https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/resources/META-INF/context.xml

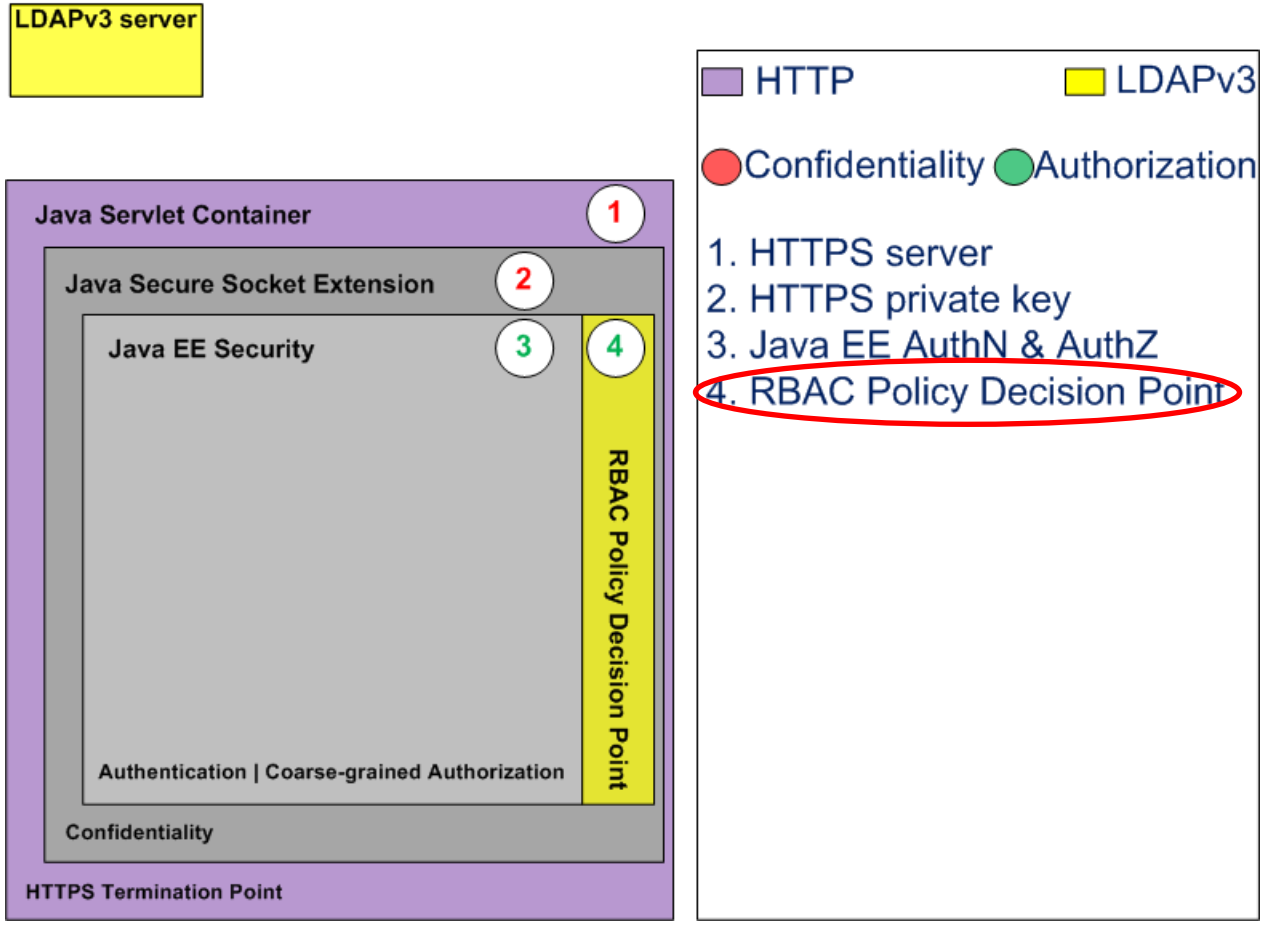
4. Setup RBAC PDP

Policy Decision Point

a. Install

b. Configure

c. Use



Install Fortress RBAC Policy Decision Point

Download and install Apache Directory Fortress

The Fortress ten minute guide provides instructions:

1. Apache Directory Server
2. Apache Directory Studio
3. Apache Fortress Core
4. Apache Fortress Realm
5. Apache Fortress Web
6. Apache Fortress Rest

Required components to apache fortress demo.

https://directory.apache.org/fortress/gen-docs/latest/apidocs/org/apache/directory/fortress/core/doc-files/ten-minute-guide.html

Use ANSI RBAC INCITS 359 Specification

RBAC0:

- Users, Roles, Perms, Sessions

RBAC1:

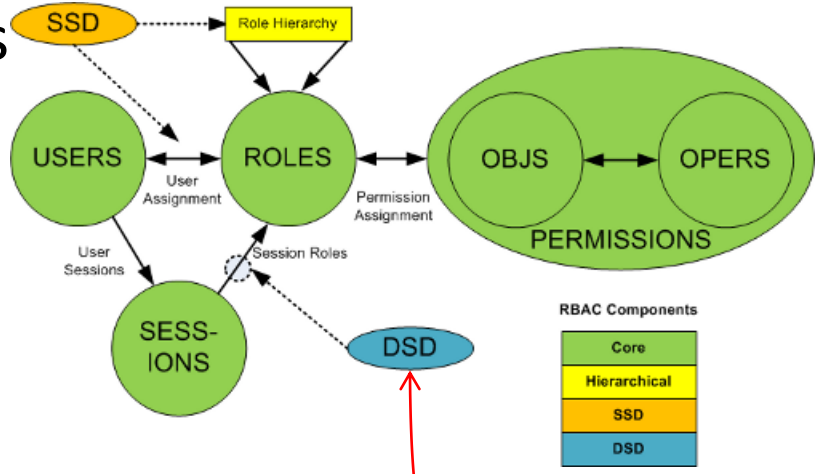
- Hierarchical Roles

RBAC2:

- Static Separation of Duties

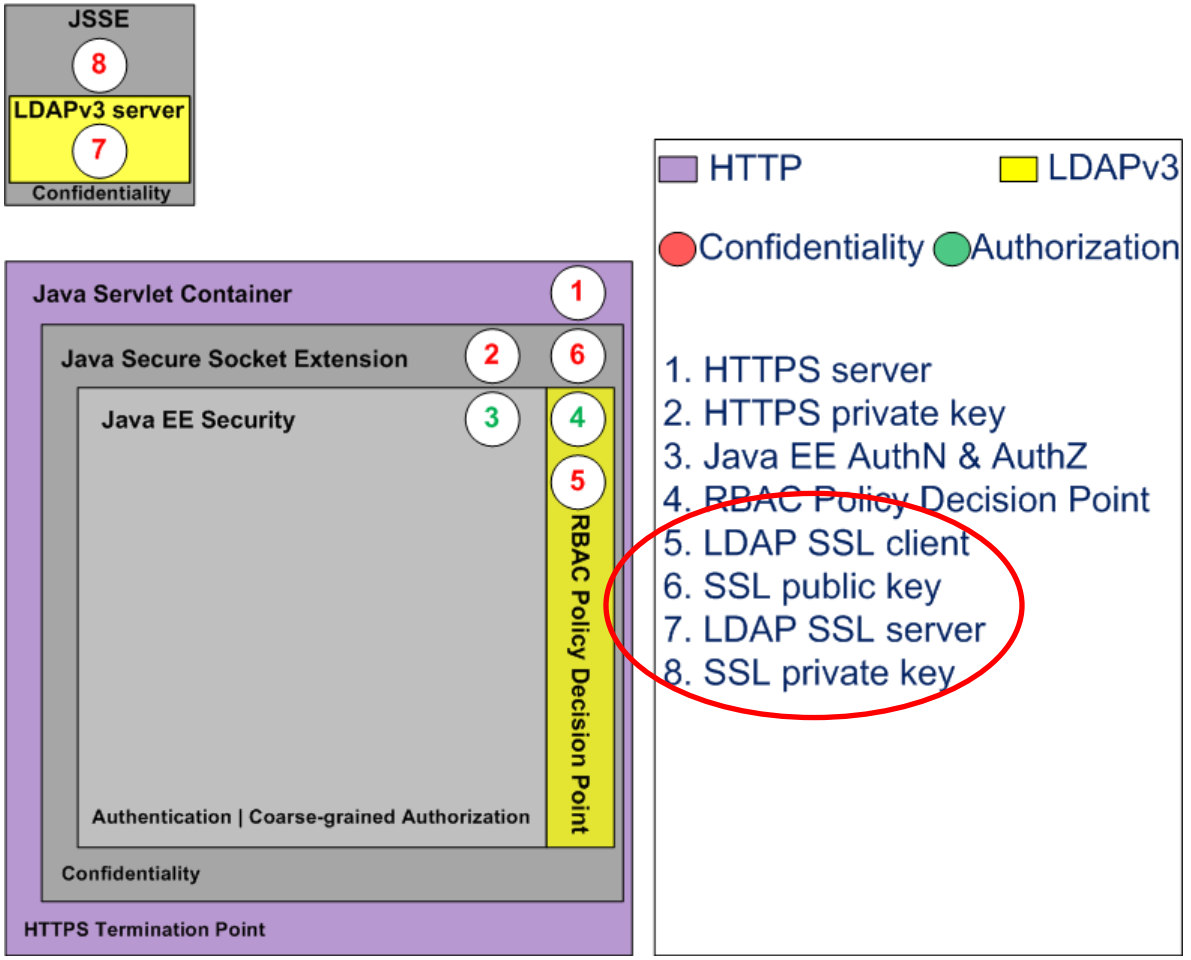
RBAC3:

- Dynamic Separation of Duties



Today we demo this

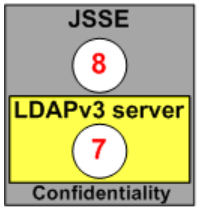
5 – 8 Enable LDAP SSL

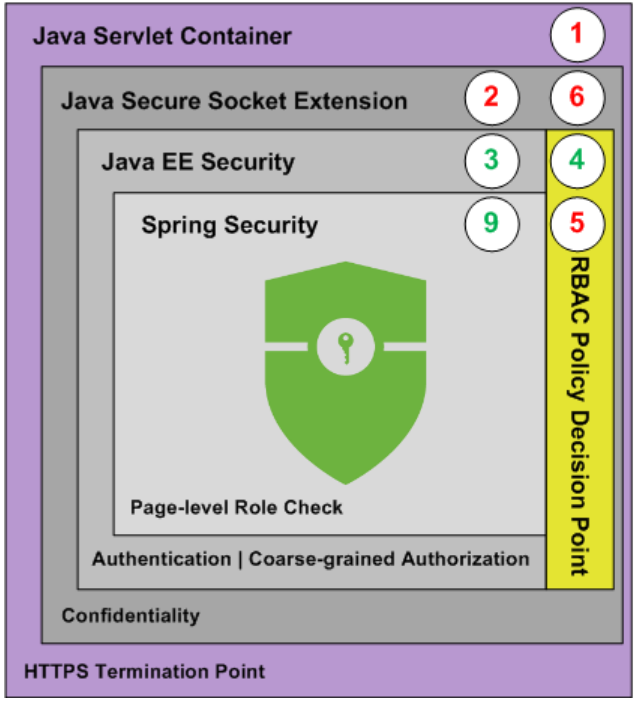


9. Enable Spring Security

a. Enable AuthZ

b. Role mapping





Enable Spring Security

Add dependencies to pom:

```
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-core </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-config </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId> spring-security-web </artifactId>
   <version>4.0.2.RELEASE</version>
</dependency>
```

Enable Spring Security Interceptor

```
<bean id="fsi“
="org.springframework.security.web.access.intercept.FilterSecurityInterceptor">

  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager" ref="httpRequestAccessDecisionManager"/>
  <property name="securityMetadataSource">
   <sec:filter-security-metadata-source use-expressions="false">
        <sec:intercept-url pattern=
          "…/com.mycompany.page1"
          access="ROLE_PAGE1"
        />
   </sec:filter-security-metadata-source>
  </property>
</bean>
```

page-level authorization (declarative)

By default name must contain ROLE_

Role Mapping

Identity Propagation Java EE -> Spring Security

Spring Security uses PreAuthenticatedAuthentication filter to get java EE role mappings.

From the applicationContext.xml:

```
<bean id="preAuthenticatedAuthenticationProvider”
class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider">
  <property name="preAuthenticatedUserDetailsService"  ref="preAuthenticatedUserDetailsService"/>
</bean>
…
```

Role Mapping

Share Roles Between Java EE and Spring

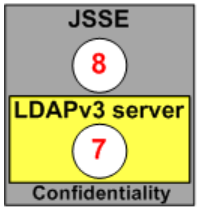
Complete list of eligible roles found in app's web.xml:

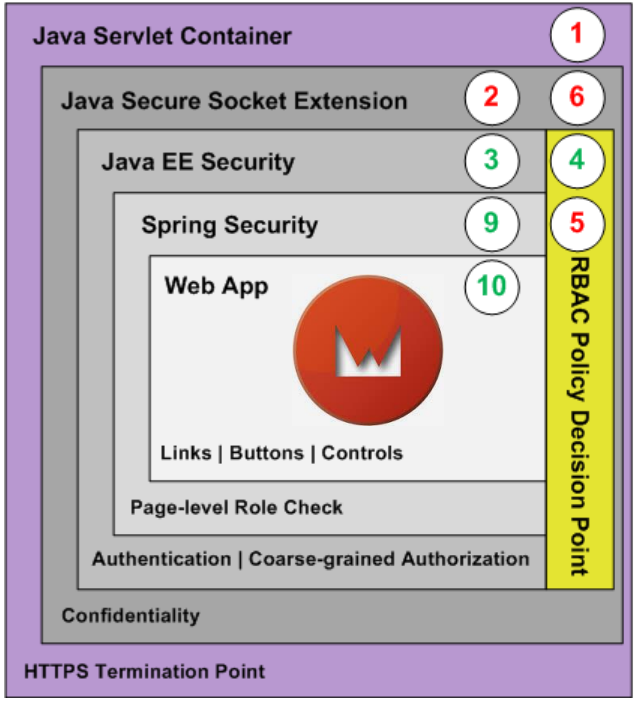
```
<!-- Declared in order to be used by Spring Security -->
<security-role>
  <role-name>ROLE_DEMO2_SUPER_USER</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE1</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE2</role-name>
</security-role>
<security-role>
  <role-name>ROLE_PAGE3</role-name>
</security-role>
```

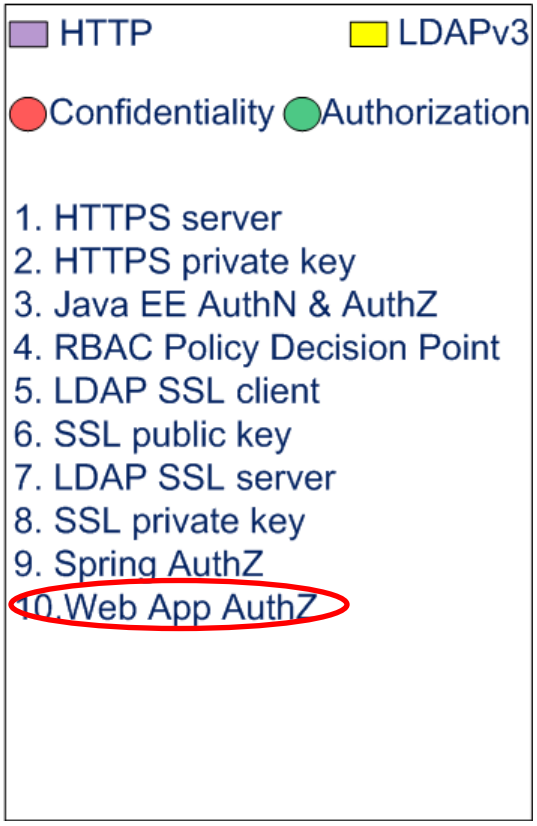
10. Web App Authorization

Add fine-grained checks:

a. Page links
b. Buttons
c. Other controls







Add Web Framework Security

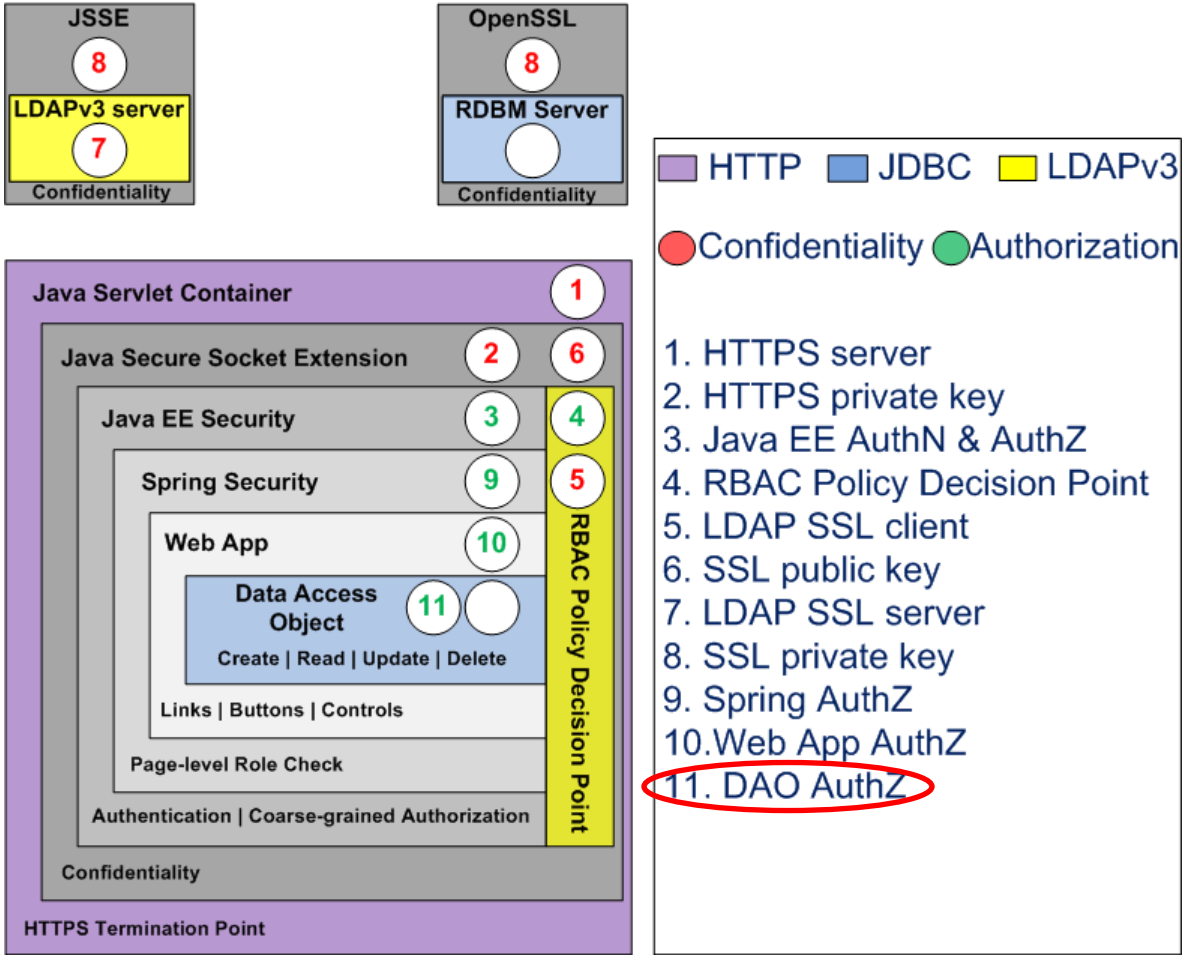
```
add(
   new SecureIndicatingAjaxButton( "Page1", "Add" )
   @Override
   protected void onSubmit( ... )
   {
      if( checkAccess( customerNumber ) )
      {
        // do something here:
      }
      else
      {
         target.appendJavaScript( ";alert('Unauthorized');" );
      }
   }
   });
```

fine-grained authorization (programmatic)

11. DAO Authorization

Add fine-grained Checks to:

a. Create
b. Read
c. Update
d. Delete



Add Security Aware DAO components

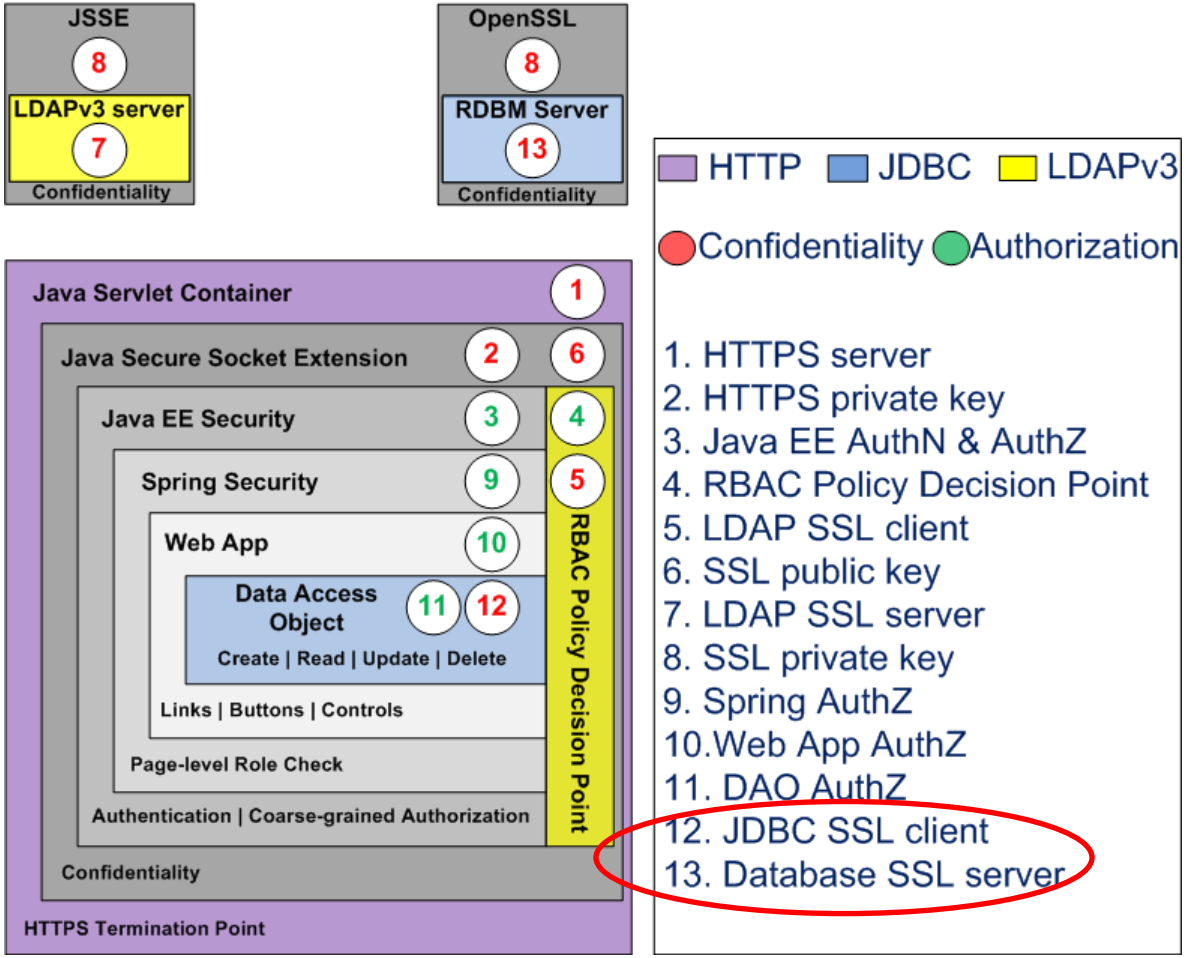
```
public Page1EO updatePage1( Page1EO entity )
{
   ...
 if(checkAccess("Page1","Update",entity.getCust()))
{
   // Call DAO.update method...
}
else
 throw new RuntimeException("Unauthorized");
...
return entity;
}
```

fine-grained authorization (programmatic)

12, 13. Enable DB SSL

12. Client
 a. public key
 b. config

13. Server
 a. private key
 b. config



Apache Fortress Demo

- Three Pages and Three Customers
- One role for every page to customer combo
- Users may be assigned to one or more roles
- One and only one role may be activated

Pages	Customer 123	Customer 456	Customer 789
Page One	PAGE1_123	PAGE1_456	PAGE1_789
Page Two	PAGE2_123	PAGE2_456	PAGE2_789
Page Three	PAGE3_123	PAGE3_456	PAGE3_789

Demo Usage Policy

- Both super and power users may access everything.
- But power users are limited to one role activation at a time.
- Super users are not restricted.

Super & Power Users	Customer 123	Customer 456	Customer 789
Page1	True	True	True
Page2	True	True	True
Page3	True	True	True

User123	Customer 123	Customer 456	Customer 789
Page1	True	False	False
Page2	True	False	False
Page3	True	False	False

User1	Customer 123	Customer 456	Customer 789
Page1	True	True	True
Page2	False	False	False
Page3	False	False	False

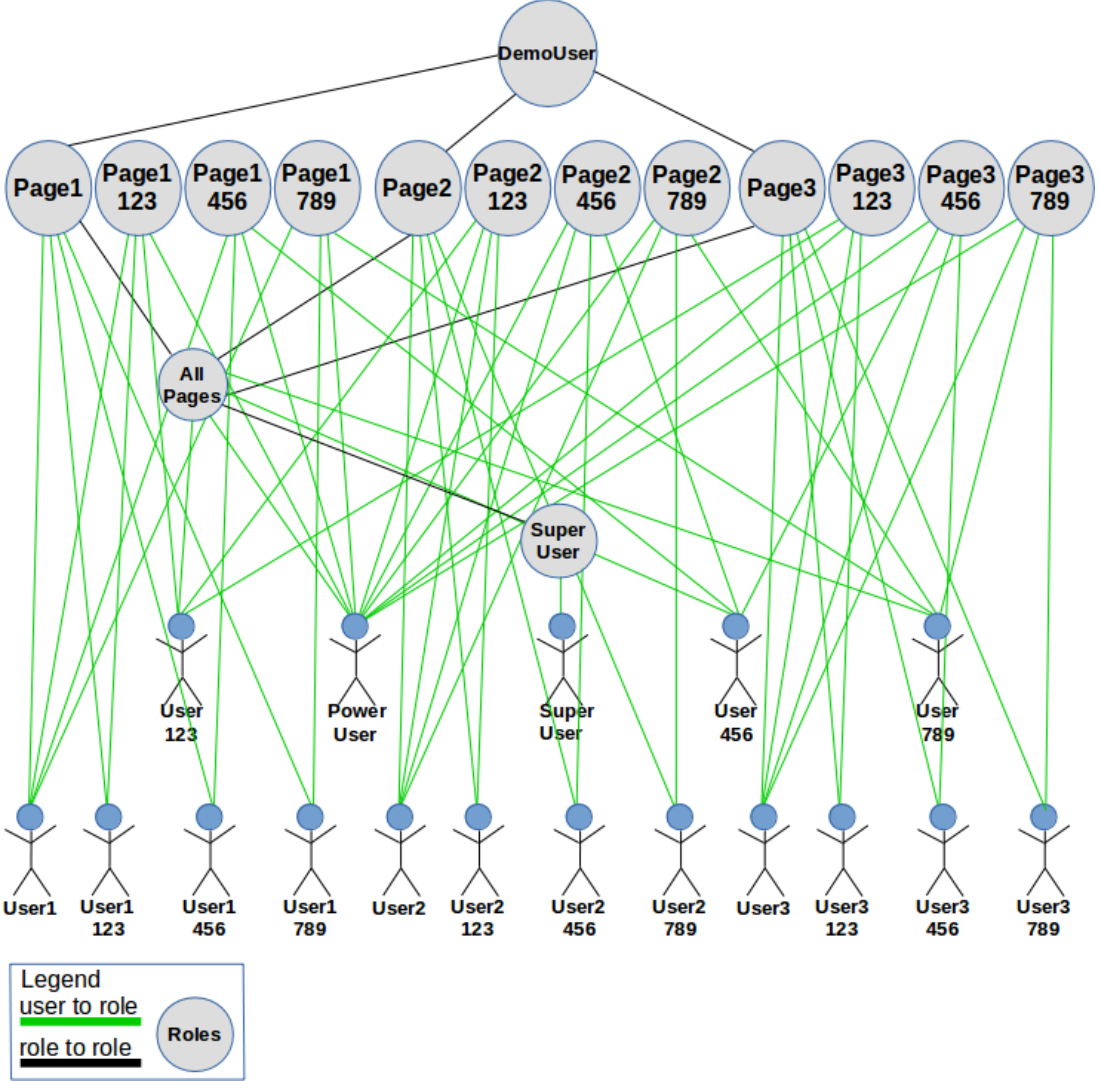
User1_123	Customer 123	Customer 456	Customer 789
Page1	True	False	False
Page2	False	False	False
Page3	False	False	False

Apache Fortress Demo

- https://github.com/shawnmckinney/apache-fortress-demo

User-tic-tac-toe	Customer 123	Customer 456	Customer 789
Page1	False	True	True
Page2	True	False	False
Page3	True	False	False

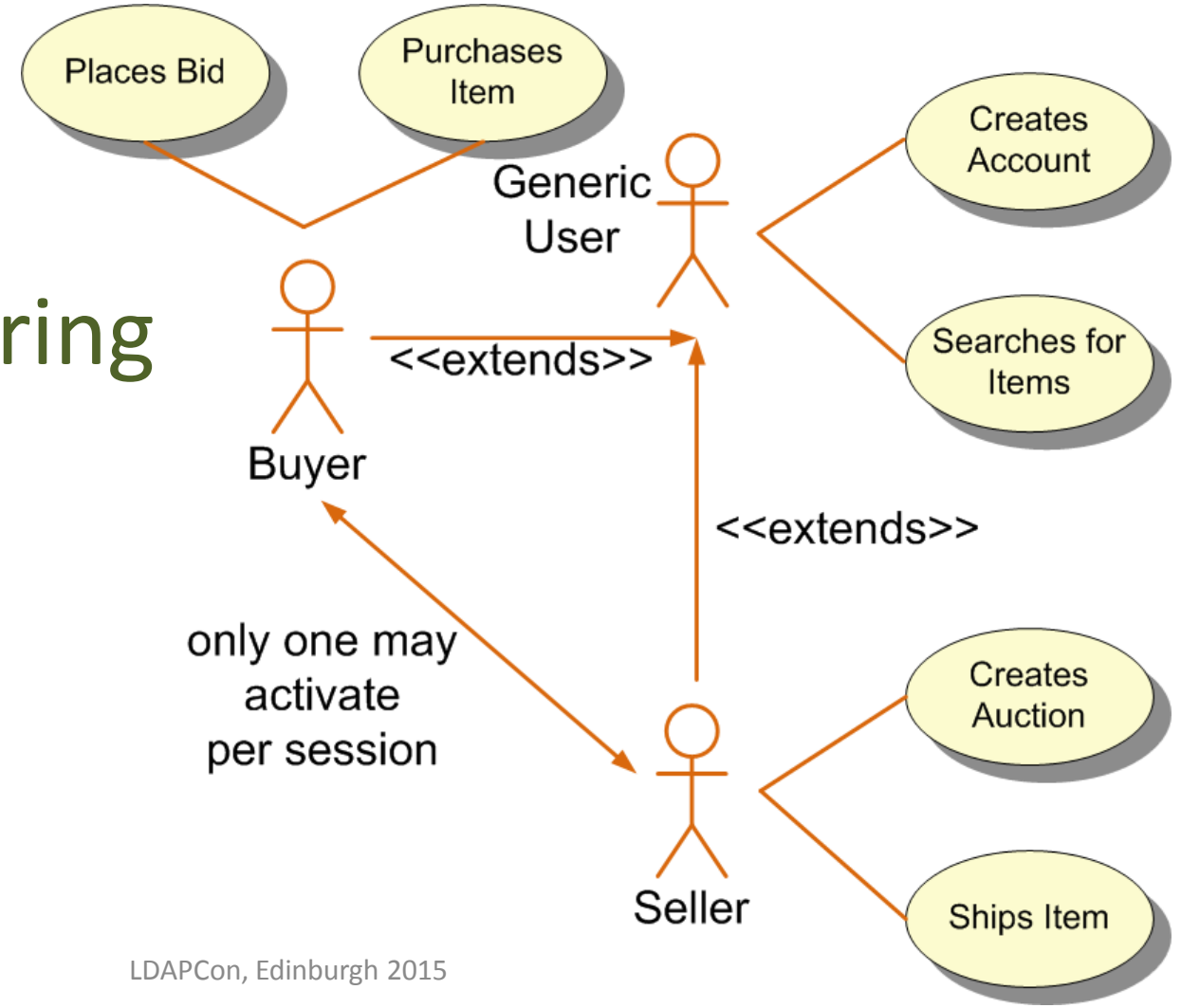
Apache Fortress End-to-End Security Tutorial



symas

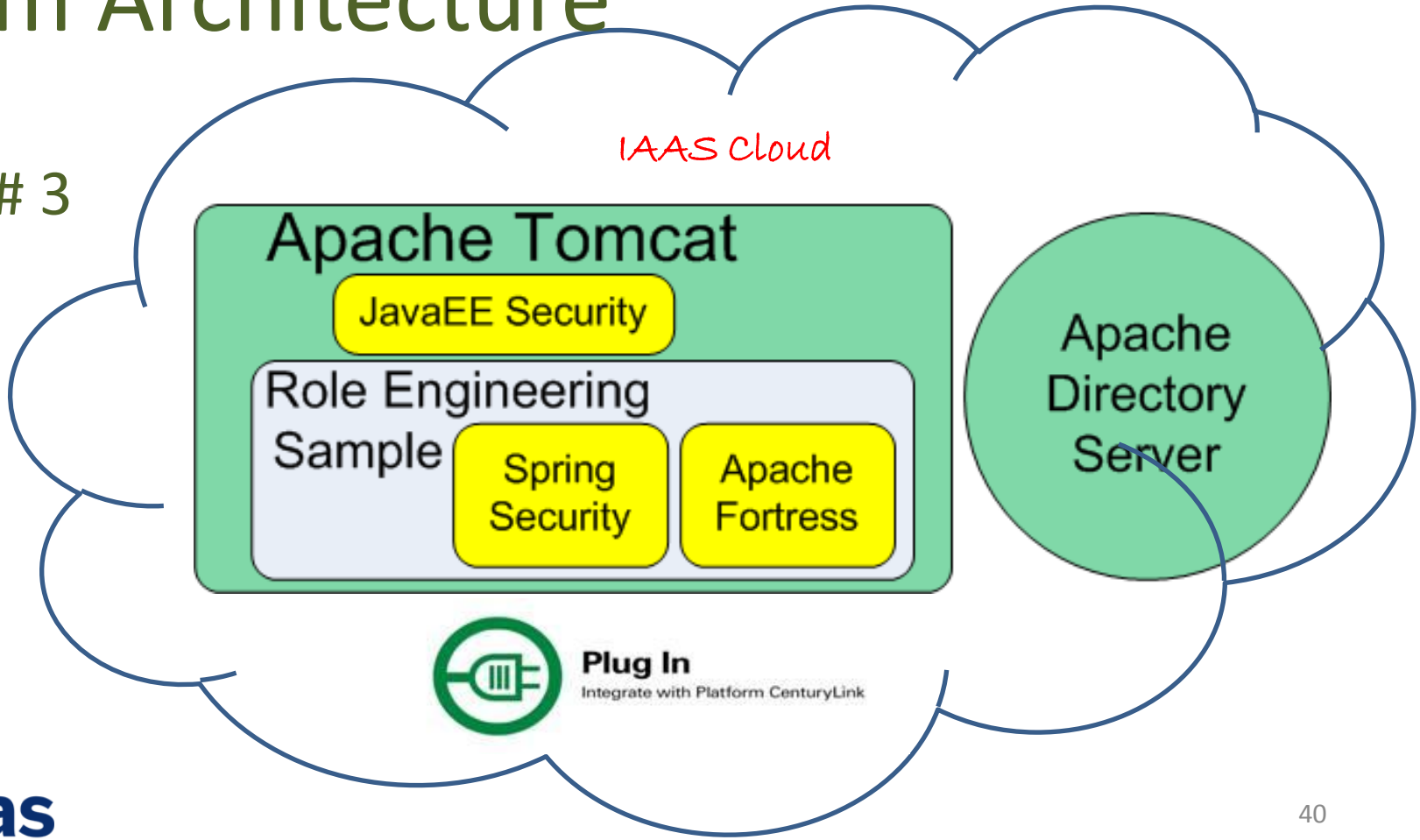
Tutorial #3

Role Engineering Sample Security Use Case



System Architecture

Tutorial # 3



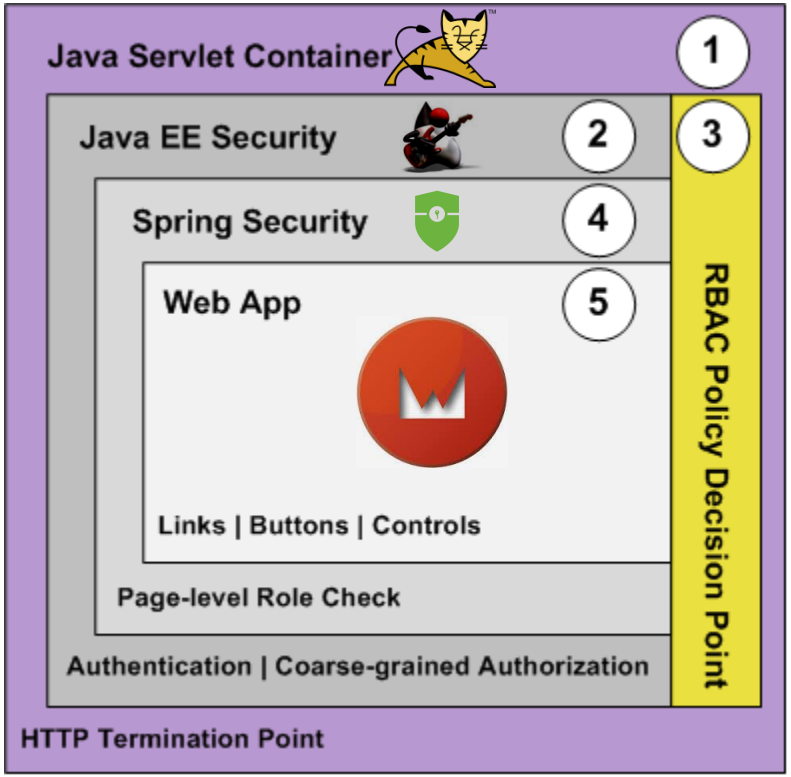
symas

Role Engineering Sample Security

1. Java EE Authentication and Authorization
2. Spring Page-level Authorization
3. RBAC Permission Checks
   - Links
   - Buttons
4. Other RBAC Controls
   - Dynamic Separation of Duty
   - Role Switcher

Declarative

Role Engineering Sample



https://github.com/shawnmckinney/role-engineering-sample

HTTP LDAPv3

1. HTTP server
2. Java EE AuthN & AuthZ
3. RBAC Policy Decision Point
4. Spring AuthZ
5. Web App AuthZ

Role Engineering Sample

- Two pages
- Each has buttons controlled by RBAC Permissions.
- One Role per page.

User to Role	Buyers Page	Sellers Page
ssmith	True	False
jtaylor	False	True
Johndoe*	True	True

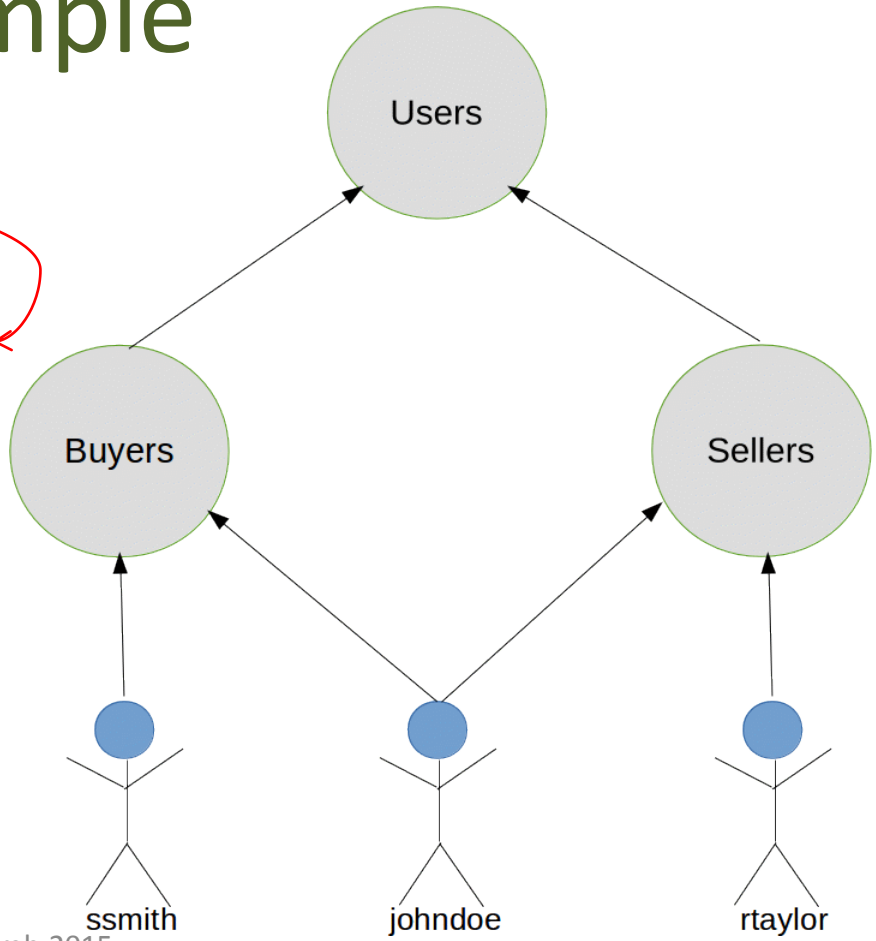
* DSD constraint limits user from activating both roles simultaneously.

Role Engineering Sample

Github link to

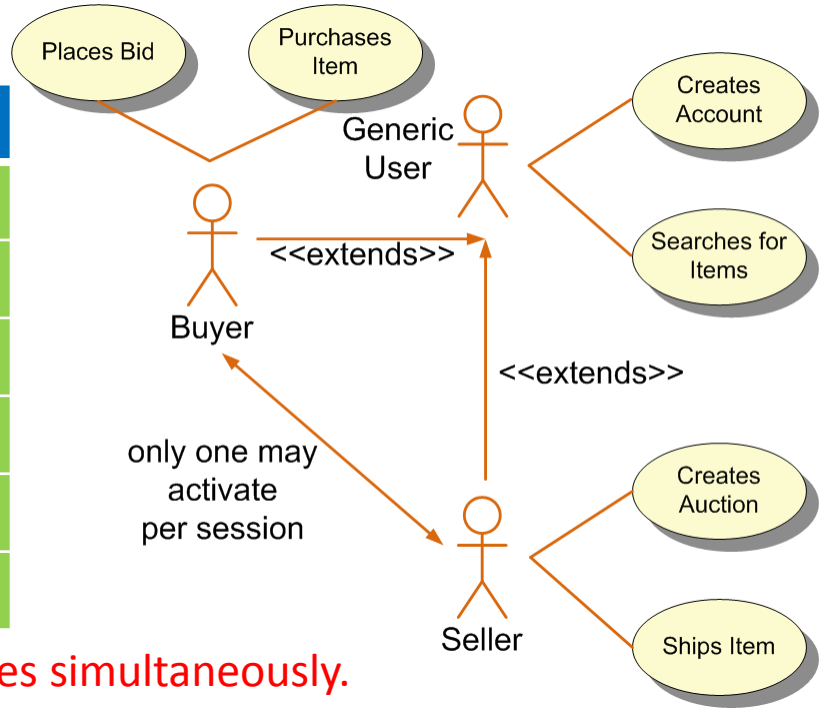
[Role Engineering Sample Policy File]

User	Buyers	Sellers
ssmith	True	False
johndoe	True	True
rtaylor	False	True



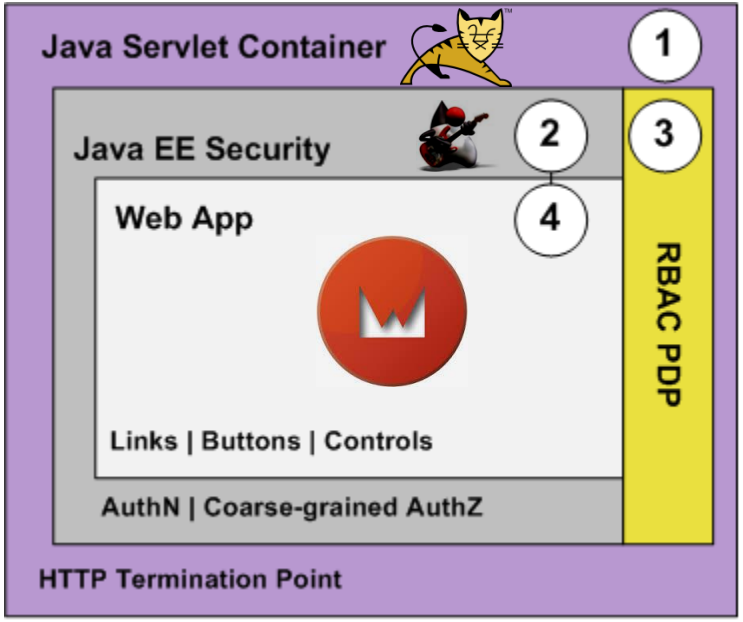
Role Engineering Sample

		Buyer	Seller	Both
	Permission	**ssmith**	**rtaylor**	**johndoe***
1	Item.bid	True	False	True
2	Item.purchase	True	False	True
3	Item.ship	False	True	True
4	Item.search	True	True	True
5	Account.create	True	True	True
6	Auction.create	False	True	True



* DSD constraint limits user from activating both roles simultaneously.

Tutorial # 3 - Demo Wicket Sample



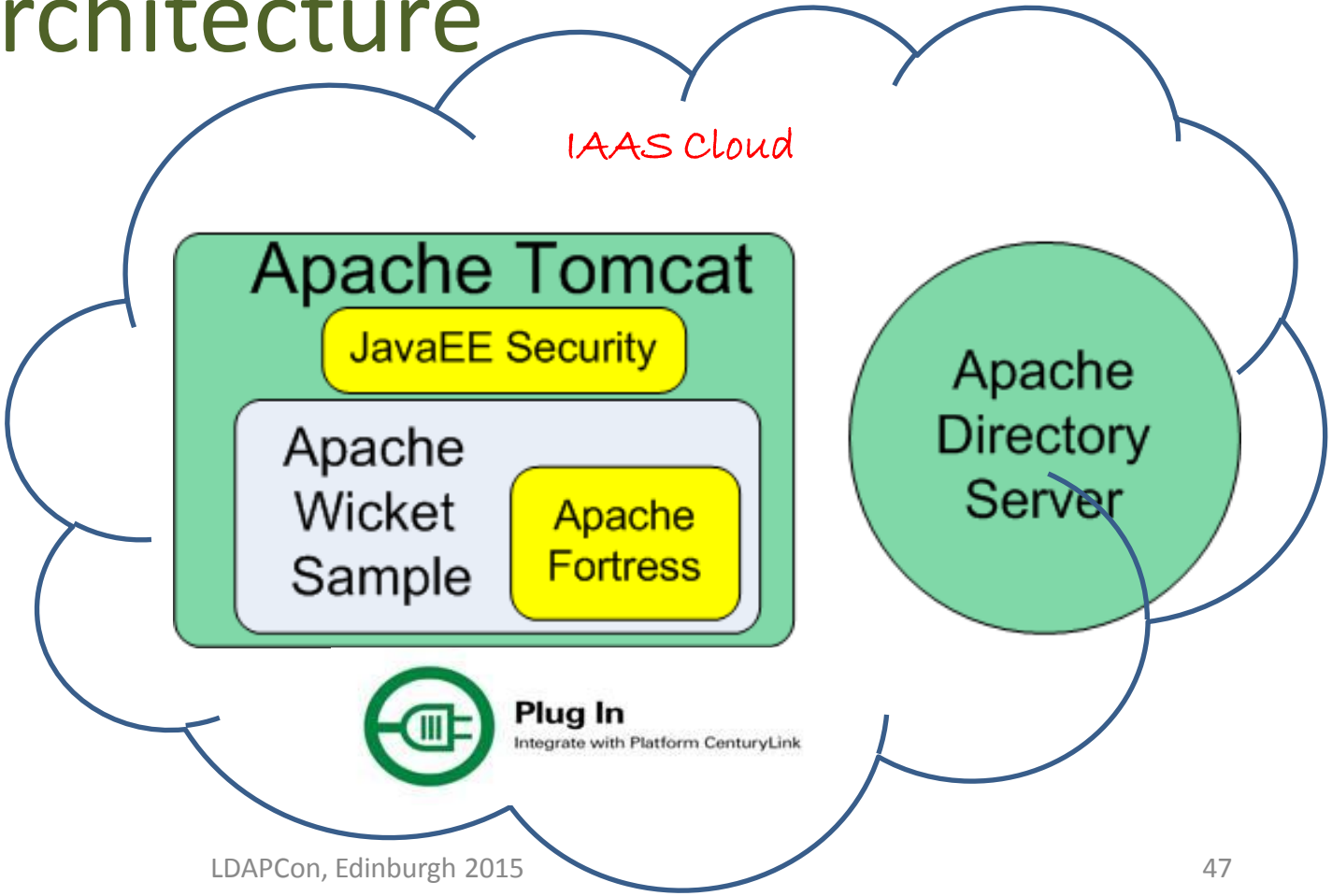
https://github.com/shawnmckinney/wicket-sample

HTTP LDAPv3

1. HTTP server
2. Java EE AuthN & AuthZ
3. RBAC Policy Decision Point
4. Web App AuthZ

System Architecture

Tutorial # 3



Security Layers with Wicket Sample

1. JSSE ← Confidentiality and Integrity
2. Java EE Security ← authN and coarse-grained authZ
3. Web App Framework ← fine-grained authZ

Add Web Framework Security

```
add(
   new SecureIndicatingAjaxButton( "Page1", "Add" )
   @Override
   protected void onSubmit( ... )
   {
      if( checkAccess( customerNumber ) )
      {
        // do something here:
      }
      else
      {
         target.appendJavaScript( ";alert('Unauthorized');" );
      }
   }
   });
```

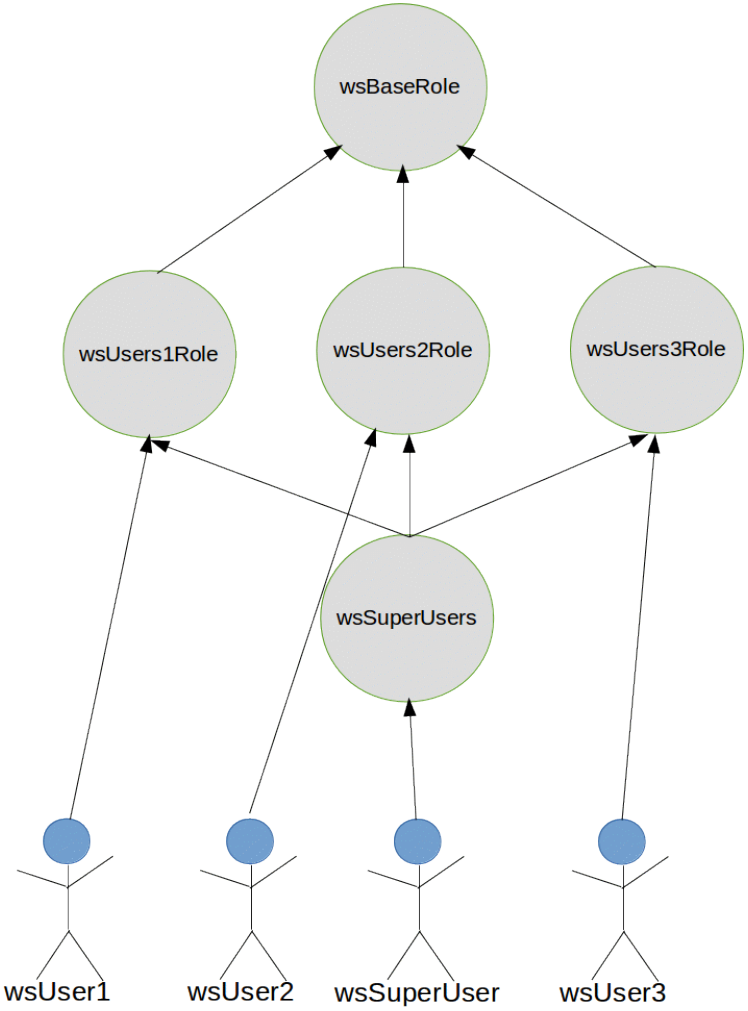
fine-grained authorization (programmatic)

Demo Wicket Sample

Github link to

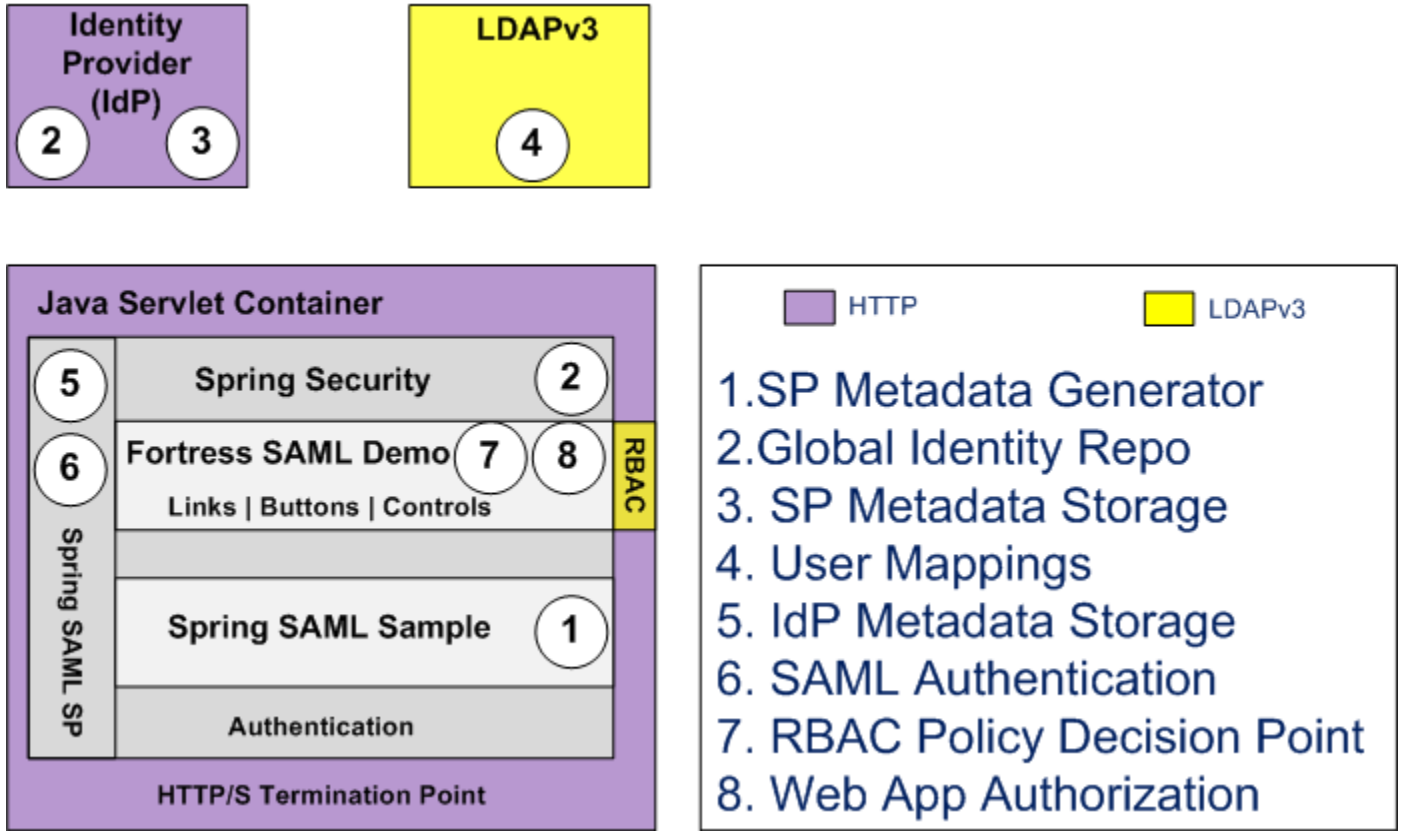
[Wicket Sample Policy File]

User	Page1	Page2	Page3
wsUser1	True	False	False
wsUser2	False	True	False
wsUser3	False	False	True
wsSuperUser	True	True	True



Tutorial #4

Apache Fortress SAML Demo



http://iamfortress.net/2015/09/01/apache-directory-fortress-saml-demo/

symas

System Architecture

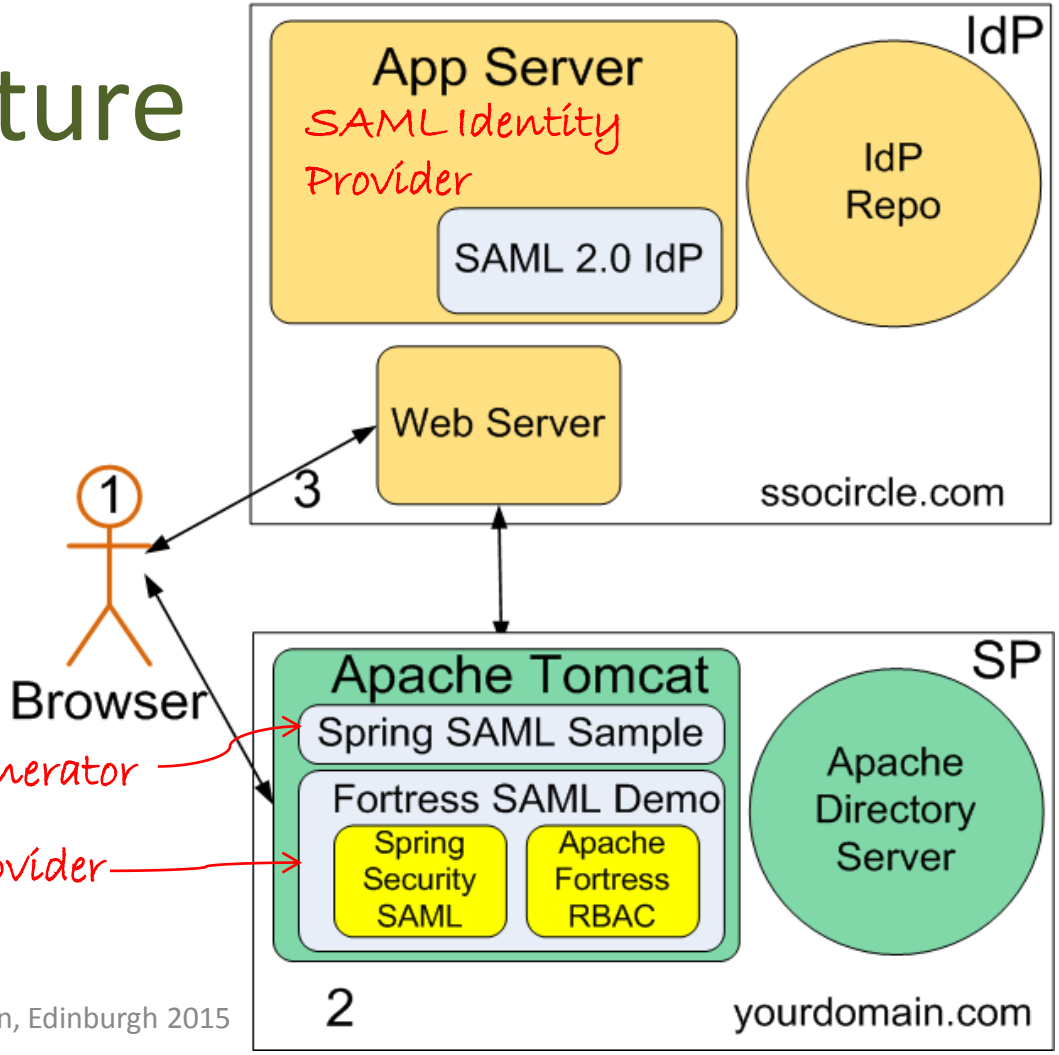
Tutorial # 2

Use

SSO Circle

Identity

Provider



Security Layers with SAML Demo

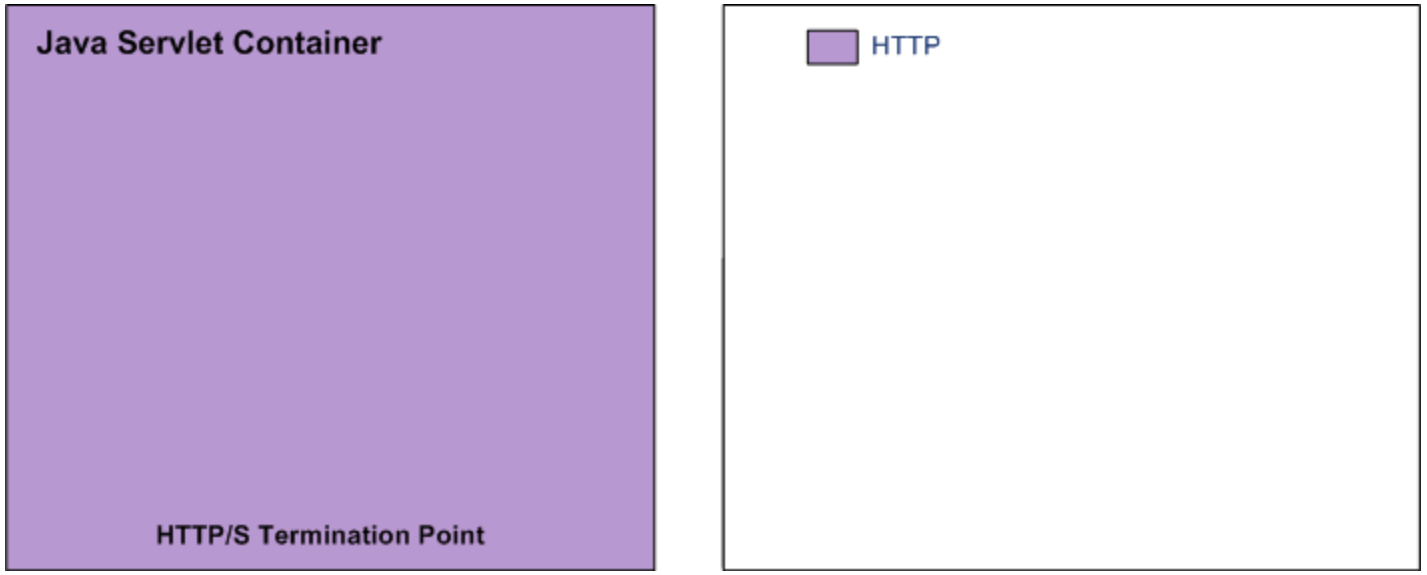
1. JSSE
2. ~~Java EE Security~~ ← Turned off (for now)
3. Spring Security ← SAML 2.0 authN
4. Web App Framework ← Fine-grained AuthZ

Two Areas of Access Control

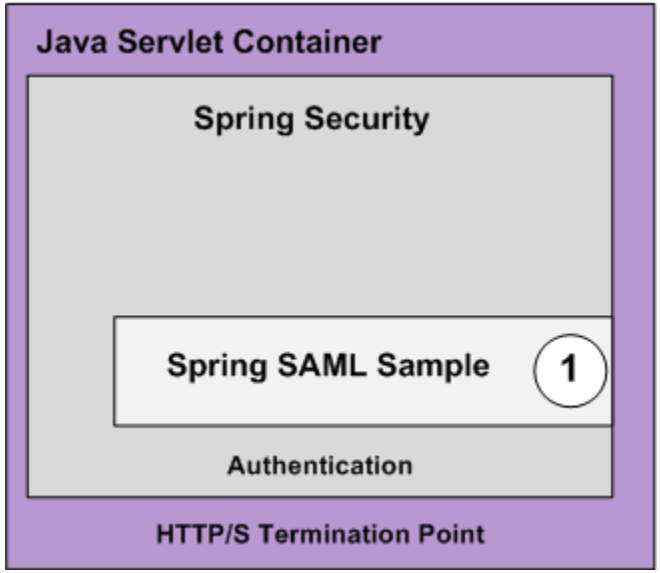
1. Spring Security Declarative checks

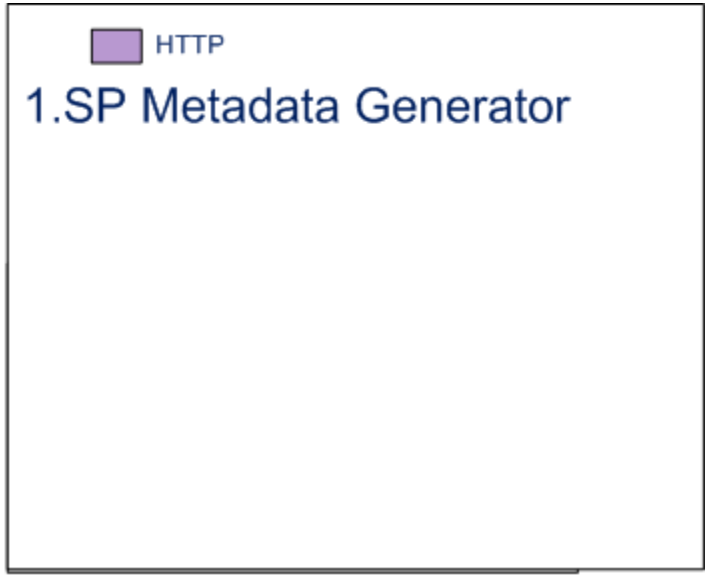
2. Apache Fortress Programmatic checks

Start with Tomcat Servlet Container



1. Deploy the Spring SAML Sample





Get the Spring SAML Sample

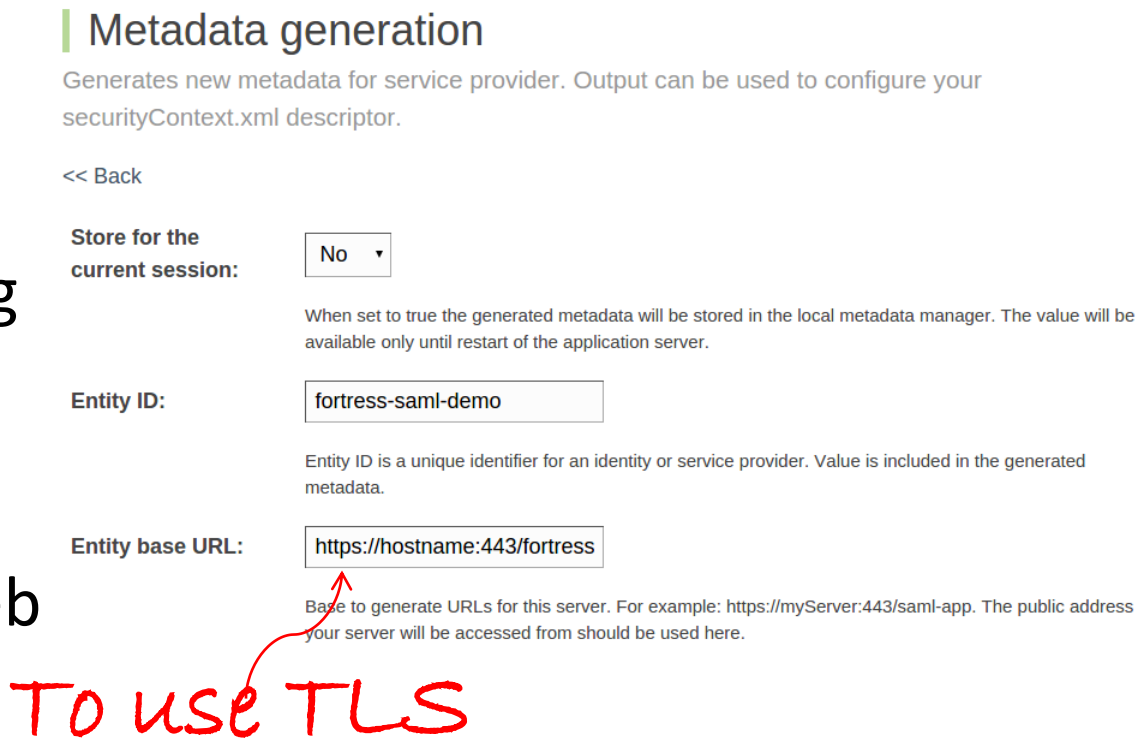
Pick one:

1. spring-security-saml - Spring's SAML sample is the first place java developers should look for basic SAML 2.0 programming concepts.
2. shibboleth-sample-java-sp - Unicon's is derived from above and is how to learn about Spring SAML's SP with a Shibboleth's IdP.

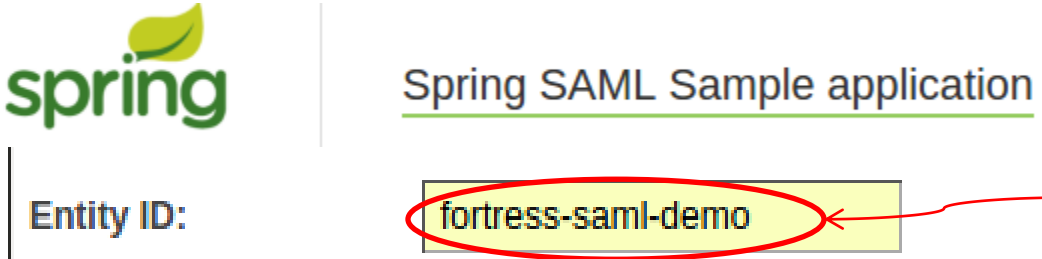
Generate SAML Service Provider Metadata

Matching Fields:

- Entity ID must match Spring config in web app
- Entity base URL must match the web app's URL.



Spring SAML Metadata Generation Tip

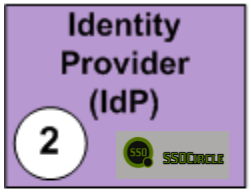


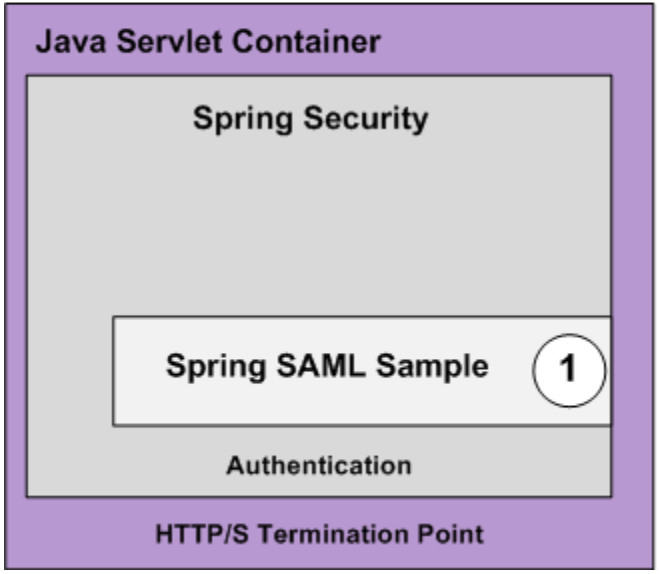
These entityId's must match

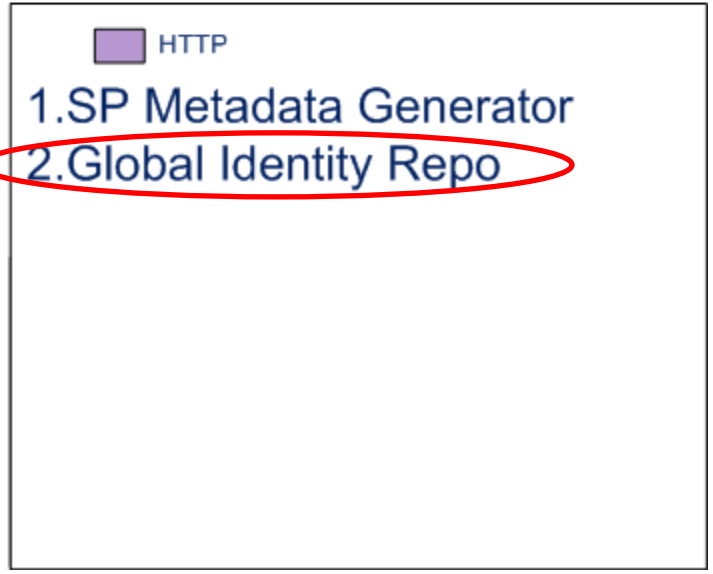
```
<bean id="metadataGenerator filter" class="org.springframework...MetadataGeneratorFilter">
  <constructor-arg>
  <bean class="org.springframework...MetadataGenerator">

    <property name="entityId" value="fortress-saml-demo"/>

  </constructor-arg>
</bean>
```

Bind the service provider with the IdP.

2. Setup Global Identity Provider







Setup SSOCircle SAMLv2.0 IdP

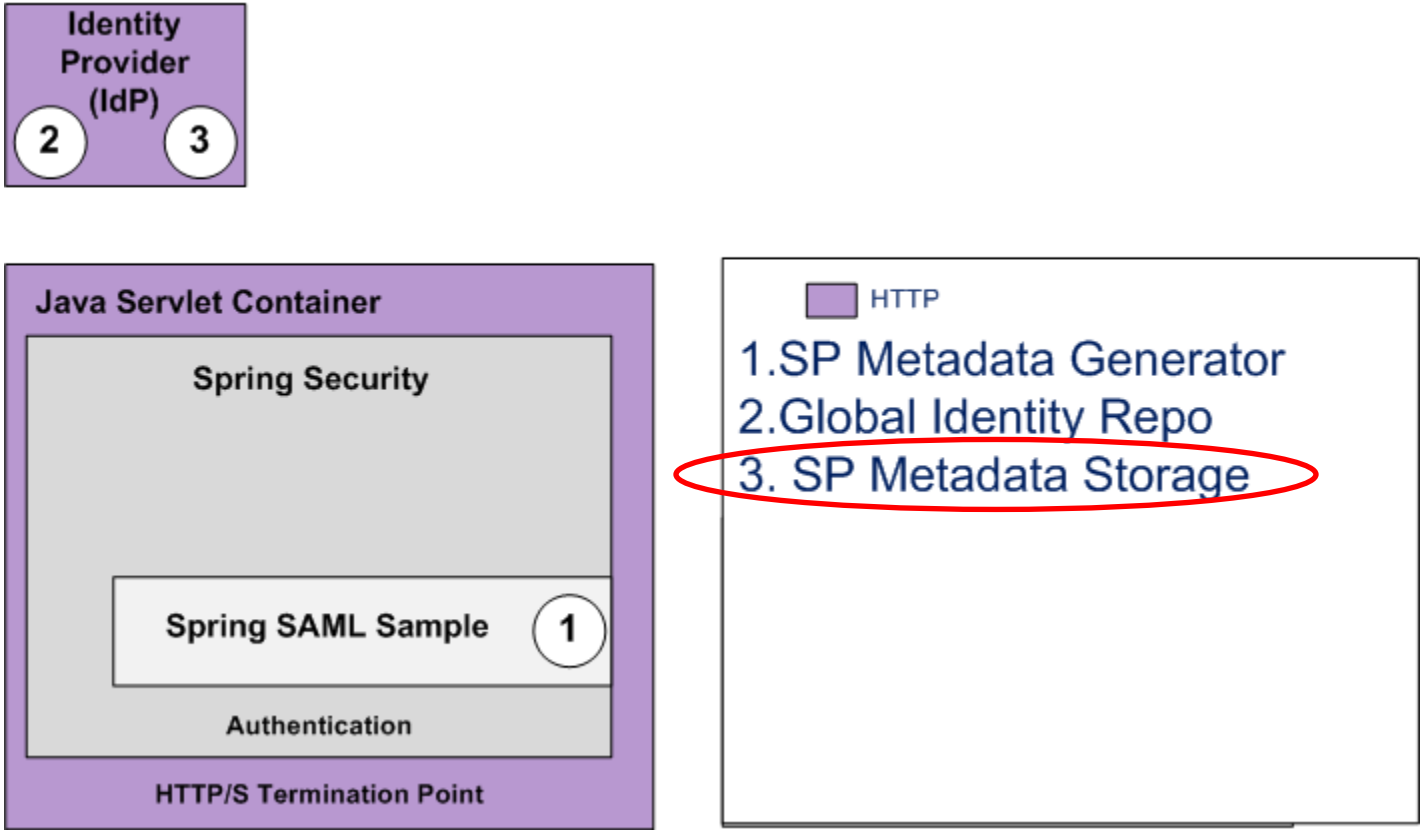
Creating your Identity with SSOCircle (from their website)

For creating your account you need to follow a few steps:

- Register at the SSOCircle SAMLv2.0 Identity Provider
- Provide the required data
- Agree to the Terms of Use
- After successful creation you will receive an email asking for confirmation of your registration. Confirm by navigating to the link supplied in the email.
- Now your account is activated and ready for use.

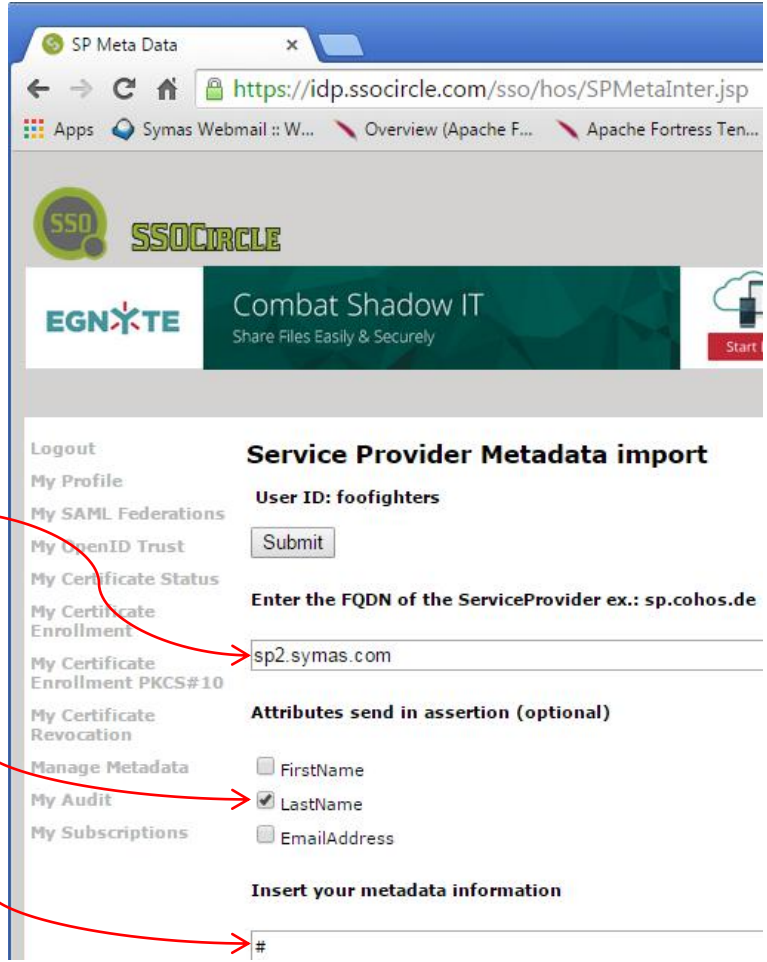
http://www.ssocircle.com/en/portfolio/publicidp/

3. Import Service Provider Metadata into IdP



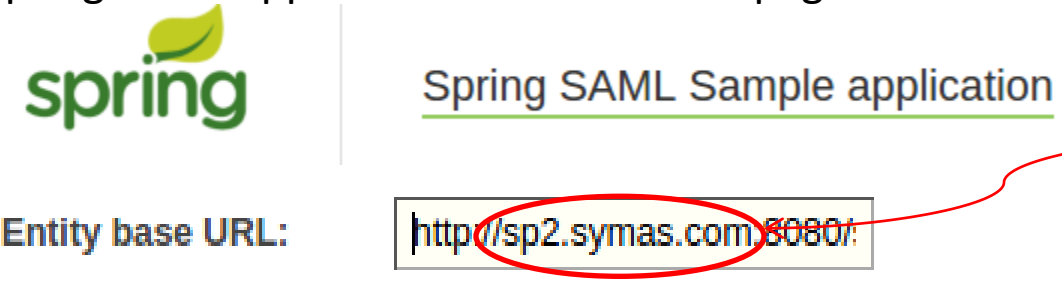
Import SP Metadata

- Logon SSOCircle
- Click on ***Manage Metadata***
- ***FQDN*** must match SP's host name
- Check the ***LastName*** box
- Paste your metadata here



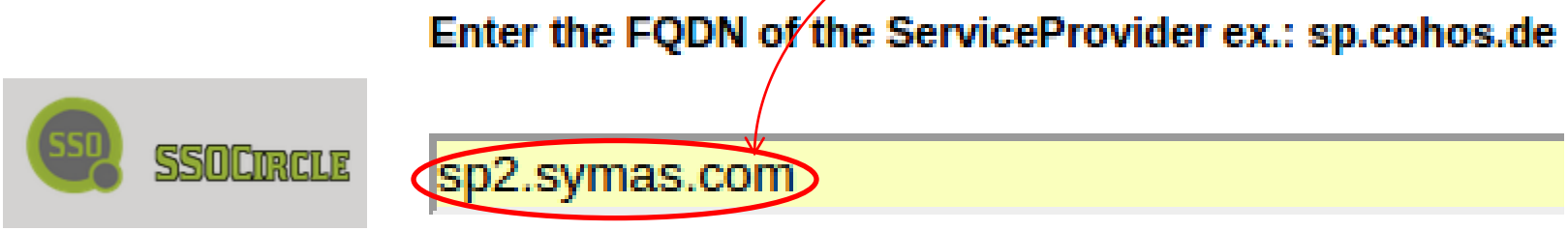
Import SP Metadata Tip

Spring SAML app Metadata Generation page:

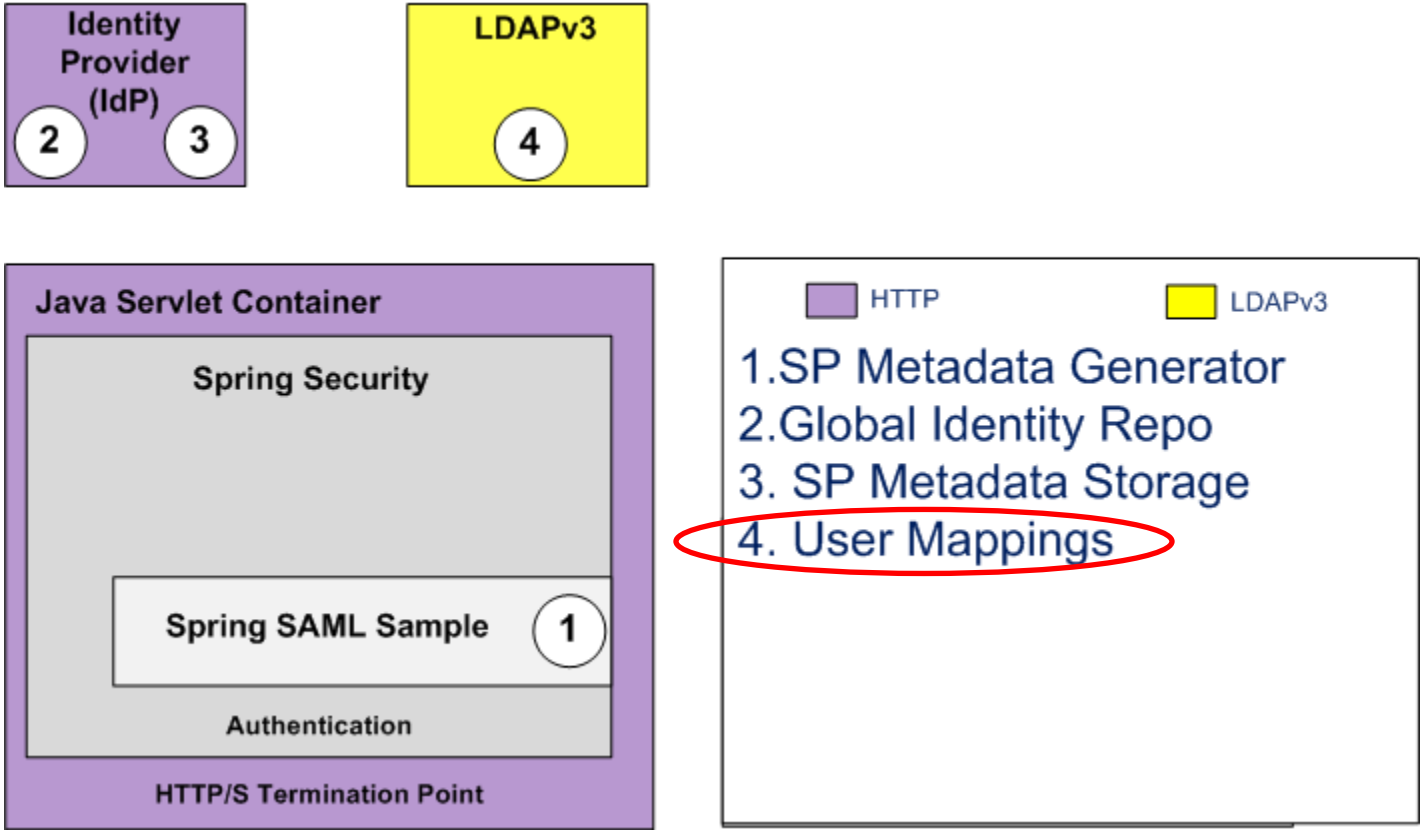


The FQDN matches base url from SP metadata gen step

SSOCircle Service Provider Metadata Import page:



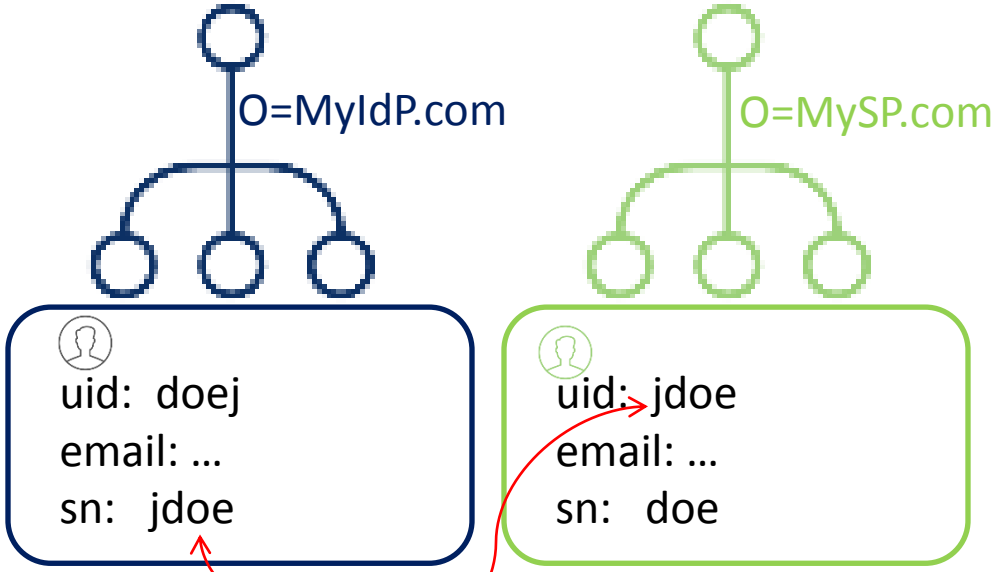
4. IdP and SP User Account Mapping



IdP and SP User Account Mapping

1. Mapping rules are specific to partners.

2. The mapping must be a one-to-one unique pairing.



fortress saml demo maps the sn on the IdP-side with uid field on the SP-side

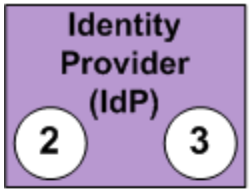
SAML Attribute Statement

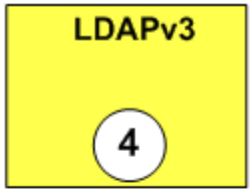
```
<?xml version="1.0" encoding="UTF-8"?><samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol”
Destination="http://sp2.symas.com:8080/fortress-saml-demo/saml/SSO"
…
<saml:AttributeStatement>
…
  <saml:Attribute Name="LastName">
  <saml:AttributeValue …
      xsi:type="xs:string">sam3</saml:AttributeValue>
  </saml:Attribute>
 </saml:AttributeStatement>
…
</samlp:Response>
```

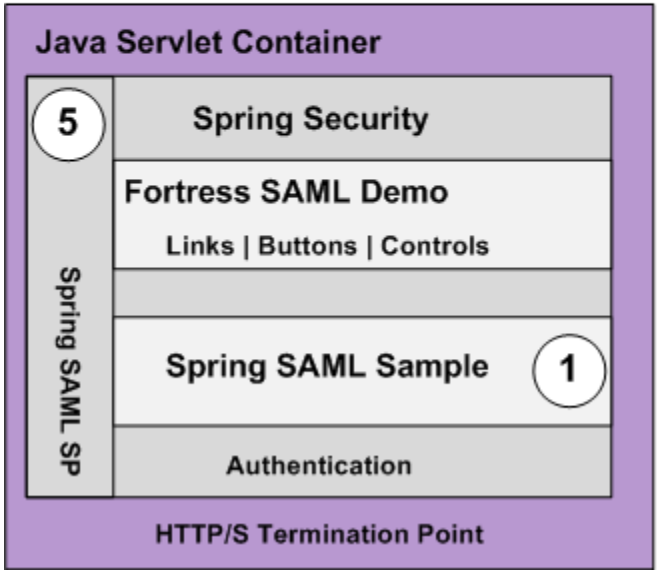
host name entered during SP Metadata import

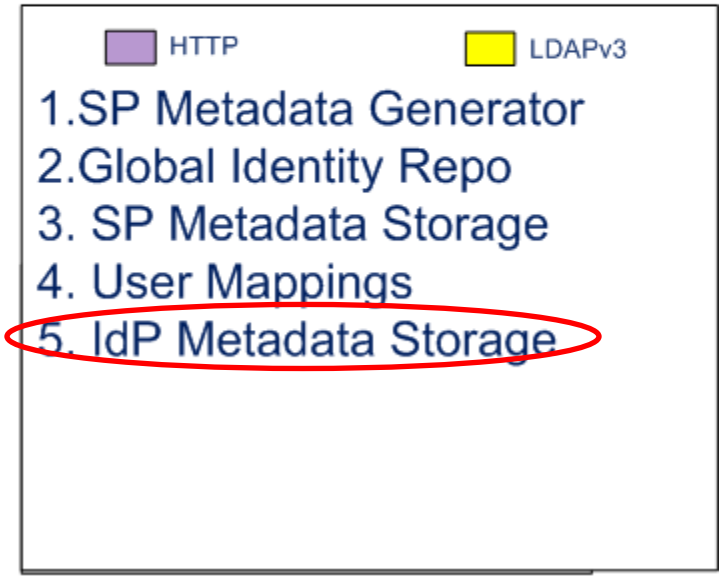
Last Name linked to userid in rbac

5. Load IdP Metadata into Service Provider









Point SP to SAML IdP

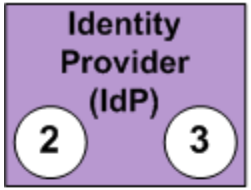
Point to the Identity Provider in securityContext.xml

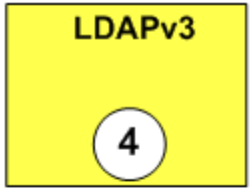
```
<bean id="metadata" class="org.springframework.security.saml.metadata.CachingMetadataManager">
 <constructor-arg>
  <list>
   <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider">
    <constructor-arg>
     <value type="java.lang.String"
```

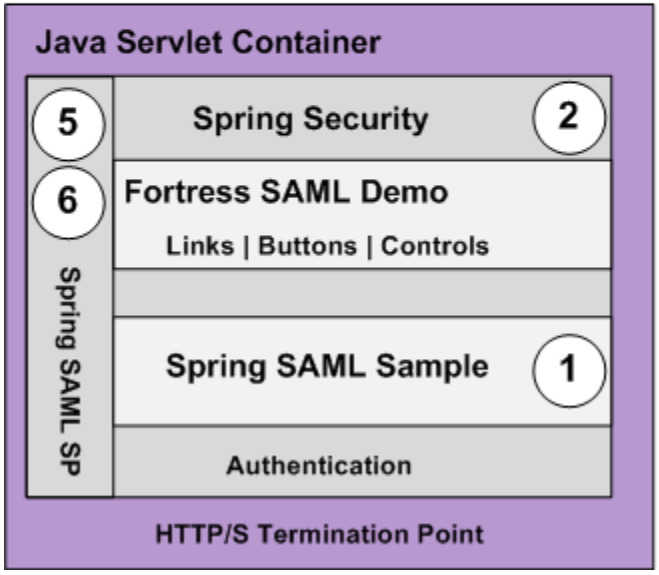
http://idp.ssocircle.com/idp-meta.xml

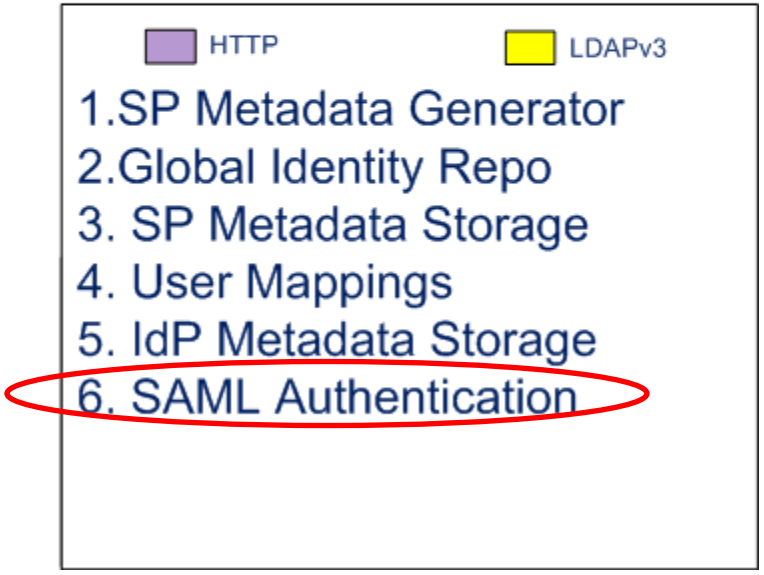
```
     </value>
    </constructor-arg>
    <constructor-arg>
     <value type="int">5000</value>
     </constructor-arg>
     <property name="parserPool" ref="parserPool"/>
   </bean>
  </list>
 </constructor-arg>
</bean>
```

6. Enable Spring SAML Authentication









Enable Spring SAML Security

Add dependencies to pom:

```
<dependency>
  <groupId>org.springframework.security.extensions</groupId>
  <artifactId> spring-security-saml2-core </artifactId>
  <version>1.0.1.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId> spring-security-config </artifactId>
  <version> 3.1.2.RELEASE* </version>
  <scope>compile</scope>
</dependency>
```

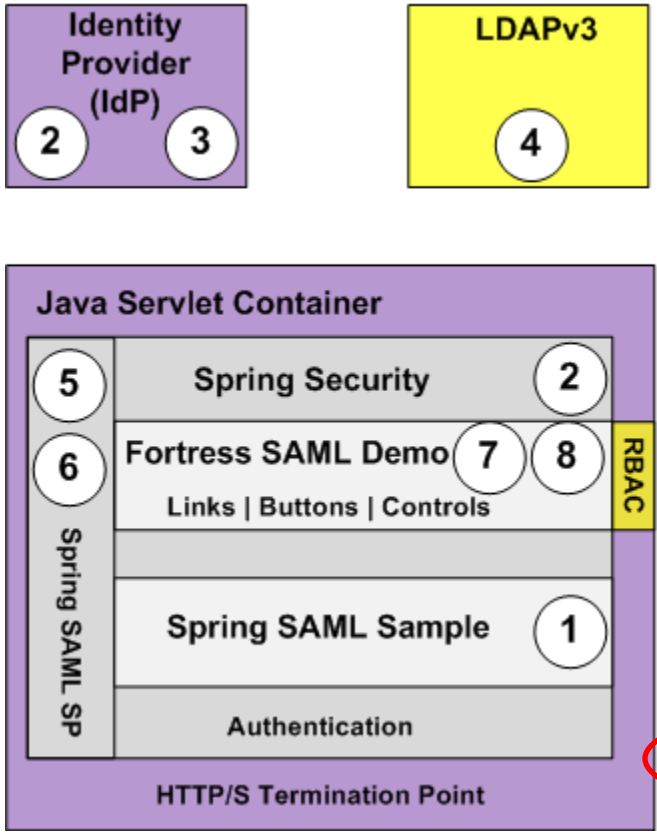
* backlog item

Enable SAML Authentication Filters

In the securityContext.xml

```
<security:http entry-point-ref="samlEntryPoint" use-expressions="false">

  <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
      <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
  <security:custom-filter before="FIRST" ref="metadataGeneratorFilter"/>
  <security:custom-filter after="BASIC_AUTH_FILTER" ref="samlFilter"/>
</security:http>

<bean id="samlFilter" class="org.springframework.security.web.FilterChainProxy">
    <security:filter-chain-map request-matcher="ant">
    <security:filter-chain pattern="/saml/login/**" filters="samlEntryPoint"/>
    <security:filter-chain pattern="/saml/logout/**" filters="samlLogoutFilter"/>
    <security:filter-chain pattern="/saml/metadata/**" filters="metadataDisplayFilter"/>
    <security:filter-chain pattern="/saml/SSO/**" filters="samlWebSSOProcessingFilter"/>
    <security:filter-chain pattern="/saml/SSOHoK/**" filters="samlWebSSOHoKProcessingFilter"/>
    <security:filter-chain pattern="/saml/SingleLogout/**" filters="samlLogoutProcessingFilter"/>
    </security:filter-chain-map>
  </bean>
```

7. Setup RBAC Policy Decision Point



Enable RBAC Policy Decision Point

```
<dependency>
  <groupId>
    org.apache.directory.fortress
  </groupId>
  <artifactId>
      fortress-realm-impl
  </artifactId>
  <version>1.0</version>
</dependency>
```

Identity Propagation SAML->RBAC

1. Spring SAML filter creates security principal based on attributes found in the SAML attribute assertion.
2. Web app calls to get the Security Principle from Spring:

```
ServletContext.getUserPrinciple();
```

standard API call

3. Web app parses the attributes contained within principal :

```
uid=getSurName((SAMLCredential)principal.getCredentials());
```

4. Web app creates a new RBAC session using attribute(s) pulled from the principal :

```
j2eePolicyMgr.createSession( new User( uid ), true );
```

isTrusted

5. Web app pushes RBAC session into HTTP session.

Apache Fortress Saml Demo

- Three Pages
- Each has buttons controlled by RBAC permissions.
- One role per page.
- Users may be assigned to one or more roles.

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True

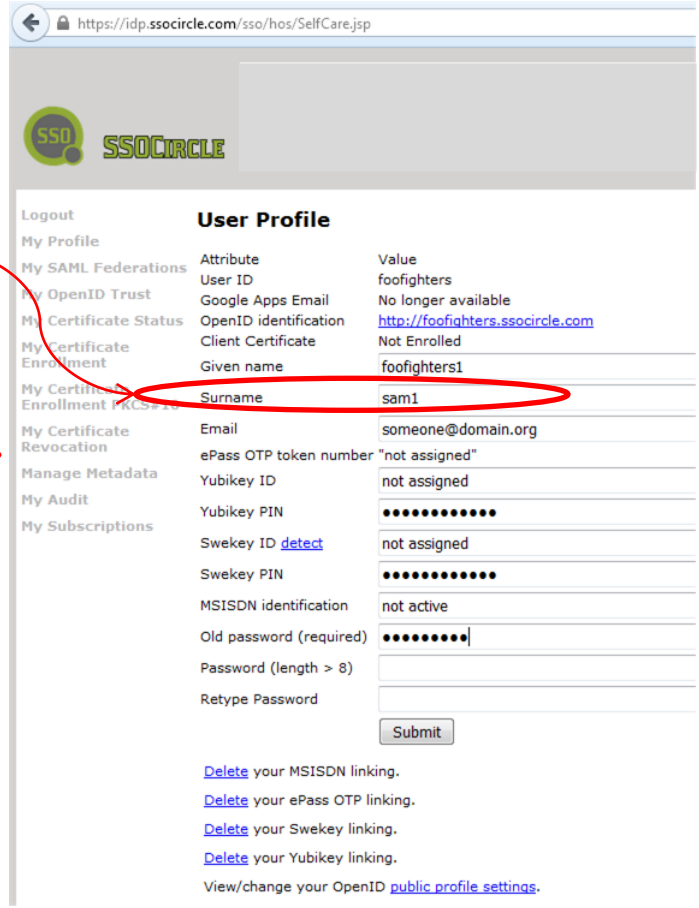
To Change Demo Users

uid=sam1,ou=People,dc=example,dc=com

DN: uid=sam1,ou=People,dc=example,dc=com

Attribute Description	Value
objectClass	***extensibleObject (auxiliary)***
objectClass	***ftMods (structural)***
objectClass	***ftProperties (structural)***
objectClass	***ftUserAttrs (structural)***
objectClass	***inetOrgPerson (structural)***
objectClass	***organizationalPerson (structural)***
objectClass	***person (structural)***
objectClass	***top (abstract)***
cn	**Sam One**
sn	**One**
description	Fortress SAML Demo User 1
displayName	Sam One
ou	org.samlsample.users
uid	sam1
userPassword	SSHA hashed password
ftCstr	*sam1$0$$$$$$$*
ftId	*a59bf210-7101-4bb9-b089-9205d594d108*
ftProps	*init:*
ftRA	*samRole1*

Change Surname field in SSO Circle Profile to use different rbac users.

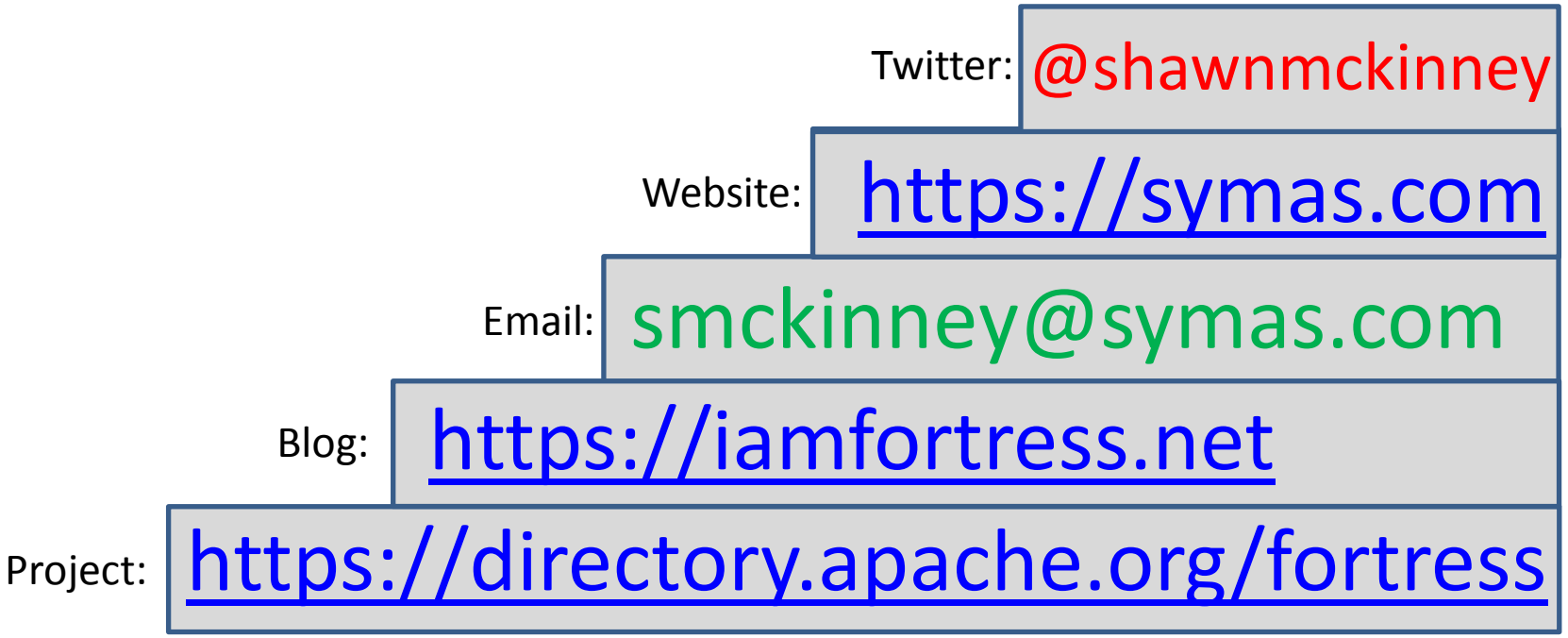


Apache Fortress SAML Demo

- https://github.com/shawnmckinney/fortress-saml-demo

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True

Contact Me


Twitter: @shawnmckinney
Website: https://symas.com
Email: smckinney@symas.com
Blog: https://iamfortress.net
Project: https://directory.apache.org/fortress