

The Anatomy of a Secure Web App Using JavaEE, Spring Security and Apache Directory Fortress

Shawn McKinney

ApacheCon : CORE Europe




October 1, 2015



Introductions

Shawn McKinney



-  **symas** Systems Architect
-  PMC Apache Directory Project
- Open  **LDAP**™ Engineering Team

Agenda

Learn Security via Two Examples:

1. Apache Fortress End-to-End Security Tutorial
 - Java EE Container Managed Security
2. Apache Fortress SAML Demo
 - SAML 2.0 Single Sign-On

Themes Covered

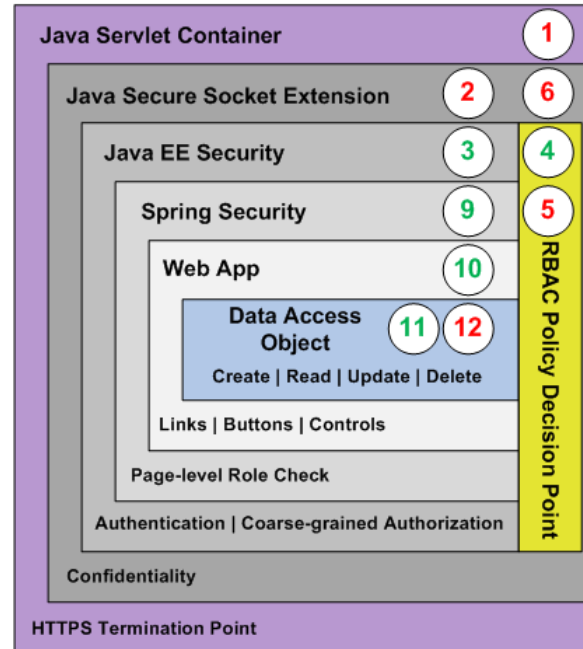
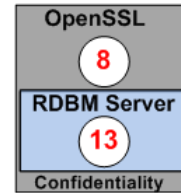
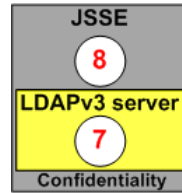
1. Simplicity

2. Common Sense

3. Household Analogies to explain
'Why'

Tutorial #1

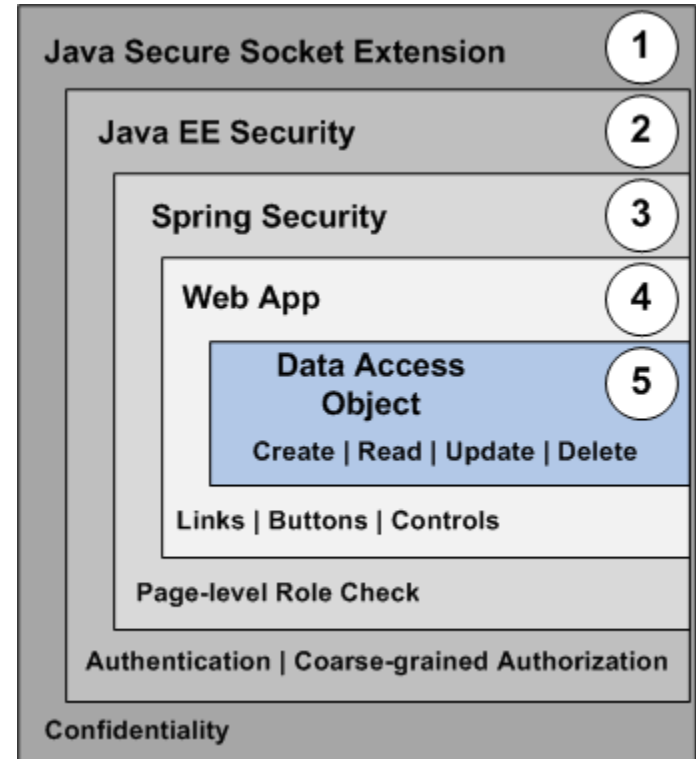
Apache Fortress End-to-End Security Tutorial



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

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



The Five Security Layers of Java Web Apps

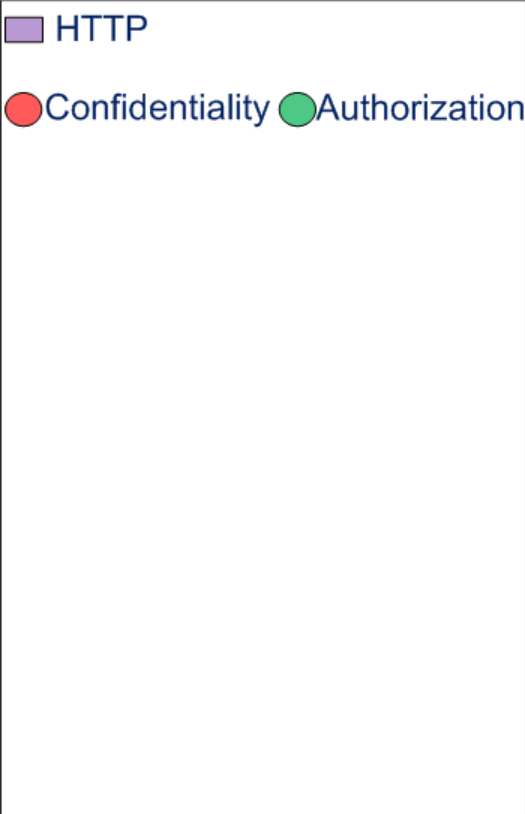
1. JSSE ← *Private conversations*
2. Java EE Security ← *Deadbolt on front door*
3. Spring Security ← *Locks on room doors*
4. Web App Framework ← *Locks on room equipment*
5. Database Functions ← *Media content filtering*

Two Areas of Access Control

1. Java EE and Spring Role Declarative checks

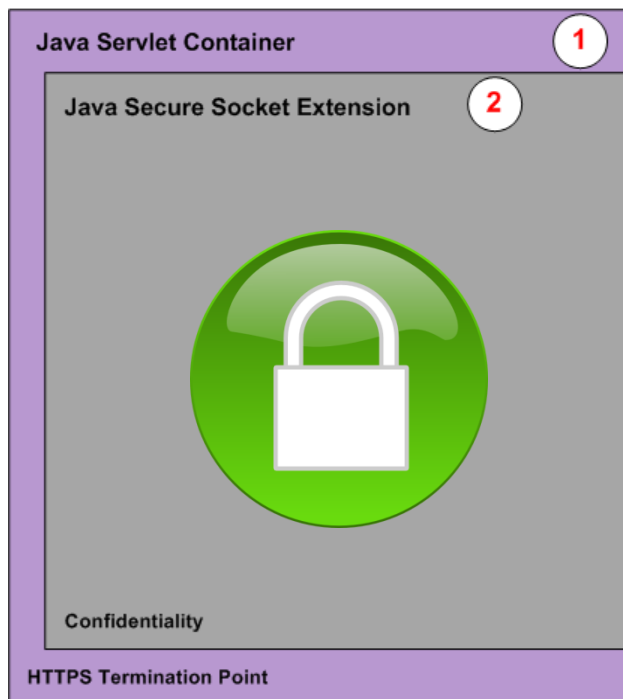
2. RBAC Permission Programmatic checks

Start with Tomcat Servlet Container



1 & 2. Enable HTTPS

1. Update the Server.xml
2. Add private key



■ HTTP

● Confidentiality ● Authorization

1. HTTPS server
2. HTTPS private key

Enable Tomcat TLS

1. Generate keystore with private key (Steps 1 - 5):

<https://symas.com/javadocs/apache-fortress-demo/doc-files/keys.html>

2. Add the following to server.xml:

```
<Connector port="8443" maxThreads="200" scheme="https" secure="true"
  SSLEnabled="true"
  keystoreFile= "/path/mykeystore"
  keystorePass= "changeit"
  clientAuth="false" sslProtocol="TLS"/>
```

*Keystore contains
private key*

<http://symas.com/javadocs/apache-fortress-demo/doc-files/apache-tomcat-ssl.html>

Change Tomcat TLS Enabled Cipher Suites

<Connector port="8443" ...

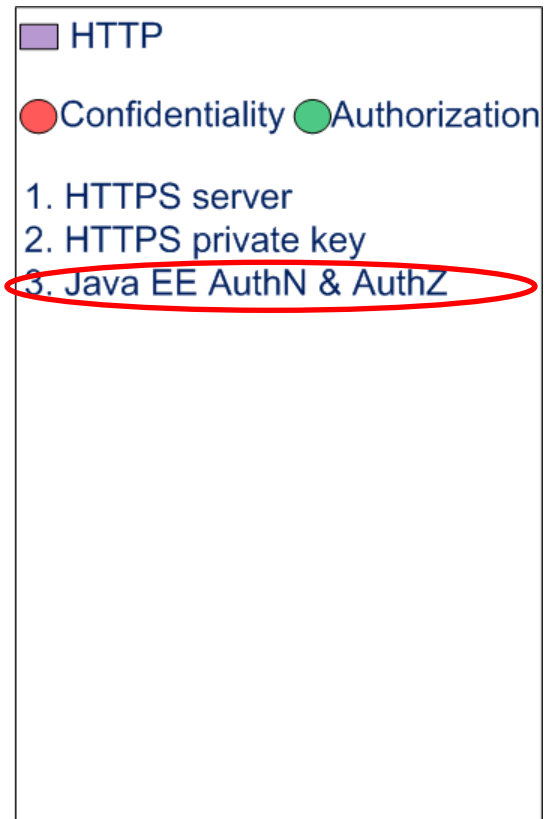
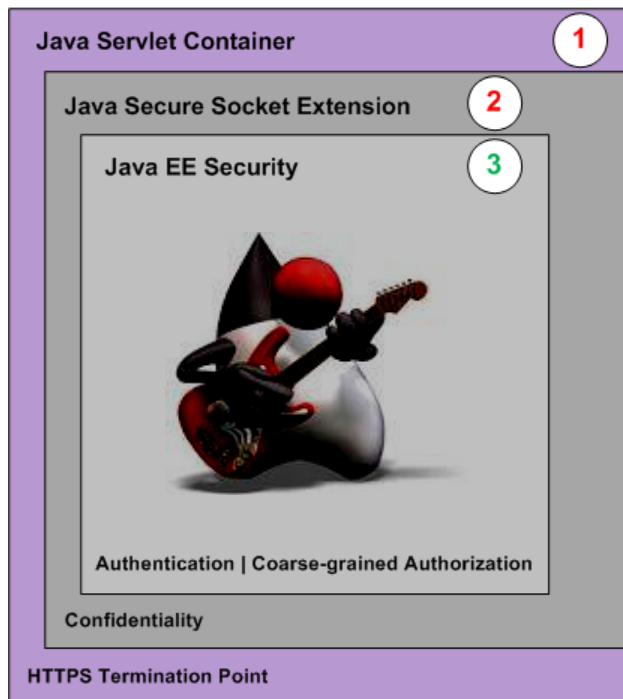
ciphers=

Disable weak Diffie-Hellman ciphers

```
"TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_ECDSA_WITH_RC4_128_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDH_RSA_WITH_RC4_128_SHA,TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_RC4_128_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_NULL_SHA,TLS_ECDH_RSA_WITH_NULL_SHA,TLS_ECDHE_ECDSA_WITH_NULL_SHA,TLS_ECDHE_RSA_WITH_NULL_SHA"
```

3. Enable Java EE Security

- Drop the proxy jar
- Update web.xml
- Add context.xml



Enable Java EE Security Realm

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

```
[root@IL1SC0LSP102 lib]# pwd
/usr/local/tomcat7/lib
[root@IL1SC0LSP102 lib]# ls -l fortress*
-rw-r--r-- 1 root root 15119 Aug 29 12:37 fortress-realm-proxy-1.0-RC41-SNAPSHOT.jar
[root@IL1SC0LSP102 lib]#
```

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

```
[root@IL1SC0LSP102 lib]# pwd
/usr/local/tomcat7/webapps/apache-fortress-demo/WEB-INF/lib
[root@IL1SC0LSP102 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@IL1SC0LSP102 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

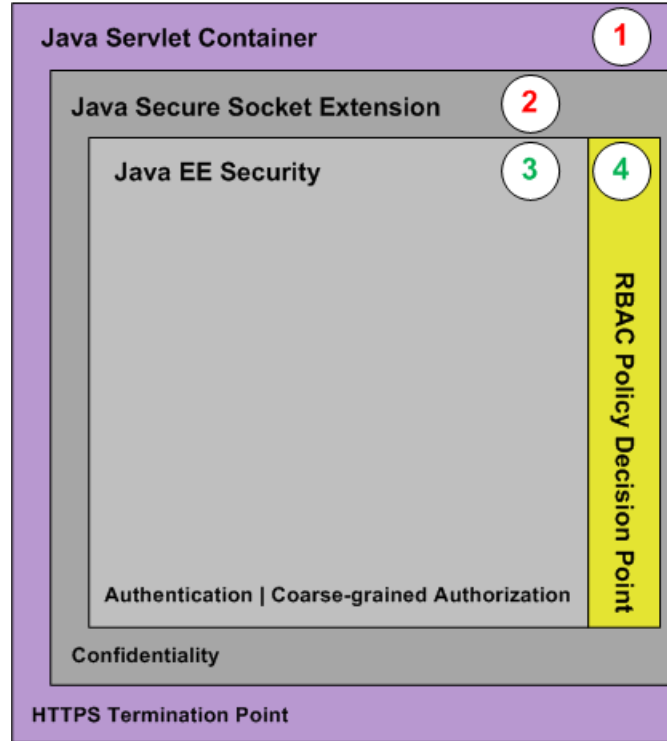
<https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/resources/META-INF/context.xml>

4. Setup RBAC PDP

Policy Decision Point

- a. Install
- b. Configure
- c. Use

LDAPv3 server



Legend:

- HTTP (purple square)
- LDAPv3 (yellow square)
- Confidentiality (red circle)
- Authorization (green circle)

1. HTTPS server
2. HTTPS private key
3. Java EE AuthN & AuthZ
4. RBAC Policy Decision Point

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>

Configure Fortress RBAC PDP

Add Fortress Dependency to web app's [pom.xml](#):

```
<dependency>  
  <groupId>org.apache.directory.fortress </groupId>  
  <artifactId>fortress-web </artifactId>  
  <version>1.0</version>  
  <classifier>classes</classifier>  
</dependency>
```

Configure Fortress RBAC PDP

Enable Spring's context file via web app's [web.xml](#) file:

```
<context-param>  
  <param-name>contextConfigLocation</param-name>  
  <param-value>  
    classpath:applicationContext.xml  
  </param-value>  
</context-param>
```

Configure Fortress RBAC PDP

Enable Fortress RBAC Spring Beans in [applicationContext.xml](#):

```
<bean id= "accessMgr"  
  class= "org.apache.directory.fortress.core.AccessMgrFactory"  
  scope="prototype"  
  factory-method="createInstance">  
  <constructor-arg value="HOME"/>  
</bean>
```

Use ANSI RBAC INCITS 359 Specification

RBAC0:

- Users, Roles, Perms, Sessions

RBAC1:

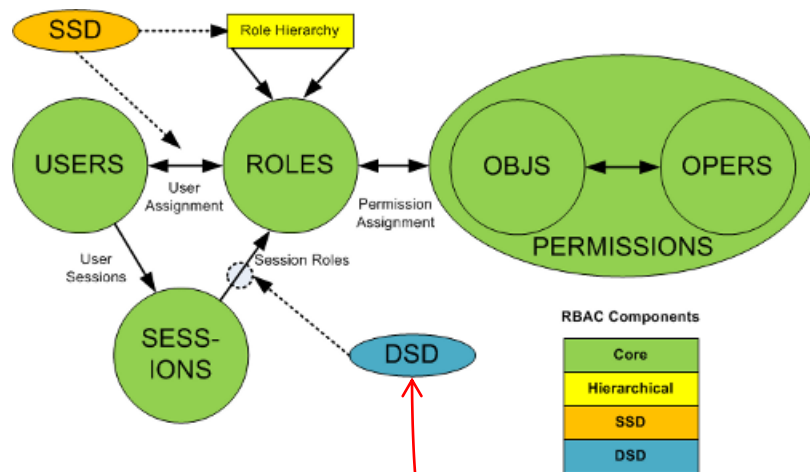
- Hierarchical Roles

RBAC2:

- Static Separation of Duties

RBAC3:

- Dynamic Separation of Duties



Today we demo this

Use RBAC Object Model

Six basic elements:

1. **User** – human or machine entity
2. **Role** – a job function within an organization
3. **Object** – maps to system resources
4. **Operation** – executable image of program
5. **Permission** – approval to perform an Operation on one or more Objects
6. **Session** – contains set of activated roles for User

Use RBAC Functional Model

APIs form three standard interfaces:

1. Admin ← Add, Update, Delete
2. Review ← Read, Search
3. System ← Access Control

*Management and
config processes*

*Demo runtime
processes*

Use RBAC Functional Model

System Manager APIs:

<https://directory.apache.org/fortress/gen-docs/latest/apidocs/org/apache/directory/fortress/core/rbac/AccessMgrImpl.html>

1. createSession – authenticate, activate roles
2. checkAccess – permission check
3. sessionPermissions – all perms active for user
4. sessionRoles – return all roles active
5. addActiveRole – add new role to session
6. dropActiveRole – remove role from session

Identity Propagation JavaEE -> Fortress

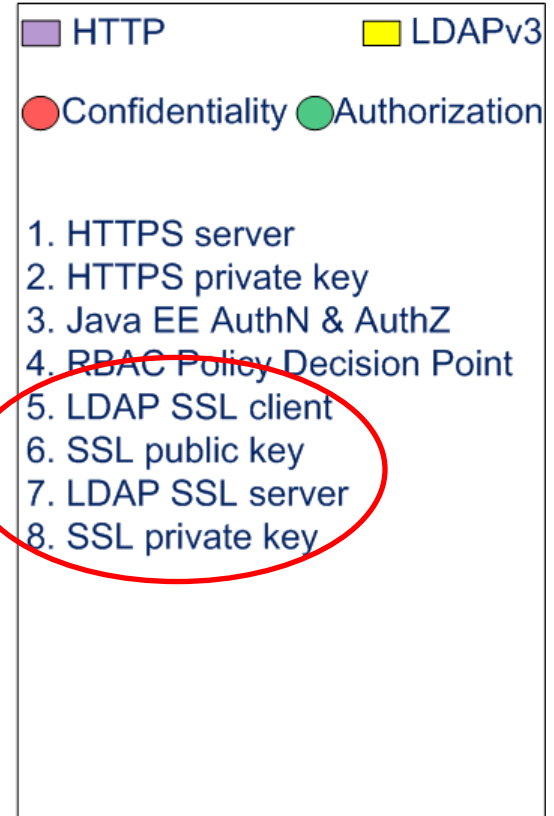
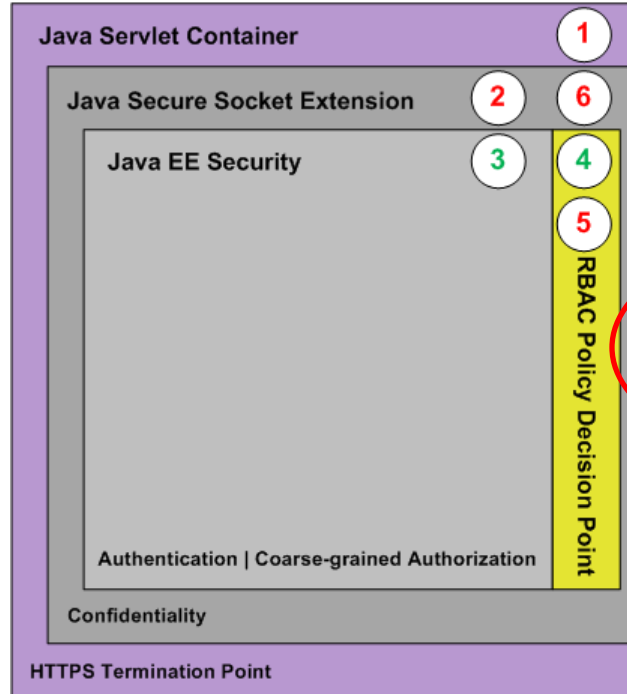
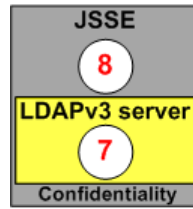
1. Java EE (Tomcat Realm) creates RBAC session based on creds collected from the user, serializes and stores inside the principal object.
2. Container hands reference to serialized principal to app caller:
`servletRequest.getUserPrincipal().toString();`
3. Web app deserializes `principal.toString()` into RBAC session :
`j2eePolicyMgr.deserialize(szPrincipal)`
4. Web app pushes RBAC session into HTTP session.

5 – 8

Enable

LDAP

SSL



Enable LDAP SSL Client

1. Import public key to java truststore:

<http://symas.com/javadocs/apache-fortress-demo/doc-files/keys.html>

2. Add to [fortress.properties](#) of Apache Fortress Demo App:

```
host=ldap-server-domain-name.com  
port=10636  
enable.ldap.ssl=true  
trust.store=/path/mytruststore  
trust.store.password=changeit
```

Enable LDAP SSL Server

1. Import keystore with Apache Directory Studio

2. Restart ApacheDS Server

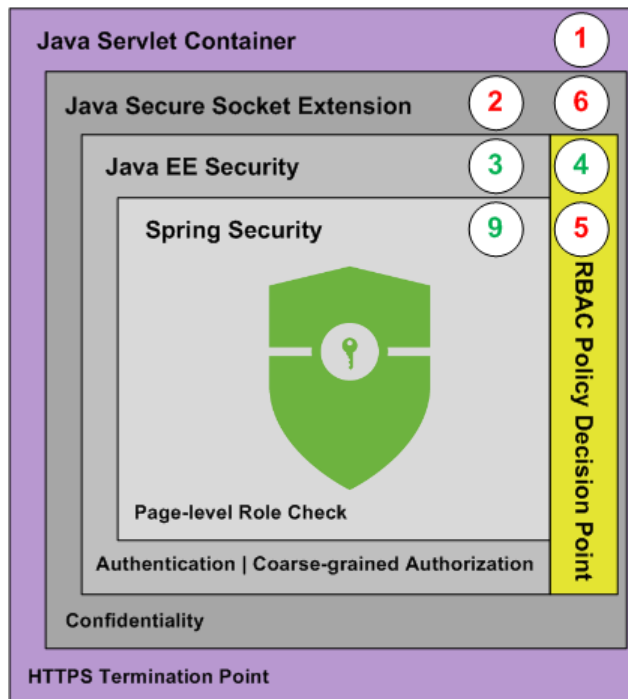
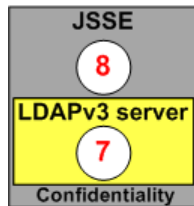
The screenshot shows the Apache Directory Studio configuration window for LDAP/LDAPS Servers. The 'SSL/Start TLS Keystore' section is highlighted with a red circle. The configuration includes the following details:

- LDAP/LDAPS Servers:**
 - Enable LDAP Server (Port: 10389)
 - Enable LDAPS Server (Port: 10636)
- Limits:**
 - Max Time Limit (ms): 15000
 - Max Size Limit (entries): 1000
- SSL/Start TLS Keystore:**
 - Keystore:
 - Password:
 - Show password
- Supported Authentication Mechanisms:**
 - Simple
 - CRAM-MD5
 - NTLM
 - GSS-API
 - DIGEST-MD5
 - GSS-SPNEGO
- SASL Settings:**
 - SASL Host:
 - Default: ldap.example.com
 - SASL Principal:
 - Default: ldap/ldap.example.com@EXAMPLE.COM
 - Search Base Dn:
 - Default: ou=users,dc=example,dc=com

The bottom of the window shows a log entry: `#! CONNECTION ldap://localhost:10389`

9. Enable Spring Security

- a. Enable AuthZ
- b. Role mapping



Legend:

- HTTP (purple square)
- LDAPv3 (yellow square)
- Confidentiality (red circle)
- Authorization (green circle)

1. HTTPS server
2. HTTPS private key
3. Java EE AuthN & AuthZ
4. RBAC Policy Decision Point
5. LDAP SSL client
6. SSL public key
7. LDAP SSL server
8. SSL private key
9. Spring AuthZ

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

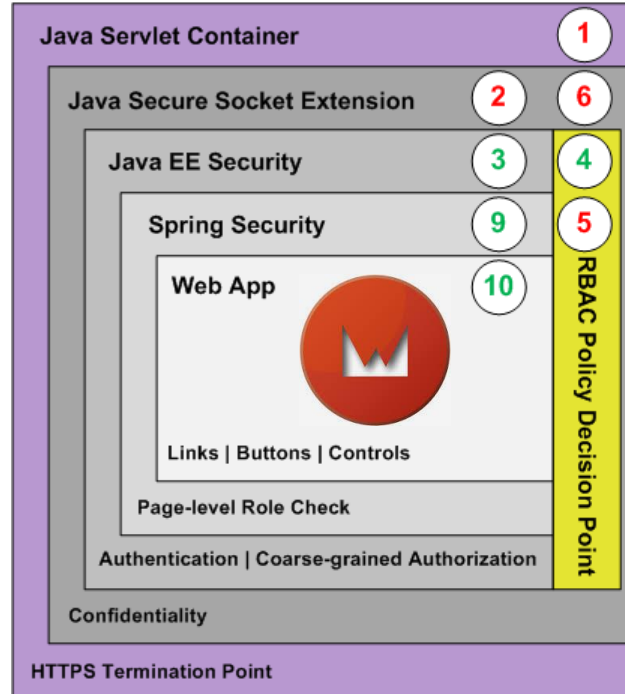
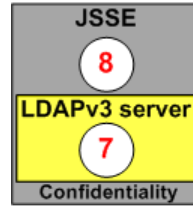
Complete list of eligible roles found in app's [web.xml](#):

```
<!-- 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



Legend: ■ HTTP ■ LDAPv3

● Confidentiality ● Authorization

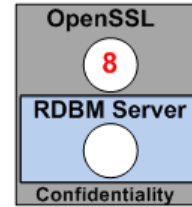
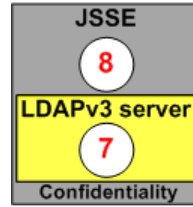
1. HTTPS server
2. HTTPS private key
3. Java EE AuthN & AuthZ
4. RBAC Policy Decision Point
5. LDAP SSL client
6. SSL public key
7. LDAP SSL server
8. SSL private key
9. Spring AuthZ
10. Web App AuthZ

Add Web Framework Security

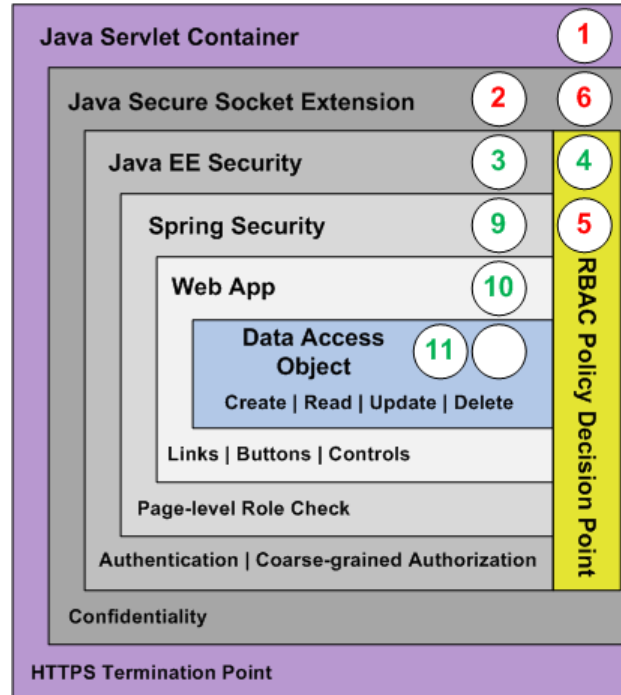
```
add(  
    new SecureIndicatingAjaxButton( "Page1", "Add" )  
    @Override  
    protected void onSubmit( ... )  
    {  
        if( checkAccess( customerNumber )  
        {  
            // do something here:  
        }  
        else  
        {  
            target.appendJavaScript( ";alert('Unauthorized');"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

*fine-grained
authorization
(programmatically)*

11. DAO Authorization



- Add fine-grained Checks to:
- Create
 - Read
 - Update
 - Delete



Legend: HTTP (purple), JDBC (blue), LDAPv3 (yellow), Confidentiality (red circle), Authorization (green circle).

1. HTTPS server
2. HTTPS private key
3. Java EE AuthN & AuthZ
4. RBAC Policy Decision Point
5. LDAP SSL client
6. SSL public key
7. LDAP SSL server
8. SSL private key
9. Spring AuthZ
10. Web App AuthZ
11. DAO AuthZ

Add Security Aware DAO components

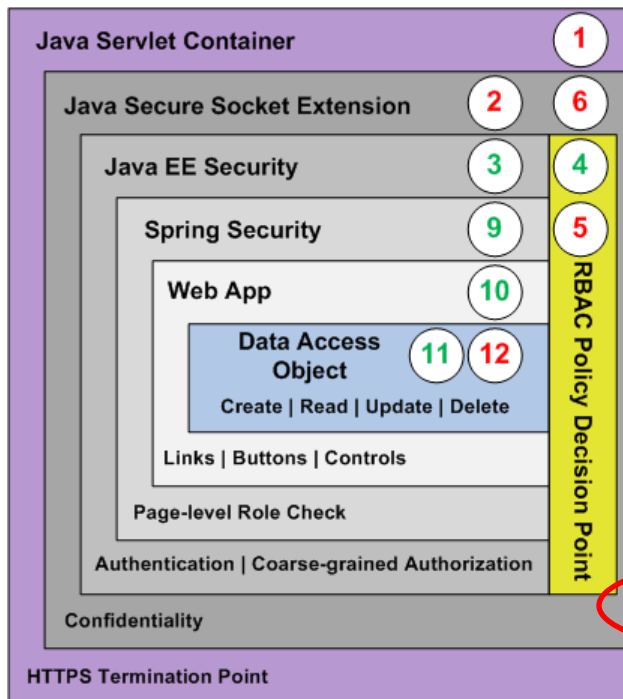
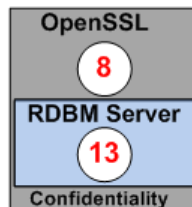
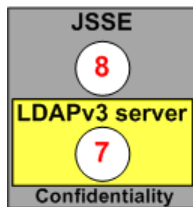
```
public Page1EO updatePage1( Page1EO entity )
{
    ...
    if (checkAccess ("Page1", "Update", entity.getCust ()))
    {
        // Call DAO.update method...
    }
    else
        throw new RuntimeException ("Unauthorized");
    ...
    return entity;
}
```

*fine-grained
authorization
(programmatically)*

12, 13. Enable DB SSL

12. Client
a. public key
b. config

13. Server
a. private key
b. config



Enable MySQL SSL Client

Add to [fortress.properties](#) of Web app:

```
# Sets trust.store params as System.property to  
# be used by JDBC driver:  
trust.store.set.prop=true
```

```
# These are the JDBC configuration params for  
# MyBatis DAO connect to MySQL database  
# example:
```

```
database.driver=com.mysql.jdbc.Driver  
database.url=db-domain-name.com:3306/  
jdbc:mysql://demoDB ?useSSL=true&requireSSL=true
```


Enable MySQL SSL Server

Add to MySQL my.cnf the server's keys:

```
ssl-ca=/path/ca-cert.pem
```

```
ssl-cert=/path/server-cert.pem
```

```
ssl-key=/path/server-key.pem
```

2. Instruct listener to use host name in certificate on server restart:

```
bind-address = db-domain-name.com
```

<http://symas.com/javadocs/apache-fortress-demo/doc-files/mysql.html>

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

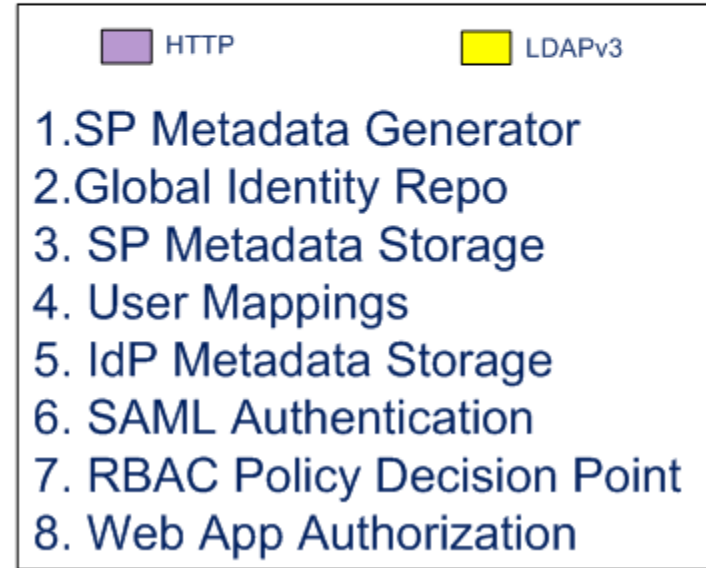
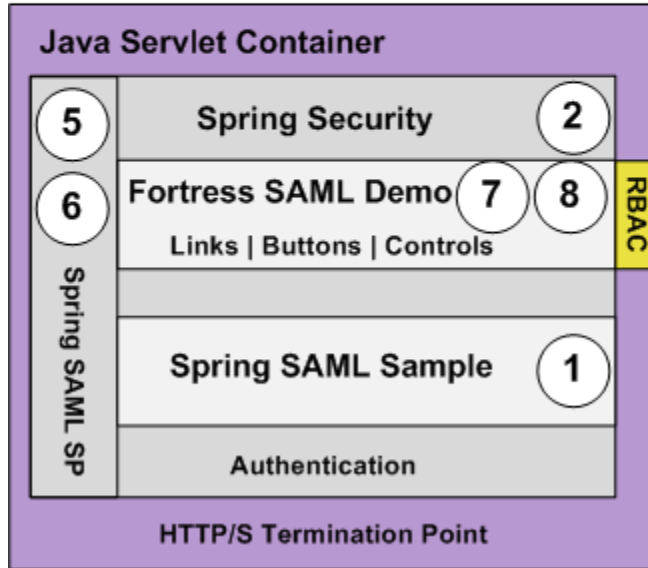
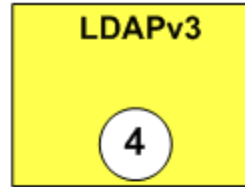
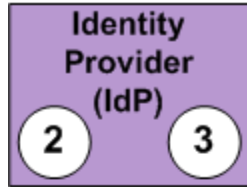
Apache Fortress Demo

- <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

Tutorial #2

Apache
Fortress
SAML
Demo



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

The Five Security Layers with SAML

1. JSSE

~~2. Java EE Security~~ ← Turned off (for now)

3. Spring Security ← Deadbolt is now here

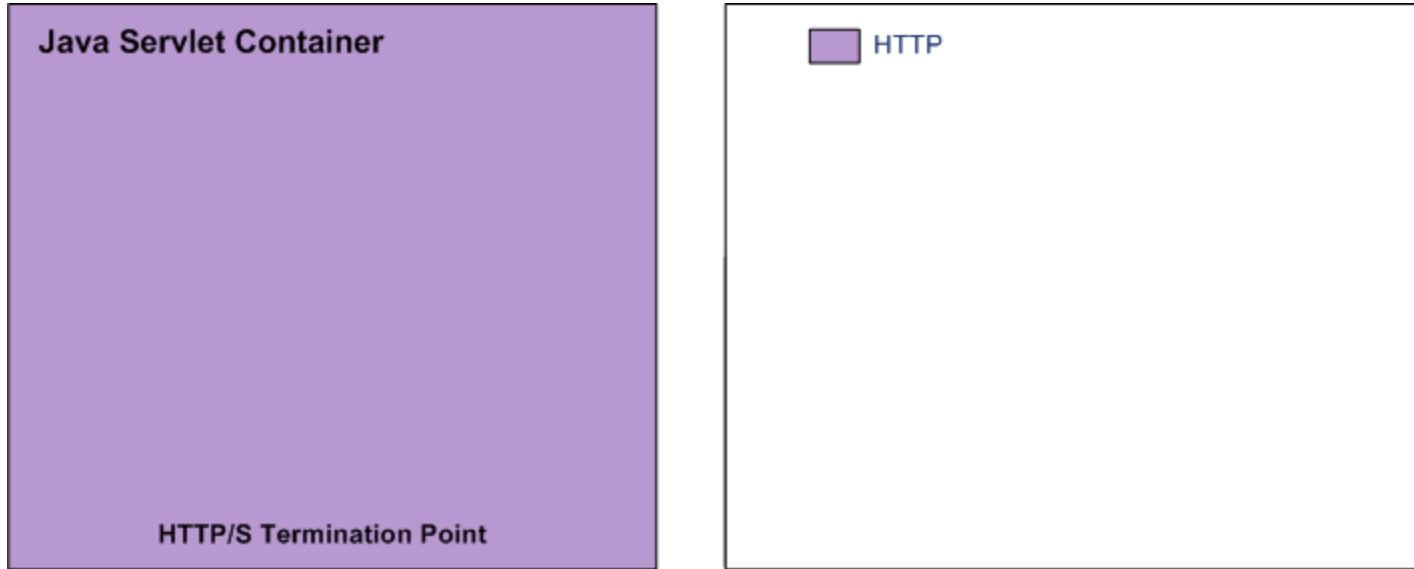
4. Web App Framework
5. Database Functions ← Not much to change

Two Areas of Access Control

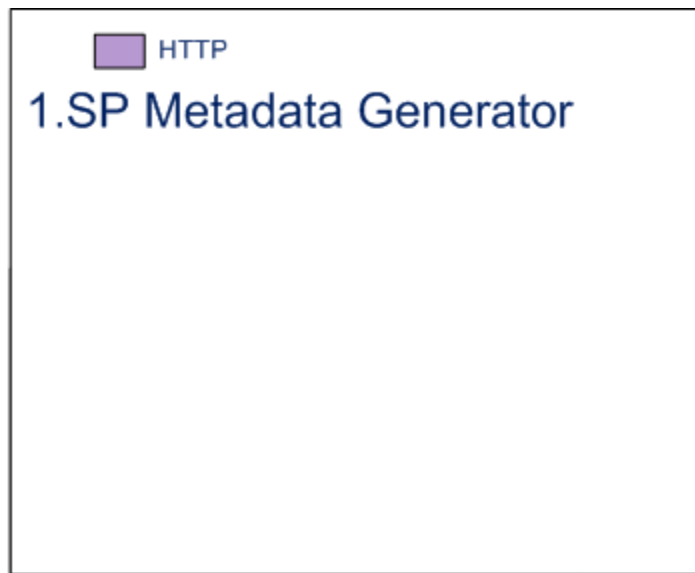
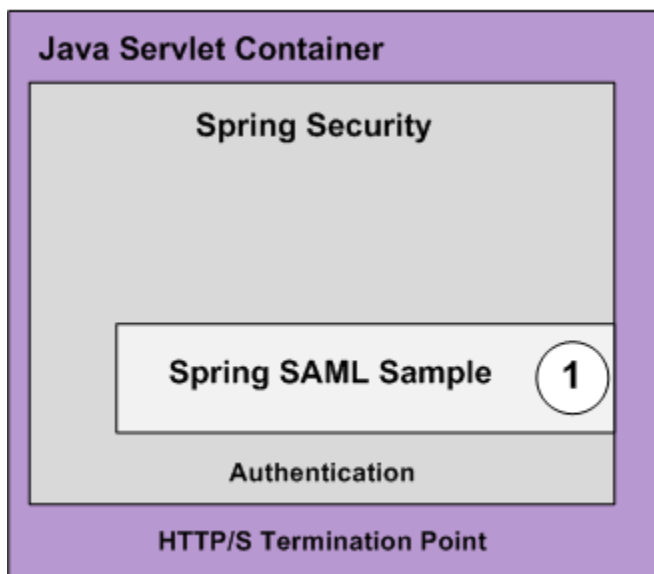
1.Spring SAML Declarative checks

2.RBAC Permission Programmatic checks

Start with Tomcat Servlet Container



1. Deploy the Spring SAML Demo



Get the Spring SAML Demo

Pick one:

- [spring-security-saml](#) - Spring's SAML sample is the first place java developers should look for basic SAML 2.0 programming concepts.
- [shibboleth-sample-java-sp](#) - Unicon's sample is where ones goes to understand how to combine Spring SAML's SP with 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.

Metadata generation

Generates new metadata for service provider. Output can be used to configure your securityContext.xml descriptor.

<< Back

Store for the current session:

No ▾

When set to true the generated metadata will be stored in the local metadata manager. The value will be available only until restart of the application server.

Entity ID:

fortress-saml-demo

Entity ID is a unique identifier for an identity or service provider. Value is included in the generated metadata.

Entity base URL:

https://hostname:443/fortress

Base to generate URLs for this server. For example: https://myServer:443/saml-app. The public address your server will be accessed from should be used here.

TO USE TLS

Spring SAML Metadata Generation Tip



Entity ID:

Spring SAML Sample application

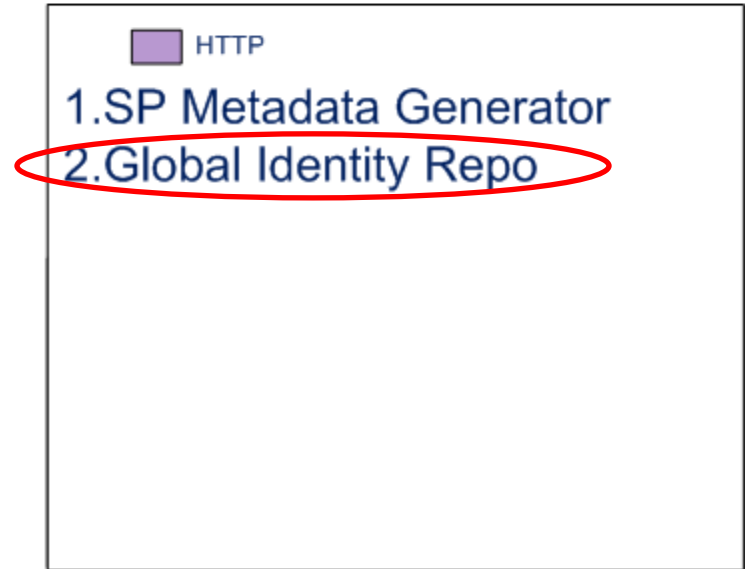
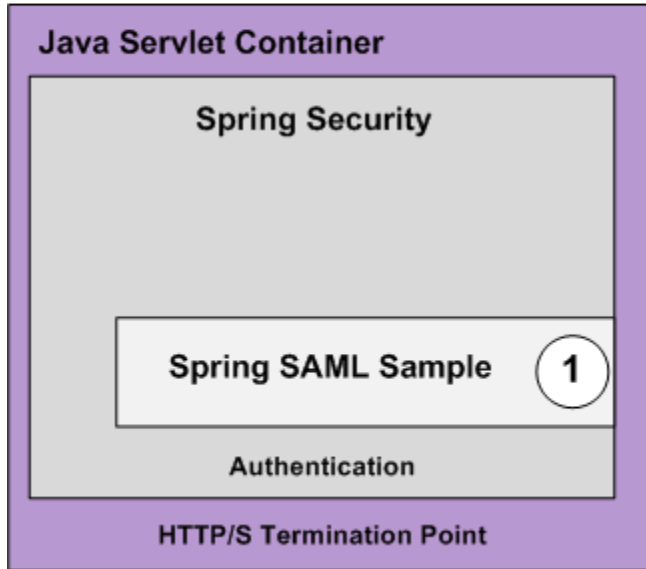
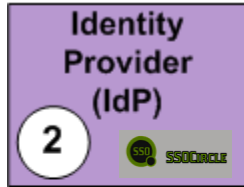
fortress-saml-demo

These
entityId's
must
match

```
<bean id="metadataGeneratorFilter" class="org.springframework...MetadataGeneratorFilter">  
  <constructor-arg>  
    <bean class="org.springframework...MetadataGenerator">  
      <property name="entityId" value="fortress-saml-demo"/>  
    </bean>  
  </constructor-arg>  
</bean>
```

Bind the service provider with the IdP.

2. Setup Global Identity Provider



Setup SSOCircle SAMLv2.0 IdP

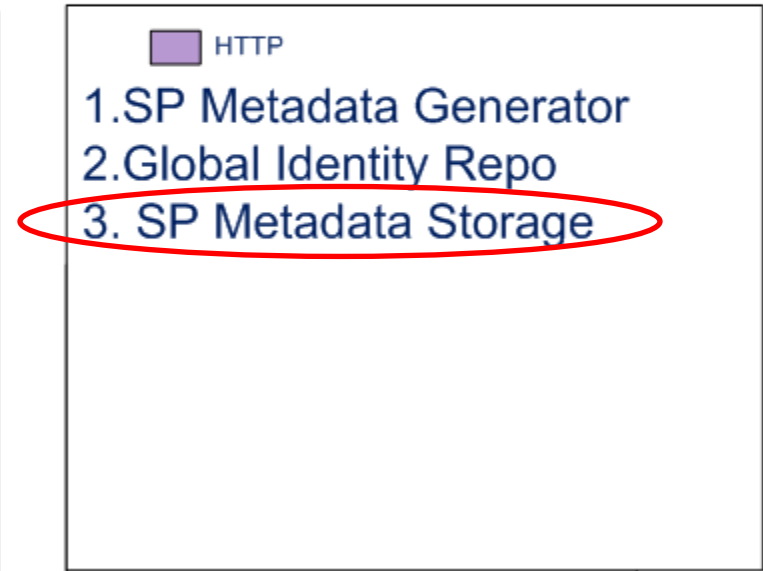
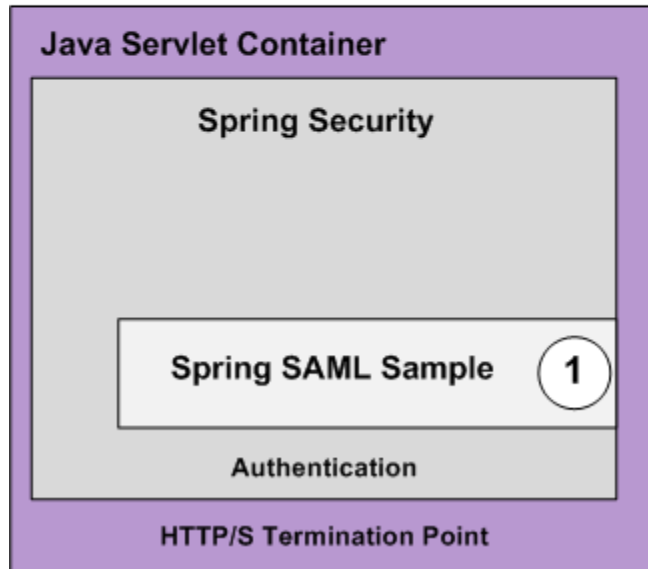
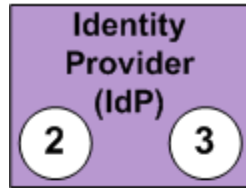
Creating your Identity with SSOCircle (from their website)

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

SP Meta Data

https://idp.ssocircle.com/sso/hos/SPMetaInter.jsp

SSOCIRCLE

EGNYTE

Combat Shadow IT
Share Files Easily & Securely

Logout

My Profile

My SAML Federations

My OpenID Trust

My Certificate Status

My Certificate Enrollment

My Certificate Enrollment PKCS#10

My Certificate Revocation

Manage Metadata

My Audit

My Subscriptions

Service Provider Metadata import

User ID: foofighters

Submit

Enter the FQDN of the ServiceProvider ex.: sp.cohos.de

sp2.symas.com

Attributes send in assertion (optional)

FirstName

LastName

EmailAddress

Insert your metadata information

#

Import SP Metadata Tip

Spring SAML app Metadata Generation page:



Spring SAML Sample application

Entity base URL:

`http://sp2.symas.com:8080/`

The FQDN matches base url from SP metadata gen step

SSOCircle Service Provider Metadata Import page:

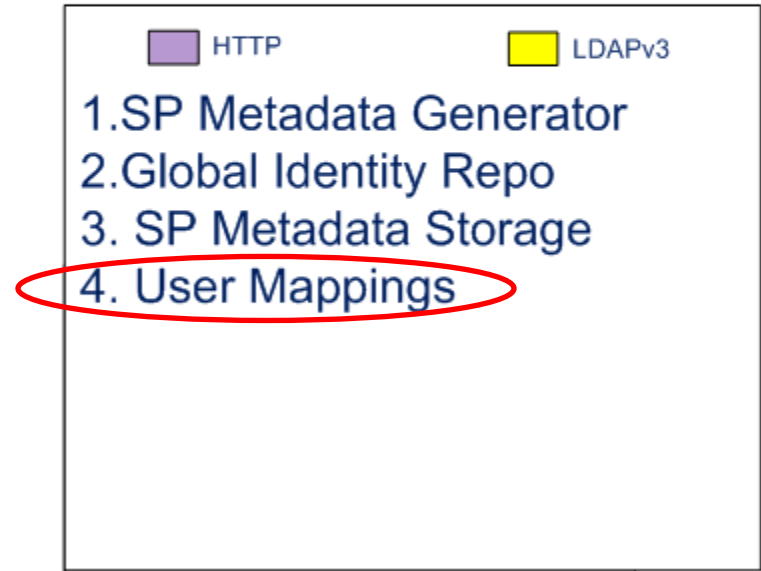
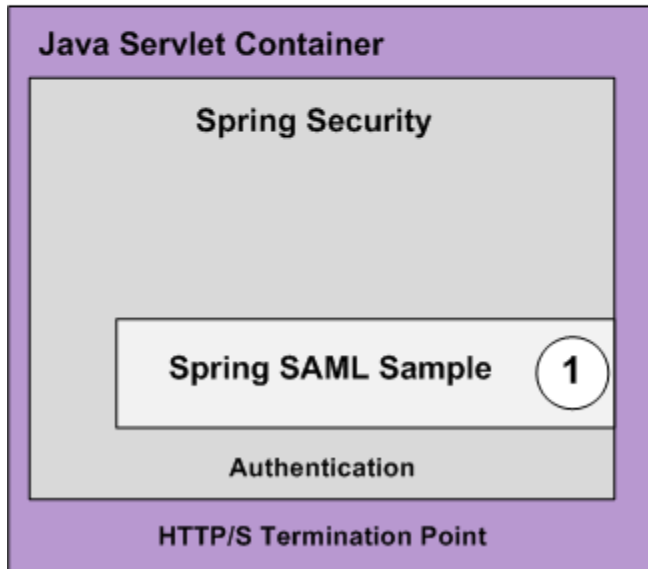
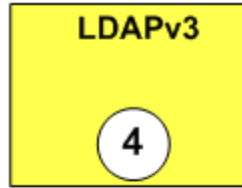
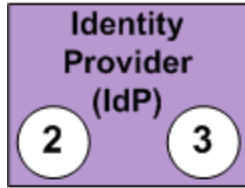
Enter the FQDN of the ServiceProvider ex.: sp.cohos.de



`sp2.symas.com`

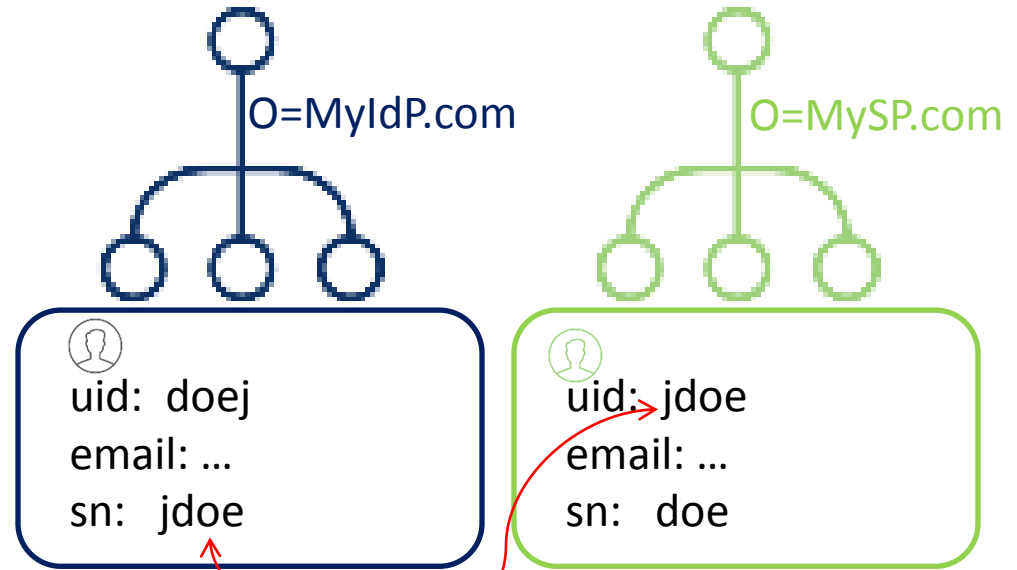


4. IdP and SP User Account Mapping



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*

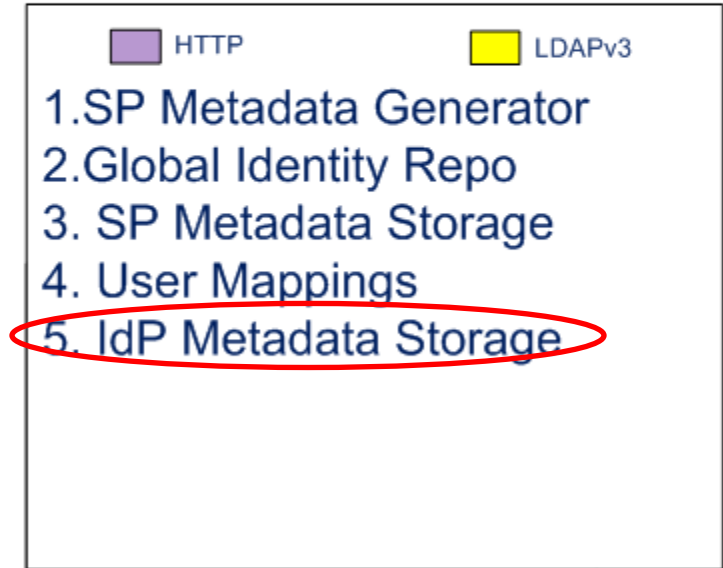
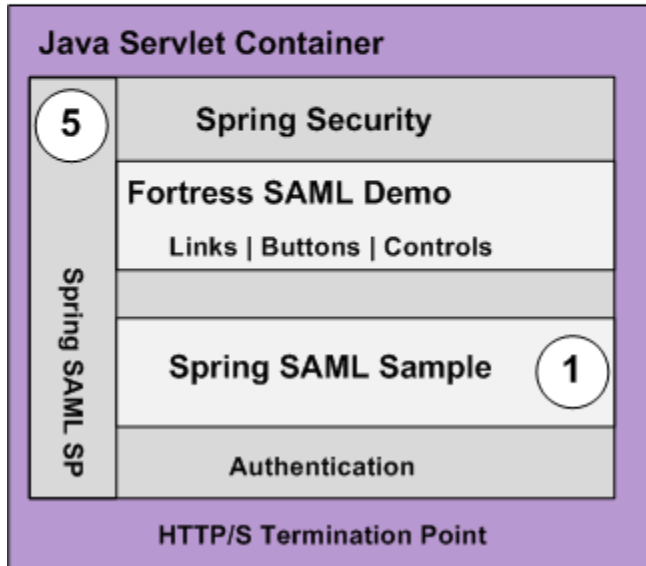
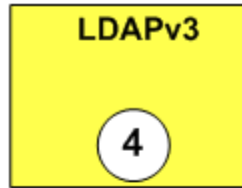
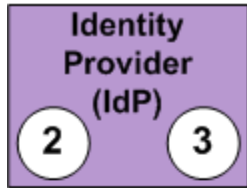
SAML Attribute Statement

```
<?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

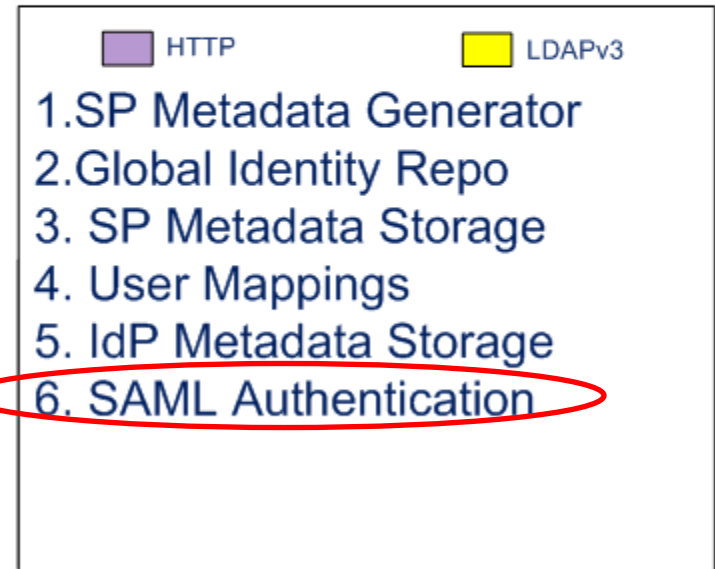
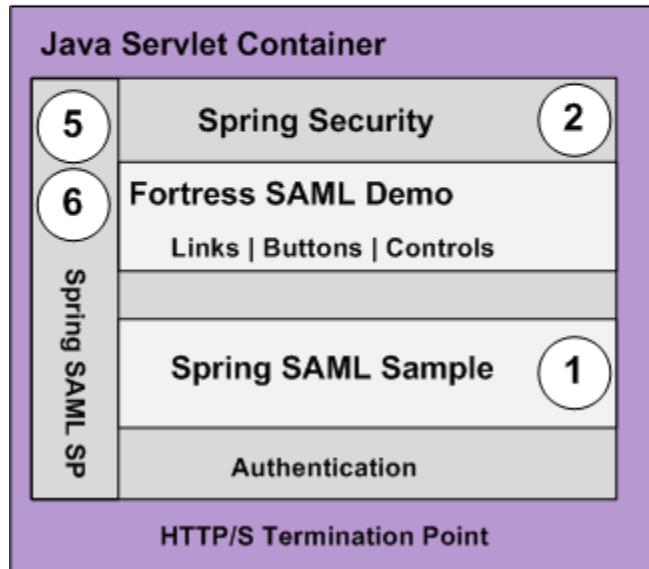
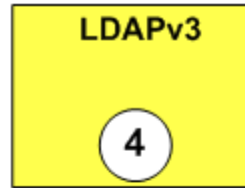
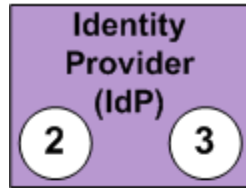


Point SP to SAML IdP

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>
```

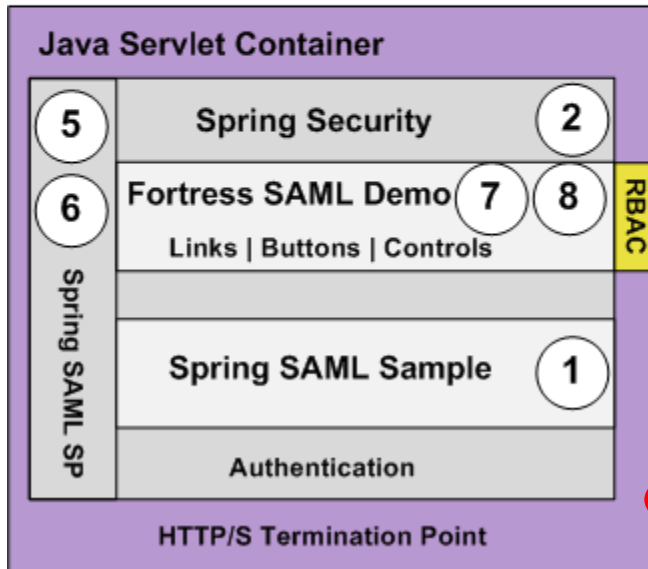
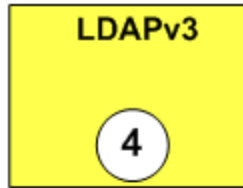
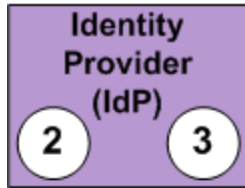
* backlog item

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 parses the attributes contained within principal :

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

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

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

4. Web app pushes RBAC session into HTTP session.

isTrusted

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

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

https://idp.ssocircle.com/sso/hos/SelfCare.jsp

SSO SSOCIRCLE

Logout

My Profile

My SAML Federations

My OpenID Trust

My Certificate Status

My Certificate Enrollment

My Certificate Enrollment PKCS#10

My Certificate Revocation

Manage Metadata

My Audit

My Subscriptions

User Profile

Attribute	Value
User ID	foofighters
Google Apps Email	No longer available
OpenID identification	http://foofighters.ssocircle.com
Client Certificate	Not Enrolled
Given name	foofighters1
Surname	sam1
Email	someone@domain.org
ePass OTP token number	"not assigned"
Yubikey ID	not assigned
Yubikey PIN
Swekey ID detect	not assigned
Swekey PIN
MSISDN identification	not active
Old password (required)
Password (length > 8)	
Retype Password	

[Delete](#) your MSISDN linking.

[Delete](#) your ePass OTP linking.

[Delete](#) your Swekey linking.

[Delete](#) your Yubikey linking.

View/change your OpenID [public profile settings](#).

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

Closing Thoughts

1. Use TLS across all remote connections
 - *Confidentiality and Integrity*
2. Apply security controls across many layers
 - *Defense in Depth*
3. Never allow users more than they need to do their jobs
 - *Principle of Least Privilege*

Contact Info

Twitter: [@shawnmckinney](https://twitter.com/shawnmckinney)

Website: <http://symas.com>

Email: smckinney@apache.org

Blog: <https://iamfortress.net>

Project: <https://directory.apache.org/fortress>