

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

September 18, 2017

Little Rock Java User Group

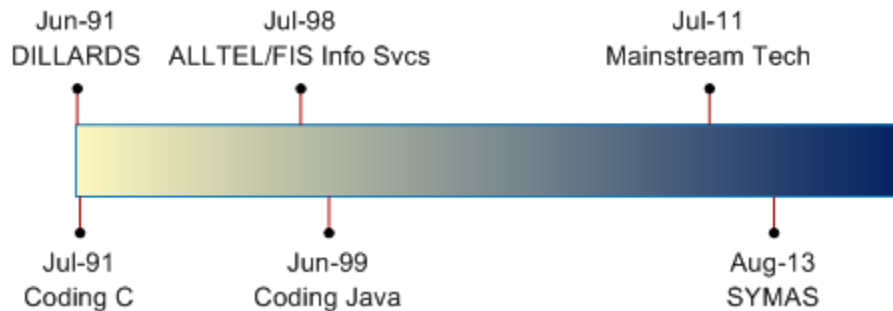





Objective

- Think about how a web app would behave, if we spared no expense for security.

Introductions

Shawn McKinney



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

Agenda

Look at two examples...

1. Apache Fortress Demo + Java EE + Spring Security
 - <https://github.com/shawnmckinney/apache-fortress-demo>
2. Fortress SAML Demo + Spring Security SP
 - <https://github.com/shawnmckinney/fortress-saml-demo>

Recommendation

Listen and try to absorb conceptually. Slides have the details.

<https://iamfortress.files.wordpress.com/2017/09/anatomy-secure-web-app-lrtf-2017-v22.pdf>

The Problem

- Equifax Breach
 - 143 million Americans' personal info, including names, addresses, dates of birth and SSNs compromised.
 - Only a veneer of security in place.

The Exploit

*“The vulnerability was Apache Struts **CVE-2017-5638**”*

<http://www.zdnet.com/article/equifax-confirms-apache-struts-flaw-it-failed-to-patch-was-to-blame-for-data-breach/>

The Exploit

“The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 mishandles file upload, which allows remote attackers to execute arbitrary commands via a #cmd= string in a crafted Content-Type HTTP header, as exploited in the wild in March 2017.”

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638>

How does it work?

- Input data deserialized into an executable object with privilege.

<http://blog.diniscruz.com/2013/12/xstream-remote-code-execution-exploit.html>

Equifax Breach

“After exploiting the vulnerability to gain a foothold, the attackers may have found scores of unprotected data immediately or may have worked over time—between mid-May and the end of July—to gain more and more access to Equifax's systems.”

<https://www.wired.com/story/equifax-breach-no-excuse/>

Equifax Breach

“Generally when you successfully exploit a web-application bug like this you will become the system user who owns the web server process,”

Alex McGeorge, the head of threat intelligence at the security firm Immunity.

<https://www.wired.com/story/equifax-breach-no-excuse/>

The Solution

Ensure all appropriate patches
have been applied and that you
aren't running software with
vulnerabilities or backdoors.



How do we ensure that all of our software is free of vulnerabilities, known or otherwise?

How do we ensure that all of our software is free of vulnerabilities, known or otherwise?

can't be done - practically

So Now What?

“Security best practices dictate that this user have as little privilege as security vulnerabilities in web applications and web servers are so commonly exploited.”

<https://www.wired.com/story/equifax-breach-no-excuse/>

The Solution (Take 2)

Usage of controls like the Java Security Manager to ensure code runs with the least amount of privilege possible.

Employ a Runtime Java Security Policy

```
grant codeBase "file:${catalina.home}/webapps/my-web-app-1/-" {  
    permission java.util.PropertyPermission "net.sf.ehcache.*", "read";  
    permission java.util.PropertyPermission "wicket.*", "read";  
    permission java.lang.RuntimePermission "getenv.*", "*";  
    permission java.lang.RuntimePermission "modifyThread";  
    permission java.net.SocketPermission "localhost", "resolve";  
    permission java.net.SocketPermission "127.0.0.1:32768", "connect,resolve";  
    permission java.lang.reflect.ReflectPermission "suppressAccessChecks";  
    permission java.io.SerializablePermission "enableSubclassImplementation";  
    ...  
};
```

Not a perfect solution

Permission Target Name	What the Permission Allows	Risks of Allowing this Permission
suppressAccessChecks	ability to access fields and invoke methods in a class. Note that this includes not only public, but protected and private fields and methods as well.	This is dangerous in that information (possibly confidential) and methods normally unavailable would be accessible to malicious code.

Related JavaOne Sessions

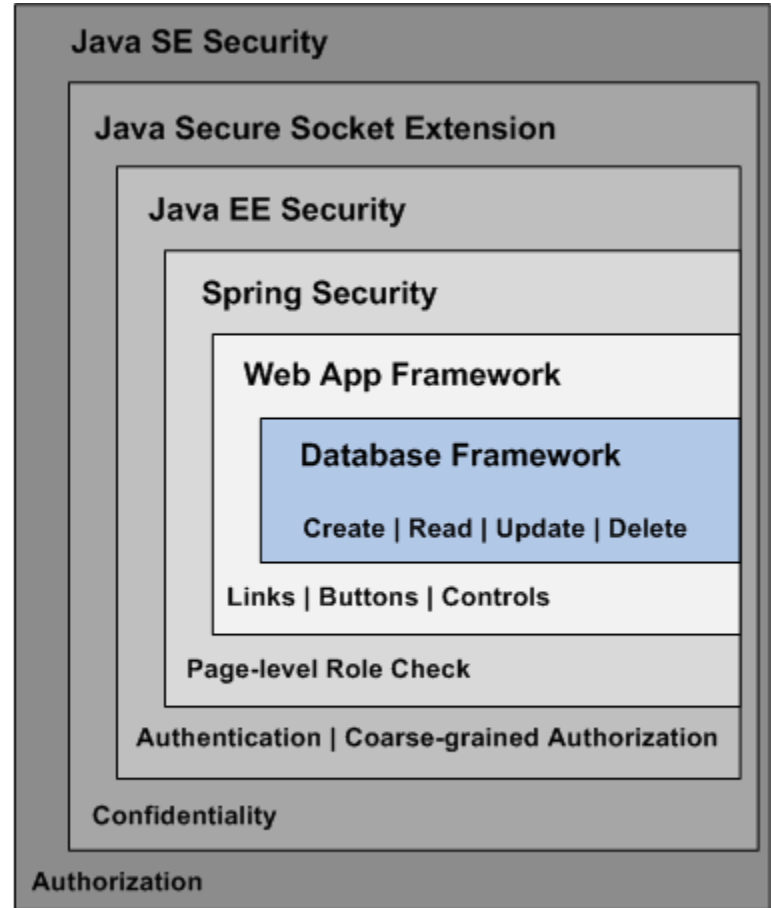
- CON4604 - Making the JDK More Secure
 - Mon, Oct 02, 4:30p
 - Moscone West - Room 2014
- CON4683 - Java Secure Coding Guidelines
 - Tue, Oct 03, 11:00a
 - Moscone West - Room 2005
- BOF5944 - Securing Your Java Application: Is SecurityManager the Answer?
 - Tue, Oct 03, 7:45p
 - Moscone West - Room 2016

The Solution Continues

- Security applied across multiple layers
- Each layer has a specific purpose

Java Web Security Layers

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



Rationale of Each

1. Java SE Security ----- *principle of least privilege*
2. JSSE ----- *private conversations*
3. Java EE Security ----- *deadbolt on front door*
4. Spring Security ----- *locks on room doors*
5. Web App Framework - *locks on equipment in rooms*
6. Database Functions ---- *content filtering*

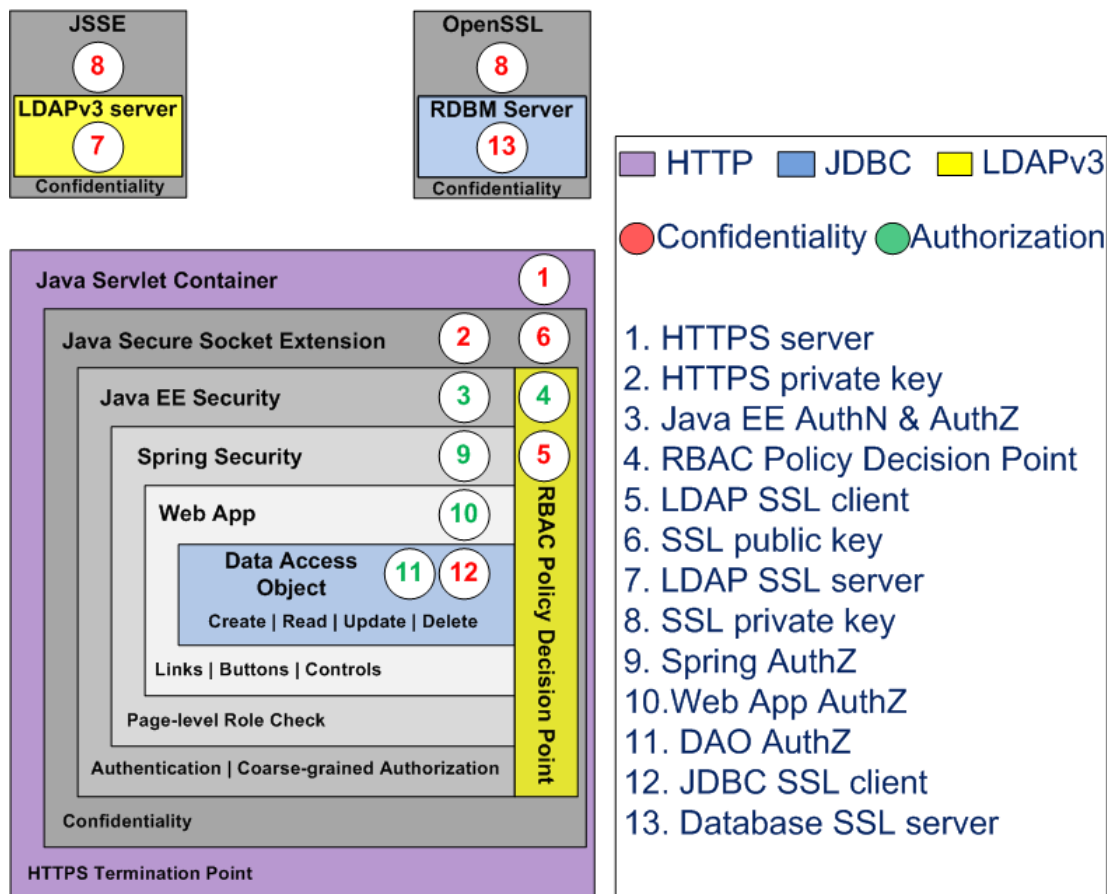
Two Areas of Access Control

1. Java and Spring Role Declarative checks

2. RBAC Permission Programmatic checks

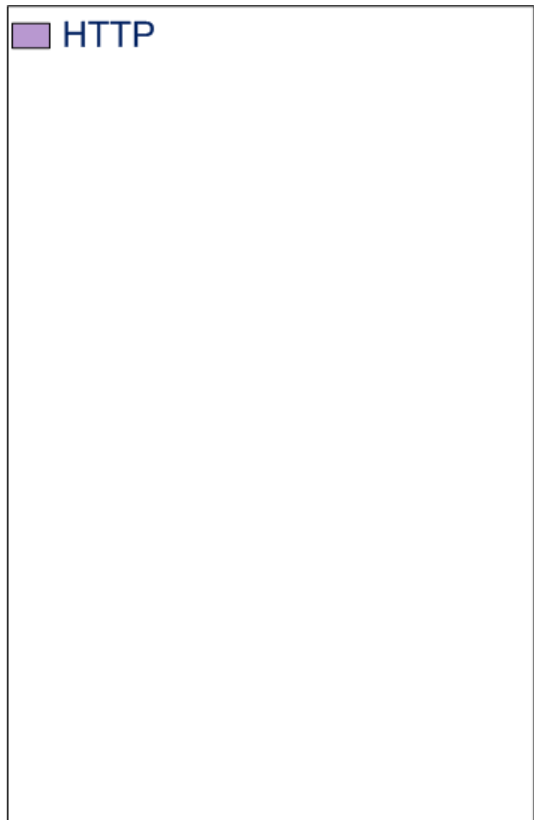
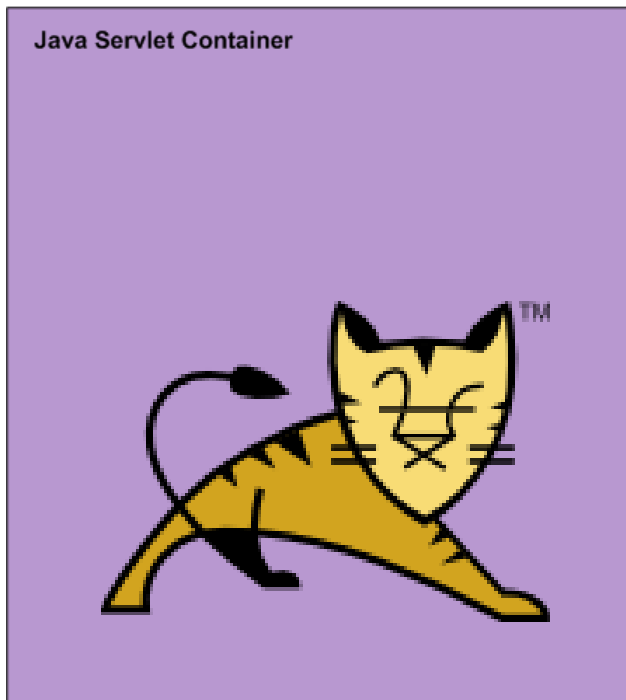
Example #1

Apache Fortress Demo



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

Start with Tomcat Servlet Container

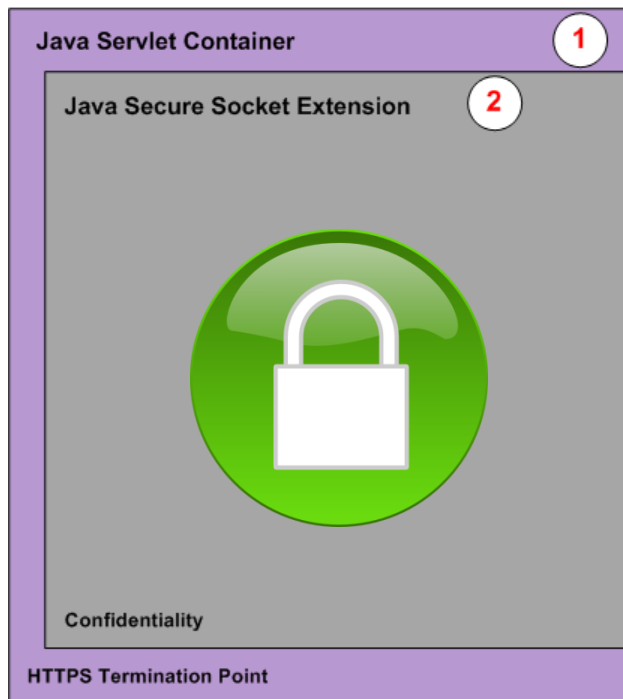


1 & 2. Enable HTTPS

ssssh!!!

1. Update the
Server.xml

2. Add private key



HTTP

Confidentiality

1. HTTPS server
2. HTTPS private key

Enable Tomcat TLS

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

<http://shawnmckinney.github.io/apache-fortress-demo/apidocs/doc-files/keys.html>

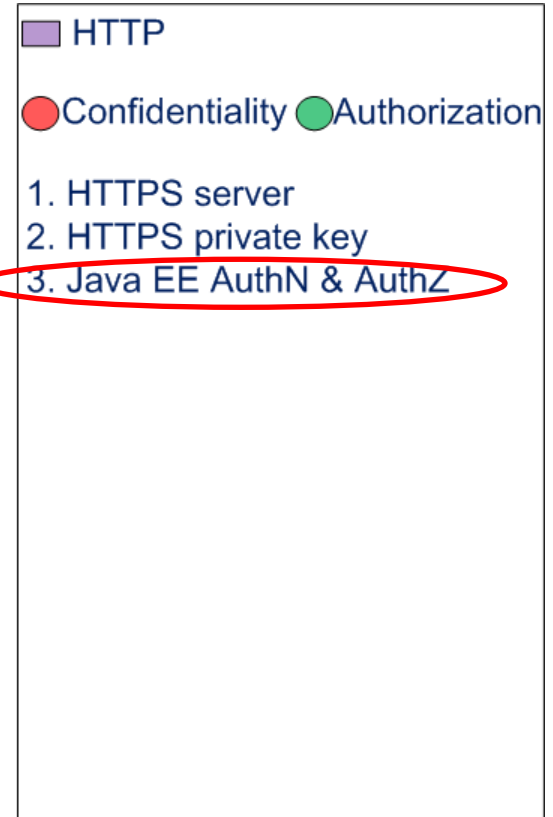
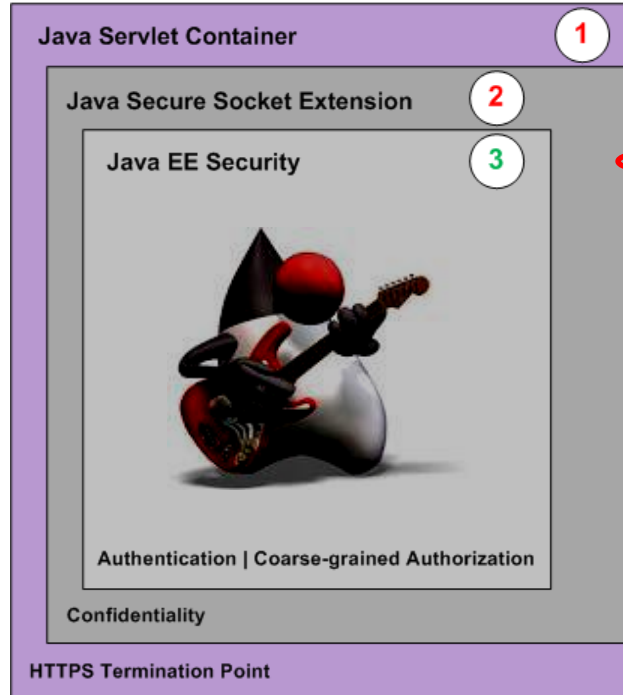
2. Add the following to server.xml:

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

<http://shawnmckinney.github.io/apache-fortress-demo/apidocs/doc-files/apache-tomcat-ssl.html>

3. Enable Java EE Security *the deadbolt*

- a. Update web.xml
- b. Drop the proxy jar
- c. Add context.xml
- d. Add fortress to pom.xml



Current Specs for Java EE Security

1. JSR-196 – JASPIC - AuthN
2. JSR-115 – JAAC - AuthZ
3. JSR-375 – JavaEE Security API

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>
```

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.

```
<form-login-page>/login/login.html</form-login-page>
```

<https://github.com/shawnmckinney/apache-fortress-demo/blob/master/src/main/webapp/WEB-INF/web.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 uses dependencies within the web app via URLClassLoader.

```
[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 [context.xml](#) to META-INF folder:

```
<Context reloadable="true">  
  <Realm className=    Apache Fortress Tomcat Realm  
    "org.apache.directory.fortress.realm.tomcat.Tc7AccessMgrProxy"  
    defaultRoles="ROLE_DEMO2 SUPER USER, DEMO2 ALL PAGES,  
                  ROLE_PAGE1, ROLE_PAGE2, ROLE_PAGE3"  
    The set of role candidates eligible to be activated into a session.  
  />  
</Context>
```

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

Enable RBAC Policy Decision Point

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

```
<dependency>
```

```
  <groupId>org.apache.directory.fortress</groupId>
```

```
  <artifactId>
```

```
    fortress-realm-impl
```

```
  </artifactId>
```

```
  <version>2.0.0</version>
```

```
</dependency>
```

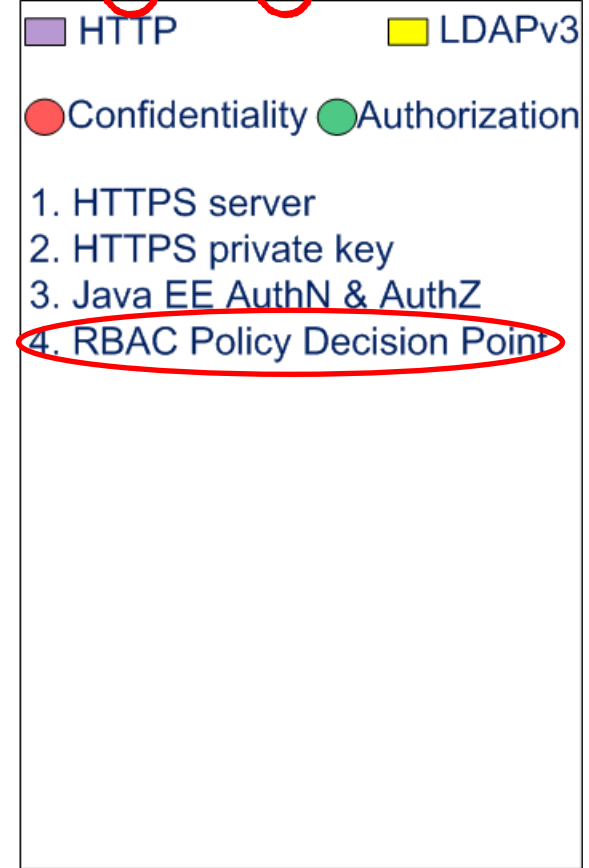
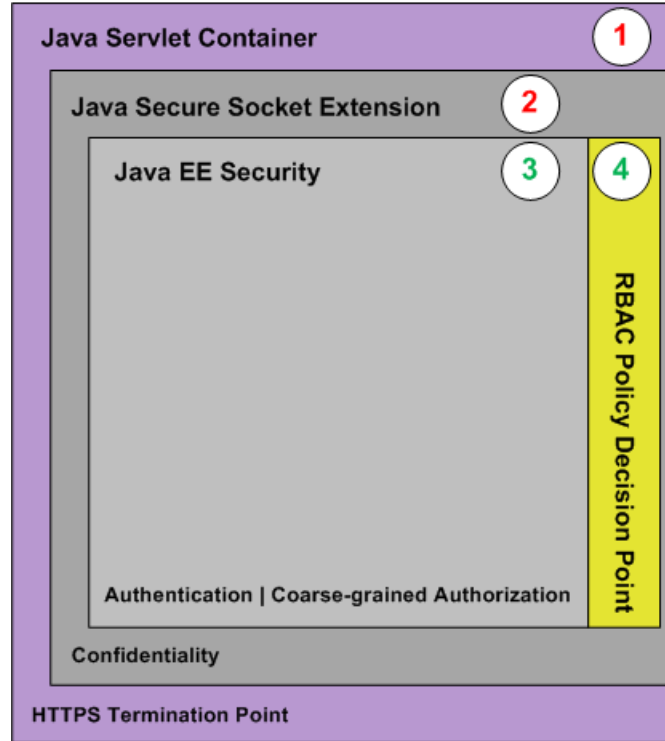

4. Setup RBAC PDP

Policy Decision Point

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

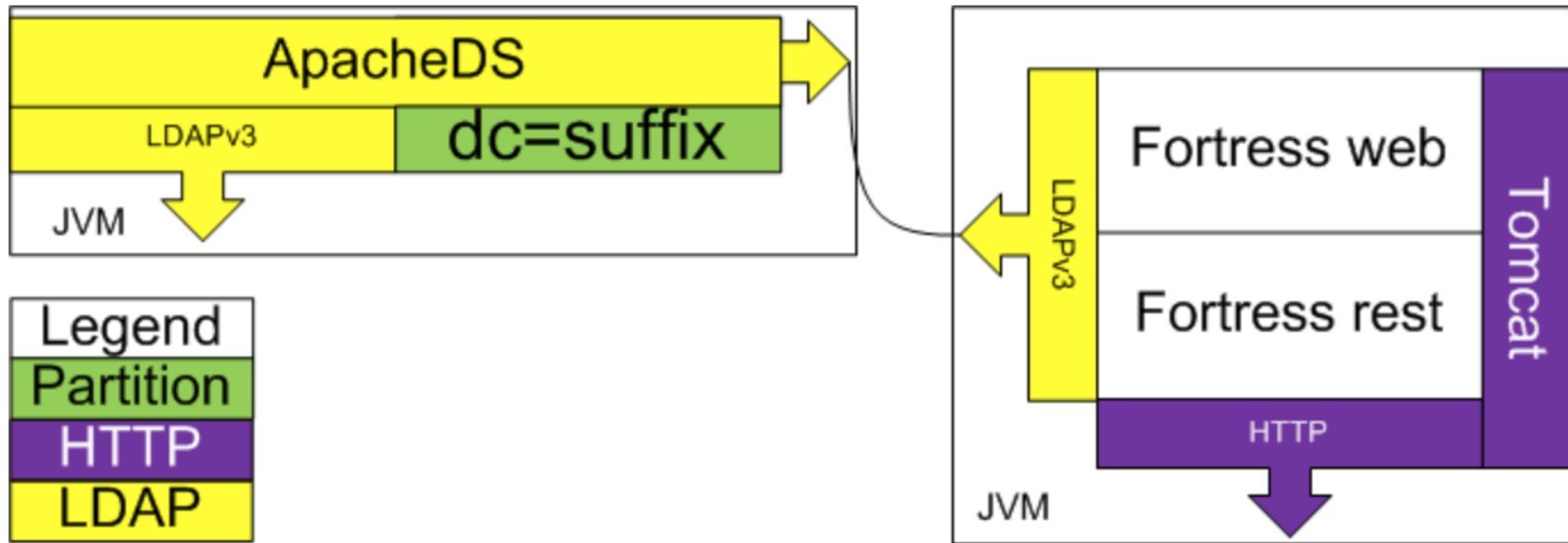
the security system

LDAPv3 server



ApacheDS & Fortress QUICKSTART

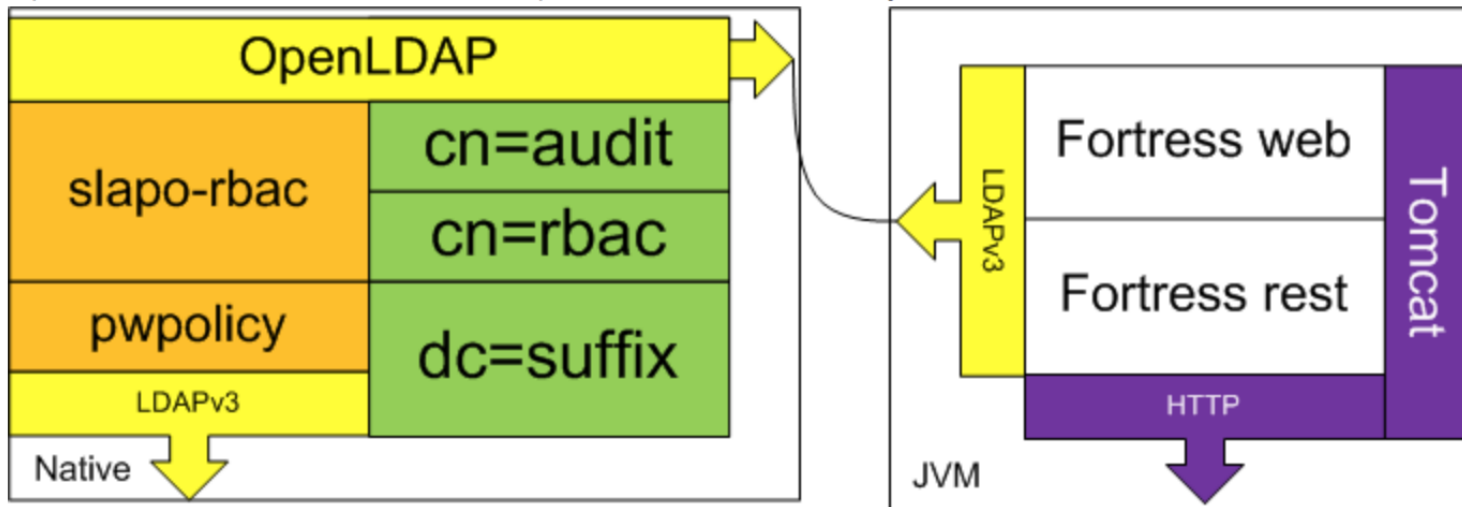
Apache Fortress 2.0.0-RC1-SNAPSHOT and ApacheDS Quickstart System Architecture



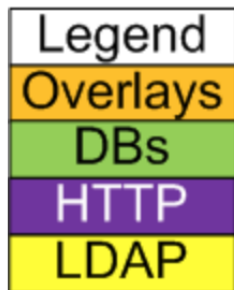
<https://github.com/apache/directory-fortress-core/blob/master/README-QUICKSTART-APACHEDS.md>

Or OpenLDAP & Fortress QUICKSTART

Apache Fortress 2.0.0-RC2 and OpenLDAP Quickstart System Architecture



<https://github.com/apache/directory-fortress-core/blob/master/README-QUICKSTART-SLAPD.md>



Use ANSI RBAC INCITS 359 Specification

RBAC0:

- Users, Roles, Perms, Sessions

RBAC1:

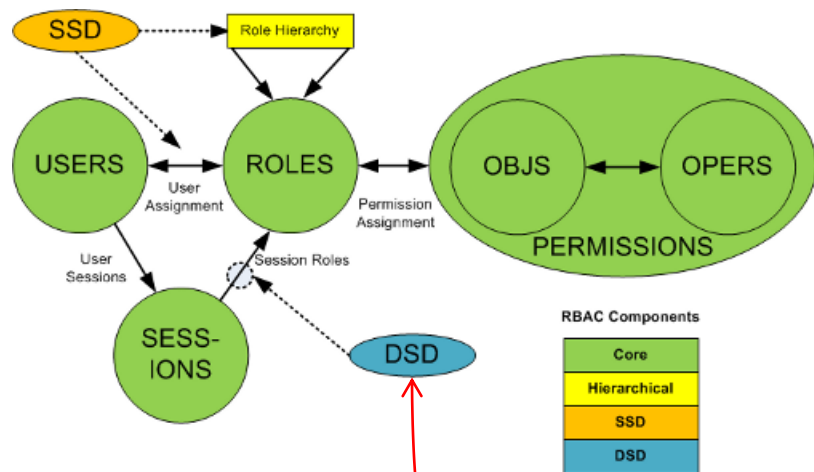
- Hierarchical Roles

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:

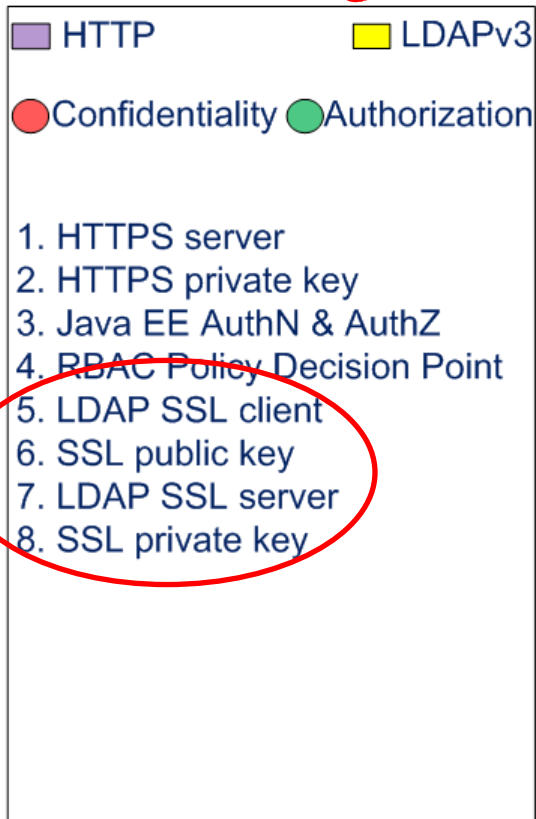
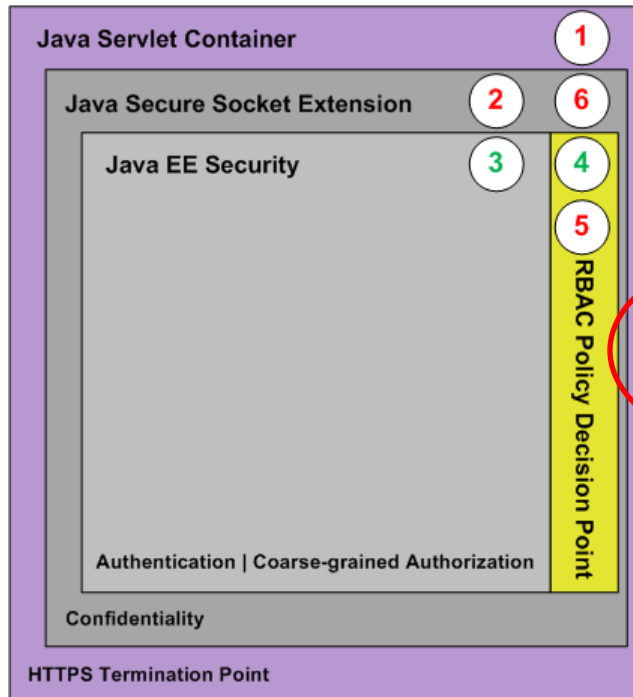
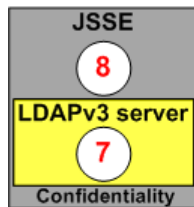
<http://directory.apache.org/fortress/gen-docs/latest/apidocs/org/apache/directory/fortress/core/impl/AccessMgrImpl.html>

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

5 – 8

Enable LDAP SSL

confidentiality



Enable LDAP SSL Client

1. Import public key to java truststore:

<http://shawnmckinney.github.io/apache-fortress-demo/apidocs/doc-files/keys.html>

2. Add to [fortress.properties](#)

host=ldap-server-domain-name.com

port=636

enable.ldap.ssl=true

trust.store=mytruststore

trust.store.password=changeit

trust.store.onclasspath=true

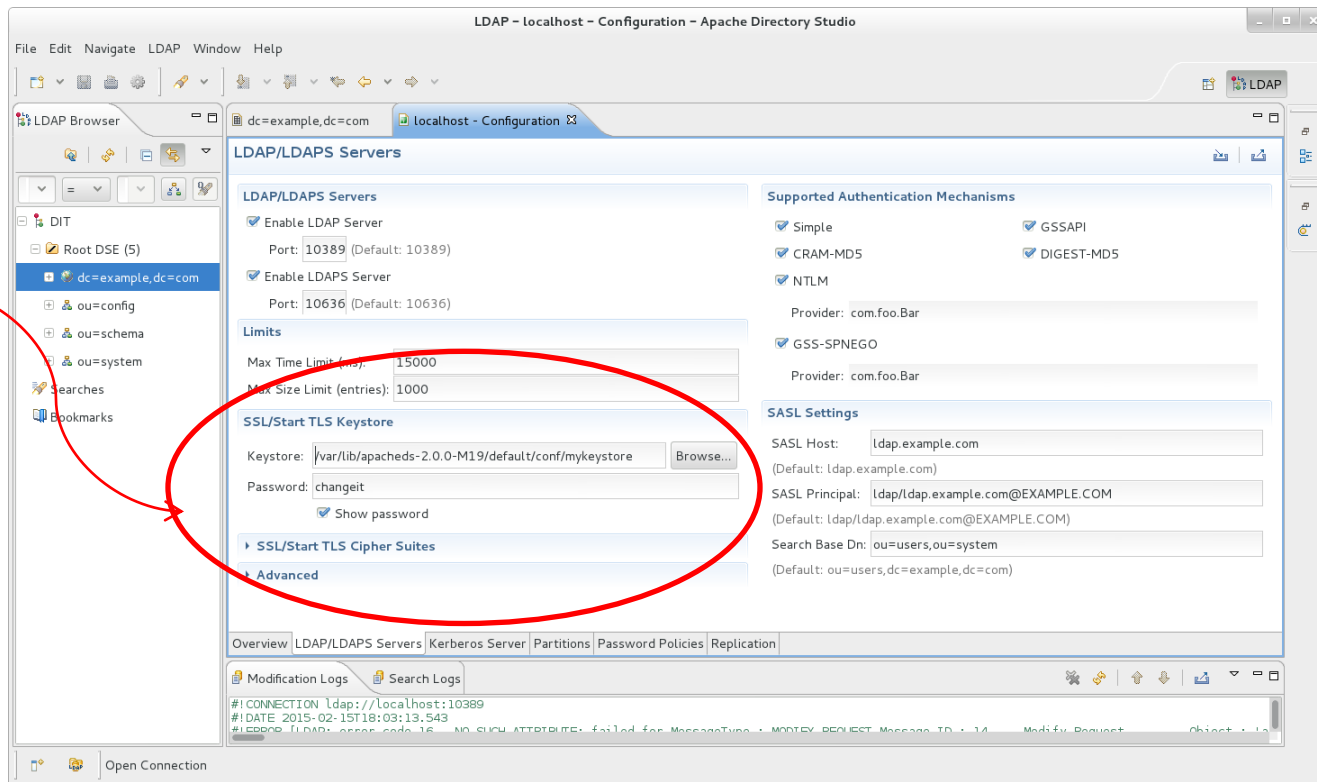
*common name
in server cert*

*can be found
on classpath*

Enable ApacheDS LDAP SSL Server

1.Import keystore
with Apache
Directory Studio

2.Restart
ApacheDS Server



Or Enable OpenLDAP SSL Server

Add locations of crypto artifacts to slapd server config:

TLSCACertificateFile /path/to/my/ca-certificate

TLSCertificateFile /path/to/my/server-certificate

TLSCertificateKeyFile /path/to/my/private-key

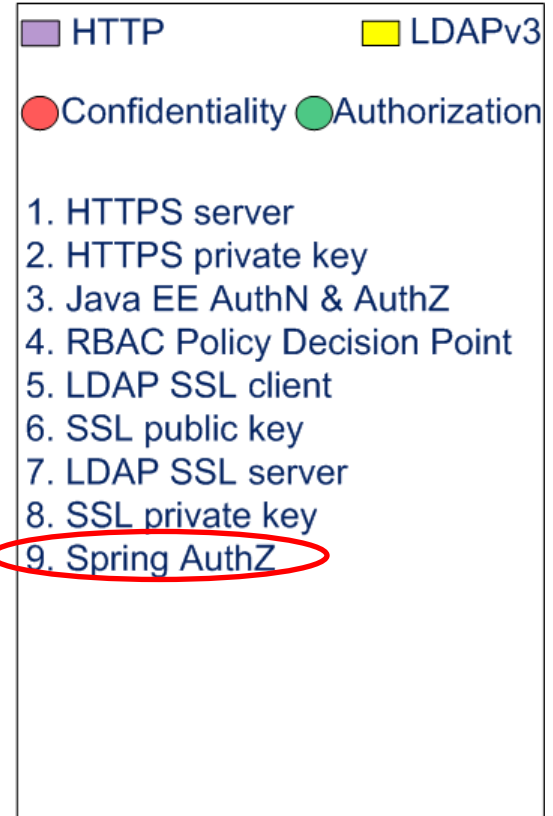
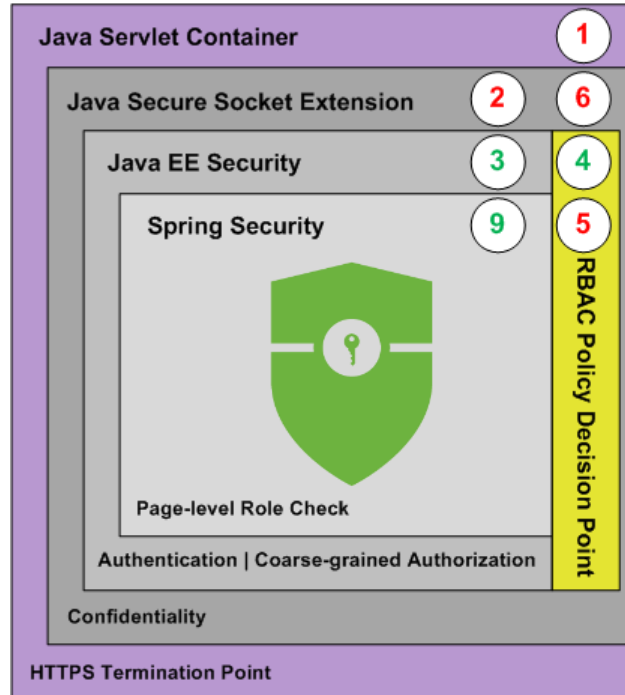
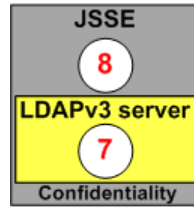
<http://shawnmckinney.github.io/apache-fortress-demo/apidocs/doc-files/openldap-ssl.html>

9. Enable Spring Security

a. Authorization

b. Role mapping

locks on the rooms



Enable Spring Security

Add dependencies to [pom](#):

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-core</artifactId>
  <version>4.1.3.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>4.1.3.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>4.1.3.RELEASE</version>
</dependency>
```

Add the Spring Context File to App

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>
```

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"  
    />
```

page-level
authorization
(declarative)

```
</sec:filter-security-metadata-source>  
</property>  
</bean>
```

By default name must contain ROLE_

Role Mapping

Role Propagation between 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

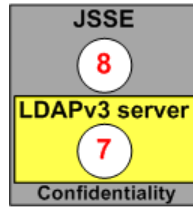
Complete list of eligible roles found in app's [web.xml](#):

```
<!-- 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>
```

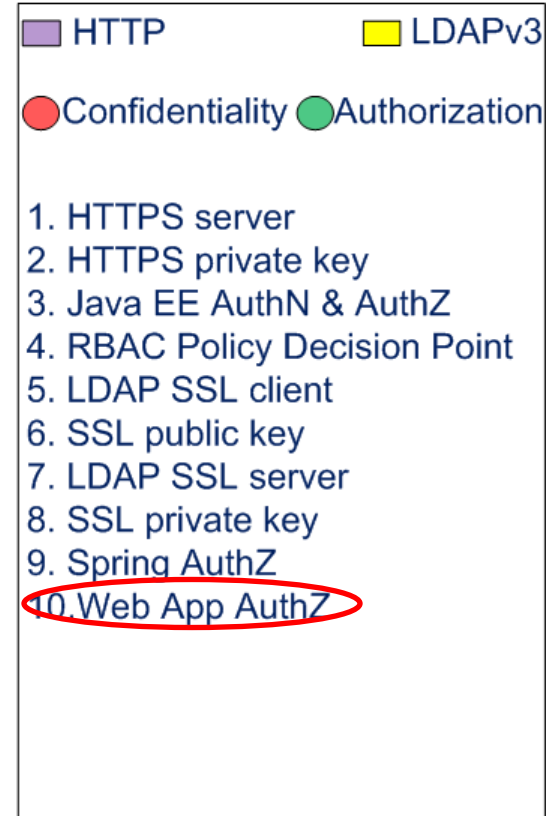
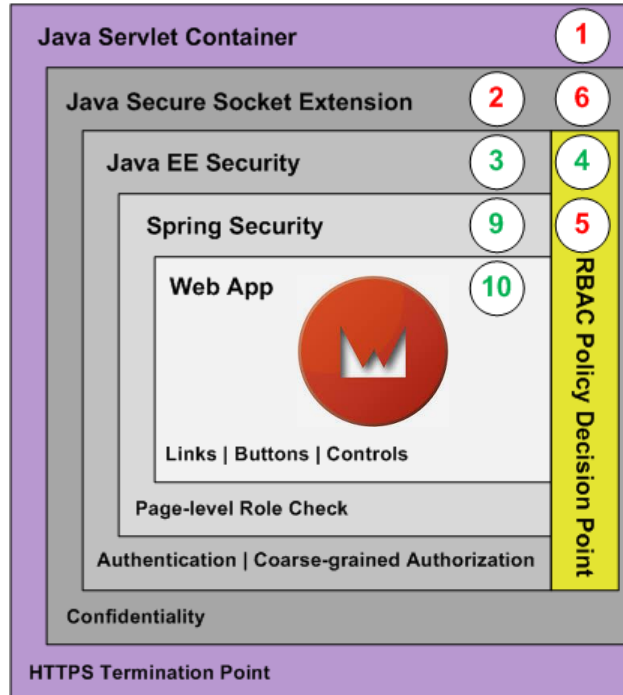
10. Web App Authorization

Add fine-grained checks:

- Page links
- Buttons
- Other controls



Locks on equipment



Add the Fortress Web Dependency

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

```
<dependency>
```

```
  <groupId>org.apache.directory.fortress</groupId>
```

```
  <artifactId>
```

```
    fortress-web
```

```
  </artifactId>
```

```
  <version>2.0.0</version>
```

```
</dependency>
```

Inject Fortress APIs via Spring Beans

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>
```

Share the Session with Tomcat

Session Propagation between Tomcat, Fortress and Web app:

1. The Fortress Tomcat Realm creates the session after user successfully authenticates. It serializes the data and stores inside a principal object.

2. Tomcat returns the serialized principal to Web app on request:

```
String szPrin = servletRequest.getUserPrincipal().toString();
```

<- Standard Java api

3. Next deserialize the java security principal into a 'Fortress' session:

```
Session ftSess = j2eePolicyMgr.deserialize( szPrin );
```

<- Fortress Realm api

4. Store the Fortress session into an HTTP session object for later usage:

```
myAppFw.setSession( ftSess );
```

<- web app's own api

Add Web Framework Security

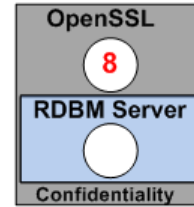
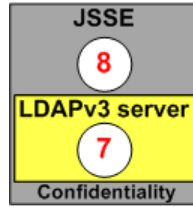
```
public class Page1 extends MyBasePage
{
    Add( new SecureIndicatingAjaxButton( "Page1", "Add" )
    {
        @Override
        protected void onSubmit( ... )
        {
            if ( checkAccess ( customerNumber )
            {
                // do something here:
            }
           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

*fine-grained
authorization
(programmatic)*

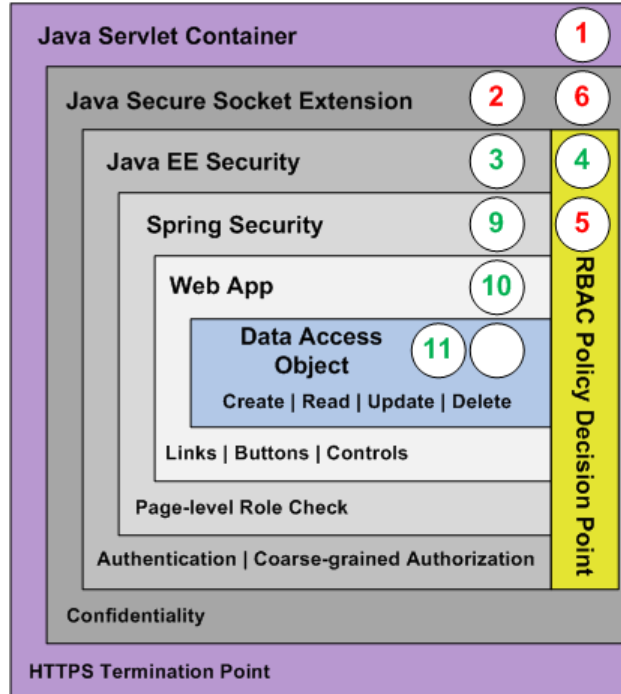
11. DAO Authorization

Add fine-grained Checks to:

- Create
- Read
- Update
- Delete



filtering



■ HTTP ■ JDBC ■ 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
11. DAO AuthZ

Add Security Aware DAO components

```
public class Page1DaoMgr implements Serializable  
{...
```

```
public Page1EO updatePage1( Page1EO entity )
```

```
    if (checkAccess ("Page1", "Update", entity.getCust()))
```

```
    {  
        // Do normal DAO.update stuff here...  
    }  
    else  
        throw new RuntimeException("Unauthorized");  
    ...  
    return entity;  
}
```

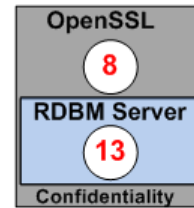
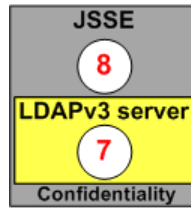
```
...  
}
```

*fine-grained
authorization
(programmatic)*

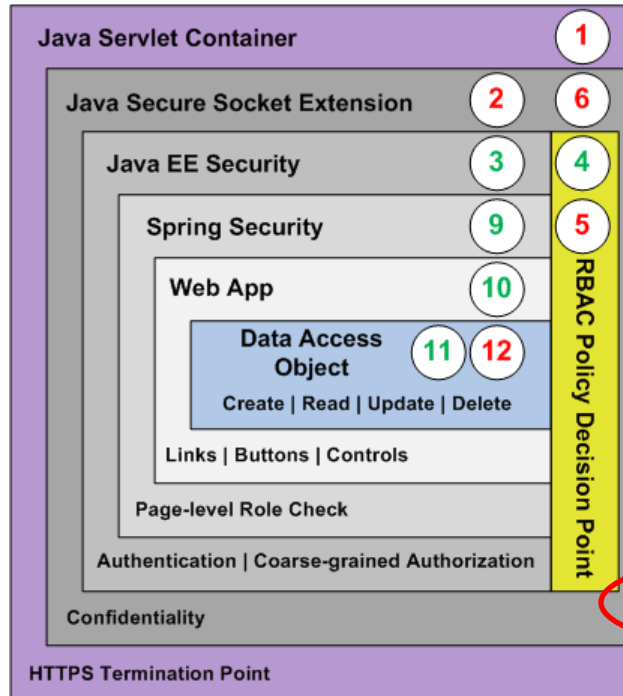
12, 13. Enable DB SSL

12. Client
a. public key
b. config

13. Server
a. private key
b. config



Confidentiality



■ HTTP ■ JDBC ■ 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
11. DAO AuthZ
12. JDBC SSL client
13. Database SSL server

Enable JDBC SSL Client

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

```
trust.store=/path/mytruststore  
trust.store.onclasspath=false
```

*must be found
on file path*

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 JDBC SSL Client

Add to [applicationContext.xml](#) of Web app:

```
<context:property-placeholder location="classpath: fortress.properties" />
  <bean class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
    <property name="targetObject">
      <bean class="org.springframework..MethodInvokingFactoryBean">
        <property name="targetClass" value="java.lang.System" />
        <property name="targetMethod" value="getProperties" />
      </bean>
    </property>
    <property name="targetMethod" value="putAll" />
    <property name="arguments">
      <util:properties>
        <prop key="javax.net.ssl.trustStore">${trust.store}</prop>
        <prop key="javax.net.ssl.trustStorePassword">${trust.store.password}</prop>
        <prop key="javax.net.debug">${enable.ldap.ssl.debug}</prop>
      </util:properties>
    </property>
  </bean>
```

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://shawnmckinney.github.io/apache-fortress-demo/apidocs/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

Apache Fortress Demo 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

Testing

- Verify security functionality via automation.
- Otherwise vulnerabilities may still exist in your system.

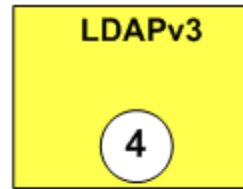
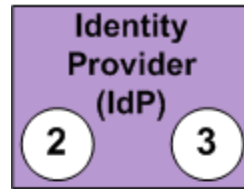
<https://github.com/shawnmckinney/apache-fortress-demo/.../ApacheFortressDemoSeleniumITCase.java>

Apache Fortress Demo

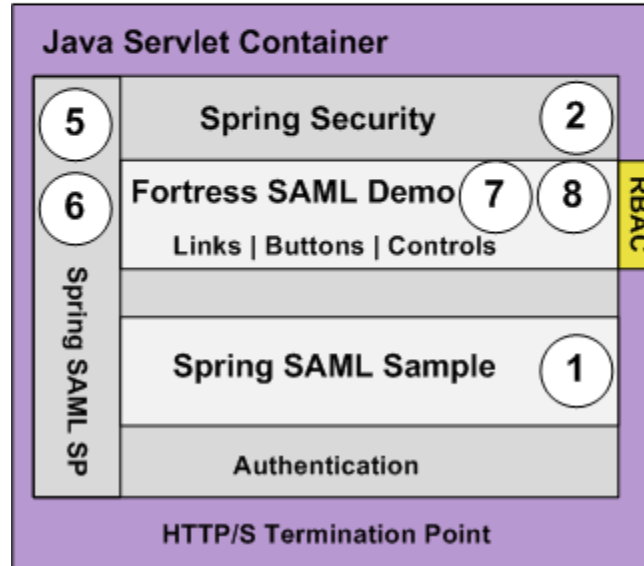
- <https://github.com/shawnmckinney/apache-fortress-demo>

User Foo	Customer 123	Customer 456	Customer 789
Page1	False	True	True
Page2	True	False	False
Page3	True	False	False

Example #2



Fortress SAML Demo



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

The Security Layers with SAML

1. Java SE Security

2. JSSE

~~3. Java EE Security~~ ← Turned off (for now)

4. Spring Security ← Deadbolt is now here

5. Web App Framework ←

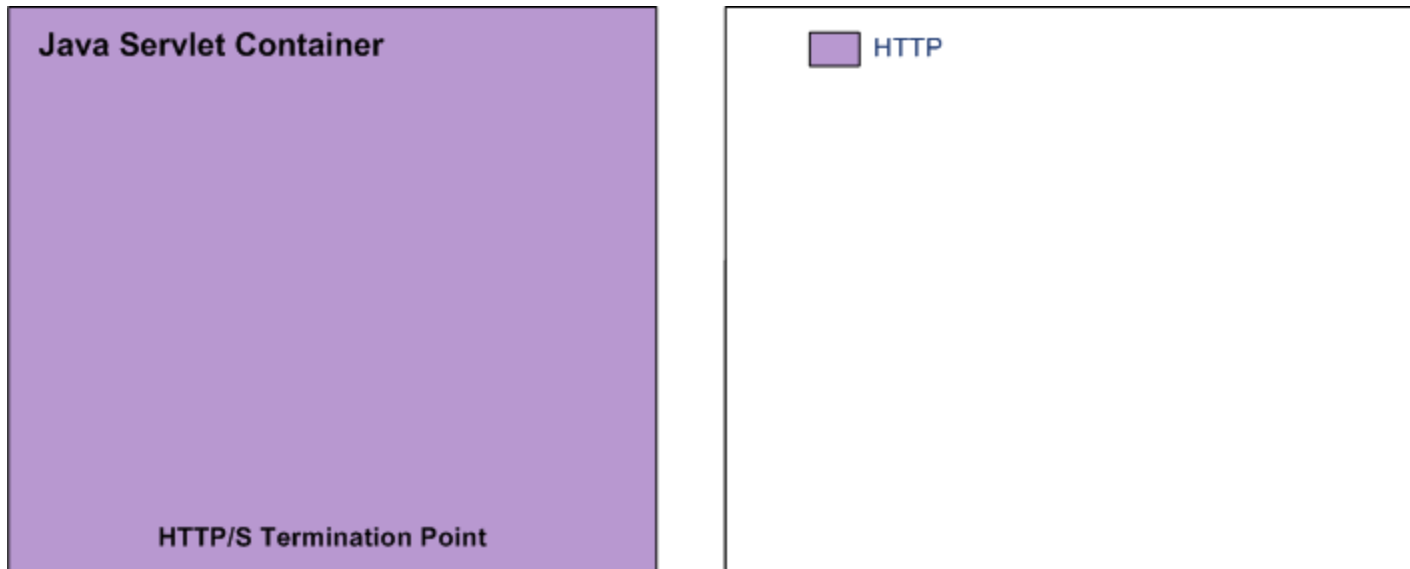
6. Database Functions ← Not much to change

Two Areas of Access Control

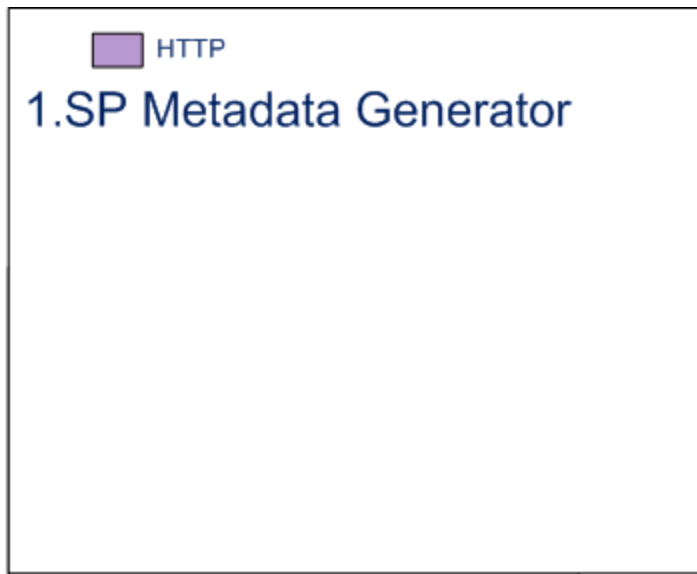
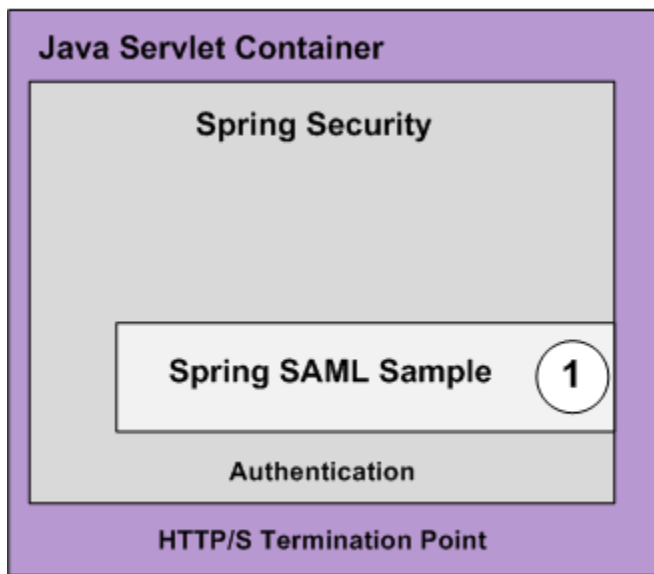
1.Spring SAML Declarative checks

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

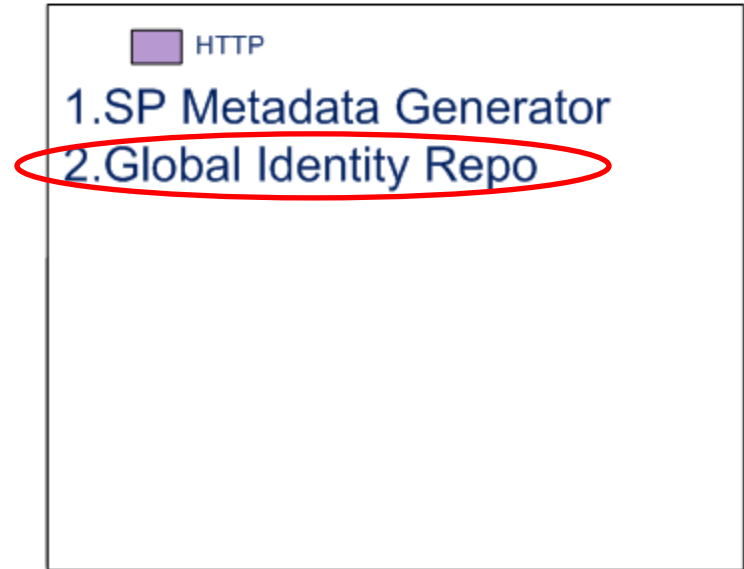
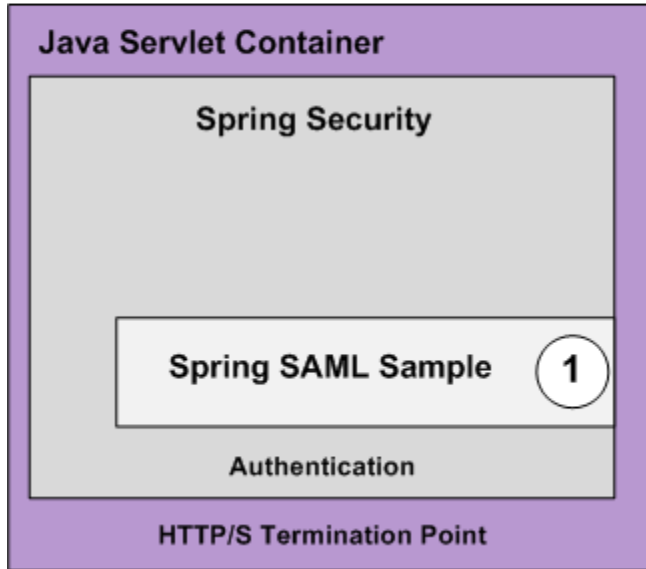
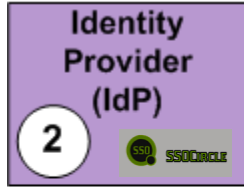
fortress-saml-demo

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

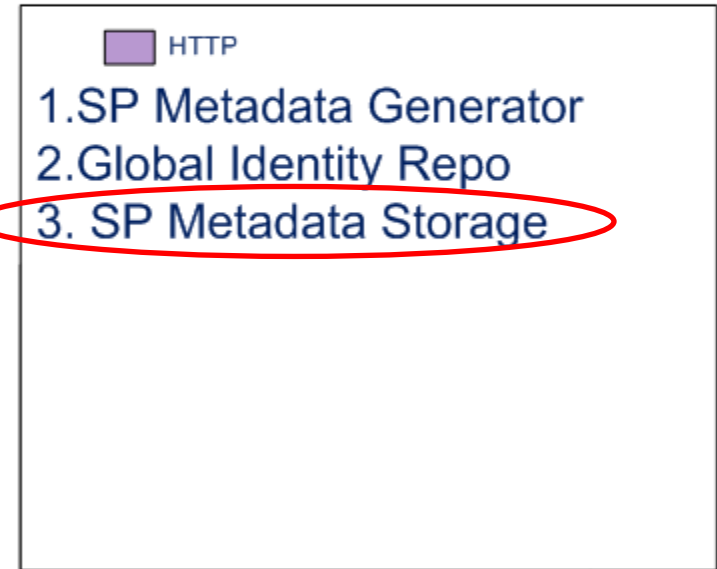
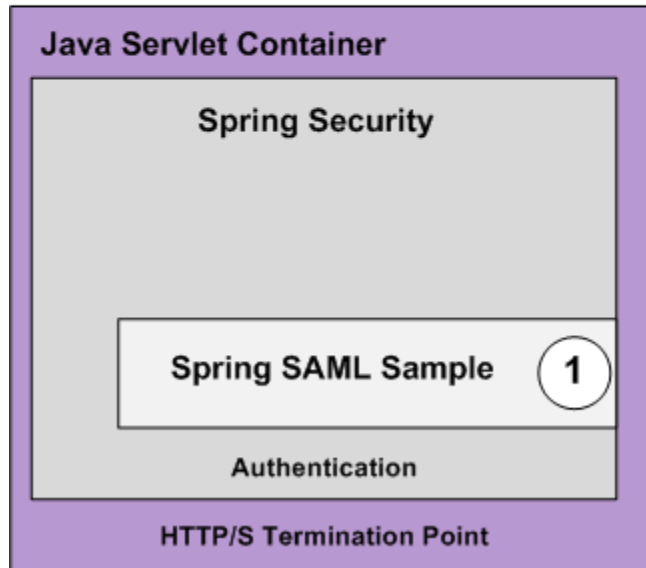
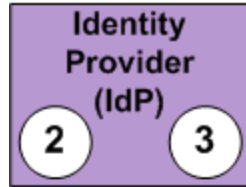
Creating your Identity with SSOCircle (from their website)

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

Start

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:

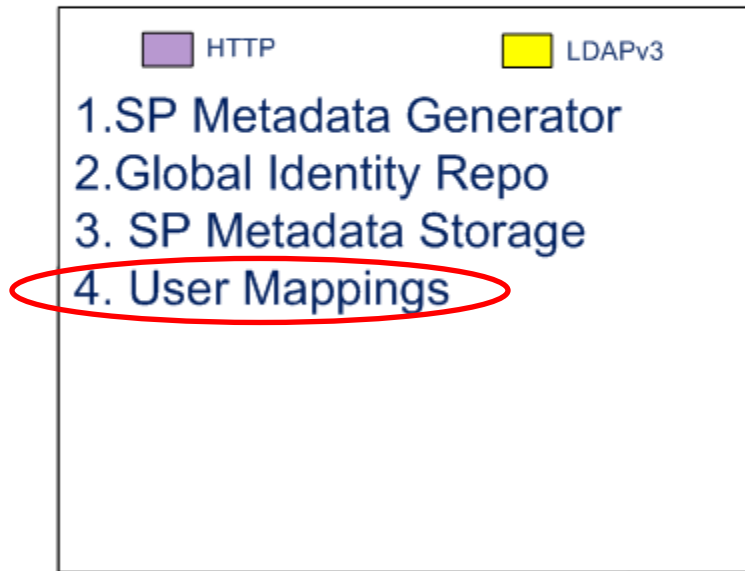
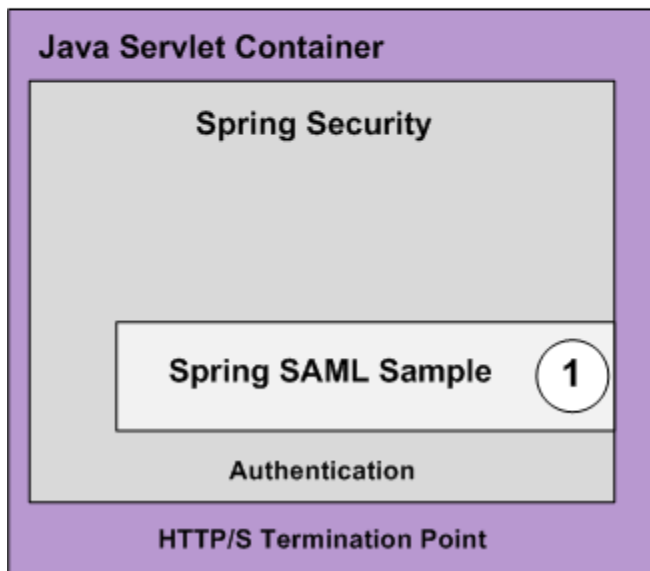
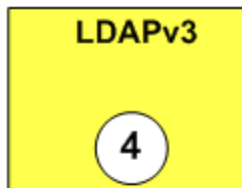
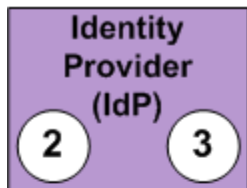
Enter the FQDN of the ServiceProvider ex.: sp.cohos.de



`sp2.symas.com`

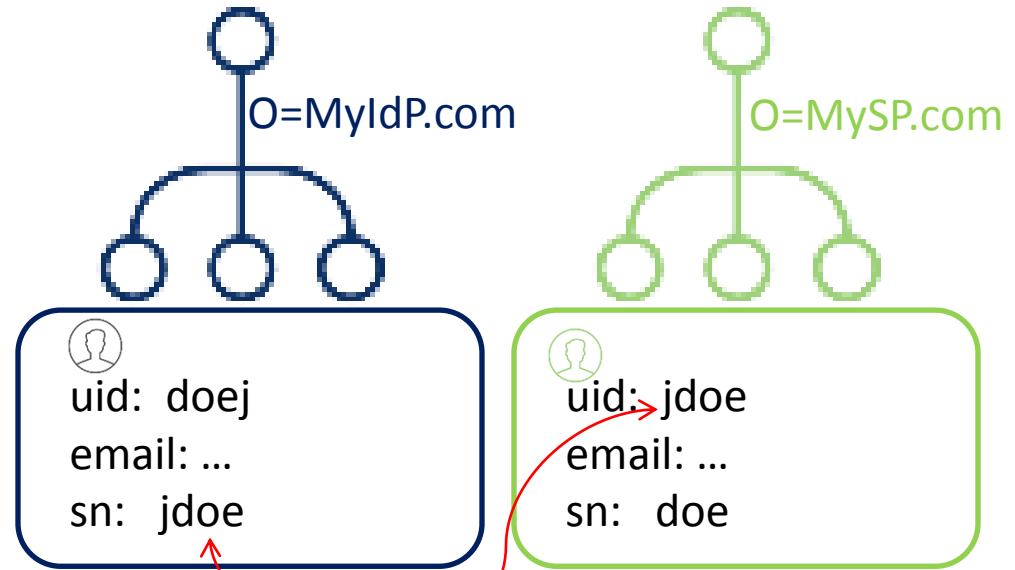


4. IdP and SP User Account Mapping



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

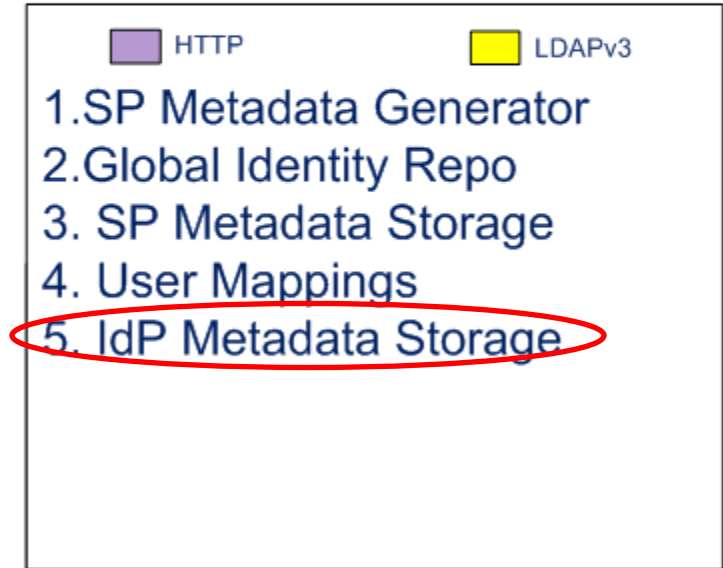
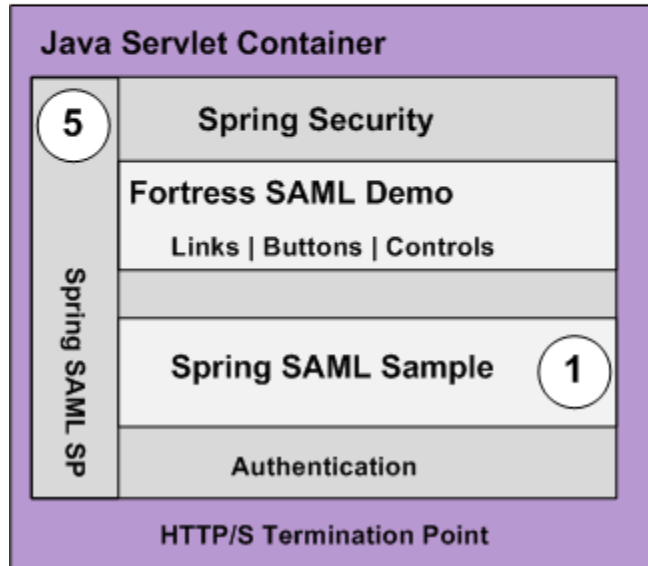
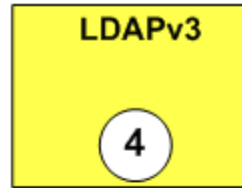
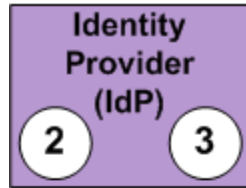
SAML Attribute Statement

```
<?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"/>
```

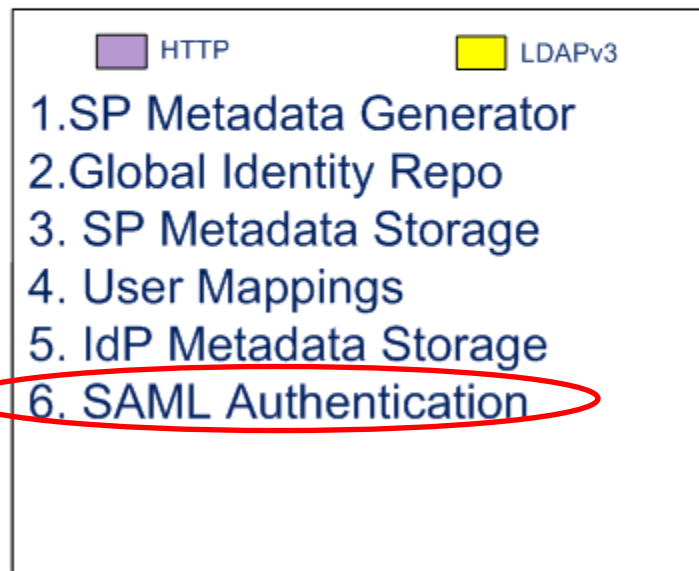
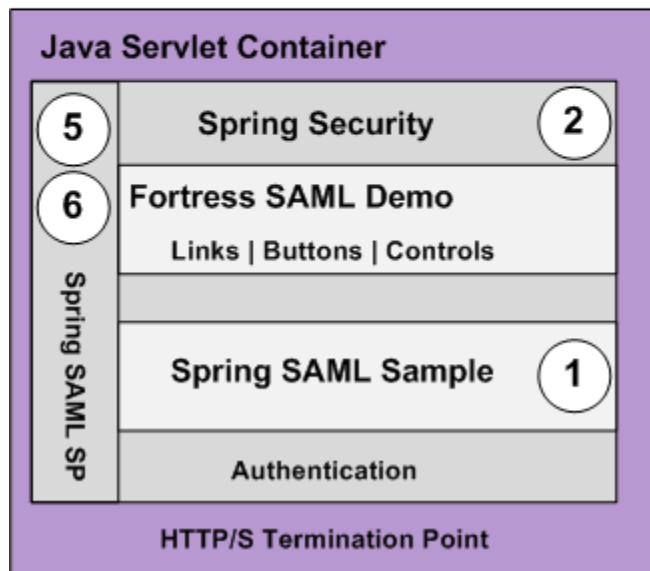
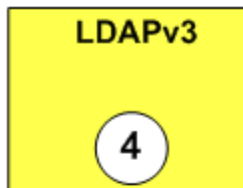
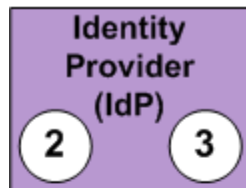
```
</bean>
```

```
</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"/>
```

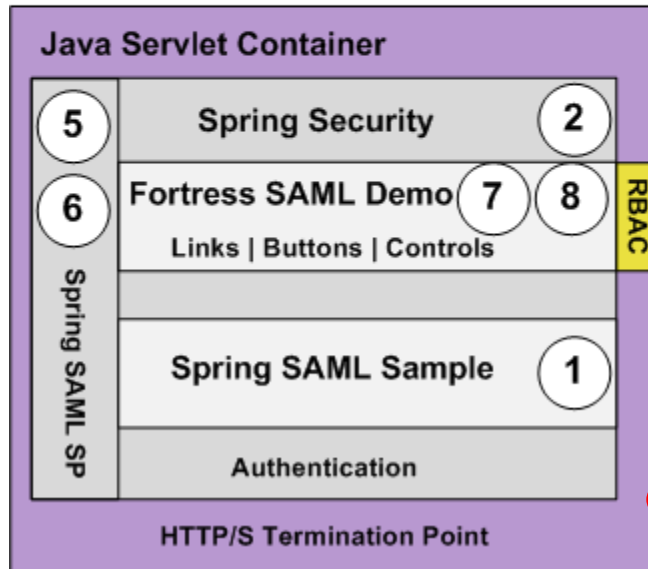
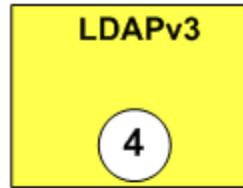
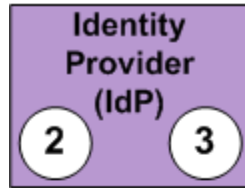
```
<security:filter-chain pattern="/saml/SSOHoK/*" filters="samlWebSSOHoKProcessingFilter"/>
```

```
<security:filter-chain pattern="/saml/SingleLogout/*" filters="samlLogoutProcessingFilter"/>
```

```
</security:filter-chain-map>
```

Little Rock Java User Group 2017

7. Setup RBAC Policy Decision Point



Enable RBAC Policy Decision Point

<dependency>

<groupId>

org.apache.directory.fortress

</groupId>

<artifactId>

fortress-realm-impl

</artifactId>

<version>2.0.0</version>

</dependency>

Share ID between Spring & Fortress

Get uid from the SAML assertion, create fortress session.

1. Spring SAML filter creates security principal based on attributes found in the SAML attribute assertion.

2. Web app parses the surName attribute contained in principal:

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

<- web app api

3. Web app creates a Fortress session using attribute in the principal:

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

<- Fortress realm api

4. Web app pushes RBAC session into HTTP session.

```
myAppFw.setSession( ftSess );
```

<- web app api

*isTrusted
(no password
req'd)*

Parse the ID from SAML Assertion

```
public class SecUtils
```

```
{ ...  
    private static String getSurName(SAMLCredential credential)  
    {  
        String userId = null;  
        for ( org.opensaml.saml2.core.Attribute attr : credential.getAttributes())  
        {  
            String name = attr.getName();  
            if(StringUtils.isEmpty( name ) )  
                break;  
            else if( name.equals( "LastName" ) )  
            {  
                String vals[] = credential.getAttributeAsStringArray( attr.getName() );  
                userId = vals[0];  
                break;  
            }  
        }  
        return userId;  
    }  
}
```

Add Secure Web Components

```
public class Page1 extends SamlSampleBasePage
{
    ...
    add( new FtIndicatingAjaxButton( "Page1", "Add" )
    {
        @Override
        protected void onSubmit( ... )
        {
            // do something here:
        }
    } );
}
```

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.saml.sample.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 SSO CIRCLE

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>