CON 2324
A Practical Guide to Role Engineering

Shawn McKinney

October 27, 2015

JavaOne

San Francisco

Session Objectives

- ✓ Gain understanding of the Role-Based Access Control (RBAC) standard
- ✓ Learn the repeatable steps of the Role Engineering Process
- ✓ Learn about RBAC using the Apache Directory Fortress engine

Introductions

Shawn McKinney

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

Agenda

- Role Engineering Defined
- RBAC (INCITS 359)
- Five Steps
- Demo
- Extra Credit
- Wrap-up

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

symas

Role Engineering Defined

Role Engineering is the process by which an organization develops, defines, enforces, and maintains role-based access control. RBAC is often seen as a way to improve security controls for access and authorization, as well as to enforce access policies such as segregation of duties (SoD) to meet regulatory compliance.

Enterprise Role Definition: Best Practices and Approach – Oracle Corporation (Blogs)
Nov 5, 2014

Intro to the RBAC Standard

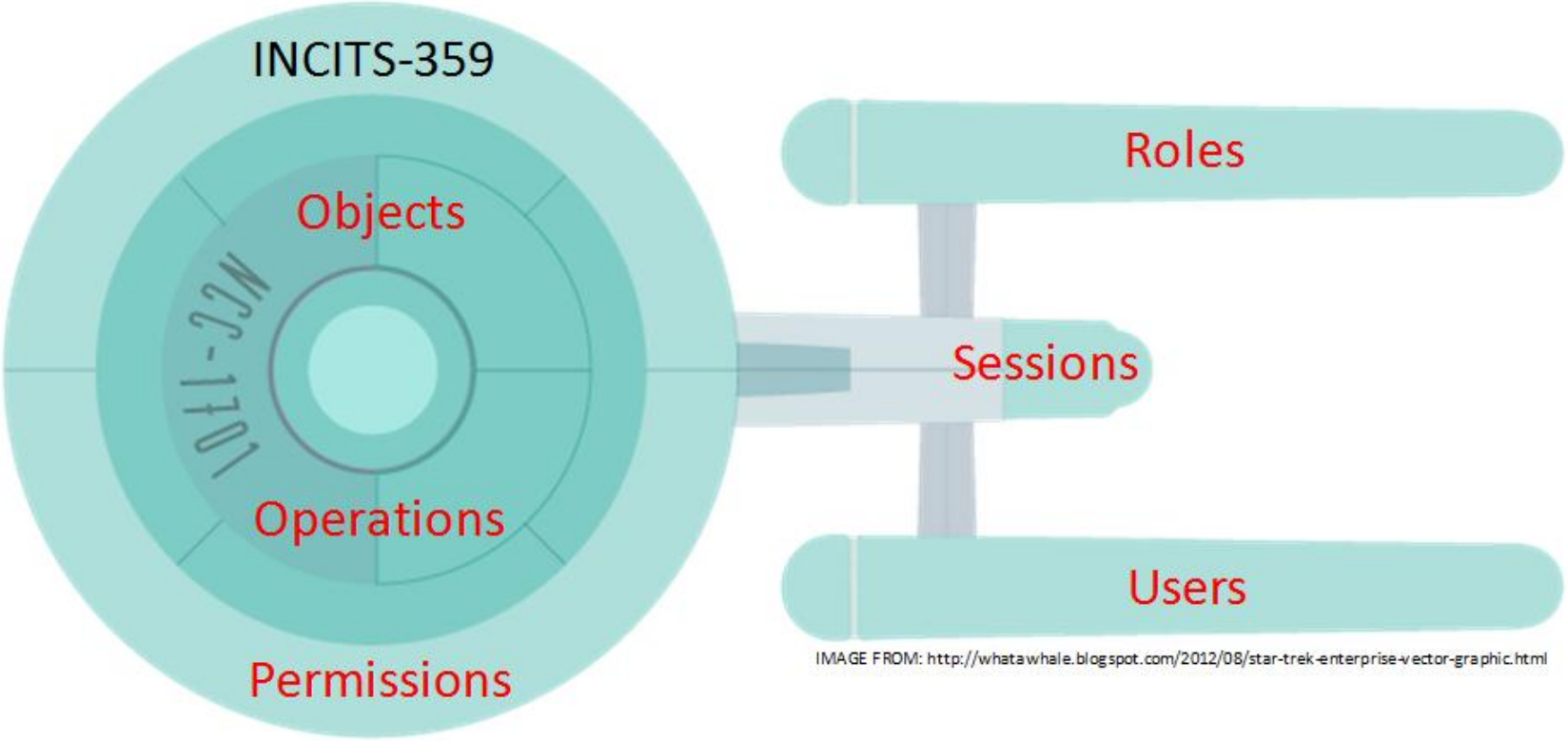


IMAGE FROM: http://whatawhale.blogspot.com/2012/08/star-trek-enterprise-vector-graphic.html

Early Years

- The Role-Based Access Control model was formally introduced in 1992 by David Ferraiolo and Richard Kuhn of National Institute of Standards and Technology.

- Their model, already in use for some time, was meant to address critical shortcomings of the Discretionary Access Control. DAC was not meeting the needs of non-DoD organizations.

- In particular integrity was lacking, defined by them, as the requirement for data and process to be modified only in authorized ways by authorized users.

Middle Years

- Eight years later, in 2000, they teamed with Ravi Sandhu and produced another influential paper entitled ‘The NIST Model for a Role-Based Access Control: Towards a Unified Standard’.

- Later the team released the RBAC formal model. One that laid out in discrete terms how these types of systems were to work. The specifications, written in Z-notation, left no ambiguity whatsoever.

- This model formed the basis for the standard that followed:
  - ANSI INCITS 359

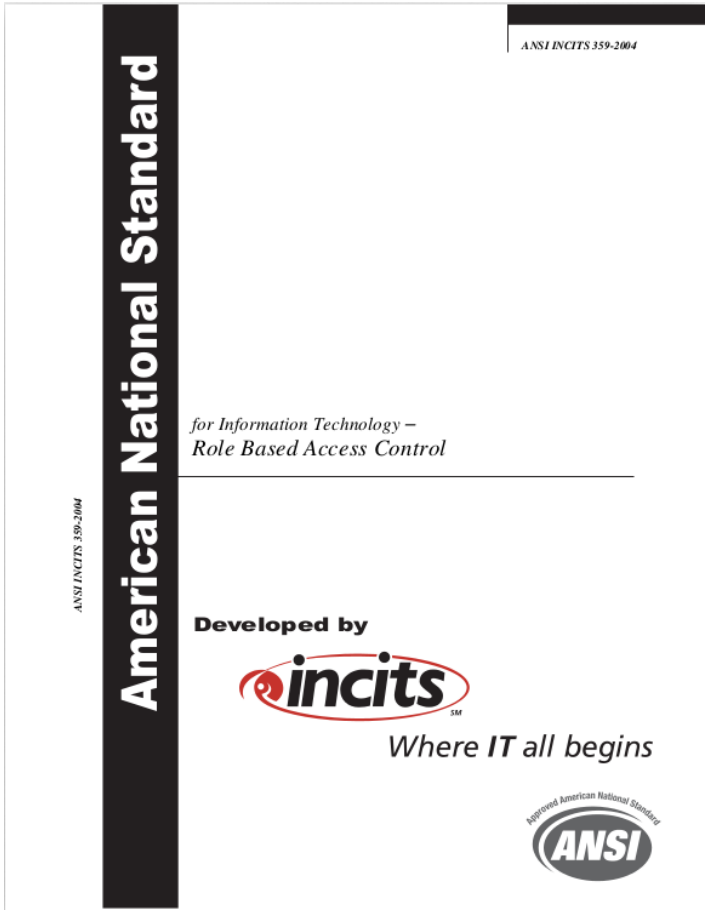
Current Years

- INCITS 359-2012 RBAC also known as Core.
- INCITS 494-2012 RBAC Policy Enhanced allows attribute modifiers on permissions specifically to provide support for fine-grained authorization.

ANSI INCITS 359

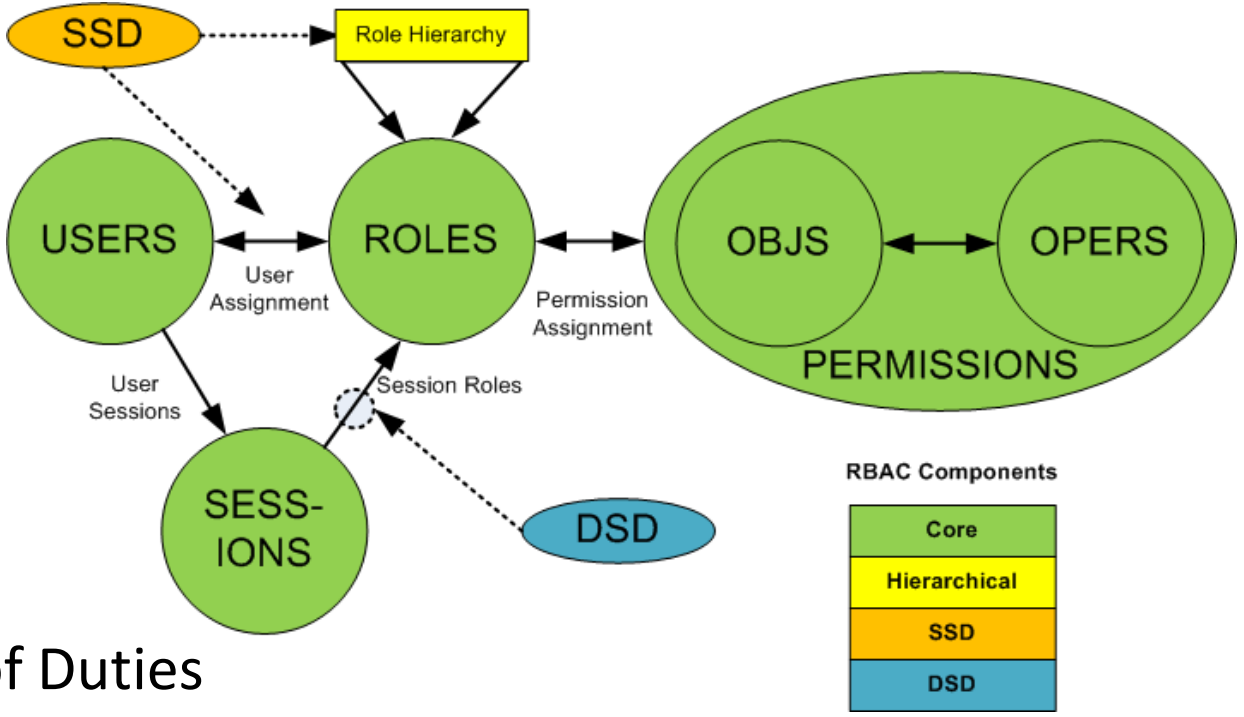
In 2004 ANSI Formalized RBAC into a Standard

(Revised in 2012)



ANSI INCITS 359

RBAC0:
Users, Roles,
Perms, Sessions
RBAC1:
Hierarchical Roles
RBAC2:
Static Separation
of Duties
RBAC3:
Dynamic Separation of Duties



ANSI RBAC Object Model

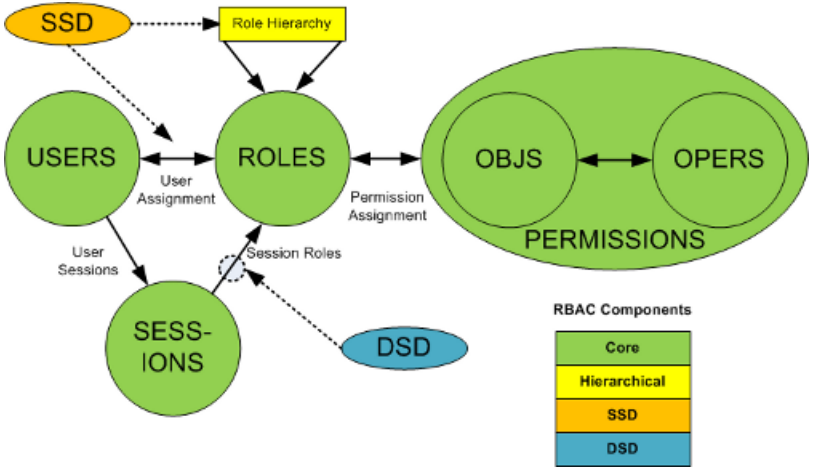
Six basic elements:

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

ANSI RBAC Functional Model

Three standard interfaces:

1. Administrative – CRUD
2. Review – policy interrogation
3. System – policy enforcement



Admin RBAC

[Link to AdminMgr javadoc]

Fortress Admin APIs map to the INCITS 359 specs

```
public interface AdminMgr {
  User addUser( User user );
  void deleteUser( User user );
  Role addRole( Role role );
  void deleteRole( Role role );
  void assignUser( UserRole uRole );
  void deassignUser( UserRole uRole );
  Permission addPermission( Permission perm );
  void deletePermission( Permission perm );
  void grantPermission( Permission perm, Role role );
  void addAscendant( Role childRole, Role parentRole);
  void addDescendant(Role parentRole, Role childRole);
  void addDsdRoleMember(SDSet dsdSet, Role role);
  void addInheritance(Role parentRole, Role childRole)
… http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git
```

$$GrantPermission(object, operation, role{:}\ NAME) \triangleleft$$

$$(operation, object) \in PERMS; role \in ROLES$$

$$PA' = PA \cup \{(operation, object) \mapsto role\}$$

$$assigned\_permissions' = assigned\_permissions \setminus \{role \mapsto assigned\_permissions(roles)\} \cup \{role \mapsto (assigned\_permissions(role) \cup \{(operation, object)\})\} \triangleright$$

[Link to INCITS 359 spec]

Review RBAC

[Link to ReviewMgr javadoc]

Fortress Review APIs map to the INCITS 359 specs

```
public interface ReviewMgr {
    Permission readPermission( Permission permission );
    List<Permission> findPermissions( Permission permission );
    User readUser( User user );
    List<User> findUsers( OrgUnit ou );
    List<User> assignedUsers( Role role );
    Set<String> authorizedRoles( User user );
    List<Permission> rolePermissions( Role role );
    List<Permission> userPermissions( User user );
    Set<String> authorizedPermissionUsers(Permission perm);
    SDSet dsdRoleSet(SDSet set);
    Set<String> dsdRoleSetRoles(SDSet dsd);
    List<SDSet> dsdRoleSets(Role role);
    SDSet ssdRoleSet(SDSet set);
    Set<String> ssdRoleSetRoles(SDSet dsd);
    List<SDSet> ssdRoleSets(Role role);
    List<Role> findRoles(String searchVal);
…
```

http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git

$$UserPermissions(user: NAME; result: 2^{PERMS}) \triangleleft$$

$$user \in USERS$$

$$result = \{r: ROLES; op: OPS; obj: OBJS \mid (user \mapsto r) \in UA \wedge ((op, obj) \mapsto r) \in PA \bullet (op, obj)\} \triangleright$$

[Link to INCITS 359 spec]

Usages for Admin and Review

- Build a web user interface
- Generate reports to verify compliance
- Migrate from one security system to another
- Covers use cases didn’t know you had

Usages for Review – Role Assigned Users

Listing of Users assigned a particular Role:

```
List<User> assignedUsers(Role role)
```

This method returns the data set of all users who are assigned the given role.

Usages for Review – Role Authorized Users

Listing of Users authorized a particular Role:

```
List<User>authorizedUsers(Role role)
```

This function returns the set of users authorized to a given role, i.e., the users that are assigned to a role that inherits the given role.

Usages for Review – User Perms

Listing of Perms for a particular User:

```
List<Permission>
    userPermissions(User user)
```

This function returns the set of permissions a given user gets through his/her authorized roles.

Usages for Review – Perm Users

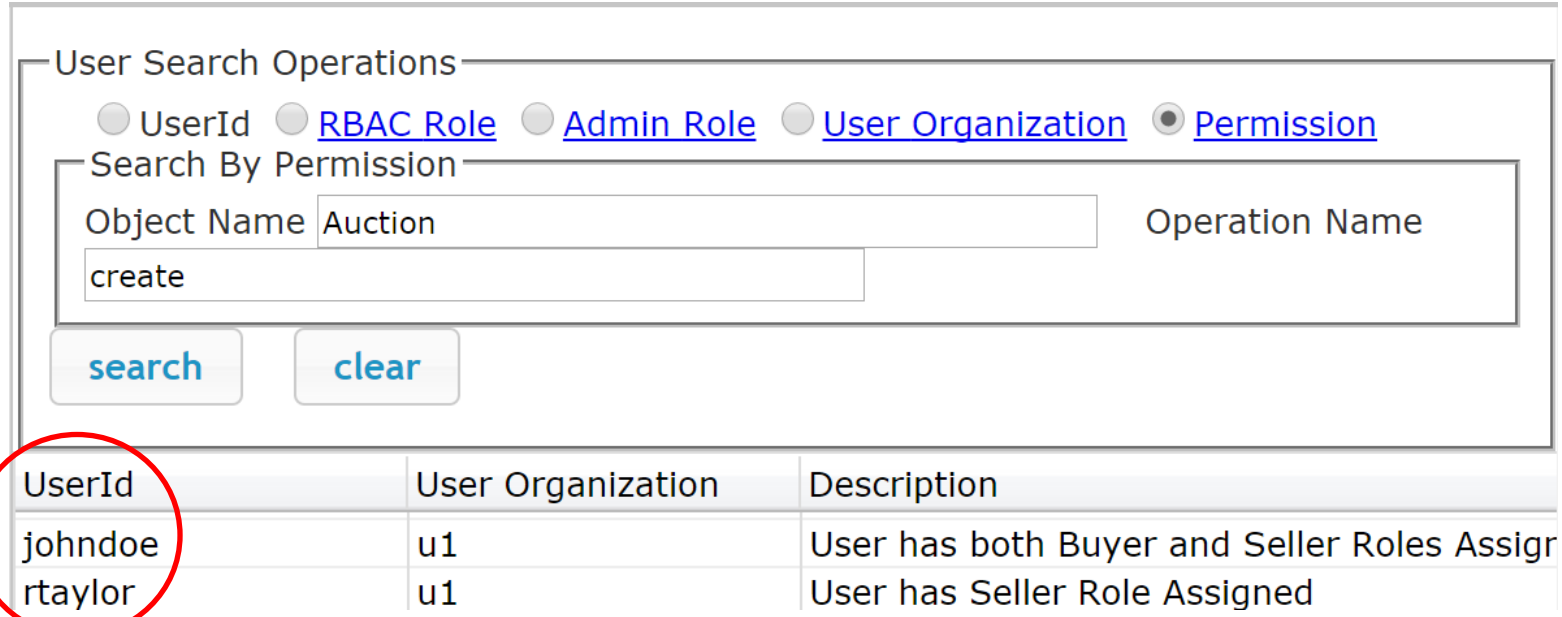
Listing of Users for a particular Perm:

```
List<String>
permissionUsers(Permission perm)
```

Return all userIds that have been granted (directly) a particular permission.

Usages for Review – Perm Users

List of Users who can create an auction:



Implementations of RBAC Admin and Review

1. Apache Fortress Rest
   - http://git-wip-us.apache.org/repos/asf/directory-fortress-enmasse.git

2. Apache Fortress Web
   - http://git-wip-us.apache.org/repos/asf/directory-fortress-commander.git

System RBAC

[Link to AccessMgr javadoc]

Fortress AccessMgr APIs map to the INCITS 359 specs

```
public interface AccessMgr {
  Session createSession( User user, boolean isTrusted );
  List<Permission> sessionPermissions( Session session );
  Set<String> authorizedRoles( Session session );
  void addActiveRole( Session session, UserRole role );
  void dropActiveRole( Session session, UserRole role );
  User getUser( Session session );
  boolean checkAccess( Session session, Permission perm);
}
```

http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git

$$CheckAccess(session, operation, object: NAME; out\ result: BOOLEAN) \triangleleft$$

$$session \in SESSIONS; operation \in OPS; object \in OBJS$$

[Link to INCITS 359 spec]

$$result = (\exists r: ROLES \bullet r \in session\_roles(session) \wedge ((operation, object) \mapsto r) \in PA) \triangleright$$

Usages for System – Check Access

Permission check for a particular User:

```
boolean checkAccess(
    Session sess,Permission perm)
```

Perform user RBAC authorization. This function returns a Boolean value meaning whether the subject of a given session is allowed or not to perform a given operation on a given object. The function is valid if and only if the session is a valid Fortress session, the object is a member of the OBJS data set, and the operation is a member of the OPS data set.

Usages for System – Session Perms

Permissions for a particular User's Session:

```
List<Permission>
sessionPermissions(Session sess)
```

This function returns the permissions of the session, i.e., the permissions assigned to its authorized roles. The function is valid if and only if the session is a valid Fortress session.

Implementations of RBAC System

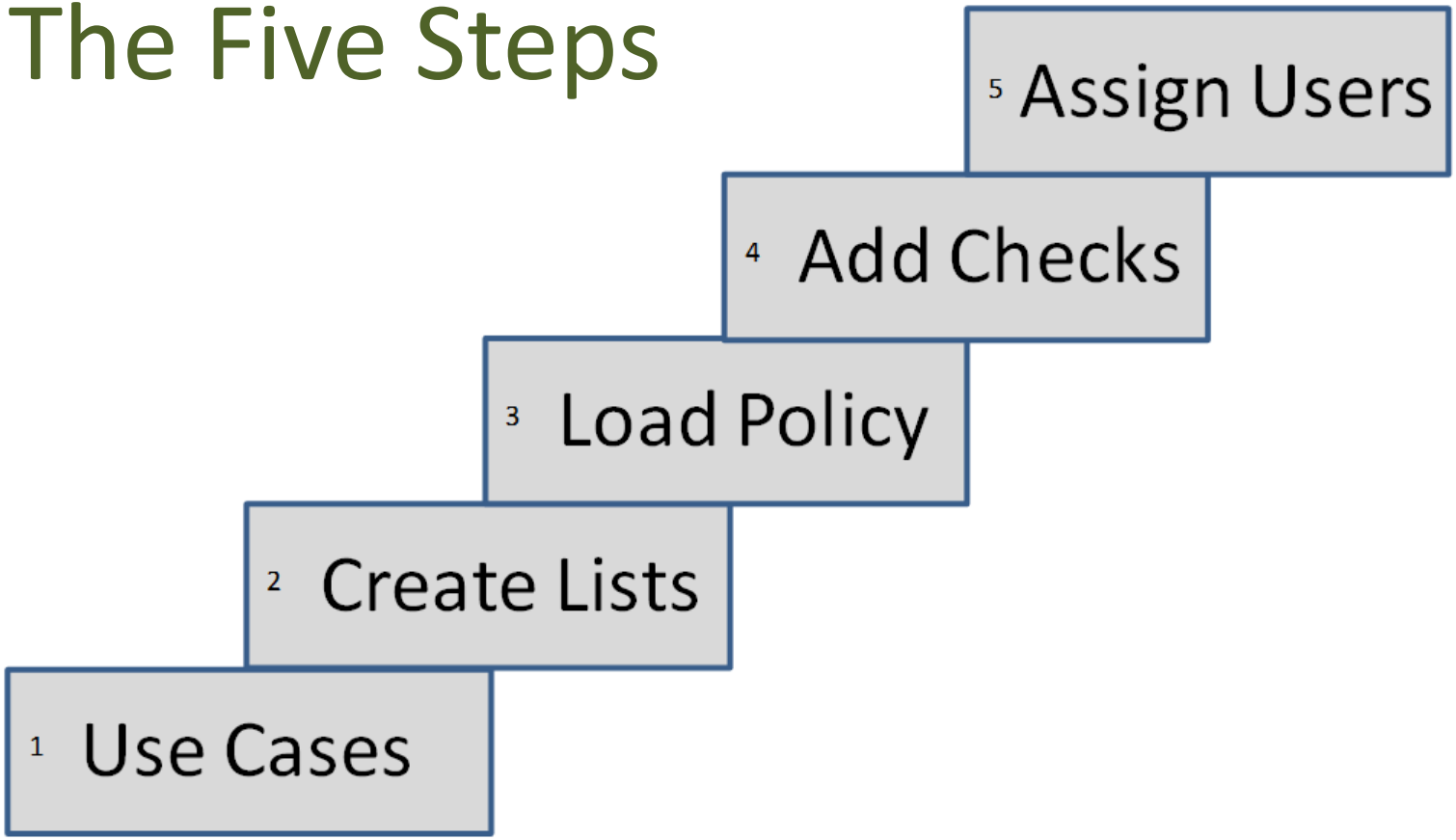
Apache Fortress Realm

- http://git-wip-us.apache.org/repos/asf/directory-fortress-realm.git

OpenLDAP Fortress Accelerator

- ssh://git-master.openldap.org/~git/git/openldap-fortress-accelerator.git

The Five Steps



Intro to the Five Steps

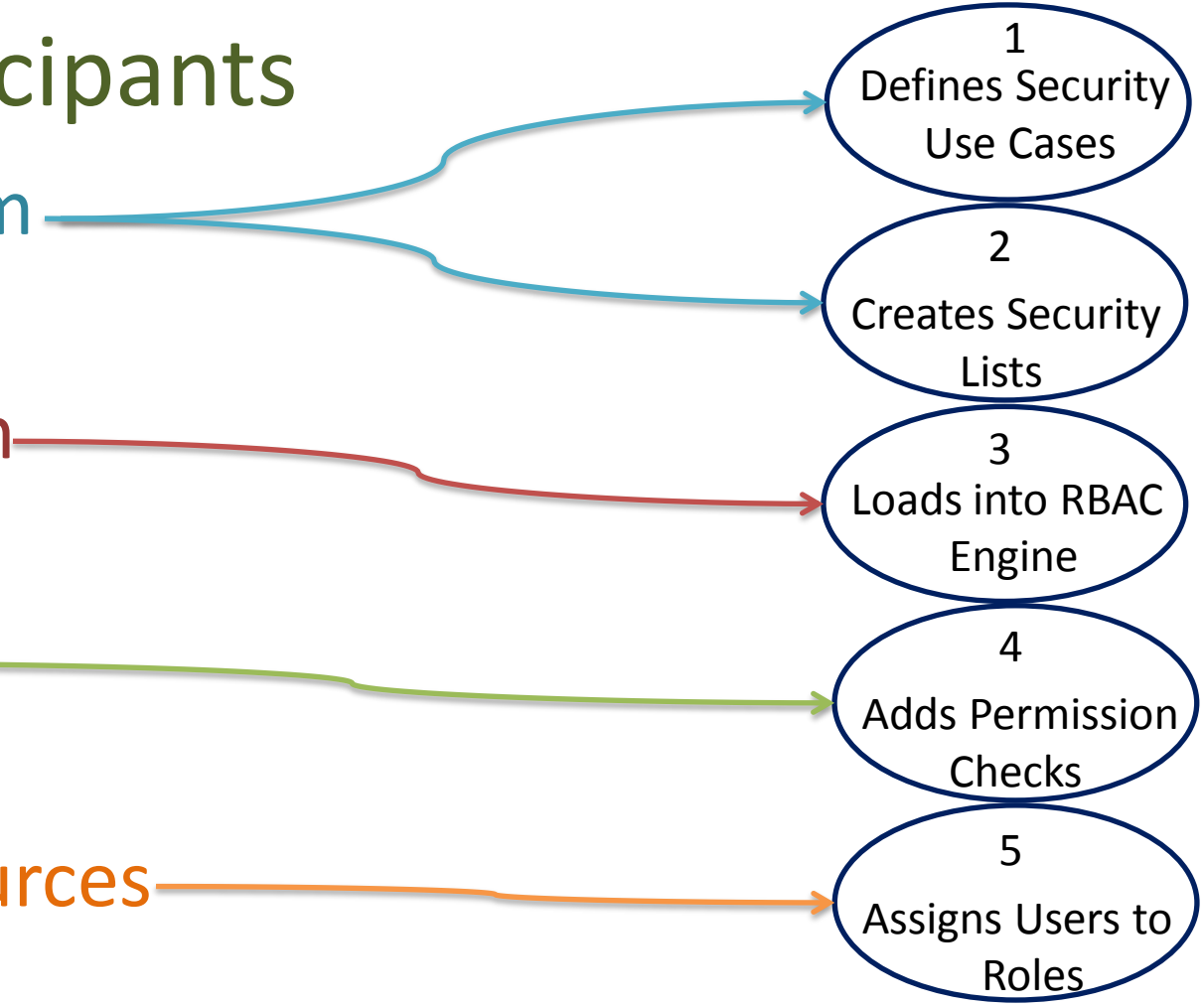
The Role engineering process goes from a human understandable policy definition, i.e. security use cases, to a format that may be loaded into an RBAC engine and used across a wide variety of applications and systems.

The Five Steps Defined

1. Define Security Use Cases
2. Derive into Entity Lists
3. Convert to Policy and Load into RBAC Engine
4. Add Policy Enforcement into the App
5. Assign Users to Roles

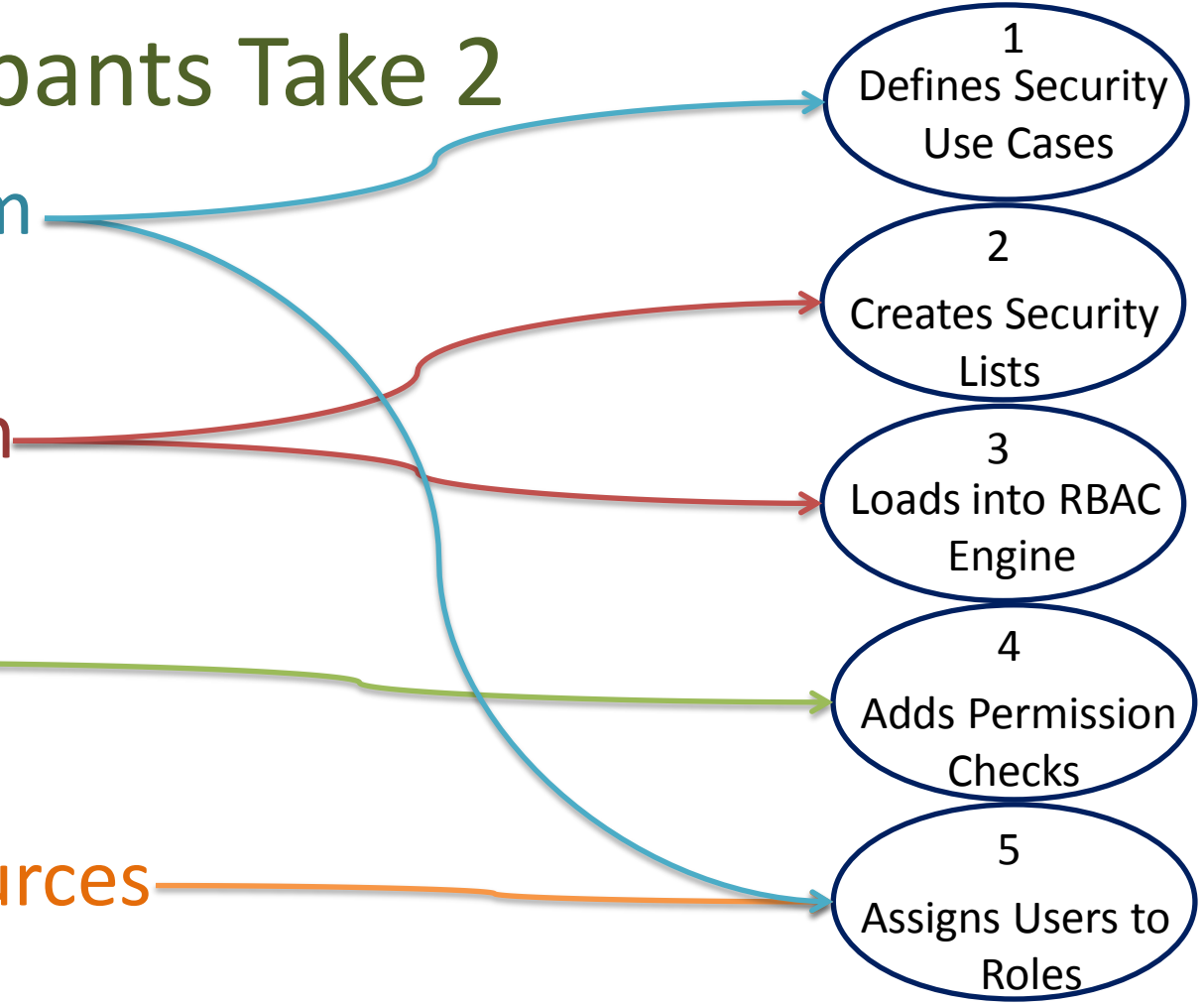
Process Participants

- Business Team
- Security Team
- Dev Team
- Human Resources

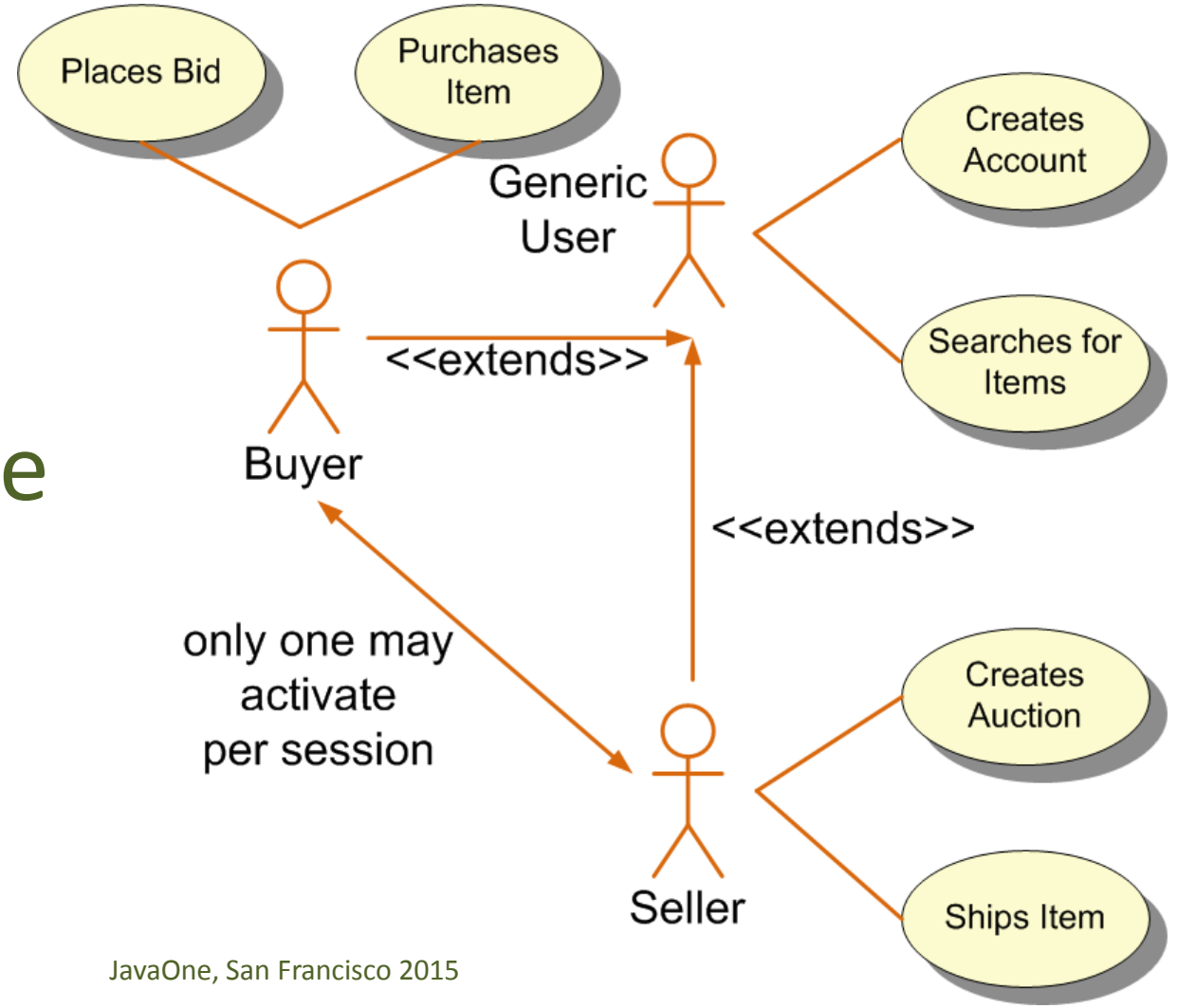


Process Participants Take 2

- Business Team
- Security Team
- Dev Team
- Human Resources



Step 1
Define the Use Cases



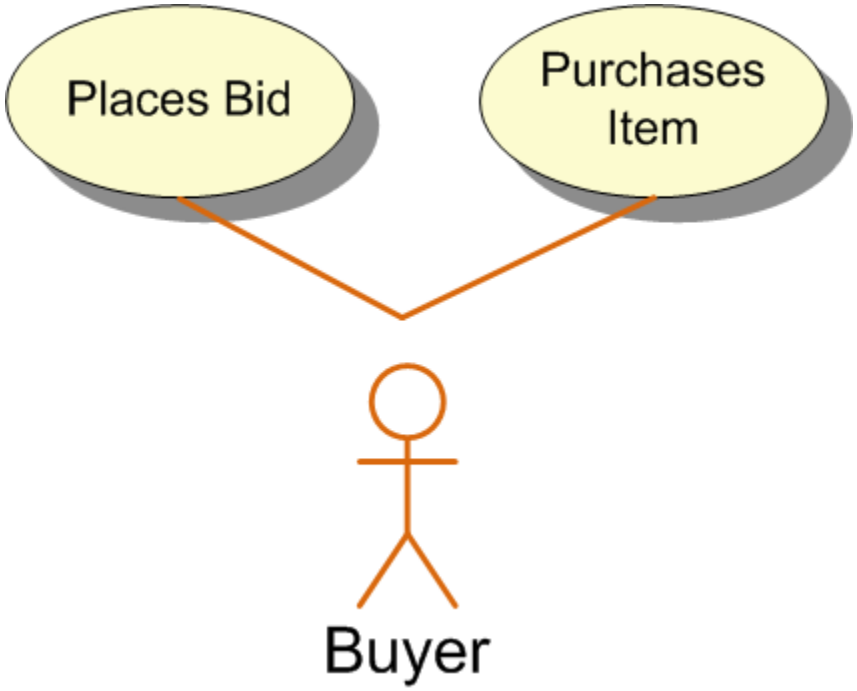
Use Case 1

Every User must authenticate before landing on the home page.

- Every set of security use cases necessarily contains a logon step.
- RBAC compliant systems call the createSession method where credentials are validated and roles activated into its session.

Use Case 2

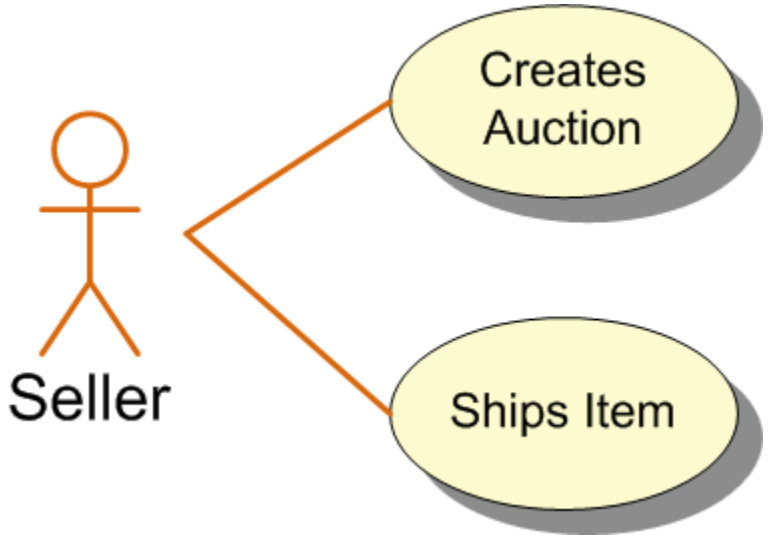
A User must be a ***Buyer*** *before placing a* ***bid*** *on an* ***auction*** *or* ***purchasing*** *an* ***item****.*



legend:
red:roles, **blue:objects**, **green:operations**

Use Case 3

*A User must be a **Seller** to **create** an **auction** or **ship** an **item** purchased.*

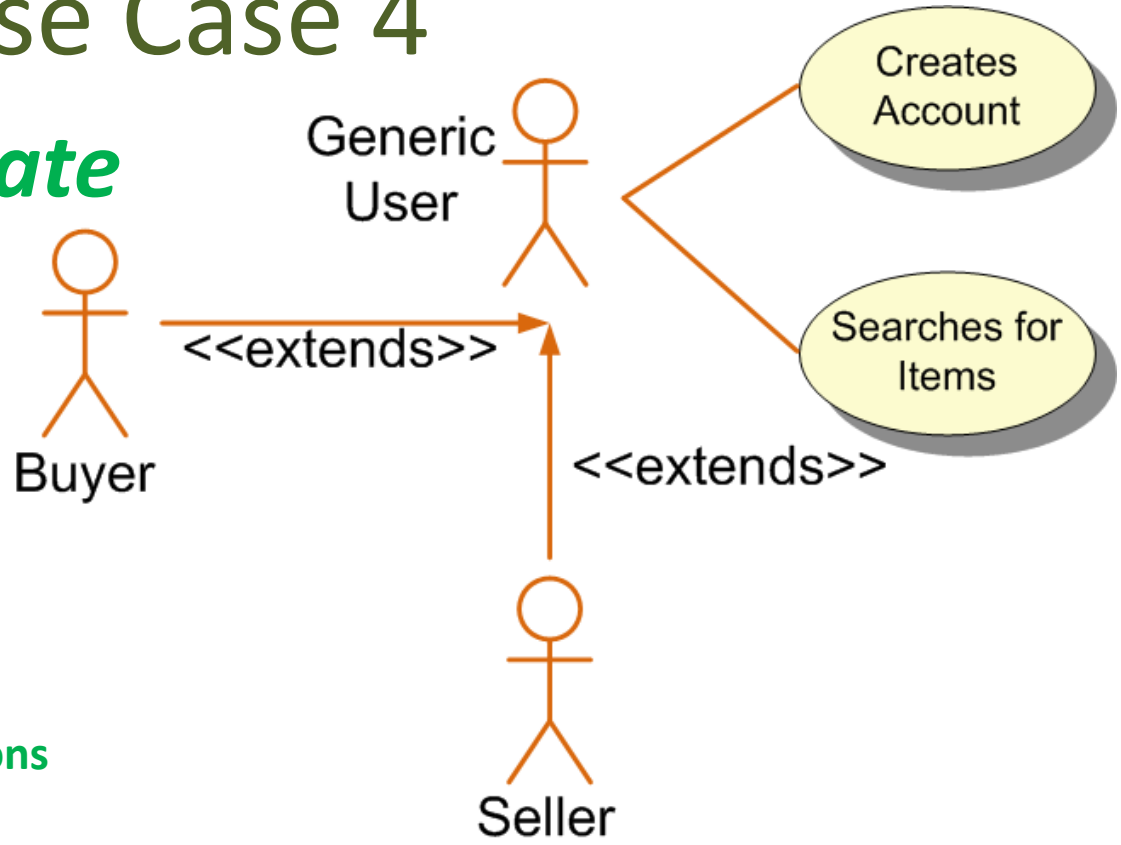


legend:

red:roles, **blue:objects**, **green:operations**

Use Case 4

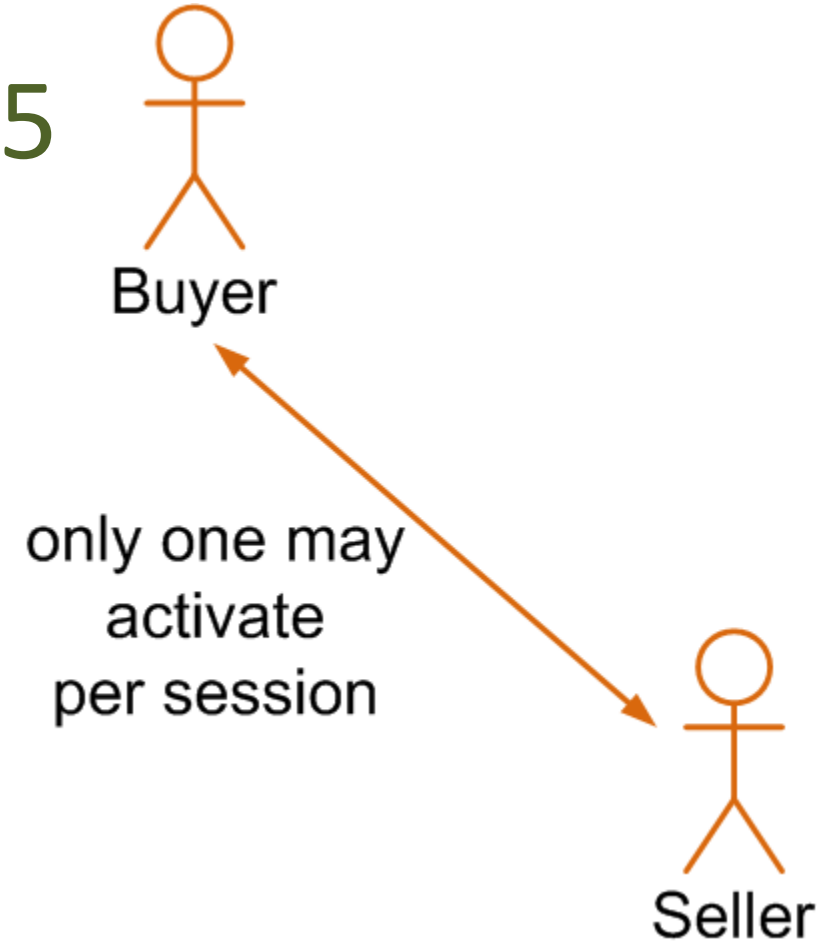
*All **Users** may **create** an **account** and **search items**.*



legend:
red:roles, **blue:objects**, **green:operations**

Use Case 5

A particular user may be a **Buyer**, or a **Seller**, but never both simultaneously.



legend:

red:roles, **blue:objects**, **green:operations**

Step 2

Create the Entity Lists

Step 2 - Create Lists

The use cases may now be converted into lists of entities. Although still human readable, the new format is aligned with RBAC.

1. List of Roles, inheritance relationships and Separation of Duty constraints.

2. List of Permissions: mappings between Objects and Operations

3. List of Perm Grants: mappings between roles and permissions

List of Role names

Roles pulled directly from use cases 2, 3, and 4.

	Name	Description
1	Users	Basic rights for all buyers and sellers
2	Buyers	May bid on and purchase items
3	Sellers	May setup auctions and ship items

Role Inheritance Relationships

Buyers and Sellers inherit from Users as described in use case #4.

	Child Name	Parent Name
1	Buyers	Users
2	Sellers	Users

Role Activation Constraints

Don't forget the role combination rule in use case #5

Set Name	Role Name	Type	Cardinality
BuySell	Buyers	Dynamic	2
	Sellers		

List of Permissions

Identified in use cases 2, 3 and 4.

	Obj Name	Oper Name	Description
1	Item	search	Search through list of items
2	Item	bid	Place a bid on a product
3	Item	purchase	Purchase a given product
4	Item	ship	Ships products after sale
5	Auction	create	May start a new auction
6	Account	create	Setup new accounts

List of Perm Grants

Also derived from the use cases.

	Role Name	Object Name	Operation Name
1	Users	Item	search
2	Users	Account	create
3	Buyers	Item	bid
4	Buyers	Item	purchase
5	Sellers	Item	ship
6	Sellers	Auction	create

symas

Step 3 - Load the Policy

- Now it is time to load your polices into the security system

Step 3 - Load the Policy

- Hand to the security team the entity lists.
- They convert to the particular syntax used by RBAC engine.
- Here we follow Fortress RBAC policy loading syntax.

Add Roles

```
<addrole>
  <role name="Users"
  description="…"/>
  <role name="Buyers"
  description="…"/>
  <role name="Sellers"
  description="…"/>
</addrole>
```

Add Role Inheritance

```
<addroleinheritance>
  <relationship
  child="Buyers" parent="Users"/>
  <relationship
  child="Sellers" parent="Users"/>
</addroleinheritance>
```

Add Dynamic SoD Constraint

```
<addsdset>
   <sdset name="BuySel"
    setmembers="Buyers,Sellers"
    cardinality="2"
    setType="DYNAMIC"
    description="only one role"/>
</addsdset>
```

Add Perm Objects

```
<addpermobj>
  <permobj objName="Item"
  description="…" ou="p1" />
  <permobj objName="Auction"
  description="…" ou="p1" />
  <permobj objName="Account"
  description="…" ou="p1" />
</addpermobj>
```

Add Perm Operations

```
<addpermop>
  <permop objName="Item" opName="bid"
   description="Item.Bid"/>
  <permop objName="Item" opName="purchase"
   description="Item.Purchase"/>
  <permop objName="Item" opName="ship"
   description="Item.Ship"/>
  <permop objName="Item" opName="search"
   description="Item.Search"/>
  <permop objName="Auction" opName="create"
   description="Auction.Create"/>
  <permop objName="Account" opName="create"
   description="Account.Create"/>
</addpermop>
```

Add Perm Grants

```
<addpermgrant>
  <permgrant roleNm="Buyers" objName="Item"
   opName="bid"/>
  <permgrant roleNm="Buyers" objName="Item"
   opName="purchase"/>
  <permgrant roleNm="Sellers" objName="Item"
   opName="ship"/>
  <permgrant roleNm="Sellers" objName="Auction"
   opName="create"/>
  <permgrant roleNm="Users" objName="Item"
   opName="search"/>
  <permgrant roleNm="Users" objName="Account"
   opName="create"/>
</addpermgrant>
```

symas

Load the Policy

Combine into a single policy file.

[Role Engineering Sample Policy File]

```
<FortressAdmin>
  <addrole>
    <role name="Role_Users" description="Base Role for all Buyers and Sellers"/>
    <role name="Role_Buyers" description="May bid on and purchase products"/>
    <role name="Role_Sellers" description="May start auctions and ship items"/>
  </addrole>
  <addroleinheritance>
    <relationship child="Role_Buyers" parent="Role_Users"/>
    <relationship child="Role_Sellers" parent="Role_Users"/>
  </addroleinheritance>
  <addpermgrant>
    <permgrant objName="SellersPage" opName="link" roleNm="Role_Sellers"/>
    <permgrant objName="BuyersPage" opName="link" roleNm="Role_Buyers"/>
    <permgrant objName="Item" opName="bid" roleNm="Role_Buyers"/>
    <permgrant objName="Item" opName="buy" roleNm="Role_Buyers"/>
    <permgrant objName="Item" opName="ship" roleNm="Role_Sellers"/>
    <permgrant objName="Auction" opName="create" roleNm="Role_Sellers"/>
    <permgrant objName="Item" opName="search" roleNm="Role_Users"/>
    <permgrant objName="Account" opName="create" roleNm="Role_Users"/>
  </addpermgrant>
  <addsdset>
    <sdset name="BuySel" setmembers="Role_Buyers,Role_Sellers" cardinality="2" setType="DYNAMIC" description="..."/>
  </addsdset>
  <addpermobj>
    <permobj objName="Item" description="…" ou="p1" />
    <permobj objName="Auction" description="…" ou="p1" />
    <permobj objName="Account" description="…" ou="p1" />
  </addpermobj>
  <addpermop>
    <permop objName="Item" opName="bid" description="Bid on a given product"/>
    <permop objName="Item" opName="buy" description="Purchase a given product"/>
    <permop objName="Item" opName="ship" description="Place a product up for sale"/>
    <permop objName="Item" opName="search" description="Search through item list"/>
    <permop objName="Auction" opName="create" description="May start a new auction"/>
    <permop objName="Account" opName="create" description="Ability to add a new account"/>
  </addpermop>
  …
</FortressAdmin>
```

symas

Load the Policy

Which is then fed into the RBAC engine as a script

```
INFO  - FortressAntTask          - FORTRESS ANT TASK NAME : FortressAdmin
INFO  - FortressAntTask          - addOrgunits name=u1 typeName=USER description=Test User Org for Rbac Role Engineering
INFO  - FortressAntTask          - addOrgunits name=p1 typeName=PERM description=Test Perm Org for Rbac Role Engineering
INFO  - FortressAntTask          - addRoles name=Role_Users description=Basic rights for all Buyers and Sellers
INFO  - FortressAntTask          - addRoles name=Role_Buyers description=May bid on and purchase products
INFO  - FortressAntTask          - addRoles name=Role_Sellers description=May start auctions and ship items
INFO  - FortressAntTask          - addRoles name=Super_Users description=May switch between buyers and sellers
INFO  - FortressAntTask          - addRoleInheritances parent=Role_Users child=Role_Buyers
INFO  - RoleUtil                 - loadGraph initializing ROLE context [HOME]
INFO  - FortressAntTask          - addRoleInheritances parent=Role_Users child=Role_Sellers
INFO  - FortressAntTask          - addSdsets name=BuySel description=User can only be activate one role of this set
INFO  - FortressAntTask          - addPermObjs objName=org.rolesample.HomePage description=Role Engineering Sample Home P
INFO  - FortressAntTask          - addPermObjs objName=SellersPage description=Used by Sellers orgUnit=p1 type=null
INFO  - FortressAntTask          - addPermObjs objName=BuyersPage description=Used by Buyers orgUnit=p1 type=null
INFO  - FortressAntTask          - addPermObjs objName=Item description=This product is available for purchase orgUnit=p1
INFO  - FortressAntTask          - addPermObjs objName=Auction description=Controls a particular online auction orgUnit=p
INFO  - FortressAntTask          - addPermObjs objName=Account description=Each user must have one of these orgUnit=p1 ty
INFO  - FortressAntTask          - addPermOps name=switchToSeller objName=org.rolesample.HomePage
INFO  - FortressAntTask          - addPermOps name=switchToBuyer objName=org.rolesample.HomePage
INFO  - FortressAntTask          - addPermOps name=switchRoles objName=org.rolesample.HomePage
INFO  - FortressAntTask          - addPermOps name=link objName=SellersPage
INFO  - FortressAntTask          - addPermOps name=link objName=BuyersPage
INFO  - FortressAntTask          - addPermOps name=bid objName=Item
INFO  - FortressAntTask          - addPermOps name=buy objName=Item
INFO  - FortressAntTask          - addPermOps name=ship objName=Item
INFO  - FortressAntTask          - addPermOps name=search objName=Item
INFO  - FortressAntTask          - addPermOps name=create objName=Auction
INFO  - FortressAntTask          - addPermOps name=create objName=Account
```

Step 4 – Add RBAC Policy Enforcement Point (PEP)

Step 4 – Add RBAC PEP

- Use RBAC System APIs for checking security.
- Apply RBAC semantics while at same time insulating from actual implementation.
- Favor declarative checking over programmatic.

Step 4 - Add RBAC PEP

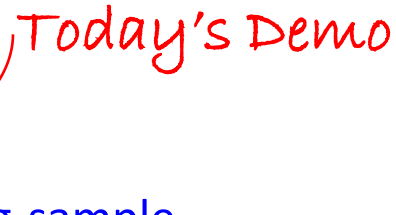
Somewhere in app is code that looks like this:

```
Session s = getRbacSession();
Permission p = new Permission(
  "Item", "bid" );
return accessMgr.checkAccess(s,p);
```

Step 4 – Add RBAC PEP

Code Examples:

1. Apache Fortress End-to-End Security Tutorial
   - https://github.com/shawnmckinney/apache-fortress-demo
2. Apache Fortress Wicket Sample
   - https://github.com/shawnmckinney/wicket-sample
3. Role Engineering Sample
   - https://github.com/shawnmckinney/role-engineering-sample
4. Apache Fortress Saml Demo
   - https://github.com/shawnmckinney/fortress-saml-demo



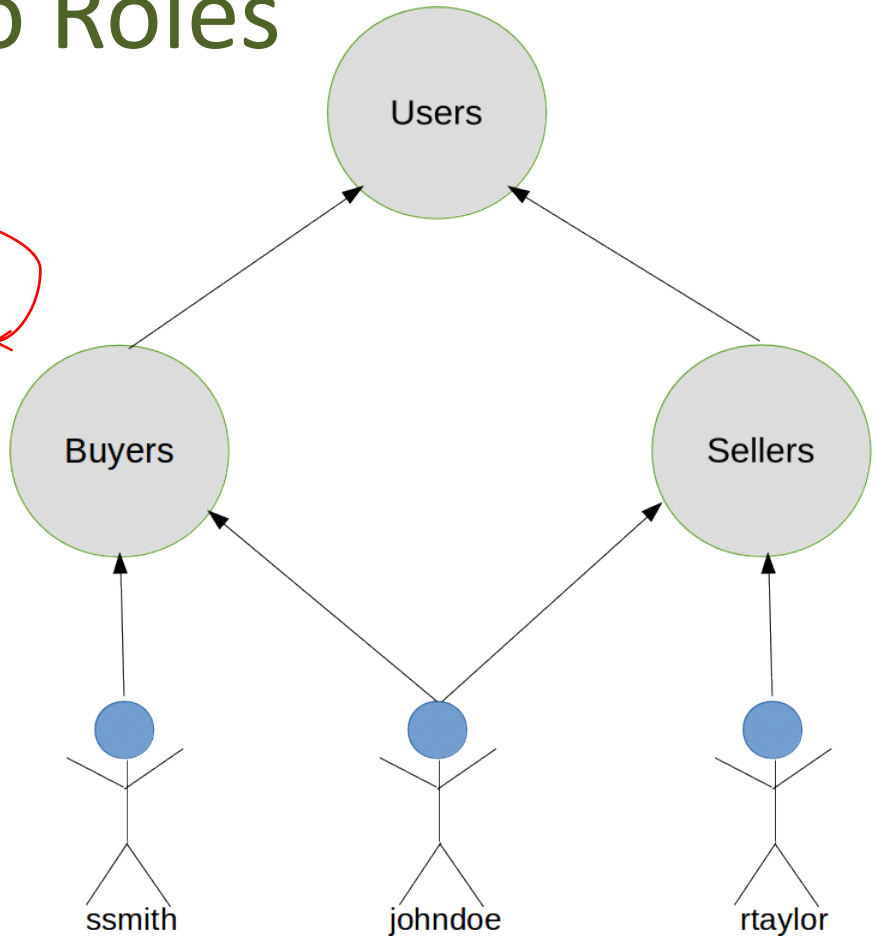
Step 5 – Assign Users to Roles

Step 5 - Assign Users to Roles

Github link to

[Role Engineering Sample Policy File]

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



symas

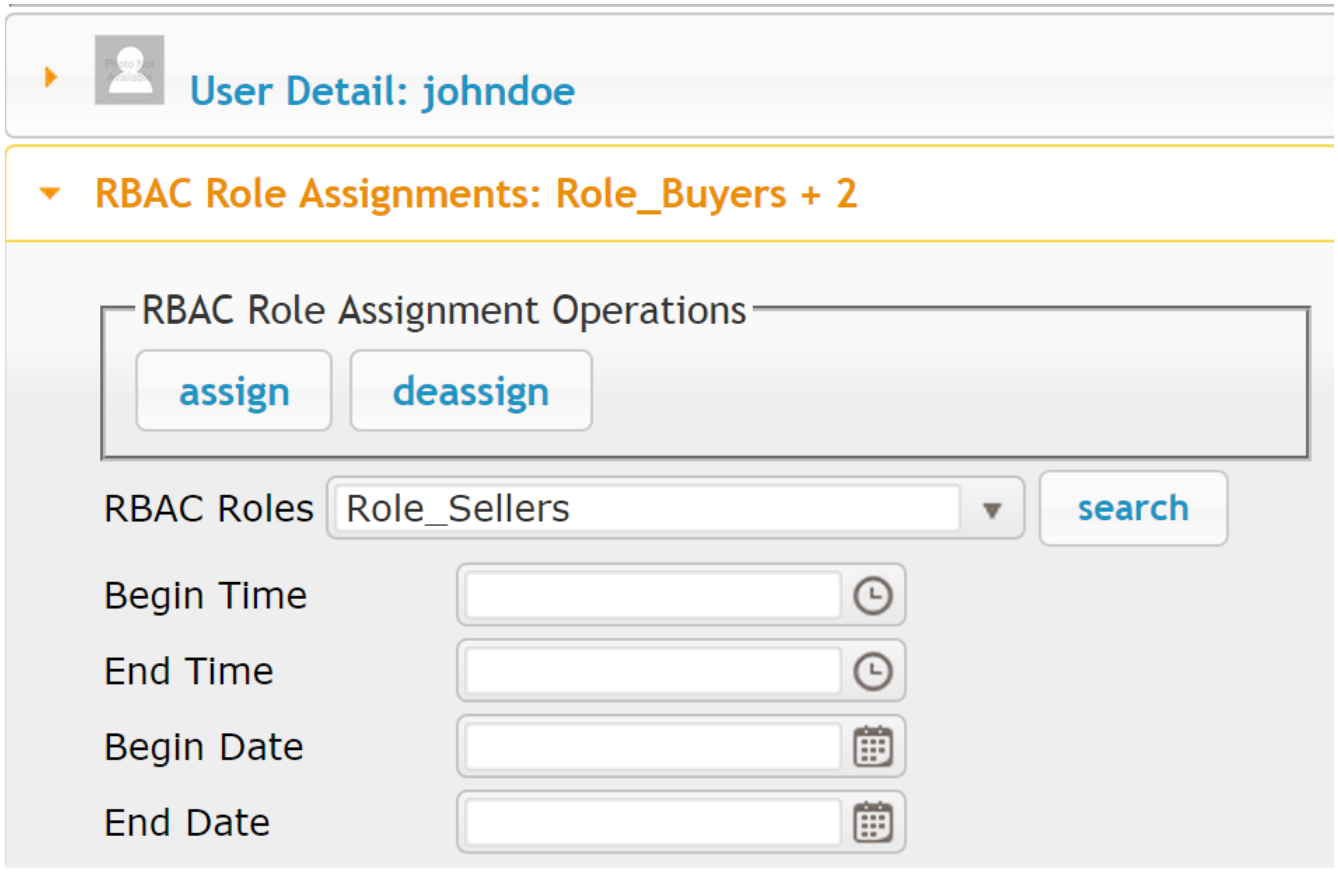
Step 5 – Assign Users to Roles

Sample user to role assignment policy file:

```
<adduserrole>
  <userrole userId="johndoe" name="Buyers"/>
  <userrole userId="johndoe" name="Sellers"/>
  <userrole userId="ssmith" name="Buyers"/>
  <userrole userId="rtaylor" name="Sellers"/>
</adduserrole>
```

Step 5 Assign Users to Roles

User to Role assignments with GUI



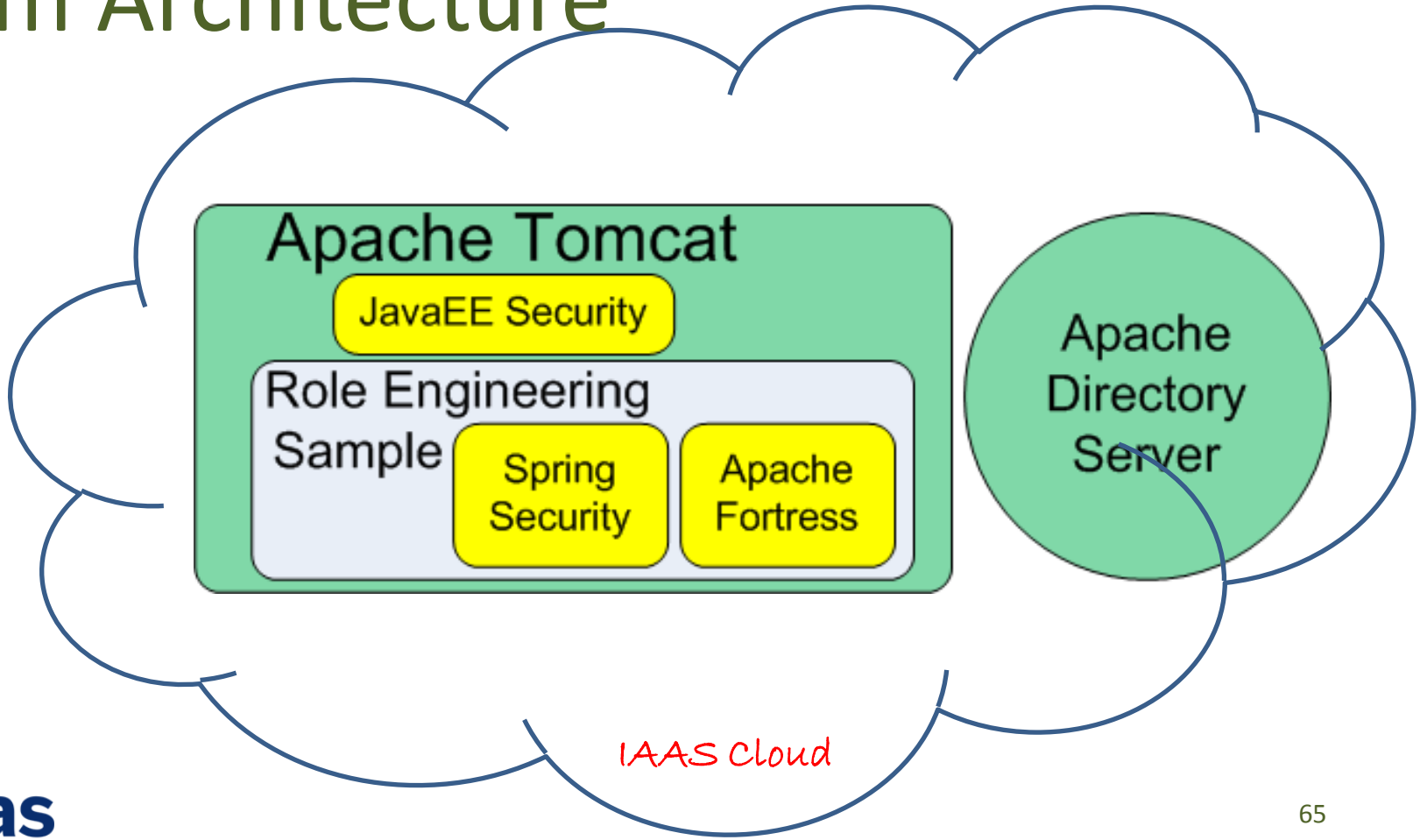
The Five Steps Summarized

1. Define Security Use Cases
2. Derive Lists RBAC Entities
3. Convert and Load into RBAC Engine
4. Add Security Checks to Apps
5. Assign Users to Roles

Demo


his is the first time you've seen this Stop error screen,
art your computer. If this screen appears again, follow
e steps:
to make sure any new hardware or software is properly installed.
his is a new installation, ask your hardware or software manufactu
any windows updates you might need.
roblems continue, disable or remove any newly
lled hardware
oftware. Disable BIOS memory options such as
g or shadowing.
ou need to use Safe Mode to remov
disable
ments, restart
computer, press F8 to select Ad
Startup
ons, and then
ct Safe Mode.
nical information:
STOP: 0x00000050 (0xFD3094C2,0x00000
000)
SPCMDCON.SYS - Address FBFE7617 base at F
eStamp 3d6d

System Architecture



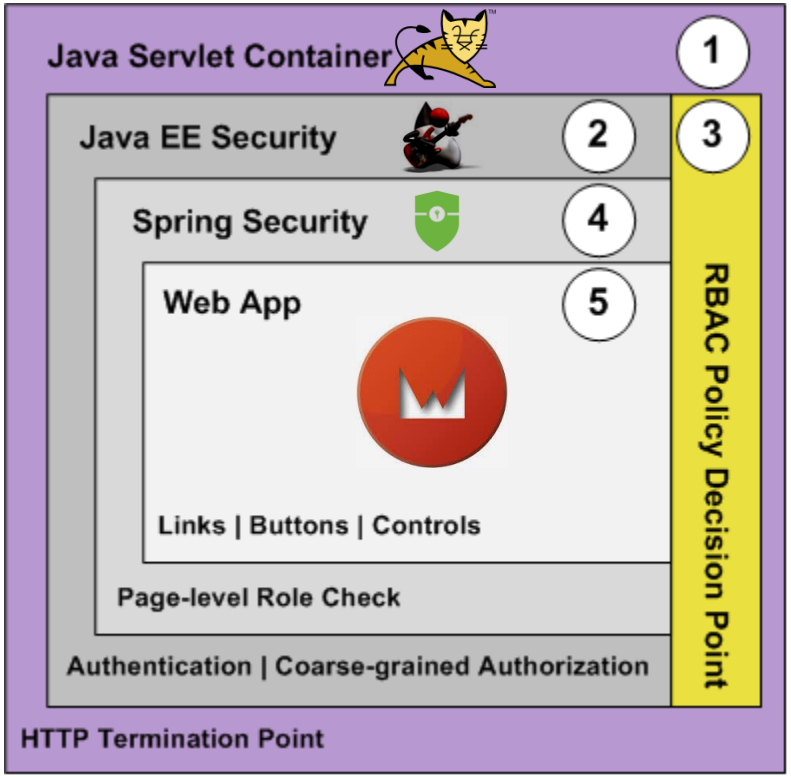
symas

Role Engineering Sample Security

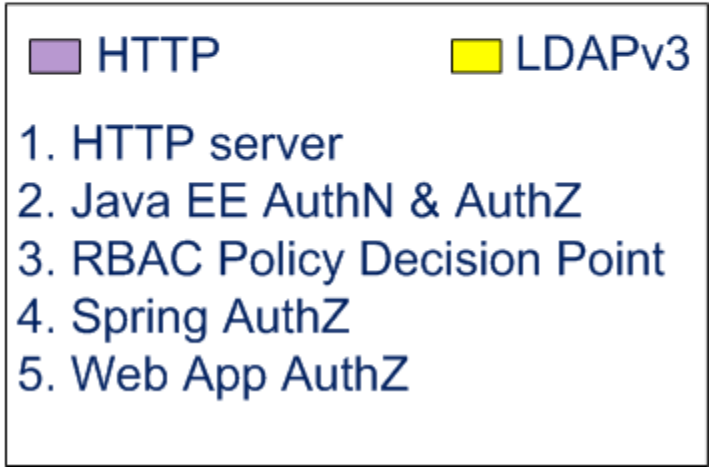
1. Java EE Authentication and Authorization
2. Spring Page-level Authorization
3. RBAC Permission Checks
   - Links
   - Buttons
4. Other RBAC Controls
   - Dynamic Separation of Duty
   - Role Switcher

Declarative

Role Engineering Sample



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



Role Engineering Sample

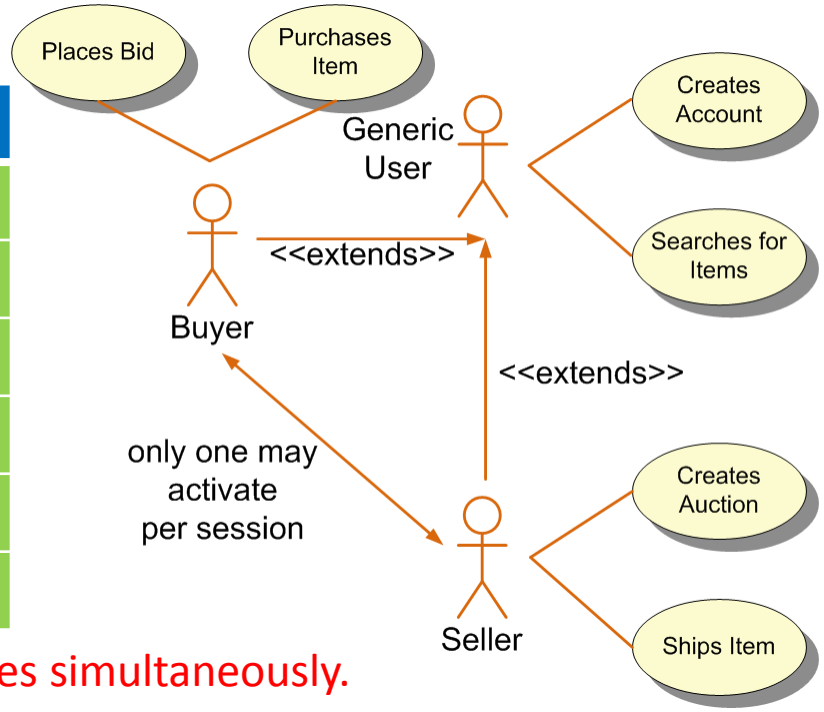
- Two pages
- Each has buttons controlled by RBAC Permissions.
- One Role per page.

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

		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.

RBAC Extra Credit

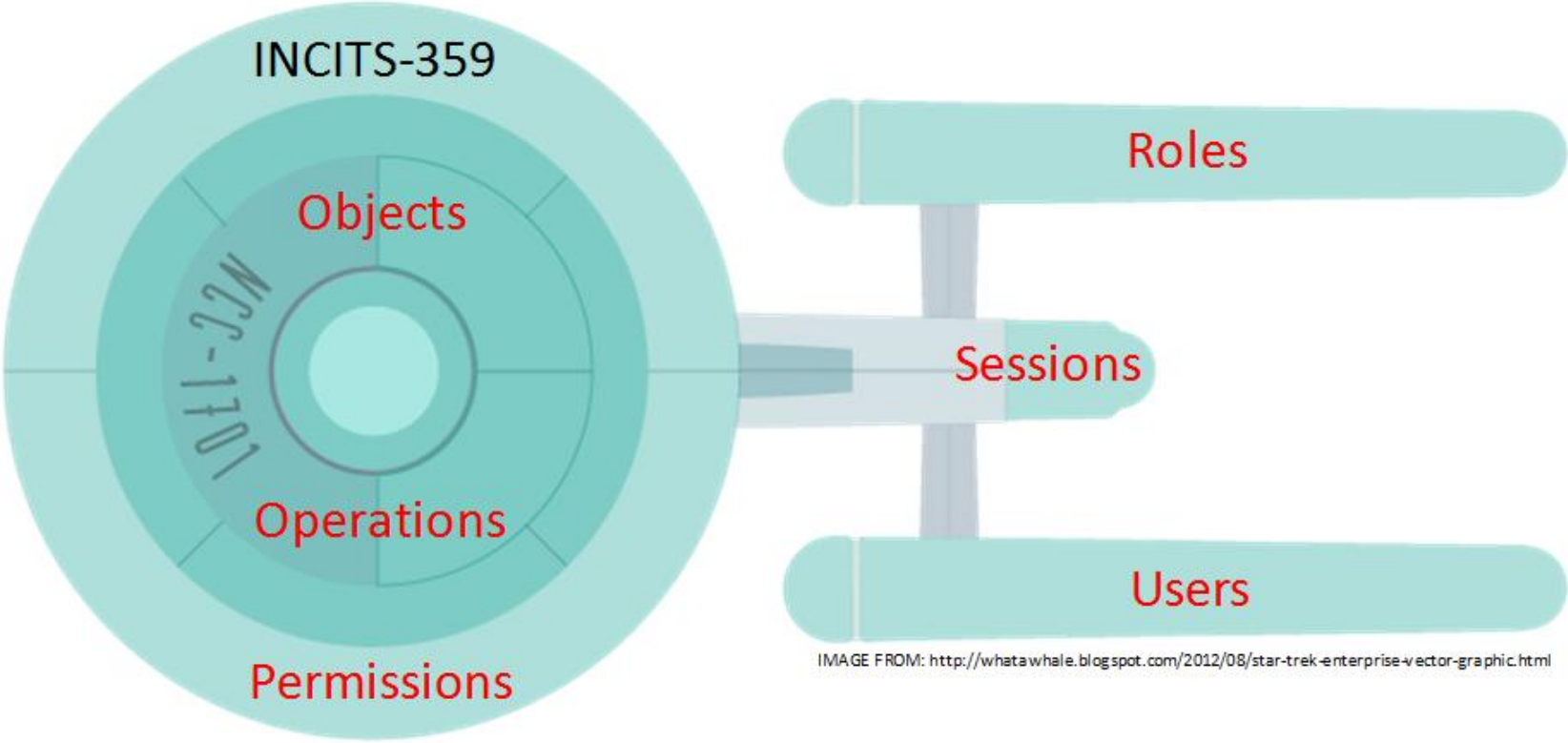


IMAGE FROM: http://whatawhale.blogspot.com/2012/08/star-trek-enterprise-vector-graphic.html

RBAC Extra Credit # 1

Question:

How do we get more attributes into the access control decision?

RBAC Extra Credit # 1

Answer:

Use ANSI INCITS 494 – allows attribute modifiers.

http://csrc.nist.gov/groups/SNS/rbac/documents/coyne-weil-13.pdf

RBAC Extra Credit # 2

Question:

What happens when a user with *conflicting roles* tries to log onto the RBAC system?

RBAC Extra Credit # 2

One Role will be discarded due to Dynamic SoD violation.

```
Enter userId:
johndoe
Enter password:
password
2015-09-22 12:16:003 INFO  RoleUtil:339 - loadGraph initializing ROLE context [HOME]
2015-09-22 12:16:003 WARN  DSDChecker:118 - validate userId [johndoe] failed activation of assignedRole
 [Role_Sellers] validates DSD Set Name:BuySel Cardinality:2
Session created successfully for userId [johndoe]
session [Session object:
    sessionId :288865bd-1be9-4d59-b6de-5084edfa8fa2
    user :User{userId='johndoe', internalId='7ed03656-c754-4db8-901f-94684f5f82ab', roles=[Role_Buyers,
 Super_Users], adminRoles=[], pwPolicy='null', cn='Jon Doe', sn='Doe', dn='uid=johndoe,ou=People,dc=exa
mple,dc=com', ou='u1', description='User has both Buyer and Seller Roles Assigned', beginTime='null', e
ndTime='null', beginDate='null', endDate='null', beginLockDate='null', endLockDate='null', dayMask='nul
l', name='Jon Doe', employeeType='null', title='null', timeout=0, reset=false, locked=false, system=nul
l, props=org.apache.directory.fortress.core.model.Props@6f3eee2d, address=Address object:
, phones=null, mobiles=null, emails=null}
    isAuthenticated :true
    lastAccess :1442949363771
    timeout :0
    graceLogins :0
    expirationSeconds :0
    errorId :0
    message :null
    warnings : Warning object:
    id :2055
    name :Role_Sellers
    type :ROLE
    msg :validate userId [johndoe] failed activation of assignedRole [Role_Sellers] validates DSD Set N
ame:BuySel Cardinality:2
```

symas

RBAC Extra Credit # 3a

Question:

How do we prevent someone from being *assigned* multiple roles simultaneously?

RBAC Extra Credit # 3a

Answer:

Use Static SoD:

```
<addsdset>
 <sdset name="BuySel2"
  setmembers="Buyers,Sellers"
   cardinality="2"
   setType="STATIC"
   description="…"/>
</addsdset>
```

symas

RBAC Extra Credit # 3b

Question:

What happens when you try to assign the 2nd Role that has Static SoD constraint?

```
Enter userId
johndoe
Enter role name
role_sellers
2015-09-24 17:24:029 INFO  RoleUtil:339 - loadGraph initializing ROLE context [HOME]
2015-09-24 17:24:029 ERROR AdminMgrConsole:657 - assignUser caught SecurityException rc=5088, msg=validateSSD new role [role_sellers] validates SSD Set Name:Buy-Sel-Static Cardinality:2
org.apache.directory.fortress.core.SecurityException: validateSSD new role [role_sellers] validates SSD Set Name:Buy-Sel-Static Cardinality:2
```

SecurityException in Admin

RBAC Extra Credit # 4

Question:

Role-Role Static SoD is just too simple to work. It only works at the role level and the toxic relationships are always between permissions.

RBAC Extra Credit # 4

Short Answer:

INCITS 494-2012 RBAC Policy Enhanced, section 5.4.2.2 Permission-permission (Static), specifically deals with this use case.

RBAC Extra Credit # 4

Longer Answer:

You can work around this problem in INCITS 359 by using RBAC1 Hierarchical Roles in conjunction of RBAC2 & 3. Do not directly assign roles that have toxic permissions directly to Users.

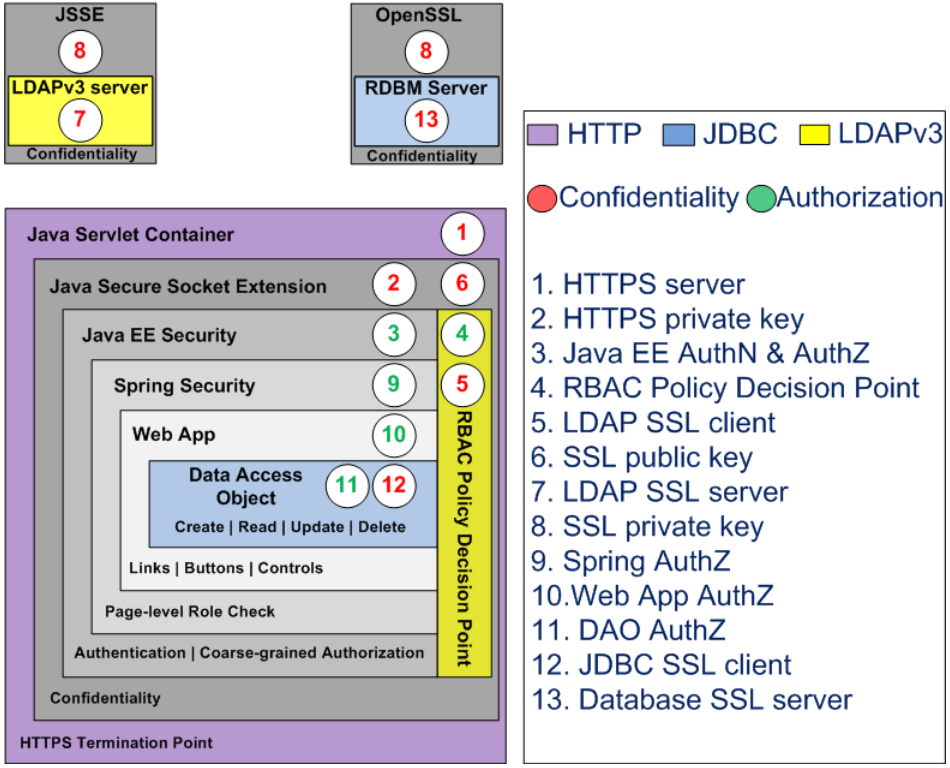
RBAC Extra Credit # 5

Question:

The session's use case was far too simple. Can we do something more realistic using RBAC?

RBAC Extra Credit # 5

Apache Fortress End-to-End Security Tutorial

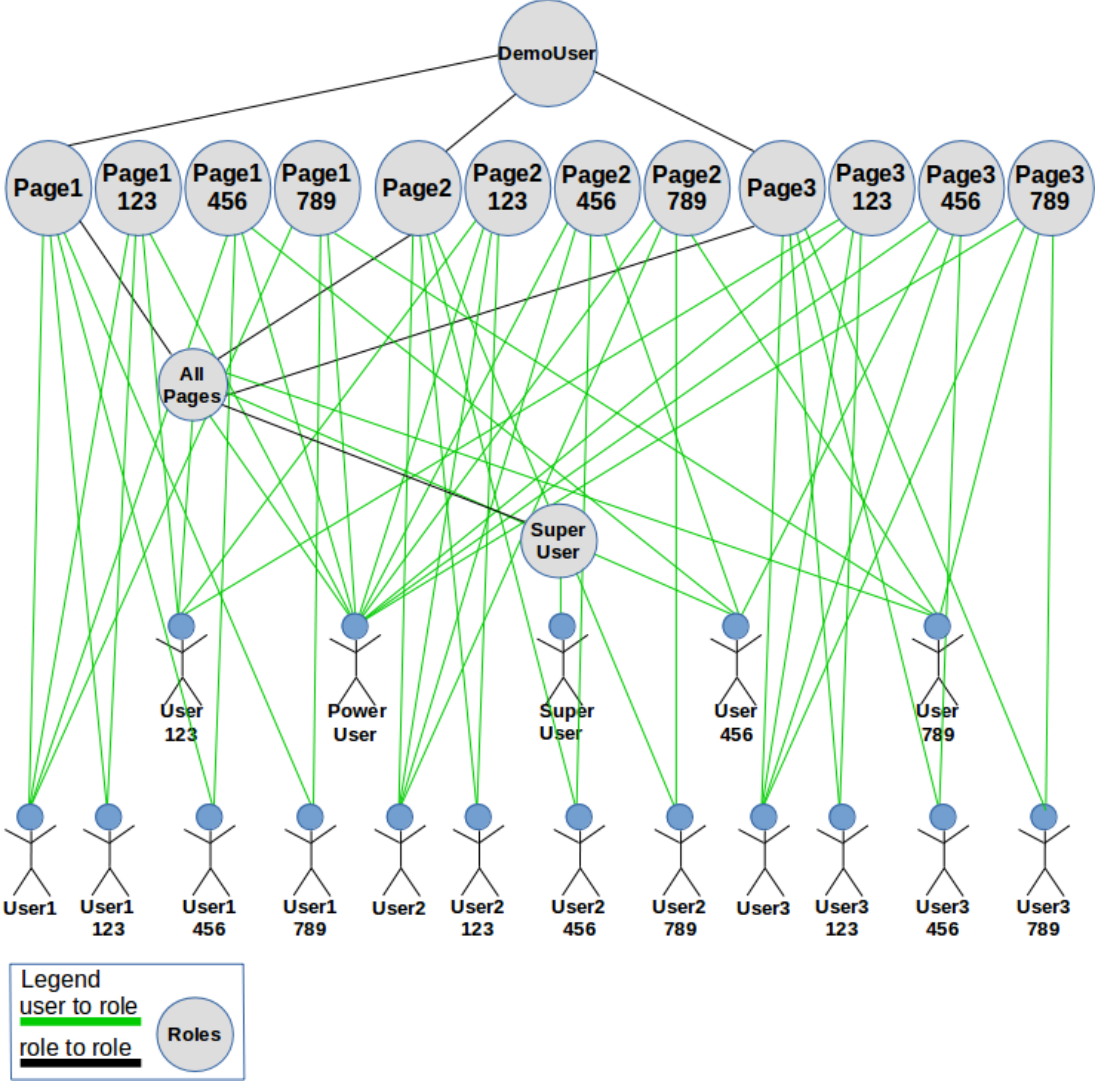


http://iamfortress.net/2015/02/16/apache-fortress-end-to-end-security-tutorial/

RBAC Extra Credit # 5

The End-to-End Security Tutorial delves into fine-grained access control of data tied to customers. It demonstrates how to implement a complicated set of requirements, verified with automated selenium test cases.

RBAC
Extra Credit
5



symas

RBAC Extra Credit # 6

Question:

To adequately model my organization's security using RBAC will cause role *explosion*. What am I to do about that?

RBAC Extra Credit # 6

Short Answer:

Fine, don’t use RBAC model, but those relationships still must reside *somewhere*.

Wrap-up

Closing Thoughts

- ✓ Using Roles for Access Control is not the same as Role-Based Access Control
- ✓ INCITS 359 is more than just access control APIs
- ✓ All need at least RBAC

Tutorial Links

In Gitub:

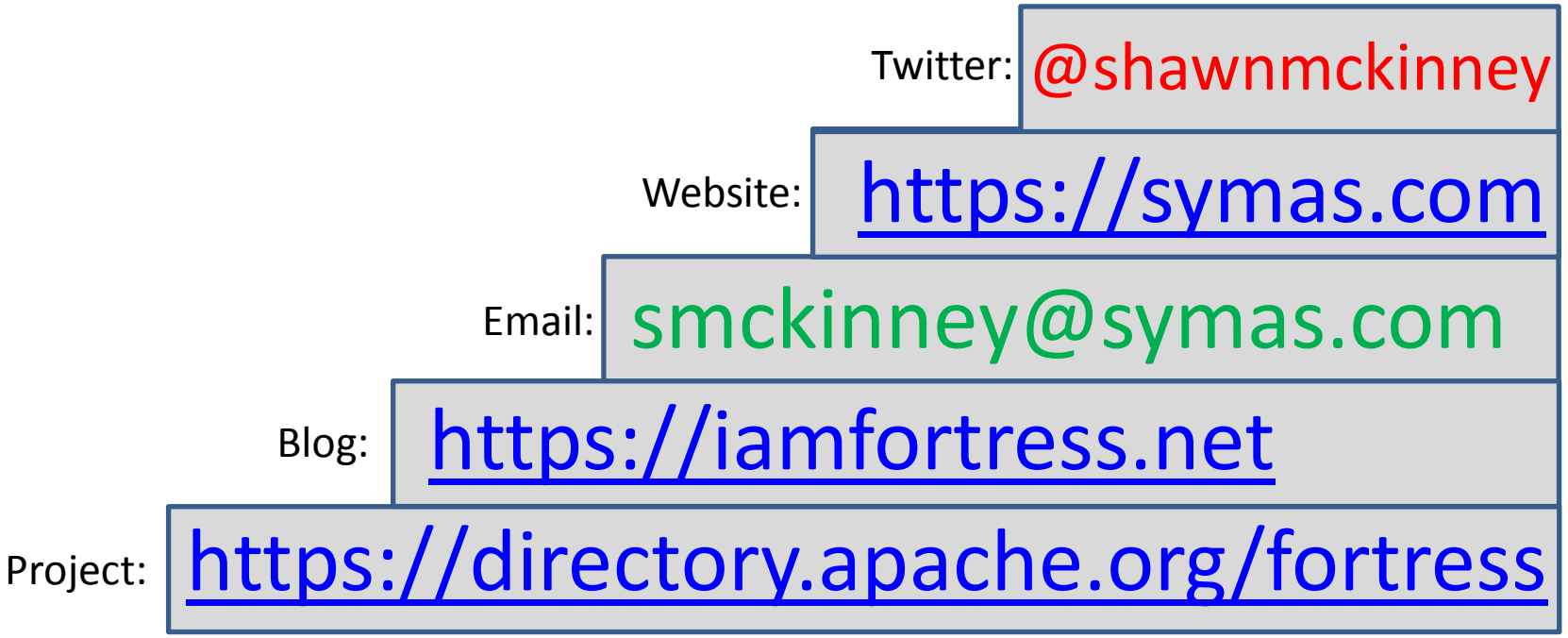
1. Role Engineering Sample:
   - https://github.com/shawnmckinney/role-engineering-sample

2. Apache Fortress End-to-End Security Demo:
   - https://github.com/shawnmckinney/apache-fortress-demo

Related Sessions

- CON3568 - Federated RBAC: Fortress, OAuth2 (Oltu), JWT, Java EE, and JASPIC
  - October 27, 11:00 am - 12:00 pm | Hilton—Plaza Room B
- CON2323 - The Anatomy of a Secure Web Application Using Java Redux
  - October 28, 3:00 pm - 4:00 pm | Hilton—Plaza Room A
- CON2325 - RBAC-Enable Your Java Web Applications with Apache Directory Fortress
  - October 29, 1:00 pm - 2:00 pm | Hilton—Plaza Room A

Contact Me


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