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™ 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
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

INCITS-359

Objects

Operations

Permissions

Roles

Sessions

Users

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)
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

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);
}

Fortress Admin APIs map to the INCITS 359 specs

GrantPermission(object, operation, role: NAME) ≤
(operation, object) ∈ PERMS; role ∈ ROLES
PA' = PA ∪ \{(operation, object) → role\}
assigned_permissions' = assigned_permissions \{role → assigned_permissions(roles)} ∪ 
{role → (assigned_permissions(role) ∪ \{(operation, object)\})} ≥

Link to AdminMgr javadoc

Link to INCITS 359 spec

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

    UserPermissions(user: NAME; result: \(2^{PERMS}\) \(\leq\) user \(\in\) USERS

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

    Link to ReviewMgr javadoc
    Fortress Review APIs map to the INCITS 359 specs
    http://git-wip-us.apache.org/repos/asf/directory-fortress-core.git
    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:

```java
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:

```java
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.
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:

<table>
<thead>
<tr>
<th>User Id</th>
<th>User Organization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>johndoe</td>
<td>u1</td>
<td>User has both Buyer and Seller Roles Assigned</td>
</tr>
<tr>
<td>rtaylor</td>
<td>u1</td>
<td>User has Seller Role Assigned</td>
</tr>
</tbody>
</table>

JavaOne, San Francisco 2015
Implementations of RBAC Admin and Review

1. Apache Fortress Rest

2. Apache Fortress Web
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
Usages for System – Check Access

Permission check for a particular User:

```java
boolean checkAccess(Session session,
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


OpenLDAP Fortress Accelerator

– ssh://git-master.openldap.org/~git/git/openldap-fortress-accelerator.git
The Five Steps

1. Use Cases
2. Create Lists
3. Load Policy
4. Add Checks
5. Assign Users
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**

1. Defines Security Use Cases
2. Creates Security Lists
3. Loads into RBAC Engine
4. Adds Permission Checks
5. Assigns Users to Roles
Process Participants Take 2

• Business Team
• Security Team
• Dev Team
• Human Resources

1. Defines Security Use Cases
2. Creates Security Lists
3. Loads into RBAC Engine
4. Adds Permission Checks
5. Assigns Users to Roles
Step 1
Define the Use Cases

- Places Bid
- Purchases Item
- Generic User
  - <extends>>
  - Buyer
    - only one may activate per session
  - <extends>>
  - Seller
    - Creates Account
    - Searches for Items
    - Creates Auction
    - Ships Item
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

Symas
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.*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Users</td>
<td>Basic rights for all buyers and sellers</td>
</tr>
<tr>
<td>2 Buyers</td>
<td>May bid on and purchase items</td>
</tr>
<tr>
<td>3 Sellers</td>
<td>May setup auctions and ship items</td>
</tr>
</tbody>
</table>
**Role Inheritance Relationships**

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

<table>
<thead>
<tr>
<th>Child Name</th>
<th>Parent Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buyers</td>
<td>Users</td>
</tr>
<tr>
<td>Sellers</td>
<td>Users</td>
</tr>
</tbody>
</table>
Role Activation Constraints

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

<table>
<thead>
<tr>
<th>Set Name</th>
<th>Role Name</th>
<th>Type</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuySell</td>
<td>Buyers</td>
<td>Dynamic</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Sellers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List of Permissions

Identified in use cases 2, 3 and 4.

<table>
<thead>
<tr>
<th>Obj Name</th>
<th>Oper Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>search</td>
<td>Search through list of items</td>
</tr>
<tr>
<td>Item</td>
<td>bid</td>
<td>Place a bid on a product</td>
</tr>
<tr>
<td>Item</td>
<td>purchase</td>
<td>Purchase a given product</td>
</tr>
<tr>
<td>Item</td>
<td>ship</td>
<td>Ships products after sale</td>
</tr>
<tr>
<td>Auction</td>
<td>create</td>
<td>May start a new auction</td>
</tr>
<tr>
<td>Account</td>
<td>create</td>
<td>Setup new accounts</td>
</tr>
</tbody>
</table>
List of Perm Grants

Also derived from the use cases.

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Object Name</th>
<th>Operation Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>Item</td>
<td>search</td>
</tr>
<tr>
<td>Users</td>
<td>Account</td>
<td>create</td>
</tr>
<tr>
<td>Buyers</td>
<td>Item</td>
<td>bid</td>
</tr>
<tr>
<td>Buyers</td>
<td>Item</td>
<td>purchase</td>
</tr>
<tr>
<td>Sellers</td>
<td>Item</td>
<td>ship</td>
</tr>
<tr>
<td>Sellers</td>
<td>Auction</td>
<td>create</td>
</tr>
</tbody>
</table>
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.
<addrole>
    <role name="Users" description="..."/>
    <role name="Buyers" description="..."/>
    <role name="Sellers" description="..."/>
</addrole>
Add Role Inheritance

```xml
<addroleinheritance>
  <relationship
    child="Buyers" parent="Users"/>
  <relationship
    child="Sellers" parent="Users"/>
</addroleinheritance>
```
Add Dynamic SoD Constraint

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

```xml
<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="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>
Load the Policy

Combine into a single policy file.

Role Engineering

Sample Policy File
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 - addSdssets name=BuySel description=User can only be active one role of this set
INFO - FortressAntTask - addPermObjs objName=org.rolesample.HomePage description=Role Engineering Sample Home Page
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=p1
INFO - FortressAntTask - addPermObjs objName=Account description=Each user must have one of these orgUnit=p1
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:

```java
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

JavaOne, San Francisco 2015
Step 5 – Assign Users to Roles
Step 5 - Assign Users to Roles

<table>
<thead>
<tr>
<th>User</th>
<th>Buyers</th>
<th>Sellers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssmith</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>johndoe</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>rtaylor</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Github link to Role Engineering Sample Policy File
Step 5 – Assign Users to Roles

Sample user to role assignment policy file:

```xml
<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

This is the first time you've seen this stop error screen. Restart your computer. If this screen appears again, follow these steps:

- First, make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer about any windows updates you might need.
- Second, problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. You need to use Safe Mode to remove or disable elements. Restart your computer, press F8 to select Advanced Startup options and then select Safe Mode.
- Technical information:
  
  STOP: 0x00000050 (0x00000304, 0x0000000000000000)

  SPMDCON.SYS - Address FDFE7617 base at FDFE7617, timestamp 3d68f

  SPMDCON.SYS - Address FDFE7617 base at FDFE7617, timestamp 3d68f
System Architecture

Apache Tomcat

- JavaEE Security

Role Engineering Sample

- Spring Security
- Apache Fortress

Apache Directory Server

IAAS Cloud
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

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

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

<table>
<thead>
<tr>
<th>User to Role</th>
<th>Buyers Page</th>
<th>Sellers Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssmith</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>jtaylor</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Johndoe*</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

* DSD constraint limits user from activating both roles simultaneously.
### Role Engineering Sample

<table>
<thead>
<tr>
<th>Permission</th>
<th>Buyer</th>
<th>Seller</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item.bid</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Item.purchase</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Item.ship</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Item.search</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Account.create</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Auction.create</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

* DSD constraint limits user from activating both roles simultaneously.

---

* DSD constraint limits user from activating both roles simultaneously.
RBAC Extra Credit

INCITS-359

Roles

Sessions

Users

Objects

Operations

Permissions

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.

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.
RBAC Extra Credit # 3a

Question:

How do we prevent someone from being assigned multiple roles simultaneously?
RBAC Extra Credit # 3a

Answer:

```xml
<addsdset>
  <sdset name="BuySel2"
    setmembers="Buyers,Sellers"
    cardinality="2"
    setType="STATIC"
    description="..."/>
</addsdset>
```
RBAC Extra Credit # 3b

Question:

What happens when you try to assign the 2\textsuperscript{nd} Role that has Static SoD constraint?
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.
Short Answer:

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

JavaOne, San Francisco 2015
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
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](https://github.com/shawnmckinney/role-engineering-sample)

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