Towards an Attribute-Based Role-Based Access Control System

Nov 5, 2019
Intro

Shawn McKinney

github/shawnmckinney

Code Monkey

symas Software Architect

Apache Directory PMC

OpenLDAP™ Engineering Team
Agenda

1. Discuss a bit on Access Control
2. Look at Apache Fortress RBAC Demo
3. “ “ “ ABAC Demo(s)
4. Next Steps
ANSI INCITS 359

Role-Based Access Control Standard
To Boldly Go

where no access control standard has been before

Spock, Kirk and McCoy

http://www.treknews.net/2015/09/08/star-trek-celebrates-49-years/
It's like déjà-vu all over again.

Yogi Berra
2011 - Heidelberg

• Pros and cons for using LDAP as backend for an RBAC system
  – Gietz, Widmer

• Open Source IAM using Fortress and OpenLDAP
  – McKinney
2013 - Paris

- Development of a standard LDAP Schema for RBAC
  - Gietz, Widmer, McKinney
- RBAC Accelerator
  - Hardin
- Fortress Open Source IAM on LDAPv3
  - McKinney
2015 - Edinburgh

• Introducing a Security Access Control Engine that resides in OpenLDAP
  – McKinney
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.

LDAPCon 2019, Sofia
ANSI RBAC INCITS 359 Specification

RBAC0:
  – Users, Roles, Perms, Sessions

RBAC1:
  – Hierarchical Roles

RBAC2:
  – Static Separation of Duties

RBAC3:
  – Dynamic Separation of Duties
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
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
Runtime processes
RBAC Functional Model

System Manager APIs:

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

Apache Fortress™

Access Management SDK and Web Components

A standards-based access management system, written in Java, supports ANSI INCITS 359 RBAC and more.

Web System Architecture

Option to use either HTTP or LDAPv3 protocol
Example 1

Apache Fortress Demo

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

<table>
<thead>
<tr>
<th>Pages</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page One</td>
<td>PAGE1_123</td>
<td>PAGE1_456</td>
<td>PAGE1_789</td>
</tr>
<tr>
<td>Page Two</td>
<td>PAGE2_123</td>
<td>PAGE2_456</td>
<td>PAGE2_789</td>
</tr>
<tr>
<td>Page Three</td>
<td>PAGE3_123</td>
<td>PAGE3_456</td>
<td>PAGE3_789</td>
</tr>
</tbody>
</table>

LDAPCon 2019, Sofia
<table>
<thead>
<tr>
<th>User123</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page2</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User1</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Page2</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User1_123</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page2</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
RBAC Demo

http://www.wright-brothers.org/Information_Desk/Help_with_Homework/Wright_Photos/Wright_Photos_images/1902_Glider_Flying.jpg
Apache Fortress Demo

- [https://github.com/shawnmckinney/apache-fortress-demo](https://github.com/shawnmckinney/apache-fortress-demo)

<table>
<thead>
<tr>
<th>User Foo</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Page2</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
Ruh Roh
Kaboom
Role Explosion

Cartesian Product

\[ A \times B = \{ (a, b) \mid a \in A \text{ and } b \in B \} \]

- A : role
- B : relationships
RBAC has also been criticized for leading to "role explosion," a problem in large enterprise systems which require access control of finer granularity than what RBAC can provide as roles are inherently assigned to operations and data types.

To support dynamic attributes, particularly in large organizations, a "role explosion" can result in thousands of separate roles being fashioned for different collections of permissions. Recent interest in attribute-based access control (ABAC) suggests that attributes and rules could either replace RBAC or make it more simple and flexible.

IEEE Computer, vol. 43, no. 6 (June, 2010), pp. 79-81
Number of Roles = sizeof(A) * sizeof(B)

Roles (A)  Relationships (B)
Role1      Customer 123
Role2      Customer 456
Role3      Customer 789

Roles
1. Role1-123
2. Role1-456
3. Role1-789
4. Role2-123
5. Role2-456
6. Role2-789
7. Role3-123
8. Role3-456
9. Role3-789
Now What?

What is Attribute-Based Access Control (ABAC)

An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf
What is ABAC

Although the concept itself existed for many years, ABAC is considered a "next generation" authorization model because it provides dynamic, context-aware and risk-intelligent access control to resources allowing access control policies that include specific attributes from many different information systems...

https://en.wikipedia.org/wiki/Attribute-based_access_control
Examples of ABAC

• Extensible Access Control Markup Language (XACML)

• Next Generation Access Control standard [ANSI499]
Examples of ABAC

The AuthZForce project provides an Attribute-Based Access Control (ABAC) framework compliant with the OASIS XACML standard v3.0, that mostly consists of an authorization policy engine and a RESTful authorization server. It was primarily developed to provide advanced access control for Web Services or APIs, but is generic enough to address all kinds of access control use cases.

https://authzforce.ow2.org
ABAC

Policy Enforcement Point (PEP)

Policy Decision Point (PDP)

Policy Administration Point (PAP)

Policy Repository

Policy Information Point (PIP)

Environment Conditions

Attribute Repository

Authorization Services

subject

object

https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf
Drawbacks

• Traction
• Complexity
• Performance
Attribute-Based Access Control

This approach might be more flexible than RBAC because it does not require separate roles for relevant sets of subject attributes, and rules can be implemented quickly to accommodate changing needs. The trade-off for this flexibility is the complexity of cases that must be considered: for \( n \) Boolean attributes or \( n \) conditions using attributes, there are \( 2^n \) possible combinations.
Figure 4: Enterprise ABAC Scenario Example

https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf
Let’s Have Another Look

Can RBAC be enhanced?
INCITS 494

Policy Enhanced RBAC
Two Phases of Activation

Attributes checked during two separate phases:

1. User-Role Activation
   - e.g., user may only activate the cashier role at store 314.

2. Role-Permission Activation
   - e.g., the action may only be performed on account 456789.
User-Role Activation

Examples:

• Apache Fortress Temporal Constraints
• Apache Fortress Dynamic Constraints (New)
Use User-Role Constraint

- Store the contextual information on the user entry’s role assignments.
- ftRC: teller@type@key@value
  - e.g. ftRC: teller@user@location@north

LDAPCon 2019, Sofia
Core RBAC

- USERS
- ROLES
- SESS-IONS
- Object
- PERMS
- OPERATION

Constraints

User Assignment
User Session
Session Roles

LDAPCon 2019, Sofia
+ ABAC Constraints

- USERS
  - Constraints (User Assignment)
  - User Session

- ROLES
  - Constraints
  - Session Roles

- SESS-IONS

- PERMS
  - Object
  - OPERT-ATION

LDAPCon 2019, Sofia
All Together Now

USERs

Role Hierarchy

SSD

Constraints

User Assignment

SESS-IONS

DSD

Session Roles

USERS

ROLES

Constraints

PERMS

Object

OPER-ATION

LDAPCon 2019, Sofia
RBAC w/ ABAC

• Opportunity to introduce arbitrary attributes into the Role activation phase.
• The Role is ‘special’ in that it will only be activated if conditions match.
Advantages

• Fixed the ‘Role explosion’ problem.
• We can continue to use our RBAC systems.
• Simpler to implement and maintain.
• No limit to the types of attributes.
Roles:
• Teller
• Coin Washer

Constraints:
• Location
e.g. User-Role-Constraint

• Curly
  - Coin Washer: North
  - Coin Washer: South
  - Teller: East

• Moe
  - Coin Washer: East
  - Coin Washer: South
  - Teller: North

• Larry
  - Coin Washer: North
  - Coin Washer: East
  - Teller: South
Number of Roles = sizeof(A) * sizeof(B)

Roles (A)
- Teller
- Washer

Relationships (B)
- North
- South
- East
- West

Roles
1. Teller-North
2. Teller-South
3. Teller-East
4. Teller-West
5. Washer-North
6. Washer-South
7. Washer-East
8. Washer-West

Just stop
Role Constraints

constraint role="Coin Washer"
  key="location"
constraint role="Teller"
  key="location"

User-Role Constraints

```xml
userId="Curly"
  role="Teller"
  key="location" value="East"

userId="Curly"
  role="Coin Washer"
  key="location" value="North"

userId="Curly"
  role="Coin Washer"
  key="location" value="South"
```
Under the Hood
RBAC w/ ABAC

LDAP - uid=curly,ou=People,dc=example,dc=com - slapd local - Apache Directory Studio

Help

- cn=default,ou= Policies, dc=ari
- dc=example, dc=com

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

<table>
<thead>
<tr>
<th>Attribute Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftSystem</td>
<td>FALSE</td>
</tr>
<tr>
<td>ftRC</td>
<td>washers$type$USER$locale$south$</td>
</tr>
<tr>
<td>ftRC</td>
<td>washers$type$USER$locale$north$</td>
</tr>
<tr>
<td>ftRC</td>
<td>tellers$type$USER$locale$east$</td>
</tr>
</tbody>
</table>
// Nothing new here:
User user = new User("curly");

// This is new:
RoleConstraint constraint = new RoleConstraint();

// In practice we're not gonna pass hard-coded key-values in here:
constraint.setKey("location");
constraint.setValue("north");

// This is just boilerplate goop:
List<RoleConstraint> constraints = new ArrayList();
constraints.add(constraint);
	ry {
  // Create the RBAC session with ABAC constraint -- location=north, asserted:
  Session session = accessMgr.createSession( user, constraints);
  ...
}
This is the first time you've seen this stop error screen. Start your computer. If this screen appears again, follow the steps:

1. To make sure any new hardware or software is properly installed, this is a new installation, ask your hardware or software manufacturer any Windows updates you might need.

2. 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 programs, restart computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

STOP: 0x00000050 (0xF03094C2, 0x0000000000000000)

SPCMDCON.SYS - Address FDFE7617 base at FDFE7600 timestamp 3d6d
Example
2
RBAC
ABAC
Sample

https://github.com/shawnmckinney/rbac-abac-sample
<table>
<thead>
<tr>
<th>User456</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Page2</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User2</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page2</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Page3</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User2_123</th>
<th>Customer 123</th>
<th>Customer 456</th>
<th>Customer 789</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page1</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Page2</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Page3</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
Example 3
Apache Fortress ABAC Demo

Java Servlet Container

Java EE Security

Spring Security

Web App

Links | Buttons | Controls

Page-level Role Check

Authentication | Coarse-grained Authorization

https://github.com/shawnmckinney/fortress-abac-demo
Next Steps

1. Dynamic Constraints Role-Permission
2. Dynamic Policies
Apache Fortress User-Role Validators

temporal.validator.0=Date
temporal.validator.1=LockDate
temporal.validator.2=Timeout
temporal.validator.3=ClockTime
temporal.validator.4=Day
temporal.validator.5= UserRoleConstraint

Since v 2.0.1
Apache Fortress Role-Perm Validators

Not implemented yet

permission.validator.0=Limit
permission.validator.1=Clearance
permission.validator.2=Domain
Closing Thoughts

Standards-based RBAC allows attributes into the mix.

• *Fine-grained Authorization*
https://directory.apache.org/fortress
Examples

1. github/shawnmckinney/apache-fortress-demo
2. github/shawnmckinney/rbac-abac-sample
3. github:/shawnmckinney/fortress-abac-demo
Contact Info

- @shawnmckinney
- http://symas.com
- smckinney@apache.org
- https://iamfortress.net
- https://directory.apache.org/fortress