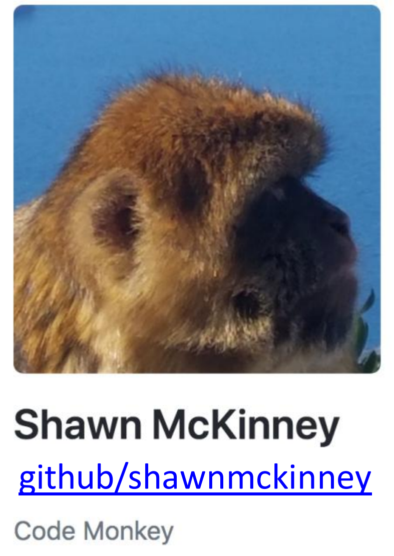
Towards an Attribute-Based Role-Based Access Control System

Nov 5, 2019

LDAPCon 2019

SOFIA

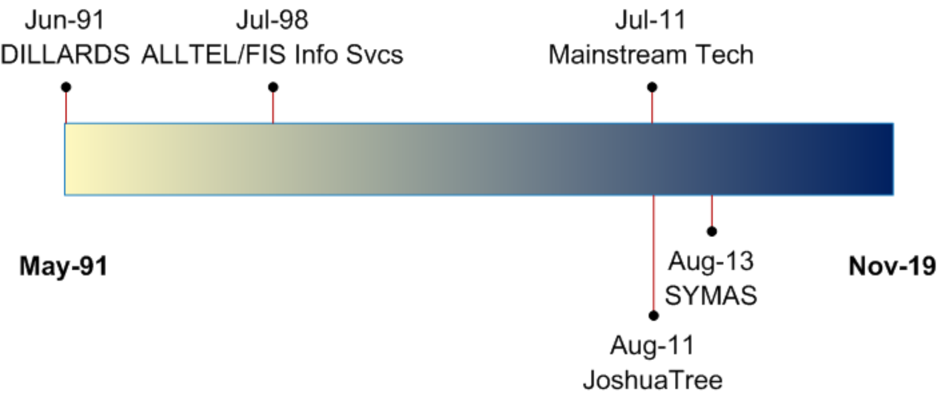
Intro



Shawn McKinney

github/shawnmckinney

Code Monkey



symas Software Architect

Apache Directory PMC

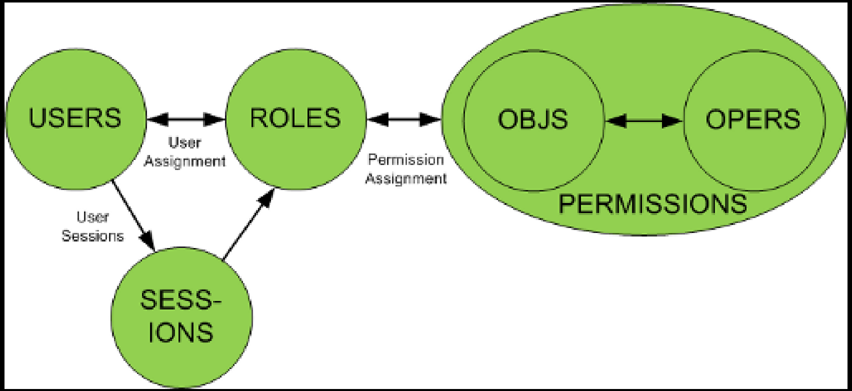
OpenLDAP™ PROJECT Engineering Team

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



Kuhn, Ferraiolo and Sandhu
https://www.facebook.com/ieeecomputersociety/posts

To Boldly Go

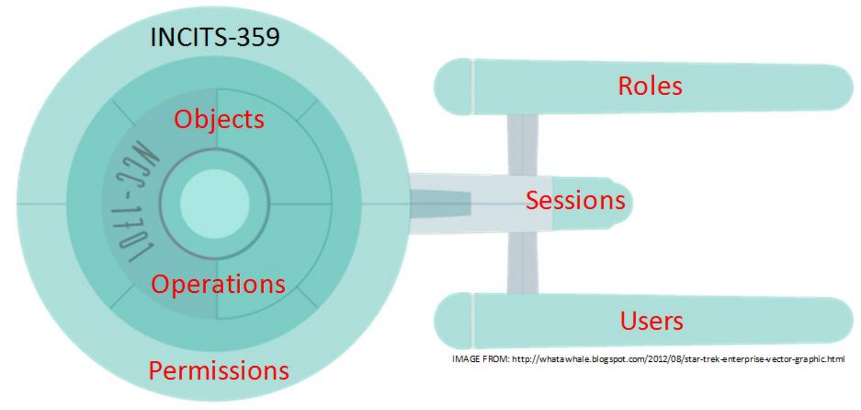
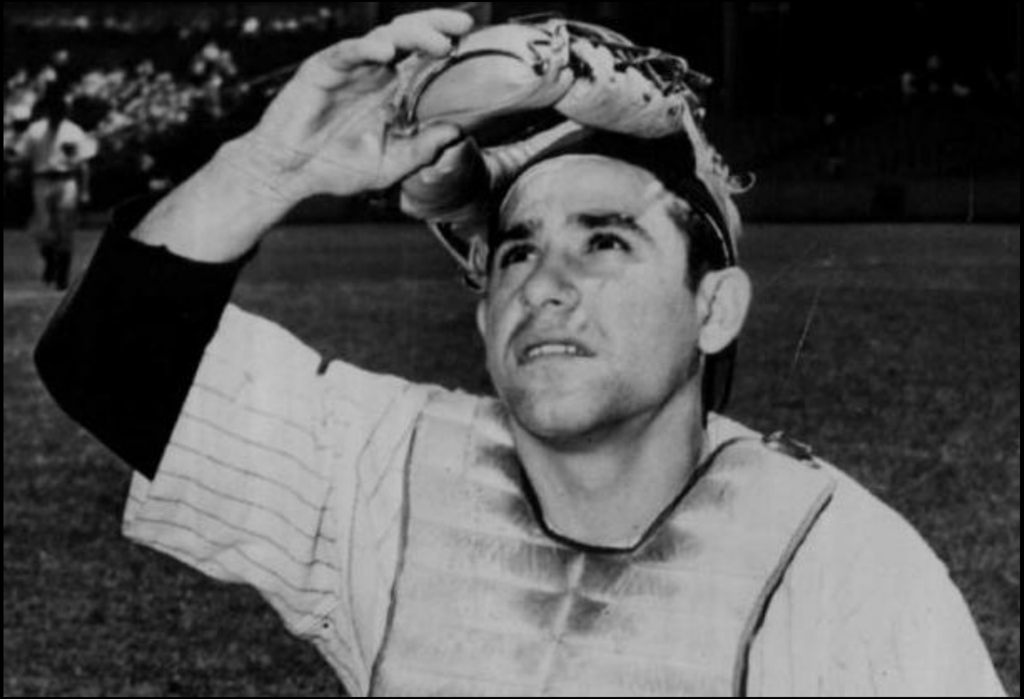


IMAGE FROM: http://whatawhale.blogspot.com/2012/08/star-trek-enterprise-vector-graphic.html

Spock, Kirk and McCoy
http://www.treknews.net/2015/09/08/star-trek-celebrates-49-years/

where no access control standard has been before

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.

ANSI RBAC INCITS 359 Specification

RBAC0:

- Users, Roles, Perms, Sessions

RBAC1:

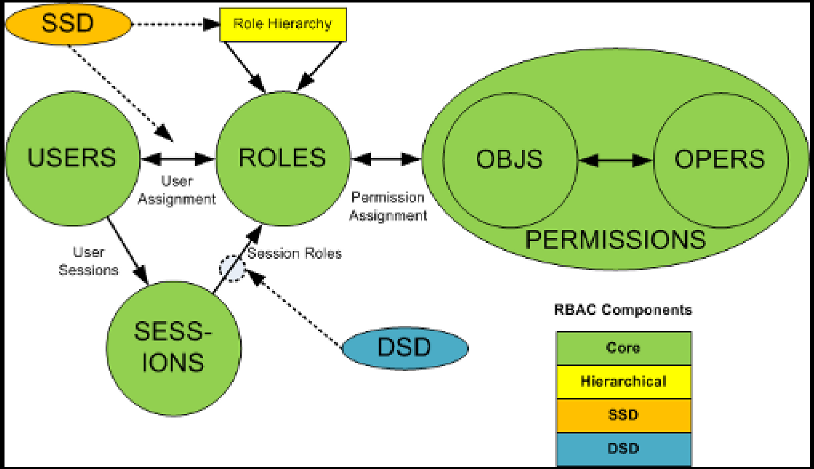
- Hierarchical Roles

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:

Management and Config processes

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

Runtime processes

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

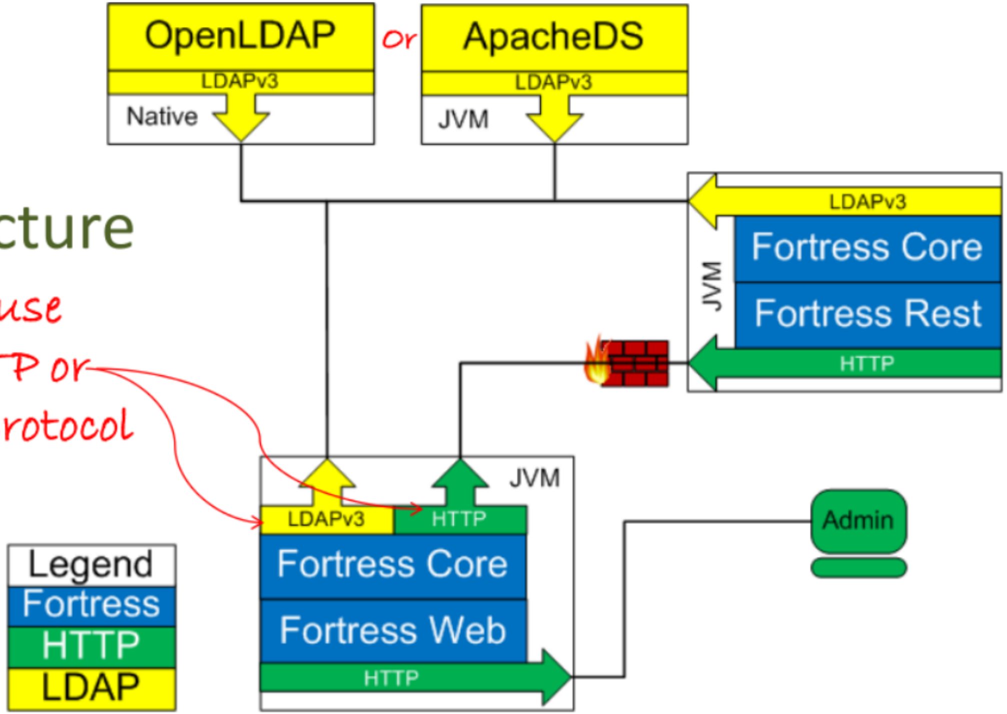
Apache Fortress™

https://directory.apache.org/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



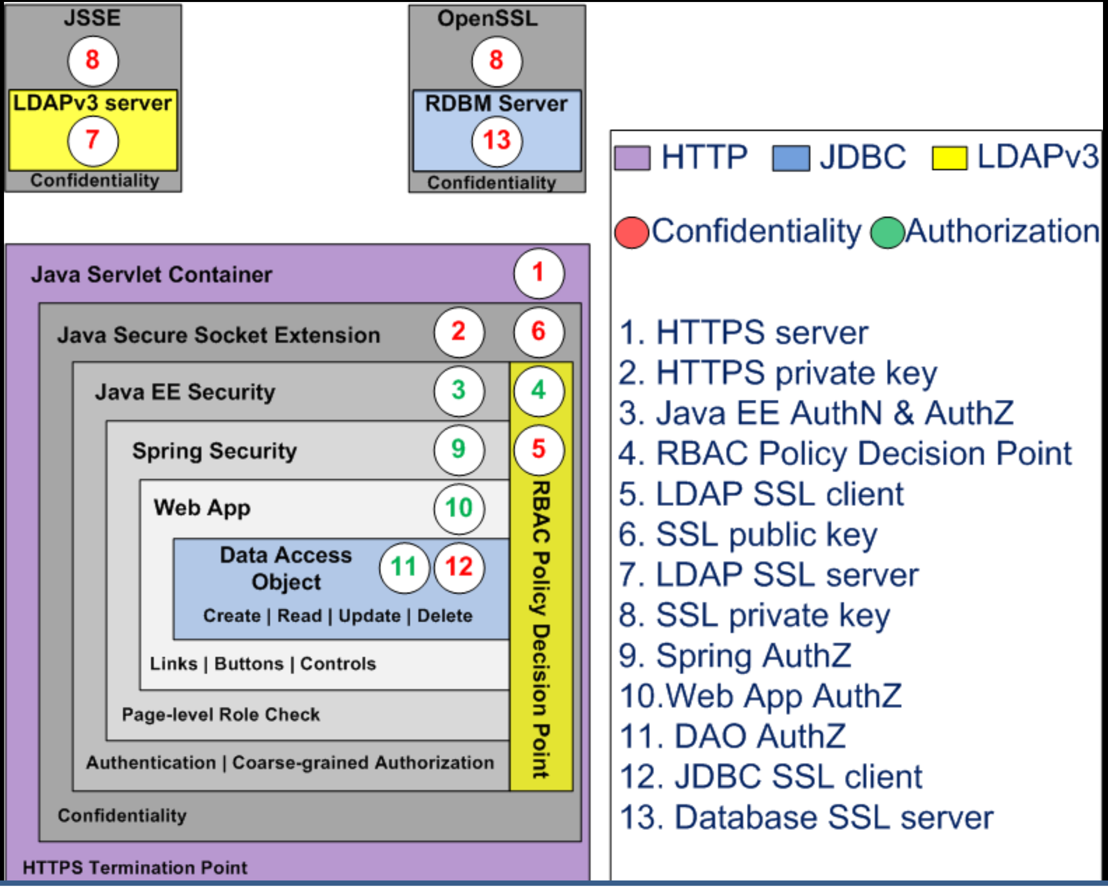
Option to use either HTTP or LDAPv3 protocol

Example 1

Apache

Fortress

Demo



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

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

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

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

RBAC Demo

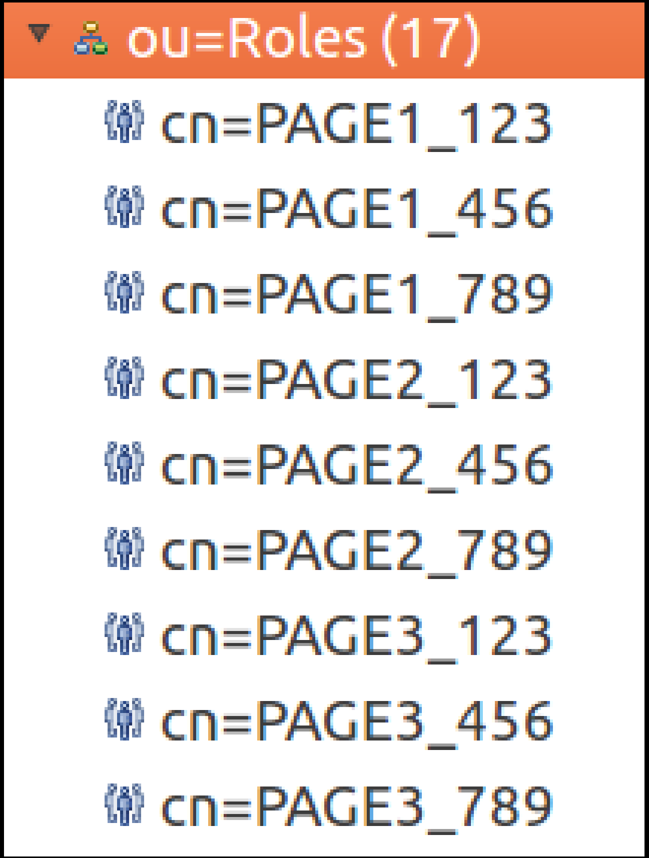
http://www.wright-brothers.org/Information_Desk/Help_with_Homework/Wright_Photos/Wright_Photos_images/1902_Glider_Flying.jpg

Apache Fortress Demo

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

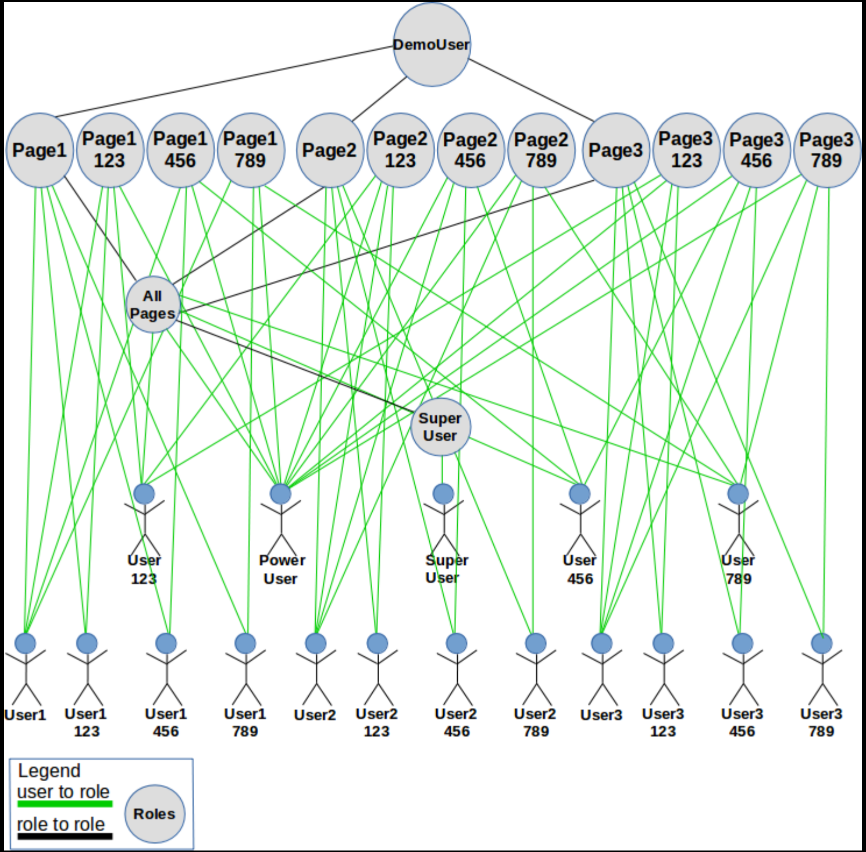
User Foo	Customer 123	Customer 456	Customer 789
Page1	False	True	True
Page2	True	False	False
Page3	True	False	False

Ruh Roh



```
▾ ou=Roles (17)
    cn=PAGE1_123
    cn=PAGE1_456
    cn=PAGE1_789
    cn=PAGE2_123
    cn=PAGE2_456
    cn=PAGE2_789
    cn=PAGE3_123
    cn=PAGE3_456
    cn=PAGE3_789
```

Kaboom

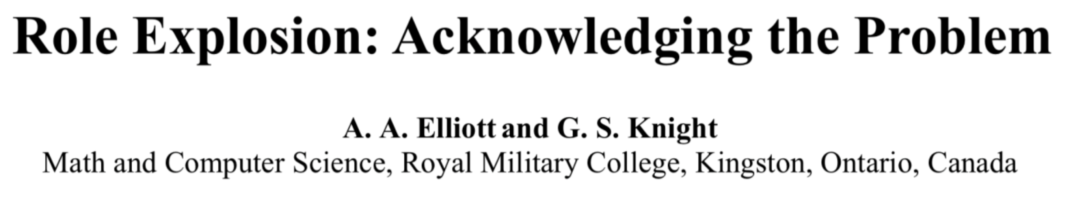

DemoUser
Page1
Page1 123
Page1 456
Page1 789
Page2
Page2 123
Page2 456
Page2 789
Page3
Page3 123
Page3 456
Page3 789
All Pages
Super User
User 123
Power User
Super User
User 456
User 789
User1
User1 123
User1 456
User1 789
User2
User2 123
User2 456
User2 789
User3
User3 123
User3 456
User3 789
Legend
user to role
role to role
Roles

Role Explosion

Cartesian Product

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

- A : role
- B : relationships



Role Explosion: Acknowledging the Problem

A. A. Elliott and G. S. Knight
Math and Computer Science, Royal Military College, Kingston, Ontario, Canada

https://pdfs.semanticscholar.org/143e/25f527eedecdf0a4f1b11646144fdfe694d5.pdf

Adding Attributes to Role-Based Access Control

D. Richard Kuhn, *National Institute of Standards and Technology*
Edward J. Coyne, *Science Applications International Corp.*
Timothy R. Weil, *Raytheon Polar Services Company*

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

RBAC has also been criticized for leading to role explosion,[12] 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.

wikipedia/Role-based_access_control

Number of Roles = sizeof(A) * sizeof(B)

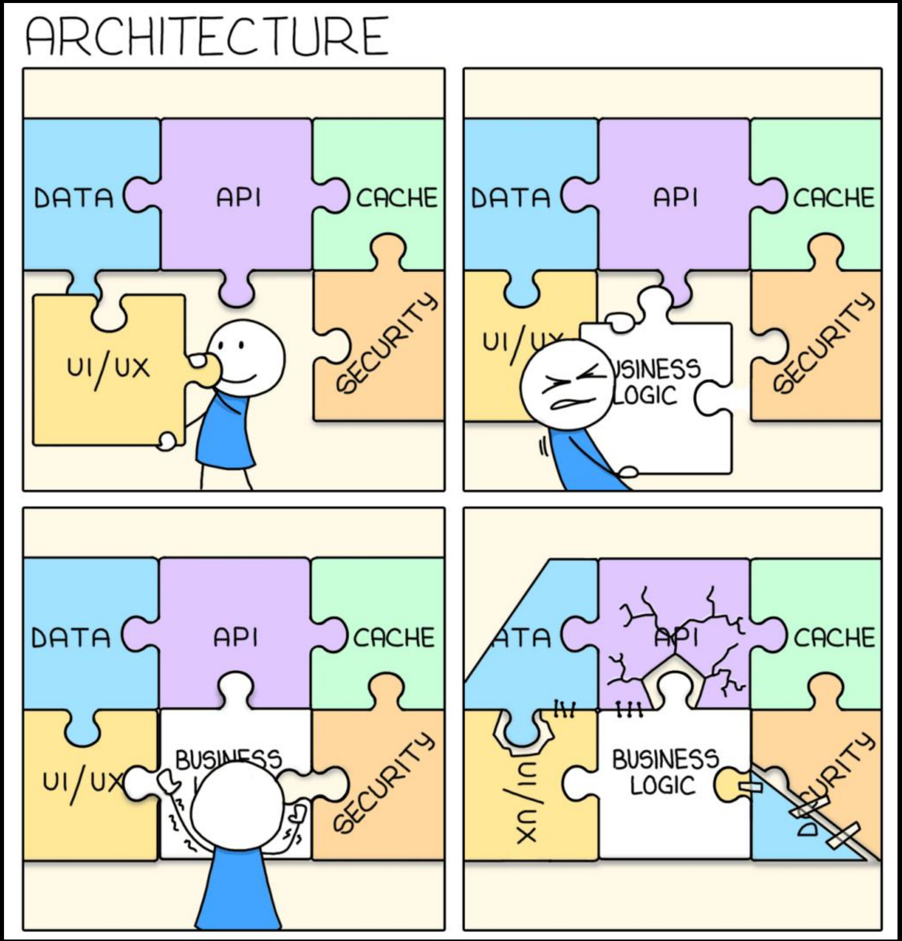
Roles (A)		Relationships (B)
Role1		Customer 123
Role2	*	Customer 456
Role3		Customer 789

=>

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?



https://www.monkeyuser.com/2018/architecture/

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

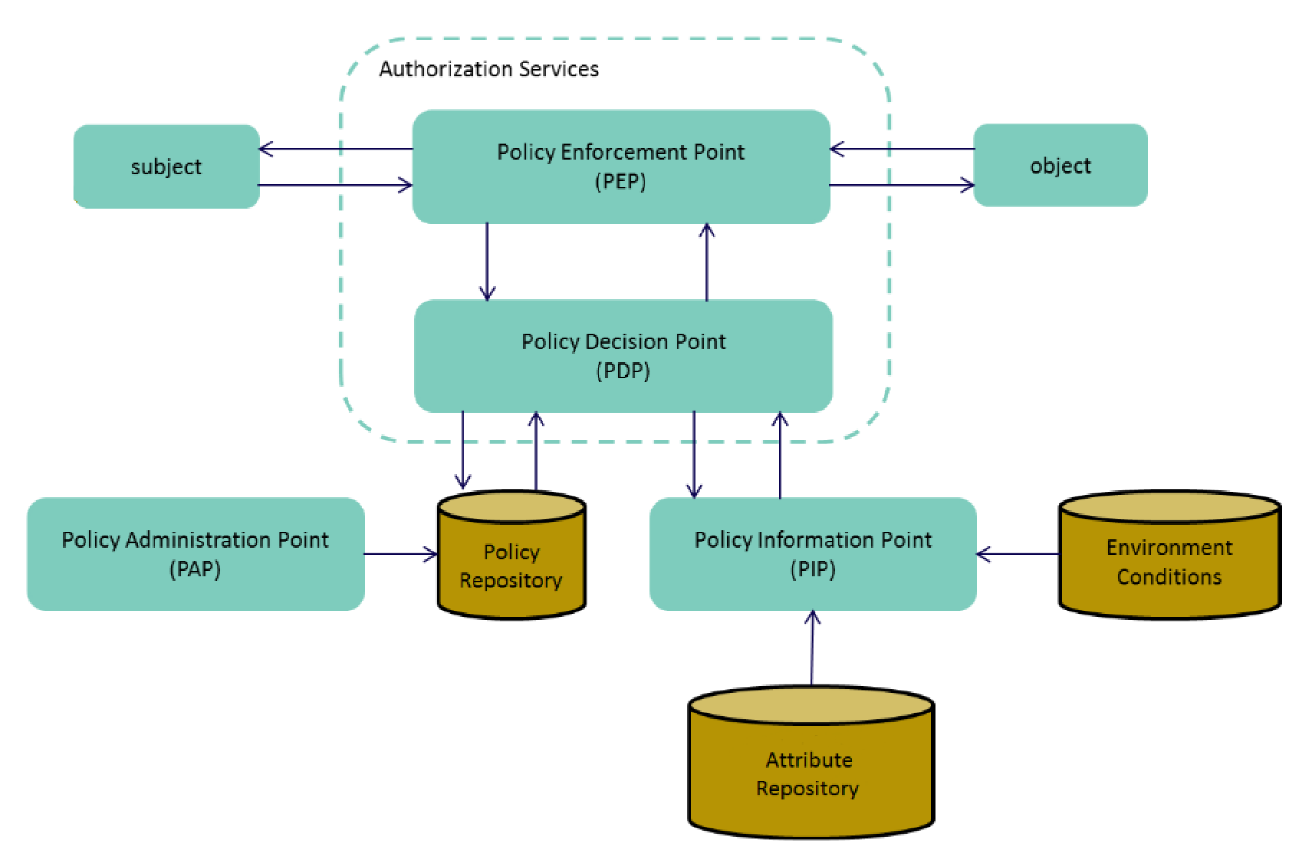
- Extensible Access Control Markup Language (XACML)
- Next Generation Access Control standard [ANSI499]

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


Authorization Services
subject
Policy Enforcement Point (PEP)
object
Policy Decision Point (PDP)
Policy Administration Point (PAP)
Policy Repository
Policy Information Point (PIP)
Environment Conditions
Attribute Repository

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

Drawbacks

- Traction
- Complexity
- Performance

Adding Attributes to Role-Based Access Control

D. Richard Kuhn, *National Institute of Standards and Technology*
Edward J. Coyne, *Science Applications International Corp.*
Timothy R. Weil, *Raytheon Polar Services Company*

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.

IEEE Computer, vol. 43, no. 6 (June, 2010) , pp. 79-81

Enterprise ABAC

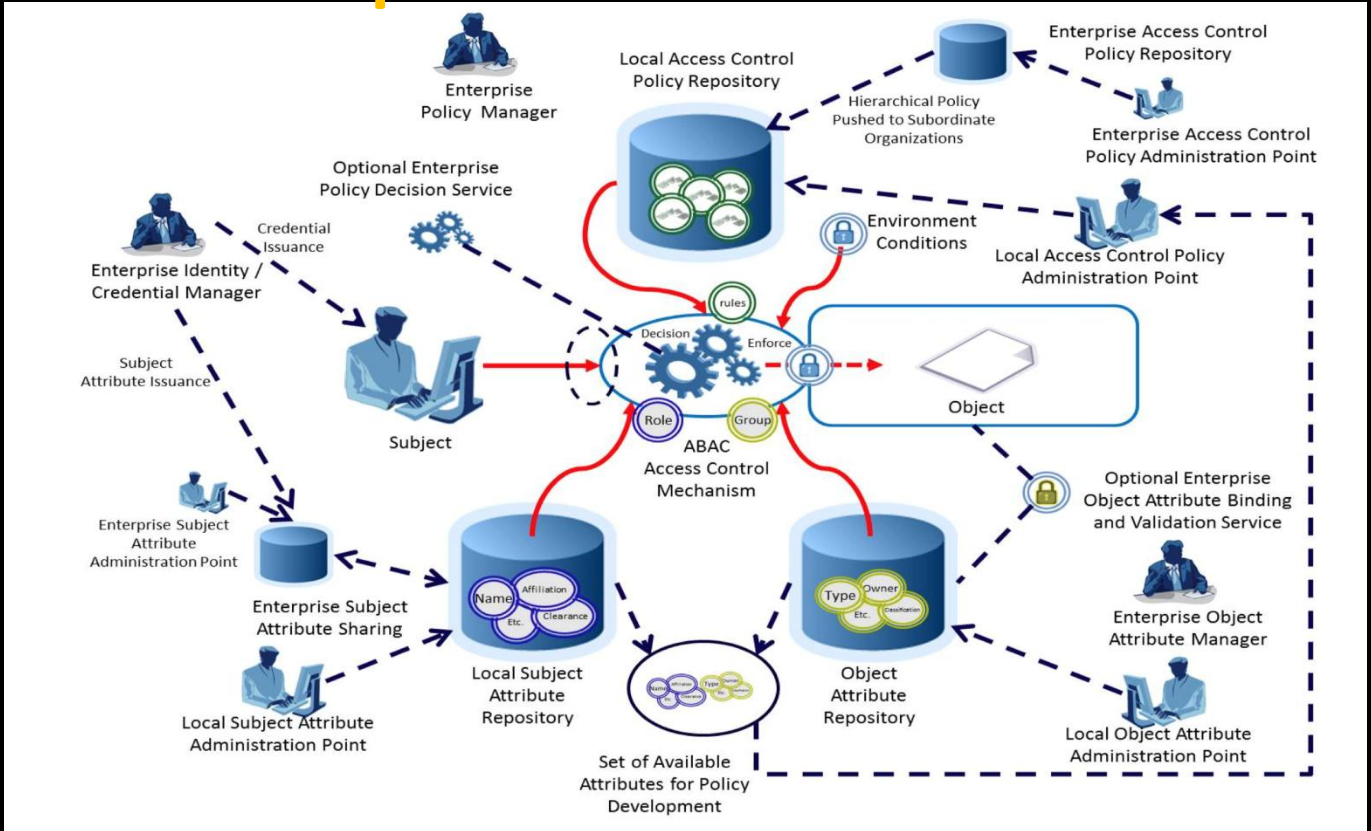

Enterprise Policy Manager
Local Access Control Policy Repository
Enterprise Access Control Policy Repository
Hierarchical Policy Pushed to Subordinate Organizations
Enterprise Access Control Policy Administration Point
Optional Enterprise Policy Decision Service
Credential Issuance
Enterprise Identity / Credential Manager
Environment Conditions
Local Access Control Policy Administration Point
rules
Decision
Enforce
Subject Attribute Issuance
Subject
Object
Role
Group
ABAC Access Control Mechanism
Optional Enterprise Object Attribute Binding and Validation Service
Enterprise Subject Attribute Administration Point
Enterprise Subject Attribute Sharing
Name
Affiliation
Clearance
Etc.
Type
Owner
Classification
Local Subject Attribute Repository
Object Attribute Repository
Enterprise Object Attribute Manager
Local Subject Attribute Administration Point
Local Object Attribute Administration Point
Set of Available Attributes for Policy Development

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

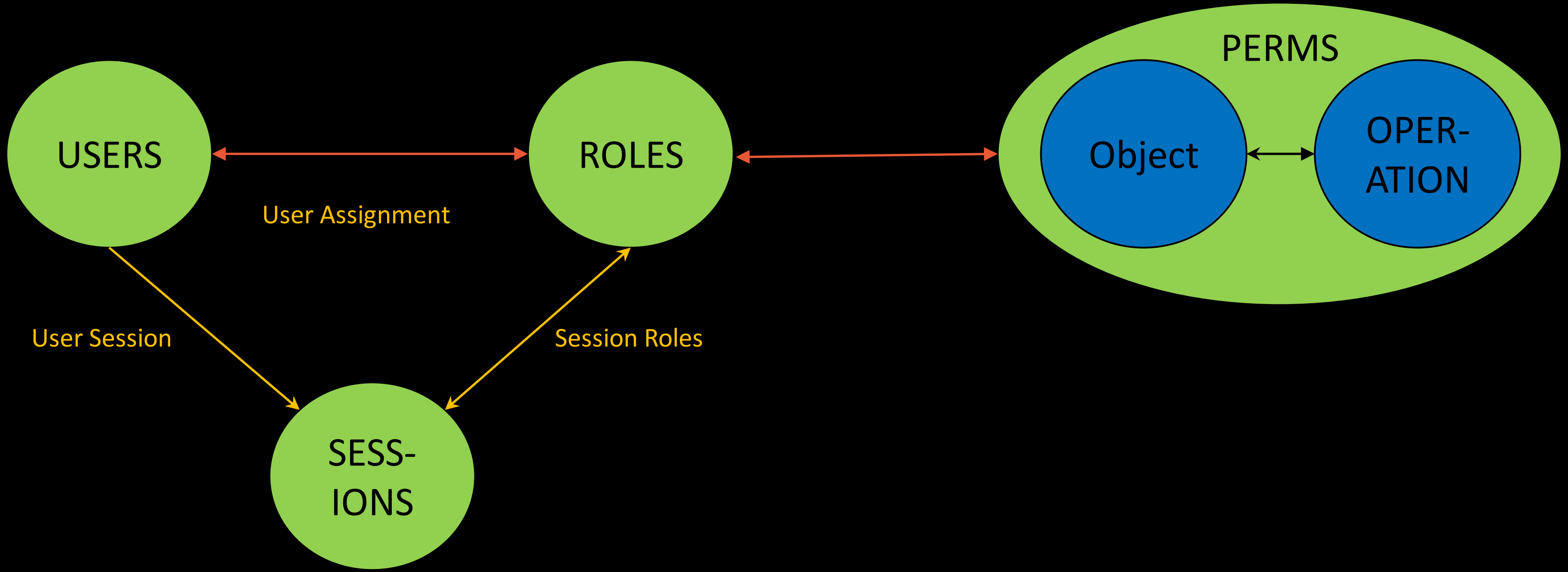
Examples:

- Apache Fortress Temporal Constraints
- Apache Fortress Dynamic Constraints (New)

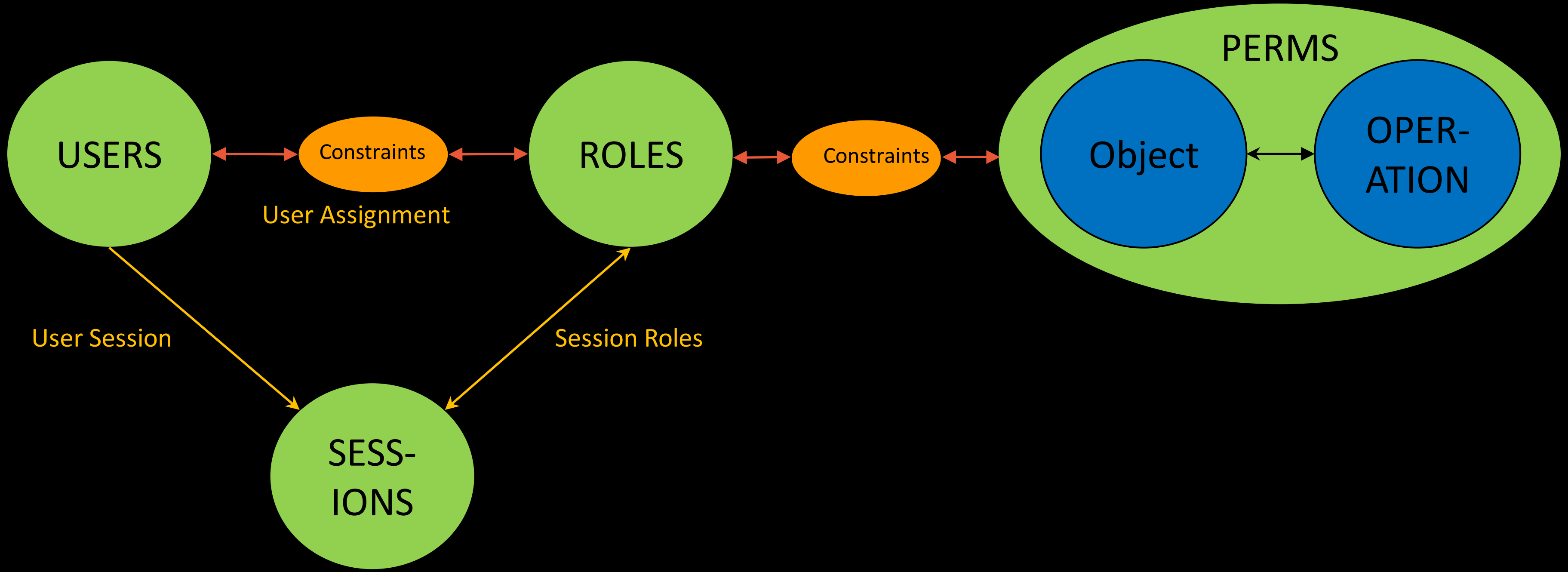
Use User-Role Constraint

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

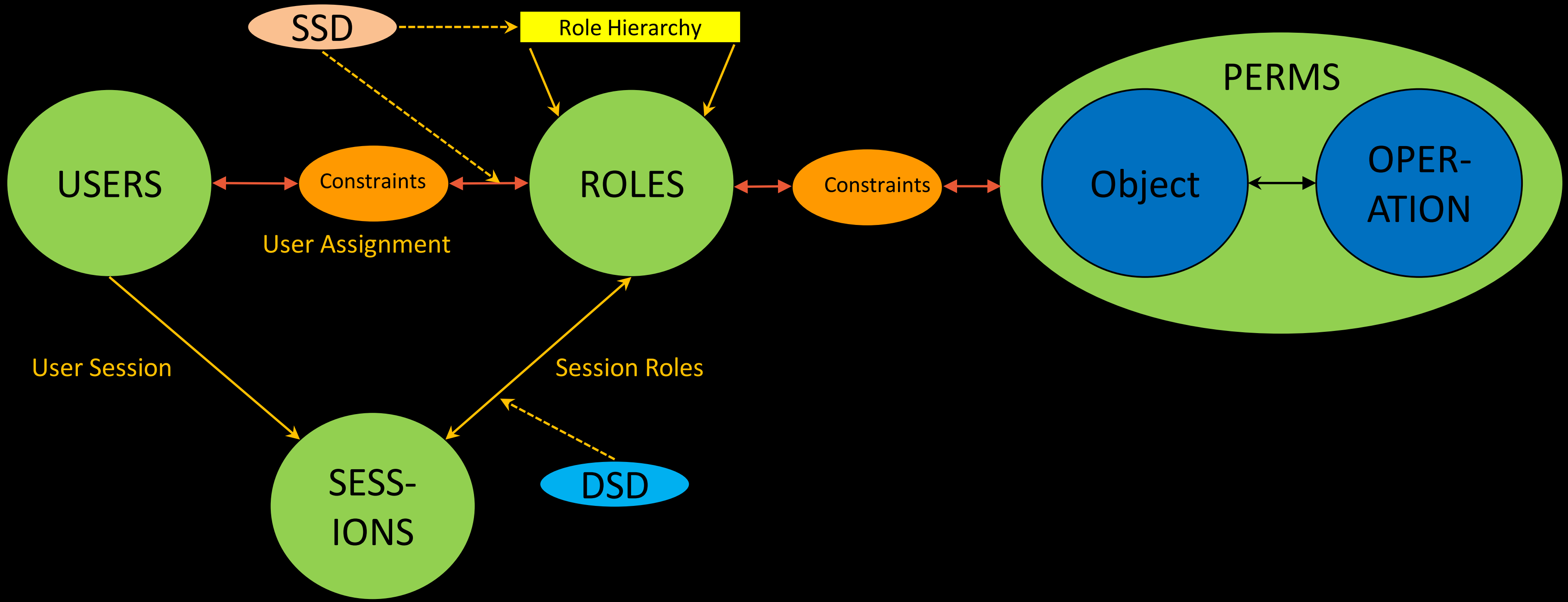
Core RBAC


USERS
ROLES
PERMS
Object
OPER-
ATION
User Assignment
User Session
Session Roles
SESS-
IONS

+ ABAC Constraints



All Together Now



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.

e.g.

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"
```

https://github.com/shawnmckinney/fortress-abac-demo/blob/master/src/main/resources/fortress-abac-demo-load-policy.xml

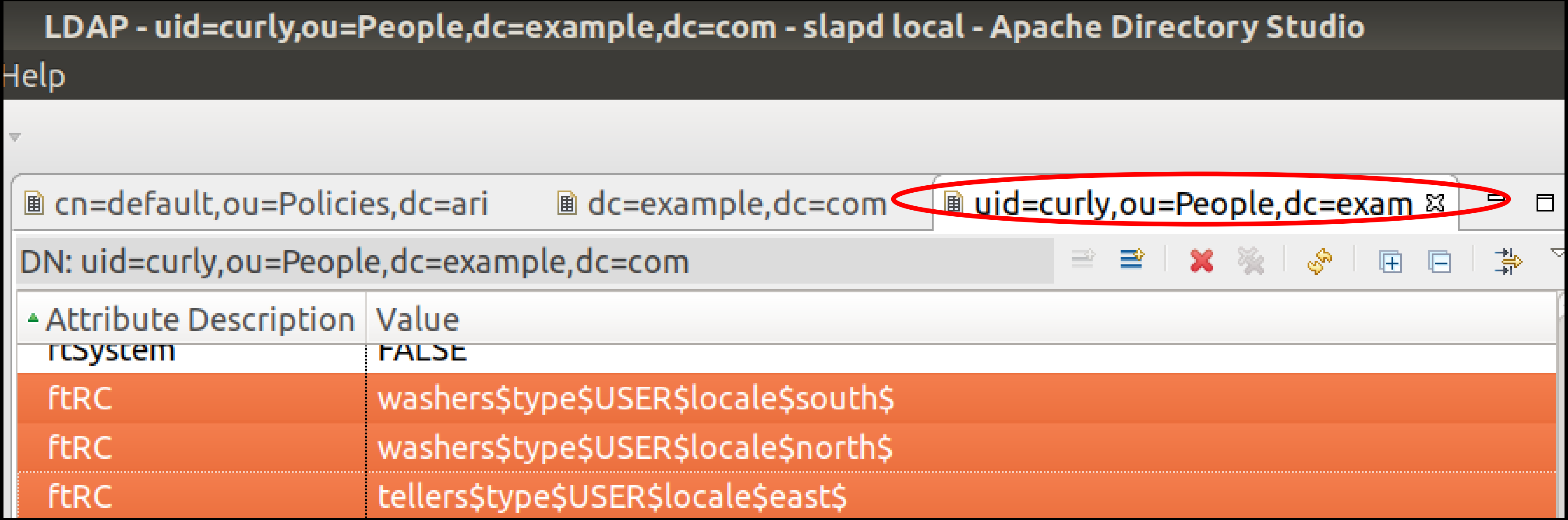
User-Role Constraints

```
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

https://appdevcloudworkshop.github.io/images/introduction/image16.png

RBAC w/ ABAC



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

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

Attribute Description	Value
ftSystem	FALSE
ftRC	washers$type$USER$locale$south$
ftRC	washers$type$USER$locale$north$
ftRC	tellers$type$USER$locale$east$

Code Sample

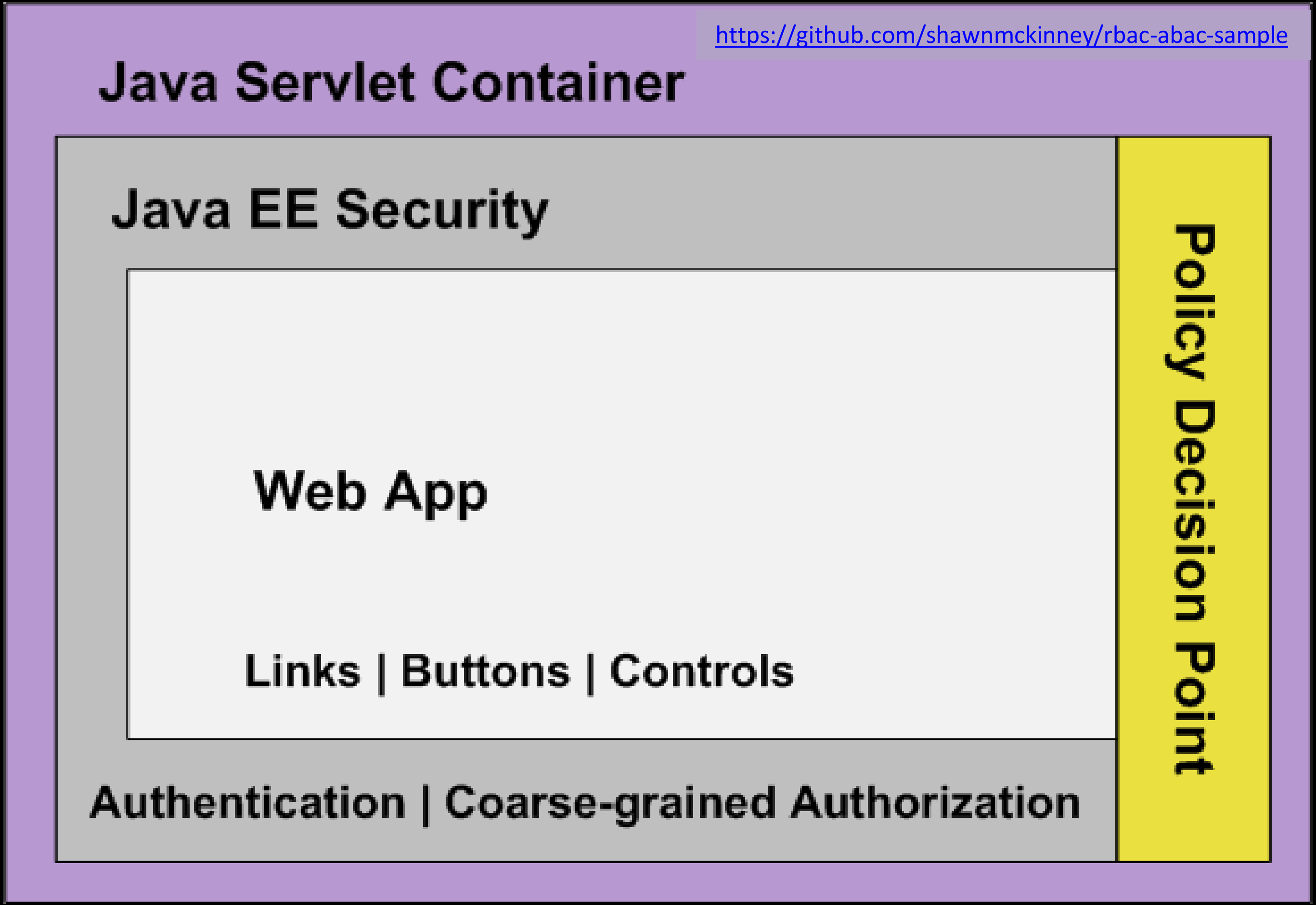
```
// Nothing new here:
User user = new User(“curly”);

// 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 );

try
{
   // Create the RBAC session with ABAC constraint -- location=north, asserted:
   Session session = accessMgr.createSession( user, constraints );
   ...
}
```

https://github.com/shawnmckinney/fortress-abac-demo/blob/master/src/main/java/com/mycompany/MyBasePage.java

ABAC Demo

```
_FAULT_IN_NONPAGED_AREA

his is the first time you've seen this Stop error screen,
art your computer. If this screen appears again, follow
e steps:

k 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   r disable      nents, restart
 computer, press F8 to select Adv     Startup      ons,  d then
ct Safe Mode.

nical information:

STOP: 0x00000050 (0xFD3094C2,0x00000                   000)


 SPCMDCON.SYS - Address FBFE7617 base at F         eStamp 3d6d
```

Example 2 RBAC ABAC Sample



https://github.com/shawnmckinney/rbac-abac-sample

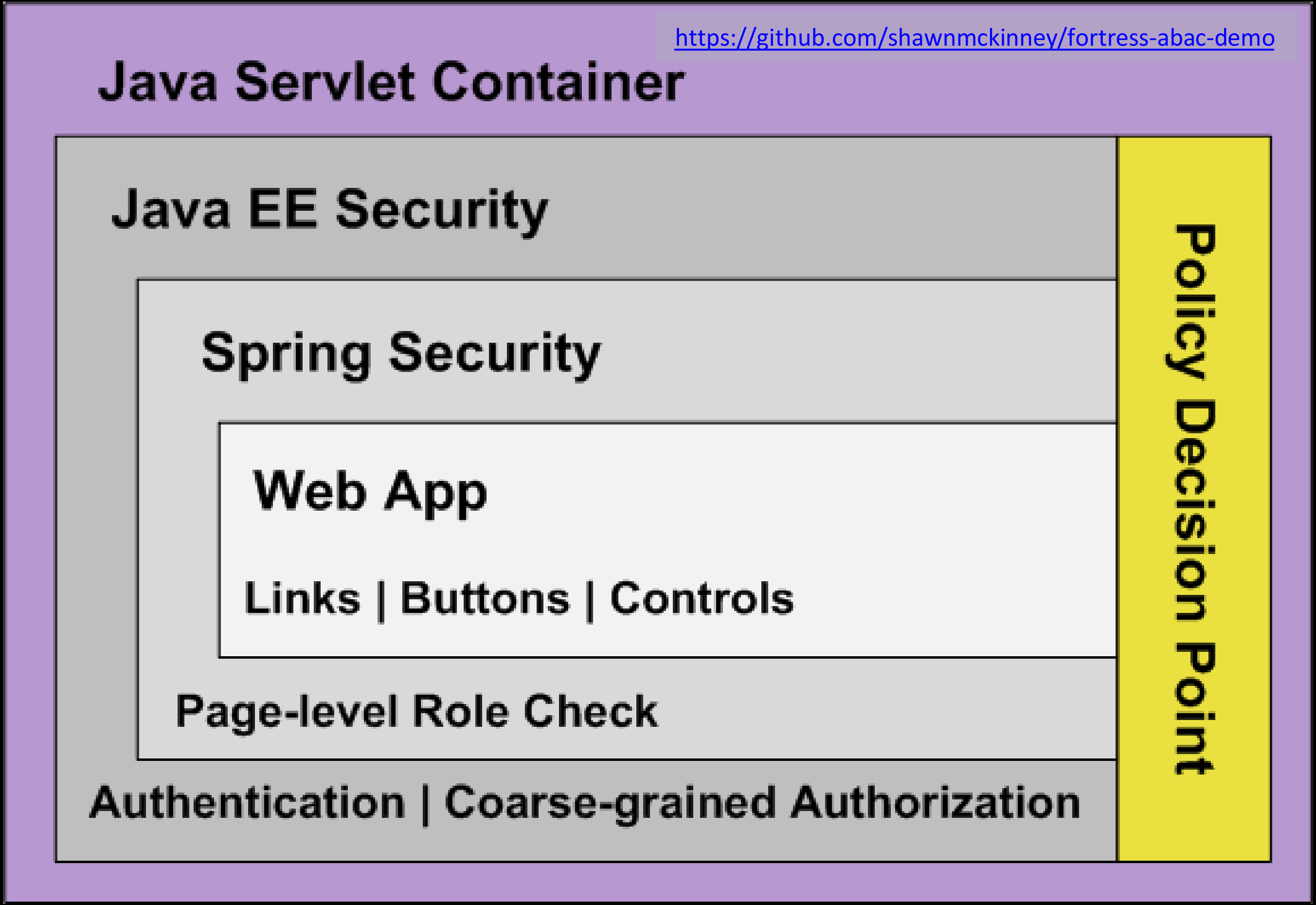
Java Servlet Container
Java EE Security
Web App
Links | Buttons | Controls
Authentication | Coarse-grained Authorization
Policy Decision Point

User456	Customer 123	Customer 456	Customer 789
Page1	False	True	False
Page2	False	True	False
Page3	False	True	False

User2	Customer 123	Customer 456	Customer 789
Page1	False	False	False
Page2	True	True	True
Page3	False	False	False

User2_123	Customer 123	Customer 456	Customer 789
Page1	False	True	False
Page2	False	False	False
Page3	False	False	False

Example 3 Apache Fortress ABAC Demo


https://github.com/shawnmckinney/fortress-abac-demo
Java Servlet Container
Java EE Security
Spring Security
Web App
Links | Buttons | Controls
Page-level Role Check
Authentication | Coarse-grained Authorization
Policy Decision Point

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